PlantPal

SMART ASSISTANCE FOR THE FLORICULTURE INDUSTRY

2023-133

SLIIT
FACULTY OF COMPUTING

MEET OUR TEAM

IT19994406

Basnayake N.S.N.

IT19169736

Gamage M.G.U.D.

IT20017088

Prabhashi P.A.N.

IT20005726

Liyanage S.R.

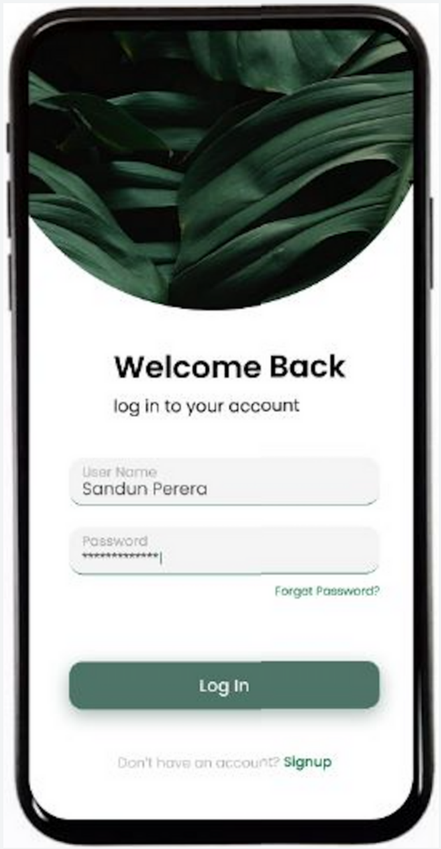
SLIIT
FACULTY OF COMPUTING

2023-133

INTRODUCTION

- Sri Lanka's floriculture industry faces technology inefficiencies.
- Manual plant monitoring makes it difficult to estimate growth.
- Detect deficiencies or mite attacks through visual inspection is also challenging.
- Beginners lack knowledge sources for plant selection and care.
- Demand forecasting relies on past experience, risking incorrect decisions and resource waste.

MAIN OBJECTIVE



Automated mobile application to provide smart assistance for the floriculture industry

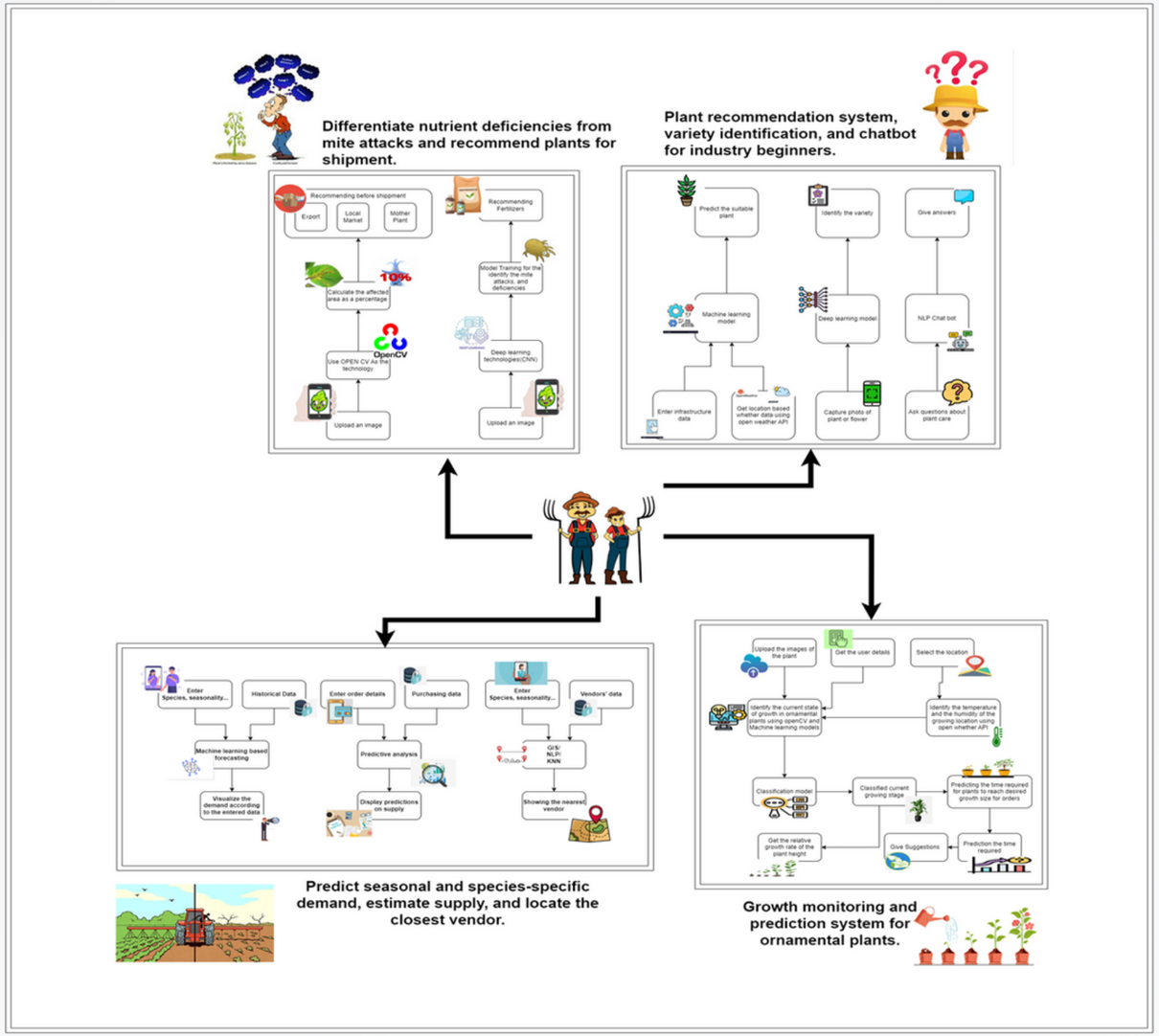
SLIIT
FACULTY OF COMPUTING

2023-133

SUB OBJECTIVES

- Growth monitoring and prediction system for ornamental plants.
- Differentiate nutrient deficiencies and recommend plants for shipment.
- Plant recommendation system and variety identification for industry beginners.
- Predict seasonal and species-specific demand, estimate supply, and locate the closest vendor.

OVERALL SYSTEM ARCITECTURE



SLIIT
FACULTY OF COMPUTING

2023-133

FIELD VISIT TO OMEGA GREEN (PVT) LTD, BADALGAMA

SLIIT
FACULTY OF COMPUTING

2023-133

IT19994406
Basnayake N.S.N.

DIFFERENTIATE NUTRIENT DEFICIENCIES AND RECOMMEND PLANTS FOR SHIPMENT

SLIIT
FACULTY OF COMPUTING

Specialization : Data Science

RESEARCH PROBLEM

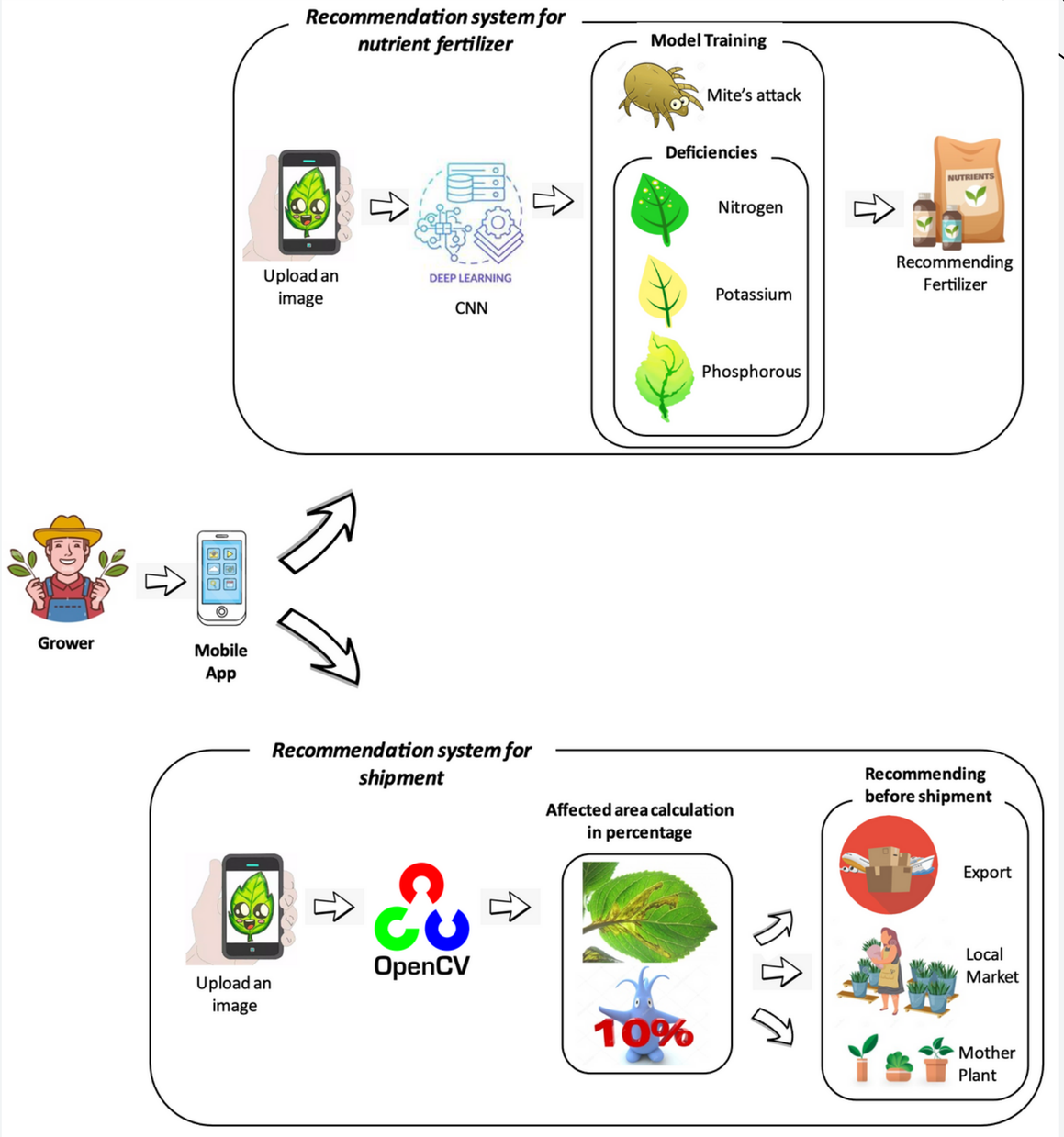
How to determine the correct nutrient fertilizer for the affected plant?

- Nitrogen
- Phosphorous
- Potassium

How to determine whether to ,

- Export leaves
- Sell them locally
- Keep them for the mother plant

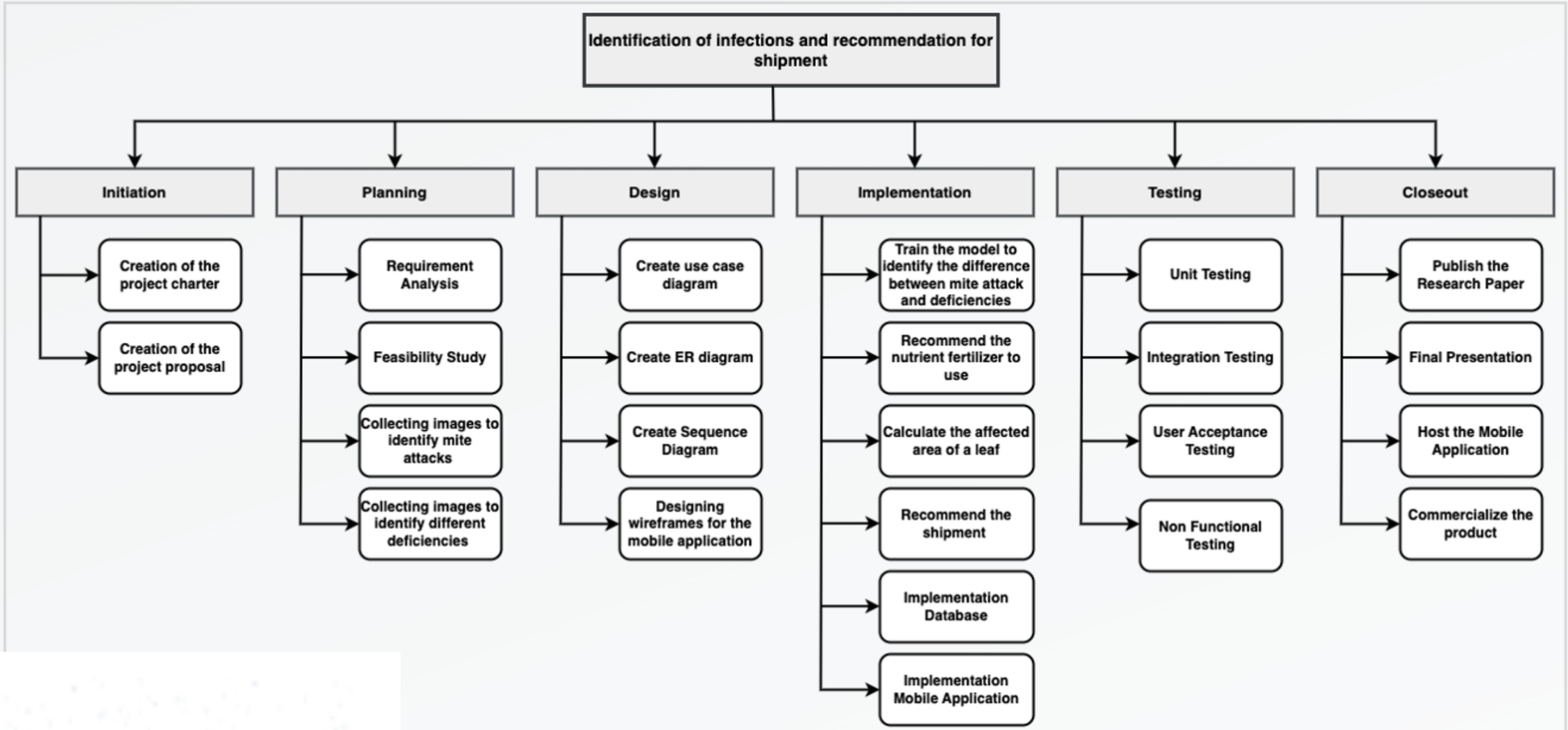
INDIVIDUAL SYSTEM DIAGRAM



SLIIT FACULTY OF COMPUTING

IT19994406 | Basnayake N.S.N. | 2023-133

WORK BREAKDOWN



SAMPLE DATA

SLIIT
FACULTY OF COMPUTING

IT19994406 | Basnayake N.S.N. | 2023-133

IMPLEMENTATION

IDENTIFYING THE DEFICIENCIES

Data Preprocessing

```
import splitfolders

input_folder = "/content/drive/MyDrive/images/deficiencies"
output_folder = "/content/drive/MyDrive/output"

splitfolders.ratio(input_folder, output=output_folder, seed=1337, ratio=(.75, .2, .05), group_prefix=None, move=False)
```

```
Copying files: 1170 files [00:50, 23.00 files/s]
```

IMPLEMENTATION

Exploratory Data Analysis

```
# Folder location
data_dir = pathlib.Path('/content/drive/MyDrive/output/train')
train_dir = '/content/drive/MyDrive/output/train'
val_dir = '/content/drive/MyDrive/output/val'
test_dir = '/content/drive/MyDrive/output/test'
```

```
[ ] # Show classes
    class_name = np.array(sorted([item.name for item in data_dir.glob("*")]))
    print(class_name)

    ['Nitrogen(N)' 'Phosphorus(P)' 'Potassium(K)']
```

```
[ ] # Show total data in folder

    for dirpath, dirnames, filenames in os.walk('/content/drive/MyDrive/output'):
        print(f"{len(dirnames)} folders and {len(filenames)} images in {dirpath}")

    3 folders and 0 images in /content/drive/MyDrive/output
    3 folders and 0 images in /content/drive/MyDrive/output/train
    0 folders and 287 images in /content/drive/MyDrive/output/train/Potassium(K)
    0 folders and 340 images in /content/drive/MyDrive/output/train/Nitrogen(N)
    0 folders and 249 images in /content/drive/MyDrive/output/train/Phosphorus(P)
    3 folders and 0 images in /content/drive/MyDrive/output/val
    0 folders and 76 images in /content/drive/MyDrive/output/val/Potassium(K)
    0 folders and 90 images in /content/drive/MyDrive/output/val/Nitrogen(N)
    0 folders and 66 images in /content/drive/MyDrive/output/val/Phosphorus(P)
    3 folders and 0 images in /content/drive/MyDrive/output/test
    0 folders and 20 images in /content/drive/MyDrive/output/test/Potassium(K)
    0 folders and 24 images in /content/drive/MyDrive/output/test/Nitrogen(N)
    0 folders and 18 images in /content/drive/MyDrive/output/test/Phosphorus(P)
```

```
[ ] for i in range(1, 6):
        img_n = view_random_image(target_dir='/content/drive/MyDrive/output/train/', target_class='Nitrogen(N)')
```

Image Shape : (4032, 589, 3)

Image Shape : (


```
[ ] for i in range(1, 6):
        img_n = view_random_image(target_dir='/content/drive/MyDrive/output/train/', target_class='Phosphorus(P)')
```

Image Shape : (3883, 384, 3)

Image Shape : (2208,


```
[ ] for i in range(1, 6):
        img_n = view_random_image(target_dir='/content/drive/MyDrive/output/train/', target_class='Potassium(K)')
```

Image Shape : (3988, 372, 3)

Image Shape : (3930, 376, 3)

SLIIT
FACULTY OF COMPUTING

IT19994406 | Basnayake N.S.N. | 2023-133

IMPLEMENTATION

Feature Engineering

```
[ ]
    # Sets the global random seed.
    tf.random.set_seed(46)

    # preprocess data
    train_datagen = ImageDataGenerator(rescale=1/255.0,
                                       rotation_range=0.2,
                                       zoom_range=0.2,
                                       width_shift_range=0.2,
                                       height_shift_range=0.2,
                                       vertical_flip=True,
                                       horizontal_flip=True)
    valid_datagen = ImageDataGenerator(rescale=1/255.0)
    test_datagen = ImageDataGenerator(rescale=1/255.0)
```

```
[ ] # flow from directory using datagen (Pipeline)
    train_data = train_datagen.flow_from_directory(train_dir,
                                                   batch_size=16,
                                                   target_size=(224, 224),
                                                   class_mode='categorical',
                                                   shuffle=True,
                                                   seed=46)

    valid_data = valid_datagen.flow_from_directory(val_dir,
                                                   batch_size=16,
                                                   target_size=(224, 224),
                                                   class_mode='categorical',
                                                   shuffle=False,
                                                   seed=46)

    test_data = test_datagen.flow_from_directory(test_dir,
                                                   batch_size=16,
                                                   target_size=(224, 224),
                                                   class_mode='categorical',
                                                   shuffle=False,
                                                   seed=46)
```

```
Found 876 images belonging to 3 classes.
Found 232 images belonging to 3 classes.
Found 62 images belonging to 3 classes.
```

```
[ ] # Check class_indices
    train_data.class_indices
```

```
{'Nitrogen(N)': 0, 'Phosphorus(P)': 1, 'Potassium(K)': 2}
```

```
[ ] # Set labels
    train_y=train_data.classes
    val_y=valid_data.classes
    test_y=test_data.classes
```

```
[ ] # Check shapes of labels
    print("train_y.shape: ", train_y.shape)
    print("val_y.shape: ", val_y.shape)
    print("test_y.shape: ", test_y.shape)
```

```
train_y.shape:  (876,)
val_y.shape:  (232,)
test_y.shape:  (62,)
```

IMPLEMENTATION

Modeling

```
[ ] # Model Definition
    base_model = tf.keras.applications.DenseNet121(input_shape=(224, 224, 3),
                                                    include_top=False,
                                                    weights='imagenet')

    base_model.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet121_weights_tf_dim_ordering_tf_kernels_notop.h5
29084464/29084464 [==============================] - 0s 0us/step
```

```
[ ] # Model Preparation
    global_average_layer = GlobalAveragePooling2D()
    prediction_layer = Dense(3)
    softmax = Activation('softmax')

    inputs = Input(shape=(224, 224, 3))
    x = base_model(inputs, training=False)
    x = Dropout(0.25)(x)
    x = global_average_layer(x)
    outputs = prediction_layer(x)
    outputs = softmax(outputs)

    model = Model(inputs, outputs)
```

```
[ ] # Model Summary
    model.summary()
```

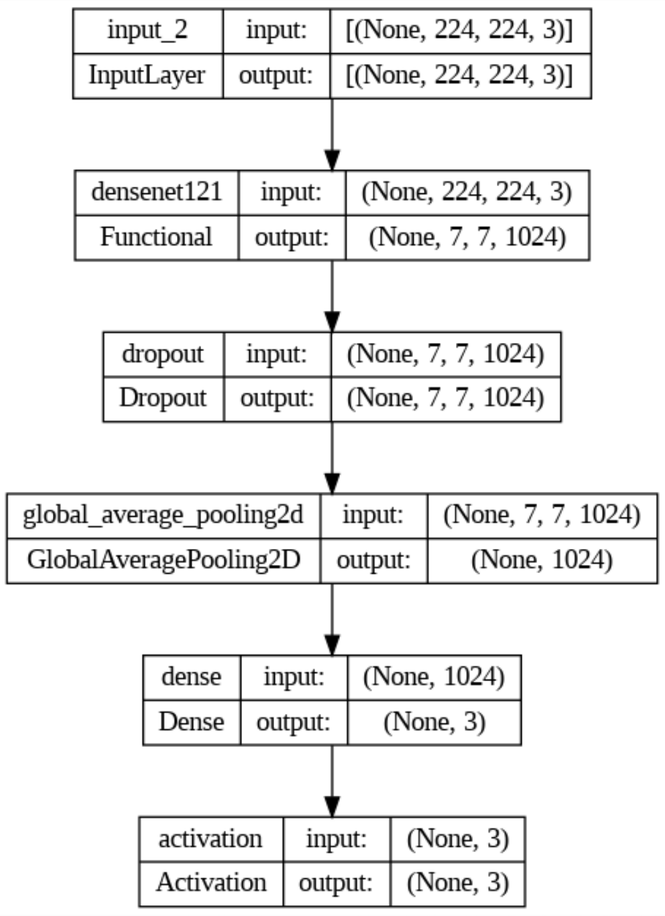
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 densenet121 (Functional)    (None, 7, 7, 1024)        7037504

 dropout (Dropout)           (None, 7, 7, 1024)        0

 global_average_pooling2d (G  (None, 1024)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 3)                 3075

 activation (Activation)     (None, 3)                 0

=================================================================
Total params: 7,040,579
Trainable params: 3,075
Non-trainable params: 7,037,504
```

Screenshot

```
[ ] # compile model
    model.compile(loss='categorical_crossentropy',
                  optimizer=Adam(learning_rate=0.001),
                  metrics=['accuracy'])

    # train model
    history = model.fit(train_data, epochs=15, validation_data=valid_data)
```

```
Epoch 1/15
55/55 [==============================] - 166s 3s/step - loss: 0.9622 - accuracy: 0.5502 - val_loss: 0.7621 - val_accuracy: 0.6552
Epoch 2/15
55/55 [==============================] - 153s 3s/step - loss: 0.6033 - accuracy: 0.7842 - val_loss: 0.5747 - val_accuracy: 0.7759
Epoch 3/15
55/55 [==============================] - 158s 3s/step - loss: 0.4751 - accuracy: 0.8345 - val_loss: 0.4858 - val_accuracy: 0.8147
Epoch 4/15
55/55 [==============================] - 156s 3s/step - loss: 0.4307 - accuracy: 0.8459 - val_loss: 0.4290 - val_accuracy: 0.8448
Epoch 5/15
55/55 [==============================] - 160s 3s/step - loss: 0.3595 - accuracy: 0.8710 - val_loss: 0.3572 - val_accuracy: 0.8836
Epoch 6/15
55/55 [==============================] - 159s 3s/step - loss: 0.3451 - accuracy: 0.8847 - val_loss: 0.3367 - val_accuracy: 0.9009
Epoch 7/15
55/55 [==============================] - 157s 3s/step - loss: 0.3273 - accuracy: 0.8881 - val_loss: 0.3171 - val_accuracy: 0.8922
Epoch 8/15
55/55 [==============================] - 146s 3s/step - loss: 0.2859 - accuracy: 0.9075 - val_loss: 0.2934 - val_accuracy: 0.9009
Epoch 9/15
55/55 [==============================] - 164s 3s/step - loss: 0.2771 - accuracy: 0.9155 - val_loss: 0.2838 - val_accuracy: 0.9138
Epoch 10/15
55/55 [==============================] - 158s 3s/step - loss: 0.2430 - accuracy: 0.9224 - val_loss: 0.2945 - val_accuracy: 0.9095
Epoch 11/15
55/55 [==============================] - 158s 3s/step - loss: 0.2623 - accuracy: 0.8938 - val_loss: 0.2504 - val_accuracy: 0.9267
Epoch 12/15
55/55 [==============================] - 141s 3s/step - loss: 0.2267 - accuracy: 0.9235 - val_loss: 0.2519 - val_accuracy: 0.9267
Epoch 13/15
55/55 [==============================] - 156s 3s/step - loss: 0.2386 - accuracy: 0.9144 - val_loss: 0.2718 - val_accuracy: 0.9095
Epoch 14/15
55/55 [==============================] - 156s 3s/step - loss: 0.2210 - accuracy: 0.9189 - val_loss: 0.2252 - val_accuracy: 0.9353
Epoch 15/15
55/55 [==============================] - 154s 3s/step - loss: 0.2131 - accuracy: 0.9338 - val_loss: 0.2628 - val_accuracy: 0.9095
```

Model Summary

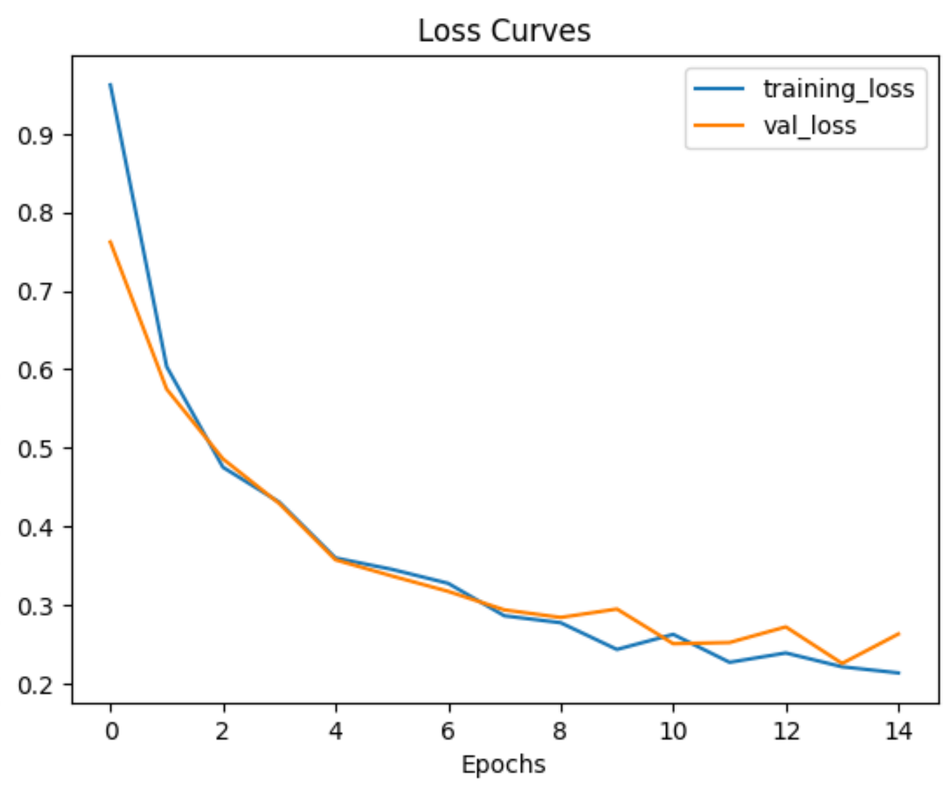


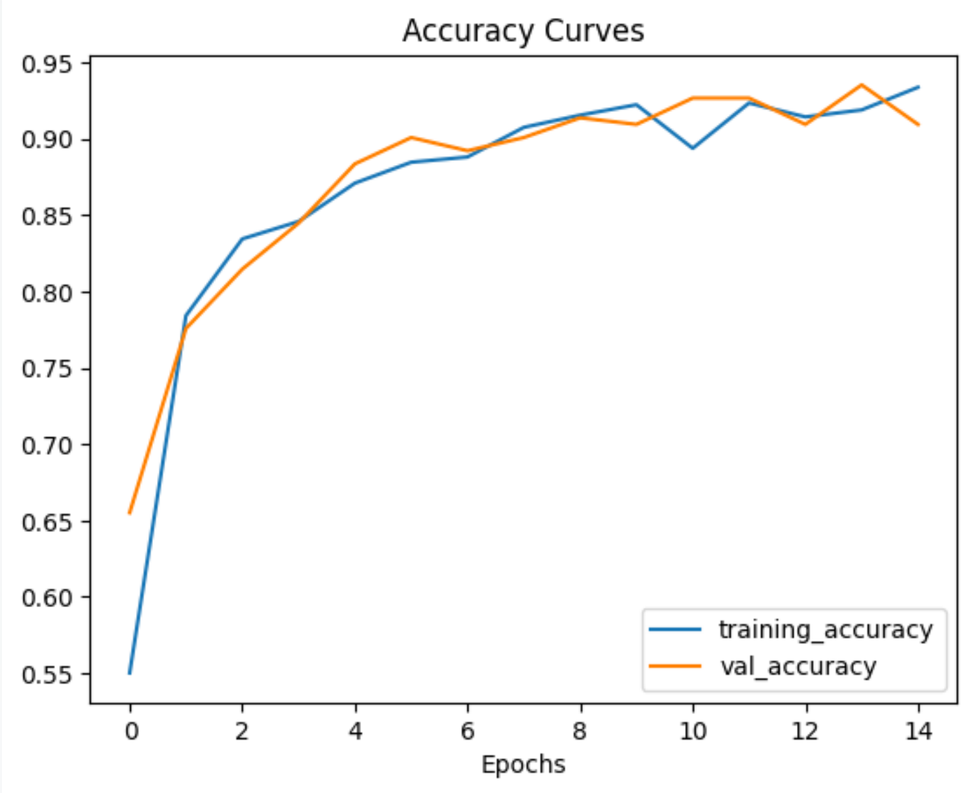
SLIIT FACULTY OF COMPUTING

IT19994406 | Basnayake N.S.N. | 2023-133

IMPLEMENTATION

Model Evaluation



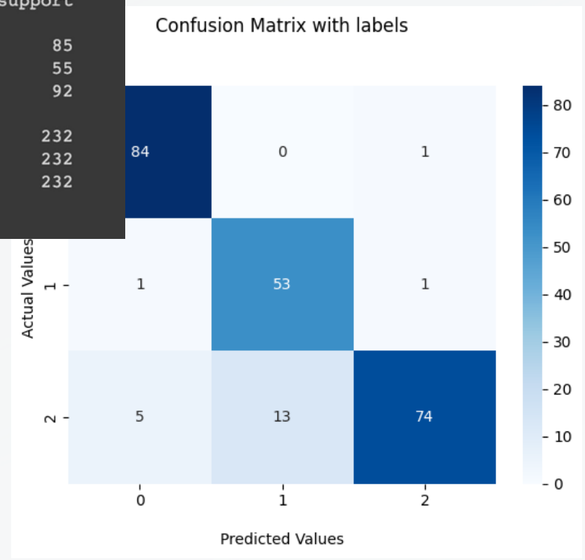


IMPLEMENTATION

Classification Report for Validate Data

```
[ ]  # classification report
     val_pred = model.predict(valid_data)
     val_pred = val_pred.argmax(axis=1)
     print(classification_report(val_pred, val_y))
```

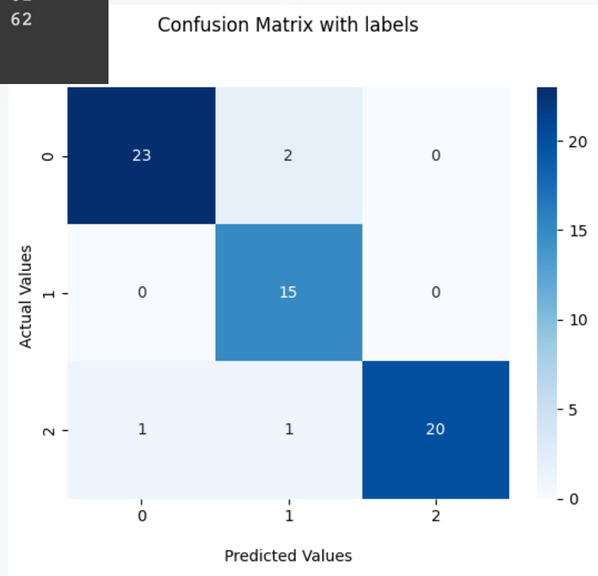
```
    15/15 [==============================] - 32s 2s/step
                  precision    recall  f1-score   support

               0       0.93      0.99      0.96        85
               1       0.80      0.96      0.88        55
               2       0.97      0.80      0.88        92

        accuracy                           0.91       232
       macro avg       0.90      0.92      0.91       232
    weighted avg       0.92      0.91      0.91       232
```



Classification Report for Test Data

```
[ ]  # classification report
     test_pred = model.predict(test_data)
     test_pred = test_pred.argmax(axis=1)
     print(classification_report(test_pred, test_y))
```

```
    4/4 [==============================] - 9s 2s/step
                  precision    recall  f1-score   support

               0       0.96      0.92      0.94        25
               1       0.83      1.00      0.91        15
               2       1.00      0.91      0.95        22

        accuracy                           0.94        62
       macro avg       0.93      0.94      0.93        62
    weighted avg       0.94      0.94      0.94        62
```



SLIIT
FACULTY OF COMPUTING

IT19994406 | Basnayake N.S.N. | 2023-133

IMPLEMENTATION

CALCULATING THE EFFECTED AREA

```
[ ]  import cv2
     import numpy as np
     #from config import input_path, leaf_area

     input_path = '/content/drive/MyDrive/leaf_images/test.jpeg'
     leaf_area = '/content/drive/MyDrive/leaf_images/mask_test.jpeg'

     # Open a simple image
     img = cv2.imread(input_path)
     filename_mask = leaf_area




     # converting from gbr to hsv color space
     img_HSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

     # leaf color range for hsv color space
     HSV_mask = cv2.inRange(img_HSV, (0, 15, 0), (17, 170, 255))

     HSV_mask = cv2.morphologyEx(HSV_mask, cv2.MORPH_OPEN, np.ones((3, 3), np.uint8))

     # converting from gbr to YCbCr color space
     img_YCrCb = cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)

     # leaf color range for hsv color space
     YCrCb_mask = cv2.inRange(img_YCrCb, (0, 135, 85), (255, 180, 135))

     YCrCb_mask = cv2.morphologyEx(YCrCb_mask, cv2.MORPH_OPEN, np.ones((3, 3), np.uint8))
     global_mask = cv2.bitwise_and(YCrCb_mask, HSV_mask)
     global_mask = cv2.medianBlur(global_mask, 3)
     global_mask = cv2.morphologyEx(global_mask, cv2.MORPH_OPEN, np.ones((4, 4), np.uint8))
     cv2.imwrite(filename_mask, global_mask)

     # counting the number of pixels
     number_of_white_pix = np.sum(global_mask == 255)
     number_of_black_pix = np.sum(global_mask == 0)
     total_pix = number_of_white_pix + number_of_black_pix
     white_percentage = (number_of_white_pix / total_pix)

     print("Leaf affected percentage: " + "{:.2%}".format(white_percentage));

     Leaf affected percentage: 12.04%
```

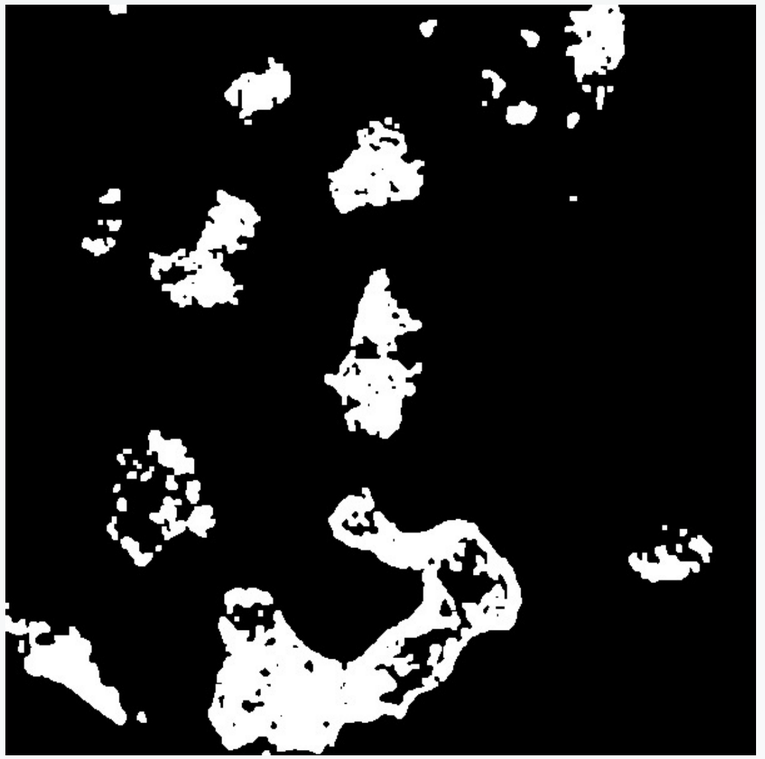
SLIIT
FACULTY OF COMPUTING

IT19994406 | Basnayake N.S.N. | 2023-133

IMPLEMENTATION

Actual image

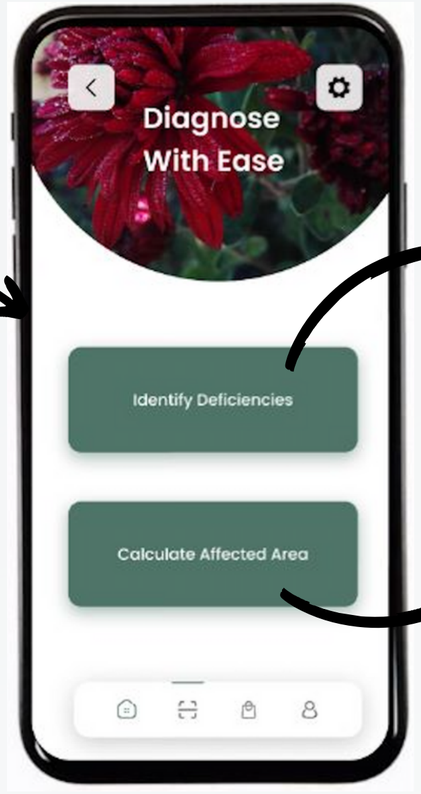
Mask image

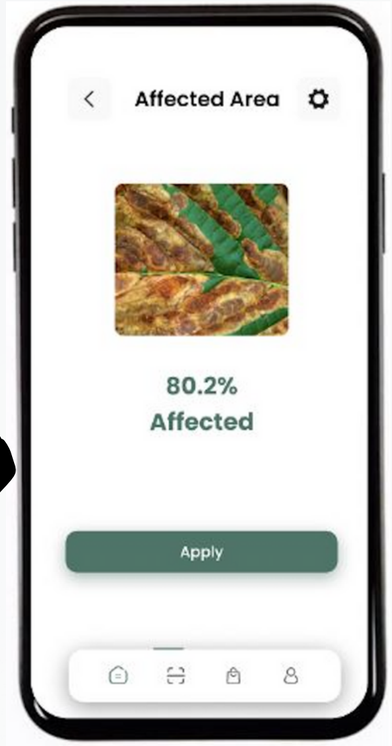


SLIIT
FACULTY OF COMPUTING

IT19994406 | Basnayake N.S.N. | 2023-133

UI DESIGNS





SLIIT
FACULTY OF COMPUTING

IT19994406 | Basnayake N.S.N. | 2023-133

FUTURE WORKS

IMPROVEMENTS

- Differentiate the nutrient deficiencies from mite attacks
- Recommend the shipment for export, local or as the mother plant

TARGETS

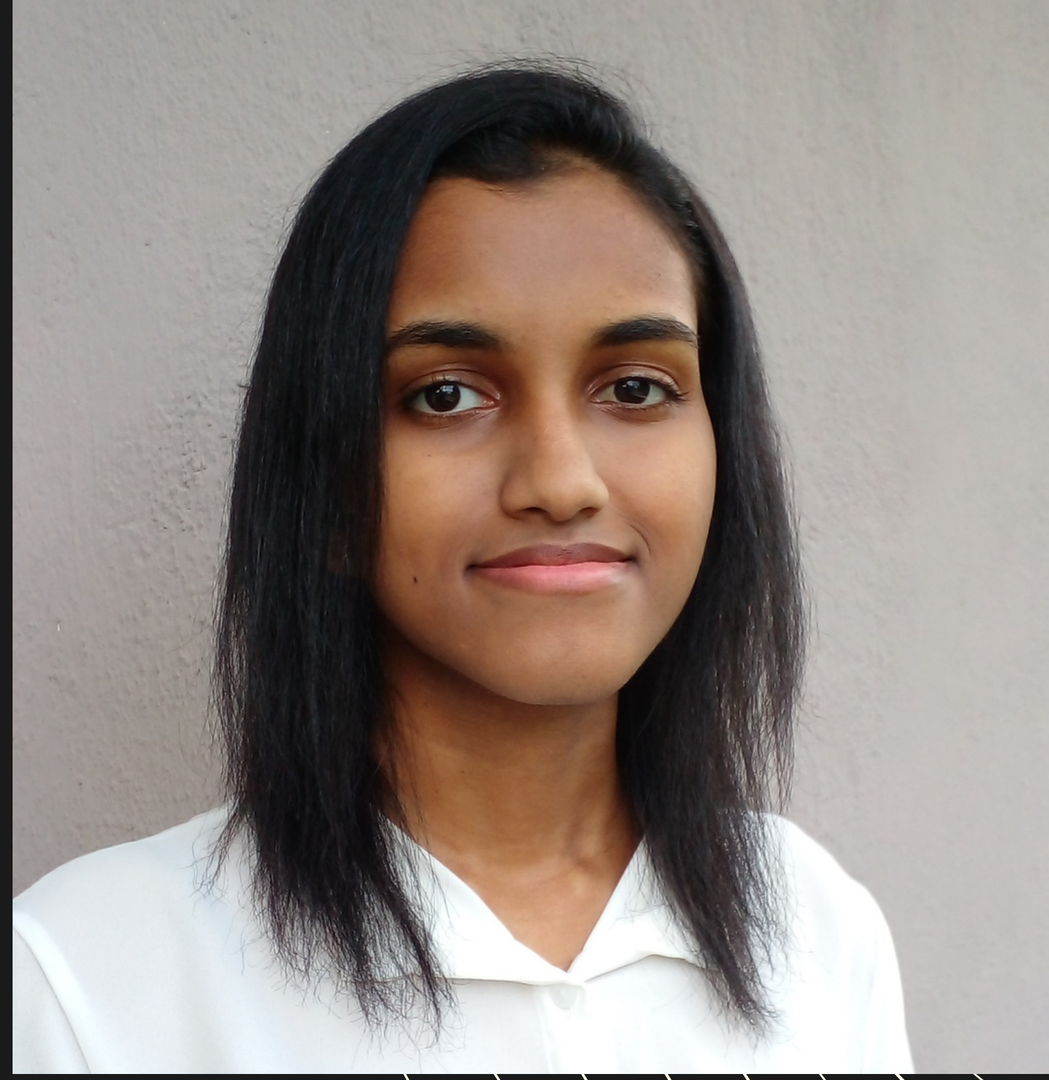
- Implement the frontend
- Build the mobile application for the working function

SLIIT
FACULTY OF COMPUTING

IT19994406 | Basnayake N.S.N. | 2023-133

IT19169736
Gamage M.G.U.D.

SELECTING THE BEST CUT FLOWER PLANT TO GROW BASED ON WEATHER CONDITIONS AND AVAILABLE RESOURCES



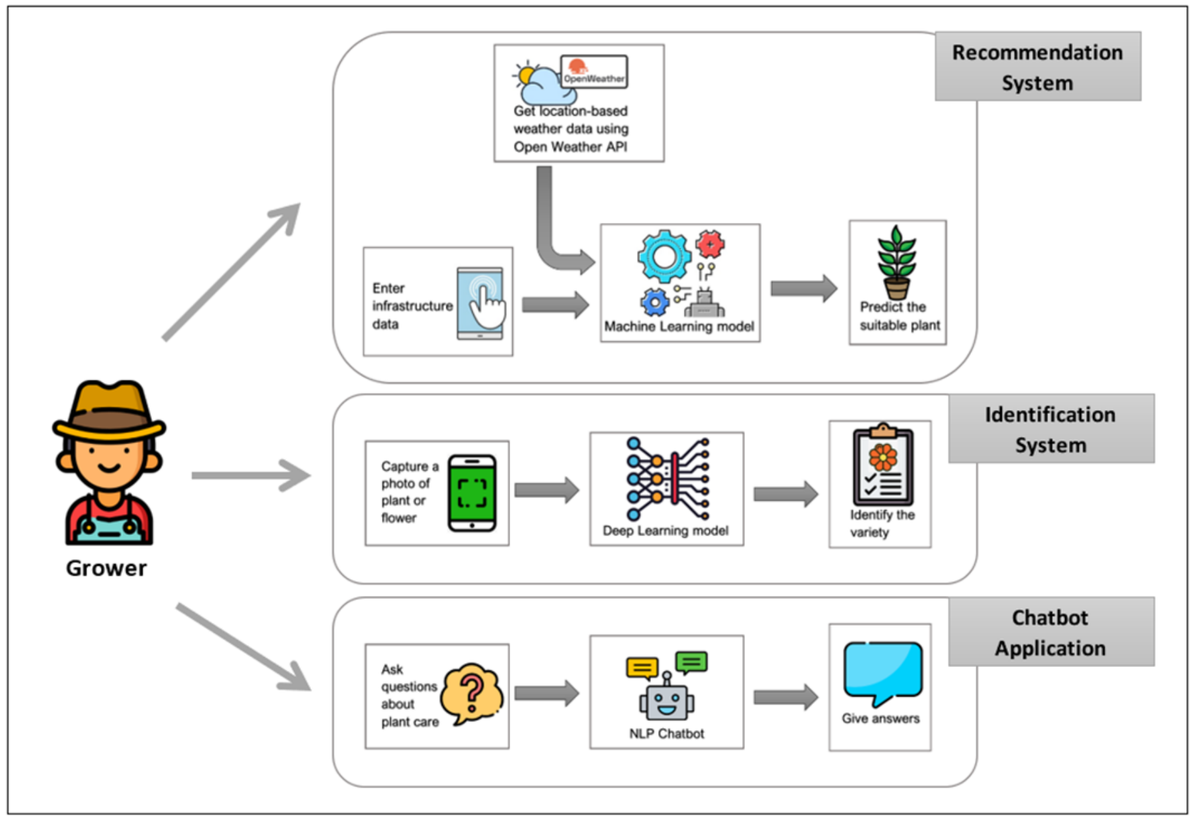
SLIIT
FACULTY OF COMPUTING

Specialization : Data Science

RESEARCH PROBLEM

- How to select the best floriculture plant to grow according to weather and resource factors?
- How to identify varieties of diverse floriculture crops?

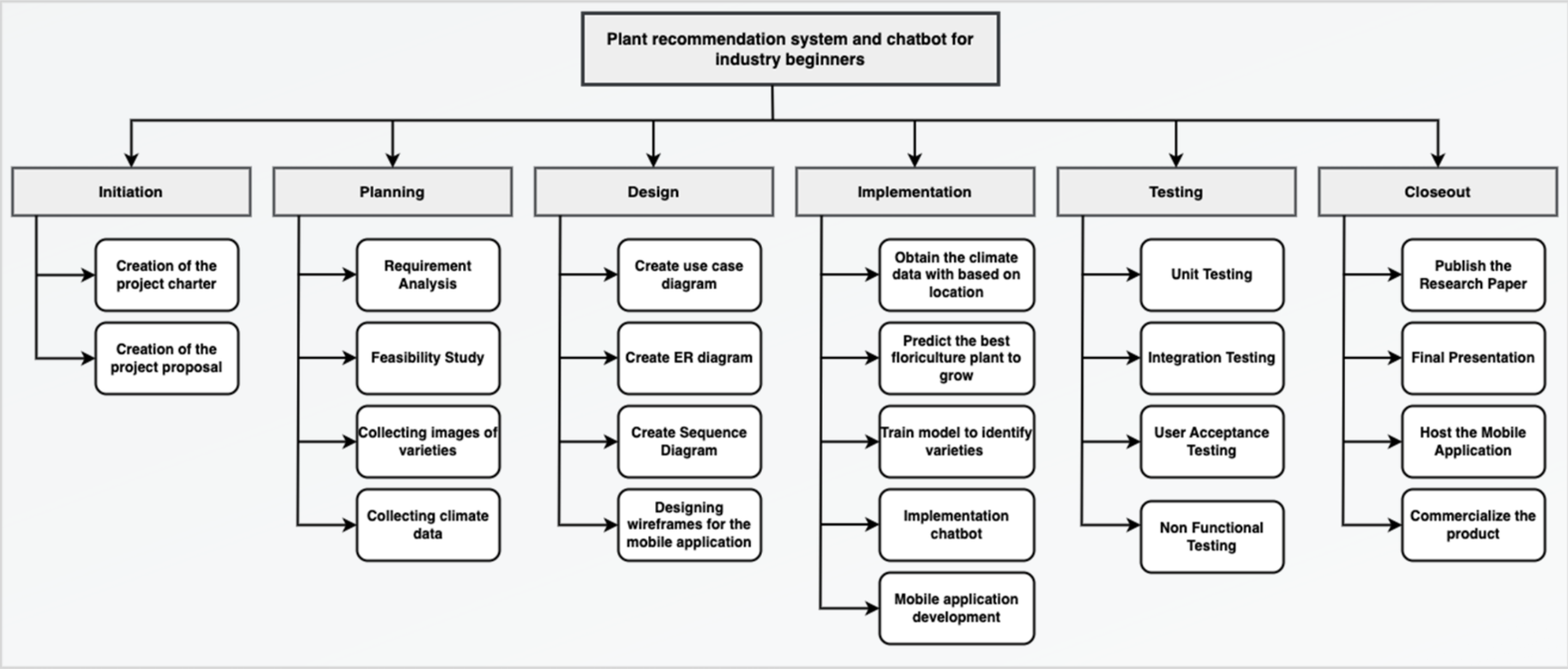
INDIVIDUAL SYSTEM DIAGRAM



SLIIT
FACULTY OF COMPUTING

IT19169736 | Gamage M.G.U.D. | 2023-133

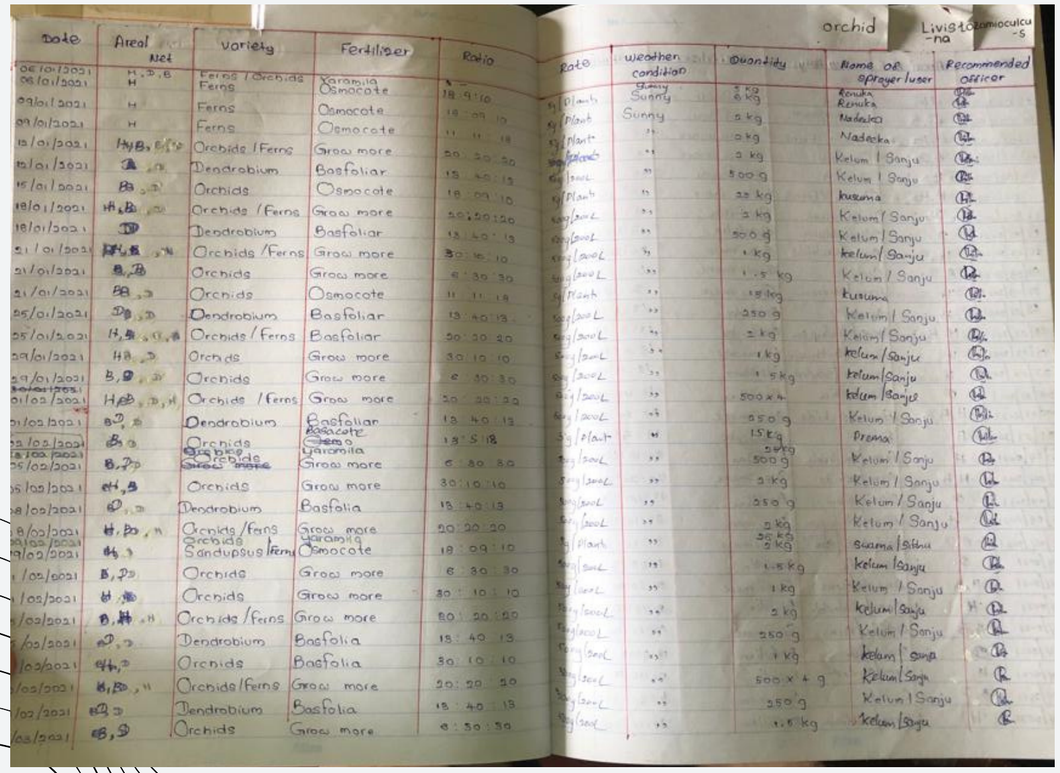
WORK BREAKDOWN



SLIIT
FACULTY OF COMPUTING

IT19169736 | Gamage M.G.U.D. | 2023-133

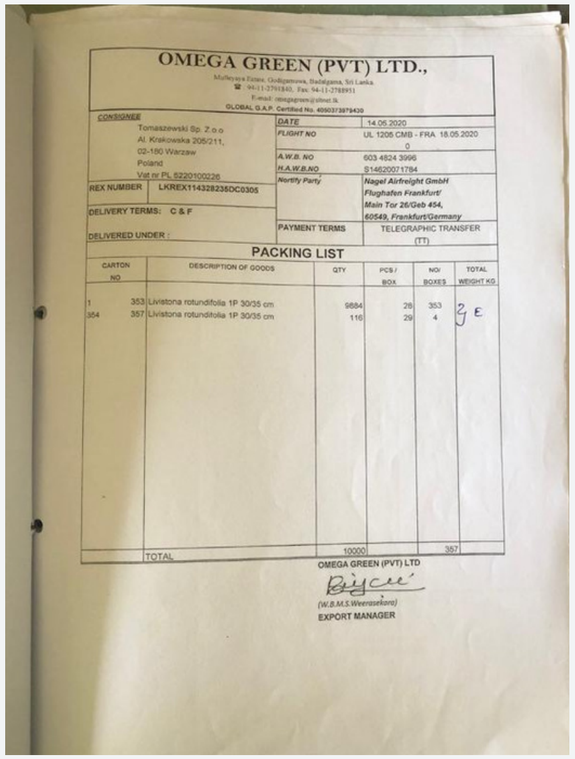
DATA COLLECTING



Date	Areal	Variety	Fertiliser	Ratio
08/01/2021	H,B,S	Ferns / Orchids	Yaramila Osmocote	18:9:10
08/01/2021	H	Ferns	Osmocote	18:09:10
09/01/2021	H	Ferns	Osmocote	11:11:18
13/01/2021	H,B	Orchids / Ferns	Grow more	30:30:30
13/01/2021	B	Dendrobium	Basfoliar	13:40:13
15/01/2021	B	Orchids	Osmocote	18:09:10
18/01/2021	H,B	Orchids / Ferns	Grow more	30:20:20
18/01/2021	D	Dendrobium	Basfoliar	13:40:13
21/01/2021	H,B	Orchids / Ferns	Grow more	30:10:10
21/01/2021	B,D	Orchids	Grow more	8:30:30
21/01/2021	B	Orchids	Osmocote	11:11:18
25/01/2021	D	Dendrobium	Basfoliar	13:40:13
25/01/2021	H,B	Orchids / Ferns	Basfoliar	30:20:20
29/01/2021	H,B	Orchids	Grow more	30:10:10
29/01/2021	B,D	Orchids	Grow more	8:30:30
01/02/2021	H,B,D,H	Orchids / Ferns	Grow more	30:20:30
01/02/2021	B,D	Dendrobium	Basfoliar	13:40:13
02/02/2021	B	Orchids	Yaramila	18:5:18
05/02/2021	B,D	Orchids	Grow more	8:30:30
05/02/2021	H,B	Orchids	Grow more	30:10:10
08/02/2021	D	Dendrobium	Basfolia	13:40:13
08/02/2021	H,B,H	Orchids / Ferns	Grow more	30:20:20
09/02/2021	H	Orchids Sandupsus	Osmocote	18:09:10
12/02/2021	B,D	Orchids	Grow more	8:30:30
15/02/2021	H	Orchids	Grow more	30:10:10
18/02/2021	B,H	Orchids / Ferns	Grow more	30:20:20
22/02/2021	D	Dendrobium	Basfolia	13:40:13
22/02/2021	H	Orchids	Basfolia	30:10:10
22/02/2021	H,B	Orchids / Ferns	Grow more	30:20:20
22/02/2021	B,D	Dendrobium	Basfolia	13:40:13
22/02/2021	B	Orchids	Grow more	8:30:30

orchid — Livistozamioculcu -s -na

Date	weather condition	Quantity	Name of sprayer/user	Recommended officer
01/Plants	Sunny	5 kg	Renuka	
01/Plant	Sunny	5 kg	Nadeeka	
01/Plant	"	2 kg	Nadeeka	
01/Plant	"	2 kg	Kelum / Sanju	
01/Plant	"	500 g	Kelum / Sanju	
01/Plant	"	22 kg	kusuma	
01/Plant	"	2 kg	Kelum / Sanju	
01/Plant	"	500 g	Kelum / Sanju	
01/Plant	"	1 kg	kelum / Sanju	
01/Plant	"	1.5 kg	Kelum / Sanju	
01/Plant	"	15 kg	kusuma	
01/Plant	"	250 g	Kelum / Sanju	
01/Plant	"	2 kg	Kelum / Sanju	
01/Plant	"	1 kg	kelum / Sanju	
01/Plant	"	1.5 kg	kelum / Sanju	
01/Plant	"	500 x 4	kelum / Sanju	
01/Plant	"	250 g	Kelum / Sanju	
01/Plant	H	15 kg	Prema	
01/Plant	"	500 g	Kelum / Sanju	
01/Plant	"	2 kg	Kelum / Sanju	
01/Plant	"	250 g	Kelum / Sanju	
01/Plant	"	2 kg	Kelum / Sanju	
01/Plant	"	1.5 kg	kelum / Sanju	
01/Plant	"	1 kg	Kelum / Sanju	
01/Plant	"	2 kg	kelum / Sanju	
01/Plant	"	250 g	Kelum / Sanju	
01/Plant	"	1 kg	kelum / Sanju	
01/Plant	"	500 x 4 g	kelum / Sanju	
01/Plant	"	250 g	Kelum / Sanju	
01/Plant	"	1.5 kg	kelum / Sanju	



OMEGA GREEN (PVT) LTD.,

Mulleriyawa Estate, Godigamuwa, Badalgama, Sri Lanka.
☎ 94-11-2791840, Fax: 94-11-2788951
E-mail: omegagreen@sltnet.lk
GLOBAL G.A.P. Certified No. 4050373879420

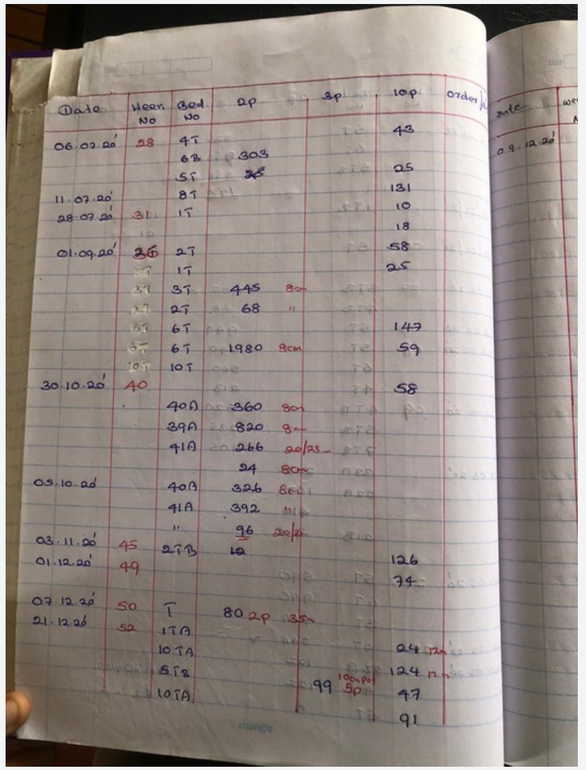
CONSIGNEE: Tomaszewski Sp. Z.o.o, Al. Krakowska 205/211, 02-180 Warzaw, Poland, Vat nr PL 8220100226

DATE: 14.05.2020
FLIGHT NO: UL 1205 CMB - FRA 18.05.2020 / 0
A.W.B. NO: 603 4824 3996
H.A.W.B.NO: S14620021784
Notify Party: Nagel Airfreight GmbH, Flughafen Frankfurt/ Main Tor 26/Geb 454, 60549, Frankfurt/Germany
REX NUMBER: LKREX114328235DC0305
DELIVERY TERMS: C & F
PAYMENT TERMS: TELEGRAPHIC TRANSFER (TT)
DELIVERED UNDER :

PACKING LIST

CARTON NO		DESCRIPTION OF GOODS	QTY	PCS / BOX	NO: BOXES	TOTAL WEIGHT KG
1	353	Livistona rotundifolia 1P 30/35 cm	9884	28	353	2 E
354	357	Livistona rotundifolia 1P 30/35 cm	116	29	4	
TOTAL			10000		357	

OMEGA GREEN (PVT) LTD
(W.B.M.S.Weerasekara)
EXPORT MANAGER



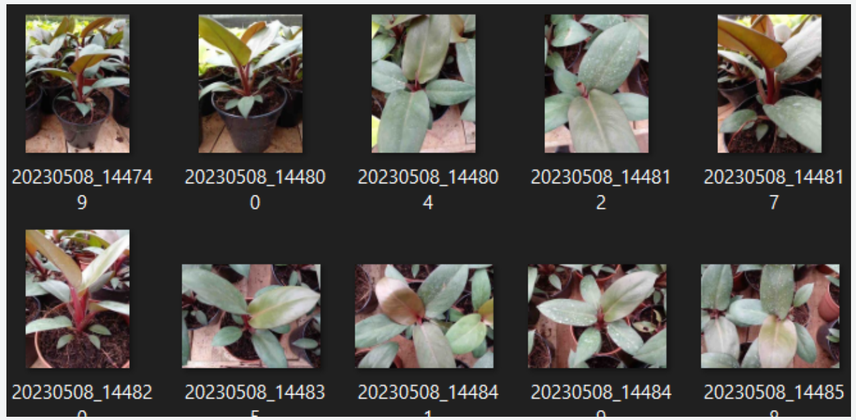
Date	Heen No	Bed No	2p	3p	10p	order
06.07.20'	28	4T			43	
		6T	303			
		5T	35		25	
		8T			131	
11.07.20'					10	
28.07.20'	31	1T			18	
					58	
01.09.20'	36	2T			25	
		1T				
		3T	445	8cm		
		2T	68	"		
		6T			147	
		6T	1980	8cm	59	
		10T				
30.10.20'	40				58	
		40A	360	8cm		
		39A	820	8cm		
		41A	266	20/25		
			24	8cm		
03.10.20		40A	326	8cm		
		41A	392	10		
		"	96	20/25		
03.11.20'	45	27B	12			
01.12.20'	49				126	
					74	
07.12.20'	50	T				
21.12.20'	52	1TA	80	2p 35cm		
		10TA			24	
		5TB			124	
		10TA		99 10cm per 5p	47	
					91	

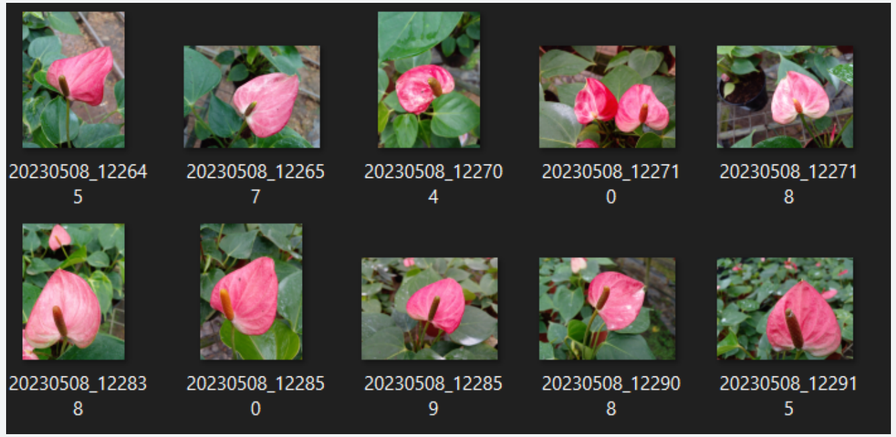
Written records about expenses

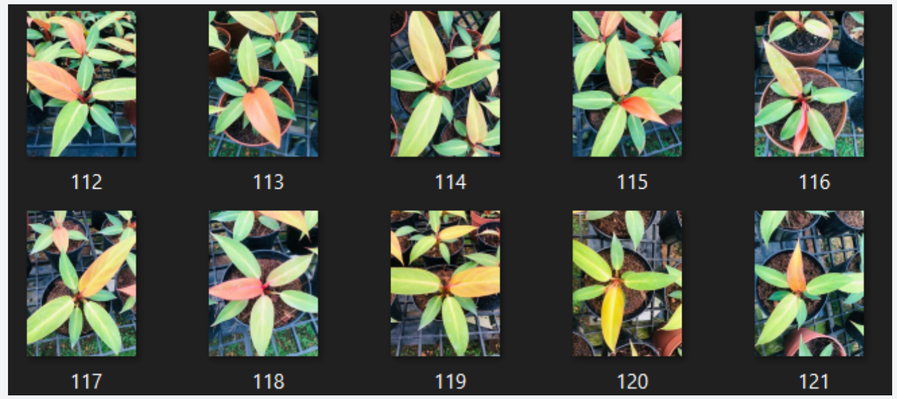
SLIIT FACULTY OF COMPUTING

IT19169736 | Gamage M.G.U.D. | 2023-133

DATA COLLECTING







Images taken from nurseries

SLIIT
FACULTY OF COMPUTING

IT19169736 | Gamage M.G.U.D. | 2023-133

DATA CLEANING

```
# handling null values
df.isna().sum()
```

```
Temperature                        1
Relative_Humidity                  0
Water_liters_per_yr                0
Land_size_in_sqft                  1
Nethouse_land_preperation_cost     1
No_of_plants                       2
Planting_cost                      1
Maintaining_cost                   0
Labour_cost                        1
Flower_yield_per_yr                1
Crop_name                          1
dtype: int64
```

```
# drop null values
df = df.dropna()
df.isna().sum()
```

```
Temperature                        0
Relative_Humidity                  0
Water_liters_per_yr                0
Land_size_in_sqft                  0
Nethouse_land_preperation_cost     0
No_of_plants                       0
Planting_cost                      0
Maintaining_cost                   0
Labour_cost                        0
Flower_yield_per_yr                0
Crop_name                          0
dtype: int64
```

Drop rows that contain null values

DATA SUMMARIZING

```
df.shape
```

```
(2379, 11)
```

```
df.dtypes
```

```
Temperature                       float64
Relative_Humidity                   int64
Water_liters_per_yr                 int64
Land_size_in_sqft                 float64
Nethouse_land_preperation_cost    float64
No_of_plants                      float64
Planting_cost                     float64
Maintaining_cost                    int64
Labour_cost                       float64
Flower_yield_per_yr               float64
Crop_name                          object
dtype: object
```

```
df.head(10)
```

	Temperature	Relative_Humidity	Water_liters_
0	30.9	74	
1	30.4	78	

```
df.describe()
```

	Temperature	Relative_Humidity	Wa
count	2379.000000	2379.000000	
mean	22.388146	80.076923	
std	5.017900	8.065671	
min	13.000000	60.000000	
25%	17.200000	74.000000	

```
df.groupby('Crop_name').size()
```

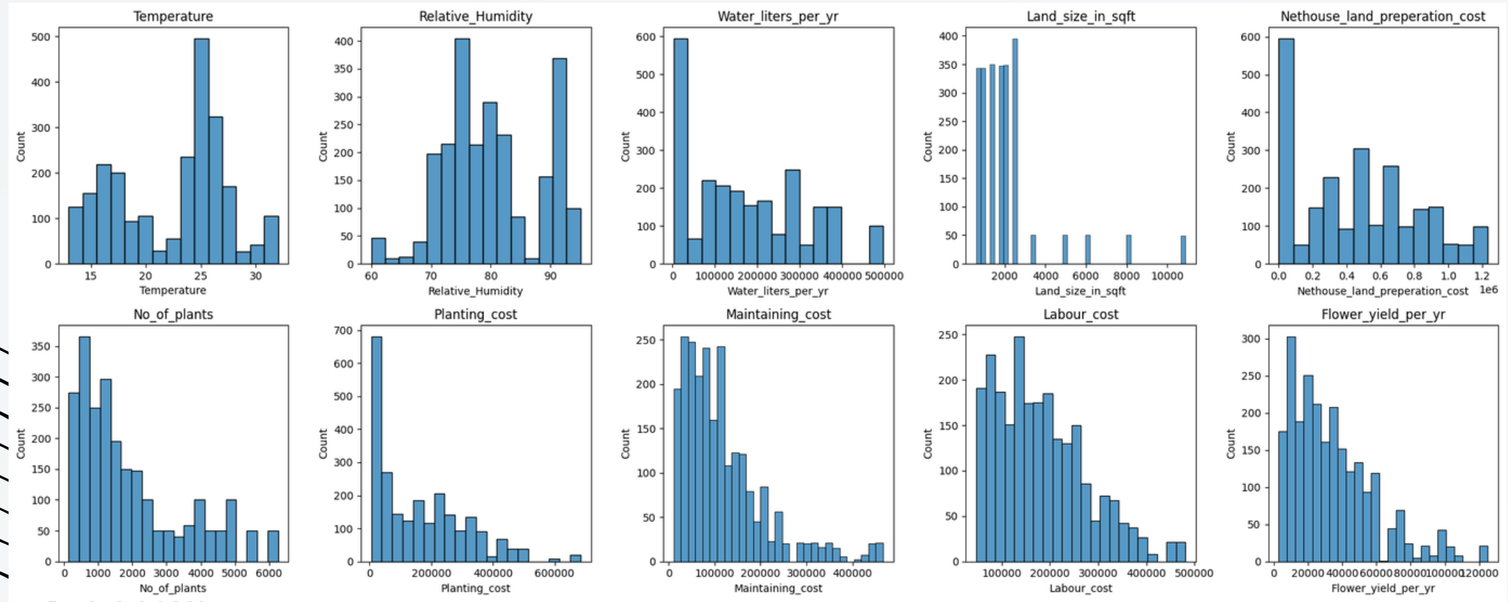
```
Crop_name
Alstroemeria      299
Anthurium         294
Carnation         300
Chrysanthemum     296
Gerbera           298
Lily              295
Orchid            300
Rose              297
dtype: int64
```

Gain insights about dimensions, basic statistical properties and each class's size of the dataset.

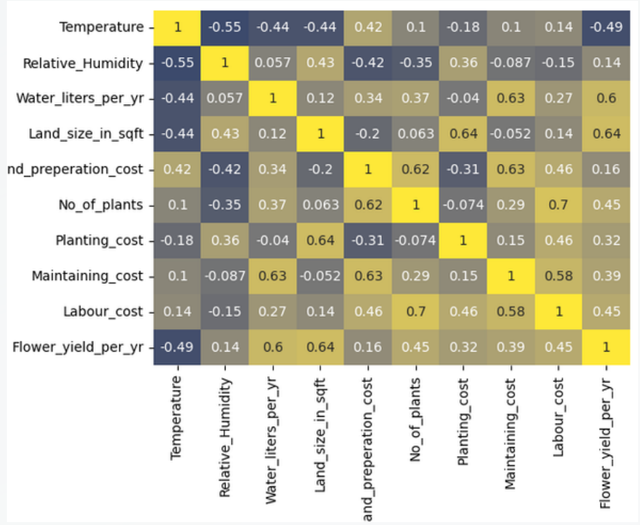
SLIIT
FACULTY OF COMPUTING

IT19169736 | Gamage M.G.U.D. | 2023-133

DATA VISUALIZING



Histograms of features

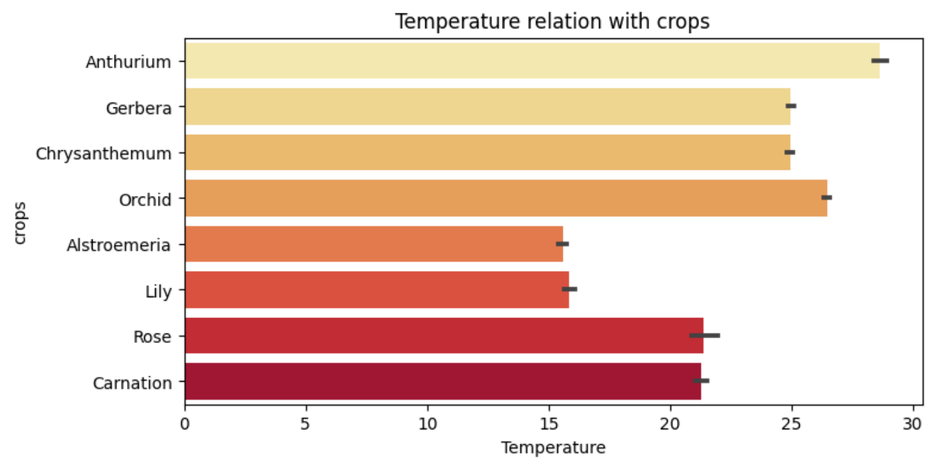


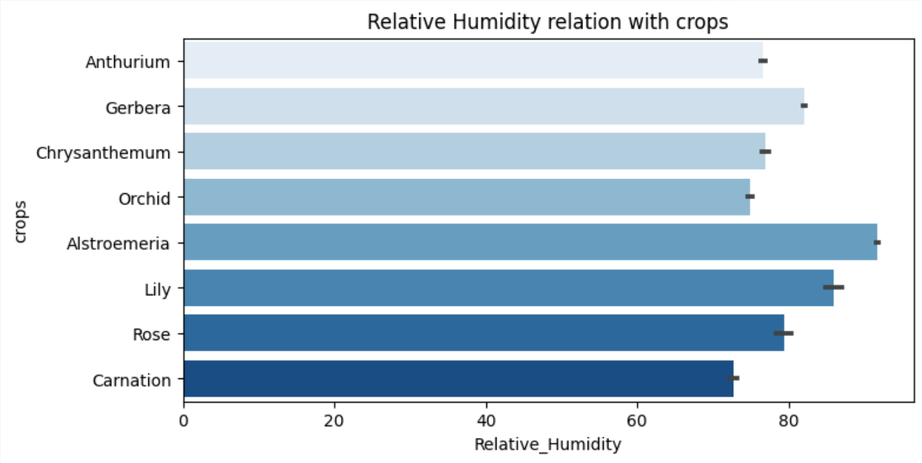
Correlation Matrix

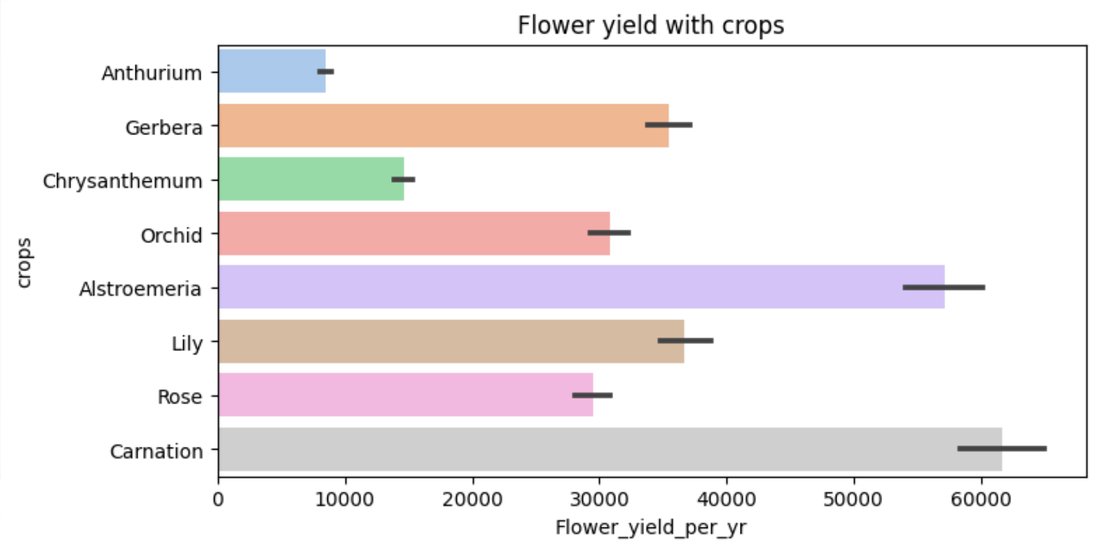
SLIIT
FACULTY OF COMPUTING

IT19169736 | Gamage M.G.U.D. | 2023-133

DATA VISUALIZING







Temperature, Relative Humidity, Flower yield relation with crops

SLIIT FACULTY OF COMPUTING

IT19169736 | Gamage M.G.U.D. | 2023-133

DATA PREPROCESSING

Select features and target values

```
# Features Selection
selected_features = {'Temperature', 'Relative_Humidity', 'Water_liters_per_yr',
       'Land_size_in_sqft', 'Nethouse_land_preperation_cost', 'No_of_plants',
       'Planting_cost', 'Maintaining_cost', 'Labour_cost',
       'Flower_yield_per_yr'}
```

```
# feature values and target values
y = df['Crop_name']
X = df[selected_features]
```

Split the dataset into training & testing sets with 33% of the data allocated for testing

```
# data splitting
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=2)
```

MODEL EVALUATION & COMPARISON

3 classification models are defined:

1. Decision Tree Classifier
2. Gaussian Naive Bayes
3. Gradient Boosting Classifier

```
# Spot Check Algorithms
models = []
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('GB', GradientBoostingClassifier()))
```

MODEL EVALUATION & COMPARISON

For each model Stratified K-Fold cross-validation with 3 folds is set up

It evaluates the model's performance using accuracy as the scoring metric

```
# evaluate each model in turn
results = []
names = []
for name, model in models:
 kfold = StratifiedKFold(n_splits=3, random_state=2, shuffle=True)
 cv_results = cross_val_score(model, X_train, y_train, cv=kfold, scoring='accuracy')
 results.append(cv_results)
 names.append(name)
 print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
```

MODEL EVALUATION & COMPARISON

Mean accuracy across the cross-validation folds = 0.997489

standard deviation as a measure of the performance consistency = 0.002349

```
CART: 0.997489 (0.002349)
NB: 0.916510 (0.017007)
GB: 0.997489 (0.002349)
```

CART & GB both shows high level of accuracy and consistency

MODEL TRAINING

Gradient Boosting Classifier is selected as the best model

```
model = GradientBoostingClassifier(n_estimators=100, learning_rate=0.001)

model.fit(X_train, y_train)
```

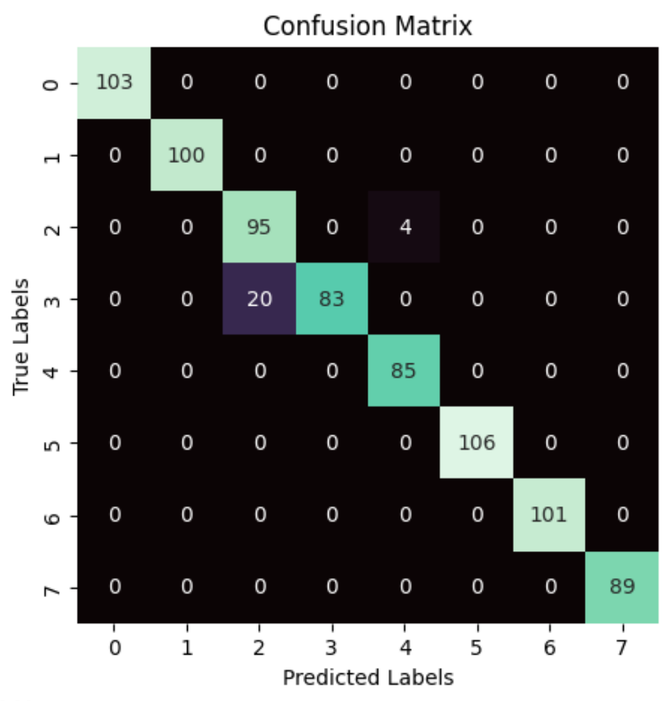
```
# make predictions on test dataset
predictions = model.predict(X_test)
```

Accuracy of 96.95% on the test dataset

```
# evaluate predictions
print("Accuracy : ", accuracy_score(y_test, predictions))
```

```
Accuracy :  0.9694656488549618
```

EVALUATION METRICS



Confusion Matrix

```
print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
Alstroemeria	1.00	1.00	1.00	103
Anthurium	1.00	1.00	1.00	100
Carnation	0.83	0.96	0.89	99
Chrysanthemum	1.00	0.81	0.89	103
Gerbera	0.96	1.00	0.98	85
Lily	1.00	1.00	1.00	106
Orchid	1.00	1.00	1.00	101
Rose	1.00	1.00	1.00	89
accuracy			0.97	786
macro avg	0.97	0.97	0.97	786
weighted avg	0.97	0.97	0.97	786

Classification Report

SLIIT
FACULTY OF COMPUTING

IT19169736 | Gamage M.G.U.D. | 2023-133

OBTAIN WEATHER DATA

OpenWeather API to retrieve Temperature & Relative Humidity for a specific city

```python
# Parameters
api_key = "71d1945ed6fb6c764f9334a8b0c1dc4a"
city_name = "Malabe"

# Construct the API call
url = f'http://api.openweathermap.org/data/2.5/weather?q={city_name}&appid={api_key}'
print(url)

# Send the request
response = requests.get(url)

# Handle the response
if response.status_code == 200:
    data = json.loads(response.text)
    temperature_kelvin = data['main']['temp']
    temperature_celsius = temperature_kelvin - 273.15
    humidity = data['main']['humidity']
    print(f"Temperature: {temperature_celsius:.2f} °C")
    print(f"Humidity: {humidity}%")
else:
    print("Error:", response.status_code)
```

```
http://api.openweathermap.org/data/2.5/weather?q=Malabe&appid=71d1945ed6fb6c764f9334a8
Temperature: 28.94 °C
Humidity: 75%
```

SLIIT
FACULTY OF COMPUTING

IT19169736 | Gamage M.G.U.D. | 2023-133

PREDICT THE BEST CROP

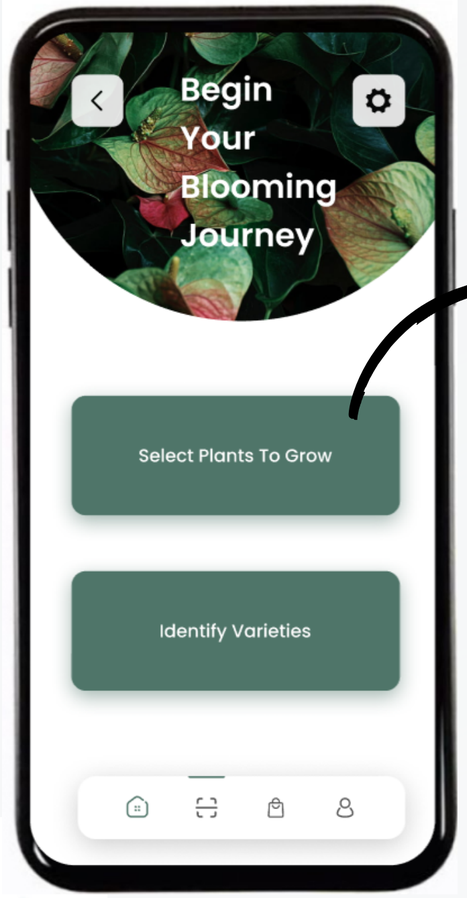
- Temperature
- Relative Humidity
- No of plants
- Flower yield
- Land and water
- Planting, maintain, nethouse, labor cost

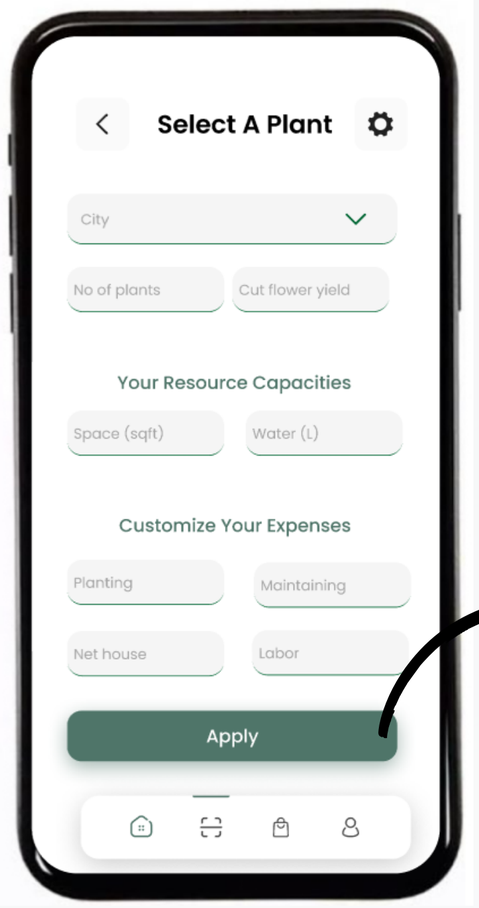
are fed into the model to get predictions

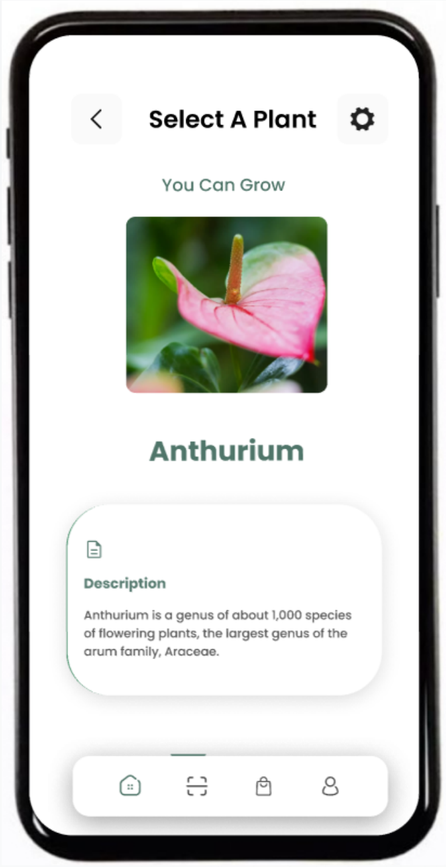
```
# predictions on the input array
prediction = pickled_model.predict(X_new)

# Print the predictions
print("The best crop to grow :", prediction[0])
```

```
The best crop to grow : Lily
```

UI DESIGNS







SLIIT
FACULTY OF COMPUTING

IT19169736 | Gamage M.G.U.D. | 2023-133

FUTURE WORKS

Improve the plant selection model

Implement the Frontend

Develop the variety identification

Complete the mobile application development

IT20017088
Prabhshi P.A.N.

FORECAST THE FLUCTUATING DEMAND PATTERNS BASED ON SEASONS AND SPECIFIC PRODUCT PREFERENCES

Specialization : Data Science

SLIIT
FACULTY OF COMPUTING

RESEARCH PROBLEM

How to predict the future demand accurately according to various factors?

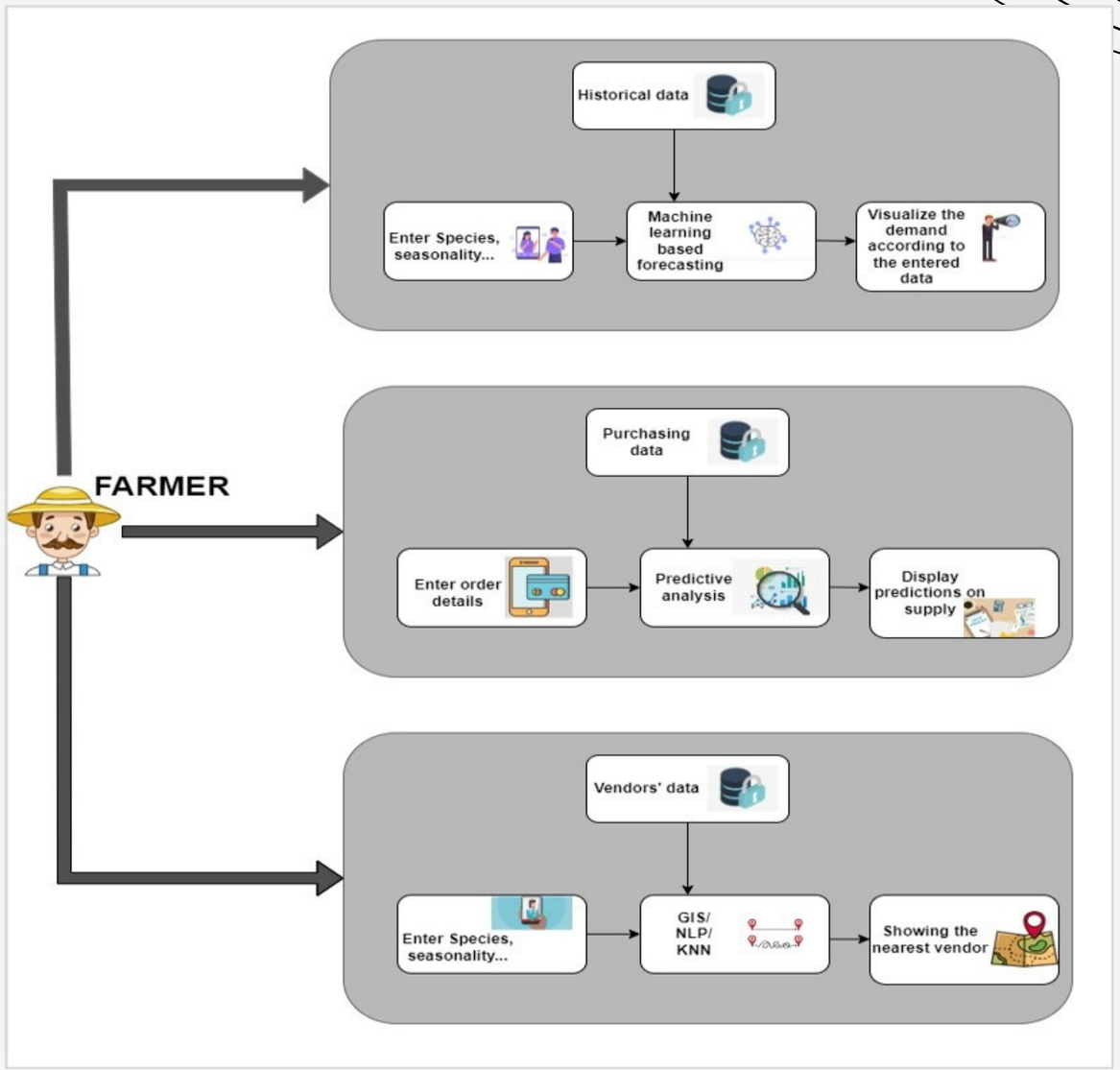
Various factors such as,

- Country / Region
- Season
- Variety

SLIIT
FACULTY OF COMPUTING

IT20017088 | Prabhashi P.A.N. | 2023-133

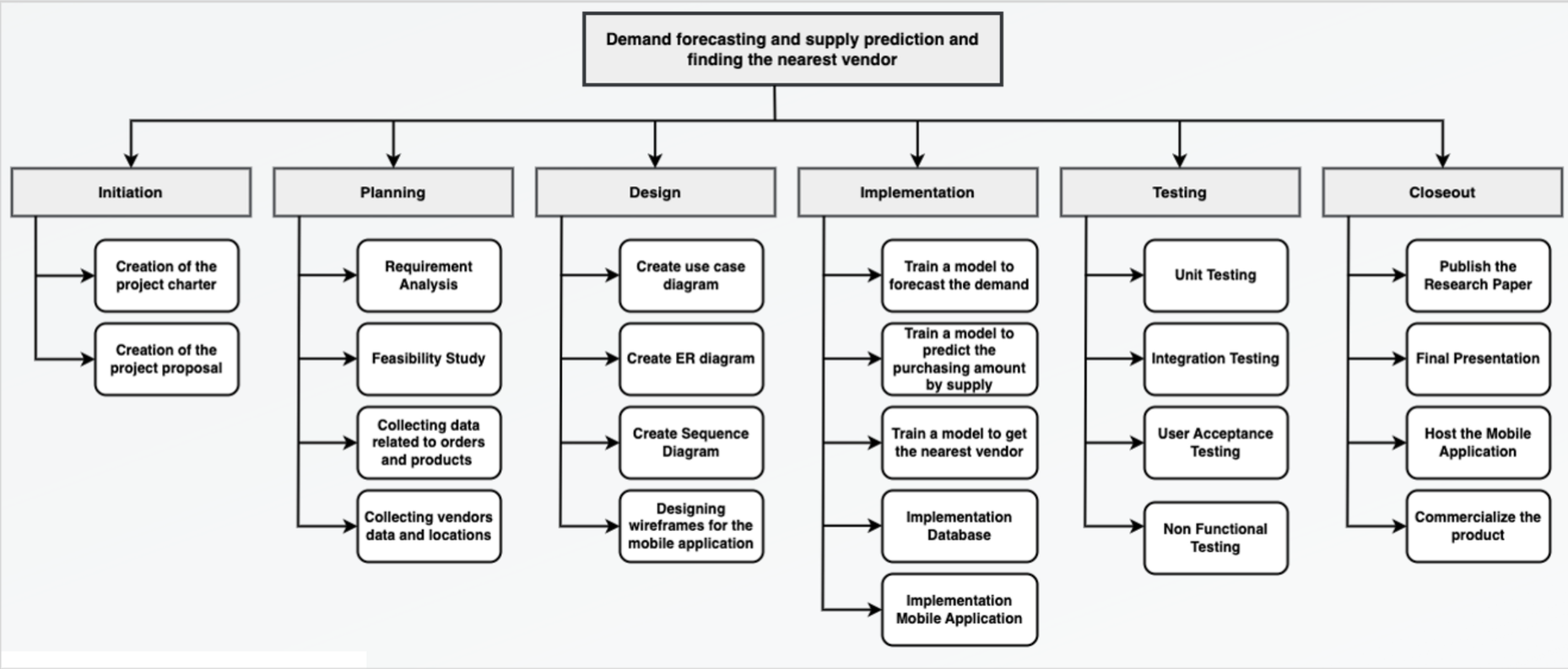
INDIVIDUAL SYSTEM DIAGRAM



SLIIT
FACULTY OF COMPUTING

IT20017088 | Prabhashi P.A.N. | 2023-133

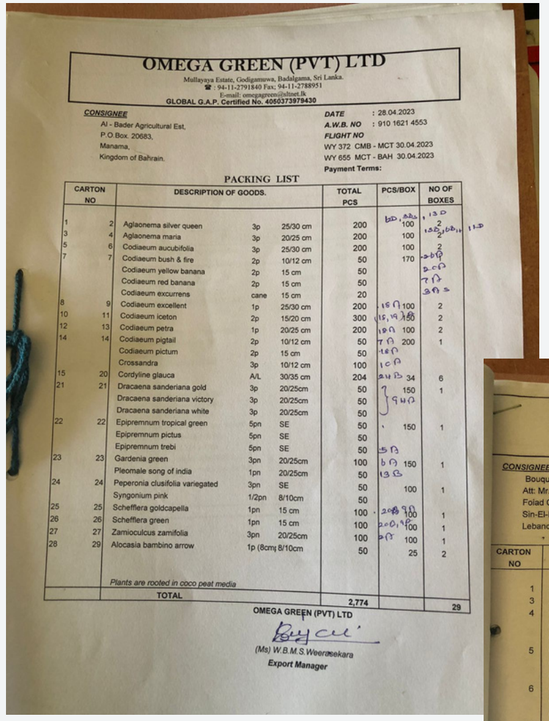
WORK BREAKDOWN



FIELD DATA

These data is collected from the omega green Pvt Ltd

And all data are collected in manual way



OMEGA GREEN (PVT) LTD

Mulleryaya Estate, Godigamuwa, Badalgama, Sri Lanka.
☎ : 94-11-2791840 Fax: 94-11-2788951
E-mail: omegagreen@sltnet.lk
GLOBAL G.A.P. Certified No. 4060373979430

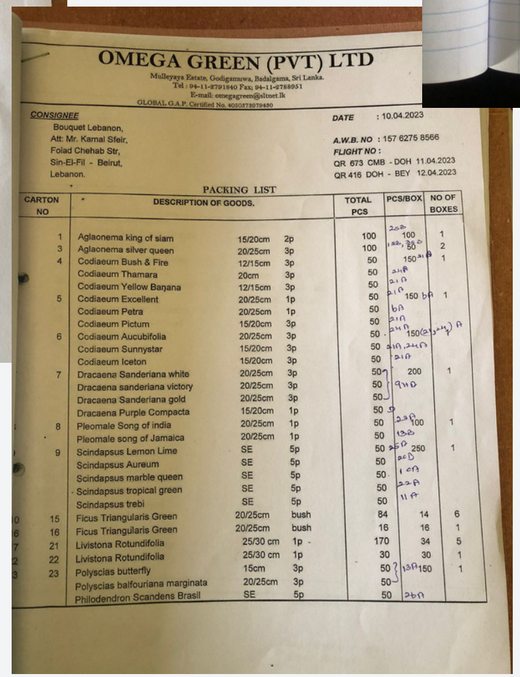
CONSIGNEE
Al - Bader Agricultural Est,
P.O Box. 20683,
Manama,
Kingdom of Bahrain.

DATE : 28.04.2023
A.W.B. NO : 910 1621 4553
FLIGHT NO
WY 372 CMB - MCT 30.04.2023
WY 655 MCT - BAH 30.04.2023
Payment Terms:

PACKING LIST

CARTON NO		DESCRIPTION OF GOODS.			TOTAL PCS	PCS/BOX	NO OF BOXES
1	2	Aglaonema silver queen	3p	25/30 cm	200	100	2
3	4	Aglaonema maria	3p	20/25 cm	200	100	2
5	6	Codiaeum aucubifolia	3p	25/30 cm	200	100	2
7	7	Codiaeum bush & fire	2p	10/12 cm	50	170	
		Codiaeum yellow banana	2p	15 cm	50		
		Codiaeum red banana	2p	15 cm	50		
		Codiaeum excurrens	cane	15 cm	20		
8	9	Codiaeum excellent	1p	25/30 cm	200	100	2
10	11	Codiaeum iceton	2p	15/20 cm	300	100	2
12	13	Codiaeum petra	1p	20/25 cm	200	100	2
14	14	Codiaeum pigtail	2p	10/12 cm	50	200	1
		Codiaeum pictum	2p	15 cm	50		
		Crossandra	3p	10/12 cm	100		
15	20	Cordyline glauca	A/L	30/35 cm	204	34	6
21	21	Dracaena sanderiana gold	3p	20/25cm	50	150	1
		Dracaena sanderiana victory	3p	20/25cm	50		
		Dracaena sanderiana white	3p	20/25cm	50		
22	22	Epipremnum tropical green	5pn	SE	50	150	1
		Epipremnum pictus	5pn	SE	50		
		Epipremnum trebi	5pn	SE	50		
23	23	Gardenia green	3pn	20/25cm	100	150	1
		Pleomale song of india	1pn	20/25cm	50		
24	24	Peperonia clusifolia variegated	3pn	SE	50	100	1
		Syngonium pink	1/2pn	8/10cm	50		
25	25	Scheffera golcapella	1pn	15 cm	100	100	1
26	26	Scheffera green	1pn	15 cm	100	100	1
27	27	Zamioculcus zamifolia	3pn	20/25cm	100	100	1
28	29	Alocasia bambino arrow	1p (8cm)	8/10cm	50	25	2
		Plants are rooted in coco peat media					
		TOTAL			2,774		29

OMEGA GREEN (PVT) LTD
(Ms) W.B.M.S.Weerasekara
Export Manager



OMEGA GREEN (PVT) LTD

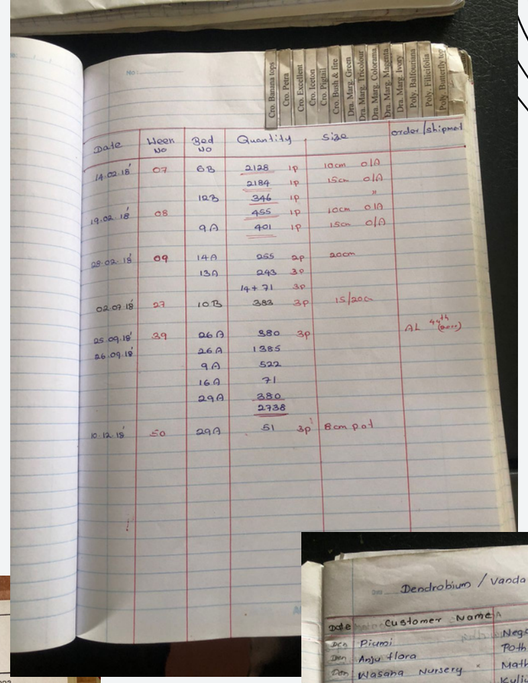
Mulleryaya Estate, Godigamuwa, Badalgama, Sri Lanka.
Tel : 94-11-2791840 Fax: 94-11-2788951
E-mail: omegagreen@sltnet.lk
GLOBAL G.A.P. Certified No. 4060373979430

CONSIGNEE
Bouquet Lebanon,
Att: Mr. Kamal Sfeir,
Folad Chehab Str,
Sin-El-Fil - Beirut,
Lebanon.

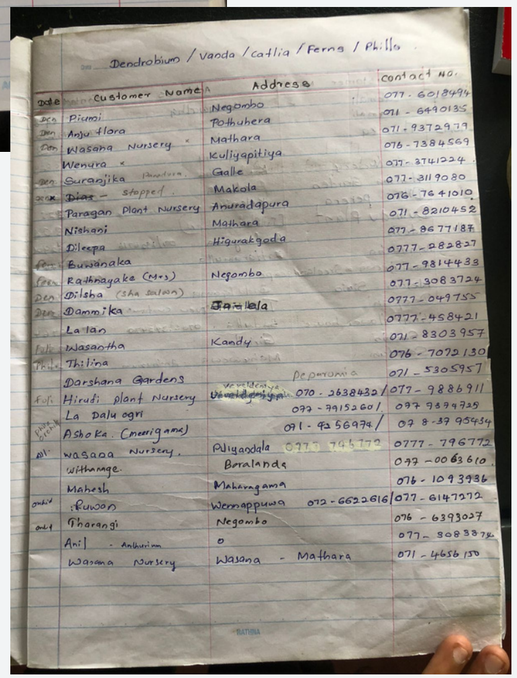
DATE : 10.04.2023
A.W.B. NO : 157 6275 8566
FLIGHT NO :
QR 673 CMB - DOH 11.04.2023
QR 416 DOH - BEY 12.04.2023

PACKING LIST

CARTON NO		DESCRIPTION OF GOODS.			TOTAL PCS	PCS/BOX	NO OF BOXES
	1	Aglaonema king of siam	15/20cm	2p	100	100	1
	3	Aglaonema silver queen	20/25cm	3p	100	100	2
	4	Codiaeum Bush & Fire	12/15cm	3p	50	150	1
		Codiaeum Thamara	20cm	3p	50		
		Codiaeum Yellow Banana	12/15cm	3p	50		
	5	Codiaeum Excellent	20/25cm	1p	50	150	1
		Codiaeum Petra	20/25cm	1p	50		
		Codiaeum Pictum	15/20cm	3p	50		
	6	Codiaeum Aucubifolia	20/25cm	3p	50	150	1
		Codiaeum Sunnystar	15/20cm	3p	50		
		Codiaeum Iceton	15/20cm	3p	50		
	7	Dracaena Sanderiana white	20/25cm	3p	50	200	1
		Dracaena sanderiana victory	20/25cm	3p	50		
		Dracaena Sanderiana gold	20/25cm	3p	50		
		Dracaena Purple Compacta	15/20cm	1p	50		
	8	Pleomale Song of india	20/25cm	1p	50	100	1
		Pleomale song of Jamaica	20/25cm	1p	50		
	9	Scindapsus Lemon Lime	SE	5p	50	250	1
		Scindapsus Aureum	SE	5p	50		
		Scindapsus marble queen	SE	5p	50		
		Scindapsus tropical green	SE	5p	50		
		Scindapsus trebi	SE	5p	50		
10	15	Ficus Triangularis Green	20/25cm	bush	84	14	6
6	16	Ficus Triangularis Green	20/25cm	bush	16	16	1
7	21	Livistona Rotundifolia	25/30 cm	1p	170	34	5
2	22	Livistona Rotundifolia	25/30 cm	1p	30	30	1
3	23	Polyscias butterfly	15cm	3p	50	150	1
		Polyscias balfouriana marginata	20/25cm	3p	50		
		Philodendron Scandens Brasil	SE	5p	50		



Date	Item No	Bed No	Quantity	Size	order/shipment
14.02.18	07	6B	2128 1p	10cm o/A	
			2184 1p	15cm o/A	
19.02.18	08	12B	346 1p	"	
			455 1p	10cm o/A	
		9A	401 1p	15cm o/A	
20.02.18	09	14A	265 3p	20cm	
		13A	293 3p		
			147 3p		
02.07.18	27	10B	383 3p	15/20cm	
25.09.18	39	26A	880 3p		AL 4th (berry)
26.09.18		26A	1385		
		9A	522		
		16A	71		
		29A	380		
			2938		
10.12.18	50	29A	51 3p	8cm pot	



Dendrobium / Vanda / cattla / Ferns / Philo

Date	Customer Name	Address	Contact No:
Jan	Pismi	Negombo	077 - 6018494
Jan	Anju Flora	Polhuhera	071 - 6491035
Jan	Wasana Nursery	Mathara	071 - 9372977
Jan	Wenura "	Kuliyapitiya	076 - 7384569
Jan	Suranjika Anthurium	Galle	077 - 3741224
Jan	Dias - stopped	Makola	077 - 3119080
Jan	Paragan Plant Nursery	Anuradapura	076 - 7641010
Feb	Nishani	Mathara	071 - 8210452
Feb	Dileepa	Higurakgoda	077 - 9677187
Feb	Buwanaka		0777 - 282827
Feb	Rathnayake (Mrs)	Negombo	077 - 9814438
Feb	Dilsha (Isha Sri Lanka)		077 - 3083724
Feb	Dammika	Jaela	0777 - 049755
Feb	Lalan		0777 - 458421
Feb	Wasantha	Kandy	071 - 8303957
Feb	Thilina		076 - 7072130
Feb	Darshana Gardens	Peperiyagoda	071 - 5305957
Feb	Hiruti plant Nursery	Dewalegama	070 - 3638432 / 077 - 9886911
Feb	La Dalla agri		072 - 7915201 / 077 9899735
Feb	Ashoka (Imerigama)		071 - 92 56974 / 07 8-37 954544
Mar	Wasana Nursery	Piliyandala	0777 - 796772
Mar	Withange	Boralande	077 - 0063610
Mar	Mahesh	Maharagama	076 - 1093936
Mar	Sluvan	Wennappuwa 072 - 6622616	077 - 6147272
Mar	Dharangi	Negombo	076 - 6393027
Mar	Anil - Anthurium	o	077 - 9083878
Mar	Wasana Nursery	Wasana - Mathara	071 - 4656150

SLIIT FACULTY OF COMPUTING

IT20017088 | Prabhashi P.A.N. | 2023-133

IMPLEMENTATION

Data Preprocessing

DATA PREPROCESSING

Identify the null values

```
[11]  1 # Handle missing values
      2 # Check for missing values
      3 Demand_df.isnull().sum()
```

```
County / Region      0
Year                 0
Season               0
Variety              0
Quantity           613
1 pcs                0
Price                0
dtype: int64
```

Selecting the mean / median to use for fill

```
[12]  1 column_quantity = Demand_df['Quantity'].mean()
```

```
[13]  1 column_quantity
```

```
1674070.7251551538
```

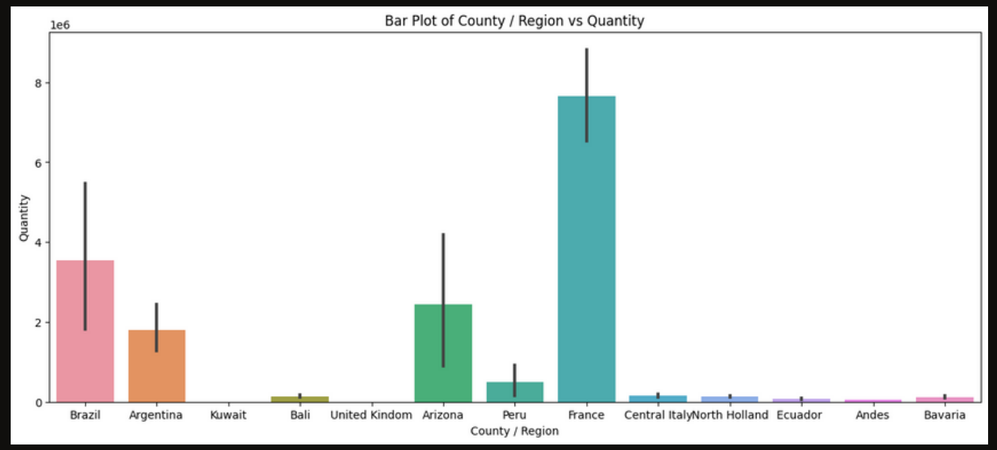
```
[14]  1 column_quantity_median = Demand_df['Quantity'].median()
```

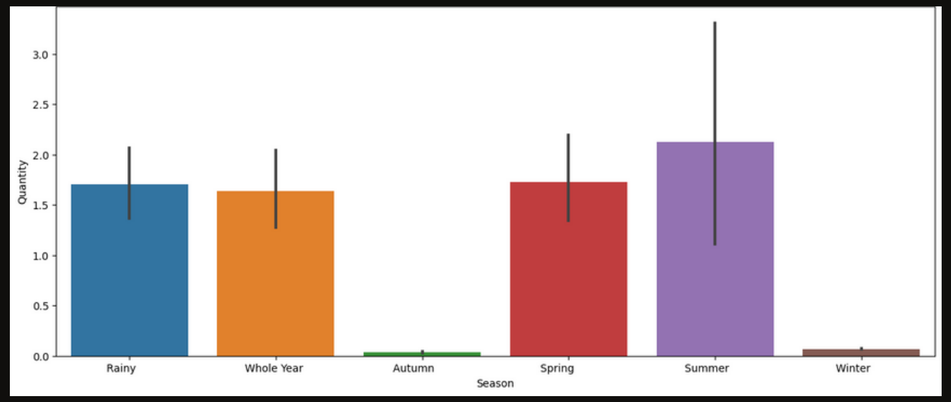
```
[15]  1 column_quantity_median
```

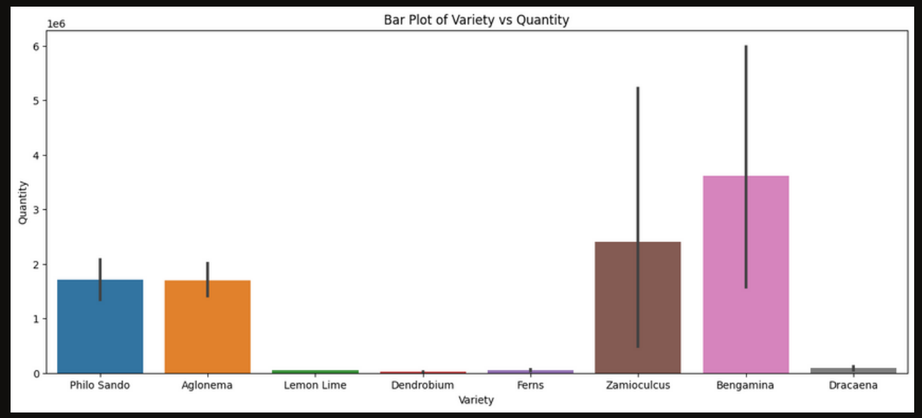
```
1043.0
```

IMPLEMENTATION

Data Visualizing with target





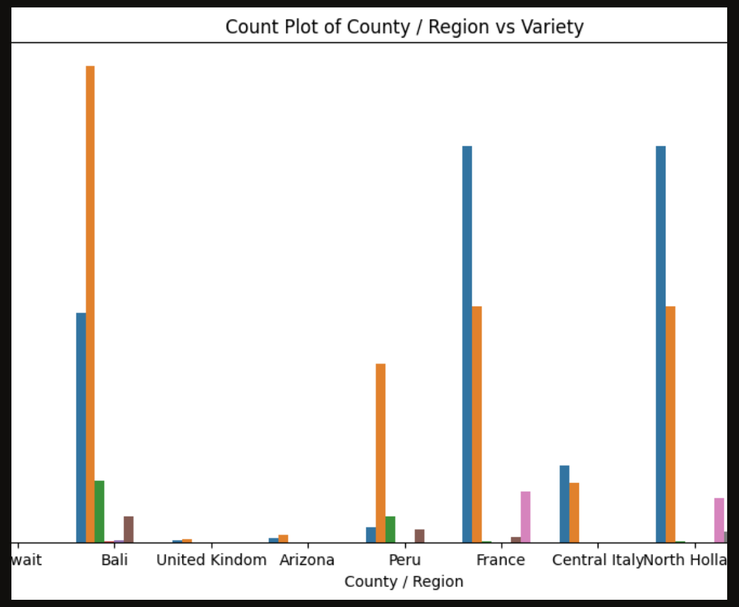


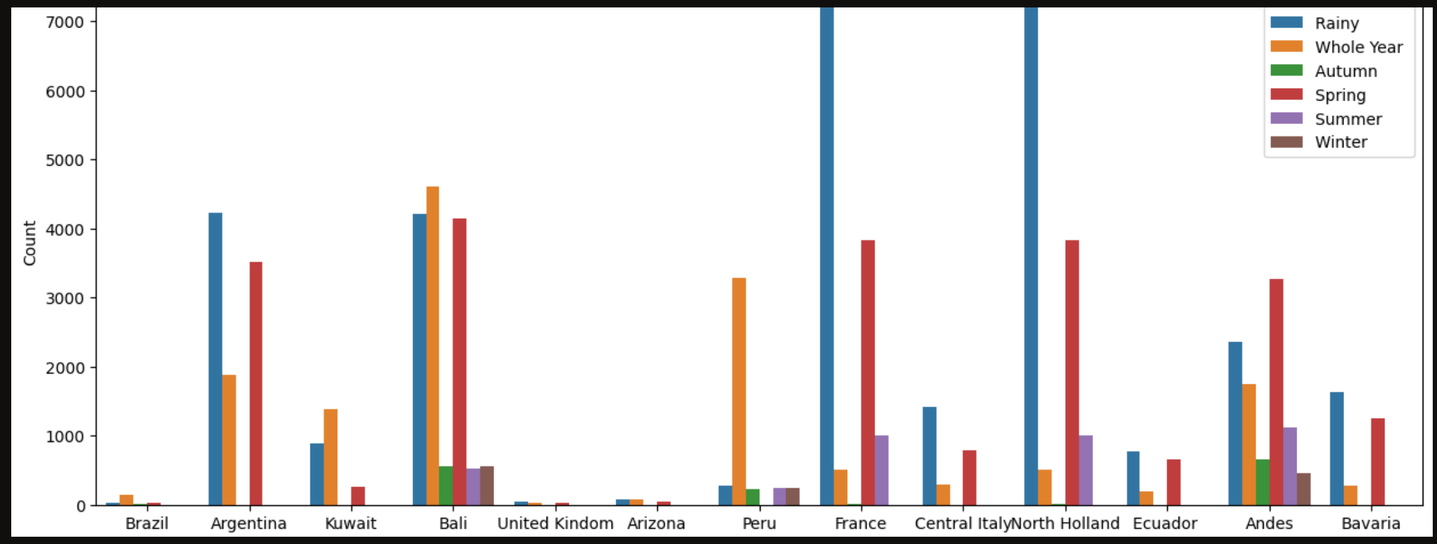
SLIIT
FACULTY OF COMPUTING

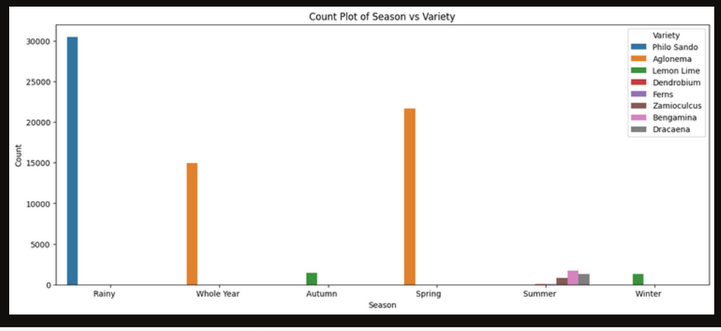
IT20017088 | Prabhashi P.A.N. | 2023-133

IMPLEMENTATION

Data Visualizing with each features







SLIIT
FACULTY OF COMPUTING

IT20017088 | Prabhashi P.A.N. | 2023-133

IMPLEMENTATION

Data Preprocessing

Numerical data scaled using the standard scaler and the catgorical variables handled by one hot encoding

Handling numerical data using Standard Scaler

```
[38]  1 # Extract the "Quantity" column
      2 quantity_data = Demand_Filled_df_Qty["Quantity"].values.reshape(-1, 1)

[39]  1 # Initia
      2 scaler =

[40]  1 # Fit ar
      2 scaled_d

[41]  1 # Create
      2 scaled_d
```

ing dummy values for categorical variables

```
1 # Create dummy variables for the categorical columns
2 dummy_columns = pd.get_dummies(Demand_Filled_df_cat[categorical_columns], prefix=categorical_columns, drop_first=T
```

```
1 dummy_columns
```

	C_County / Region_Argentina	C_County / Region_Arizona	C_County / Region_Bali	C_County / Region_Bavaria	C_County / Region_Brazil	C_County / Region_Central Italy	C_County / Region_Ecuador
0	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0
2	0	0	0	0	1	0	0
3	0	0	0	0	1	0	0
4	0	0	0	0	1	0	0

IMPLEMENTATION

Splitting data in to x & y

The data set has split in to two as independent and dependent variables by 20% of the data taken as the test size

Splitting training and testing data set

```
[63]  1 X = shuffled_df.drop("Scaled_Quantity", axis=1)
      2 y = shuffled_df["Scaled_Quantity"]
      3 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
```

```
[ ]  1 X
```

	C_County / Region_Argentina	C_County / Region_Arizona	C_County / Region_Bali	C_County / Region_Bavaria	C_County / Region_Brazil	Regi
0	0	0	0	0	0	
1	0	0	0	0	0	
2	1	0	0	0	0	
3	0	0	0	0	0	
4	1	0	0	0	0	
...	...	...	...	...	...	
73729	0	0	0	0	0	
73730	1	0	0	0	0	
73731	0	0	0	0	0	
73732	1	0	0	0	0	
73733	0	0	1	0	0	

```
[ ]  1 y

0       -0.056071
1       -0.056018
2       -0.055635
3       -0.056129
4       -0.056089
           ...
73729   -0.055686
73730   -0.056129
73731   -0.056123
73732   -0.056128
73733   -0.056114
Name: Scaled_Quantity, Length: 73734, dtype: float64
```

IMPLEMENTATION

After checking outliers models are defined I have selected 3 types of regression models to train and they are

1.Linear regression

2.Extreme gradient boosting

3.Random forest

MODEL TRAINING

01.Random forest regression

```
[67]  1 # Random Forest Regression
      2 rf = RandomForestRegressor()
      3 rf.fit(X_train, y_train)
      4 rf_pred_train = rf.predict(X_train)
      5 rf_pred_test = rf.predict(X_test)
      6 rf_mse_train = mean_squared_error(y_train, rf_pred_train)
      7 rf_mse_test = mean_squared_error(y_test, rf_pred_test)
      8 print("Random Forest - Train MSE:", rf_mse_train)
      9 print("Random Forest - Test MSE:", rf_mse_test)
```

```
Random Forest - Train MSE: 0.9526527655989198
Random Forest - Test MSE: 1.13248521470431
```

03.Lineare Regression

```
[73]  1 # Linear Regression
      2 lr = LinearRegression()
      3 lr.fit(X_train, y_train)
      4 lr_pred_train = lr.predict(X_train)
      5 lr_pred_test = lr.predict(X_test)
      6 lr_mse_train = mean_squared_error(y_train, lr_pred_trai
      7 lr_mse_test = mean_squared_error(y_test, lr_pred_test)
      8 print("Linear Regression - Train MSE:", lr_mse_train)
      9 print("Linear Regression - Test MSE:", lr_mse_test)
```

```
Linear Regression - Train MSE: 0.9544264394774482
Linear Regression - Test MSE: 1.1360368381360995
```

02.XGBoost Regression

```
[70]  1 # XGBoost Regression
      2 xgb = XGBRegressor()
      3 xgb.fit(X_train, y_train)
      4 xgb_pred_train = xgb.predict(X_train)
      5 xgb_pred_test = xgb.predict(X_test)
      6 xgb_mse_train = mean_squared_error(y_train, xg
      7 xgb_mse_test = mean_squared_error(y_test, xgb_
      8 print("XGBoost - Train MSE:", xgb_mse_train)
      9 print("XGBoost - Test MSE:", xgb_mse_test)
```

```
XGBoost - Train MSE: 0.9526493973594272
XGBoost - Test MSE: 1.1325992861367322
```

IMPLEMENTATION

After model evaluation part according to MAE, MSE and R squared values Extreme Gradient Boost is the best model for this

Model	Train MSE	Test MSE	Train MAE	Test MAE	Train R-squared	Test R-square
Linear Regression	0.954426	1.13604	0.112089	0.120399	0.00936312	0.0088274
XGBoost	0.952649	1.1326	0.104581	0.112726	0.0112076	0.0118267
Random Forest	0.952653	1.13249	0.104738	0.112885	0.0112041	0.0119262

And accuracy is also calculated for all three models

Additionaly done the cross validation parts for avoid overfitting as a part of hyper parameter tuning

IMPLEMENTATION

Accuracy for each model

```
[87]  1 threshold = 0.1  # Define the threshold for considering predictions as accurate
      2
      3 # Calculate the percentage of predictions within the threshold
      4 accurate_predictions = np.abs(y_test - lr_pred_test) <= threshold
      5 accuracy_percentage = round((np.sum(accurate_predictions) / len(y_test)) * 100, 2)
      6
      7 print("Linear Regression - Accuracy (% within threshold):", accuracy_percentage, "%")

  Linear Regression - Accuracy (% within threshold): 81.95 %
```

```
[83]  1 # Calculate the percentage of predictions within the threshold
      2 accurate_predictions = np.abs(y_test - xgb_pred_test) <= threshold
      3 accuracy_percentage = round((np.sum(accurate_predictions) / len(y_test)) * 100, 2)
      4
      5 print("XGBoost - Accuracy (% within threshold):", accuracy_percentage, "%")

  XGBoost - Accuracy (% within threshold): 79.63 %
```

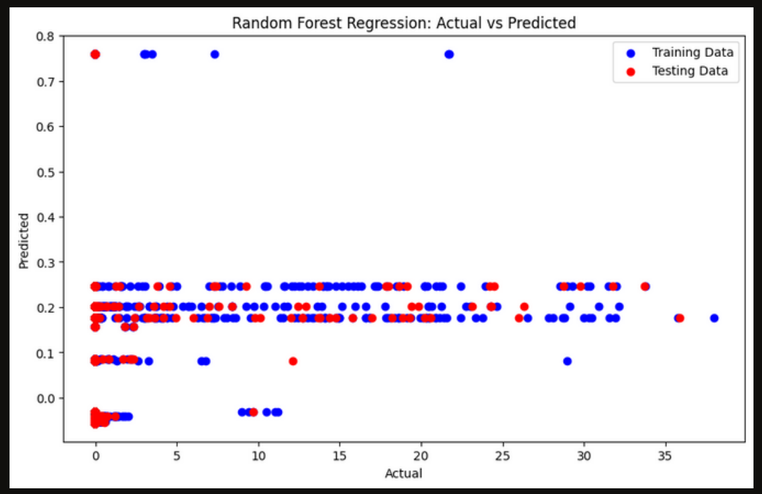
```
[89]  1 # Calculate the percentage of predictions within the threshold
      2 accurate_predictions = np.abs(y_test - rf_pred_test) <= threshold
      3 accuracy_percentage = round((np.sum(accurate_predictions) / len(y_test)) * 100, 2)
      4
      5 print("Random Forest - Accuracy (% within threshold):", accuracy_percentage, "%")

    Random Forest - Accuracy (% within threshold): 79.63 %
```

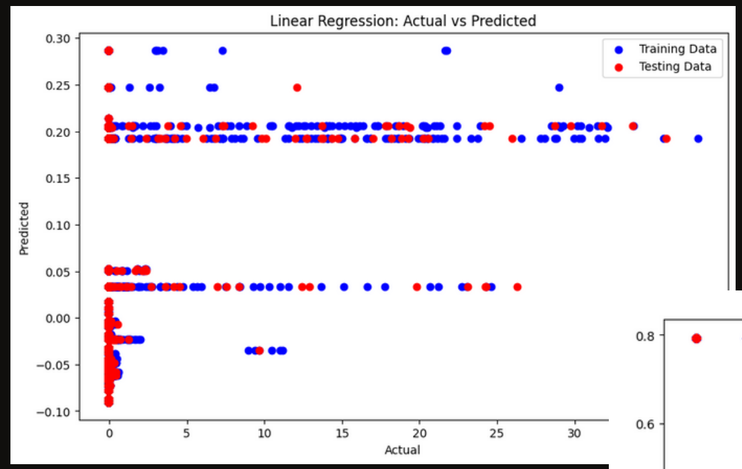
```
[79]  1 # XGBoost Regression
      2 xgb = XGBRegressor()
      3 xgb_pipeline = make_pipeline(SimpleImputer
      4 xgb_cv_scores = cross_val_score(xgb_pipeli
      5 xgb_cv_mse = -np.mean(xgb_cv_scores)
      6 print("XGBoost - Cross-validated MSE:", xgb

  XGBoost - Cross-validated MSE: 0.9538598053789
```

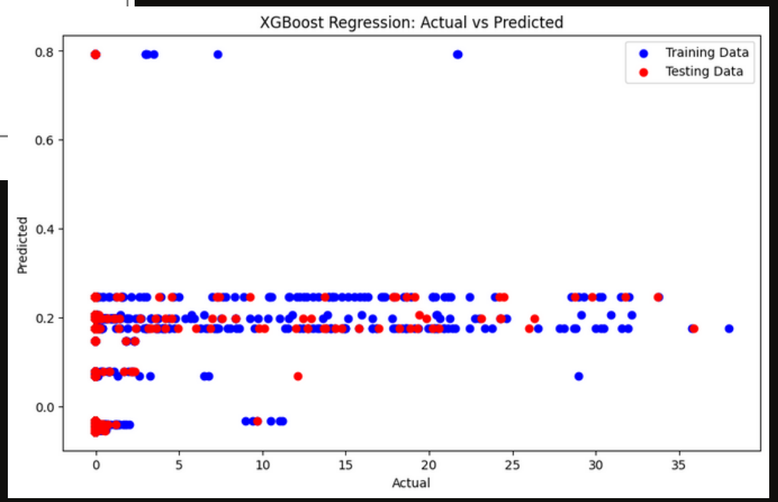
```
[80]  1 # Cross-validation for Linear Regression
      2 lr_cv_scores = cross_val_score(lr, X_train,
      3 lr_cv_mse = -np.mean(lr_cv_scores)
      4 print("Linear Regression - Cross-validated

    Linear Regression - Cross-validated MSE: 0.954
```

IMPLEMENTATION

Scatter plots for actual and predicted



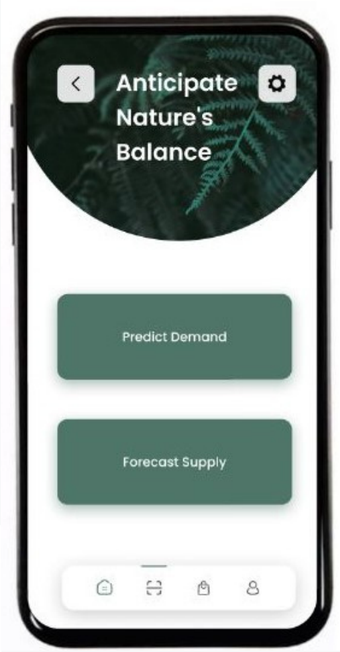


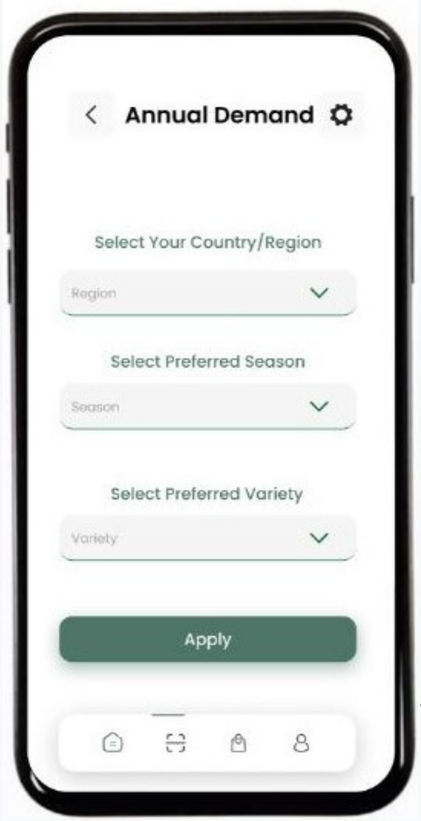


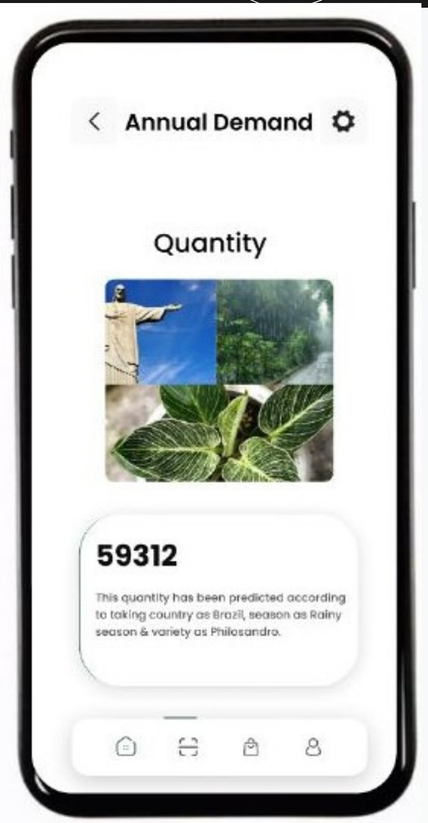
SLIIT
FACULTY OF COMPUTING

IT20017088 | Prabhashi P.A.N. | 2023-133

UI







SLIIT
FACULTY OF COMPUTING

IT20017088 | Prabhashi P.A.N. | 2023-133

FUTURE

DEVELOPING THE

IMPROVE THE CURRENT MODEL AND APPLY THE SUPPLY PREDICTION

DEVELOPING THE MODEL TO FIND THE NEAREST VENDOR

DEVELOP THE FRONT END AND INTEGRATE THE APPLICATION

IT20005726
Liyanage S.R

GROWTH MONITORING AND PREDICTION SYSTEM FOR ORNAMENTAL PLANTS

SLIIT
FACULTY OF COMPUTING

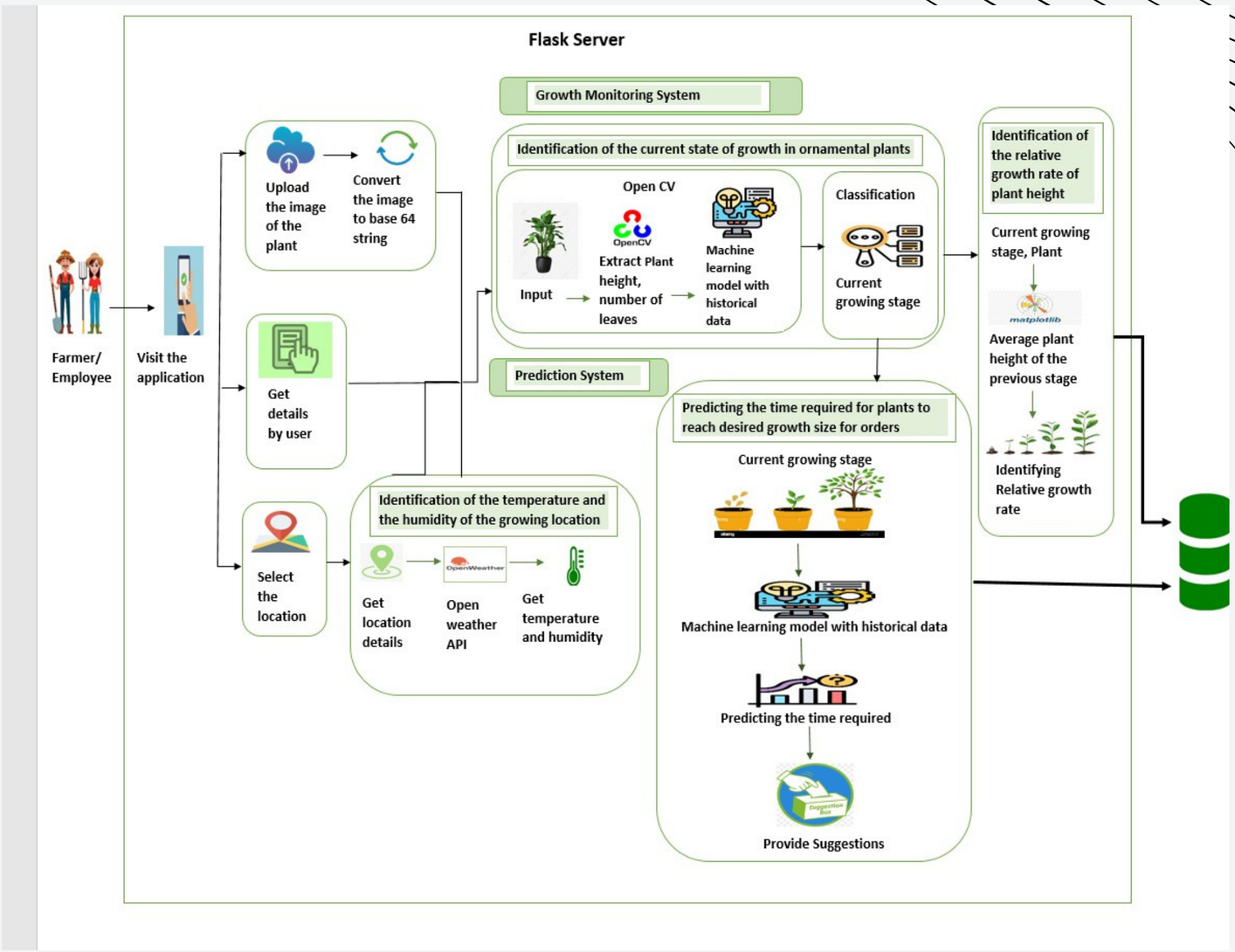
Specialization : Data Science

RESEARCH PROBLEM

- How can the growth of ornamental plants be monitored, and their current state of growth predicted?

- How can the time it takes for ornamental plants to reach the required growth size for an order be predicted?

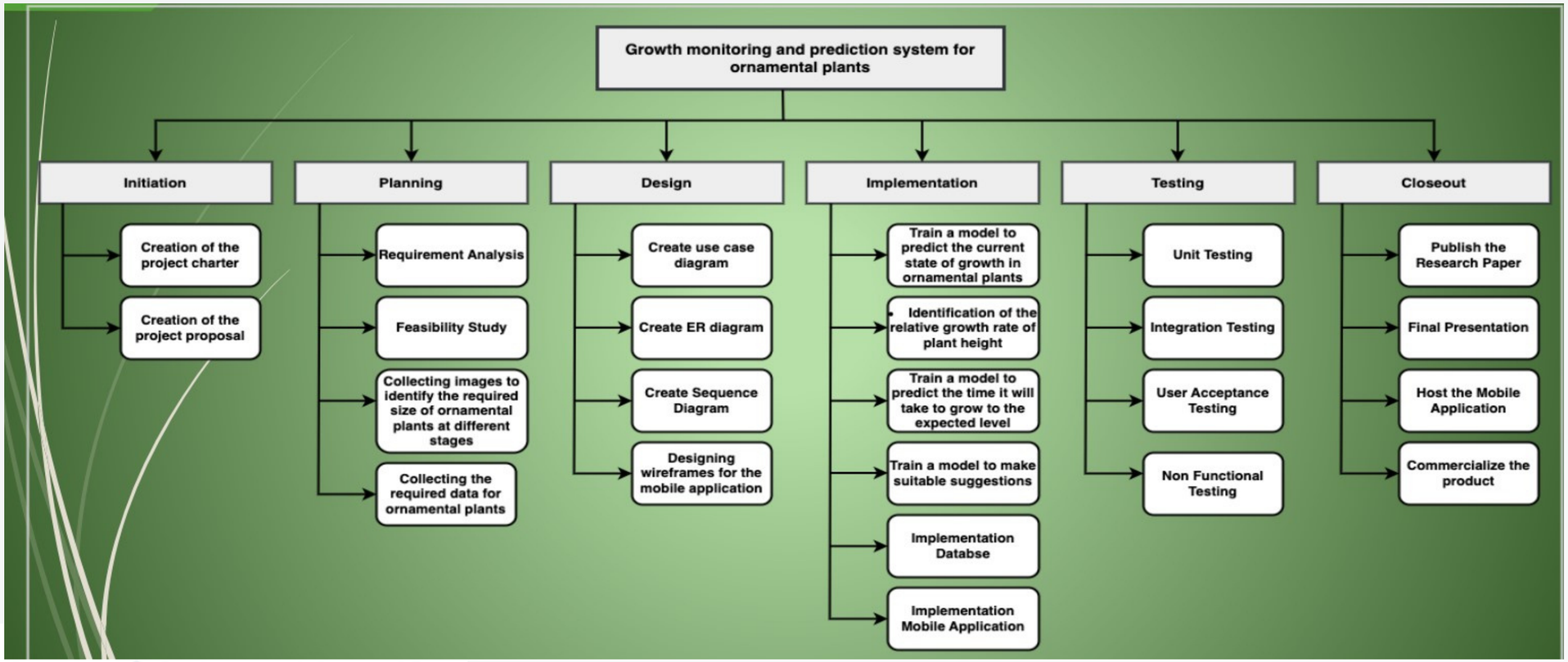
INDIVIDUAL SYSTEM DIAGRAM



SLIIT FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

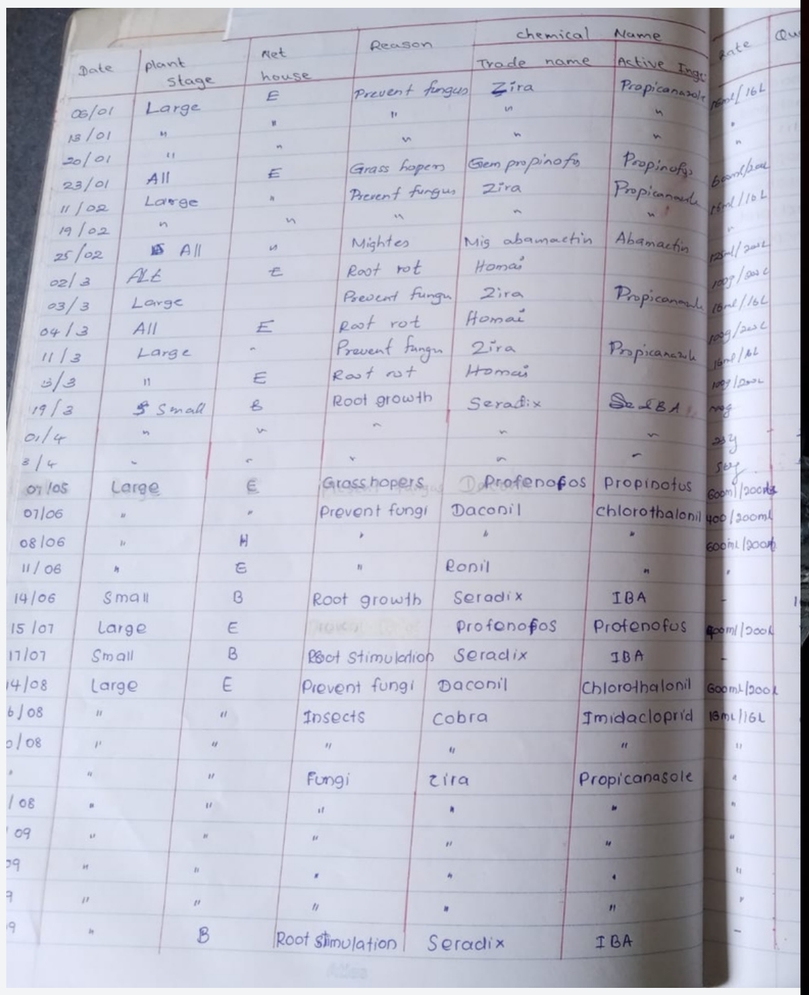
WORK BREAKDOWN



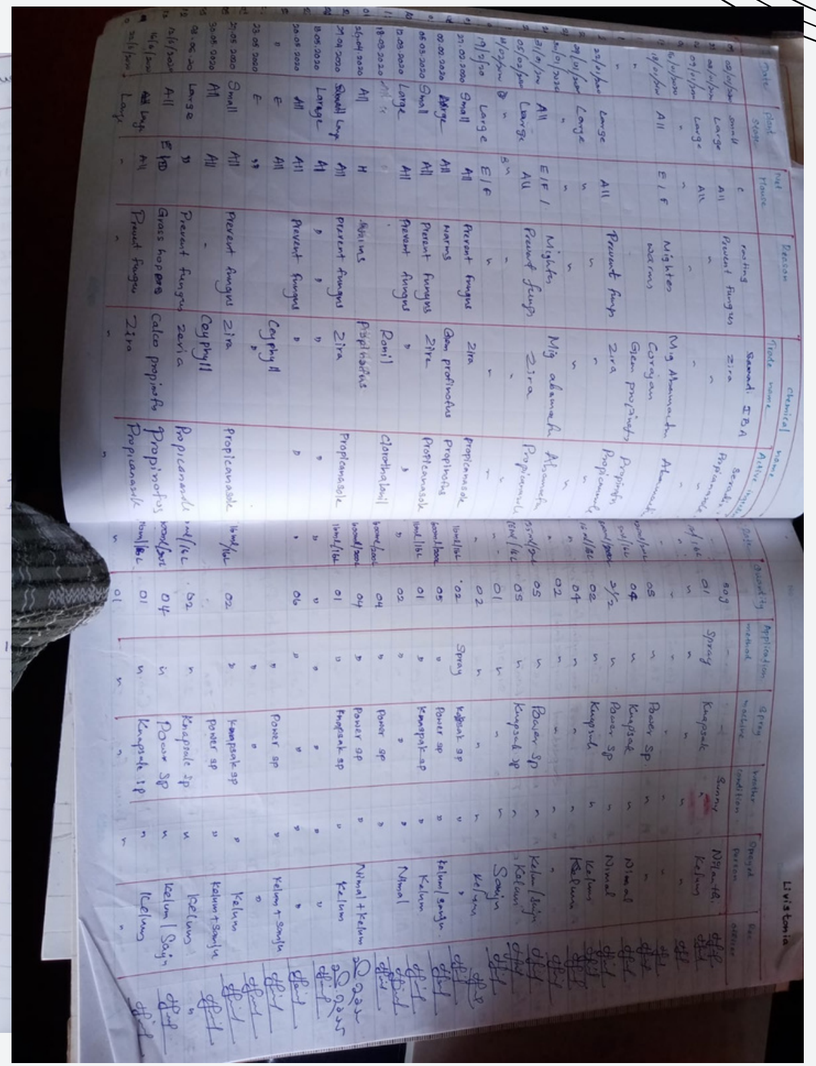
DATA GATHERING

These data is collected from the omega green Pvt Ltd

And all data related to growth monitoring collected in manual way



Date	plant stage	Net house	Reason	chemical Trade name	Name Active Ingr.
06/01	Large	E	Prevent fungus	Zira	Propicanazole
18/01	"	"	"	"	"
20/01	"	"	"	"	"
23/01	All	E	Grass hopers	Gem propinafos	Propinafos
11/02	Large	"	Prevent fungus	Zira	Propicanazole
19/02	"	"	"	"	"
25/02	All	"	Mighter	Mig abamactin	Abamactin
02/3	All	E	Root rot	Homai	
03/3	Large		Prevent fungu	Zira	Propicanazole
04/3	All	E	Root rot	Homai	
11/3	Large	"	Prevent fungu	Zira	Propicanazole
13/3	"	E	Root rot	Homai	
19/3	Small	B	Root growth	Seradix	IBA
01/4	"	"	"	"	"
8/4	"	"	"	"	"
07/05	Large	E	Grasshopers	Profenofos	Propinofus
07/06	"	"	Prevent fungi	Daconil	chlorothalonil
08/06	"	H	"	"	"
11/06	"	E	"	Ronil	"
14/06	Small	B	Root growth	Seradix	IBA
15/07	Large	E		Profenofos	Profenofus
17/07	Small	B	Root stimulation	Seradix	IBA
14/08	Large	E	Prevent fungi	Daconil	Chlorothalonil
6/08	"	"	Insects	Cobra	Imidacloprid
0/08	"	"	"	"	"
	"	"	Fungi	Zira	Propicanasole
/08	"	"	"	"	"
09	"	"	"	"	"
09	"	"	"	"	"
9	"	"	"	"	"
9	"	B	Root stimulation	Seradix	IBA

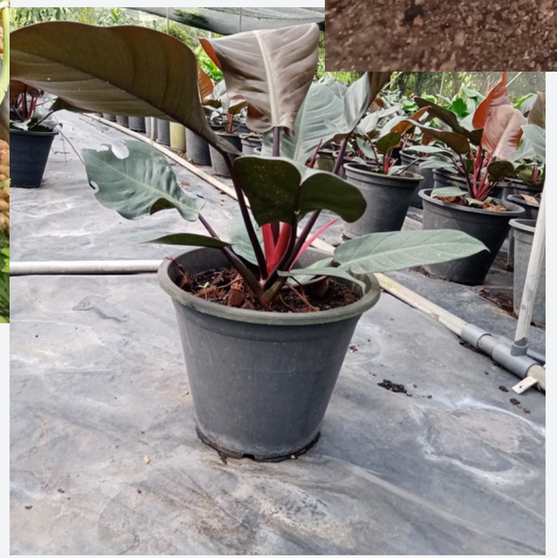


SLIIT FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

DATA GATHERING

Pictures of main plant types considered



SLIIT
FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

MODEL CREATION

Data Preprocessing

```
#cheking  data types of the coulmn
Growth_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2184 entries, 0 to 2183
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Plant_Bed_number    2184 non-null   object
 1   Plant_number        2184 non-null   object
 2   Plant_type          2184 non-null   object
 3   Variety             2184 non-null   object
 4   Media_Grew          2157 non-null   object
 5   Checked_Date        2184 non-null   object
 6   Temperature         2145 non-null   float64
 7   Humidity            2147 non-null   float64
 8   Fertilizer          2180 non-null   object
 9   Fertilizer_Amount   2184 non-null   int64
 10  Growth_Category     2184 non-null   object
 11  Current_pot_type    2184 non-null   object
 12  Current_life_time   2178 non-null   object
 13  Average_min_height  2184 non-null   int64
 14  Average_max_height  2184 non-null   int64
 15  Planted_Date        2184 non-null   object
dtypes: float64(2), int64(3), object(11)
memory usage: 273.1+ KB
```

```
[7]  #cheking the available columns in the data set
     Growth_Data.columns
```

```
Index(['Plant_Bed_number', 'Plant_number', 'Plant_type ', 'Variety',
       'Media_Grew', 'Checked_Date', 'Temperature', 'Humidity', 'Fertilizer',
       'Fertilizer_Amount', 'Growth_Category', 'Current_pot_type',
```

Identify null values

```
[1]  #check null values of the trainin gdata set

     Growth_Data.isnull()
```

	Plant_Bed_number	Plant_number	Plant_type	Variety	Media_Grew	Checked_Date	Temperature	Humidity	Fertilizer	Fertilizer_Amount	Growth_Category	Current_pot_type	Current_life_time	Ave
0	False	False	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2179	False	False	False	False	False	False	False	False	False	False	False	False	False	
2180	False	False	False	False	False	False	False	False	False	False	False	False	False	
2181	False	False	False	False	False	False	False	False	False	False	False	False	False	
2182	False	False	False	False	False	False	False	False	False	False	False	False	False	
2183	False	False	False	False	False	False	False	False	False	False	False	False	False	

2184 rows × 16 columns

```
[11] Growth_Data.isnull().sum().sort_values(ascending=False)
```

```
Temperature          39
Humidity             37
Media_Grew           27
Current_life_time     6
Fertilizer            4
```

SLIIT FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

MODEL CREATION

Data Preprocessing

```
[20]
Growth_Data.loc[Growth_Data['Growth_Category'] == 'coir,cow_dung,coco chips', 'Media_Grew'] = 'coir,cowdung,coco chips'

Growth_Data
```

	Plant_Bed_number	Plant_number	Plant_type	Variety	Media_Grew	Checked_Date	Temperature	Humidity	Fert
0	22-TGN-07	TGN249	Tropical Green enjoy	pothos	coir,cow_dung,coco chips	2022.10.09	22.0	94.0	Yaramila Ap
1	20-MQ-01	MQ-73	Marble Queen	phothos	coir,cowdung,coco chips	2021.04.26	33.0	46.0	Yaramila,Gro
2	20-MQ-01	MQ-449	Marble Queen	phothos	coir,cowdung,coco chips	2020.02.29	36.0	46.0	Yaramila,Gro
3	20-MQ-08	MQ-344	Marble Queen	phothos	coir,cowdung	2020.10.07	28.0	85.0	Yaramila,Gro
4	22-LL-27	LL-292	Lemon Lime	phothos	coir,cowdung,coco chips	2022.05.26	27.0	86.0	Ya
...	...	...	...	...	...	...	...	...	
2179	21-BC-76	BC-17	Black Cardinal	philodrendon	coir,cowdung	2021.12.21	30.0	64.0	Yaramila App

```
#cheking  data types of the coulmn
Growth_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2184 entries, 0 to 2183
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Plant_Bed_number    2184 non-null   object
 1   Plant_number        2184 non-null   object
 2   Plant_type          2184 non-null   object
 3   Variety             2184 non-null   object
 4   Media_Grew          2157 non-null   object
 5   Checked_Date        2184 non-null   object
 6   Temperature         2145 non-null   float64
 7   Humidity            2147 non-null   float64
 8   Fertilizer          2180 non-null   object
 9   Fertilizer_Amount   2184 non-null   int64
 10  Growth_Category     2184 non-null   object
 11  Current_pot_type    2184 non-null   object
 12  Current_life_time   2178 non-null   object
 13  Average_min_height  2184 non-null   int64
 14  Average_max_height  2184 non-null   int64
 15  Planted_Date        2184 non-null   object
dtypes: float64(2), int64(3), object(11)
memory usage: 273.1+ KB
```

```
[7]  #cheking the available columns in the data set
     Growth_Data.columns
```

```
Index(['Plant_Bed_number', 'Plant_number', 'Plant_type ', 'Variety',
       'Media_Grew', 'Checked_Date', 'Temperature', 'Humidity', 'Fertilizer',
       'Fertilizer_Amount', 'Growth_Category', 'Current_pot_type',
```

SLIIT FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

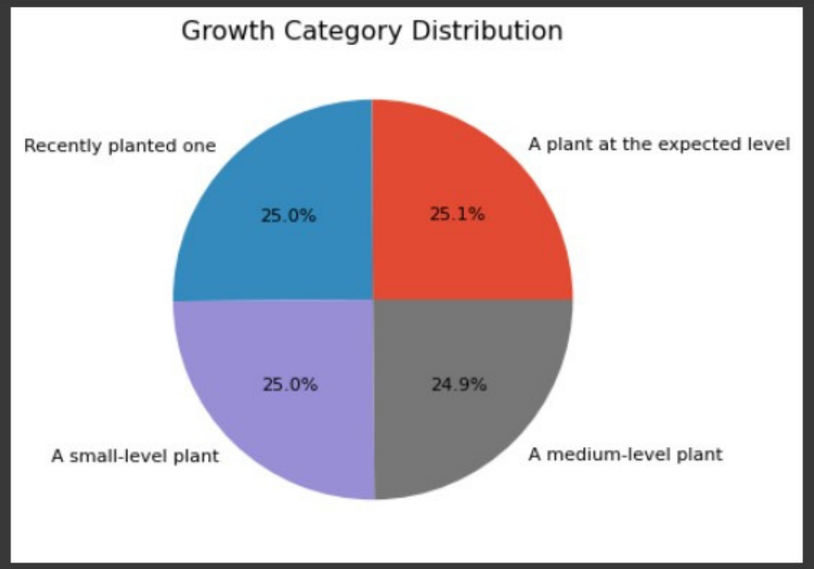
MODEL CREATION

Checking the balnce of class

Checking the class distribution from each plant type

+ Code + Text

```
[22] plt.figure(figsize=(6,5),dpi=80)
     # calculate the number of individuals in each experience level
     Category_counts = Growth_Data['Growth_Category'].value_counts()

     # create a pie chart
     plt.pie(Category_counts.values, labels=Category_counts.index, autopct='%1.1f%%')

     # add title
     plt.title('Growth Category Distribution')

     # display plot
     plt.show()
```



SLIIT FACULTY OF COMPUTING

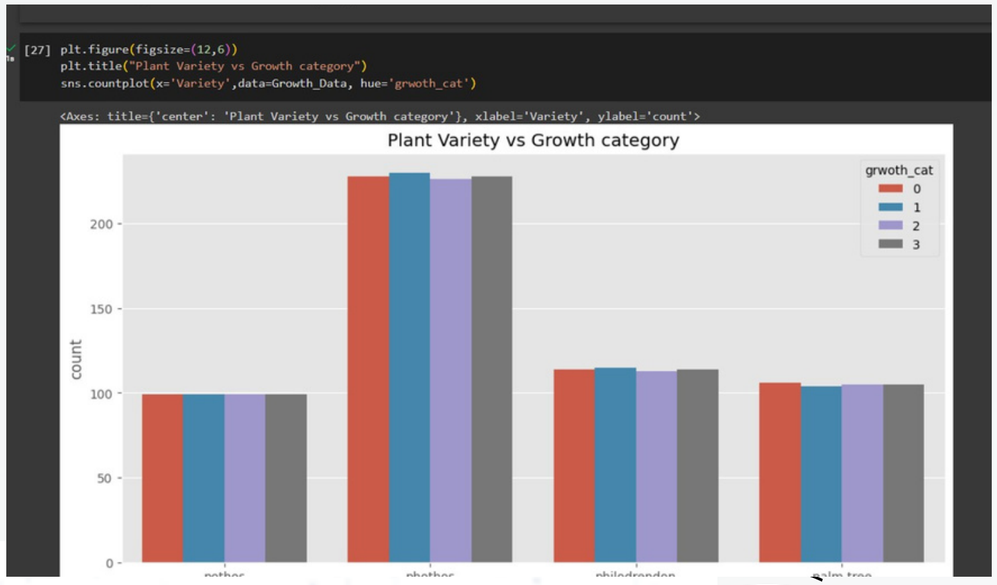
IT20005726 | Liyanage S. R | 2023-133

MODEL CREATION

Exploratory data analysis

Categorical data vs Target

```
[27] plt.figure(figsize=(12,6))
     plt.title("Plant Variety vs Growth category")
     sns.countplot(x='Variety',data=Growth_Data, hue='grwoth_cat')

<Axes: title={'center': 'Plant Variety vs Growth category'}, xlabel='Variety', ylabel='count'>
```



SLIIT FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

MODEL CREATION

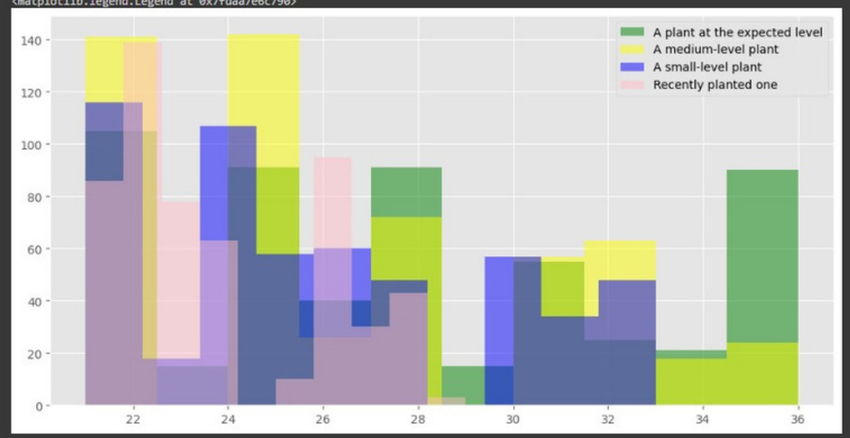
Exploratory data analysis

Numerical data vs Target

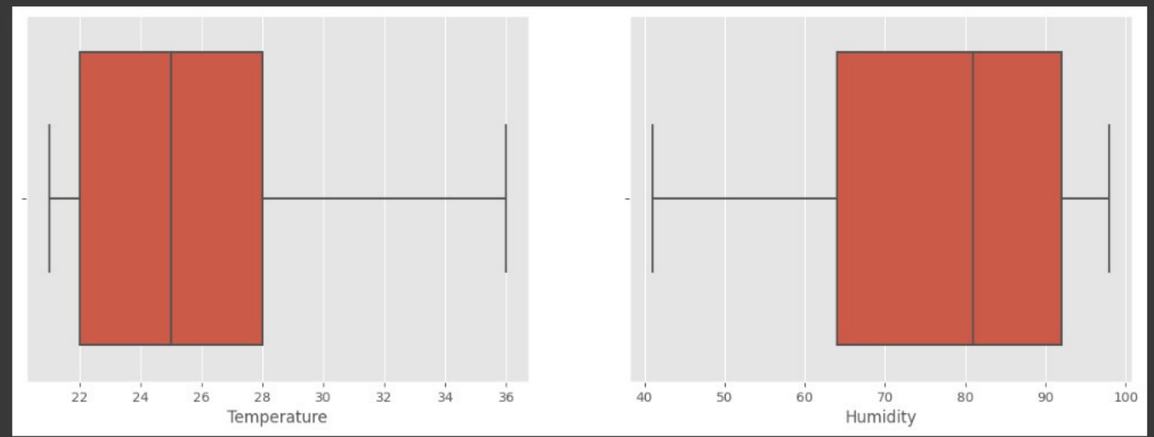
Temperature

```
plt.figure(figsize=(12,6))
plt.hist(Growth_Data[Growth_Data['grwoth_cat']==1]['Temperature'],color='green',alpha=0.5,label='A plant at the expected level')
plt.hist(Growth_Data[Growth_Data['grwoth_cat']==2]['Temperature'],color='yellow',alpha=0.5,label='A medium-level plant')
plt.hist(Growth_Data[Growth_Data['grwoth_cat']==3]['Temperature'],color='blue',alpha=0.5,label='A small-level plant')
plt.hist(Growth_Data[Growth_Data['grwoth_cat']==0]['Temperature'],color='pink',alpha=0.5,label='Recently planted one')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fdaa7e6c790>
```




```
numerical = ['Temperature','Humidity']
i = 0
while i < 2:
    fig = plt.figure(figsize=[15,5])
    plt.subplot(1,2,1)
    sns.boxplot(x=numerical[i],data=Growth_Data)
    i += 1
    plt.subplot(1,2,2)
    sns.boxplot(x=numerical[i],data=Growth_Data)
    i += 1
    plt.show()
```



SLIIT FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

MODEL CREATION

One hot encoding

Handling Categorical Data with OneHotEncoding

```
[34] #one hot encoding data
     from sklearn.preprocessing import OneHotEncoder

     numerical_cols_train = Input_columns.select_dtypes(include=np.number).columns.tolist()
     categorical_cols_train = Input_columns.select_dtypes('object').columns.tolist()
     categorical_cols_train
```

```
['Plant_type ', 'Variety', 'Media_Grew', 'Fertilizer']
```

```
[35] numerical_cols_train
```

```
['Temperature',
 'Humidity',
 'Fertilizer_Amount',
 'Average_min_height',
 'Average_max_height',
 'grwoth_cat']
```

```
[36] encoder = OneHotEncoder(sparse=False,handle_unknown='ignore').fit(Input_columns[categorical_cols_train])
     encoder
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be
  warnings.warn(
```

```
OneHotEncoder(handle_unknown='ignore', sparse=False, sparse_output=False)
```

```
[37] encoded_cols = list(encoder.get_feature_names_out(categorical_cols_train))
     encoded_cols
```

```
['Plant_type _Black Cardinal',
 'Plant_type _Lemon Lime',
 'Plant_type _Livistonia',
 'Plant_type _Marble Queen',
 'Plant_type _Tropical Green enjoy',
 'Variety_palm tree',
 'Variety_philodrendon',
```

SLIIT FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

MODEL CREATION

Splitting the data set

Scaling the data set

Splitting training and testing data set with ratio of 80:20

+ Code + Text

```
[45] from sklearn.model_selection import train_test_split
     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,random_state=42)
```

```
[46] x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1747 entries, 812 to 860
Data columns (total 30 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Temperature                             1747 non-null   float64
 1   Humidity                                1747 non-null   float64
 2   Fertilizer_Amount                       1747 non-null   int64
 3   Average_min_height                      1747 non-null   int64
 4   Average_max_height                      1747 non-null   int64
 5   Plant_type _Black Cardinal              1747 non-null   float64
 6   Plant_type _Lemon Lime                  1747 non-null   float64
 7   Plant_type _Livistonia                  1747 non-null   float64
 8   Plant_type _Marble Queen                1747 non-null   float64
 9   Plant_type _Tropical Green enjoy        1747 non-null   float64
 10  Variety_palm tree                       1747 non-null   float64
 11  Variety_philodrendon                    1747 non-null   float64
 12  Variety_phothos                         1747 non-null   float64
 13  Variety_pothos                          1747 non-null   float64
 14  Media_Grew_coir                         1747 non-null   float64
 15  Media_Grew_coir,cow_dung                1747 non-null   float64
 16  Media_Grew_coir,cow_dung,coco chips     1747 non-null   float64
 17  Media_Grew_coir,cowdung                 1747 non-null   float64
 18  Media_Grew_coir,cowdung,coco chips      1747 non-null   float64
 19  Media_Grew_not_given                    1747 non-null   float64
 20  Fertilizer_Foliar application k44       1747 non-null   float64
 21  Fertilizer_Forliar Appliation           1747 non-null   float64
 22  Fertilizer_Gem Profinofus               1747 non-null   float64
 23  Fertilizer_Yaramila                     1747 non-null   float64
 24  Fertilizer_Yaramila,Floiar Application  1747 non-null   float64
```

```
# Scale the numeric features
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(x_train)
X_test_scaled = scaler.transform(x_test)
```

```
[49] #checking the data set
     X_train_scaled
```

```
array([[0.        , 0.96491228, 1.        , ..., 0.        , 0.        ,
        0.        ],
       [0.2       , 0.87719298, 0.00401606, ..., 0.        , 0.        ,
        0.        ],
       [0.06666667, 0.85964912, 0.09638554, ..., 0.        , 1.        ,
        0.        ],
       ...,
       [0.06666667, 1.        , 0.0060241 , ..., 0.        , 0.        ,
        0.        ],
       [0.6       , 0.40350877, 0.00200803, ..., 0.        , 0.        ,
        0.        ],
       [0.73333333, 0.29824561, 0.00200803, ..., 0.        , 0.        ,
```

SLIIT
FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

MODEL CREATION

Model Training

Random Forest Classifier

```
# Random Forest with regularization
rf_classifier = RandomForestClassifier(n_estimators=100, max_depth=5, min_samples_split=10, min_samples_leaf=5, random_
rf_classifier.fit(X_train_scaled, y_train)
y_pred_rf_rg = rf_classifier.predict(X_test_scaled)

# Evaluate Random Forest model
acc_rf = accuracy_score(y_test, y_pred_rf_rg)
recall_rf = recall_score(y_test, y_pred_rf_rg, average='micro')
precision_rf = precision_score(y_test,y_pred_rf_rg, average='micro')
f1score_rf = f1_score(y_test, y_pred_rf_rg, average='micro')

print("Random Forest Classifier")
print("Accuracy: {:.2f}".format(acc_rf))
print("Precision: {:.2f}".format(precision_rf))
print("Recall: {:.2f}".format(recall_rf))
print("F1-Score: {:.2f}".format(f1score_rf))
```

```
Random Forest Classifier
Accuracy: 0.99
Precision: 0.99
Recall: 0.99
F1-Score: 0.99
```

```
[54] #PRINT CLASSIFIACTION REPORT
print(classification_report(y_pred_rf_rg,y_test))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	117
1	1.00	1.00	1.00	109
2	1.00	0.97	0.99	106
3	0.97	1.00	0.99	105
accuracy			0.99	437
macro avg	0.99	0.99	0.99	437
weighted avg	0.99	0.99	0.99	437

MODEL CREATION

Model Training

02.XGBoost Model

```
## ---------- XGBoost model v1 ----------
## base run of model with default hyperparameters
import xgboost as xgb
xgb_clf = xgb.XGBClassifier(objective='multi:softmax',
                            num_class=4,
                            missing=1,
                            early_stopping_rounds=10,
                            learning_rate=0.001,
                            eval_metric=['merror','mlogloss'],
                            seed=42)
xgb_clf.fit(x_train,
            y_train,
            verbose=0, # set to 1 to see xgb training round intermediate results
            eval_set=[(x_train, y_train), (x_test, y_test)])
# XGBoost with regularization
xgb_classifier = xgb.XGBClassifier(reg_alpha=0.1, reg_lambda=0.1,learning_rate=0.001, eval_metric=['merror','mlogloss'], random_state=42 )
xgb_classifier.fit(X_train_scaled, y_train,verbose=0,eval_set=[(X_train_scaled, y_train), (X_test_scaled, y_test)])
# # preparing evaluation metric plots
```

XG boost

```
------------------ Confusion Matrix -----------------


-------------------- Key Metrics --------------------

Accuracy: 1.00
Balanced Accuracy: 1.00

Micro Precision: 1.00
Micro Recall: 1.00
Micro F1-score: 1.00

Macro Precision: 1.00
Macro Recall: 1.00
Macro F1-score: 1.00

Weighted Precision: 1.00
Weighted Recall: 1.00
Weighted F1-score: 1.00

--------------- Classification Report ---------------

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       117
           1       1.00      1.00      1.00       109
           2       1.00      1.00      1.00       103
           3       1.00      1.00      1.00       108

    accuracy                           1.00       437
   macro avg       1.00      1.00      1.00       437
weighted avg       1.00      1.00      1.00       437

---------------------- XGBoost ----------------------
```

SLIIT FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

MODEL CREATION

Model Training

03.Decision Tree Classsifier

```
[57] # decision tree classifier
     from sklearn.tree import DecisionTreeClassifier
     DT_model = DecisionTreeClassifier()
     #Training the model
     DT_model.fit(X_train_scaled, y_train)

     y_pred = DT_model.predict(X_test_scaled)
     RF_probability = DT_model.predict_proba(X_test_scaled)[:,1]

     dt_classifier_model = DecisionTreeClassifier(max_depth=5, min_samples_split=10, min_samples_leaf=5, random_state=42)

     # Train the decision tree classifier
     dt_classifier_model.fit(X_train_scaled, y_train)

     y_pred_DT_rg = dt_classifier_model.predict(X_test_scaled)
     #getting the accuracy , reacall and precission of the model using test set

     acc_DT=accuracy_score(y_test,y_pred_DT_rg)
     recall_DT=recall_score(y_test,y_pred_DT_rg,average='micro')
     precision_DT=precision_score(y_test,y_pred_DT_rg,average='micro')
     f1score_DT=f1_score(y_test,y_pred_DT_rg,average='micro')

     print("Accuracy : ", acc_DT)
     print("Precision:",precision_DT)
     print("Recall:",recall_DT)
     print("F1-Score:",f1score_DT)
```

```
Accuracy :  1.0
Precision: 1.0
Recall: 1.0
F1-Score: 1.0
```

Decision Tree Classifier

SLIIT FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

MODEL CREATION

Model Training

04. Naive Bayes classifier

```
[58] # train a Gaussian Naive Bayes classifier on the training set
     from sklearn.naive_bayes import GaussianNB


     # instantiate the model
     gnb = GaussianNB()


     # fit the model
     gnb.fit(X_train_scaled, y_train)
     y_pred_GNB = gnb.predict(X_test_scaled)

     #GETTING THE scores
     acc_GNB=accuracy_score(y_test,y_pred_GNB)
     recall_GNB=recall_score(y_test,y_pred_GNB,average='micro')
     precision_GNB=precision_score(y_test,y_pred_GNB,average='micro')
     f1score_GNB=f1_score(y_test,y_pred_GNB,average='micro')

     print("Accuracy : ", acc_GNB)
     print("Precision:",recall_GNB)
     print("Recall:",precision_GNB)
     print("F1-Score:",f1score_GNB)
```

```
Accuracy :  0.9725400457665904
Precision: 0.9725400457665904
Recall: 0.9725400457665904
F1-Score: 0.9725400457665903
```

Naive bayes Classifier

```
[59] from sklearn.metrics import confusion_matrix

     cm = confusion_matrix(y_test, y_pred_GNB)

     print('Confusion matrix\n\n', cm)

     print('\nTrue Positives(TP) = ', cm[0,0])

     print('\nTrue Negatives(TN) = ', cm[1,1])

     print('\nFalse Positives(FP) = ', cm[0,1])

     print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[117   0   0   0]
 [  0 109   0   0]
 [  0   9  94   0]
 [  0   0   3 105]]

True Positives(TP) =  117

True Negatives(TN) =  109

False Positives(FP) =  0

False Negatives(FN) =  0
```

SLIIT
FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

MODEL CREATION

Comparison between models

COMPARISON AMONG THE MODELS

```
[60] ind=['XGBoost','Randomforest','Decision Tree','GaussianNB']
     data={"Accuracy":[acc_xg,acc_rf,acc_DT,acc_GNB],"Recall":[recall_xg,recall_rf,recall_DT,recall_GNB],"Precision":[precision_xg,precision_rf,precision_DT,precision_GNB],
         'f1_score':[f1score_xg,f1score_rf,f1score_DT,f1score_GNB]}
     result=pd.DataFrame(data=data,index=ind)
     result
```

	Accuracy	Recall	Precision	f1_score
XGBoost	1.000000	1.000000	1.000000	1.000000
Randomforest	0.993135	0.993135	0.993135	0.993135
Decision Tree	1.000000	1.000000	1.000000	1.000000
GaussianNB	0.972540	0.972540	0.972540	0.972540

MODEL CREATION

Checking the model work flow

```
import pyowm
owm = pyowm.OWM(OWM_API_KEY)
weather_mgr = owm.weather_manager()
place = 'Gampaha, LK'
observation = weather_mgr.weather_at_place(place)
```

```
[67] temperature = observation.weather.temperature("celsius")["temp"]
humidity = observation.weather.humidity
wind = observation.weather.wind()
print(f'Temperature(°C): {temperature}')
print(f'Humidity(%): {humidity}')
```

```
Temperature(°C): 26.47
Humidity(%): 86
```

Temperature and humidity through Open weather API

SLIIT FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

MODEL CREATION

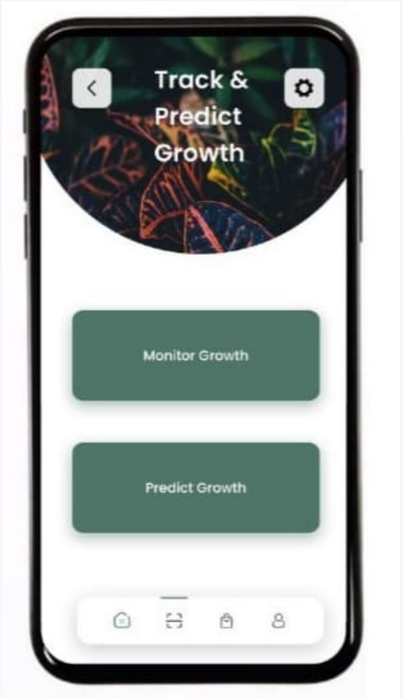
Checking the model work flow

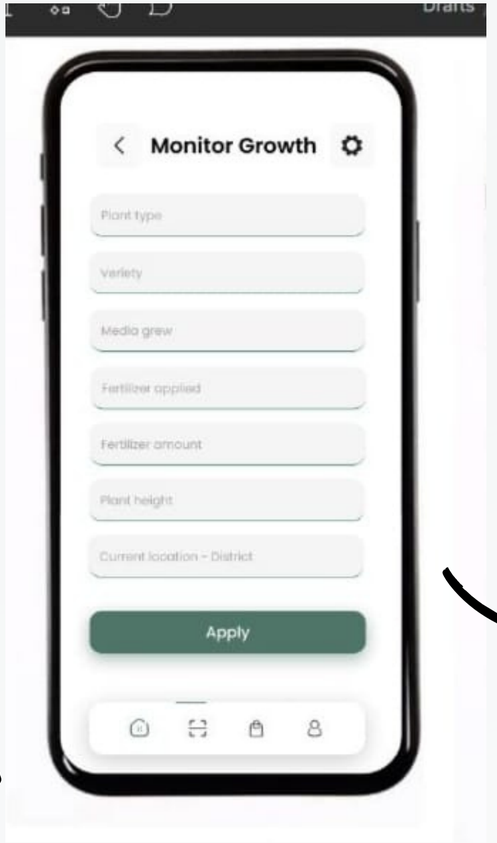
```
[70] # Scale the numeric features
     scaler = MinMaxScaler()
     X_new_scaled = scaler.fit_transform(X_new )
```

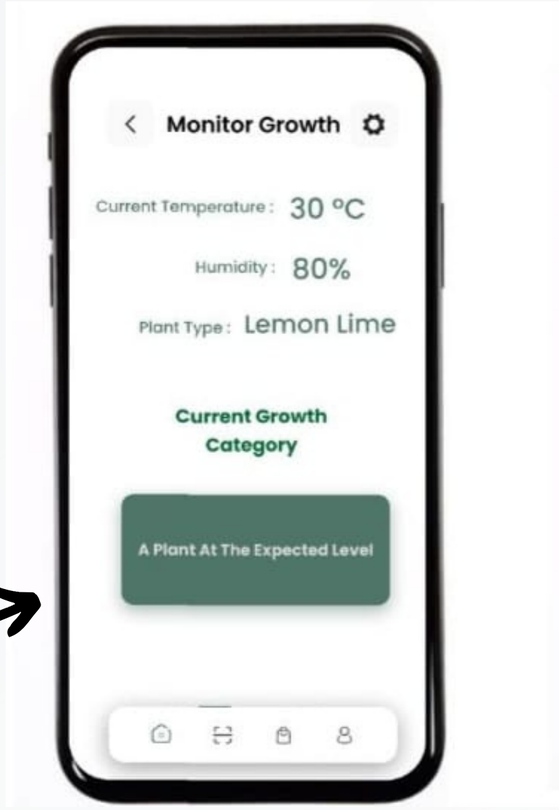
```
[71] #making the new prediction
     My_predictions = rf_classifier.predict(X_new_scaled)
     if My_predictions ==[0]:
          print('Recently planted one')
     elif My_predictions ==[1]:
       print('A plant at small level')
     elif My_predictions ==[2]:
       print('A plant at medium level')
     elif My_predictions ==[3]:
       print('A plant at expected level')

     Recently planted one
```

Model predict the expected value correctly

UI







SLIIT
FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

FUTURE

Imporove the growth monitoring model

Create a model for predictions

Developing the UI's

SLIIT
FACULTY OF COMPUTING

IT20005726 | Liyanage S. R | 2023-133

COMMERCIALIZATION & BUSINESS PLAN

Commodity Version

- Prediction of the current state of growth in ornamental plants
- Identification of the relative growth rate of plant height
- Distinguishing nutrient deficiencies from mite attacks
- Identification of the most suitable floriculture plant to grow based on weather and resource factors
- Forecastination of the demand for local and international markets

Premium Version (100$annually)

- Prediction of the time it will take for a plant to reach the required growth size for an order
- Providing suggestions to promote growth and advance the ornamental plant to the next level
- Identifying various floriculture crop varieties
- Recommendation of plants for shipments
- Predicting the optimal supply strategy for meeting the required output quantity based on geographical areas and finding the nearest vendor
- Providing packaging recommendations based on the plant's lifespan

COMMERCIALIZATION & BUSINESS PLAN

Target Audience:

- Floriculture businesses (small, medium, & large scale)
- Farmers and gardeners who grow ornamental plants
- Horticulture enthusiasts
- Research and educational institutions

Marketplace:

- Mobile application for Android & iOS devices
- Secure & user-friendly interface for customers & vendors

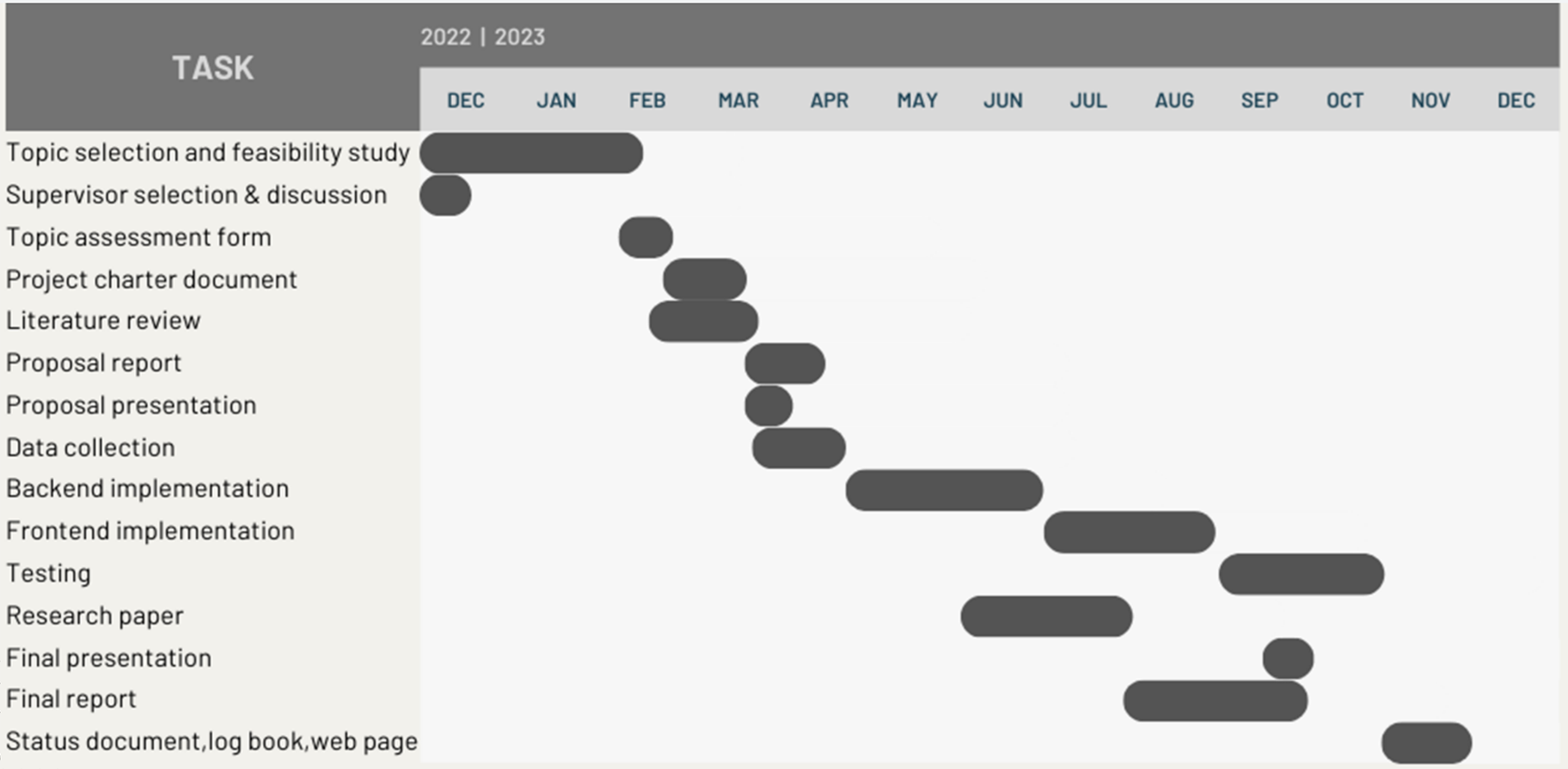
Marketing Plan:

- Social media marketing
- (Facebook, Instagram, Twitter)
- Email marketing campaigns
- Share leaflets

SLIIT
FACULTY OF COMPUTING

2023-133

GANTT CHART



SLIIT
FACULTY OF COMPUTING

2023-133

THANK YOU

PlantPal