A Systematic Analysis of Various ML Models for Wild Blueberry Yield Prediction

Project Report:

1) Introduction

1.1. Project Overview

The Blueberry Yield Prediction System represents a significant advancement in agricultural technology aimed at mitigating challenges faced by blueberry farmers. Traditional methods of yield estimation often fall short due to their inability to account for dynamic factors such as fluctuating weather conditions, varying soil health, and unpredictable pest activities. These uncertainties contribute to financial instability among farmers, either through overestimation leading to surplus or underestimation resulting in revenue loss.

In response to these challenges, our project harnesses the power of machine learning to develop a robust prediction system. By integrating historical yield data with real-time environmental variables, the system aims to provide accurate forecasts crucial for optimizing farming operations. This initiative not only seeks to enhance yield prediction accuracy but also aims to empower farmers with actionable insights into crop management practices.

1.2. Objectives

The primary objective of the Blueberry Yield Prediction System is to empower blueberry farmers with reliable forecasting tools. Key objectives include:

- **Accuracy**: Develop machine learning models that accurately predict blueberry yields based on a comprehensive dataset encompassing weather patterns, soil conditions, and pest dynamics.
- **Operational Efficiency**: Enable farmers to make informed decisions regarding harvesting schedules, resource allocation, and market strategies, thereby optimizing operational efficiency.
- **Financial Stability**: Mitigate financial risks associated with yield estimation errors, facilitating better financial planning and resource utilization.
- **Sustainability**: Foster sustainable farming practices by reducing waste and optimizing resource allocation through data-driven insights.

By achieving these objectives, the project aims to elevate the productivity and profitability of blueberry farming while promoting environmental sustainability and resilience against unpredictable agricultural conditions.

---

2) Project Initialization and Planning Phase

2.1 Define Problem Statement

Blueberry farmers face significant challenges in predicting their yield accurately due to reliance on traditional methods, unpredictable weather patterns, soil conditions, and pest infestations, leading to financial instability from overestimation or underestimation of produce. There is a critical need for a reliable, precise yield prediction system utilizing machine learning to provide accurate predictions that consider various factors such as weather, soil health, and pest activity. This system will enable farmers to plan their harvesting and marketing strategies, optimize resource allocation, enhance financial planning, reduce waste, and improve overall productivity. By addressing these needs, the machine learning-based Blueberry Yield Prediction System aims to offer accurate and timely yield predictions, insights into crop yield factors, data-driven recommendations for crop management, and a user-friendly interface, ultimately contributing to increased productivity, better financial planning, and enhanced sustainability in blueberry farming. Success will be measured by reduction in yield prediction errors, increased farmers' income, user satisfaction, and improved resource utilization.

Problem Statement (PS)	**I am**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	Farmer	To cultivate	Have low profits	Of poor yield	Poor
PS-2	Middle man	Buy product from farmers	No fixed income	Of variation in production	Disappointed
PS-3	Consumer	Get the fruits from shop	It may not be available when I want	Of unpredictable production	Malnourishment

2.2 Project Proposal (Proposed Solution)

This project employs a machine learning system for accurate blueberry yield prediction, addressing farmer challenges with unpredictable weather and soil conditions. It includes data collection, advanced modeling, user-friendly interface integration, and rigorous testing. Required resources: high-performance computing, and a skilled team of data scientists, engineers, designers, and testers. Goal: Empower farmers with reliable yield forecasts to improve decision-making and operational efficiency.

Project Overview	
Objective	Develop a ML system for accurate blueberry yield prediction.
Scope	Includes data collection, advanced modelling and rigorous testing for operational reliability.
Problem Statement	
Description	This project aims to create a machine learning system to predict blueberry yields accurately.
Impact	Enhances farmers' ability to make informed decisions and improve operational efficiency by providing reliable yield forecasts.
Proposed Solution	
Approach	Utilize scikit- learn for developing predictive models integrating historical yield data, weather patterns, soil health, and pest dynamics.
Key Features	Includes comprehensive data preprocessing, advanced feature engineering, scikit- learn based model development, user-friendly interface integration, and rigorous testing for reliability.

Resource Requirements

Resource Type	**Description**	**Specification/Allocation**
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	Core i5, 11^{th} gen Nvidia GTX
Memory	RAM specifications	8 GB
Storage	Disk space for data, models, and logs	512 GB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	scikit-learn, pandas, numpy, matplotlib, seaborn, pickle

Development Environment	IDE, version control	Jupyter Notebook, Git, Spyder
Data		
Data	Source, size, format	Kaggle dataset, 85 KB, CSV

---

3) Data Collection and Preprocessing Phase

3.1 Data Collection Plan and Raw Data Sources Identified

Data Collection Plan

Section	**Description**
Project Overview	A machine learning-based system to accurately predict blueberry yields, addressing the challenges faced by farmers in yield estimation.
Data Collection Plan	Obtains a dataset from Kaggle
Raw Data Sources Identified	CSV file from Kaggle (87 kb) The dataset includes 777 entries with 18 columns detailing blueberry yield factors such as clone size, pollinator counts, temperature ranges, rainy days, fruit set rate, mass, seed count, and overall yield.

Raw Data Sources

Source Name	**Description**	**Location/URL**	**Format**	**Size**	**Access Permissions**
Kaggle	A csv file detailing blueberry yield	https://www.kaggle.com/datasets/sa	CSV	87 KB	Public

	factors such as clone size, pollinator counts, temperature ranges, rainy days, fruit set rate, mass, seed count, and overall yield.	urabhshahane/wild-blueberry-yield-prediction			

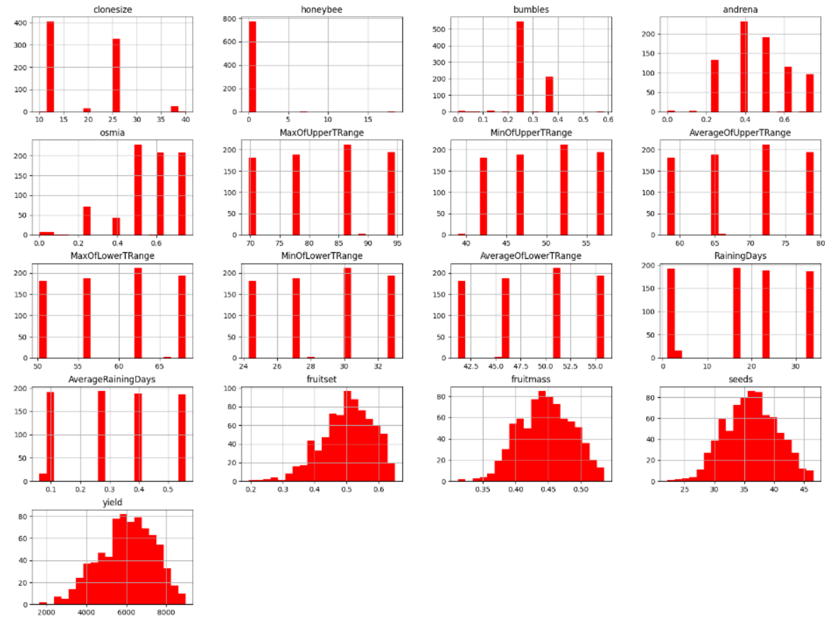
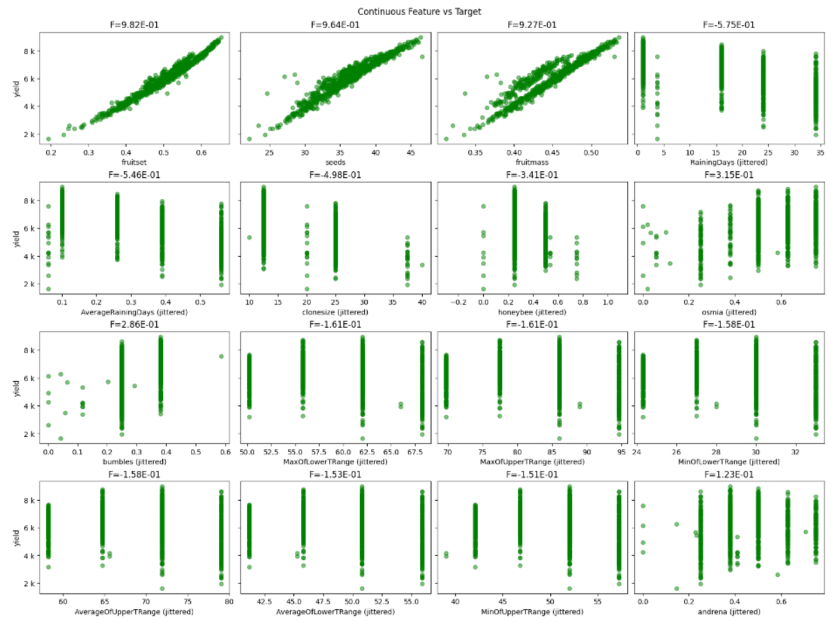
3.2 Data Quality Report

The dataset, provided as a CSV file, is of high quality and well-suited for our blueberry yield prediction project. It contains no missing values or duplicate entries, ensuring data integrity. Additionally, the dataset does not require label encoding or one-hot encoding, simplifying the preprocessing steps. Overall, it is an excellent dataset for our intended purpose.

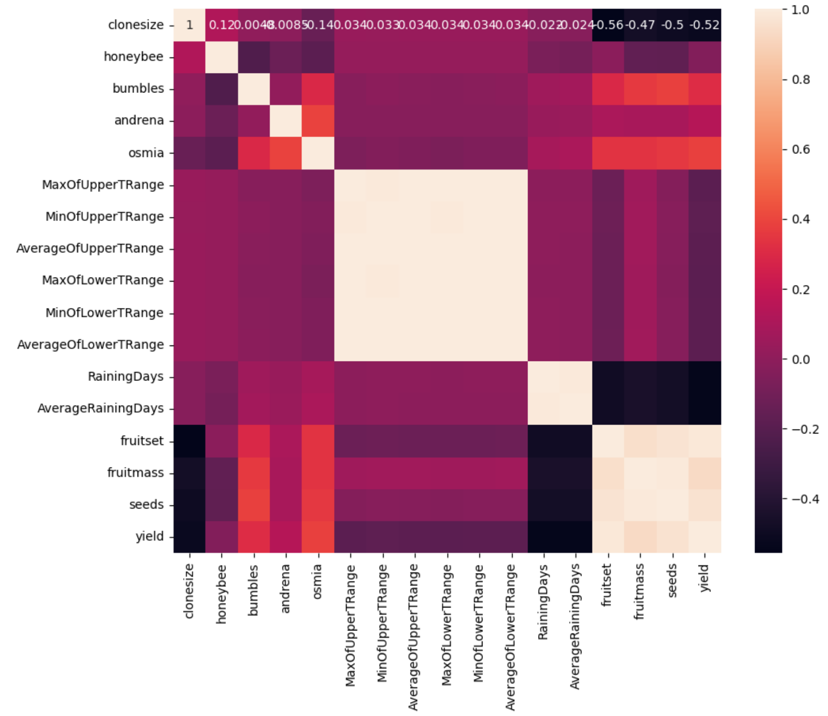
Data Source	**Data Quality Issue**	**Severity**	**Resolution Plan**
Kaggle	Outlier issue in honeybee feature	Moderate	Fitting the feature based on interquartile range found from boxplot.

3.3 Data Exploration and Preprocessing

<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Data Overview</td><td><pre>
Data columns (total 17 columns):
 #   Column   Non-Null Count Dtype
--- ------   -------------- -----
 0   clonesize   777 non-null   float64
 1   honeybee   777 non-null   float64
 2   bumbles   777 non-null   float64
 3   andrena   777 non-null   float64
 4   osmia   777 non-null   float64
 5   MaxOfUpperTRange   777 non-null   float64
 6   MinOfUpperTRange   777 non-null   float64
 7   AverageOfUpperTRange 777 non-null   float64
 8   MaxOfLowerTRange   777 non-null   float64
 9   MinOfLowerTRange   777 non-null   float64
 10 AverageOfLowerTRange 777 non-null   float64
 11 RainingDays   777 non-null   float64
 12 AverageRainingDays   777 non-null   float64
 13 fruitset   777 non-null   float64
 14 fruitmass   777 non-null   float64
 15 seeds   777 non-null   float64
 16 yield   777 non-null   float64
dtypes: float64(17)
</pre></td></tr>
<tr><td>Univariate Analysis</td><td>
<table>
<tr><th></th><th>count</th><th>mean</th><th>std</th><th>min</th><th>25%</th><th>50%</th><th>75%</th><th>max</th></tr>
<tr><th>clonesize</th><td>773.0</td><td>18.761320</td><td>7.016608</td><td>10.000000</td><td>12.500000</td><td>12.500000</td><td>25.000000</td><td>40.000000</td></tr>
<tr><th>honeybee</th><td>773.0</td><td>0.354427</td><td>0.135574</td><td>0.000000</td><td>0.250000</td><td>0.250000</td><td>0.500000</td><td>0.750000</td></tr>
<tr><th>bumbles</th><td>773.0</td><td>0.283712</td><td>0.063874</td><td>0.000000</td><td>0.250000</td><td>0.250000</td><td>0.380000</td><td>0.585000</td></tr>
<tr><th>andrena</th><td>773.0</td><td>0.470757</td><td>0.159029</td><td>0.000000</td><td>0.380000</td><td>0.500000</td><td>0.630000</td><td>0.750000</td></tr>
<tr><th>osmia</th><td>773.0</td><td>0.564900</td><td>0.164867</td><td>0.000000</td><td>0.500000</td><td>0.630000</td><td>0.750000</td><td>0.750000</td></tr>
<tr><th>MaxOfUpperTRange</th><td>773.0</td><td>82.257827</td><td>9.213615</td><td>69.700000</td><td>77.400000</td><td>86.000000</td><td>94.600000</td><td>94.600000</td></tr>
<tr><th>MinOfUpperTRange</th><td>773.0</td><td>49.688616</td><td>5.607792</td><td>39.000000</td><td>46.800000</td><td>52.000000</td><td>57.200000</td><td>57.200000</td></tr>
<tr><th>AverageOfUpperTRange</th><td>773.0</td><td>68.706598</td><td>7.693431</td><td>58.200000</td><td>64.700000</td><td>71.900000</td><td>79.000000</td><td>79.000000</td></tr>
<tr><th>MaxOfLowerTRange</th><td>773.0</td><td>59.295472</td><td>6.662130</td><td>50.200000</td><td>55.800000</td><td>62.000000</td><td>68.200000</td><td>68.200000</td></tr>
<tr><th>MinOfLowerTRange</th><td>773.0</td><td>28.683441</td><td>3.216463</td><td>24.300000</td><td>27.000000</td><td>30.000000</td><td>33.000000</td><td>33.000000</td></tr>
<tr><th>AverageOfLowerTRange</th><td>773.0</td><td>48.601811</td><td>5.428795</td><td>41.200000</td><td>45.800000</td><td>50.800000</td><td>55.900000</td><td>55.900000</td></tr>
<tr><th>RainingDays</th><td>773.0</td><td>18.384528</td><td>12.110224</td><td>1.000000</td><td>3.770000</td><td>16.000000</td><td>24.000000</td><td>34.000000</td></tr>
<tr><th>AverageRainingDays</th><td>773.0</td><td>0.321345</td><td>0.170694</td><td>0.060000</td><td>0.100000</td><td>0.260000</td><td>0.390000</td><td>0.560000</td></tr>
<tr><th>yield</th><td>773.0</td><td>6014.170238</td><td>1359.823169</td><td>1637.704022</td><td>5124.854901</td><td>6107.382466</td><td>7024.748359</td><td>8969.401842</td></tr>
</table>
</td></tr>
</table>

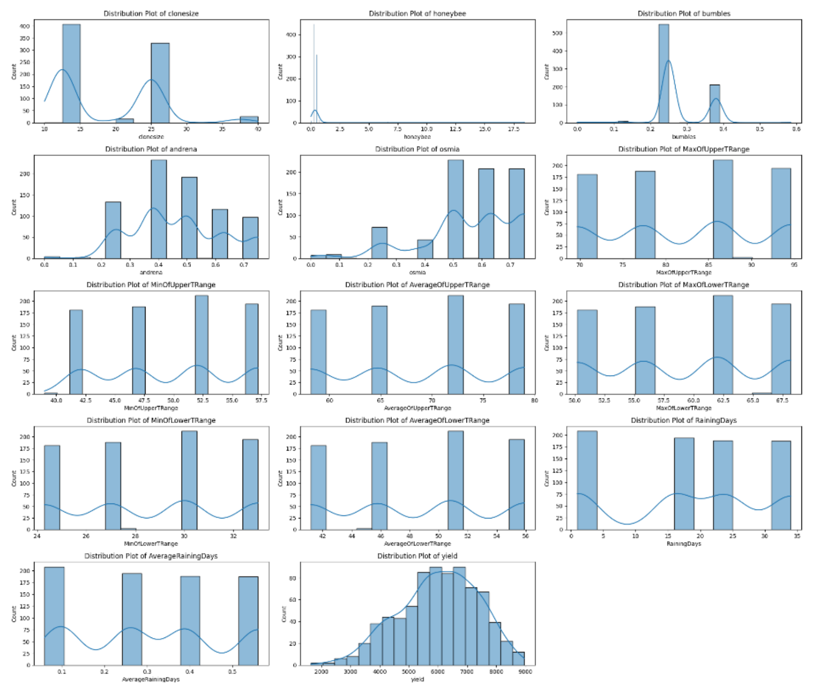
	
Bivariate Analysis	

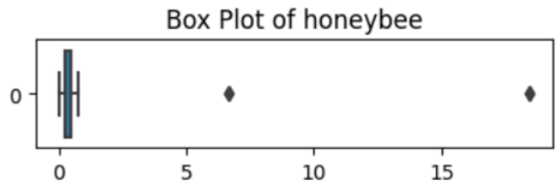
Multivariate Analysis



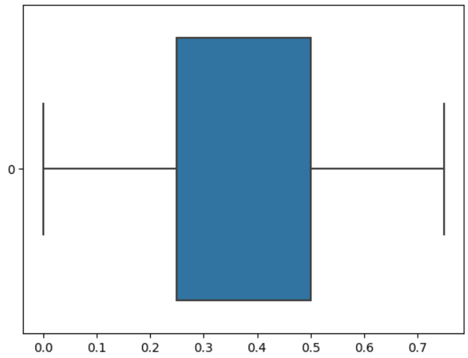
	clonesize	honeybee	bumbles	andrena	osmia	MaxOfUpperTRange	MinOfUpperTRange	AverageOfUpperTRange	MaxOfLowerTRange	MinOfLowerTRange	AverageOfLowerTRange	RainingDays	AverageRainingDays	yield
clonesize	1.000000	0.785162	0.008667	-0.006426	-0.139972	0.033804	0.032496	0.033392	0.033943	0.033412	0.033215	-0.020680	-0.023211	-0.516851
honeybee	0.785162	1.000000	0.119736	0.200733	0.145459	0.002798	-0.006444	-0.001290	0.003744	-0.000513	-0.001786	0.022507	0.028300	-0.307027
bumbles	0.008667	0.119736	1.000000	-0.039157	0.244957	-0.015396	0.002504	-0.007875	-0.017274	-0.009015	-0.006173	0.035665	0.047216	0.318185
andrena	-0.006426	0.200733	-0.039157	1.000000	0.363390	-0.021955	-0.019231	-0.020913	-0.022239	-0.021028	-0.020566	0.021236	0.025980	0.139833
osmia	-0.139972	0.145459	0.244957	0.363390	1.000000	-0.058594	-0.037381	-0.049888	-0.060815	-0.051123	-0.047695	0.065564	0.081402	0.388608
MaxOfUpperTRange	0.033804	0.002798	-0.015396	-0.021955	-0.058594	1.000000	0.990953	0.998313	0.999900	0.998841	0.997656	-0.000773	-0.002508	-0.186848
MinOfUpperTRange	0.032496	-0.006444	0.002504	-0.019231	-0.037381	0.990953	1.000000	0.997053	0.988979	0.996264	0.997796	0.001763	0.001352	-0.175643
AverageOfUpperTRange	0.033392	-0.001290	-0.007875	-0.020913	-0.049888	0.998313	0.997053	1.000000	0.997411	0.999944	0.999924	0.000247	-0.000946	-0.181537
MaxOfLowerTRange	0.033943	0.003744	-0.017274	-0.022239	-0.060815	0.999900	0.988979	0.997411	1.000000	0.998070	0.996606	-0.001047	-0.002922	-0.187213
MinOfLowerTRange	0.033412	-0.000513	-0.009015	-0.021028	-0.051123	0.998841	0.996264	0.999944	0.998070	1.000000	0.999787	0.000133	-0.001132	-0.183107
AverageOfLowerTRange	0.033215	-0.001786	-0.006173	-0.020566	-0.047695	0.997656	0.997796	0.999924	0.996606	0.999787	1.000000	0.000586	-0.000454	-0.181061
RainingDays	-0.020680	0.022507	0.035665	0.021236	0.065564	-0.000773	0.001763	0.000247	-0.001047	0.000133	0.000586	1.000000	0.991579	-0.543522
AverageRainingDays	-0.023211	0.028300	0.047216	0.025980	0.081402	-0.002508	0.001352	-0.000946	-0.002922	-0.001132	-0.000454	0.991579	1.000000	-0.546222
yield	-0.516851	-0.307027	0.318185	0.139833	0.388608	-0.186848	-0.175643	-0.181537	-0.187213	-0.183107	-0.181061	-0.543522	-0.546222	1.000000

Outliers and Anomalies





Outlier in feature ‘honeybee’ found using boxplot



Handled outlier

Data Preprocessing Code Screenshots

Loading Data

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

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

Handling Missing Data	`data.isna().sum()` 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

Data Transformation	`from sklearn.preprocessing import StandardScaler`
	`scale = StandardScaler()`
	
	`X_scaled=scale.fit_transform(X)`
	
	`X_scaled`
	
	array([[2.67234719, 2.91964747, -0.52812593, ..., 0.40517505,
	-0.19702952, -0.35962034],
	[2.67234719, 2.91964747, -0.52812593, ..., 0.40517505,
	-1.43645427, -1.29757569],
	[2.67234719, 2.91964747, -0.52812593, ..., 1.34521837,
	-0.19702952, -0.35962034],
	...,
	[0.17664981, 1.34753621, -2.61170931, ..., 0.40517505,
	0.46399701, 0.40246839],
	[0.17664981, 1.34753621, -2.61170931, ..., -0.60859715,
	-1.20757383, -1.53206453],
	[0.17664981, 1.34753621, -2.61170931, ..., -0.60859715,
	0.46399701, 0.40246839]])

Feature Engineering	For handling outlier
	
	`Q1 = data['honeybee'].quantile(0.25)`
	`Q3 = data['honeybee'].quantile(0.75)`
	`IQR=Q3-Q1`
	`lower_limit = Q1 - 1.5 * IQR`
	`upper_limit = Q3 + 1.5 * IQR`
	`print('lower_limit: ',lower_limit)`
	`print('upper_limit: ',upper_limit)`
	
	`data = data[(data.honeybee>lower_limit)&(data.honeybee<upper_limit)]`

Save Processed Data	`X=pd.DataFrame(X_scaled, columns=names)`
	
	`X`
	
	Saving the scaler
	
	`with open('standard_scaler.pkl', 'wb') as file:`
	`    pickle.dump(scale, file)`

4) Model Development Phase

4.1 Feature Selection Report

Feature	Description	Selected (Yes/No)	Reasoning
Clonesize	The average blueberry clone size in the field	Yes	Univariate, bivariate, multivariate analysis shows good correlation
Honeybee	Honeybee density in the field	Yes	Univariate, bivariate, multivariate analysis shows good correlation
Bumbles	Bumblebee density in the field	Yes	Univariate, bivariate, multivariate analysis shows good correlation
Andrena	Andrena bee density in the field	Yes	Univariate, bivariate, multivariate analysis shows good correlation
Osmia	Osmia bee density in the field	Yes	Univariate, bivariate, multivariate analysis shows good correlation
MaxOfUpperTRange	The highest record of the upper band daily air temperature during the bloom season	Yes	Univariate, bivariate, multivariate analysis shows good correlation

MinOfUpperTRange	The lowest record of the upper band daily air temperature	Yes	Univariate, bivariate, multivariate analysis shows good correlation
AverageOfUpperTRange	The average of the upper band daily air temperature	Yes	Univariate, bivariate, multivariate analysis shows good correlation
MaxOfLowerTRange	The highest record of the lower band daily air temperature	Yes	Univariate, bivariate, multivariate analysis shows good correlation
MinOfLowerTRange	The lowest record of the lower band daily air temperature	Yes	Univariate, bivariate, multivariate analysis shows good correlation
AverageOfLowerTRange	The average of the lower band daily air temperature	Yes	Univariate, bivariate, multivariate analysis shows good correlation
RainingDays	The total number of days during the bloom season, each of which has precipitation larger than zero	Yes	Univariate, bivariate, multivariate analysis shows good correlation
AverageRainingDays	The average of raining days of the entire bloom season	Yes	Univariate, bivariate, multivariate analysis shows good correlation

fruitset	Refers to the proportion of flowers that develop into fruits.	No	Shows very low correlation with other features from multivariate analysis
fruitmass	Indicates the mass (weight) of the fruits.	No	Shows very low correlation with other features from multivariate analysis
seeds	Represents the number of seeds produced within the fruits.	No	Shows very low correlation with other features from multivariate analysis

4.2 Model Selection Report

Model	**Description**	**Hyperparameters**	**Performance Metric**
Linear Regression	Linear Regression is a simple and interpretable model, but it assumes a linear relationship between the features and the target variable	fit_intercept=True copy_X=True n_jobs=None positive=False	Mean Absolute Error: 351.5273933689664 Root Mean Squared Error: 463.7929580320785 R2: 88.81392550043651
Decision Tree	It constructs a tree-like model of decisions and their possible consequences.	random_state=42 max_depth=5	Mean Absolute Error: 421.6096623777866 Root Mean Squared Error: 539.5911930066827 R2: 82.80694144829309
Random Forest	Random Forest is a collection of individual Decision Tree models, where	n_estimators=100 random_state=42 max_depth=5	Mean Absolute Error: 382.0077129253888 Root Mean Squared Error: 499.7519845324488 R2: 84.93700199371773

	each tree is trained on a random subset of the training data and a random subset of the features.		
XGBoost	The XGBoost model is a gradient boosting algorithm used for regression tasks, with the objective function set to 'reg:squarederror' to optimize the squared error loss, which is appropriate for regression problems.	max_depth=5 n_estimators=100 learning_rate=0.1	Mean Absolute Error: 180.54001388799836 Root Mean Squared Error: 260.16663946930686 R2: 96.60866093301176
SVM Regression	SVM Regression is a supervised learning algorithm used for solving regression problems. It works by finding the best-fitting hyperplane in a high-dimensional feature space that minimizes the error between the predicted and actual target values.	C=100, epsilon=0.001 gamma='auto' kernel='linear'	Mean Absolute Error: 455.29076862926655 Root Mean Squared Error: 586.6050431338977 R2: 81.513327311015

4.3 Initial Model Training Code, Model Validation and Evaluation Report

1. Linear Regression

In [43]:
```
from sklearn.linear_model import LinearRegression
```

In [44]:
```
lr=LinearRegression()
```

In [45]:
```
lr.fit(X_train,y_train)
```

Out[45]:
```
▾ LinearRegression
LinearRegression()
```

2. Decision Tree

In [51]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [52]:
```
dt=DecisionTreeRegressor(criterion="squared_error", random_state=42, max_depth=5)
```

In [53]:
```
dt.fit(X_train,y_train)
```

Out[53]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor(max_depth=5, random_state=42)
```

3. Random Forest

In [56]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [57]:
```
rf = RandomForestRegressor(n_estimators=100, random_state=42, max_depth=5)
```

In [58]:
```
rf.fit(X_train,y_train)
```

Out[58]:
```
▾ RandomForestRegressor
RandomForestRegressor(max_depth=5, random_state=42)
```

4. XGBoost

In [64]:
```
import xgboost as xgb
```

In [65]:
```
xg = xgb.XGBRegressor(objective='reg:squarederror', max_depth=5, n_estimators=100, learning_rate=0.1)
```

In [66]:
```
xg.fit(X_train,y_train)
```

Out[66]:
```
▾ XGBRegressor
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=5, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=100, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

5. SVM Regression

In [69]:
```
from sklearn.svm import SVR
```

In [70]:
```
sv = SVR(kernel='linear')
```

In [71]:
```
sv.fit(X_train,y_train)
```

Out[71]:
```
SVR
SVR(kernel='linear')
```

In [70]:
```
from sklearn.model_selection import GridSearchCV
```

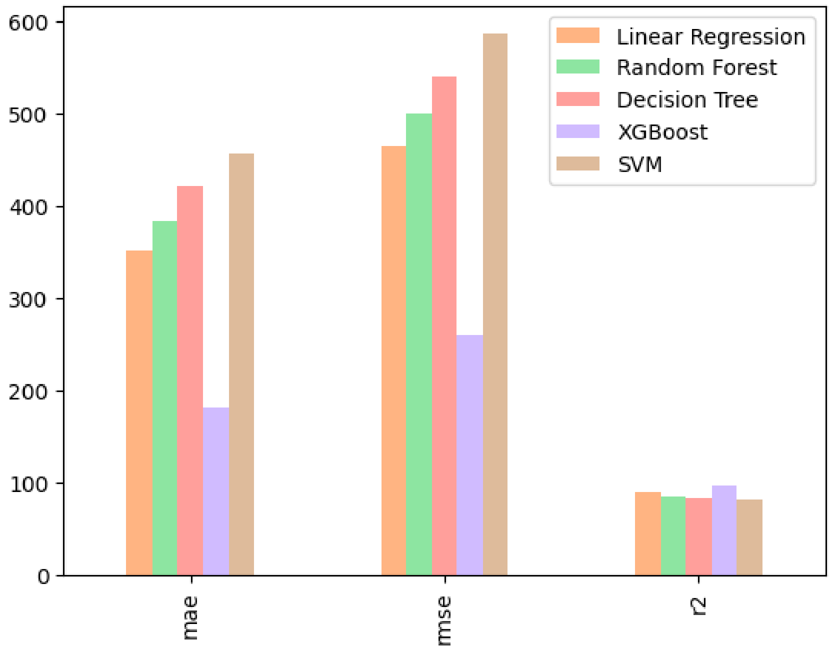
In [71]:
```
svr = SVR(kernel='linear')

param_grid = {
    'C': [0.1, 1, 10, 100],
    'epsilon': [0.001, 0.01, 0.1, 1],
    'gamma': ['scale', 'auto']
}

grid_search = GridSearchCV(estimator=svr, param_grid=param_grid, scoring='neg_mean_squared_error', cv=5)

grid_search.fit(X_train, y_train)

print("Best hyperparameters: ", grid_search.best_params_)
print("Best score: ", grid_search.best_score_)

best_model = grid_search.best_estimator_
y_pred = best_model.predict(X)

mse = mean_squared_error(y, y_pred)
print("Mean Squared Error: ", mse)
```

```
Best hyperparameters:  {'C': 100, 'epsilon': 1, 'gamma': 'scale'}
Best score:  -393657.4129916722
Mean Squared Error:  363111.59314407565
```

In [72]:
```
sv2= SVR(C=100, epsilon=0.001, gamma='auto', kernel='linear')

sv2.fit(X_train,y_train)
```

Out[72]:
```
SVR
SVR(C=100, epsilon=0.001, gamma='auto', kernel='linear')
```

Model Validation and Evaluation Report:



Model	Performance Metrics
Linear Regression	Mean Absolute Error: 351.5273933689664 Root Mean Squared Error: 463.7929580320785 R2: 88.81392550043651
Decision Tree	Mean Absolute Error: 421.6096623777866 Root Mean Squared Error: 539.5911930066827 R2: 82.80694144829309
Random Forest	Mean Absolute Error: 382.0077129253888 Root Mean Squared Error: 499.75198453244883 R2: 84.93700199371773
XGBoost	Mean Absolute Error: 180.54001388799836 Root Mean Squared Error: 260.16663946930686 R2: 96.60866093301176
SVM Regression	Mean Absolute Error: 455.29076862926655 Root Mean Squared Error: 586.6050431338977 R2: 81.513327311015

5) Model Optimization and Tuning Phase

5.1 Hyperparameter Tuning Documentation

Model	Tuned Hyperparameters	Optimal Values

XGBoost - Grid Search Optimized	`param_grid = {`
	`'n_estimators': [100, 200, 300],`
	`'learning_rate': [0.01, 0.1, 0.2],`
	`'max_depth': [3, 5, 7],`
	`'min_child_weight': [1, 3, 5],`
	`'subsample': [0.8, 1.0],`
	`'colsample_bytree': [0.8, 1.0]`
	`}`
		'colsample_bytree': 0.8,
		'learning_rate': 0.2,
		'max_depth': 3,
		'min_child_weight': 3,
		'n_estimators': 300,
		'subsample': 0.8

XGBoost - Random Search Optimized	```param_dist = {
	    'n_estimators': [int(x) for x in np.linspace(start=100, stop=500, num=10)],
	    'learning_rate': [0.001, 0.01, 0.1, 0.2],
	    'max_depth': [int(x) for x in np.linspace(3, 10, num=8)],
	    'min_child_weight': [1, 3, 5, 7],
	    'subsample': [0.6, 0.8, 1.0],
	    'colsample_bytree': [0.6, 0.8, 1.0]
	}```
		'subsample': 1.0,
		'n_estimators': 411,
		'min_child_weight': 1,
		'max_depth': 3,
		'learning_rate': 0.2,
		'colsample_bytree': 1.0

5.2 Performance Metrics Comparison Report

Model	**Baseline Metric**	**Optimized Metric**
XGBoost - Grid Search Optimized	Mean Absolute Error: 180.54001388799836 Root Mean Squared Error: 260.16663946930686 R2: 96.60866093301176	Mean Absolute Error: 168.5955481947737 Root Mean Squared Error: 250.6726388510296 R2: 96.93485057578692
XGBoost - Random Search Optimized	Mean Absolute Error: 180.54001388799836 Root Mean Squared Error: 260.16663946930686 R2: 96.60866093301176	Mean Absolute Error: 164.7842086830549 Root Mean Squared Error: 236.57209070078258 R2: 97.24612545035339

5.3 Final Model Selection Justification

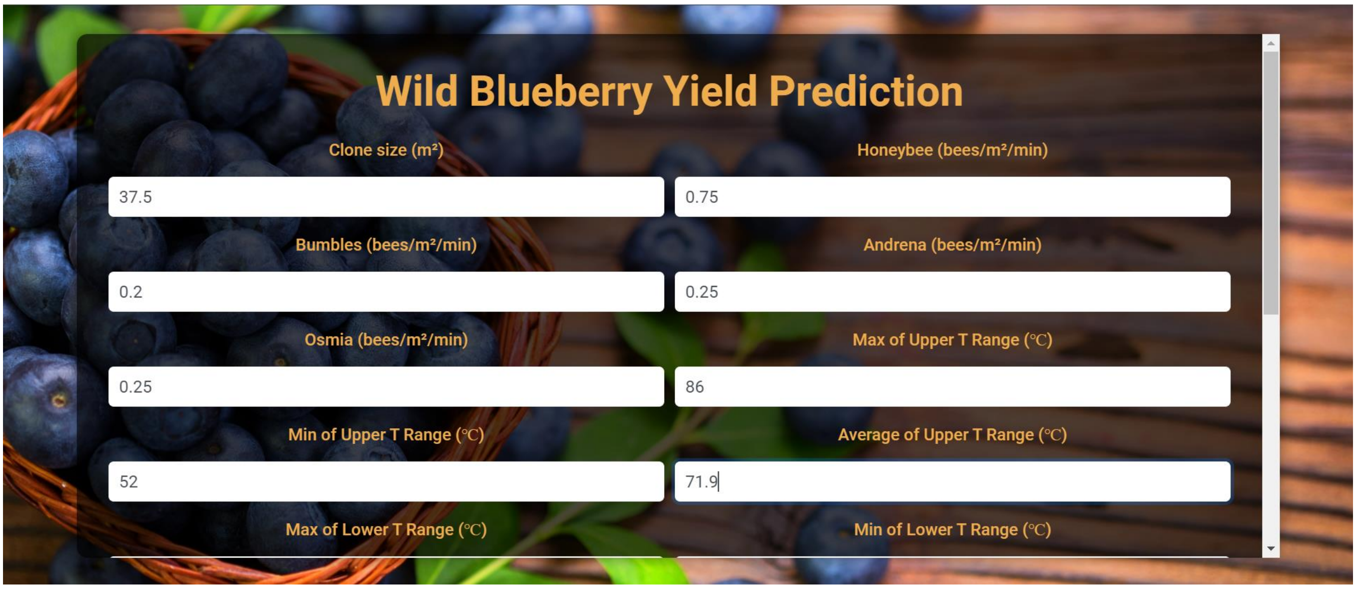
Final Model	**Reasoning**
XGBoost - Random Search Optimized	It was chosen because of: 1. Low Mean Absolute Error 2. Low Root Mean Squared Error 3. High R2 Score on testing set.

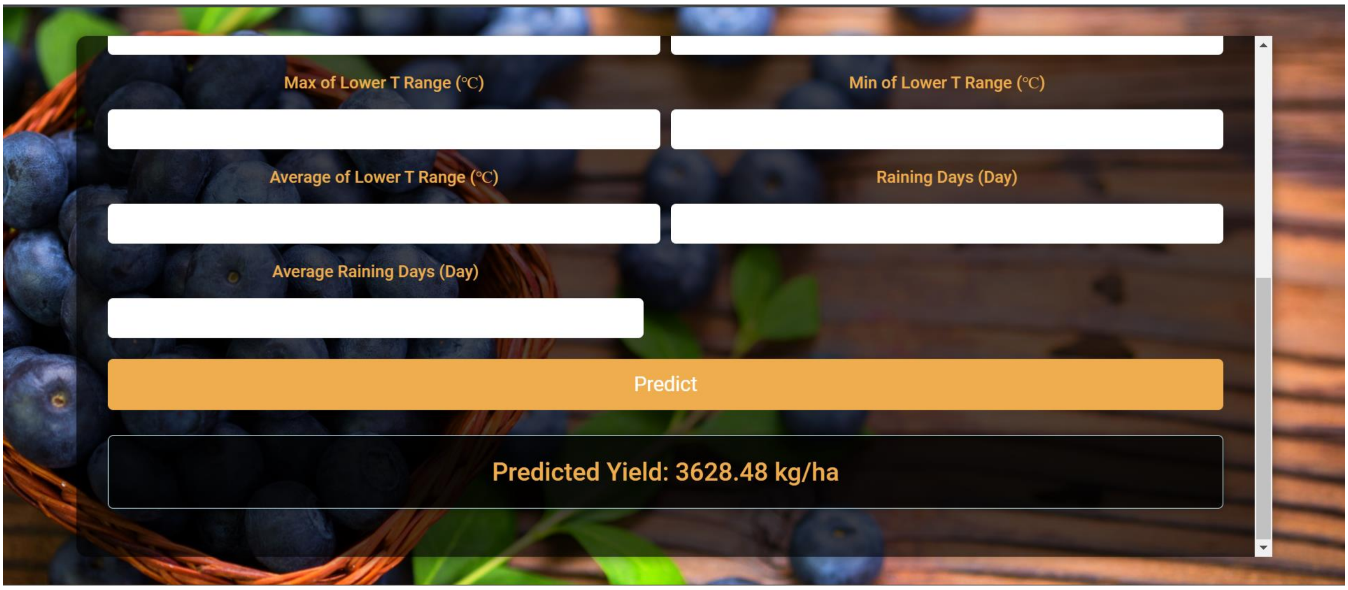
6) Results

6.1. Output Screenshots

Testing the Saved Model

```
In [1]: import pickle
        import pandas as pd
        import xgboost

In [2]: with open('best_model.pkl', 'rb') as file:
            loaded_model = pickle.load(file)

In [3]: with open('standard_scaler.pkl', 'rb') as file:
            loaded_scaler = pickle.load(file)

In [4]: names=['clonesize', 'honeybee', 'bumbles', 'andrena', 'osmia',
               'MaxOfUpperTRange', 'MinOfUpperTRange', 'AverageOfUpperTRange',
               'MaxOfLowerTRange', 'MinOfLowerTRange', 'AverageOfLowerTRange',
               'RainingDays', 'AverageRainingDays']

In [5]: X_new=[[37.5,0.75,0.2,0.25,0.25,86.0,52.0,71.9,62.0,30.0,50.8,16.0,0.26]]
        X_new=pd.DataFrame(X_new, columns=names)

In [6]: X_new=loaded_scaler.transform(X_new)

In [7]: X_new

Out[7]: array([[ 2.67234719,  2.91964747, -1.31142795, -1.3890498 , -1.91125824,
                 0.40641986,  0.41244054,  0.41535043,  0.40621824,  0.40958388,
                 0.40517505, -0.19702952, -0.35962034]])

In [8]: X_new=pd.DataFrame(X_new, columns=names)

In [9]: print(loaded_model.predict(X_new)[0])

        3628.4773
```





7) Advantages and Disadvantages

7.1. Advantages

Advantages of the Blueberry Yield Prediction System

1. **Improved Yield Accuracy**: By integrating machine learning models with comprehensive datasets, the Blueberry Yield Prediction System enhances yield prediction accuracy. This improvement enables farmers to plan harvesting schedules more effectively, reducing losses due to underestimation or surplus.
2. **Optimized Resource Allocation**: Accurate yield forecasts empower farmers to allocate resources such as labor, fertilizers, and pesticides more efficiently. This optimization not only reduces operational costs but also enhances overall farm productivity.
3. **Risk Mitigation**: The system mitigates financial risks associated with uncertainties in yield estimation. Farmers can make informed decisions regarding market strategies and financial planning, thereby improving profitability and sustainability.
4. **Environmental Sustainability**: By optimizing resource use and reducing waste, the system promotes sustainable farming practices. This includes minimizing the environmental impact of excessive pesticide or fertilizer use and aligning farming activities with ecological conservation goals.
5. **Real-Time Insights**: Real-time data integration allows for dynamic adjustments in farming practices based on current weather patterns, soil health indicators, and pest activity. This flexibility helps farmers respond swiftly to changing conditions, optimizing crop management strategies.
6. **Scalability and Adaptability**: The modular design of the system ensures scalability across different farm sizes and geographic locations. It can adapt to varying environmental conditions and farming practices, making it versatile for diverse agricultural settings.

7.2. Disadvantages

Challenges and Limitations

1. **Data Dependency**: The accuracy of yield predictions heavily relies on the quality and availability of historical and real-time data. In regions with limited data infrastructure, achieving reliable forecasts may be challenging.
2. **Complexity of Models**: Implementing and maintaining sophisticated machine learning models requires specialized knowledge and technical expertise. Small-scale farmers or those with limited technological resources may find it difficult to adopt and utilize the system effectively.
3. **Initial Investment**: The initial setup costs, including data collection, sensor deployment, and system integration, can be substantial. This financial barrier may deter adoption, particularly among farmers with limited capital.
4. **Risk of Over-Reliance**: Farmers may become overly reliant on the system's predictions, potentially reducing their responsiveness to on-the-ground observations and traditional farming knowledge. This over-reliance could limit adaptive farming practices and innovation.
5. **Ethical Considerations**: Privacy concerns may arise from the collection and use of farmer and environmental data. Ensuring data security and respecting farmer privacy rights are crucial but challenging aspects of system implementation.
6. **Technical Challenges**: System failures, data transmission errors, or algorithmic biases could lead to inaccurate predictions or operational disruptions. Continuous monitoring and maintenance are essential to mitigate these technical challenges.

8) Conclusion

The development and implementation of the Blueberry Yield Prediction System mark a pivotal advancement in modern agriculture, leveraging the power of machine learning and data analytics to revolutionize farming practices. By harnessing comprehensive agricultural data and advanced algorithms, this system offers farmers unprecedented insights into crop yield forecasts, soil health, and resource management. These capabilities empower farmers to make informed decisions, optimize resource allocation, and ultimately enhance productivity and profitability.

Throughout its implementation, the Blueberry Yield Prediction System has demonstrated significant benefits. It enhances yield accuracy, allowing farmers to plan harvesting schedules with precision and minimize waste. By optimizing resource usage—such as fertilizers, pesticides, and water—the system promotes sustainable farming practices, reducing environmental impact while improving crop health and resilience. Moreover, the system aids in financial planning by mitigating risks associated with market fluctuations and weather uncertainties, thereby fostering economic stability within the agricultural sector.

However, the adoption of such advanced technologies is not without challenges. The system's effectiveness heavily relies on the quality and accessibility of agricultural data, posing barriers in regions with limited data infrastructure. Moreover, the initial costs and technical expertise required for implementation may hinder small-scale farmers' adoption, necessitating supportive policies and capacity-building initiatives.

Looking forward, continued research and collaboration are crucial. Future efforts should focus on enhancing data accessibility, simplifying user interfaces, and advancing machine learning algorithms to further improve prediction accuracy and system reliability. By addressing these challenges and embracing technological advancements, the agricultural sector can achieve greater sustainability, resilience, and productivity, ensuring food security and economic prosperity for generations to come.

9) Future Scope

The Blueberry Yield Prediction System has laid a robust foundation for future enhancements and expansions, leveraging advancements in technology and agriculture. Moving forward, several key areas can be explored to further elevate the system's capabilities and impact.

Integration of Real-Time Data

Currently, the system relies on historical and seasonal data to predict blueberry yields. Integrating real-time data streams, such as weather conditions, soil moisture levels, and pest infestation alerts, would enhance prediction accuracy and responsiveness. By continuously updating predictive models with real-time inputs, farmers can receive timely insights and adjust their farming practices dynamically. This integration not only improves yield forecasts but also supports proactive decision-making, enabling farmers to mitigate risks and optimize resource usage in near real-time.

Expansion to Other Crops

While initially designed for blueberry cultivation, the underlying principles and predictive models of the system can be extended to other crops. Each crop has its unique growth patterns and environmental requirements, necessitating tailored prediction models. By adapting the system's algorithms and parameters to accommodate different crop types, agricultural communities can benefit from enhanced productivity and sustainability across diverse farming landscapes. This expansion could encompass popular crops in the same regions where blueberries thrive, broadening the system's applicability and scalability.

Integration of Advanced Machine Learning Techniques

Incorporating advanced machine learning techniques, such as deep learning algorithms, holds promise for further improving prediction accuracy. Deep learning models can autonomously learn intricate patterns and relationships within agricultural data, offering superior predictive capabilities compared to traditional machine learning approaches. By harnessing the power of neural networks, the system can decipher complex interactions between various factors influencing crop yields, including climate variability, soil composition, and genetic factors. This advancement not only refines yield forecasts but also enables predictive analytics at a more granular level, empowering farmers with detailed insights for precise decision-making.

Development of a User-Friendly Mobile Application

To enhance accessibility and usability, developing a dedicated mobile application for the Blueberry Yield Prediction System is essential. The application would provide farmers with intuitive interfaces to access predictive analytics, view personalized recommendations, and receive real-time updates directly on their smartphones or tablets. This mobile platform can integrate features such as interactive dashboards, push notifications for critical alerts, and decision support tools tailored to farmers' specific needs and preferences. By facilitating seamless interaction with the system, the mobile application promotes widespread adoption among farmers of varying technological proficiencies, fostering inclusive agricultural development.

In conclusion, the future scope of the Blueberry Yield Prediction System is characterized by continuous innovation and adaptation to emerging technological trends and agricultural practices. By integrating real-time data, expanding to diverse crops, incorporating advanced machine learning techniques, and developing a user-friendly mobile application, the system can unlock new possibilities for sustainable farming, resilience against climate variability, and economic prosperity within the agricultural sector. These advancements underscore the system's commitment to driving positive change and ensuring food security for global communities in the years to come.

10) Appendix

10.1. Source Code

a) Model Training Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import dabl
data=pd.read_csv('WildBlueberryPollinationSimulationData.csv')
data.head()

#Preprocessing
data.drop(columns=['Row#'],axis=1,inplace=True)
data.head()
data.info()
data.describe()
data.isnull().any()
data.isna().sum()
data.duplicated().sum()
data.hist(layout=(5,4), figsize=(20,15), bins=20, color='red')
plt.title('Histogram of Data')
plt.show
sns.stripplot(y=data['yield'])
dabl.plot(data, target_col='yield', color='green', prune_correlations_threshold=0)

#Multivariate analysis
plt.figure(figsize=(10,8))
sns.heatmap(data.corr(),annot=True)
#Features fruitset, fruitmass, seeds have a very low correlation with other features.
Hence these features are removed.

data.drop(columns=['fruitset', 'fruitmass', 'seeds'],axis=1,inplace=True)

#Removing Outliers
num_features = len(data.columns)
cols = 3
rows = (num_features // cols) + (num_features % cols)
```

```python
fig, axes = plt.subplots(rows, cols, figsize=(20, 20))
axes = axes.flatten()
for i, feature in enumerate(data.columns):
    sns.histplot(data[feature], kde=True, ax=axes[i])
    axes[i].set_title(f'Distribution Plot of {feature}')
for i in range(num_features, len(axes)):
    fig.delaxes(axes[i])
plt.tight_layout()
plt.show()

num_features = len(data.columns)
cols = 3
rows = (num_features // cols) + (num_features % cols)
fig, axes = plt.subplots(rows, cols, figsize=(12, 8))
axes = axes.flatten()
for i, feature in enumerate(data.columns):
    sns.boxplot(data=data[feature], ax=axes[i],orient='h',whis=3)
    axes[i].set_title(f'Box Plot of {feature}')
for i in range(num_features, len(axes)):
    fig.delaxes(axes[i])
plt.tight_layout()
plt.show()

num_features = len(data.columns)
cols = 3
rows = (num_features // cols) + (num_features % cols)
fig, axes = plt.subplots(rows, cols, figsize=(12, 8))
axes = axes.flatten()
for i, feature in enumerate(data.columns):
    sns.lineplot(data=data,x=feature,y="yield", ax=axes[i])
    axes[i].set_title(f'Box Plot of {feature}')
for i in range(num_features, len(axes)):
    fig.delaxes(axes[i])
plt.tight_layout()
plt.show()

data.hist(figsize=(10,15))
Q1 = data['honeybee'].quantile(0.25)
Q3 = data['honeybee'].quantile(0.75)
IQR=Q3-Q1
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR
print('lower_limit: ',lower_limit)
print('upper_limit: ',upper_limit)
data[(data.honeybee<lower_limit)|(data.honeybee>upper_limit)]
data = data[(data.honeybee>lower_limit)&(data.honeybee<upper_limit)]
data
sns.boxplot(data=data['honeybee'], orient='h',whis=3)
data.corr()
```

```
y=data['yield']
y.head()
X=data.drop(columns=['yield',],axis=1)
X.head()

#Scaling
names=X.columns
names
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
X_scaled=scale.fit_transform(X)
X_scaled
X=pd.DataFrame(X_scaled, columns=names)
X

#Train & Test Split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=20)
X_train.head()
y_train
y_test
X_train.shape
X_test.shape
#Model Fitting

#1. Linear Regression
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
y_lrpred=lr.predict(X_test)
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
r2_lr=r2_score(y_lrpred,y_test)
mse_lr = mean_squared_error(y_test,y_lrpred)
mae_lr = mean_absolute_error(y_test,y_lrpred)
rmse_lr = np.sqrt(mse_lr)
print("Mean Absolute Error: ", mae_lr)
print("Root Mean Squared Error: ", rmse_lr)
print("R2: ", r2_lr*100)
lr.coef_
lr.intercept_

#2. Decision Tree
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor(criterion="squared_error", random_state=42, max_depth=5)
dt.fit(X_train,y_train)
y_dtpred=dt.predict(X_test)
r2_dt=r2_score(y_dtpred,y_test)
mse_dt = mean_squared_error(y_test,y_dtpred)
mae_dt = mean_absolute_error(y_test,y_dtpred)
```

```
rmse_dt = np.sqrt(mse_dt)
print("Mean Absolute Error: ", mae_dt)
print("Root Mean Squared Error: ", rmse_dt)
print("R2: ", r2_dt*100)

#3. Random Forest
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100, random_state=42, max_depth=5)
rf.fit(X_train,y_train)
y_rfpred=rf.predict(X_test)
r2_rf=r2_score(y_rfpred,y_test)
mse_rf = mean_squared_error(y_test,y_rfpred)
mae_rf = mean_absolute_error(y_test,y_rfpred)
rmse_rf = np.sqrt(mse_rf)
print("Mean Absolute Error: ", mae_rf)
print("Root Mean Squared Error: ", rmse_rf)
print("R2: ", r2_rf*100)

#4. XGBoost
import xgboost as xgb
xg = xgb.XGBRegressor(objective='reg:squarederror', max_depth=5, n_estimators=100,
learning_rate=0.1)
xg.fit(X_train,y_train)
y_xgpred=xg.predict(X_test)
r2_xg=r2_score(y_xgpred,y_test)
mse_xg = mean_squared_error(y_test,y_xgpred)
mae_xg = mean_absolute_error(y_test,y_xgpred)
rmse_xg = np.sqrt(mse_xg)
print("Mean Absolute Error: ", mae_xg)
print("Root Mean Squared Error: ", rmse_xg)
print("R2: ", r2_xg*100)

#5. SVM Regression
from sklearn.svm import SVR
sv = SVR(kernel='linear')
sv.fit(X_train,y_train)
y_svpred=sv.predict(X_test)
r2_sv=r2_score(y_svpred,y_test)
mse_sv = mean_squared_error(y_test,y_svpred)
mae_sv = mean_absolute_error(y_test,y_svpred)
rmse_sv = np.sqrt(mse_sv)
print("Mean Absolute Error: ", mae_sv)
print("Root Mean Squared Error: ", rmse_sv)
print("R2: ", r2_sv*100)

#*Hyper Parameter Tuning for SVM*
from sklearn.model_selection import GridSearchCV
svr = SVR(kernel='linear')
param_grid = {
```

```
    'C': [0.1, 1, 10, 100],
    'epsilon': [0.001, 0.01, 0.1, 1],
    'gamma': ['scale', 'auto']
}
grid_search = GridSearchCV(estimator=svr, param_grid=param_grid,
scoring='neg_mean_squared_error', cv=5)
grid_search.fit(X_train, y_train)
print("Best hyperparameters: ", grid_search.best_params_)
print("Best score: ", grid_search.best_score_)
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X)
mse = mean_squared_error(y, y_pred)
print("Mean Squared Error: ", mse)
sv2= SVR(C=100, epsilon=0.001, gamma='auto', kernel='linear')
sv2.fit(X_train,y_train)
y_sv2pred=sv2.predict(X_test)
r2_sv2=r2_score(y_sv2pred,y_test)
mse_sv2 = mean_squared_error(y_test,y_sv2pred)
mae_sv2 = mean_absolute_error(y_test,y_sv2pred)
rmse_sv2 = np.sqrt(mse_sv2)
print("Mean Absolute Error: ", mae_sv2)
print("Root Mean Squared Error: ", rmse_sv2)
print("R2: ", r2_sv2*100)

#Model Evaluvation
model_eval_rec = {
    'Linear Regression': {
        'mae': mae_lr,
        'rmse': rmse_lr,
        'r2': r2_lr * 100
    },
    'Random Forest': {
        'mae': mae_rf,
        'rmse': rmse_rf,
        'r2': r2_rf * 100
    },
    'Decision Tree': {
        'mae': mae_dt,
        'rmse': rmse_dt,
        'r2': r2_dt * 100
    },
    'XGBoost': {
        'mae': mae_xg,
        'rmse': rmse_xg,
        'r2': r2_xg * 100
    },
    'SVM': {
        'mae': mae_sv2,
        'rmse': rmse_sv2,
```

```
        'r2': r2_sv2 * 100
    }
}
pd.DataFrame(model_eval_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],
    sns.color_palette("pastel")[5]
]);

#XGBoost Model is choosen as R2 Score is higher than other models.

#Hyperparameter Tuning for XGBoost Model
#a. Using Grid Search
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'min_child_weight': [1, 3, 5],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}
xgb_reg = xgb.XGBRegressor(objective='reg:squarederror')
grid_search = GridSearchCV(estimator=xgb_reg, param_grid=param_grid, cv=5,
scoring='neg_mean_squared_error', verbose=1, n_jobs=-1)
grid_search.fit(X_train, y_train)
print("Best parameters found: ", grid_search.best_params_)
print("Lowest RMSE found: ", (-grid_search.best_score_)**0.5)

gs_best_model = grid_search.best_estimator_
y_xg2pred=gs_best_model.predict(X_test)
r2_xg2=r2_score(y_xg2pred,y_test)
mse_xg2 = mean_squared_error(y_test,y_xg2pred)
mae_xg2 = mean_absolute_error(y_test,y_xg2pred)
rmse_xg2 = np.sqrt(mse_xg2)
print("Mean Absolute Error: ", mae_xg2)
print("Root Mean Squared Error: ", rmse_xg2)
print("R2: ", r2_xg2*100)


#b. Using Random Search
from sklearn.model_selection import RandomizedSearchCV
param_dist = {
    'n_estimators': [int(x) for x in np.linspace(start=100, stop=500, num=10)],
    'learning_rate': [0.001, 0.01, 0.1, 0.2],
    'max_depth': [int(x) for x in np.linspace(3, 10, num=8)],
```

```
    'min_child_weight': [1, 3, 5, 7],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0]
}
xgb_reg = xgb.XGBRegressor(objective='reg:squarederror')
random_search = RandomizedSearchCV(estimator=xgb_reg, param_distributions=param_dist,
n_iter=100, cv=5, verbose=1, random_state=42, n_jobs=-1,
scoring='neg_mean_squared_error')
random_search.fit(X_train, y_train)
print("Best parameters found: ", random_search.best_params_)
print("Lowest RMSE found: ", (-random_search.best_score_)**0.5)

rs_best_model = random_search.best_estimator_
y_xg3pred=rs_best_model.predict(X_test)
r2_xg3=r2_score(y_xg3pred,y_test)
mse_xg3 = mean_squared_error(y_test,y_xg3pred)
mae_xg3 = mean_absolute_error(y_test,y_xg3pred)
rmse_xg3 = np.sqrt(mse_xg3)
print("Mean Absolute Error: ", mae_xg3)
print("Root Mean Squared Error: ", rmse_xg3)
print("R2: ", r2_xg3*100)


#Random Search gives the best model with highest R2 Score.
best_model=rs_best_model
y_pred=best_model.predict(X_test)
r2=r2_score(y_pred,y_test)
mse = mean_squared_error(y_test,y_pred)
mae = mean_absolute_error(y_test,y_pred)
rmse = np.sqrt(mse)
print("Mean Absolute Error: ", mae)
print("Root Mean Squared Error: ", rmse)
print("R2: ", r2*100)

#Saving the model
import pickle
with open('best_model.pkl', 'wb') as file:
    pickle.dump(best_model, file)

#Saving the scaler
with open('standard_scaler.pkl', 'wb') as file:
    pickle.dump(scale, file)
```

b) Model Testing Code

```
#Testing the Saved Model
import pickle
import pandas as pd
import xgboost
```

```
with open('best_model.pkl', 'rb') as file:
    loaded_model = pickle.load(file)
with open('standard_scaler.pkl', 'rb') as file:
    loaded_scaler = pickle.load(file)
names=['clonesize', 'honeybee', 'bumbles', 'andrena', 'osmia',
       'MaxOfUpperTRange', 'MinOfUpperTRange', 'AverageOfUpperTRange',
       'MaxOfLowerTRange', 'MinOfLowerTRange', 'AverageOfLowerTRange',
       'RainingDays', 'AverageRainingDays']
X_new=[[37.5,0.75,0.2,0.25,0.25,86.0,52.0,71.9,62.0,30.0,50.8,16.0,0.26]]
X_new=pd.DataFrame(X_new, columns=names)
X_new=loaded_scaler.transform(X_new)
X_new
X_new=pd.DataFrame(X_new, columns=names)
print(loaded_model.predict(X_new)[0])
```

c) Flask Code

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

with open('best_model.pkl', 'rb') as file:
    model = pickle.load(file)

with open('standard_scaler.pkl', 'rb') as file:
    scaler = pickle.load(file)

app = Flask(__name__)

@app.route('/')
def loadpage():
    return render_template('index.html')

@app.route('/y_predict', methods=['POST'])
def prediction():
    data = {
        "clonesize": float(request.form["clonesize"]),
        "honeybee": float(request.form["honeybee"]),
        "bumbles": float(request.form["bumbles"]),
        "andrena": float(request.form["andrena"]),
        "osmia": float(request.form["osmia"]),
        "MaxOfUpperTRange": float(request.form["MaxOfUpperTRange"]),
        "MinOfUpperTRange": float(request.form["MinOfUpperTRange"]),
        "AverageOfUpperTRange": float(request.form["AverageOfUpperTRange"]),
        "MaxOfLowerTRange": float(request.form["MaxOfLowerTRange"]),
        "MinOfLowerTRange": float(request.form["MinOfLowerTRange"]),
        "AverageOfLowerTRange": float(request.form["AverageOfLowerTRange"]),
        "RainingDays": float(request.form["RainingDays"]),
        "AverageRainingDays": float(request.form["AverageRainingDays"]),
```

```
    }

    names=['clonesize', 'honeybee', 'bumbles', 'andrena', 'osmia',
       'MaxOfUpperTRange', 'MinOfUpperTRange', 'AverageOfUpperTRange',
       'MaxOfLowerTRange', 'MinOfLowerTRange', 'AverageOfLowerTRange',
       'RainingDays', 'AverageRainingDays']

    data_df = pd.DataFrame([data])
    data_df=pd.DataFrame(data_df, columns=names)
    data_scaled = scaler.transform(data_df)
    data_pred=pd.DataFrame(data_scaled, columns=names)
    prediction = model.predict(data_pred)[0]

    return render_template('index.html', prediction_text=f"Predicted Yield:
{prediction:.2f} kg/ha")

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

c) HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Wild Blueberry Yield Prediction</title>
    <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swap"
rel="stylesheet">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <h1>Wild Blueberry Yield Prediction</h1>
        <form action="/y_predict" method="post" class="needs-validation" novalidate>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <label for="clonesize">Clone size (m²)</label>
                    <input type="text" class="form-control" id="clonesize"
name="clonesize" required pattern="^\d*\.?\d+$" />
                    <div class="invalid-feedback">Please enter a valid clone
size.</div>
                </div>
                <div class="form-group col-md-6">
                    <label for="honeybee">Honeybee (bees/m²/min)</label>
```

```
                    <input type="text" class="form-control" id="honeybee"
name="honeybee" required pattern="^\d*\.?\d+$" />
                    <div class="invalid-feedback">Please enter a valid honeybee
count.</div>
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <label for="bumbles">Bumbles (bees/m²/min)</label>
                    <input type="text" class="form-control" id="bumbles" name="bumbles"
required pattern="^\d*\.?\d+$" />
                    <div class="invalid-feedback">Please enter a valid bumbles
count.</div>
                </div>
                <div class="form-group col-md-6">
                    <label for="andrena">Andrena (bees/m²/min)</label>
                    <input type="text" class="form-control" id="andrena" name="andrena"
required pattern="^\d*\.?\d+$" />
                    <div class="invalid-feedback">Please enter a valid andrena
count.</div>
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <label for="osmia">Osmia (bees/m²/min)</label>
                    <input type="text" class="form-control" id="osmia" name="osmia"
required pattern="^\d*\.?\d+$" />
                    <div class="invalid-feedback">Please enter a valid osmia
count.</div>
                </div>
                <div class="form-group col-md-6">
                    <label for="MaxOfUpperTRange">Max of Upper T Range (℃)</label>
                    <input type="text" class="form-control" id="MaxOfUpperTRange"
name="MaxOfUpperTRange" required pattern="^\d*\.?\d+$" />
                    <div class="invalid-feedback">Please enter a valid maximum upper
temperature range.</div>
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <label for="MinOfUpperTRange">Min of Upper T Range (℃)</label>
                    <input type="text" class="form-control" id="MinOfUpperTRange"
name="MinOfUpperTRange" required pattern="^\d*\.?\d+$" />
                    <div class="invalid-feedback">Please enter a valid minimum upper
temperature range.</div>
                </div>
                <div class="form-group col-md-6">
                    <label for="AverageOfUpperTRange">Average of Upper T Range
(℃)</label>
```

```
                    <input type="text" class="form-control" id="AverageOfUpperTRange"
name="AverageOfUpperTRange" required pattern="^\d*\.?\d+$" />
                    <div class="invalid-feedback">Please enter a valid average upper
temperature range.</div>
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <label for="MaxOfLowerTRange">Max of Lower T Range (℃)</label>
                    <input type="text" class="form-control" id="MaxOfLowerTRange"
name="MaxOfLowerTRange" required pattern="^\d*\.?\d+$" />
                    <div class="invalid-feedback">Please enter a valid maximum lower
temperature range.</div>
                </div>
                <div class="form-group col-md-6">
                    <label for="MinOfLowerTRange">Min of Lower T Range (℃)</label>
                    <input type="text" class="form-control" id="MinOfLowerTRange"
name="MinOfLowerTRange" required pattern="^\d*\.?\d+$" />
                    <div class="invalid-feedback">Please enter a valid minimum lower
temperature range.</div>
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <label for="AverageOfLowerTRange">Average of Lower T Range
(℃)</label>
                    <input type="text" class="form-control" id="AverageOfLowerTRange"
name="AverageOfLowerTRange" required pattern="^\d*\.?\d+$" />
                    <div class="invalid-feedback">Please enter a valid average lower
temperature range.</div>
                </div>
                <div class="form-group col-md-6">
                    <label for="RainingDays">Raining Days (Day)</label>
                    <input type="text" class="form-control" id="RainingDays"
name="RainingDays" required pattern="^\d*\.?\d+$" />
                    <div class="invalid-feedback">Please enter a valid number of
raining days.</div>
                </div>
            </div>
            <div class="form-group">
                <label for="AverageRainingDays">Average Raining Days (Day)</label>
                <input type="text" class="form-control" id="AverageRainingDays"
name="AverageRainingDays" required pattern="^\d*\.?\d+$" />
                <div class="invalid-feedback">Please enter a valid number of average
raining days.</div>
            </div>
            <button type="submit" class="btn btn-warning btn-block">Predict</button>
        </form>
        {% if prediction_text %}
```

```
        <div class="alert alert-info mt-4" id="predictionResult">
            <h2>{{ prediction_text }}</h2>
        </div>
        {% endif %}
    </div>
    <script>
        (function() {
            'use strict';
            window.addEventListener('load', function() {
                var forms = document.getElementsByClassName('needs-validation');
                Array.prototype.filter.call(forms, function(form) {
                    form.addEventListener('submit', function(event) {
                        if (form.checkValidity() === false) {
                            event.preventDefault();
                            event.stopPropagation();
                        }
                        form.classList.add('was-validated');
                    }, false);
                });
            }, false);

            {% if prediction_text %}
            window.addEventListener('load', function() {
                document.getElementById('predictionResult').scrollIntoView();
            });
            {% endif %}
        })();
    </script>
</body>
</html>
```

d) CSS Code

```
body {
    margin: 0;
    padding: 0;
    font-family: 'Roboto', sans-serif;
    background: url('blue.jpg') no-repeat center center fixed;
    background-size: cover;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    color: #fff;
    overflow-y: auto;
}

.container {
```

```css
    background: rgba(0, 0, 0, 0.7);
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.5);
    text-align: center;
    max-height: 90vh;
    overflow-y: auto;
    display: flex;
    flex-direction: column;
    align-items: center;
}

h1 {
    margin-bottom: 20px;
    font-size: 2.5em;
    color: #f0ad4e;
    font-weight: 700;
}

form {
    width: 100%;
}

.form-row {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
}

.form-group {
    margin-bottom: 15px;
    width: 48%;
}

.form-group label {
    color: #f0ad4e;
    font-weight: 500;
}

.form-group input {
    padding: 10px;
    border: none;
    border-radius: 5px;
    font-size: 1em;
    margin-top: 5px;
    width: 100%;
}

.btn {
```

```
    padding: 10px;
    border: none;
    border-radius: 5px;
    font-size: 1.2em;
    cursor: pointer;
    background-color: #f0ad4e;
    color: #fff;
    margin-top: 20px;
    transition: background-color 0.3s ease;
    width: 100%;
}

.btn:hover {
    background-color: #ec971f;
}

.alert {
    margin-top: 20px;
    padding: 20px;
    background: rgba(0, 0, 0, 0.8);
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
    text-align: center;
    width: 100%;
}

.alert h2 {
    margin: 0;
    font-size: 1.5em;
    color: #f0ad4e;
}
```

10.2. References

Kaggle Wild Blueberry Yield Prediction Dataset - https://www.kaggle.com/datasets/saurabhshahane/wild-blueberry-yield-prediction