Maago – A Smart System for Mango Plantation Management

Project ID : 2023-309

IT20466008
Withanaarachchi S. P.

Specialization: Computer Systems and Network Engineering

Smart Watering System Enabled by IoT and Machine Learning

Introduction

Sri Lanka has a long and rich history of mango cultivation. The objective is to create a smart watering system based on IoT and Machine Learning to boost mango production and reduce water usage.

Research Question

- How to identify the water needs of the plantation provide water through a smart system using IoT and machine Learning technologies?

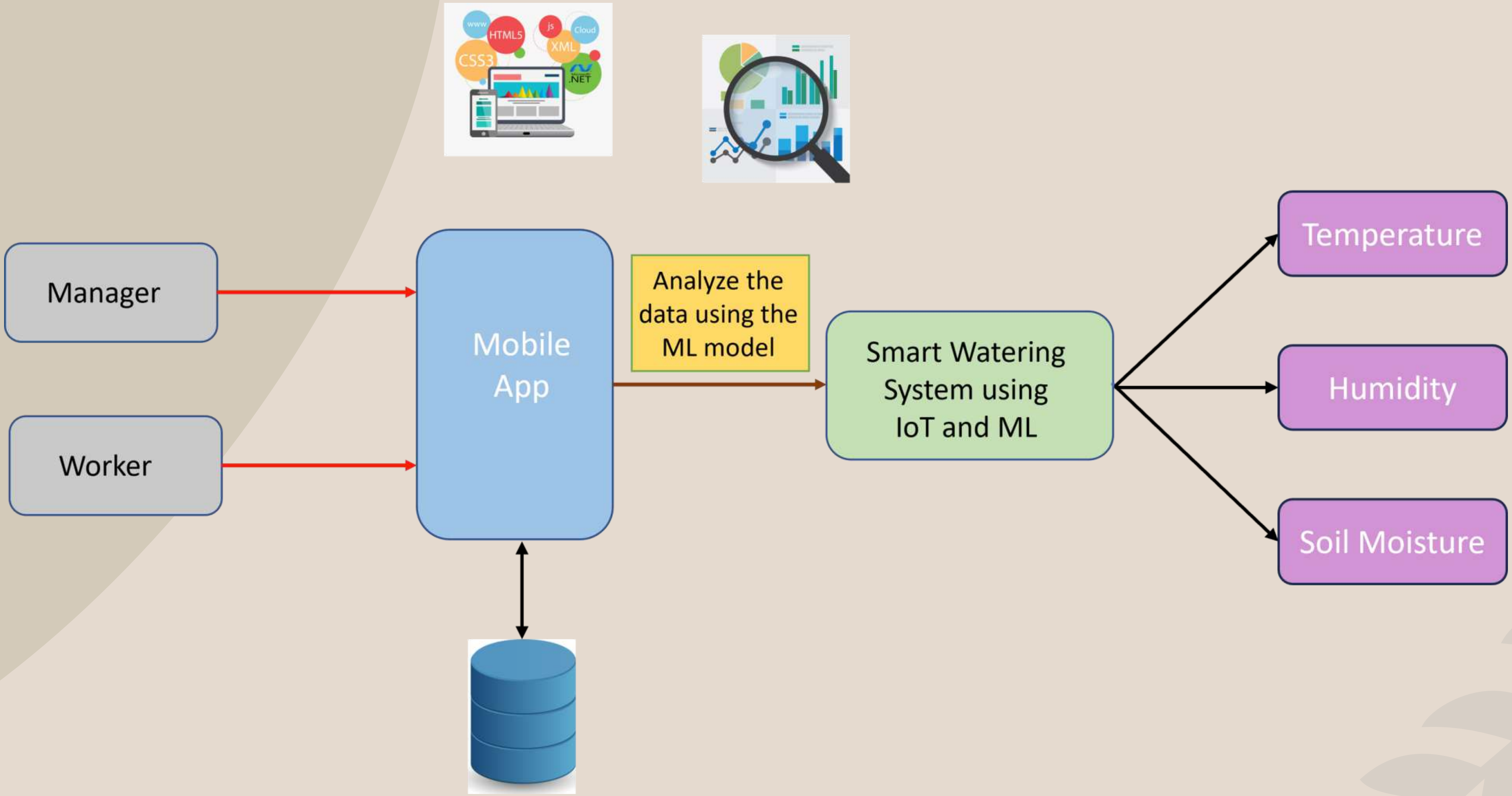
Objective

The main objective of this component is to implement a smart water management system that provides water to the plantation at the correct time and in the correct amount.

Methodology

- An IoT device is developed to measure the soil conditions. The device will be equipped with temperature, humidity, and soil moisture sensors.
- A machine learning model is developed to predict the water needed for the plantation according to the sensor readings. The IoT device and machine learning model will be integrated to create the proposed smart watering system.
- A mobile application is used to update the real-time data to the users.

System Diagram



Technologies

- Arduino
- Python
- C
- Google Colab
- Algorithms - Linear Regression Algorithm

Tools

- ESP32 module
- DHT-11 sensor
- Soil moisture sensor
- Relay Module

Completion of the IoT device

sketch_may23a | Arduino 1.8.13

File Edit Sketch Tools Help

sketch_may23a

```
////// Humidity & Temperature Monitoring sensor readings ////////
#include <DHT.h>
#define DHT_SENSOR_PIN  21 // ESP32 pin GIOP21 connected to DHT11 sensor
#define DHT_SENSOR_TYPE DHT11

////////////////////////////////// Moisture PIN Initialization ////////////////////////////////////////////
const int moisturePin = A0;             // moisteure sensor pin
float moisturePercentage;              //moisture reading


DHT dht_sensor(DHT_SENSOR_PIN, DHT_SENSOR_TYPE);

/// Define a struct to return values of Humidity and Temperature ///
struct HumidityTemperatureData {
  bool success;
  float humidity;
  float temperatureC;
  float temperatureF;
};

void setup() {
  Serial.begin(9600);
  dht_sensor.begin(); // initialize the DHT sensor
}

void loop() {
  HumidityTemperatureData data = readHumidityTemperature();
  if (data.success) {
    Serial.print("Humidity: ");
    Serial.print(data.humidity);
    Serial.print("%");

    Serial.print("  |  ");
```

sketch_may23a | Arduino 1.8.13

File Edit Sketch Tools Help

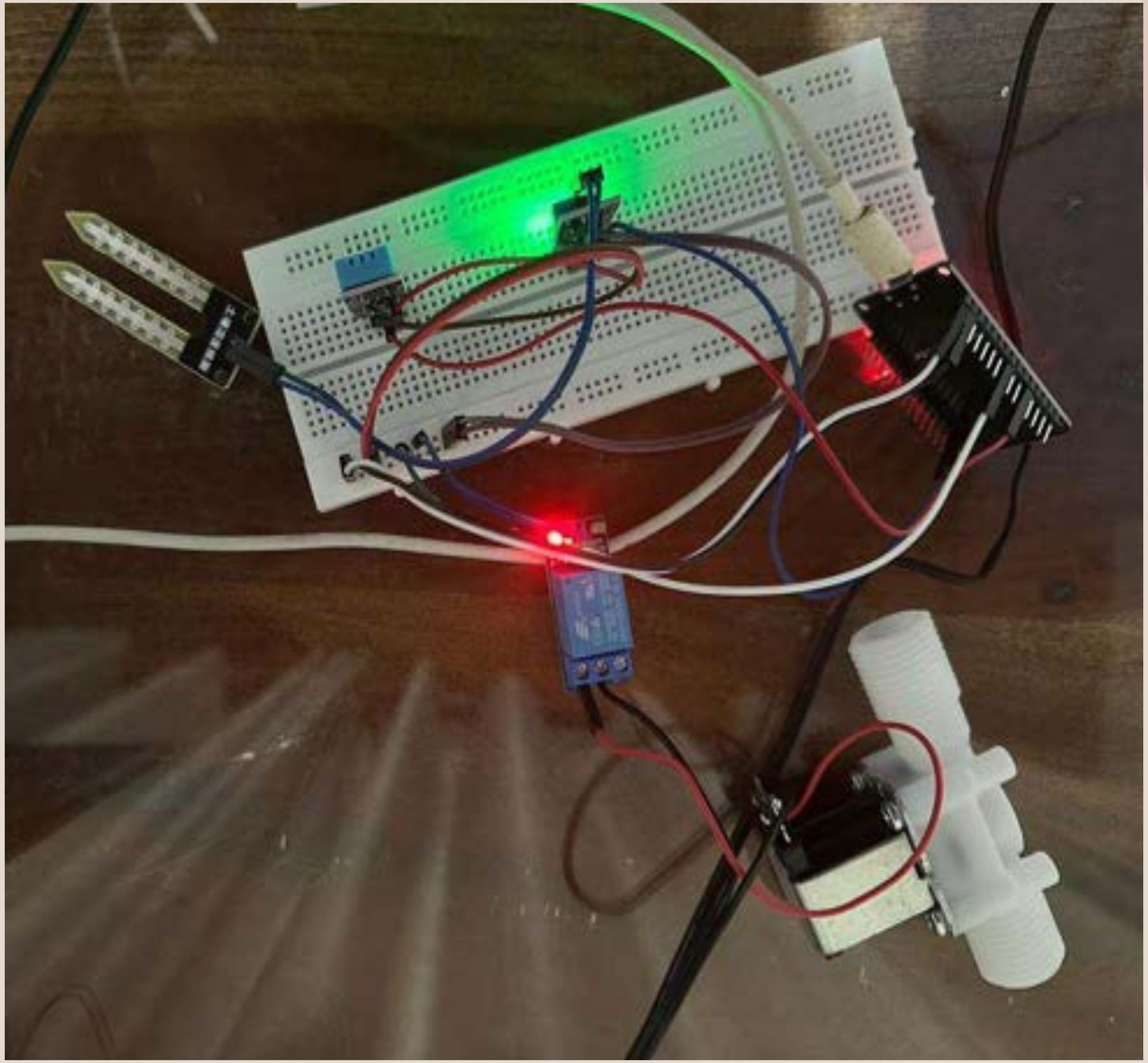
sketch_may23a

```
  // wait a 2 seconds between readings
  delay(2000);
}

float ReadMoisture(){
  Serial.println("Start Moisture Reading");
  moisturePercentage = ( 100 - ( (analogRead(moisturePin) / 1023.00) * 100 ) );
  Serial.print("Soil Moisture is  = ");
  Serial.print(moisturePercentage);
  Serial.println("%");
  return moisturePercentage;
  }

HumidityTemperatureData readHumidityTemperature() {
  HumidityTemperatureData data;
  data.humidity = dht_sensor.readHumidity();            // read humidity
  data.temperatureC = dht_sensor.readTemperature();    // read temperature in Celsius
  data.temperatureF = dht_sensor.readTemperature(true);    // read temperature in Fahrenheit

  // check whether the reading is successful or not
  bool isReadingSuccessful = !(isnan(data.temperatureC) || isnan(data.temperatureF) || isnan(data.humidity));
  data.success = isReadingSuccessful;

  if (!isReadingSuccessful) {
    data.humidity = 0.0;
    data.temperatureC = 0.0;
    data.temperatureF = 0.0;
  }

  return data;
}
```

Done compiling.

```
Sketch uses 268425 bytes (20%) of program storage space. Maximum is 1310720 bytes.
Global variables use 22552 bytes (6%) of dynamic memory, leaving 305128 bytes for local variables. Maximum is 327680 bytes.
```

83 ESP32 Dev Module, Disabled, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, Core 1, Core 1, None, Disabled on COM4



Completion of the ML model

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Assuming you have already created the dataframe 'df' with the provided data

# Split the data into features (X) and target variable (y)
X = df.drop('Amount of Water Needed (ml)', axis=1)
y = df['Amount of Water Needed (ml)']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the linear regression model
reg_model = LinearRegression()
reg_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = reg_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared Score:", r2)
```

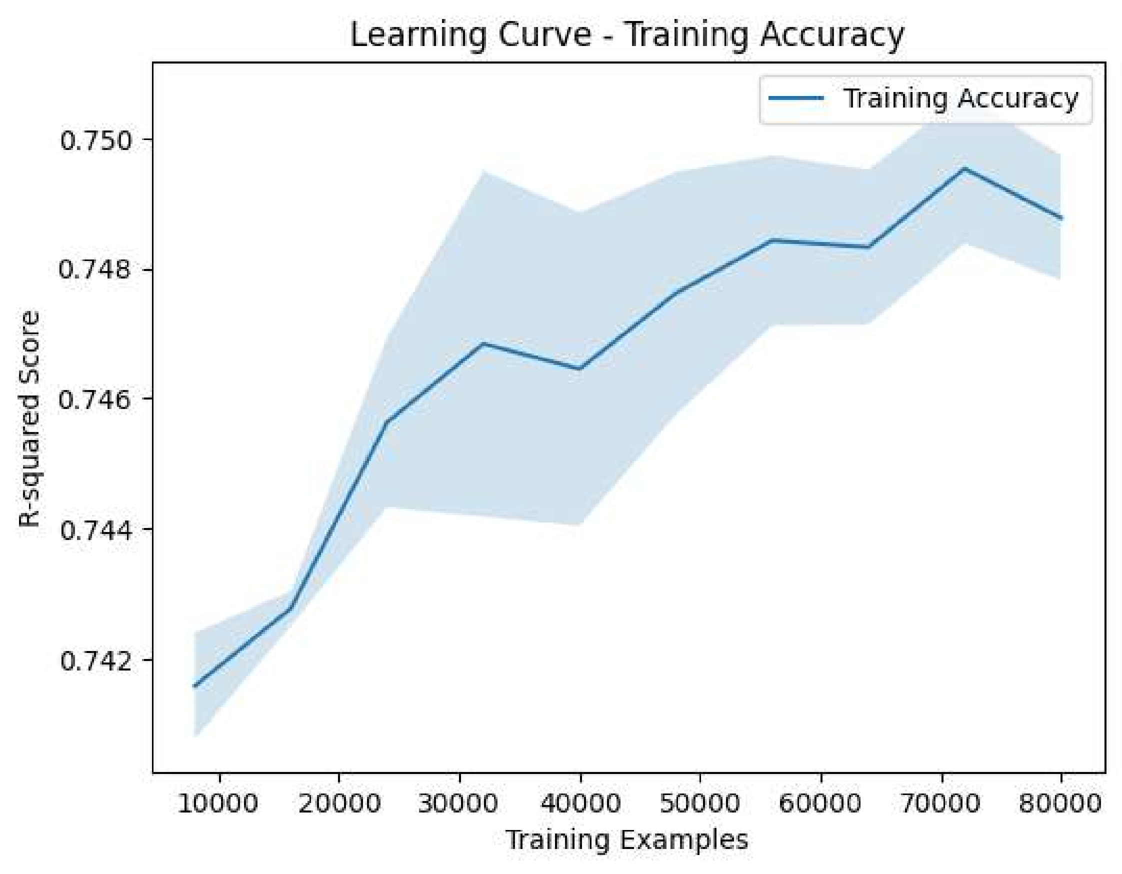
```
Mean Squared Error: 5.290136427517269
R-squared Score: 0.7462031399185647
```

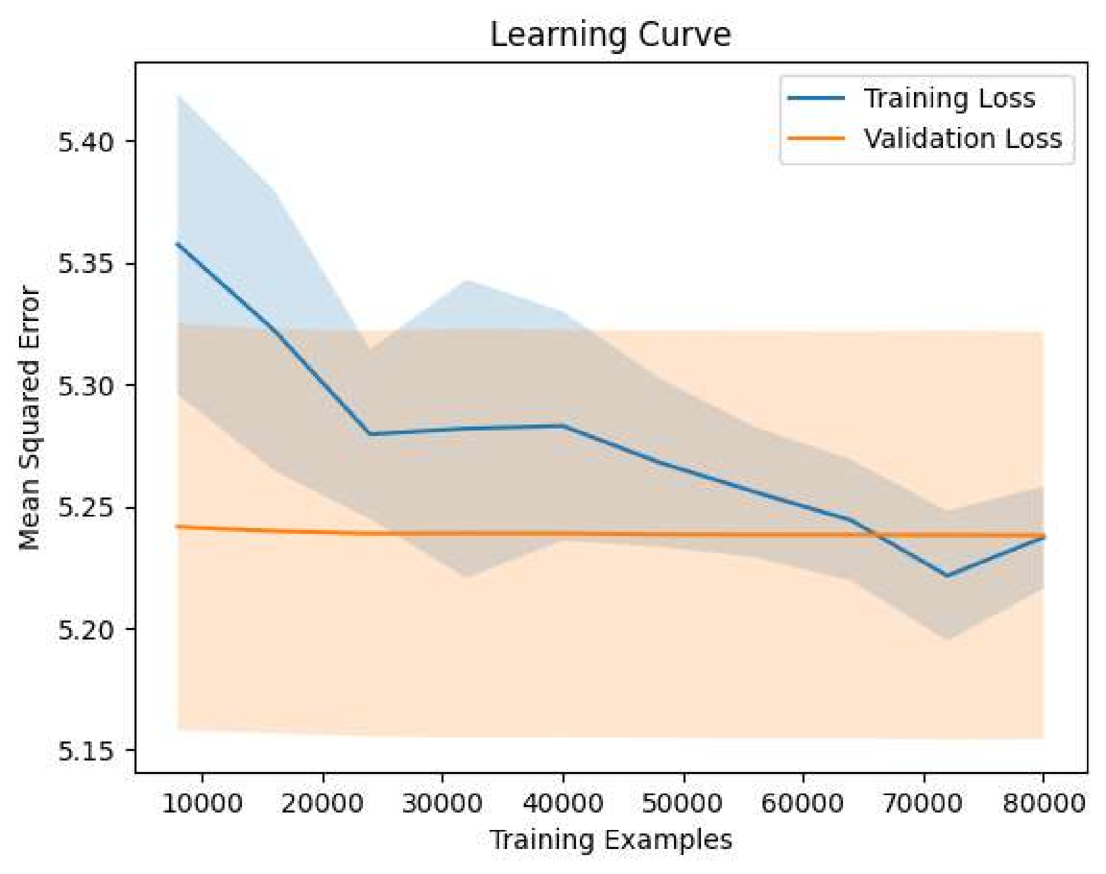
```
import pandas as pd
import numpy as np
```

	Humidity Level	Moisture Level	Temperature	Amount of Water Needed (ml)
0	34.186645	0.861426	30.204245	21.589984
1	62.073150	0.600127	36.866552	30.865282
2	60.894998	0.077513	28.794518	28.019393
3	54.746297	0.094907	25.593974	24.644907
4	27.272004	0.850259	34.522049	26.019462
...	...	...	...	...
99995	69.547929	0.507732	22.270037	17.192713
99996	96.436557	0.207808	25.964946	23.886867
99997	62.802012	0.788196	28.656678	20.774722
99998	69.531031	0.732225	31.624691	24.302440
99999	33.915464	0.924799	33.569151	24.321158

100000 rows × 4 columns

Completion of the ML model





Progress at the moment

- Implementation of the soil moisture sensor and temperature and humidity sensor
- Implementation the ML model to identify the water needed for the plantation
- Update the database with real-time data.
- Development of the mobile application

What's to be done

- Do further improvements in the mobile application
- Introduce the system to the clients and research how they interact.

IT20280260
Aksham M.Z.M

Specialization: Information Technology

Identify diseases using a scanning system

Introduction

- To implement a system to identify the disease on the mango leaves using the scanner
- Identifying disease on mango leaves is to facilitate early detection and
  control of plant diseases to increase the mango production

Research Question

- How to identify the mango disease and increase the production through a scanning system?

Methodology

- Machine learning-based image analysis:

  Method for spotting diseases of mango tree leaves is to analyze photographs of the leaves and look for patterns that correspond to various diseases. In order to construct a predictive model for disease identification, the system may need to be trained a large dataset of images of both healthy and diseased mango leaves.

Tools and Technologies

- Python
- Google Colab
- Pandas
- NumPy

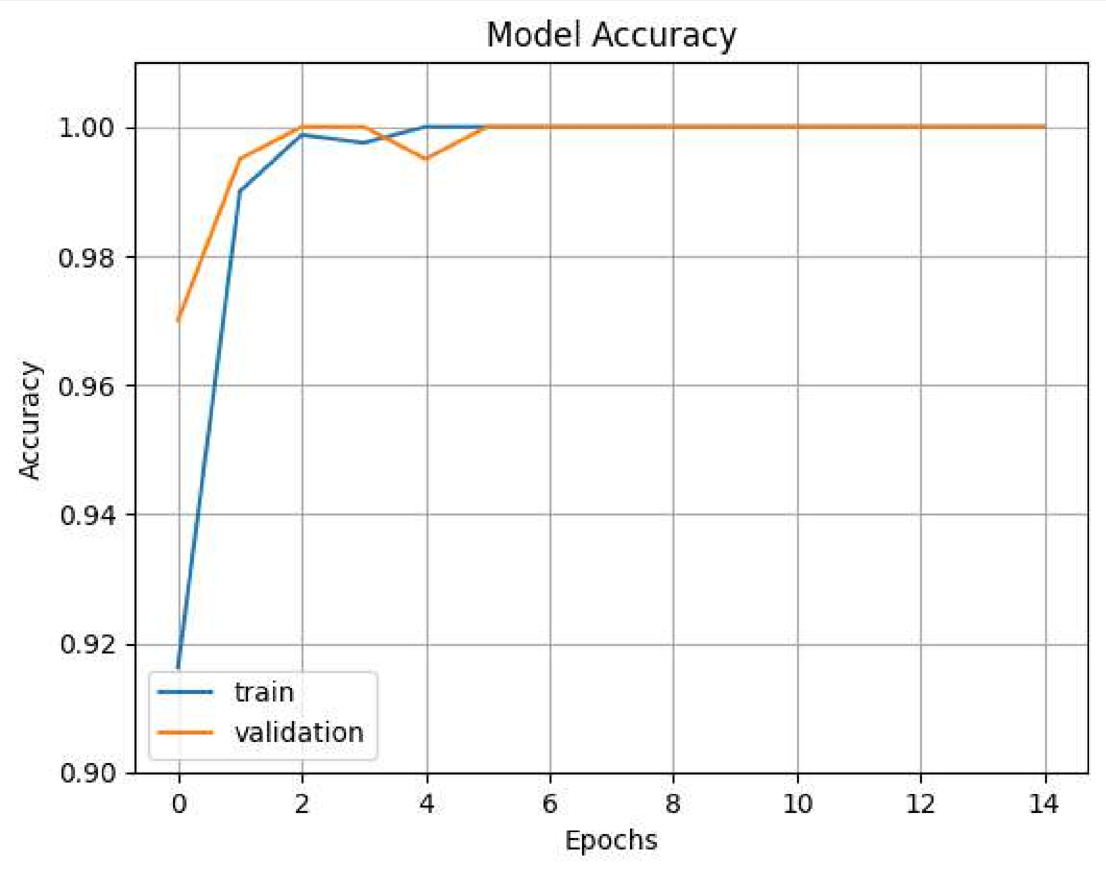
Train the model

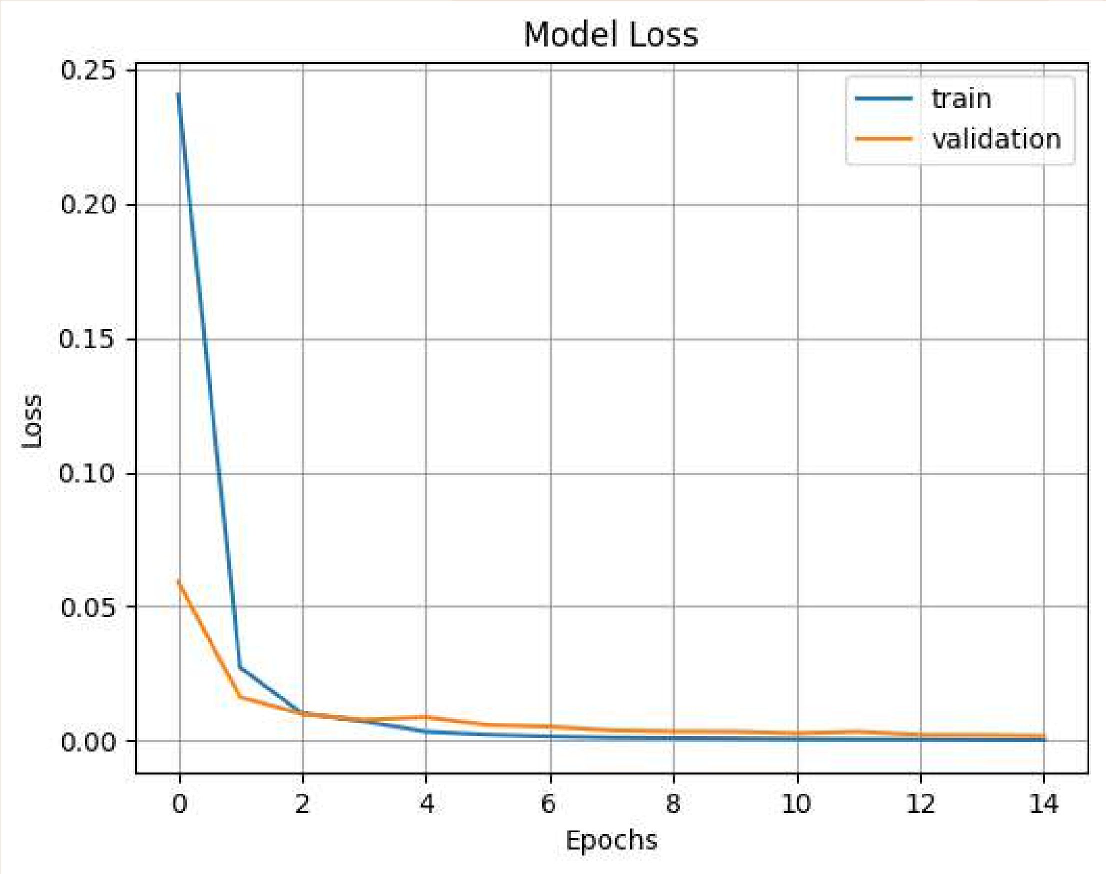
```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.python.keras.layers import Dense, Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
```

```
# have to create a new folder called data and then add the folder mannualy

from pathlib import Path
data_dir =Path(r"/content/Data")
```

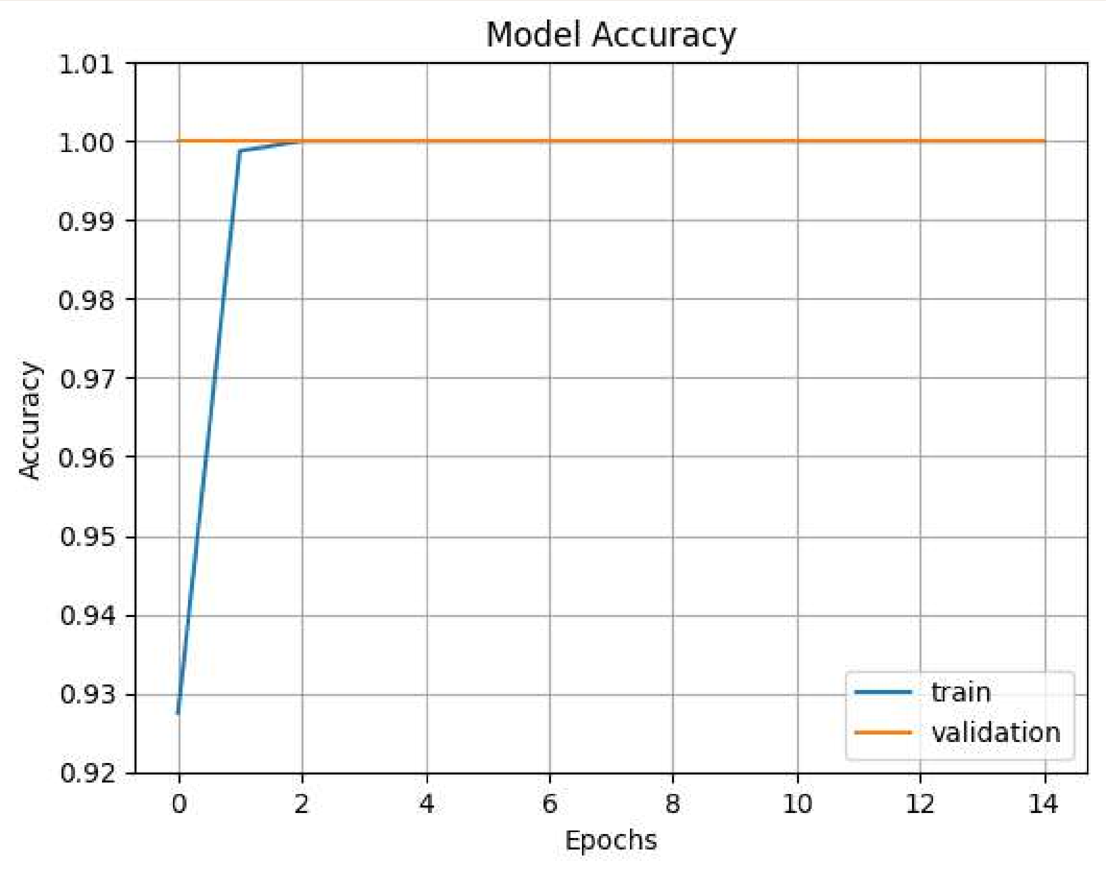
```
img_height,img_width=224,224
batch_size=32
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

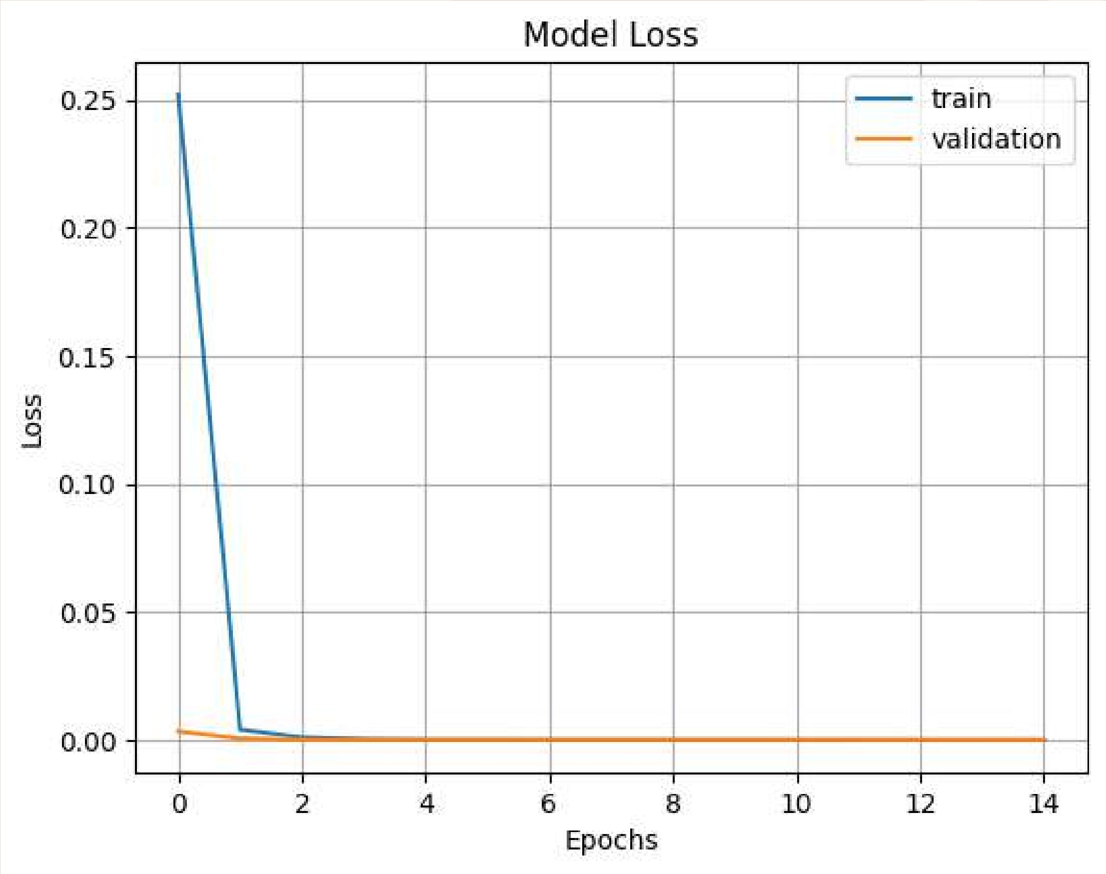
Completion of the model



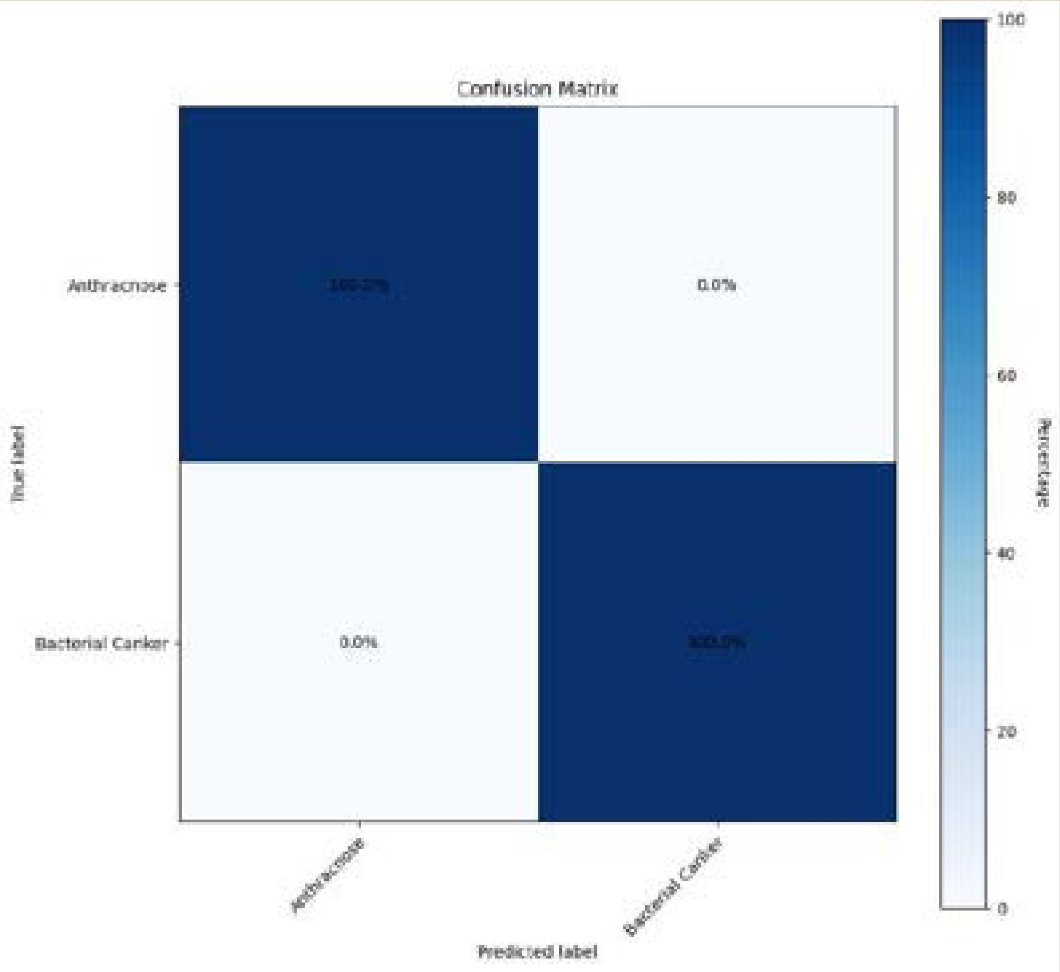


Completion of the model





Confusion Matrix



	precision	recall	f1-score	support
Anthracnose	1.00	1.00	1.00	25
Bacterial Canker	1.00	1.00	1.00	25
accuracy			1.00	50
macro avg	1.00	1.00	1.00	50
weighted avg	1.00	1.00	1.00	50

Completed Implementations

- Collect Dataset.
- Train the Restnet50 ML model using 80% of the training data
- Adjusting and training the model to achieve high accuracy.
- Validate the model using the 20% of test data
- Identify the mango disease.

Future Implementations

- Add 3 more disease to train model and add to system.
- Suggest the prevention solution for the disease.
- Design and Develop the Mobile App.

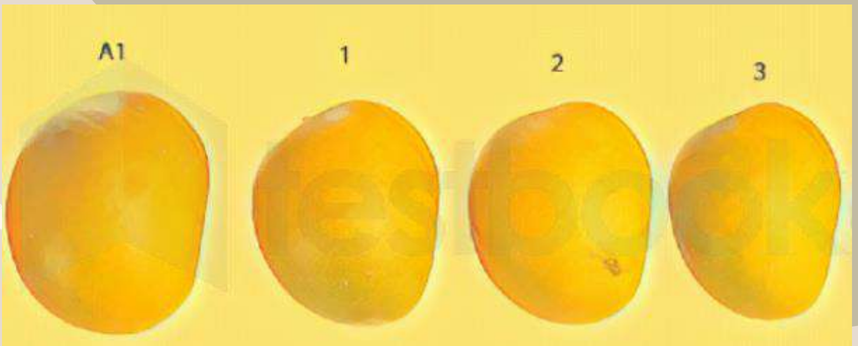
IT20276928
Jayamanne B.D.N

Specialization: Computer Systems and Network Engineering

IoT Based Mango Quality Grading System

Introduction

- Traditional mango grading methods are manual and subjective, with human experiences.
- In grading systems, mangoes are classified into different grades, such as “A, B, and C” classes.
- The proposed system will aim to implement an IoT device and machine learning to develop an automated mango grading system.



RESEARCH QUESTION

- ✓ **How mango farmers increase their cost of mango production ?**
- ✓ **How to increase accuracy of mango grading ?**
- ✓ **How farmers reduce time consuming and workforce?**

Sub Objective

Implement of a machine learning model to identify the grade of mangos from the images.

An IoT device captures mango images and weighs them using a camera and weight sensor.

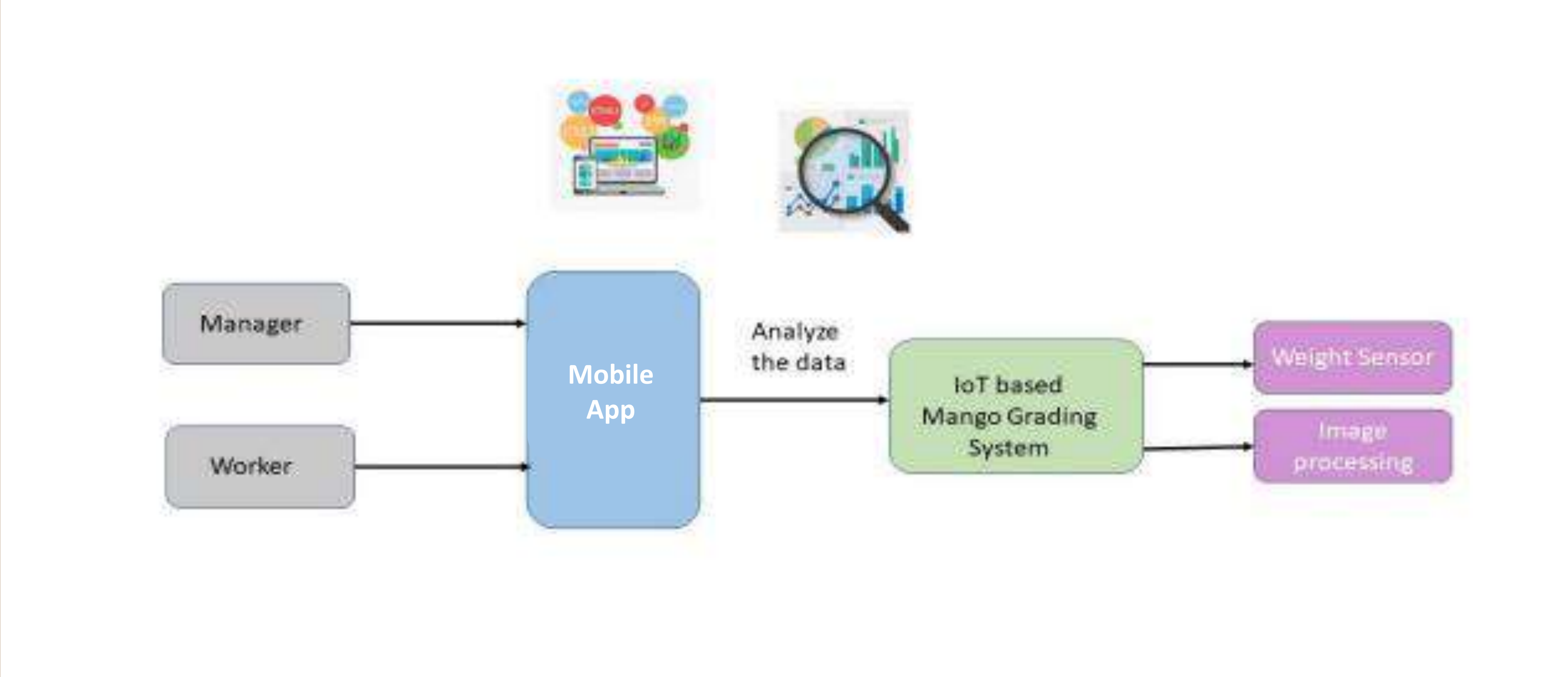
Methodology

A machine learning model is developed to classify mangoes based on their maturity stage and quality grade. The model is trained using the collected dataset of mango images.

An IoT device will be developed to capture images of mangoes and weigh them. The device will be equipped with a camera and a weight sensor, and it will be connected to a central server using wireless communication protocols such as Wi-Fi or Bluetooth.

The IoT device and machine learning model will be integrated to create the proposed mango grading solution. The IoT device will capture images and weight the mango, which will be analyzed by the machine learning model to determine its grade.

System Diagram



Technologies and Tools

- **Python**
- **Google Colab**
- **CNN –Transfer Learning**

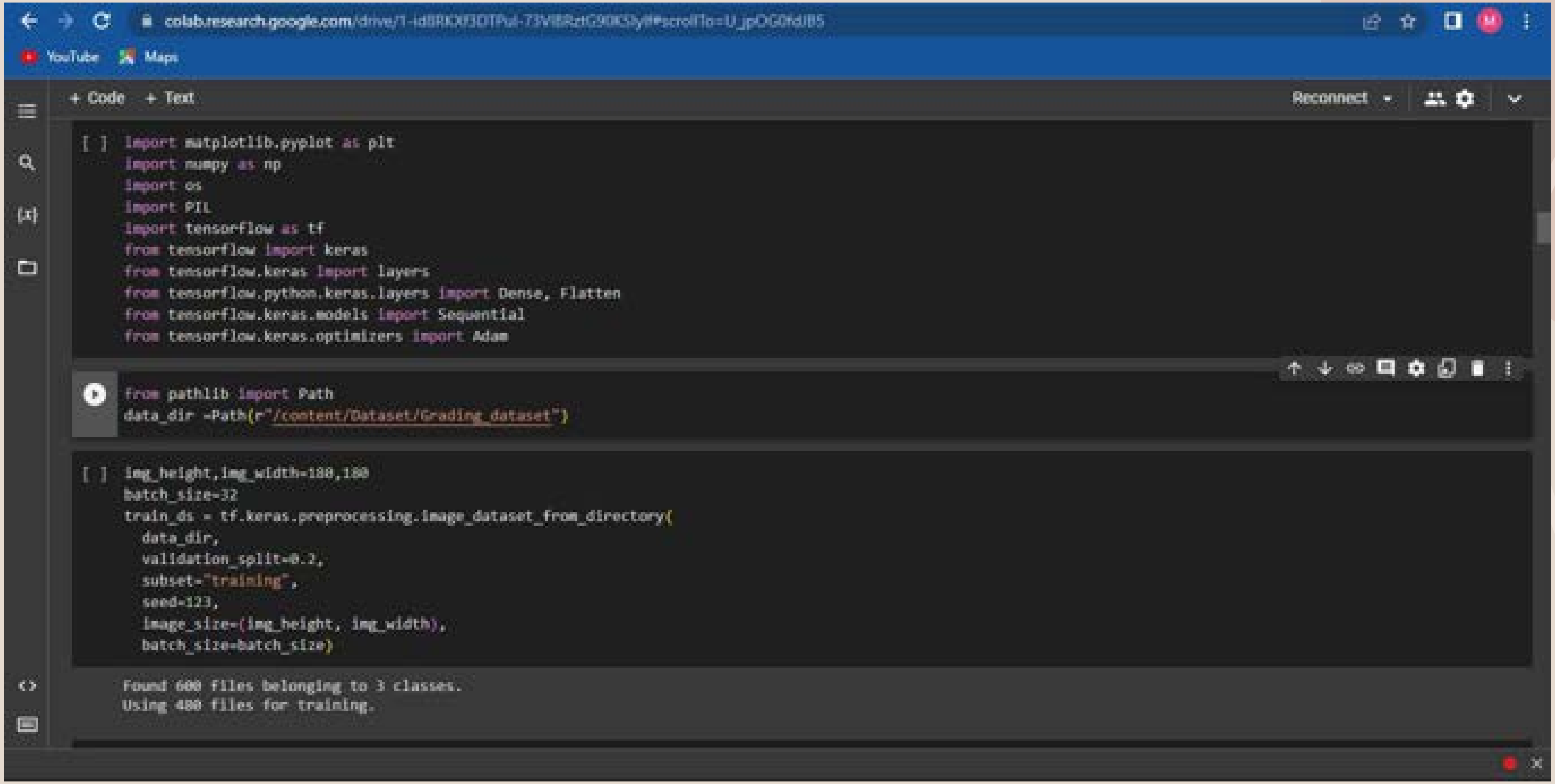
- **Python**
- **Raspberry Pi**
- **Load Sensor**

Collected Dataset

```
! unzip  /content/mango-varieties-classification.zip

Archive:  /content/mango-varieties-classification.zip
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_102834.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_102839.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_102859.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_102913.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_102920.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_102934.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_102950.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103000.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103004.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103018.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103027.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103032.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103040.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103046.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103050.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103118.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103128.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103131.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103135.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103138.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103141.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103144.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103155.jpg
  inflating: Dataset/Classification_dataset/Anwar Ratool/IMG_20210630_103158.jpg
```

Completion

Training the model



```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 600 files belonging to 3 classes.
Using 120 files for validation.
```

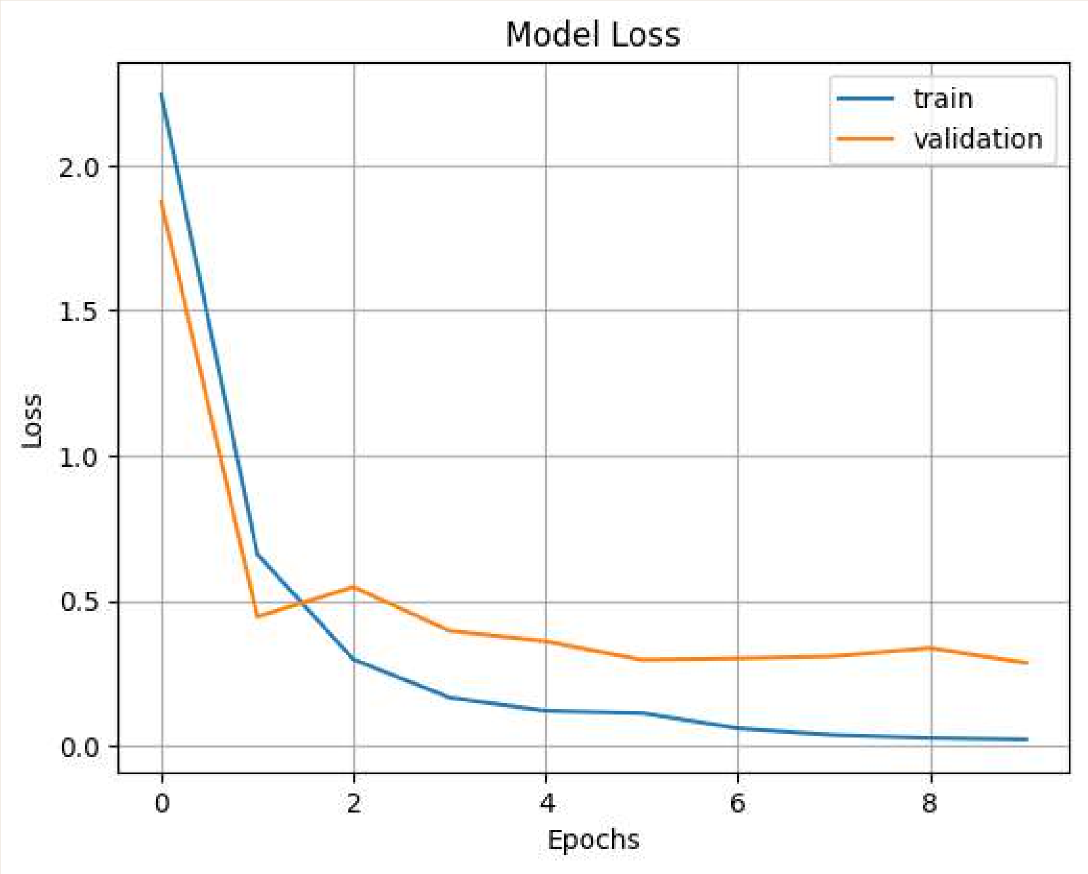
```
class_names = train_ds.class_names
print(class_names)
```

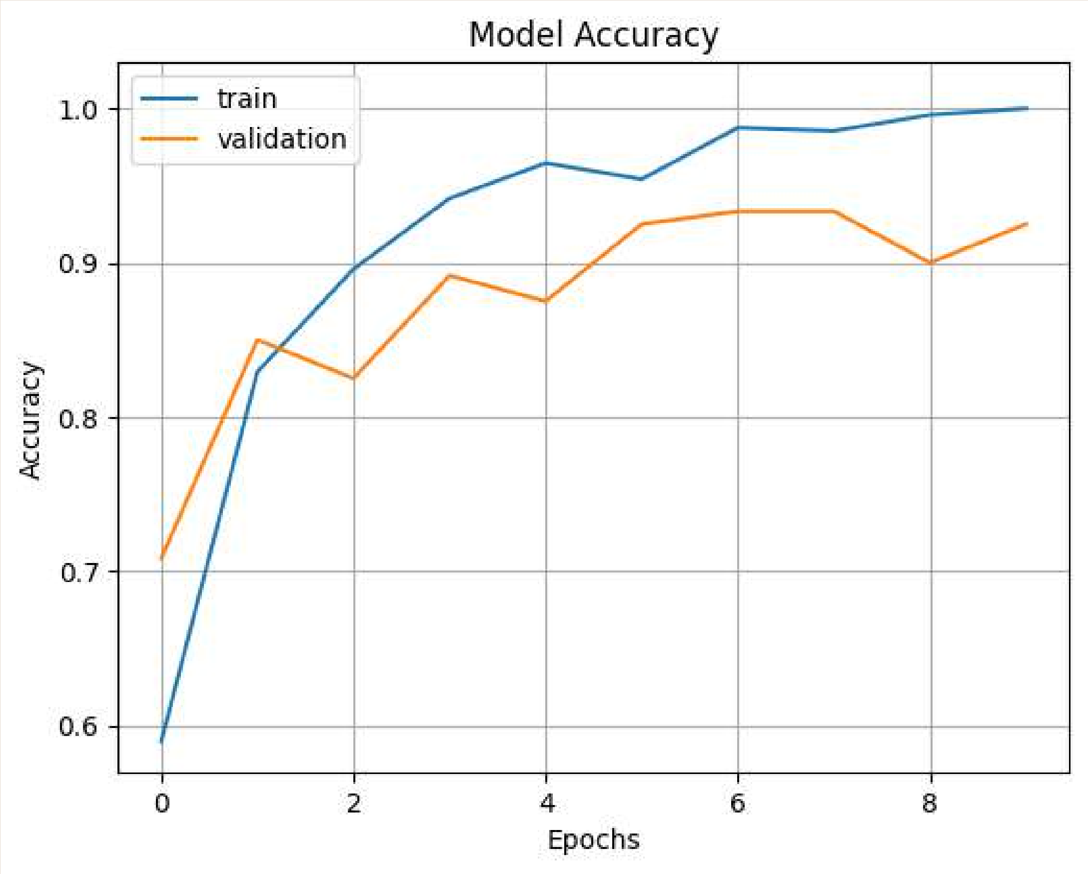
```
['Class_I', 'Class_II', 'Extra_Class']
```

```
VGG16_model = Sequential()
pretrained_VGG16 = tf.keras.applications.VGG16(
    input_shape=(180 , 180 , 3),
    include_top=False,
    weights="imagenet",
    pooling="avg",
    classes=3,

)

for layer in pretrained_VGG16.layers:
        layer.trainable=False

VGG16_model.add(pretrained_VGG16)
VGG16_model.add(Flatten())
VGG16_model.add(Dense(512, activation='relu'))
VGG16_model.add(Dense(3, activation='softmax'))

VGG16_model.compile(optimizer=Adam(learning_rate=0.001),loss='sparse_categorical_crossentropy',metrics=['accuracy'])

epochs=10
history_VGG16 = VGG16_model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

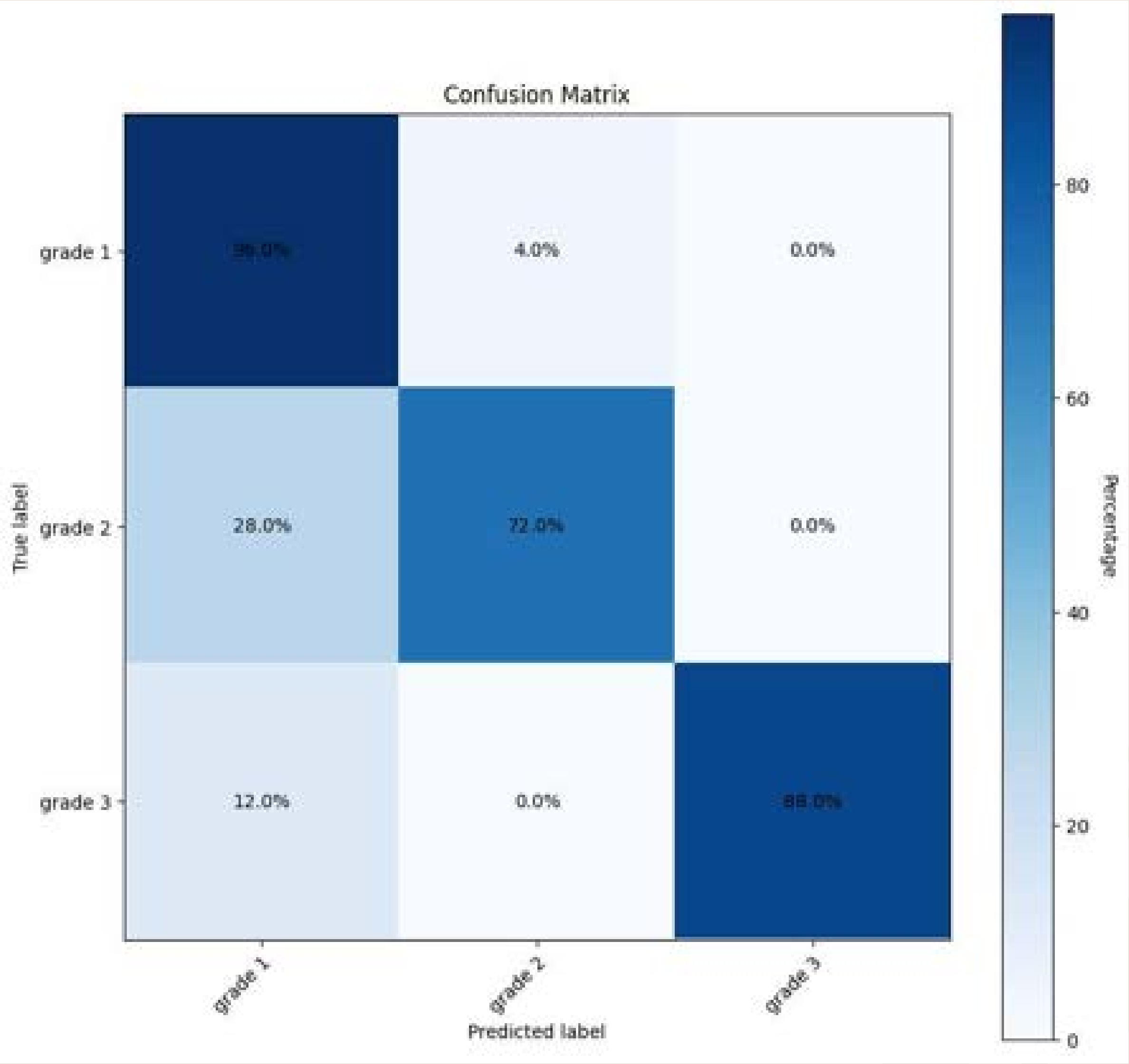
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
```

Accuracy And Loss





Confusion Matrix



Classification Report

```
[17] from sklearn.metrics import classification_report

     target_names =  ["grade 1" ,"grade 2" , "grade 3"]

     classification_rep = classification_report(y_true, y_pred, target_names=target_names)
     print(classification_rep)
```

```
              precision    recall  f1-score   support

     grade 1       0.71      0.96      0.81        25
     grade 2       0.95      0.72      0.82        25
     grade 3       1.00      0.88      0.94        25

    accuracy                           0.85        75
   macro avg       0.88      0.85      0.86        75
weighted avg       0.88      0.85      0.86        75
```

Completed Implementations

- **Collect Dataset.**
- **Train the VGG16 ML model using 80% of the training data**
- **Validate the model using the 20% of test data**
- **Trained the ML model to identify the grade of mangos from the images.**
- **An IoT device will be developed to capture images of mangoes and weigh them.**

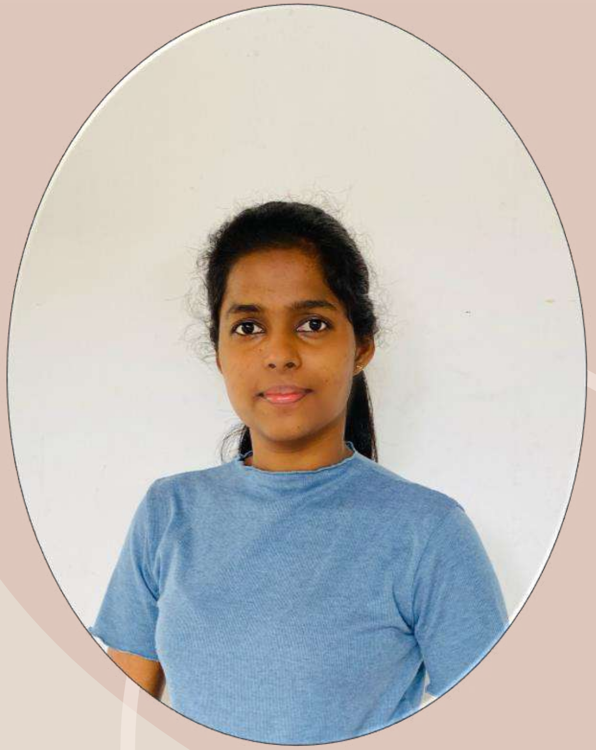
Future Implementations

- **Testing the finished system .**
- **Introduce the system to the clients and research how they interact.**

References

- [1] p. P. L. Yi, "Influences of Different Storage Conditions on Postharvest Quality of Mango," Influences of Different Storage Conditions on Postharvest Quality of Mango , 2019-09-26.
- [2] S. Krug, "AgriEnvironment," 02 May 2023. [Online]. Available: https://www.mdpi.com/2624- 7402/5/2/50.
- [3] T. R. Razak, "Towards Capturing Mango Grading From Human Experts - A Comprehensive User Study," Towards Capturing Mango Grading From Human Experts - A Comprehensive User Study, 14 february 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9703830. [Accessed 10 march 2023].
- [4] L. Pauly, "IEEE," 15 July 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7154891.
- [5] A. K. R. K. A. M. R. K. Virender Singh, "Adoption of post-harvest management practices," 02 10 2020. [Online]. Available: https://www.researchgate.net/profile/Rajesh-Kumar314/publication/343627242_Adoption_of_postharvest_management_practicesby_Mango_growers_of_Haryana/links/5f34c659a6fdcccc43c5ac9e/ Adoption-of-post-harvest-management-practicesby-Mango-growers-of-Haryana.pd. [Accessed 5 May 2023].
- [6] [Online]. Available: https://ja-si.com/gps-tracking-technology/. [7] C. N. d. Ricerche, "Marking Standing Trees with RFID Tags," Consiglio Nazionale delle Ricerche, 29 January 2020. [8] "Development of higher yield and high-quality mango production system based on Internet of Things," 02 April 2019.

IT20103354
Niroshani A.



Specialization: Information Technology

Smart Mango Yield Prediction System

Introduction

- Mango is a highly valued fruit crop globally.
- The ability to predict mango yield in advance helps farmers to make informed decisions regarding resource allocation, harvest planning, and marketing strategies.
- In this research, we aim to develop a Regression model for predicting the yield of mango based on some important key factors.

Research Question

How can the yield of mango be predicted based on factors such as soil PH, soil moisture, temperature, humidity, rainfall, light exposure, life span, disease, and fertilizer usage?

Objective

To develop a Lenear Regression predictive model that accurately estimates mango yield based on the factors of soil pH, soil moisture, temperature, humidity, rainfall, light exposure, lifespan, disease, and fertilizer, thereby providing valuable insights for farmers to optimize mango cultivation and maximize crop productivity.

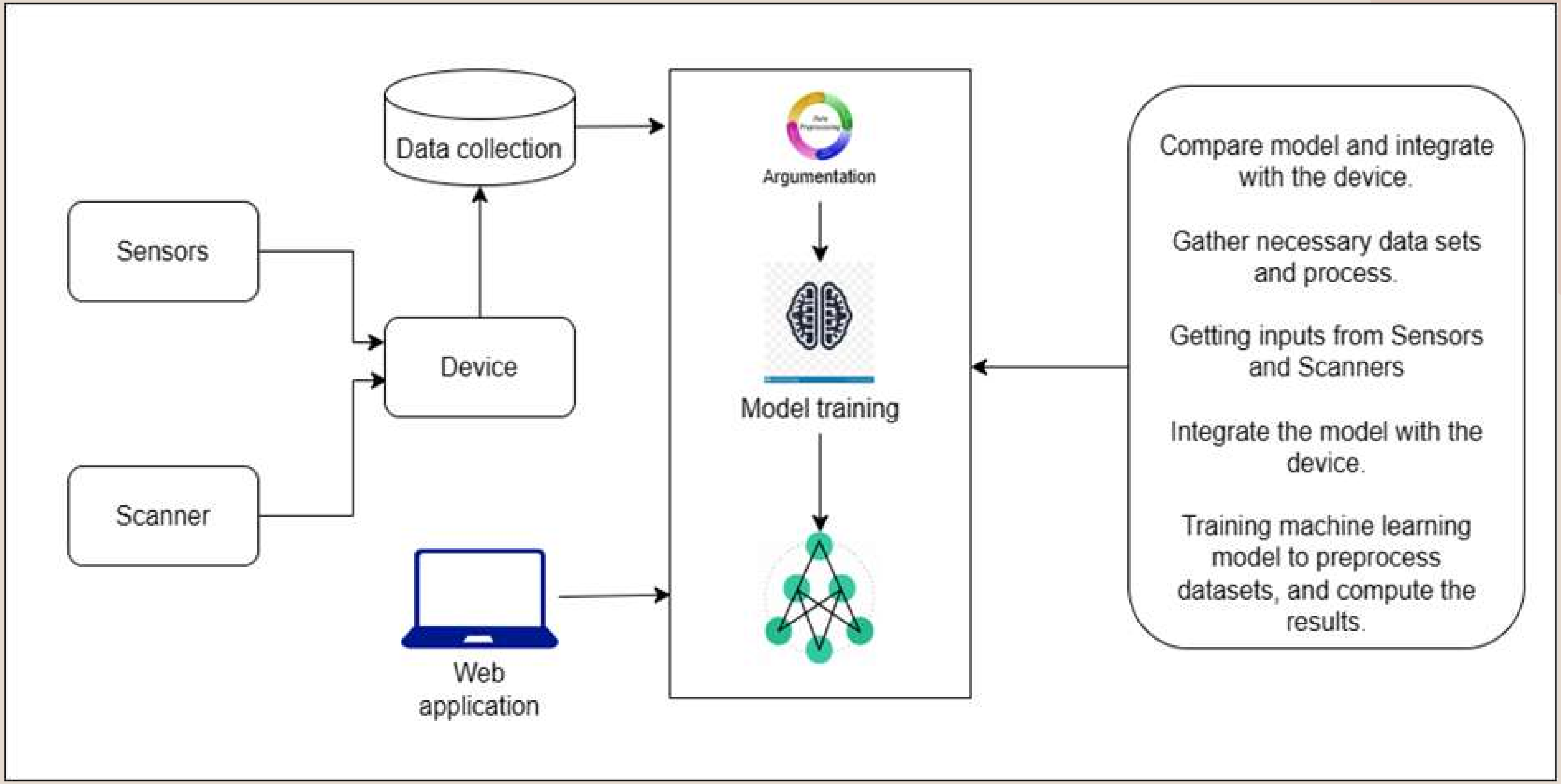
Methodology

- Collect data on various factors that affect mango yields such as soil PH, soil moisture, temperature, humidity, rainfall, light exposure, life span, disease, and fertilizer. And pre-processed collected data.

- Using linear regression machine learning algorithm creating machine learning models for Yield prediction with web application.

- The website will be used to update the real data to the user.

System Diagram



Tools and Technologies

Python

Google colab

Pandas

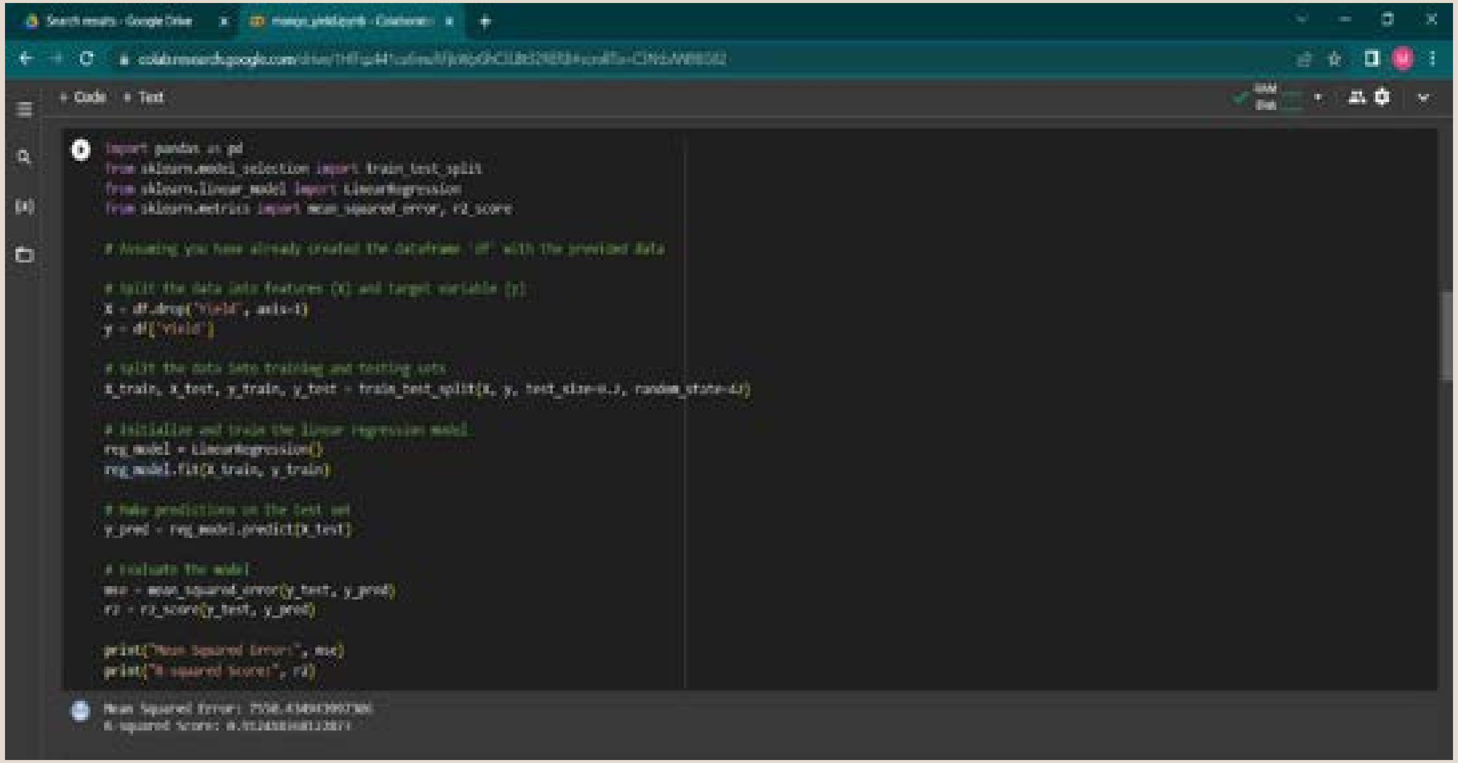
NumPy

Dataset

A	B	C	D	E	F	G	H
Soil pH	Soil Moisture	Temperature	Humidity	Rainfall	Light Exposure	Life Span	Yield
5.592830449	0.4728860193	37.89690025	49.6772381	1515.167403	9.661759319	76.48022885	456.5401036
4.722204137	0.2805666112	39.29586185	30.31741978	477.5824121	2.978145835	15.21388495	177.6788351
5.761179642	0.09031500466	8.09994735	18.30529534	2308.356148	10.67439193	95.87637855	311.1653353
6.124865484	0.5108153096	15.42829299	80.39357692	2180.486784	18.09088132	62.56039669	938.6031243
7.43321794	0.6723658048	14.09918752	55.453696	1760.748061	7.972517802	95.674312	649.1567049
7.623064009	0.9105699222	16.14327768	85.84430575	164.6020408	7.217473809	73.46391903	894.9638337
5.479666675	0.3001829174	12.99905068	10.9145057	2224.789691	13.01585617	28.18743903	225.6819991
7.256968584	0.3890290181	11.02858817	38.18937812	1448.59849	8.432238809	97.13779701	474.9725569
5.421231874	0.05568393994	21.30826799	39.04128671	1959.500361	14.14087635	89.42770838	458.2993329
6.119803179	0.3771212559	30.17791636	30.99466975	245.86488	4.512982075	14.19323201	244.9998651
7.722803312	0.5296243783	7.731203222	43.45903234	1147.125455	11.08305942	53.5980184	559.0569233
7.638132346	0.5560486355	25.78913422	65.42210969	438.1980238	9.16069635	10.06463613	653.4152939
7.683748135	0.8968055854	17.4212078	36.98815192	959.7365844	0.8993979323	53.73565776	392.8483072
5.055587327	0.390905887	17.4973084	23.36042732	2116.289734	15.84379229	60.9965506	335.8197156
7.433524826	0.5567551105	21.44282787	76.47709083	1922.348571	14.21867684	42.23757854	852.8916494
6.403297832	0.6234785439	19.79588143	61.95845258	807.9318554	5.287948544	23.51240441	633.0464441
6.199430288	0.3424612988	25.11189041	62.56851104	1177.978365	7.258874519	45.6950911	625.1284024
6.469795673	0.6636879104	18.46510694	38.65855024	1610.385712	14.3238067	33.61595429	477.739417
7.501090044	0.4011548044	17.80868858	18.62722757	1278.540171	6.891846046	33.38080304	238.9876333
7.101573373	0.5483300725	16.74036677	55.59217672	636.1400944	11.28774114	85.47674375	611.8053479
5.582336033	-0.01142694861	30.11745498	42.85656026	670.3655846	14.9236133	71.41460641	415.3007572
7.6994176	0.4714687831	10.83680186	79.42912891	2072.348325	12.45300168	44.10550931	935.1747855
8.28234654	0.9269928489	17.70369453	68.64614204	2132.486663	7.947347517	92.72425621	783.3004821
5.013083626	0.113663944	34.95815749	35.87499468	2141.678446	9.937777871	90.50459317	336.3323627
6.828172228	0.4978209407	14.76510421	43.04370102	399.3078733	6.183372469	23.87883026	458.0063279
6.135141348	0.5827830263	22.96930619	45.31336539	1074.430575	7.328254971	92.12324328	467.2765695
4.906093531	0.1175580029	39.38071961	12.37956696	658.0211342	1.623838713	73.44010024	-5.474208056
5.089430245	0.1245967242	40.12512647	26.49319027	990.4401709	12.62948138	65.77233079	188.0843468
6.753534958	0.3891913825	10.15943874	43.54870795	1606.087043	11.19301225	23.54681693	550.7772973
7.734238216	0.5559698016	38.70538399	46.58628833	1070.248656	7.224087232	33.19467529	408.1367724

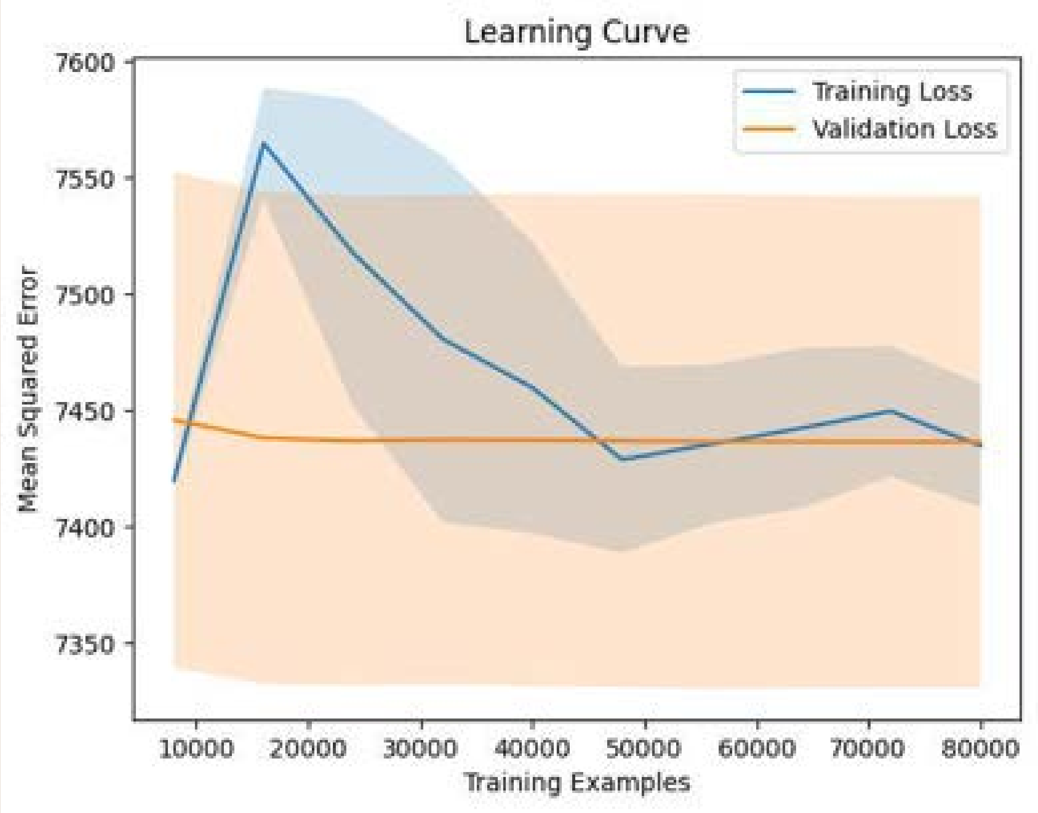
- Collected a dataset with various factors that affect mango yield. (Agriculture department and
jewelex agro plantation)

Train the ML Model



- The Model trained using Google Colab online tool
- Separate dataset is used to validate the model

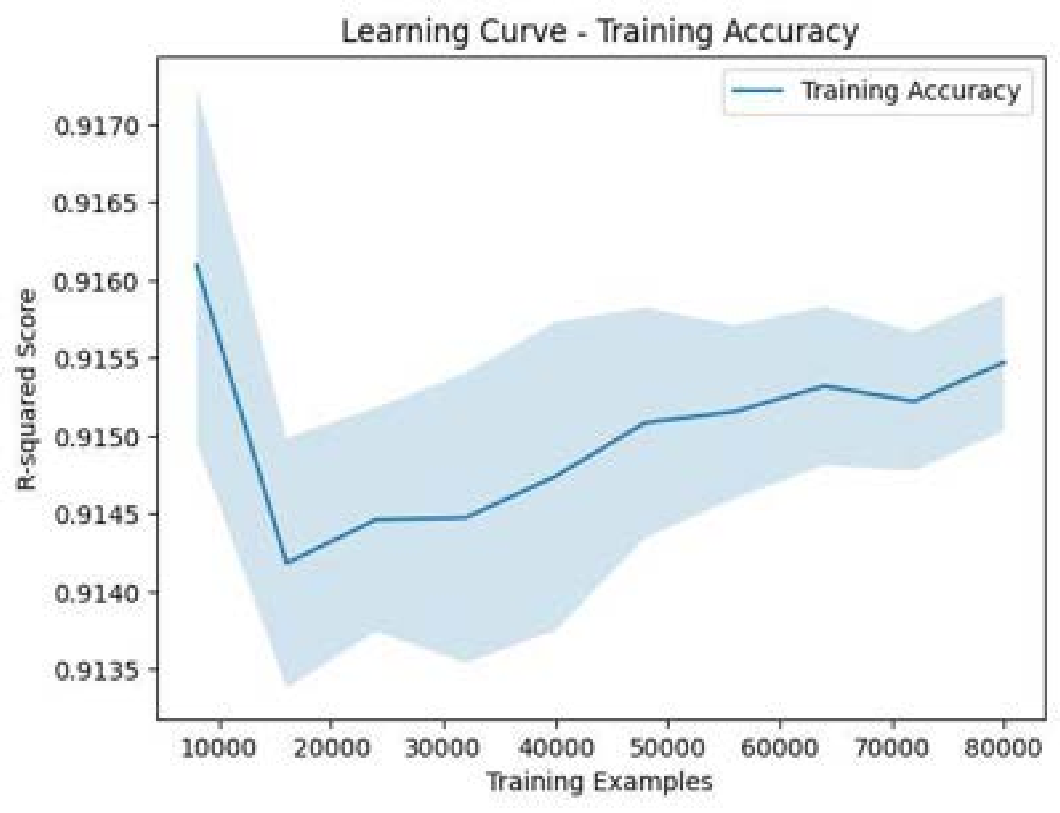
Learning Curve



Loss: 7550.434

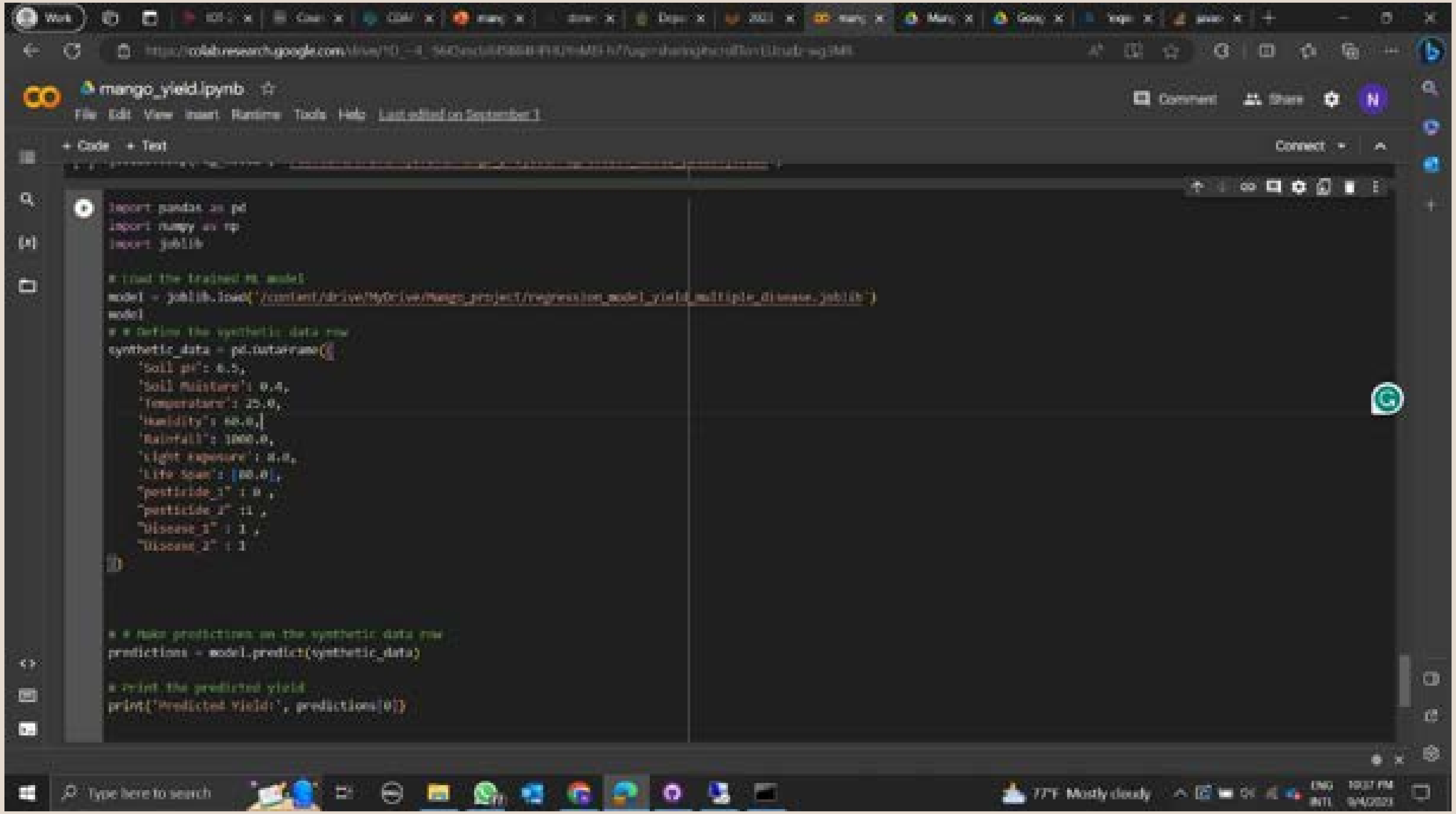
Validation Loss: 7448.531

Learning Curve–Training Accuracy



Accuracy: 0.912418368122873

Test and load ML Model



Predicted Yield: 259.86 t/ha (metric tons per hectare)

Functional Requirements

- The user should be able to generate accurate and reliable predictions.
- The system should be designed to handle a large volume of data and provide timely predictions.
- The system should have an intuitive and user-friendly interface that allows users to easily manage.

Completed Implementations

- Collect Dataset.
- Train the Linear regression ML model using 80% of the training data
- Adjusting and training the model to achieve high accuracy.
- Validate the model using the 20% of test data
- Predict the mango yield.
- Add disease and pesticide parameters as independent variables.
- Implement the real-time data input feature using the mobile app.
- Design and Develop the Frontend.

Future Implementations

- Do further improvements in the mobile application
- Introduce the system to the clients and research how they interact.