

Industrial Internship – BCSE399J
Artificial Intelligence using Google TensorFlow
Blueberry Yield Prediction

A REPORT

submitted by

KUMAR ABHISHEK – 22BCE1907

in partial fulfilment for the award

of

B. Tech. Computer Science and Engineering

School of Computer Science and Engineering



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

October 2024



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering



VIT[®]

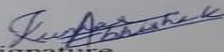
Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

DECLARATION

I hereby declare that the project entitled "**Artificial Intelligence using Google Tensorflow – Blueberry Yield Prediction**" submitted by me to the School of Computer Science and Engineering, Vellore Institute of Technology, Chennai Campus, Chennai 600127 in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology – Computer Science and Engineering** is a record of bonafide work carried out by myself. I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.


Signature

KUMAR ABHISHEK – 22BCE1907



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering



VIT[®]

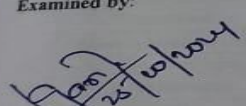
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

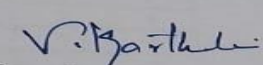
School of Computer Science and Engineering

CERTIFICATE

The project report entitled "**Artificial Intelligence using Google TensorFlow – Blueberry Yield Prediction**" is prepared and submitted by **Kumar Abhishek** (Roll No: **22BCE1907**). It has been found satisfactory in terms of scope, quality, and presentation as partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology – Computer Science and Engineering** at **Vellore Institute of Technology, Chennai, India**.

Examined by:


Examiner I
Dr. KAVITHA J C


Examiner II
Prof. V. KARTHIKA

Certificate of Merit

This is to certify that Mr/Ms. KUMAR ABHISHEK (22BCE1907) has completed the course in **Artificial Intelligence using Google TensorFlow** from **June 2024 to July 2024** securing an overall performance score of **77 out of 100**

Evaluation Metrics:

Metric	Parameter Name	Weightage	Total Score	Your Score
Project Initialization and Planning Phase	Define Problem Statements	3	10	7
	Project Proposal (Proposed Solution)	3		
	Initial Project Planning Report	4		
Data Collection and Preprocessing Phase	Data Collection Plan and Raw Data Sources Identification Report	2	10	6
	Data Quality Report	2		
	Preprocessing Report	6		
Model Development Phase	Model Selection Report	5	15	12
	Initial Model Training Code, Model Validation and Evaluation Report	10		
Model Optimization and Tuning Phase	Tuning Documentation	8	10	8
	Final Model Selection Justification Report	2		
Project Executable Files	Project Executable Files	10	10	9
Project Documentation	Project Documentation	5	5	4
Project Demonstration	Project Demonstration	5	5	4
Attendance	Attendance	5	5	5
Grand Assessment	Grand Assessment	30	30	22
Total			100	77

Issued date: August 12, 2024



Amarendar Katkam
 Founder & CEO, SmartBridge



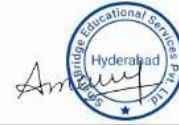
Certificate of Completion

This is to certify that KUMAR ABHISHEK (22BCE1907) has successfully
completed the Credit Course on Artificial Intelligence using Google TensorFlow Powered
by Google Developers from June 2024 to July 2024 and fulfilled the project work
requirements.

Certificate ID: CC-AI-2024-5001

August 12, 2024

Issued Date



Amarendar Katkam
Founder & CEO, SmartBridge





PROJECT COMPLETION CERTIFICATE

THIS CERTIFIES THAT

KUMAR ABHISHEK

has successfully completed the project titled
"Blueberry Yield Prediction"
with expected outcome.

APPROVED BY

TEAM SMARTINTERNZ

August 08, 2024

Verify at: https://skillwallet.smartinternz.com/guided_projects/certificates/5751ec3e9a4feab575962e78e006250d

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to the following individuals for their support and guidance throughout this project:

- Dr. Nithyanandam P, Head of Department (HOD), B.Tech Computer Science and Engineering, School of Computer Science and Engineering, VIT Chennai
- Dr. Ganesan R, Dean of the School of Computer Science & Engineering, VIT Chennai
- Dr. Parvathi R , Associate Dean(Academics) of the School of Computer Science & Engineering, VIT Chennai
- Dr. Geetha S , Associate Dean(Research) of the School of Computer Science & Engineering, VIT Chennai

Their guidance and support have been instrumental in the successful completion of this project.

CONTENTS

Chapter	Title	Page
	Title Page	i
	Declaration	ii
	Certificate	iii
	Industry certificate	iv
	Acknowledgement	v
	Table of contents	vi
	List of Tables	vii
	List of Abbreviations	viii
	Abstract	ix
1	Introduction	11
2	Data Collection and Preprocessing 2.1 Data Exploration and Preprocessing	12
3	Data Quality Report	16
4	Model Development Phase 4.1 Feature Selection Report 4.2 Initial Model Training Code, Model Validation and Evaluation Report 4.3 Model Validation and Evaluation Report 4.4 Model Selection Report	17
5	Model Optimization and Tuning Phase 5.1 Hyperparameter Tuning Documentation 5.2 Performance Metrics Comparison Report 5.3 Final Model Selection Justification	24
6	Web Application Using Flask 6.1 Model Frontend Implementation 6.2 Images of the Web Application	26
7	Conclusion	28
	References	29
	Appendix – I Source Code of Flask Application	30

LIST OF TABLES

Title	Page
Table 1: Data exploration and preprocessing	12
Table 2: Data quality	16
Table 3: Feature selection	17
Table 4: Model validation and evaluation	21
Table 5: Model selection	23
Table 6: Model performance	25

LIST OF ABBREVIATIONS

Abbreviation	Expansion
AI	Artificial Intelligence
ML	Machine Learning
GPU	Graphical Processing Unit
RMSE	Root Mean Squared Error
MSE	Mean Squared Error
CSV	Comma Separated Values
XGB	Extreme Gradient Boosting
RF	Random Forest

ABSTRACT

The agricultural sector is increasingly adopting data-driven methods to enhance crop yields and boost efficiency. Predicting the yield of wild blueberries, however, poses a challenge due to the influence of fluctuating environmental factors like temperature, rainfall, and pollination rates. Traditional yield prediction methods rely heavily on historical data and expert judgment, which can be time-consuming and prone to inaccuracies. Moreover, the absence of real-time forecasting tools limits farmers' ability to make timely decisions regarding resource allocation and crop management.

A key drawback of current practices is their reliance on static data, preventing mid-season adjustments that could optimize yield. Manual prediction processes also introduce the risk of human error and subjective bias, leading to inconsistent outcomes.

This project addresses these challenges by proposing a machine learning-based approach to predict wild blueberry yields. The model analyzes critical environmental and pollination factors—such as temperature, precipitation, and bee activity—to provide more accurate and timely predictions. This enables real-time data processing, empowering farmers to make proactive management decisions.

The solution is deployed as a web application built with the Flask framework, offering a user-friendly interface where farmers can input relevant data. The backend processes the input using a trained machine learning model, delivering predictions that are more reliable and scalable than traditional approaches. Ultimately, this project aims to enhance the accuracy of yield forecasts, promoting more sustainable and efficient blueberry farming practices.

1. INTRODUCTION

The Blueberry Yield Predictor is a web application powered by machine learning, designed to forecast wild blueberry yields. This project combines a machine learning model with a Flask-based interface, providing farmers, researchers, and agricultural experts with an accessible and effective tool for yield prediction. The objective is to support informed decision-making by analyzing key environmental and pollination factors that influence blueberry production.

Users can enter variables such as temperature ranges, rainy days, and pollinator activities (e.g., honeybee counts). These inputs are processed by the backend and fed into a machine learning model trained on agricultural data. The model utilizes algorithms like linear regression and random forests and has been fine-tuned to ensure accurate predictions by identifying patterns in the data.

A significant challenge in the project was managing the variability of environmental conditions, such as weather fluctuations and pollinator behavior. By incorporating a wide range of input features, the model adapts to diverse scenarios, offering reliable predictions.

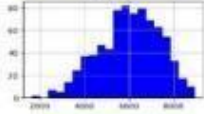
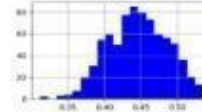
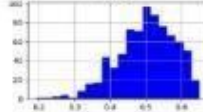
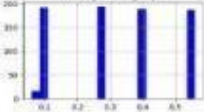
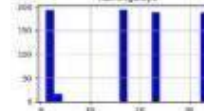
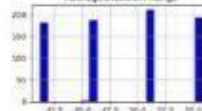
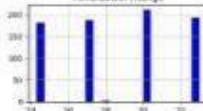
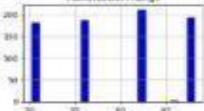
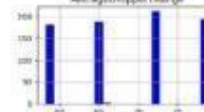



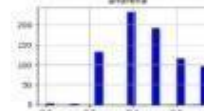
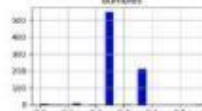

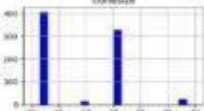
The application's interface, developed using the Flask web framework, provides a simple and user-friendly experience. Users can enter data through easy-to-use forms, and the system generates quick yield predictions, enabling better agricultural planning and resource optimization.

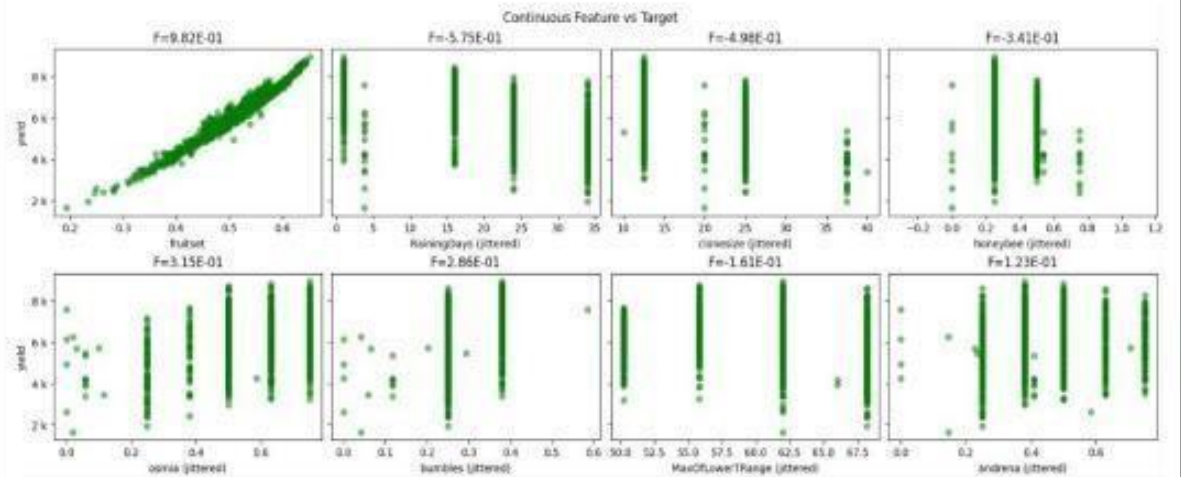
In summary, the kumaBlueberry Yield Predictor demonstrates the power of machine learning in agriculture by enabling precise, data-driven decisions. Through this integration of advanced technology, the project offers a practical tool to enhance the efficiency and sustainability of blueberry farming.

1. DATA COLLECTION AND PREPROCESSING

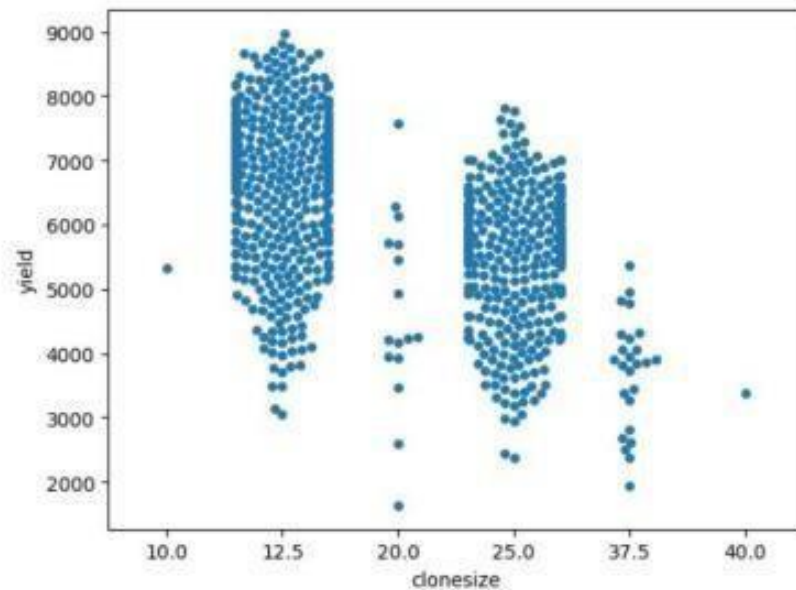
2.1 Data Exploration and Preprocessing

Identifying data sources, assessing quality issues like missing values and duplicates, and implementing resolution plans to ensure accurate and reliable analysis.

Section	Description																																																																																										
Data Overview	<table><thead><tr><th></th><th>clonesize</th><th>honeybee</th><th>bumbles</th><th>andrena</th><th>osmia</th><th>MaxOfUpperTrange</th><th>MinOfUpperTrange</th><th>AverageOfUpperTrange</th><th>MaxOfLowerTrange</th></tr></thead><tbody><tr><td>count</td><td>777.000000</td><td>777.000000</td><td>777.000000</td><td>777.000000</td><td>777.000000</td><td>777.000000</td><td>777.000000</td><td>777.000000</td><td>777.000000</td></tr><tr><td>mean</td><td>18.767696</td><td>0.417133</td><td>0.282389</td><td>0.468817</td><td>0.562062</td><td>82.277091</td><td>49.700515</td><td>68.723037</td><td>59.309395</td></tr><tr><td>std</td><td>6.999063</td><td>0.978904</td><td>0.066343</td><td>0.161052</td><td>0.169119</td><td>9.193745</td><td>5.595769</td><td>7.676984</td><td>6.647760</td></tr><tr><td>min</td><td>10.000000</td><td>0.000000</td><td>0.000000</td><td>0.000000</td><td>0.000000</td><td>69.700000</td><td>39.000000</td><td>58.200000</td><td>50.200000</td></tr><tr><td>25%</td><td>12.500000</td><td>0.250000</td><td>0.250000</td><td>0.380000</td><td>0.500000</td><td>77.400000</td><td>46.800000</td><td>64.700000</td><td>55.800000</td></tr><tr><td>50%</td><td>12.500000</td><td>0.250000</td><td>0.250000</td><td>0.500000</td><td>0.630000</td><td>86.000000</td><td>52.000000</td><td>71.900000</td><td>62.000000</td></tr><tr><td>75%</td><td>25.000000</td><td>0.500000</td><td>0.380000</td><td>0.630000</td><td>0.750000</td><td>89.000000</td><td>52.000000</td><td>71.900000</td><td>66.000000</td></tr><tr><td>max</td><td>40.000000</td><td>18.430000</td><td>0.585000</td><td>0.750000</td><td>0.750000</td><td>94.600000</td><td>57.200000</td><td>79.000000</td><td>68.200000</td></tr></tbody></table>		clonesize	honeybee	bumbles	andrena	osmia	MaxOfUpperTrange	MinOfUpperTrange	AverageOfUpperTrange	MaxOfLowerTrange	count	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	mean	18.767696	0.417133	0.282389	0.468817	0.562062	82.277091	49.700515	68.723037	59.309395	std	6.999063	0.978904	0.066343	0.161052	0.169119	9.193745	5.595769	7.676984	6.647760	min	10.000000	0.000000	0.000000	0.000000	0.000000	69.700000	39.000000	58.200000	50.200000	25%	12.500000	0.250000	0.250000	0.380000	0.500000	77.400000	46.800000	64.700000	55.800000	50%	12.500000	0.250000	0.250000	0.500000	0.630000	86.000000	52.000000	71.900000	62.000000	75%	25.000000	0.500000	0.380000	0.630000	0.750000	89.000000	52.000000	71.900000	66.000000	max	40.000000	18.430000	0.585000	0.750000	0.750000	94.600000	57.200000	79.000000	68.200000
		clonesize	honeybee	bumbles	andrena	osmia	MaxOfUpperTrange	MinOfUpperTrange	AverageOfUpperTrange	MaxOfLowerTrange																																																																																	
	count	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000																																																																																	
	mean	18.767696	0.417133	0.282389	0.468817	0.562062	82.277091	49.700515	68.723037	59.309395																																																																																	
	std	6.999063	0.978904	0.066343	0.161052	0.169119	9.193745	5.595769	7.676984	6.647760																																																																																	
	min	10.000000	0.000000	0.000000	0.000000	0.000000	69.700000	39.000000	58.200000	50.200000																																																																																	
	25%	12.500000	0.250000	0.250000	0.380000	0.500000	77.400000	46.800000	64.700000	55.800000																																																																																	
	50%	12.500000	0.250000	0.250000	0.500000	0.630000	86.000000	52.000000	71.900000	62.000000																																																																																	
	75%	25.000000	0.500000	0.380000	0.630000	0.750000	89.000000	52.000000	71.900000	66.000000																																																																																	
max	40.000000	18.430000	0.585000	0.750000	0.750000	94.600000	57.200000	79.000000	68.200000																																																																																		
Univariate Analysis	<div></div>																																																																																										

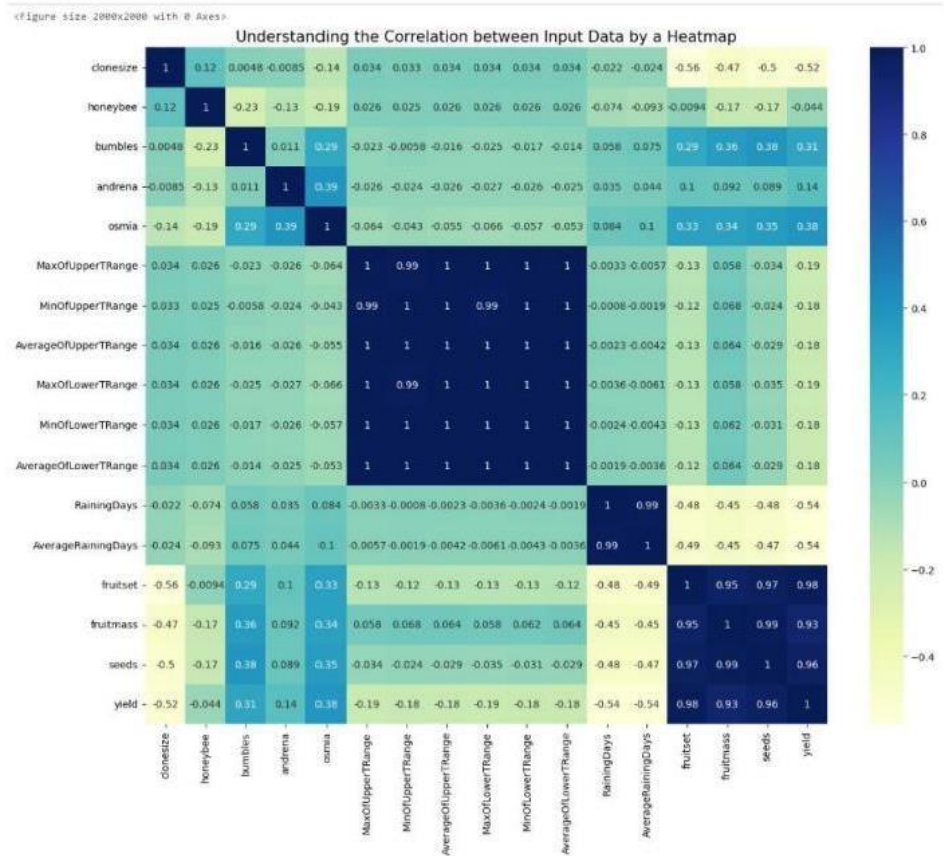


<Axes: xlabel='clonesize', ylabel='yield'>



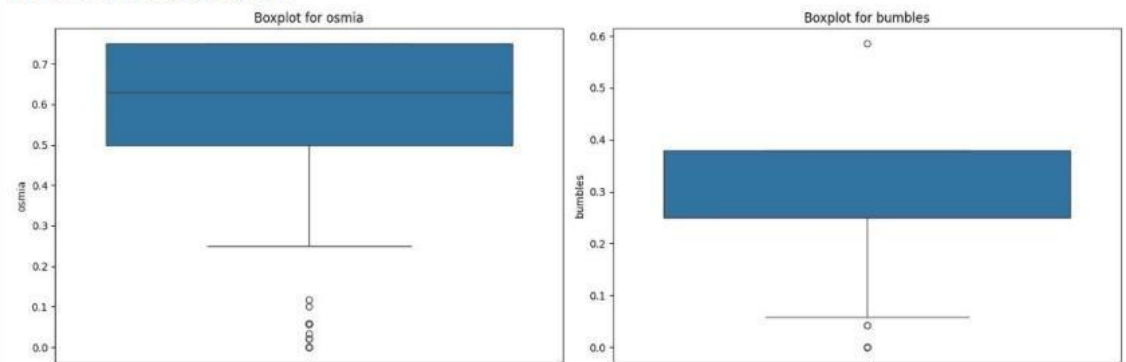
Bivariate Analysis

Multivariate Analysis

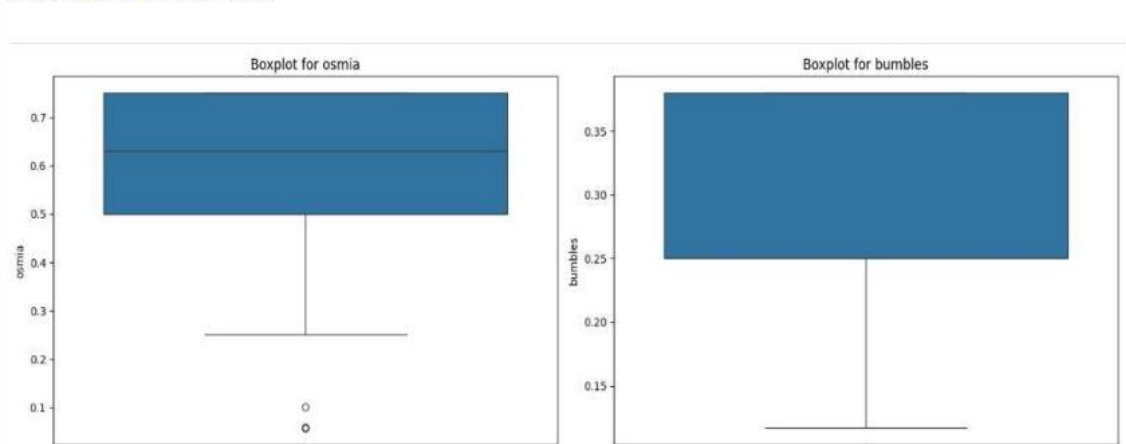


Outliers and Anomalies

BEFORE REMOVAL



AFTER REMOVAL



Data Preprocessing Code Screenshots

Loading Data

```
df = pd.read_csv('WildBlueberryPollinationSimulationData (1).csv')
df
```

	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	
1	1	37.5	0.750	0.250	0.250	0.250	86.0	52.0	71.9	62.0	
2	2	37.5	0.750	0.250	0.250	0.250	94.6	57.2	79.0	68.2	
3	3	37.5	0.750	0.250	0.250	0.250	94.6	57.2	79.0	68.2	
4	4	37.5	0.750	0.250	0.250	0.250	86.0	52.0	71.9	62.0	
...
772	772	10.0	0.537	0.117	0.409	0.058	86.0	52.0	71.9	62.0	
773	773	40.0	0.537	0.117	0.409	0.058	86.0	52.0	71.9	62.0	
774	774	20.0	0.537	0.117	0.409	0.058	86.0	52.0	71.9	62.0	

Handling Missing Data

No missing values in the dataset –

```
df.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

Removing outliers –

```
from scipy import stats
# Removing outliers
z_scores = np.abs(stats.zscore(df.select_dtypes(include=[np.number])))
threshold = 3
outliers = (z_scores > threshold).any(axis=1)
new_df = df[~outliers]
num_outliers_removed = outliers.sum()
print(f"Number of outliers removed: {num_outliers_removed}")
```

Number of outliers removed: 13

Feature Engineering

Removing unwanted columns after visualizing the dataset –

```
new_df = new_df.drop(columns=['bumbles', 'fruitmass', 'AverageRainingDays', 'fruitset', 'MaxOfUpperTRange', 'MaxOfLowerTRange', 'MinOfLowerTRange'])
new_df.head()
```

ALL THE ABOVE COLUMNS HAVE HIGH CORRELATION WITH OTHER COLUMNS, SO THEY ARE BEING REMOVED

	clonesize	honeybee	andrena	osmia	MinOfUpperTRange	AverageOfUpperTRange	AverageOfLowerTRange	RainingDays	seeds	yield
0	37.5	0.75	0.25	0.25	52.0	71.9	50.8	16.0	31.678898	3813.165795
1	37.5	0.75	0.25	0.25	52.0	71.9	50.8	1.0	33.449385	4947.605663
2	37.5	0.75	0.25	0.25	57.2	79.0	55.9	16.0	30.546306	3866.798965
3	37.5	0.75	0.25	0.25	57.2	79.0	55.9	1.0	31.562586	4303.943030
4	37.5	0.75	0.25	0.25	52.0	71.9	50.8	24.0	28.873714	3436.493543

Table-1 : Data exploration and preprocessing

3) DATA QUALITY REPORT

The Data Quality Report will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

Data Source	Data Quality Issue	Severity	Resolution Plan
Kaggle Dataset	Outliers present in the dataset	Moderate	z-scores are used to standardize the data and effectively detect and remove extreme values that may distort model performance.
Kaggle Dataset	Redundant columns in the dataset	Moderate	Through data visualizations like bar graphs, histograms and correlation heatmaps, columns with high correlation are recognized and removed.

Table-2 : Data quality

4) MODEL DEVELOPMENT PHASE

4.1 Feature Selection Report

This report outlines the process and rationale behind the selection and removal of features for the Wild Blueberry Pollination Simulation dataset. The goal of feature selection is to improve model performance and interpretability by focusing on the most impactful variables. Through data visualization and analysis, we identified key features that significantly contribute to predicting the target outcomes while removing redundant or non-informative variables. The following sections detail the reasons for selecting or removing each feature, ensuring a robust and efficient model.

Feature	Description	Selected (Yes/No)	Reasoning
Row#	Unique identifier for each row of the dataset	No	For predicting the yield value, Row number is not required.
clonesize	The size of the blueberry clone being studied.	Yes	Influences yield and fruit production due to varying pollination dynamics in different clone sizes.
honeybee	The density or activity level of honeybee pollinators.	Yes	Key pollinator affecting yield and seed production.
bumbles	The density or activity level of bumblebee pollinators.	No	Likely redundant with other bee density variables or showed little correlation with the target variable.

andrena	The density or activity level of Andrena bee pollinators.	Yes	Significant pollinator, adding diversity in understanding pollination effects.
osmia	The density or activity level of Osmia bee pollinators.	Yes	Important pollinator, contributing to overall pollination success.
MaxOfUpperTRange	The maximum temperature of the upper temperature range recorded.	No	Little variation or weak correlation with outcomes; redundant with other temperature measures.
MinOfUpperTRange	The minimum temperature of the upper temperature range recorded.	Yes	Influences pollinator activity and plant growth under extreme temperatures.
AverageOfUpperTRange	The average temperature of the upper temperature range.	Yes	Impacts pollinator behavior and plant physiology under consistent temperatures.
MaxOfLowerTRange	The maximum temperature of the lower temperature range recorded.	No	Limited variation or significance; possibly redundant with other temperature features.
MinOfLowerTRange	The minimum temperature of the lower temperature range recorded.	No	Showed little correlation with outcomes; redundancy with other temperature variables.
AverageOfLowerTRange	The average temperature of the lower temperature range.	Yes	Important for understanding cooler conditions affecting plant stress and pollination.

RainingDays	The total number of raining days.	Yes	Affects pollinator activity and plant health; significant for crop success.
AverageRainingDays	The average number of raining days.	No	Non-informative or redundant with total raining days (RainingDays).
fruitset	The proportion of flowers that set fruit.	No	Redundant with other yield-related measures.
fruitmass	The mass of the fruit produced.	No	Highly correlated with yield, making it redundant.
seeds	The number of seeds per fruit.	Yes	Direct measure of successful pollination and fruit development.

Table-3: Feature selection

4.2 Initial Model Training Code, Model Validation and Evaluation Report

This report outlines the process and results of developing and validating regression models to predict the outcomes for the Wild Blueberry Pollination Simulation dataset. The goal of this phase is to build accurate and reliable models by selecting relevant features, evaluating various algorithms, and fine-tuning hyperparameters. This report includes detailed evaluation metrics, visualization, and insights into the performance of each model, ensuring a comprehensive understanding of their predictive capabilities.

Initial Model Training Code:

```
# Importing and building Linear Regression model
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
model.score(X_test, y_test)

y_preds = model.predict(X_test)

print("Regression metrics on the test set")
print(f"R2 score: {r2_score(y_test, y_preds)}")
print(f"MAE: {mean_absolute_error(y_test, y_preds)}")
print(f"MSE: {mean_squared_error(y_test, y_preds)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_preds))}")
```

```
# Importing and building the Random forest regressor model
from sklearn.ensemble import RandomForestRegressor

model2 = RandomForestRegressor()
model2.fit(X_train, y_train)

y_preds2 = model2.predict(X_test)

print("Regression metrics on the test set")
print(f"R2 score: {r2_score(y_test, y_preds2)}")
print(f"MAE: {mean_absolute_error(y_test, y_preds2)}")
print(f"MSE: {mean_squared_error(y_test, y_preds2)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_preds2))}")
```



```

# Importing and building Decision tree regressor model
from sklearn.tree import DecisionTreeRegressor

model3 = DecisionTreeRegressor()
model3.fit(X_train,y_train)
y_preds3 = model3.predict(X_test)

print("Regression metrics on the test set")
print(f"R2 score: {r2_score(y_test, y_preds3)}")
print(f"MAE: {mean_absolute_error(y_test,y_preds3)}")
print(f"MSE: {mean_squared_error(y_test, y_preds3)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_preds3))}")

# Importing and building XGB Regressor model
from xgboost import XGBRegressor
model4 = XGBRegressor()
model4.fit(X_train,y_train)
model4.score(X_test,y_test)

y_preds4 = model4.predict(X_test)

print("Regression metrics on the test set")
print(f"R2 score: {r2_score(y_test, y_preds4)}")
print(f"MAE: {mean_absolute_error(y_test,y_preds4)}")
print(f"MSE: {mean_squared_error(y_test, y_preds4)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_preds4))}")

```

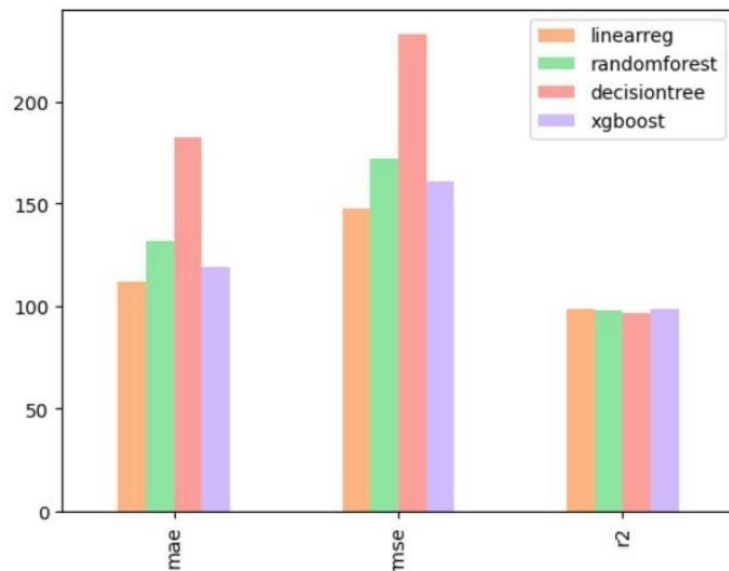
4.3 Model Validation and Evaluation Report:

Model	Mean Absolute Error (MAE)	Mean Squared Error (MSE)	R2 Score	Screenshots of the evaluation report
Linear Regression	111.5204986497179	21684.627147497344	0.9850297685259484	<pre> print("Regression metrics on the test set") print(f"R2 score: {r2_score(y_test, y_preds)}") print(f"MAE: {mean_absolute_error(y_test,y_preds)}") print(f"MSE: {mean_squared_error(y_test, y_preds)}") print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_preds))}") </pre> <p>Regression metrics on the test set R2 score: 0.9850297685259484 MAE: 111.5204986497179 MSE: 21684.627147497344 RMSE: 147.25701052071287</p>

Random Forest Regressor	133.30059127843145	30238.378584205282	0.9791245879522628	<pre>print("Regression metrics on the test set") print(f"R2 score: {r2_score(y_test, y_preds2)}") print(f"MAE: {mean_absolute_error(y_test,y_preds2)}") print(f"MSE: {mean_squared_error(y_test, y_preds2)}") print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_preds2))}")</pre> <p>Regression metrics on the test set R2 score: 0.9791245879522628 MAE: 133.30059127843145 MSE: 30238.378584205282 RMSE: 173.8918588784572</p>
Decision Tree Regressor	177.92750329411768	51633.51043756918	0.9643542128803687	<pre>print("Regression metrics on the test set") print(f"R2 score: {r2_score(y_test, y_preds3)}") print(f"MAE: {mean_absolute_error(y_test,y_preds3)}") print(f"MSE: {mean_squared_error(y_test, y_preds3)}") print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_preds3))}")</pre> <p>Regression metrics on the test set R2 score: 0.9643542128803687 MAE: 177.92750329411768 MSE: 51633.51043756918 RMSE: 227.23008259816564</p>
XGB Regressor	119.19604783486517	25823.161181161377	0.982172684010463	<pre>print("Regression metrics on the test set") print(f"R2 score: {r2_score(y_test, y_preds4)}") print(f"MAE: {mean_absolute_error(y_test,y_preds4)}") print(f"MSE: {mean_squared_error(y_test, y_preds4)}") print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_preds4))}")</pre> <p>Regression metrics on the test set R2 score: 0.982172684010463 MAE: 119.19604783486517 MSE: 25823.161181161377 RMSE: 160.69586547625107</p>

Table -4: model validation and evaluation

This bar chart compares the performance of four regression models—Linear Regression, Random Forest, Decision Tree, and XGBoost—using three evaluation metrics: Mean AbsoluteError (MAE), Mean Squared Error (MSE), and R-squared (R^2).



4.4 Model Selection Report

In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including MSE, MAE or R2 score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model	Description	Hyperparameters	Performance Metric (e.g., MSE, MAE, R2 Score)
Linear Regression	A simple model that assumes a linear relationship between the input variables (features) and the output variable (target). It fits a straight line to the data.	-	R2 score = 0.985
Random Forest Regressor	An ensemble learning method that constructs multiple decision trees during training and outputs the average prediction of the individual trees. It reduces overfitting and improves accuracy	-	R2 score = 0.979
Decision Tree Regressor	A non-linear model that splits the data into subsets based on feature values, creating a tree-like structure. Each leaf represents a predicted outcome.	-	R2 score = 0.961
XGB Regressor	An advanced ensemble technique that uses gradient boosting on decision trees. It iteratively improves model performance by minimizing errors of previous models, known for its high accuracy and efficiency.	-	R2 score = 0.982

Table-5: model selection

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.

5.1 Hyperparameter Tuning Documentation

Model	Tuned Hyperparameters	Optimal Values
Linear Regression	-	Regression metrics on the test set R2 score: 0.9850297685259484 MAE: 111.5204986497179 MSE: 21684.627147497344 RMSE: 147.25701052071287
Random Forest Regressor	<pre>from sklearn.model_selection import GridSearchCV param_grid = { 'n_estimators': [100, 200, 300], 'max_features': ['auto', 'sqrt', 'log2'], 'max_depth': [10, 20, 30, None], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'bootstrap': [True, False] } rf = RandomForestRegressor() grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3, n_jobs=-1, verbose=2)</pre>	<p>Fitting 3 folds for each of 648 candidates, totalling 1944 fits</p> <p>Best Parameters: {'bootstrap': False, 'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}</p> <p>Mean Squared Error: 23036.512805030303</p> <p>R-squared: 0.9840964787311984</p>
DecisionTree Regressor	<pre>param_grid = { 'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'max_features': [None, 'auto', 'sqrt', 'log2'] } dt = DecisionTreeRegressor() grid_search = GridSearchCV(estimator=dt, param_grid=param_grid, cv=3, n_jobs=-1, verbose=2)</pre>	<p>Fitting 3 folds for each of 144 candidates, totalling 432 fits</p> <p>Best Parameters: {'max_depth': 30, 'max_features': None, 'min_samples_leaf': 4, 'min_samples_split': 2}</p> <p>Mean Squared Error: 39945.42992962877</p> <p>R-squared: 0.9724232135369657</p>
XGB Regressor	<pre>param_grid = { 'n_estimators': [100, 200, 300], 'learning_rate': [0.01, 0.1, 0.2], 'max_depth': [3, 6, 9], 'subsample': [0.6, 0.8, 1.0], 'colsample_bytree': [0.6, 0.8, 1.0] } xgb = XGBRegressor() grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid, cv=3, n_jobs=-1, verbose=2)</pre>	<p>Fitting 3 folds for each of 243 candidates, totalling 729 fits</p> <p>Best Parameters: {'colsample_bytree': 1.0, 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 200, 'subsample': 0.6}</p> <p>Mean Squared Error: 19206.509150671092</p> <p>R-squared: 0.9867405657547946</p>

5.2 Performance Metrics Comparison Report

Model	Baseline Metric	Optimized Metric
Linear Regression	Regression metrics on the test set R2 score: 0.9850297685259484 MAE: 111.5204986497179 MSE: 21684.627147497344 RMSE: 147.25701052071287	-
Random Forest Regressor	Regression metrics on the test set R2 score: 0.9794220417457439 MAE: 132.1608756852944 MSE: 29807.511859370235 RMSE: 172.64852116183977	Regression metrics on the test set R2 score: 0.9840964787311984 MAE: 120.52941809457525 MSE: 23036.512805030303 RMSE: 151.7778402963697
Decision Tree Regressor	Regression metrics on the test set R2 score: 0.9621834493005946 MAE: 181.75958724836605 MSE: 54777.89727850463 RMSE: 234.04678437975736	Regression metrics on the test set R2 score: 0.9724232135369657 MAE: 155.86761325904146 MSE: 39945.42992962877 RMSE: 199.86352826273423
XGB Regressor	Regression metrics on the test set R2 score: 0.982172684010463 MAE: 119.19604783486517 MSE: 25823.161181161377 RMSE: 160.69586547625107	Regression metrics on the test set R2 score: 0.9867405657547946 MAE: 108.15756897906451 MSE: 19206.509150671092 RMSE: 138.5875504894689

Table-6: performance

5.3 Final Model Selection Justification

Final Model	Reasoning
XGB Regressor	The XGB Regressor model was selected for its superior performance, exhibiting the highest R2 score 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

6) WEB APPLICATION USING FLASK

6.1 Model Frontend Implementation

The Model Frontend Implementation Phase involves creating front end implementations so that real time users can interact with it without having any prior ML or DL knowledge. It is a Flask Framework based frontend implementation having a main folder called Flask. Flask folder has multiple sub-folders called templates and models and a file called app.py, templates folder has two files called index.html, predict.html and also a background image, models folder has the model file named model0.pkl. While execution we shall go to the Flask folder directory through anaconda prompt and type Python app.py followed by enter.

6.2 Images of the Web Application



output screen - 1

Predict Wild Blueberry Yield

127.0.0.1:5000

Honeybee:

Andrena:

Osmia:

Min Of Upper T Range:

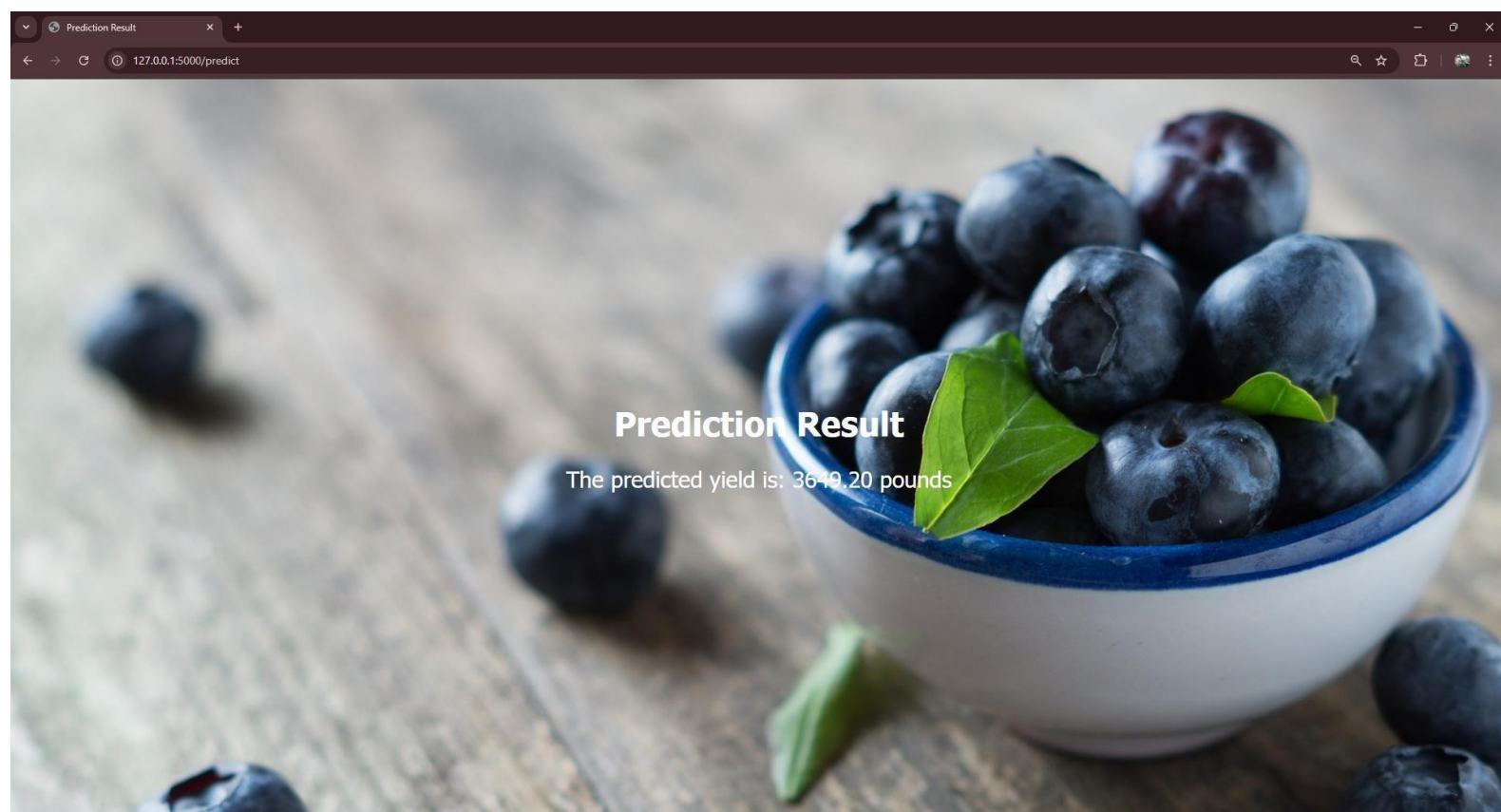
Average Of Upper T Range:

Average Of Lower T Range:

Raining Days:

Seeds:

output screen - 2



prediction screen

7) CONCLUSION

In conclusion, the Blueberry Yield Predictor project demonstrates the effective use of machine learning technology to solve a practical agricultural challenge—predicting the yield of wild blueberry crops. By integrating a Flask-based web application with a trained machine learning model, this project successfully offers a tool that can help farmers and agricultural professionals make informed decisions based on critical environmental and pollination factors. The user-friendly interface allows for quick and easy input of relevant data, and the backend processes this information to provide real-time predictions.

The development process highlighted the importance of careful dataset selection, preprocessing, and model tuning to ensure high accuracy in predictions. Despite challenges such as the variability of natural conditions, the model was designed to account for a wide range of factors, making it a reliable tool across different scenarios. The use of machine learning allowed the system to identify trends and correlations that might not be immediately apparent through traditional analysis.

This project showcases the potential of applying machine learning to agricultural processes, helping to optimize resources and improve crop management practices. As technology continues to evolve, further improvements to the model's accuracy and functionality can be made, such as incorporating real-time environmental data and extending the model to other crops.

Overall, the Blueberry Yield Predictor project represents a significant step toward the integration of modern technology into agriculture, offering a scalable solution that can positively impact crop yield prediction and contribute to more sustainable farming practices. It serves as an example of how data-driven insights can transform traditional industries and create opportunities for innovation.

REFERENCES

- [1] B. Bell and L. Kaiser, "Machine Learning in Agriculture: Applications and Algorithms," Springer, 2021.
- [2] B. Bengfort, R. Bilbro, and T. Ojeda, "Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning," O'Reilly Media, 2018.
- [3] C. M. Bishop, "Pattern Recognition and Machine Learning," Springer, 2006.
- [4] P. Flach, "Machine Learning: The Art and Science of Algorithms that Make Sense of Data," Cambridge University Press, 2012.
- [5] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, 2016.
- [6] M. Grinberg, "Flask Web Development: Developing Web Applications with Python," O'Reilly Media, 2018.

Additional Resources :

https://scikit-learn.org/stable/supervised_learning.html

<https://scikit-learn.org/stable/modules/tree.html>

<https://scikit-learn.org/stable/modules/ensemble.html>

<https://online.stat.psu.edu/stat462/node/89/>

<https://www.servicenow.com/community/intelligence-ml-articles/tuning-predictive-intelligence-models-part-5-regression/ta-p/2470134>

<https://stackoverflow.com/questions/60454618/is-it-possible-to-tune-the-linear-regression-hyperparameter-in-sklearn>

<https://www.kaggle.com/learn/feature-engineering>

<https://www.kaggle.com/datasets/saurabhshahane/wild-blueberry-yield-prediction>

APPENDIX I

SOURCE CODE

Flask Application -

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Predict Wild Blueberry Yield</title>
  <style>
    body {
      background-image: url("{% url_for('static', filename='blueberry.jpg') %}"); /* Background
image */
      background-size: cover; /* Cover the entire page */
      background-position: center; /* Center the image */
      font-family: Tahoma, sans-serif;
      display: flex;
      justify-content: flex-start; /* Align content to the left */
      align-items: center;
      height: 100vh;
      margin: 0;
      padding: 0;
      color: white; /* Changed text color to white */
    }

    .form-container {
      background-color: rgba(37, 37, 99, 0.5); /* Semi-transparent grey background for the form
container */
      padding: 30px;
      border-radius: 30px;
      box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
      width: 80%;
      max-width: 600px;
      overflow-y: auto;
      max-height: calc(100vh - 150px); /* Slightly reduced height */
      color: white; /* Changed text color to white */
      margin-left: 50px; /* Move form 50px to the left */
    }

    .form-container h1 {
      text-align: center;
```

```

        color: white; /* Heading color */
        font-size: 3rem; /* Increased font size */
        margin-bottom: 20px; /* Added some bottom margin */
    }

    .form-container label {
        display: block;
        margin-bottom: 12px;
        font-weight: bold;
        color: white; /* Label text color */
    }

    .form-container input[type="text"],
    .form-container input[type="date"],
    .form-container input[type="submit"] {
        width: calc(100% - 24px);
        height: 60px; /* Reduced input height */
        padding: 12px;
        margin-bottom: 15px;
        border: 4px solid white;
        border-radius: 10px;
        transition: all 0.3s ease;
        font-size: 1.2rem; /* Increased font size */
        box-sizing: border-box;
        background-color: rgba(0, 0, 0, 0.5); /* Darkened background color */
        color: white; /* Text color */
    }

    .form-container input[type="submit"] {
        background-color: #4CAF50;
        color: white;
        cursor: pointer;
        font-size: 1.3rem; /* Increased font size */
        font-weight: bold;
        border: none;
        border-radius: 10px;
        padding: 14px 24px; /* Increased padding */
    }

    .form-container input[type="submit"]:hover {
        background-color: #45a049;
    }
</style>
</head>
<body>

```



```

<div class="Form-container">
  <h1>BERRY_PREDICTOR.IO</h1>
  <form action="/predict" method="post">
    <label for="clonesize">Clonesize:</label>
    <input type="text" id="clonesize" name="clonesize"><br>
    <label for="honeybee">Honeybee:</label>
    <input type="text" id="honeybee" name="honeybee"><br>
    <label for="andrena">Andrena:</label>
    <input type="text" id="andrena" name="andrena"><br>
    <label for="osmia">Osmia:</label>
    <input type="text" id="osmia" name="osmia"><br>
    <label for="MinOfUpperTRange">Min Of Upper T Range:</label>
    <input type="text" id="MinOfUpperTRange" name="MinOfUpperTRange"><br>
    <label for="AverageOfUpperTRange">Average Of Upper T Range:</label>
    <input type="text" id="AverageOfUpperTRange" name="AverageOfUpperTRange"><br>
    <label for="AverageOfLowerTRange">Average Of Lower T Range:</label>
    <input type="text" id="AverageOfLowerTRange" name="AverageOfLowerTRange"><br>
    <label for="RainingDays">Raining Days:</label>
    <input type="text" id="RainingDays" name="RainingDays"><br>
    <label for="seeds">Seeds:</label>
    <input type="text" id="seeds" name="seeds"><br>
    <input type="submit" value="Predict">
  </form>
</div>
</body>
</html>

```

predict.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Prediction Result</title>
  <style>
    body {
      /* background-color: #000000; */
      background-image: url("{ url_for('static', filename='blueberry2.jpg') }");
      background-repeat: no-repeat;
      background-size: cover;
      background-position: center;
      font-family: Tahoma, sans-serif;
      display: flex;
    }
  </style>

```

```

        justify-content: center; /* Center content horizontally */
        align-items: center;
        height: 100vh;
        margin: 0;
        padding: 0;
        color: white; /* Changed text color to white */
    }

    .result-container {
        text-align: center;
    }

    .result-container h1 {
        font-size: 3rem;
        margin-bottom: 20px;
    }

    .result-container p {
        font-size: 2rem;
    }
</style>
</head>
<body>
    <div class="result-container">
        <h1>Prediction Result</h1>
        <p>The predicted yield is: {{ '%.2f' % prediction }} pounds</p>
    </div>
</body>
</html>

```

app.py

```

from flask import Flask, request, render_template
import joblib
import numpy as np

app = Flask(__name__)

# Load the model
model = joblib.load('model0.pkl')

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

```

```

@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        # Get the data from the form
        data = [
            float(request.form['clonesize']),
            float(request.form['honeybee']),
            float(request.form['andrena']),
            float(request.form['osmia']),
            float(request.form['MinOfUpperTRange']),
            float(request.form['AverageOfUpperTRange']),
            float(request.form['AverageOfLowerTRange']),
            float(request.form['RainingDays']),
            float(request.form['seeds'])
        ]

        # Convert to numpy array and reshape for the model
        data = np.array(data).reshape(1, -1)

        # Predict using the model
        prediction = model.predict(data)

        return render_template('predict.html', prediction=prediction[0])

if __name__ == '__main__':
    app.run(debug=True)

```