BlueBerry Yield Prediction

1.Introduction

1.1Project Overview

BlueBerries are small, round, and typically blue or purple fruits that are enjoyed worldwide for their delicious taste and numerous health benefits. Popular fruit crop that are grown in many regions around the world, including North America, Europe, and Asia. These berries are known for their unique flavor and high antioxidant content, and they are used in a wide range of food products, including jams, juices, and baked goods.

Blueberry farming is a delicate balance of science and art. Predicting blueberry yield is essential for effective farm management, ensuring that growers can optimize their resources and maximize their profits. Here, we delve into the factors influencing blueberry yield and the modern techniques used to predict it accurately. Predicting blueberry yield is a multi-faceted approach combining traditional knowledge with cutting-edge technology. By understanding and leveraging these factors, blueberry farmers can improve their yield predictions, optimize their farming practices, and ultimately enhance their harvests.

1.2 Project Objective

Blueberry yield prediction is a critical aspect of modern agriculture, helping farmers and stakeholders make informed decisions to optimize production, manage resources efficiently, and maximize profits. Here's a comprehensive overview of blueberry yield prediction: Blueberry yield prediction involves using various data sources and analytical methods to forecast the quantity of blueberries that will be harvested in a given season. Accurate yield prediction can significantly enhance the efficiency of blueberry farming by enabling better planning and resource allocation.

2. Project Initialization and Planning Phase

The "Project Initialization and Planning Phase" marks the project's outset, defining goals, scope, and stakeholders. This crucial phase establishes project parameters, identifies key team members, allocates resources, and outlines a realistic timeline. It also involves risk assessment and mitigation planning. Successful initiation sets the foundation for a well-organized and efficiently executed machine learning project, ensuring clarity, alignment, and proactive measures for potential challenges.

Activity 1: Define Problem Statement

Problem Statement: A blueberry farmer, utilizing traditional farming methods, seeks to accurately predict the annual yield of blueberries. The challenge lies in understanding and effectively utilizing climatic factors such as temperature, rainfall, and pollination conditions, which significantly impact blueberry yield. Despite the farmer's optimism about achieving high yields, uncertainty persists regarding the accurate estimation of these factors' influence on crop productivity.

Ref. template: **Click Here**
Blueberry yield prediction Problem Statement Report: Click Here

Activity 2: Project Proposal (Proposed Solution)

The proposed project aims to leverage advanced analytics and machine learning techniques to enhance the accuracy of blueberry yield predictions. By analyzing a comprehensive dataset encompassing climatic factors, pollinating conditions, and historical yield data, the project seeks to develop a predictive model that optimizes agricultural practices and decision-making in blueberry cultivation. Develop and deploy a machine learning model to predict blueberry yield based on the dataset's variables.

Ref. template: Click Here
Blueberry yield prediction Project Proposal Report: Click Here

Activity 3: Initial Project Planning

Initial Project Planning involves outlining key objectives, defining scope, and identifying the yield prediction. It encompasses setting timelines, allocating resources, and determining the overall project strategy. During this phase, the team establishes a clear understanding of the dataset, formulates goals for analysis, and plans the workflow for data processing. Effective initial planning lays the foundation for a systematic and well-executed project, ensuring successful outcomes.

Ref. template: Click Here
Blueberry yield prediction Initial Project Planning Report: Click Here

3. Data Collection and Preprocessing Phase

The Data Collection and Preprocessing Phase involves executing a plan to gather relevant BlueBerry Yield prediction data from Kaggle, ensuring data quality through verification and addressing missing values. Preprocessing tasks include cleaning, encoding, and organizing the dataset for subsequent exploratory analysis and machine learning model development.

Activity 1: Data Collection Plan, Raw Data Sources Identified

The dataset for "Blueberry Yield Prediction" is sourced from Kaggle, a reputable platform known for its diverse collection of datasets in agricultural sciences and predictive analytics.This dataset is meticulously curated to encompass a wide array of variables essential for accurate blueberry yield prediction. These variables include climatic factors,Pollinating factors, and historical yield data. This comprehensive dataset provides a robust foundation for developing predictive models.

Ref. template: Click Here
Blueberry yield prediction Raw Data Sources Report: Click Here

Activity 2: Data Quality Report

The dataset for " Blueberry Yield Prediction" is sourced from Kaggle. It includes climatic factors,Pollinating factors and historical yield data. . Data quality is ensured through verification, addressing missing values and handing Outliers, establishing a reliable foundation for predictive modeling.

Ref. template: Click Here
Blueberry yield prediction Data Quality Report: Click Here

Activity 3: Data Exploration and Preprocessing

Data Exploration involves analyzing the loan applicant dataset to understand patterns, distributions, and outliers. Preprocessing includes handling missing values, scaling, and encodingcategorical variables. These crucial steps enhance data quality, ensuring the reliability and effectiveness of subsequent analyses in the loan approval project.

Ref. template: Click Here
Blueberry yield prediction Data Exploration and Preprocessing Report: Click Here

4. Model Development Phase

The Model Development Phase entails crafting a predictive model for loan approval. It encompasses strategic feature selection, evaluating and selecting models (Linear Regession,Random Forest, Decision Tree, XGB), initiating training with code, and rigorously validating and assessing model performance for informed decision-making in the lending process.

Activity 1: Feature Selection Report

The Feature Selection Report outlines the rationale behind choosing specific features (e.g.,honeybees,MaxOfUpeerTRange,RainingDays…) for the Yield prediction model. It evaluates relevance, importance, and impact on predictive accuracy, ensuring the inclusion of key factors influencing the model's ability to predict the yield.

Ref. template: Click Here
Blueberry yield predict Feature Selection Report: Click Here

Activity 2:Model Selection Report

The Model Selection Report details the rationale behind choosing Linear Regression, Random Forest, Decision Tree, and XGB models for loan approval prediction. It considers each model's strengths in handling complex relationships, interpretability, adaptability, and overall predictive performance, ensuring an informed choice aligned with project objectives.

Ref. template: Click Here
Blueberry yield Model Selection Report: Click Here

Activity 3: Initial Model Training Code, Model Validation and Evaluation Report

The Initial Model Training Code employs selected algorithms on the loan approval dataset, setting the foundation for predictive modeling. The subsequent Model Validation and Evaluation Report rigorously assesses model performance, employing metrics like MAE,MSE,R-Squared and accuracy to ensure reliability and effectiveness in predicting loan outcomes.

Ref. template: Click Here
Blueberry yield Model Development Phase Template: Click Here

5. Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Activity 1: Hyperparameter Tuning Documentation

The XGBoost model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model.

Ref. template: Click Here
Blueberry yield Hyperparameter Tuning Report: Click Here

Activity 2: Performance Metrics Comparison Report

The Performance Metrics Comparison Report contrasts the baseline and optimized metrics for various models, specifically highlighting the enhanced performance of the XGBoost model. This assessment provides a clear understanding of the refined predictive capabilities achieved through hyperparameter tuning.

Ref. template: Click Here
Blueberry yield Performance Metrics comparison Report: Click Here

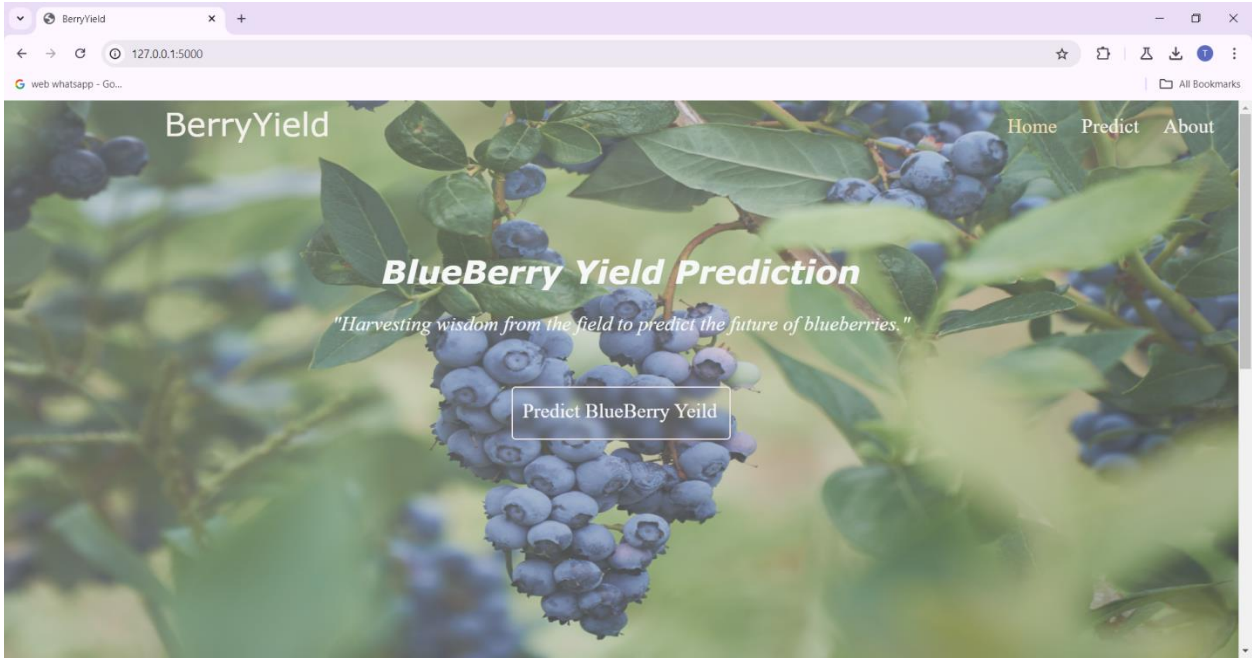
Activity 3: Final Model Selection Justification

The Final Model Selection Justification articulates the rationale for choosing XGBoost as the ultimate model. Its exceptional accuracy, ability to handle complexity, and successful hyperparameter tuning align with project objectives, ensuring optimal Yield predictions.

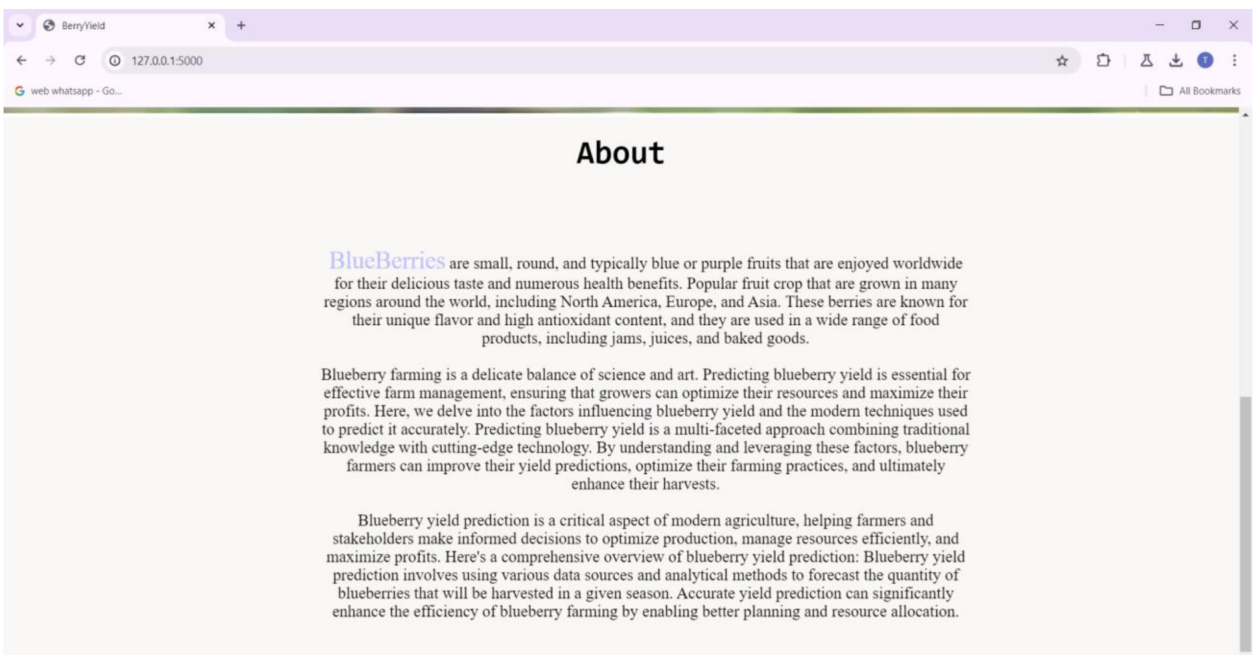
Ref. template: Click Here
Blueberry yield Final Model Selection Justification Report: Click Here

6.RESULT

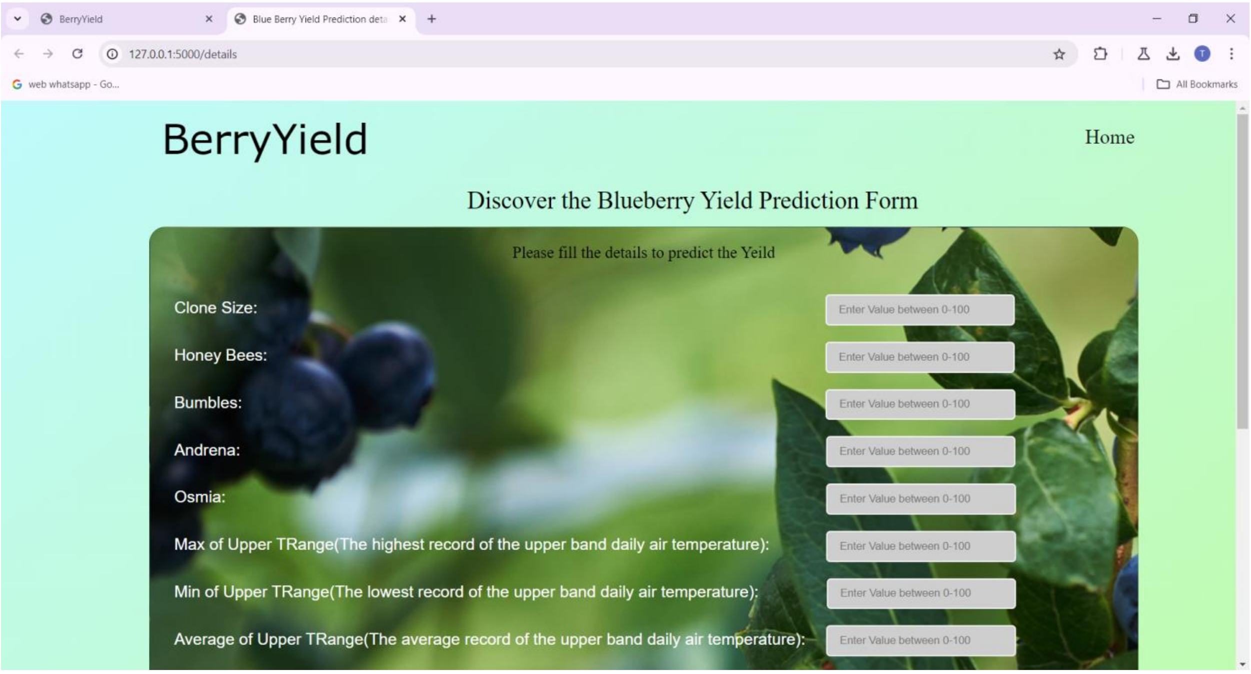
HOME PAGE

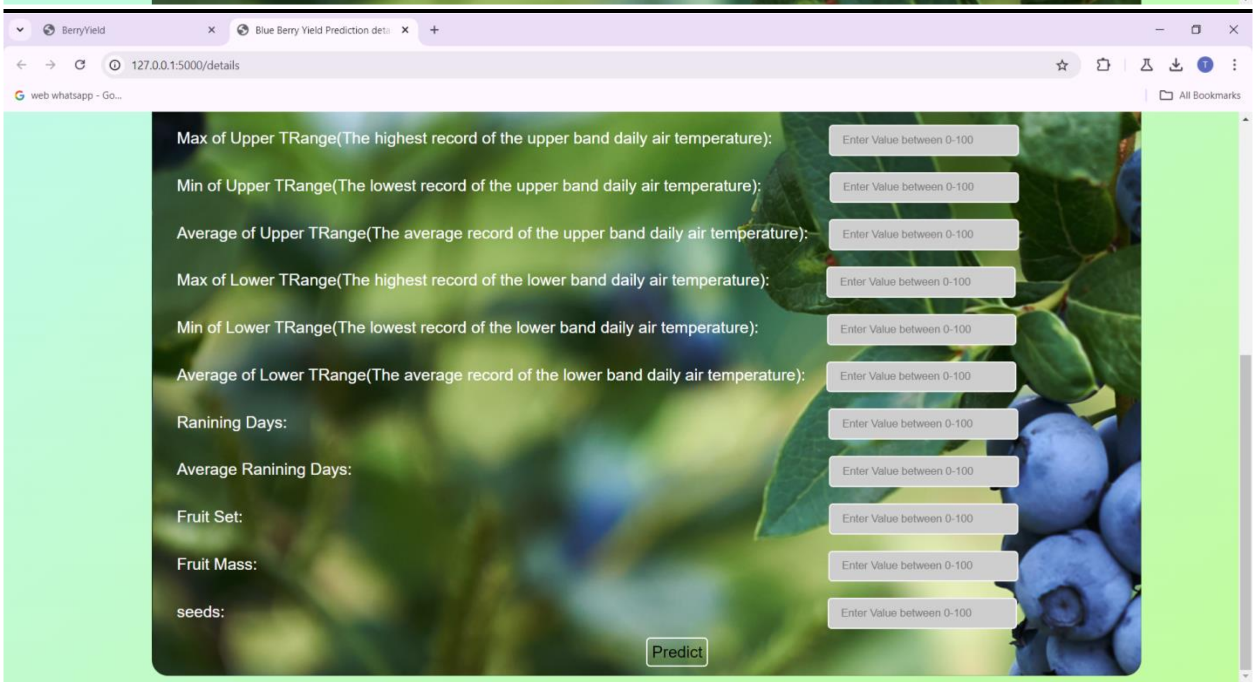


ABOUT PAGE

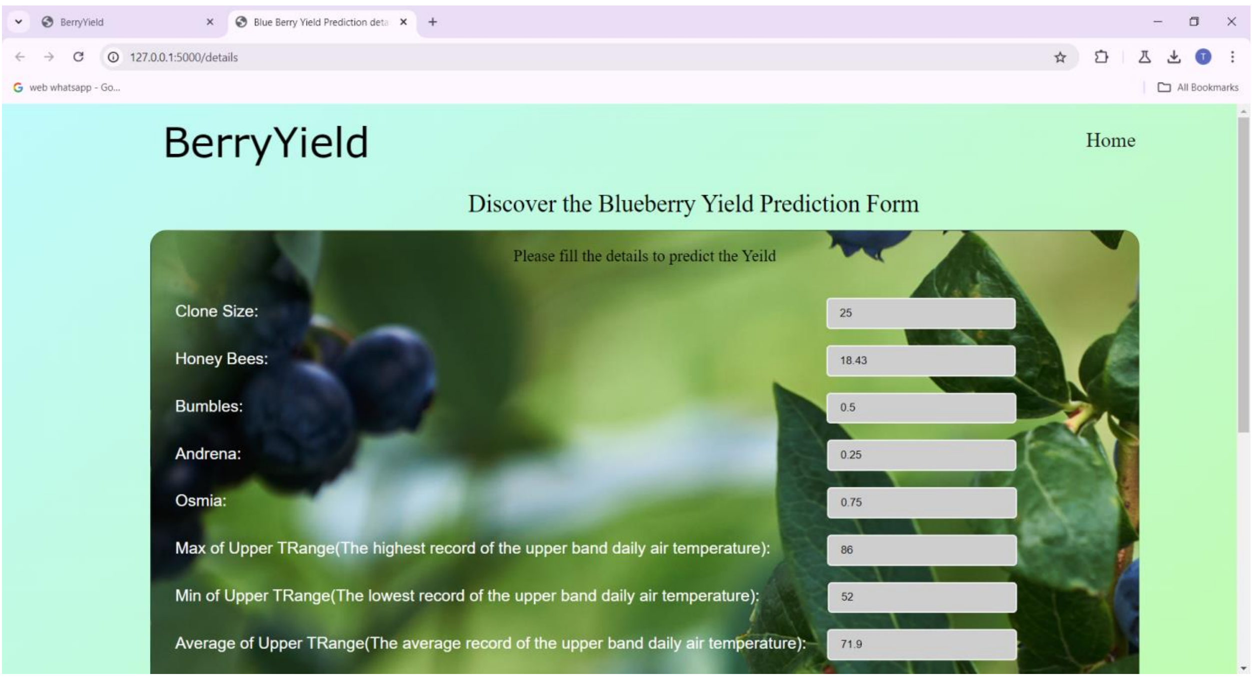


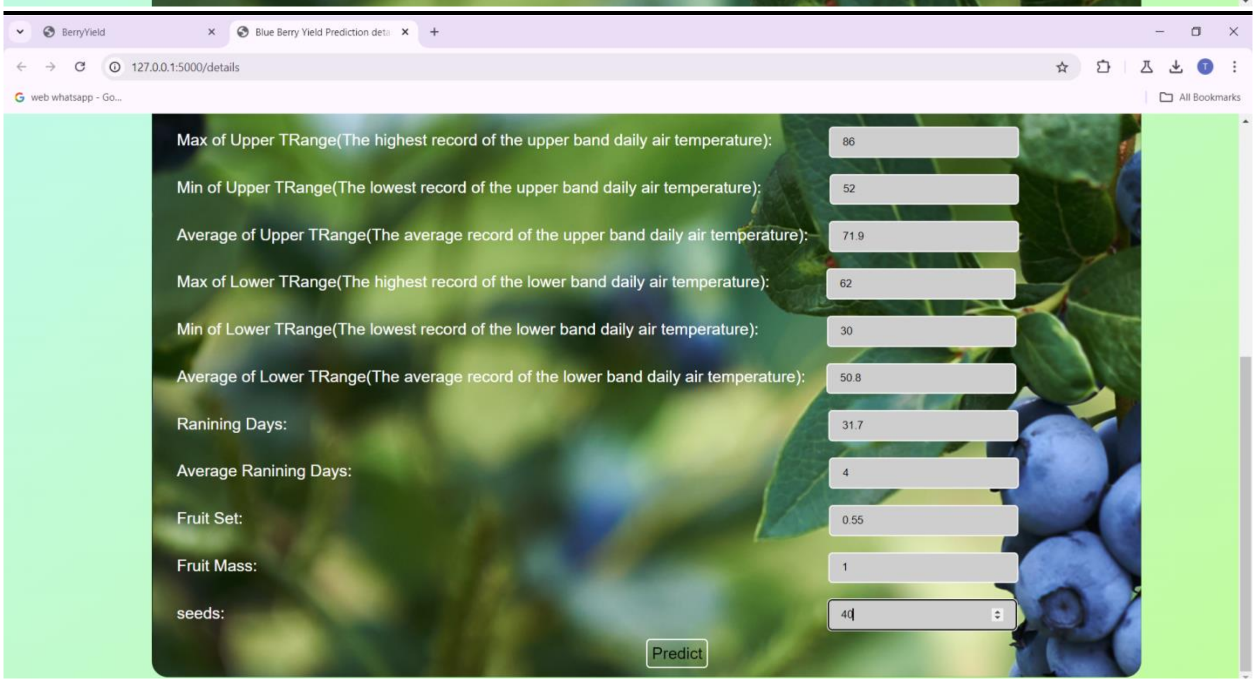
DETAILS PAGE



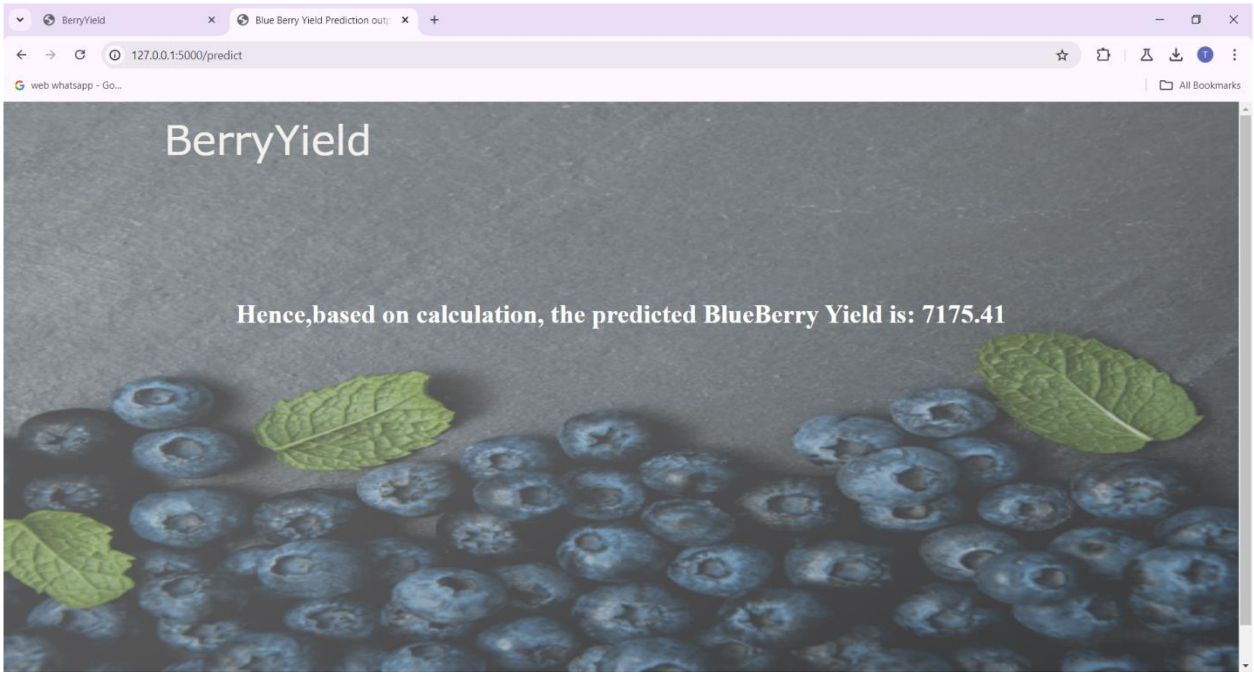


Prediction 1

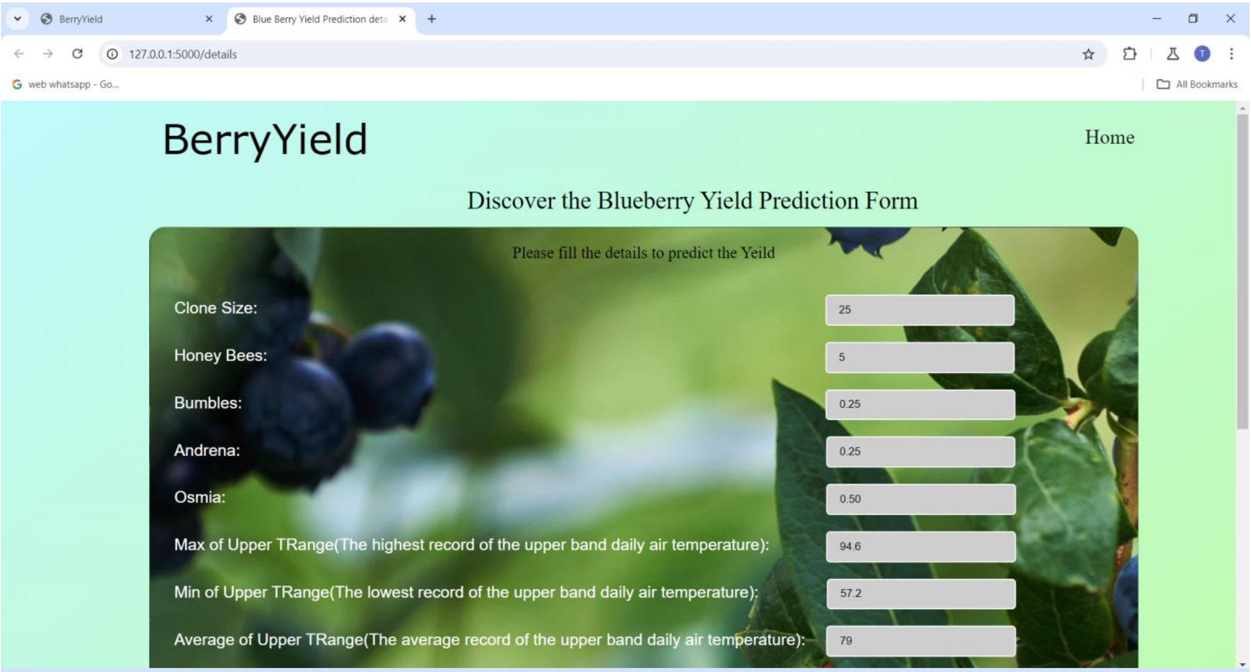


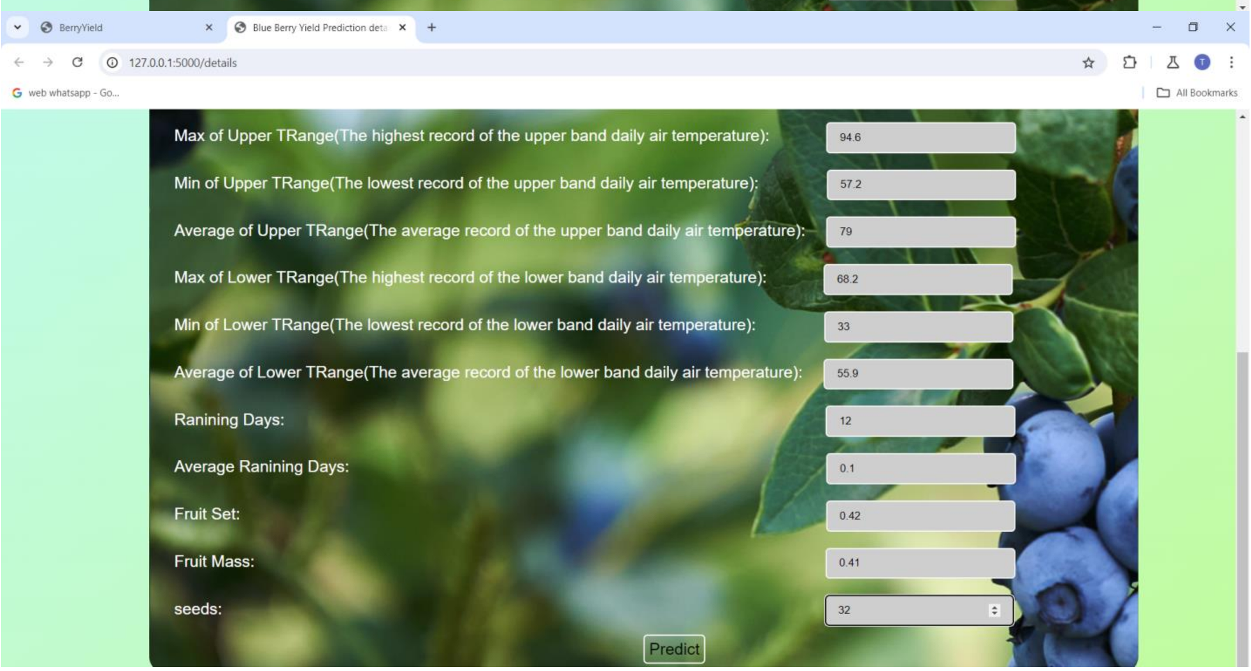


OUTPUT

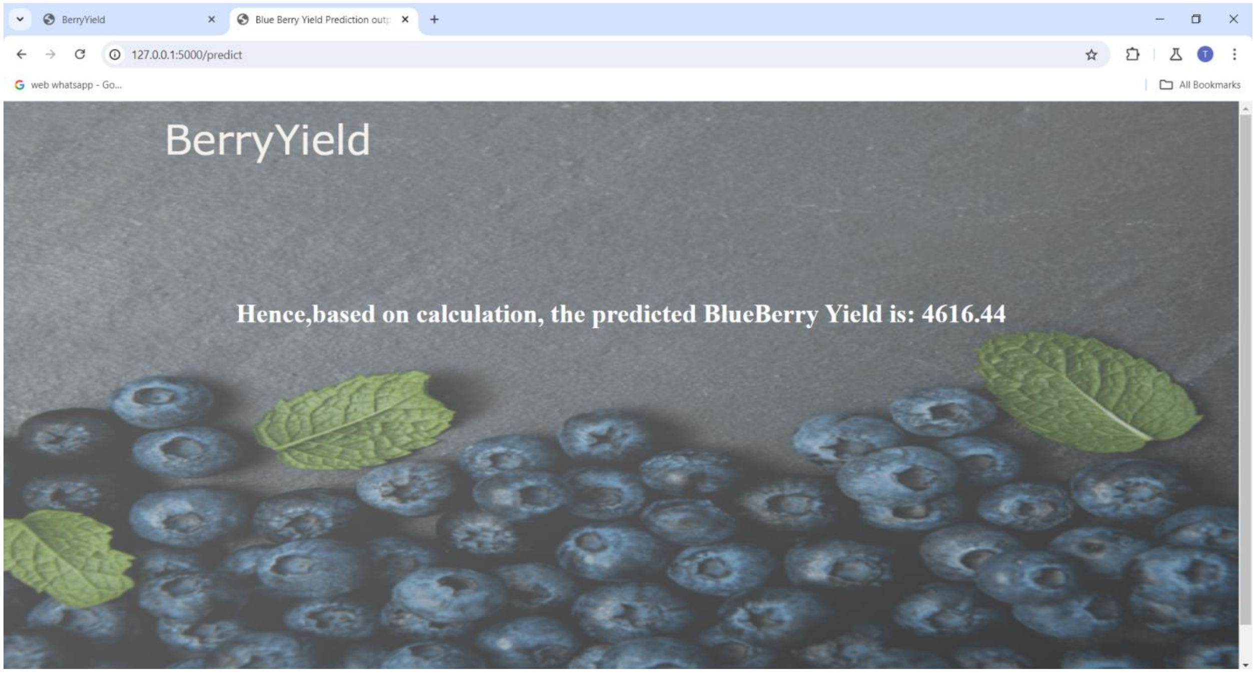

BerryYield
Blue Berry Yield Prediction outp
127.0.0.1:5000/predict
web whatsapp - Go...
All Bookmarks
BerryYield
Hence,based on calculation, the predicted BlueBerry Yield is: 7175.41

Prediction 2





OUTPUT


BerryYield
Blue Berry Yield Prediction outp
127.0.0.1:5000/predict
web whatsapp - Go...
All Bookmarks
BerryYield
Hence,based on calculation, the predicted BlueBerry Yield is: 4616.44

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

1. **Improved Yield Forecasting:** Allows farmers to anticipate and plan for optimal yield based on predictive models.
2. **Resource Optimization:** Helps in efficient allocation of resources such as water, fertilizers, and labor.
3. **Risk Mitigation:** Enables farmers to mitigate risks related to weather fluctuations and climate change impacts
4. **Enhanced Crop Management:** Facilitates better management practices like pest control and disease prevention.
5. **Financial Planning:** Assists in financial planning and budgeting by predicting potential income from harvest.

DISADVANTAGES:

1. **Data Dependence:** Relies heavily on accurate and comprehensive data related to weather, soil conditions, and historical yield records.
2. **Technical Complexity:** Requires technical expertise to develop and interpret predictive models accurately.
3. **Model Uncertainty:** Predictive models may have uncertainties, especially when dealing with complex interactions of environmental factors.
4. **Variable Accuracy:** The accuracy of predictions can vary depending on the quality and relevance of input data and model assumptions.

8.CONCLUSION

► In conclusion, the proposed Blueberry Yield Prediction system offers a comprehensive solution to enhance agricultural productivity and sustainability. By leveraging advanced predictive modeling and data analysis techniques, the system provides farmers with valuable insights into potential yield outcomes. This empowers them to make informed decisions regarding resource allocation, crop management, and market strategies, ultimately optimizing their operations.

► The project's key components, including data collection, preprocessing, feature selection, model training, and deployment, create a robust framework for accurate yield predictions. Additionally, the integration of real-time data and user-friendly interfaces ensures that the system remains practical and accessible for end-users.

► By predicting blueberry yields effectively, this system not only supports individual farmers but also contributes to broader agricultural planning, supply chain management, and economic stability. The holistic approach of the Blueberry Yield Prediction project fosters innovation and sustainability within the agricultural sector, paving the way for a more efficient and productive future.

9.FUTURE SCOPE

1. **Global Expansion**: Extend the prediction model to various regions and countries with different climates and soil types to address blueberry yield issues on a global scale. This will involve collecting and integrating diverse data sets from different geographical areas to enhance the system's accuracy and applicability worldwide.
2. **Advanced Technology Integration**: Incorporate IoT sensor networks and precision agriculture technologies for real-time monitoring of environmental factors such as soil moisture, temperature, and pest activity. This integration will facilitate more accurate and timely predictions, enabling farmers to take proactive measures.
3. **Climate Change Adaptation**: Develop advanced models to predict how climate change impacts blueberry yields over short and long terms. This will help farmers and policymakers create adaptive strategies to mitigate adverse effects and ensure sustainable blueberry production.
4. **Collaboration with Agronomists and Researchers**: Work closely with agricultural experts and researchers to continuously refine the prediction models by incorporating new findings and innovative agricultural practices. This collaboration will help keep the system at the forefront of agricultural technology.
5. **Market and Economic Forecasting**: Expand the system's capabilities to predict market trends and economic impacts related to blueberry yields. This will assist farmers in making informed decisions about planting, harvesting, and selling their crops, ultimately leading to better economic outcomes.
6. **Integration with Supply Chain Management**: Develop features that link yield predictions with supply chain management systems to optimize logistics, reduce waste, and ensure timely delivery of blueberries from farms to markets.
7. **Educational Tools for Farmers**: Create user-friendly educational tools and resources within the system to help farmers understand and leverage prediction insights. This will empower them to implement best practices and maximize their yields.

These future developments will enhance the robustness, accuracy, and usability of the blueberry yield prediction system, ultimately supporting global agricultural sustainability and productivity.

10.APPENDIX

10.1 SOURCE CODE:

Index.HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>BerryYield</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='assets/css/index.css') }}">
   <meta name="viewport" content="width=device-width,initial-scale=1.0">
  </head>
  <body>
      <div id="lg">
       <header id="name">BerryYield</header>
        <nav id="nav">
          <ul>
            <li><a href="#" class="active">Home</a></li>
            <li><a href="/details" target="_blank">Predict</a></li>
            <li><a href="#about">About</a></li>
          </ul>
        </nav>
        <h1 id="title">BlueBerry Yield Prediction</h1>
        <p id="quo">"Harvesting wisdom from the field to predict the future of blueberries."</p>
        <div id="predbut">
           <div id="box">
            <a href="/details" target="_blank">Predict BlueBerry Yeild</a>
          </div>
        </div>
      </div>
      <div id="about">
        <h1 id="head">About</h1>
         <p id="bb"><span id="b">BlueBerries</span> are small, round, and typically blue or
              Purple fruits that are enjoyed worldwide for their delicious taste and numerous health
              benefits.Popular fruit crop that are grown in many regions around the world, including
              North America, Europe, and Asia.  These berries are known for their unique flavor and
              high antioxidant content, and they are used in a wide range of food products,including
              jams, juices, and baked goods.
              <br><br>
             Blueberry farming is a delicate balance of science and art. Predicting blueberry yield is
             essential for effective farm management, ensuring that growers can optimize their
             resources and maximize their profits Here, we delve into the factors influencing
             blueberry yield and the modern techniques used to predict it accurately.Predicting
            blueberry yield is a multi-faceted approach combining traditional knowledge with
            cutting-edge technology. By understanding and leveraging these factors, blueberry
           farmers can improve their yield predictions, optimize their farming practices, and
            ultimately enhance their harvests.
```

```
<br><br>
Blueberry yield prediction is a critical aspect of modern agriculture, helping farmers
and stakeholders make informed decisions to optimize production, manage resources
efficiently, and maximize profits. Here's a comprehensive overview of blueberry yield


prediction:Blueberry yield prediction involves using various data sources and analytical
methods to forecast the quantity of blueberries that will be harvested in a given season.
Accurate yield prediction can significantly enhance the efficiency of blueberry farming
by enabling better planning and resource allocation.
</p>
</div>
</body>
</html>
```

Details.HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Blue Berry Yield Prediction details</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='assets/css/details.css') }}">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
  </head>
  <body>
    <div id="lg">
      <div id="heads">
      <header id="name">BerryYield</header>
      <a href="/" id="home" target="_blank">Home</a>
      </div>
     <header id="head">Discover the Blueberry Yield Prediction Form</header>
       <div id="boxform">
         <form method="POST" action="/predict" id="form" autocomplete="on">
           <p id="fd">Please fill the details to predict the Yeild</p>
           <label for="cs">Clone Size:</label>
           <input type="number" id="cs" name="clonesize" placeholder="Enter Value between
               0-100" step="0.01" required>
           <br>
           <label for="hb">Honey Bees:</label>
           <input type="number" id="hb" name="honeybee" placeholder="Enter Value
             between 0-100" step="0.01" required>
           <br>
           <label for="bum">Bumbles:</label>
           <input type="number" id="bum" name="bumbles" placeholder="Enter Value
             between 0-100" step="0.01" required>
           <br>
           <label for="an">Andrena:</label>
           <input type="number" id="an" name="andrena" placeholder="Enter Value between
             0-100" step="0.01" required>
           <br>
           <label for="os">Osmia:</label>
```

```
<input type="number" id="os" name="osmia" placeholder="Enter Value between 0-
  100" step="0.01" required>
<br>
<label for="maxut">Max of Upper TRange(The highest record of the upper band
  daily air temperature):</label>
<input type="number" id="maxut" name="MaxOfUpperTRange"
     placeholder="Enter Value between 0-100" step="0.01" required>
<br>
<label for="minut">Min of Upper TRange(The lowest record of the upper band daily
     air temperature):</label>
<input type="number" id="minut" name="MinOfUpperTRange" placeholder="Enter
        Value between 0-100" step="0.01" required>
<br>
<label for="avgut">Average of Upper TRange(The average record of the upper
        band daily air temperature):</label>
<input type="number" id="avgut" name="AverageOfUpperTRange"
     placeholder="Enter Value between 0-100" step="0.01" required>
<br>
<label for="maxlt">Max of Lower TRange(The highest record of the lower band
    daily air temperature):</label>
<input type="number" id="maxlt" name="MaxOfLowerTRange"
     placeholder="Enter Value between 0-100" step="0.01" required>
<br>
<label for="minlt">Min of Lower TRange(The lowest record of the lower band
      daily air temperature):</label>
<input type="number" id="minlt" name="MinOfLowerTRange" placeholder="Enter
        Value between 0-100" step="0.01" required>
<br>
<label for="avglt">Average of Lower TRange(The average record of the lower band
    daily air temperature):</label>
<input type="number" id="avglt" name="AverageOfLowerTRange"
      placeholder="Enter Value between 0-100" step="0.01" required>
<br>
<label for="rd">Ranining Days:</label>
<input type="number" id="rd" name="RainingDays" placeholder="Enter Value
        between 0-100" step="0.01" required>
<br>
<label for="avgrd">Average Ranining Days:</label>
<input type="number" id="avgrd" name="AverageRainingDays"
     placeholder="Enter Value between 0-100" step="0.01" required>
<br>
<label for="fs">Fruit Set:</label>
<input type="number" id="fs" name="fruitset" placeholder="Enter Value between
     0-100" step="0.01" required>
<br>
<label for="fm">Fruit Mass:</label>
```

```
                  <input type="number" id="fm" name="fruitmass" placeholder="Enter Value
                     between 0-100" step="0.01" required>
                 <br>
                 <label for="seeds">seeds:</label>
                 <input type="number" id="seeds" name="seeds" placeholder="Enter Value between
                    0-100" step="0.01" required>
                 <br>

               <button type="submit" id="predbut">Predict</button>
              </form>

            </div>
        </div>
    </body>
</html>
```

Predict.HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Blue Berry Yield Prediction output</title>
        <link rel="stylesheet" href="{{ url_for('static', filename='assets/css/predict.css') }}">
        <meta name="viewport" content="width=device-width,initial-scale=1.0">
    </head>
    <body>
        <div id="lig">
          <header id="name">BerryYield</header>
          <h1 id="output">{{ prediction_text }}</h1>
        </div>
    </body>
```

App.py

```
import pickle
from flask import Flask,render_template,request
import pandas as pd
import numpy as np
import xgboost

model = pickle.load(open('bbyp.pkl','rb'))
app=Flask(__name__)

@app.route('/', methods=["GET"])
def home():
    return render_template('index.html')

@app.route('/details', methods=["GET"])
def show_form():
    return render_template('details.html')

@app.route('/predict',methods=["POST","GET"])
def predict():
    input_features = [float(x) for x in request.form.values()]
    features_values = [np.array(input_features) ]
    print(features_values)

    col = ['clonesize','honeybee','bumbles','andrena','osmia','MaxOfUpperTRange',
         'MinOfUpperTRange','AverageOfUpperTRange''MaxOfLowerTRange','MinOfLowerTRange',
        'AverageOfLowerTRange','RainingDays', 'AverageRainingDays','fruitset','fruitmass','seeds' ]


    df = pd.DataFrame(features_values, columns= col)

    prediction = model.predict(df)
    print(prediction[0])
    rounded_value = round(prediction[0], 2)
    text="Hence,based on calculation, the predicted BlueBerry Yield is: "

    return render_template('predict.html', prediction_text=text + str(rounded_value))




if __name__ == "__main__":
    app.run(debug=False,port= 5000)
```

CODE SNIPPETS

DATA COLLECTION

Importing necessary libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest, mutual_info_regression
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
import xgboost
from xgboost import XGBRegressor


from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score


import warnings
warnings.filterwarnings("ignore")
```

Reading Dataset

```
data=pd.read_csv("WildBlueberryPollinationSimulationData.csv")
data
```

	Row#	clonesize	honeybee	bumbles	andrena	osmia	MaxOfUpperTRange	MinOfUpperTRange	AverageOfUpperTRange	MaxOfLowerTRange	MinOfLowerTRange
0	0	37.5	0.750	0.250	0.250	0.250	86.0	52.0	71.9	62.0	30.0
1	1	37.5	0.750	0.250	0.250	0.250	86.0	52.0	71.9	62.0	30.0
2	2	37.5	0.750	0.250	0.250	0.250	94.6	57.2	79.0	68.2	33.0
3	3	37.5	0.750	0.250	0.250	0.250	94.6	57.2	79.0	68.2	33.0

Dataset shape

```
data.shape
```

```
(777, 18)
```

DATA PREPROCESSING

Datatypes

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 777 entries, 0 to 776
Data columns (total 18 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Row#                  777 non-null    int64  
 1   clonesize             777 non-null    float64
 2   honeybee              777 non-null    float64
 3   bumbles               777 non-null    float64
 4   andrena               777 non-null    float64
 5   osmia                 777 non-null    float64
 6   MaxOfUpperTRange      777 non-null    float64
 7   MinOfUpperTRange      777 non-null    float64
 8   AverageOfUpperTRange  777 non-null    float64
 9   MaxOfLowerTRange      777 non-null    float64
 10  MinOfLowerTRange      777 non-null    float64
 11  AverageOfLowerTRange  777 non-null    float64
 12  RainingDays           777 non-null    float64
 13  AverageRainingDays    777 non-null    float64
 14  fruitset              777 non-null    float64
 15  fruitmass             777 non-null    float64
 16  seeds                 777 non-null    float64
 17  yield                 777 non-null    float64
dtypes: float64(17), int64(1)
memory usage: 109.4 KB
```

Handling null values

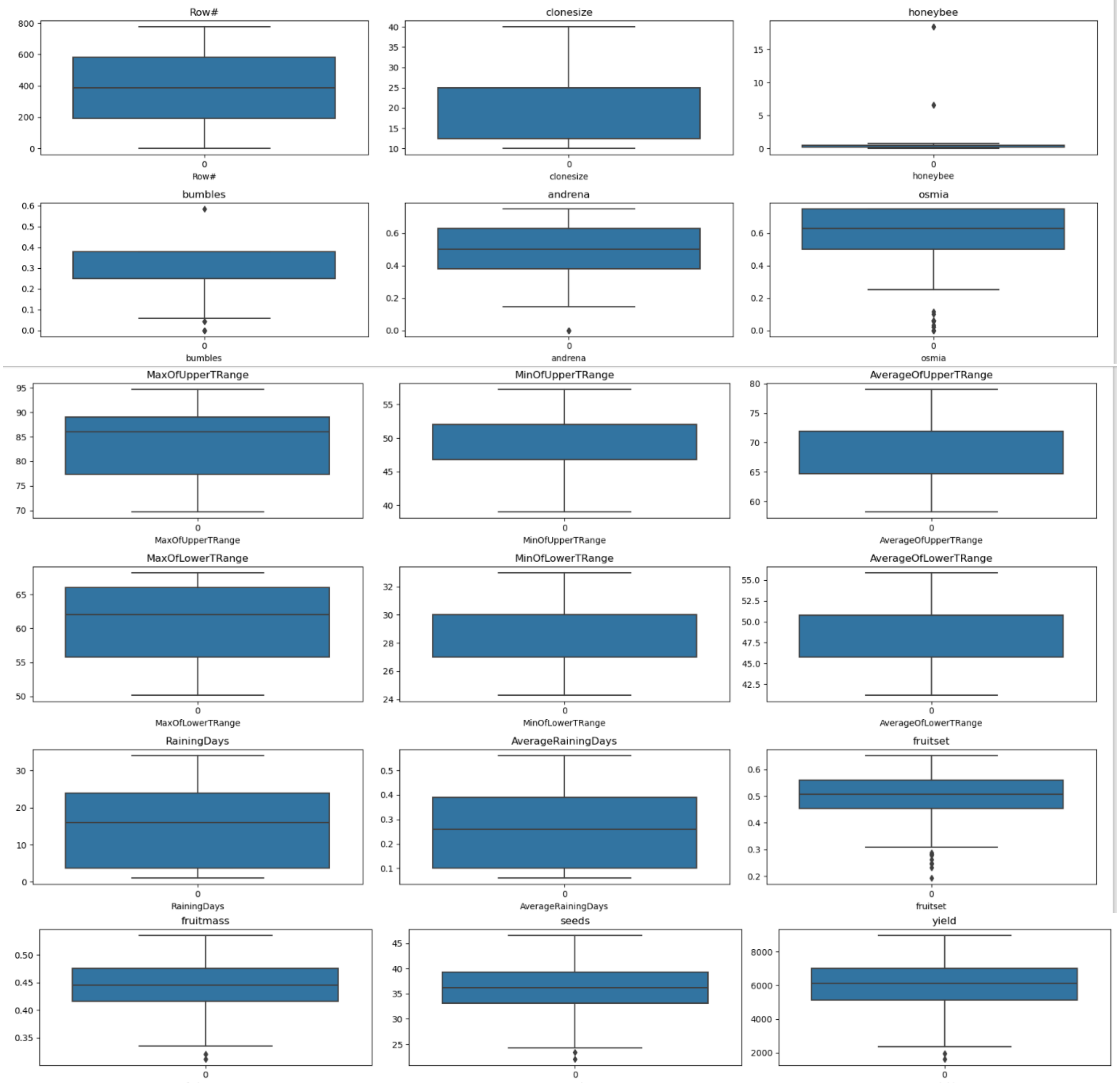
[8]:
```
data.isnull().sum()
```

[8]:
```
Row#                    0
clonesize               0
honeybee                0
bumbles                 0
andrena                 0
osmia                   0
MaxOfUpperTRange        0
MinOfUpperTRange        0
AverageOfUpperTRange    0
MaxOfLowerTRange        0
MinOfLowerTRange        0
AverageOfLowerTRange    0
RainingDays             0
AverageRainingDays      0
fruitset                0
fruitmass               0
seeds                   0
yield                   0
dtype: int64
```

veiwing imbalanced data

using boxpot

```
plt.figure(figsize=(18,18))
for i,col in enumerate (data.columns):
    plt.subplot(6,3,i+1)
    sns.boxplot(data[col])
    plt.xlabel(col)
    plt.title(col)
plt.tight_layout()
```



handling imbalance data

by removing outliers

[223]:
```
x=data
q1=x.quantile(0.25)
q3=x.quantile(0.75)
iqr=q3-q1
iqr
```

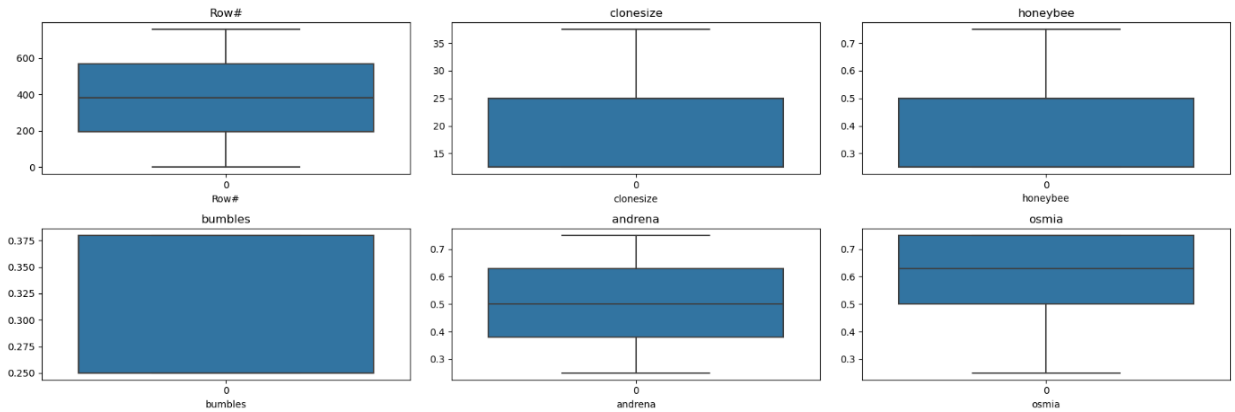
[223]:
```
Row#                      388.000000
clonesize                  12.500000
honeybee                    0.250000
bumbles                     0.130000
andrena                     0.250000
osmia                       0.250000
MaxOfUpperTRange           11.600000
MinOfUpperTRange            5.200000
AverageOfUpperTRange        7.200000
MaxOfLowerTRange           10.200000
MinOfLowerTRange            3.000000
AverageOfLowerTRange        5.000000
RainingDays                20.230000
AverageRainingDays          0.290000
fruitset                    0.106571
fruitmass                   0.059869
seeds                       6.123577
yield                    1897.334830
dtype: float64
```

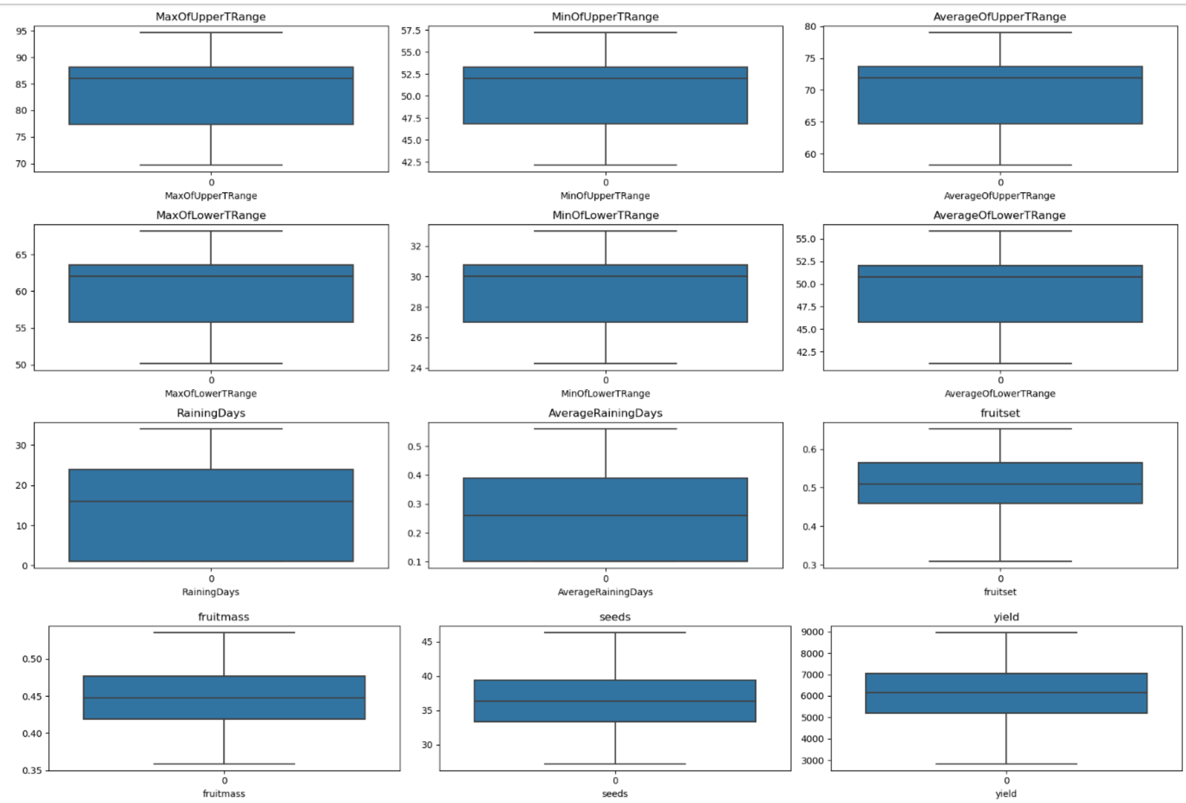
```
p_d=data[~((data<(q1-1.5*iqr)) | (data>(q3+1.5*iqr))).any(axis=1)]
p_d.shape
```

```
(752, 18)
```

Data after removing outliers

```python
plt.figure(figsize=(18,18))
for i,col in enumerate(data.columns):
    plt.subplot(6,3,i + 1)
    sns.boxplot(p_d[col])
    plt.xlabel(col)
    plt.title(col)
plt.tight_layout()
```





Descriptive statistical

```
[14]: p_d.describe()
```

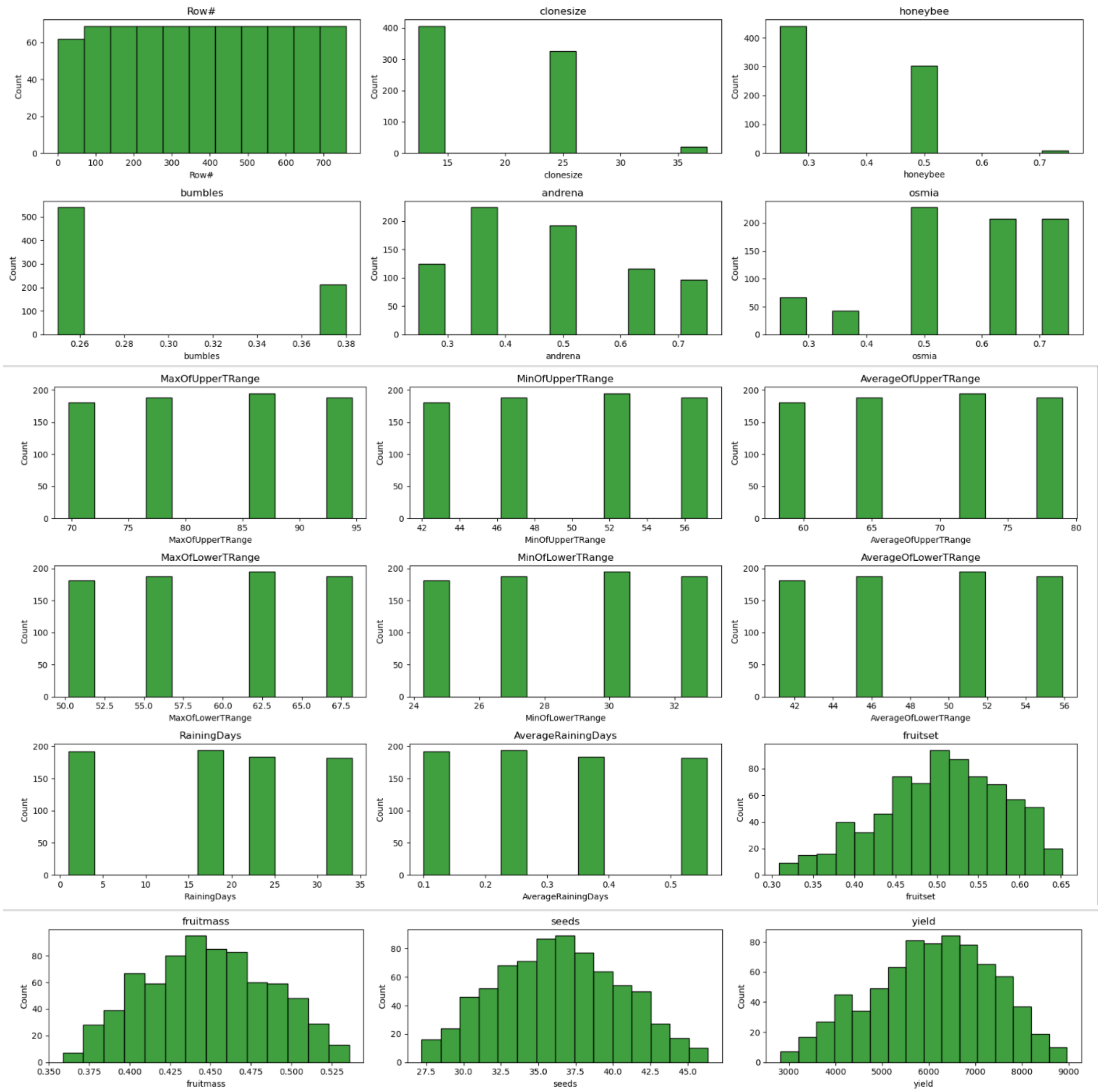
[14]:

	Row#	clonesize	honeybee	bumbles	andrena	osmia	MaxOfUpperTRange	MinOfUpperTRange	AverageOfUpperTRange	MaxOfLowerTRange
count	752.000000	752.000000	752.000000	752.000000	752.000000	752.000000	752.000000	752.000000	752.000000	752.000000
mean	382.337766	18.583777	0.356383	0.286649	0.475000	0.576463	82.076729	49.617154	68.577527	59.159840
std	217.501250	6.885425	0.129602	0.058530	0.156807	0.149782	9.254791	5.610176	7.731659	6.687814
min	0.000000	12.500000	0.250000	0.250000	0.250000	0.250000	69.700000	42.100000	58.200000	50.200000
25%	194.750000	12.500000	0.250000	0.250000	0.380000	0.500000	77.400000	46.800000	64.700000	55.800000
50%	382.500000	12.500000	0.250000	0.250000	0.500000	0.630000	86.000000	52.000000	71.900000	62.000000
75%	570.250000	25.000000	0.500000	0.380000	0.630000	0.750000	88.150000	53.300000	73.675000	63.550000
max	758.000000	37.500000	0.750000	0.380000	0.750000	0.750000	94.600000	57.200000	79.000000	68.200000

VISUAL ANALYSIS

Univariate Analysis(Histplot)

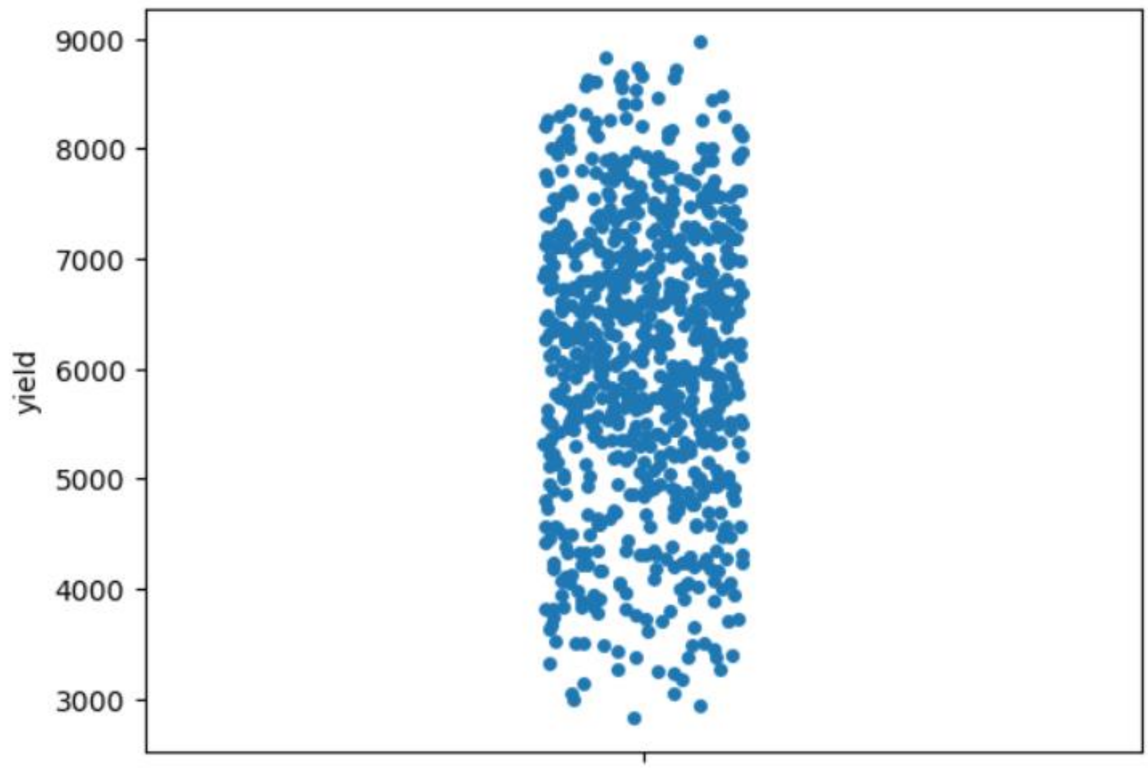
```
plt.figure(figsize=(18,18))
for i,col in enumerate(data.columns):
    plt.subplot(6,3,i+1)
    sns.histplot(p_d[col],color='green')
    plt.xlabel(col)
    plt.title(col)
plt.tight_layout()
```



stirplot

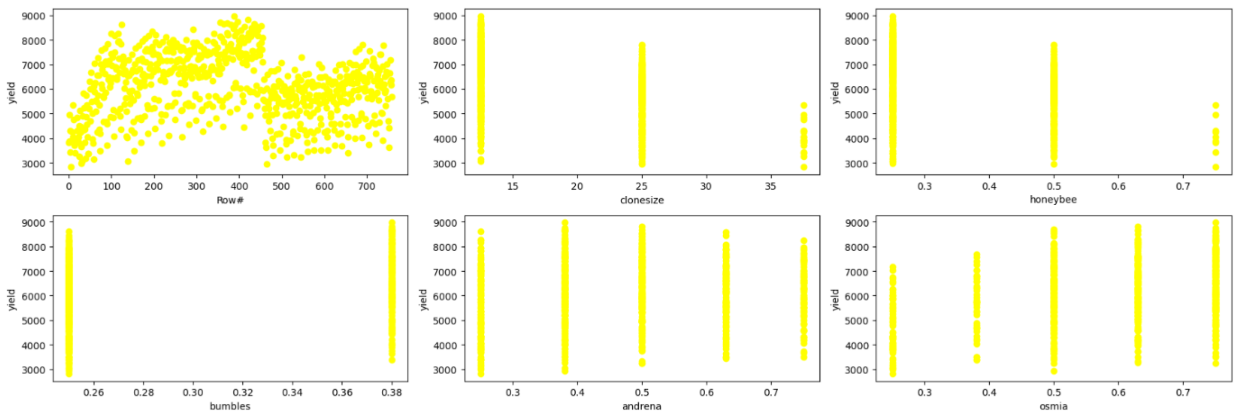
```
sns.stripplot(y=p_d['yield'])
```

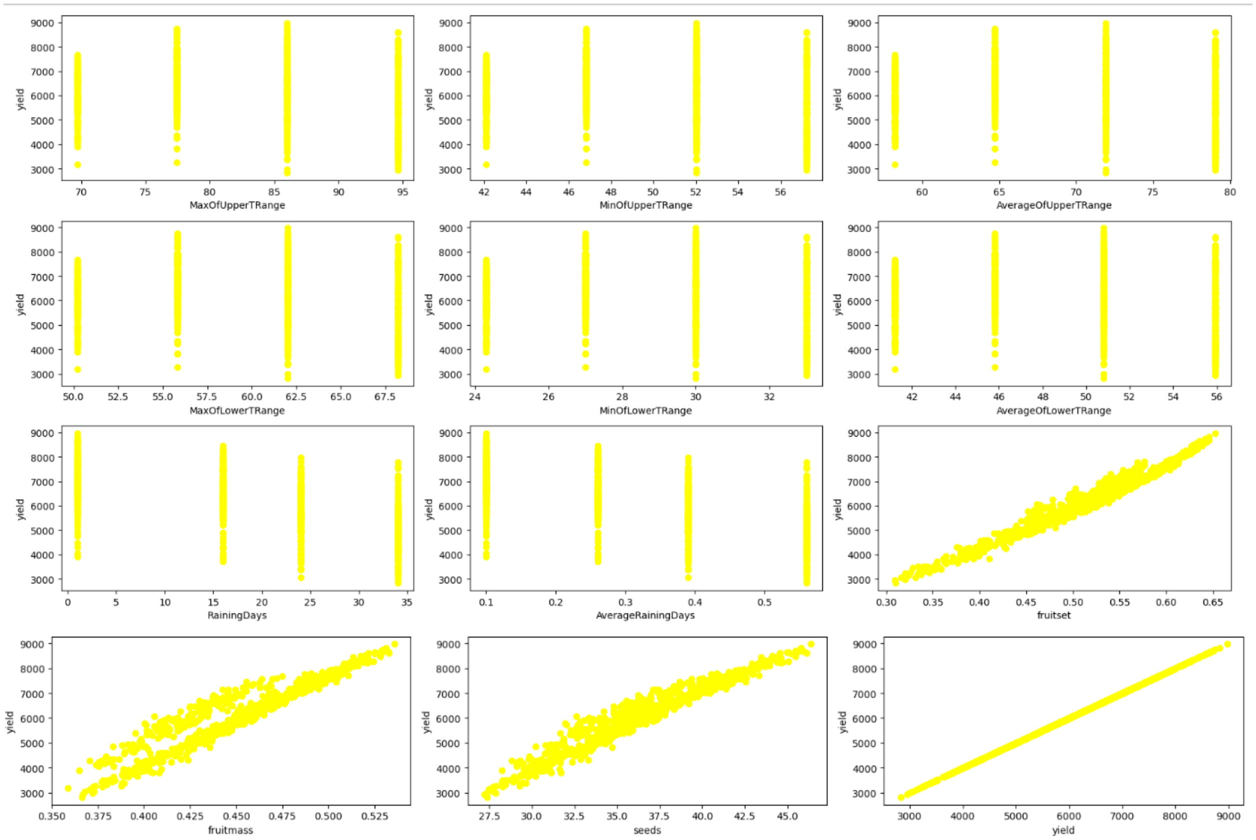
```
<Axes: ylabel='yield'>
```



Bivariate Analysis(Scatterplot)

```
plt.figure(figsize=(18,18))
for i,col in enumerate(data.columns):
    plt.subplot(6,3,i+1)
    plt.scatter(x=p_d[col],y=p_d['yield'],color='yellow')
    plt.xlabel(col)
    plt.ylabel('yield')
plt.tight_layout()
```

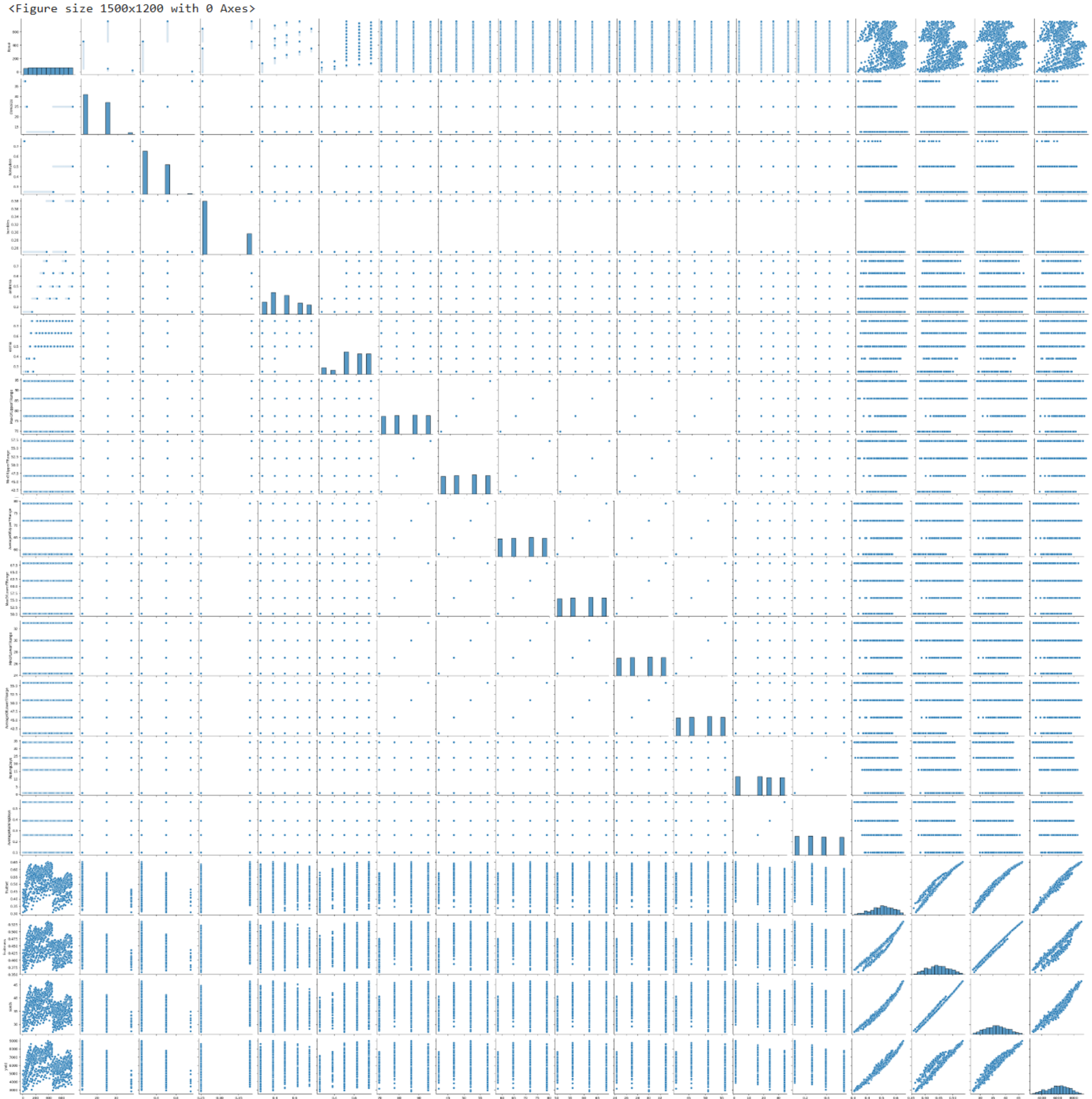




Multivariate Analysis(Pairplot)

```
plt.figure(figsize=(15,12))
sns.pairplot(p_d)
plt.show()
```

```
<Figure size 1500x1200 with 0 Axes>
```



MODEL BULIDING

removing target value from dataset

```
x=p_d.drop(columns=['yield'])
y=p_d[['yield']]
```

splitting data

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

Model Traning & Model Evalution

Linear Regression

```
lr = LinearRegression()
lr.fit(x_train,y_train)
pred_lr=lr.predict(x_test)


mae_lr = mean_absolute_error(y_test,pred_lr)
mse_lr = mean_squared_error(y_test,pred_lr)
rmse_lr = np.sqrt(mse_lr)
rsq_lr = r2_score(y_test,pred_lr)

print("MAE:%.3f" % mae_lr)
print("MSE:%.3f" % mse_lr)
print("RSME:%.3f" % rmse_lr)
print("R-Square:%.3f" % rsq_lr)
print("training accuracy",lr.score(x_train,y_train))
print("testing accuracy",lr.score(x_test,y_test))
```

```
MAE:87.009
MSE:12093.240
RSME:109.969
R-Square:0.993
training accuracy 0.9918401736922372
testing accuracy 0.992515823698853
```

Random Forest Regressor

```
rf=RandomForestRegressor(max_depth=1)
rf.fit(x_train,y_train)
pred_rf=rf.predict(x_test)
pred_rf_train=rf.predict(x_train)

mae_rf_train=mean_absolute_error(y_train,pred_rf_train)
mae_rf = mean_absolute_error(y_test,pred_rf)
mse_rf = mean_squared_error(y_test,pred_rf)
rmse_rf = np.sqrt(mse_rf)
rsq_rf = r2_score(y_test,pred_rf)

print("MAE_train:%.3f" % mae_rf_train)
print("MAE:%.3f" % mae_rf)
print("MSE:%.3f" % mse_rf)
print("RSME:%.3f" % rmse_rf)
print("R-Square:%.3f" % rsq_rf)
print("training accuracy",rf.score(x_train,y_train))
print("testing accuracy",rf.score(x_test,y_test))
```

```
MAE_train:598.746
MAE:596.199
MSE:491642.205
RSME:701.172
R-Square:0.696
training accuracy 0.6922597007723057
testing accuracy 0.6957360469382565
```

Decision Tree Regressor

```
dt=DecisionTreeRegressor()
dt.fit(x_train,y_train)
pred_dt=dt.predict(x_test)

mae_dt = mean_absolute_error(y_test,pred_dt)
mse_dt = mean_squared_error(y_test,pred_dt)
rmse_dt = np.sqrt(mse_dt)
rsq_dt = r2_score(y_test,pred_dt)

print("MAE:%.3f" % mae_dt)
print("MSE:%.3f" % mse_dt)
print("RSME:%.3f" % rmse_dt)
print("R-Sqare:%.3f" % rsq_dt)
print("training accuracy",dt.score(x_train,y_train))
print("testing accuracy",dt.score(x_test,y_test))
```

```
MAE:159.751
MSE:42218.450
RSME:205.471
R-Sqare:0.974
training accuracy 1.0
testing accuracy 0.9738721523374747
```

XGBoost

```
xgb=XGBRegressor()
xgb.fit(x_train,y_train)
pred_xgb=xgb.predict(x_test)

mae_xgb = mean_absolute_error(y_test,pred_xgb)
mse_xgb = mean_squared_error(y_test,pred_xgb)
rmse_xgb = np.sqrt(mse_xgb)
rsq_xgb = r2_score(y_test,pred_xgb)

print("MAE:%.3f" % mae_xgb)
print("MSE:%.3f" % mse_xgb)
print("RSME:%.3f" % rmse_xgb)
print("R-Sqare:%.3f" % rsq_xgb)
print("training accuracy",xgb.score(x_train,y_train))
print("testing accuracy",xgb.score(x_test,y_test))
```

```
MAE:107.235
MSE:19564.571
RSME:139.873
R-Sqare:0.988
training accuracy 0.9999402183903177
testing accuracy 0.9878920205537743
```

Hyperparameter Tuning

Linear Regression

```python
from sklearn.linear_model import Ridge
ridge = Ridge()
parameters = {'alpha': [0.1, 1, 10]}  # Example values for regularization strength

ridge_regressor = GridSearchCV(ridge, parameters, scoring='neg_mean_squared_error', cv=5)
ridge_regressor.fit(x_train, y_train)

best_alpha = ridge_regressor.best_params_['alpha']
print("Best Alpha:", best_alpha)

# Using the best model found by GridSearchCV
best_ridge = ridge_regressor.best_estimator_
best_ridge.fit(x_train, y_train)
pred_ridge = best_ridge.predict(x_test)
```

```python
mae_ridge = mean_absolute_error(y_test, pred_ridge)
mse_ridge = mean_squared_error(y_test, pred_ridge)
rmse_ridge = np.sqrt(mse_ridge)
rsq_ridge = r2_score(y_test, pred_ridge)

print("MAE: %.3f" % mae_ridge)
print("MSE: %.3f" % mse_ridge)
print("RMSE: %.3f" % rmse_ridge)
print("R-Square: %.3f" % rsq_ridge)
print("Training Accuracy:", best_ridge.score(x_train, y_train))
print("Testing Accuracy:", best_ridge.score(x_test, y_test))
```

```
Best Alpha: 0.1
MAE: 95.466
MSE: 14043.502
RMSE: 118.505
R-Square: 0.991
Training Accuracy: 0.991011446378135
Testing Accuracy: 0.9913088598782471
```

XGBoost

```
xgb = XGBRegressor()
param_grid = {
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'min_child_weight': [1, 3, 5],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0]
}

grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid,
                           scoring='neg_mean_squared_error', cv=5, verbose=1)

grid_search.fit(x_train, y_train)

print("Best Parameters:", grid_search.best_params_)
print("Best CV Score:", grid_search.best_score_)

best_xgb = grid_search.best_estimator_

pred_xgb_tuned = best_xgb.predict(x_test)
```

```
mae_xgb_tuned = mean_absolute_error(y_test, pred_xgb_tuned)
mse_xgb_tuned = mean_squared_error(y_test, pred_xgb_tuned)
rmse_xgb_tuned = np.sqrt(mse_xgb_tuned)
rsq_xgb_tuned = r2_score(y_test, pred_xgb_tuned)

print("\nTuned Model Metrics:")
print("MAE: %.3f" % mae_xgb_tuned)
print("MSE: %.3f" % mse_xgb_tuned)
print("RMSE: %.3f" % rmse_xgb_tuned)
print("R-Squared: %.3f" % rsq_xgb_tuned)
print("Training Accuracy:", best_xgb.score(x_train, y_train))
print("Testing Accuracy:", best_xgb.score(x_test, y_test))
```

```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
Best Parameters: {'colsample_bytree': 0.8, 'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1, 'subsample': 0.6}
Best CV Score: -16626.085239377753

Tuned Model Metrics:
MAE: 94.131
MSE: 14517.358
RMSE: 120.488
R-Squared: 0.991
Training Accuracy: 0.9951537856788809
Testing Accuracy: 0.9910156029061967
```

Model Testing

```
lr = LinearRegression()
lr.fit(x_train,y_train)
pred_lr=lr.predict(x_test)

print(lr.predict([[37.5,0.75,0.25,0.25,0.25,86,52,71.9,62,30,50.8,16,0.26,0.410652063,0.408159008,31.67889844]]))
```

```
[[4353.34667969]]
```

```
lr = LinearRegression()
lr.fit(x_train,y_train)
pred_lr=lr.predict(x_test)

print(lr.predict([[25,0.25,0.25,0.25,0.25,94.6,57.2,79,68.2,33,55.9,1,0.2,0.425,0.417,32.460]]))
```

```
[[4436.09667969]]
```

```
lr = LinearRegression()
lr.fit(x_train,y_train)
pred_lr=lr.predict(x_test)

print(lr.predict([[25,18.43,0,0,0,86,52,71.9,62 ,30 ,50.8,3.77,0.06,0.559628479,0.364936839,27.10639138]]))
```

```
[[13291.59667969]]
```

```
xgb = XGBRegressor()
xgb.fit(x_train,y_train)
pred_xgb=xgb.predict(x_test)

print(xgb.predict([[37.5,0.75,0.25,0.25,0.25,86,52,71.9,62,30,50.8,16,0.26,0.410652063,0.408159008,31.67889844]]))
```

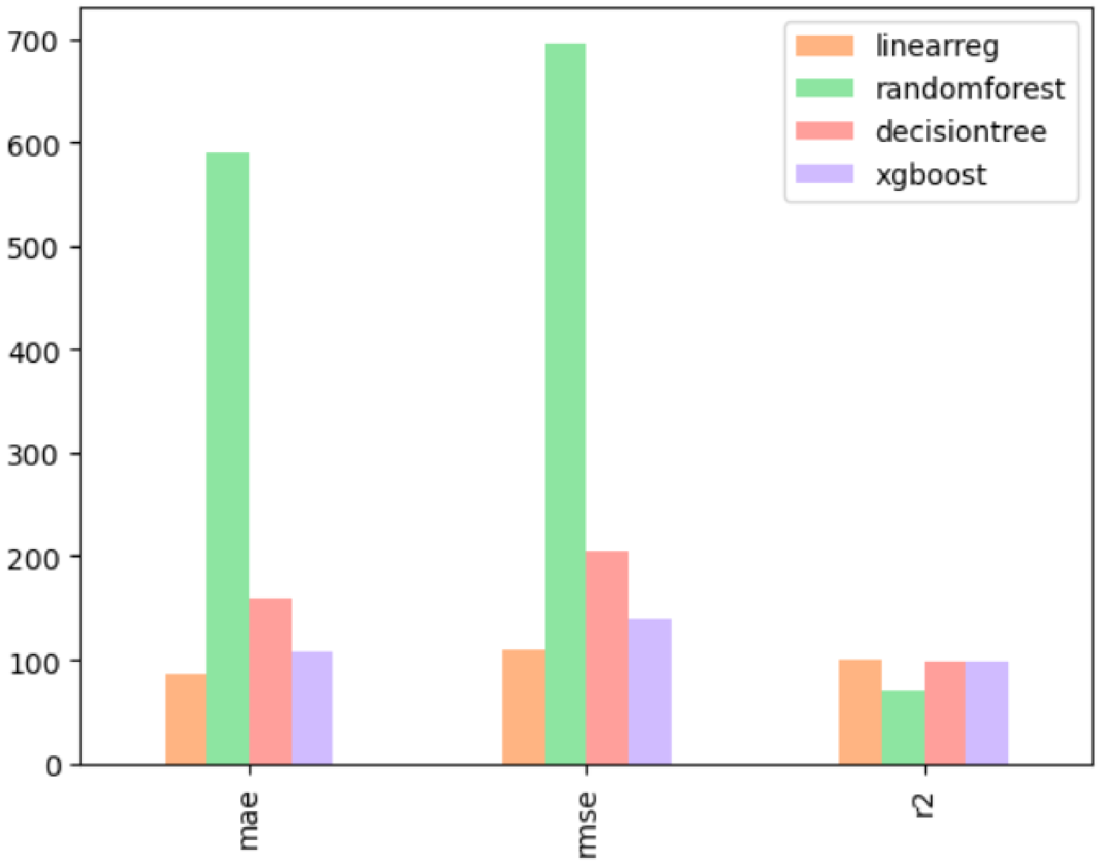
```
[3823.4697]
```

```
best_xgb = XGBRegressor()
best_xgb.fit(x_train,y_train)
pred_xgb1=best_xgb.predict(x_test)

print(best_xgb.predict([[37.5,0.75,0.25,0.25,0.25,86,52,71.9,62,30,50.8,16,0.26,0.41,0.40,31.67]]))
```

Models Comparison

```
error_rec={
    'linearreg':{
        "mae":mae_lr,
        "rmse":rmse_lr,
        "r2":rsq_lr*100
    },
    'randomforest':{
        "mae":mae_rf,
        "rmse":rmse_rf,
        "r2":rsq_rf*100
    },
    'decisiontree':{
        "mae":mae_dt,
        "rmse":rmse_dt,
        "r2":rsq_dt*100
    },
    'xgboost':{
        "mae":mae_xgb,
        "rmse":rmse_xgb,
        "r2":rsq_xgb*100
    },
}
pd.DataFrame(error_rec).plot(kind="bar",
                             color=[
                                 sns.color_palette("pastel")[1],
                                 sns.color_palette("pastel")[2],
                                 sns.color_palette("pastel")[3],
                                 sns.color_palette("pastel")[4]]);
```



Saving the model

```
import pickle
pickle.dump(xgb,open('bbyp.pkl','wb'))
```

10.2 Github & project Demo Link:

Github link : [Click Here]