Apeksha Hospital Donor Engagement System

2023-24-100

Team Members

Student Name	Student ID
Punchihewa S.N	IT20665166
Prabodha K.W.D.S	IT20665098
Bandara H.R.H.S	IT20662028
Wijesooriya P.L.P.G.D.S	IT20660352

Supervisor : Mrs.Lokesha Weerasinghe

Co-Supervisor : Ms.Chamali Pabasara

Introduction

Our project aims to revolutionize the donation process at Apeksha Hospital, ensuring high-quality donations and patient care.

Research Problem

The lack of a reliable and comprehensive platform for donations at Apeksha Hospital poses several challenges, including fraudulent activities and the absence of a specific and trustworthy platform for hair donations. These issues have a significant impact on the overall donation process, leading to inefficiencies, a lack of transparency, and a potential compromise in the quality and suitability of donated items for patient care.

Objectives

Main Objective

Improve communication, engagement, transparency, and security, while also optimizing donation campaigns and promoting high-quality donations.

Sub Objective

- Seamless Integration of Components
- User-Friendly Interface and Experience
- Scalability and Performance Optimization
- Continuous Monitoring and Improvement

Components

MAIN

Improve communication, engagement, transparency, and security, while also optimizing donation campaigns and promoting high-quality donations.

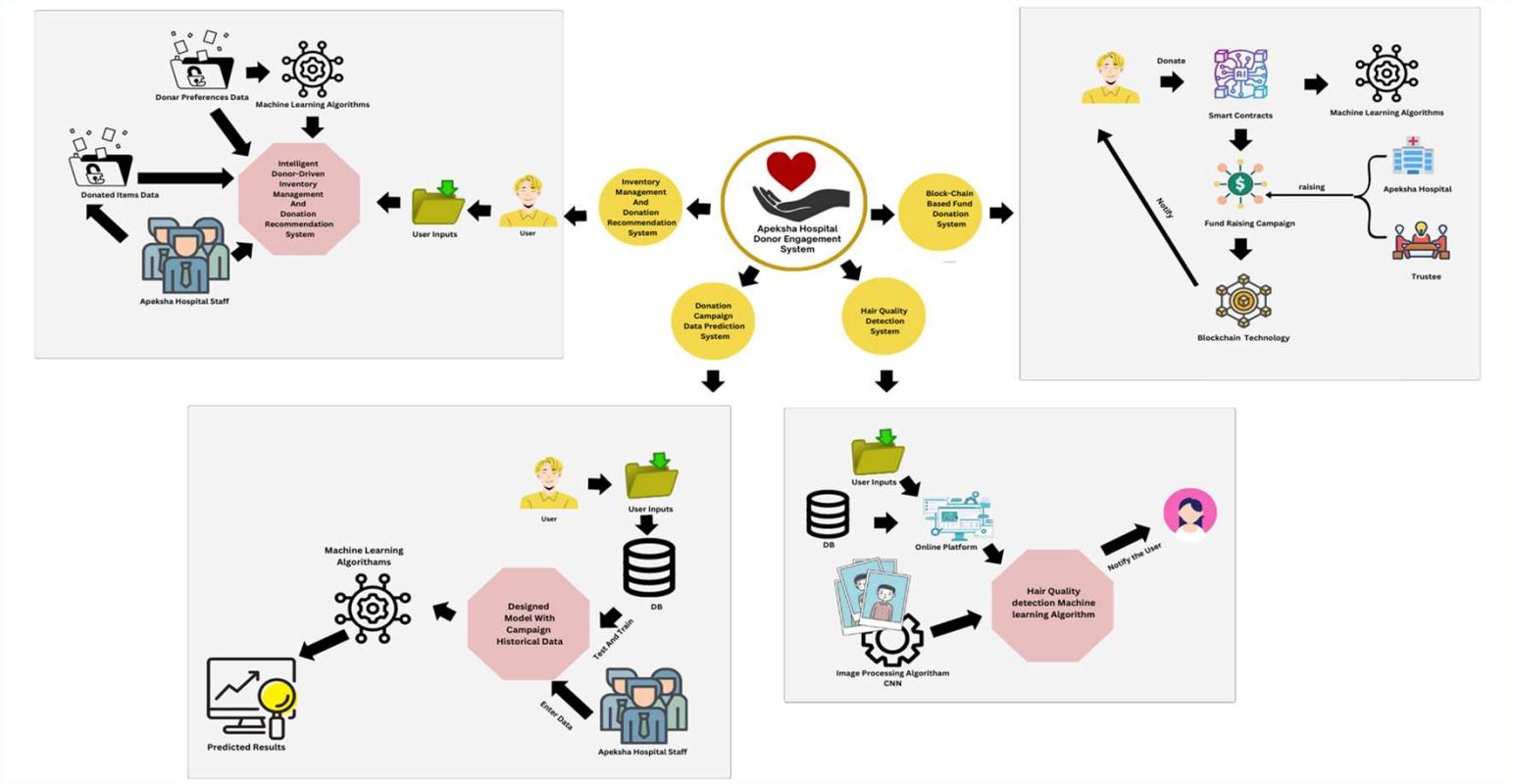
01 **Intelligent Donor-Driven Inventory System for Essential Items**

02 **Critical Medication Priority Recommender System**

03 **Predictive Analytics for Donation Campaign Success**

04 **Promoting quality hair donation for cancer patients**

System Overview Diagram



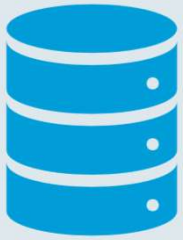
IT20665166
Punchihewa S.N

Bachelor of Science (Hons) in Information Technology
Specializing in Software Engineering

Introduction

Intelligent Donor-Driven Inventory System for Essential Items

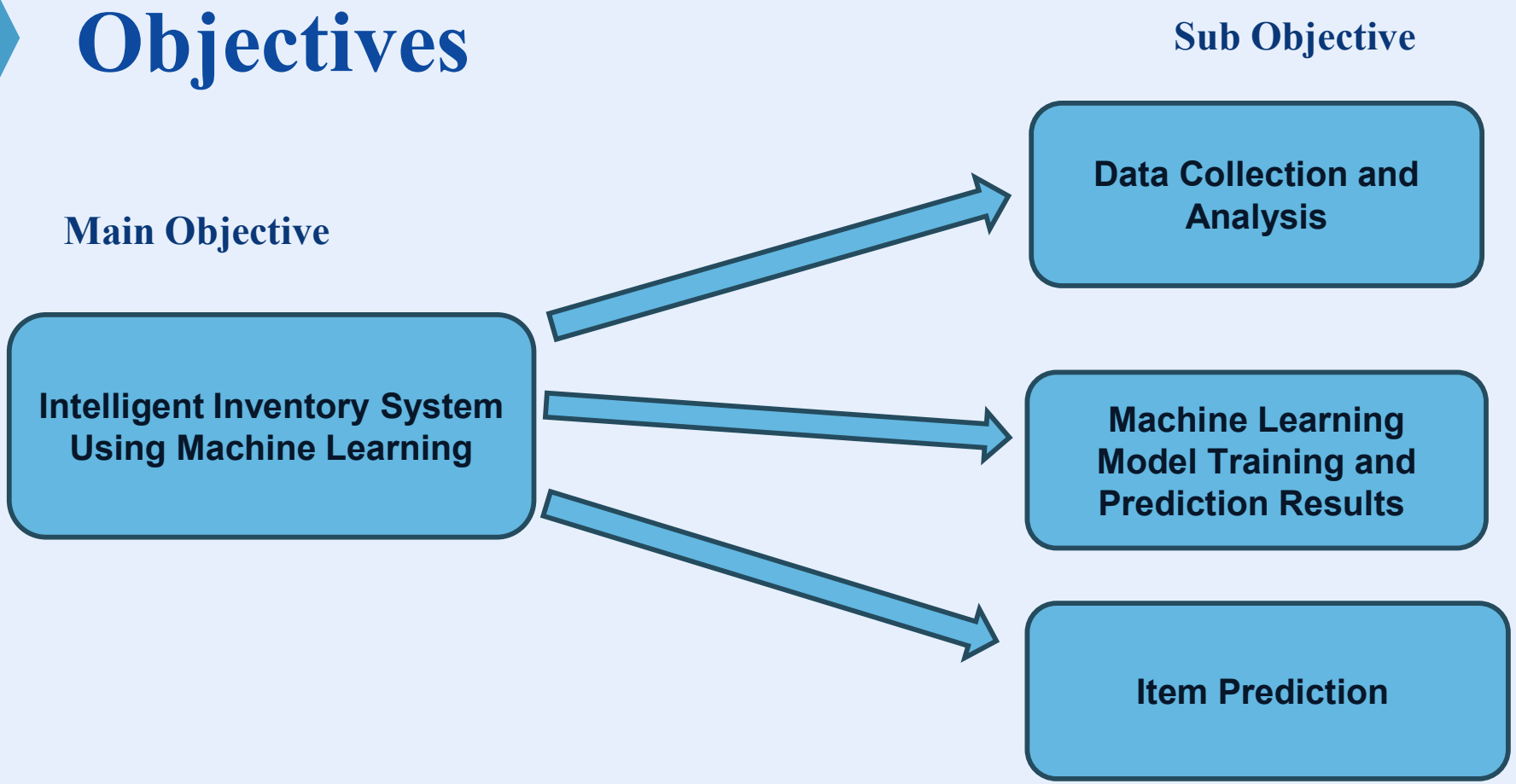
Background



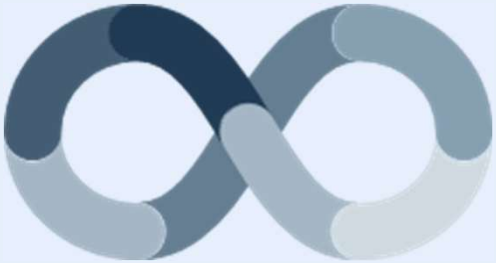
WHAT IS INTELLIGENT DONOR-DRIVEN INVENTORY SYSTEM?

USING MACHINE LEARNING TO UNDERSTAND ESSENTIAL ITEMS IN INVENTORY SYSTEM?

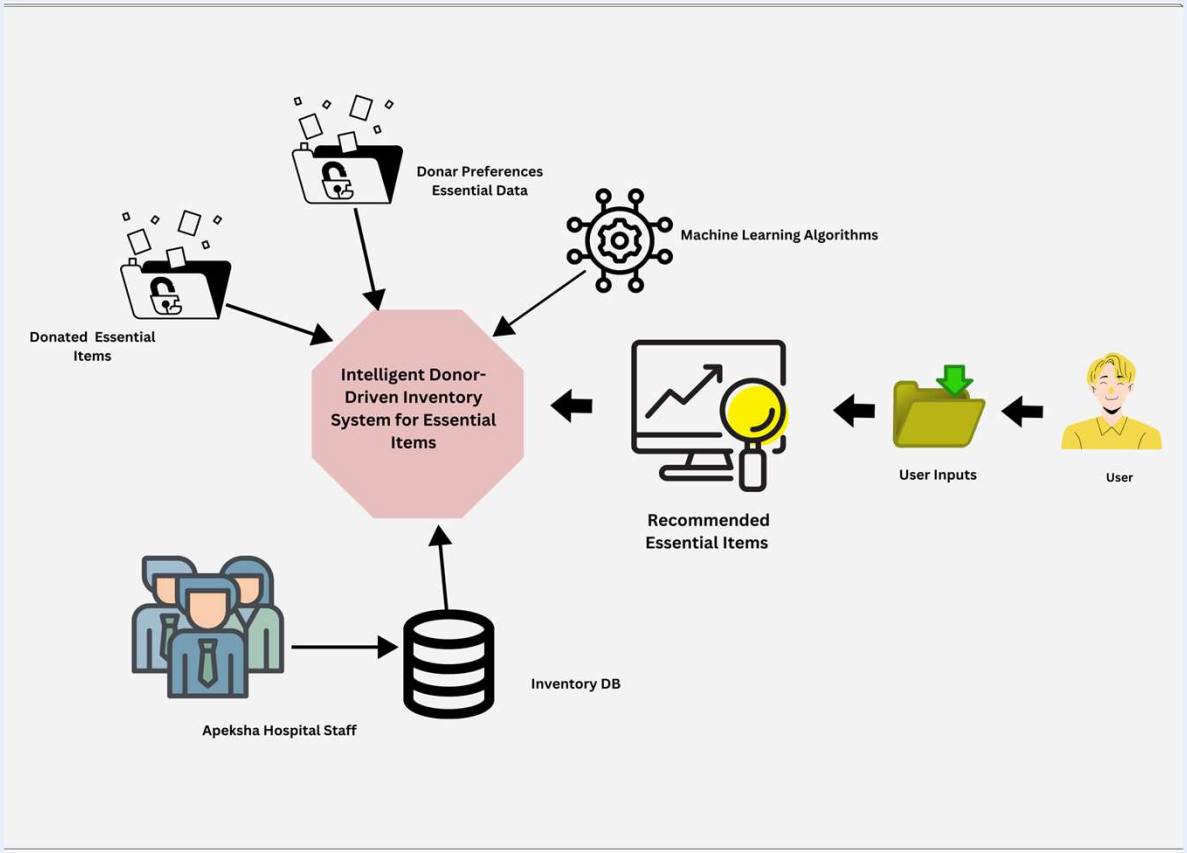
Objectives


Main Objective
Intelligent Inventory System Using Machine Learning
Sub Objective
Data Collection and Analysis
Machine Learning Model Training and Prediction Results
Item Prediction

Research Methodology



Component Diagram



Technologies and Techniques

Programming Language

- Python

Tools

- Jupyter Notebook
- Scikit Learn
- Anaconda Navigator

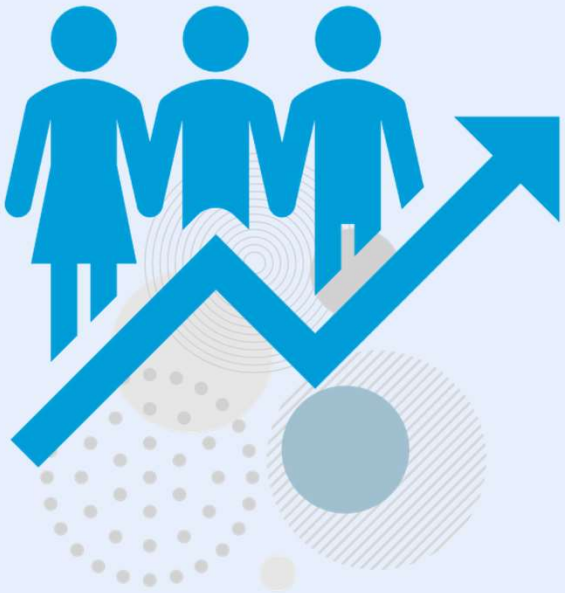
Version Controlling

- GitHub

Algorithm

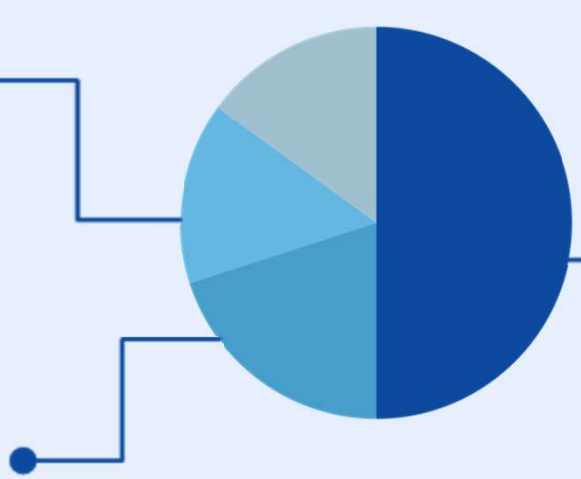
- Linear Regression
- Decision Tree Regression

PROGRESS



Current Progress

Suitable Machine Learning Algorithm Identification



Machine Learning Model Train using Linear Regression Algorithm

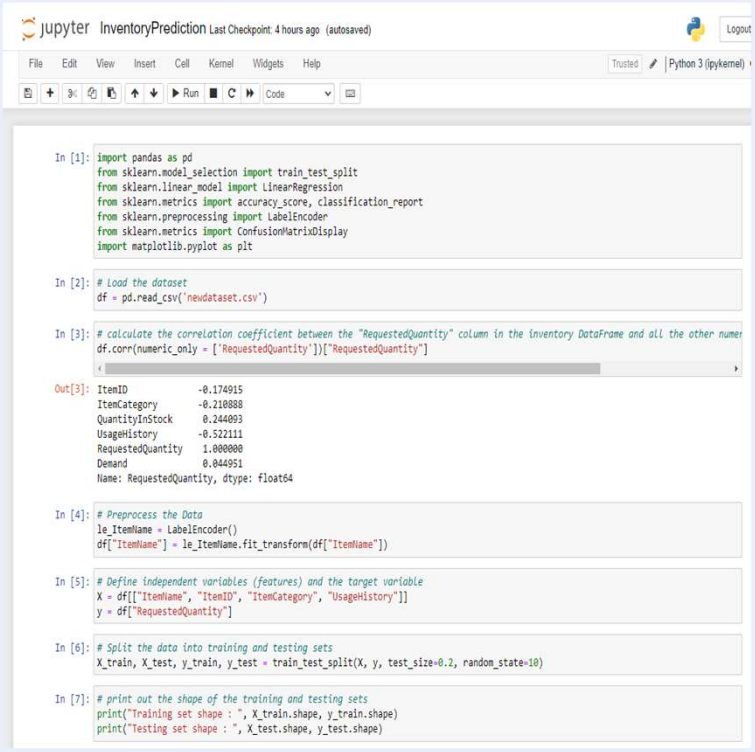
Inventory Requirement Prediction

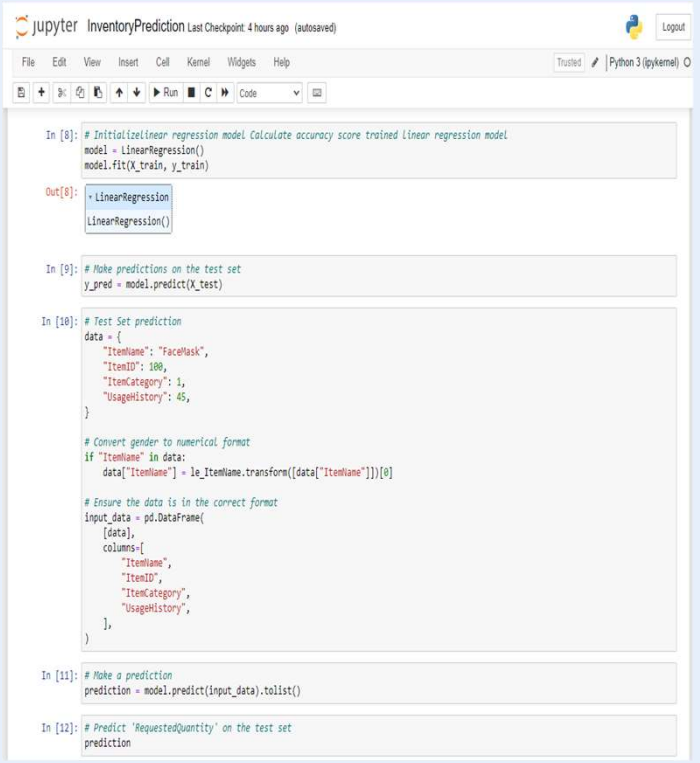
Next Expected Progress

- Identification of best architecture to Implement system
- Complete input identification
- Deploy Machine Learning Model using FastAPI
- Improve the accuracy of model
- Implement the Use-friendly Interface

Project Evidence

Linear Regression

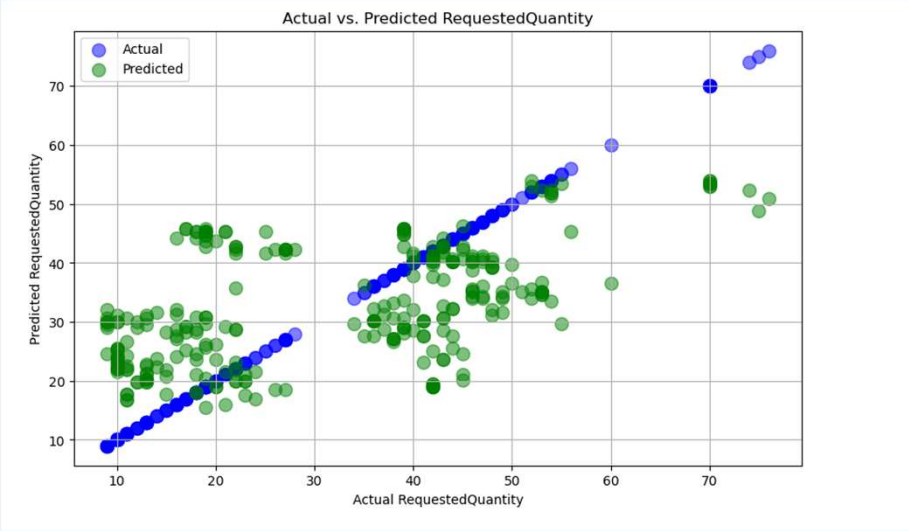


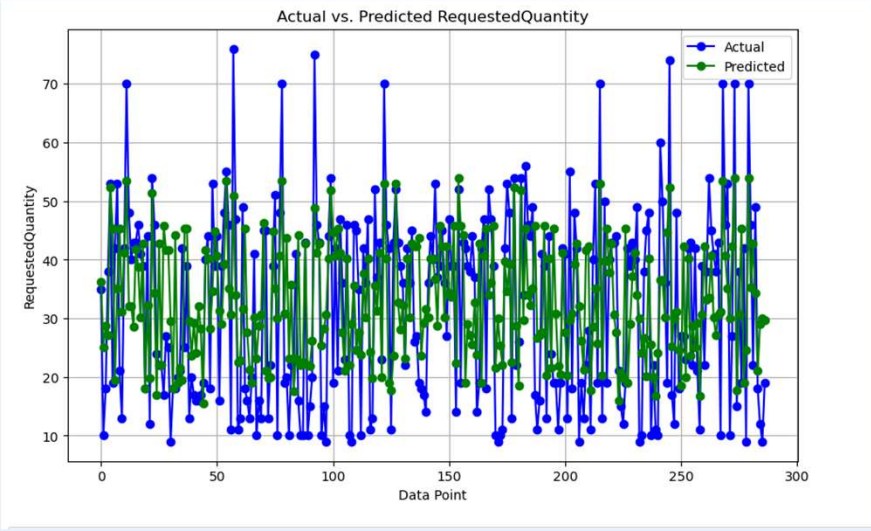


Linear Regression

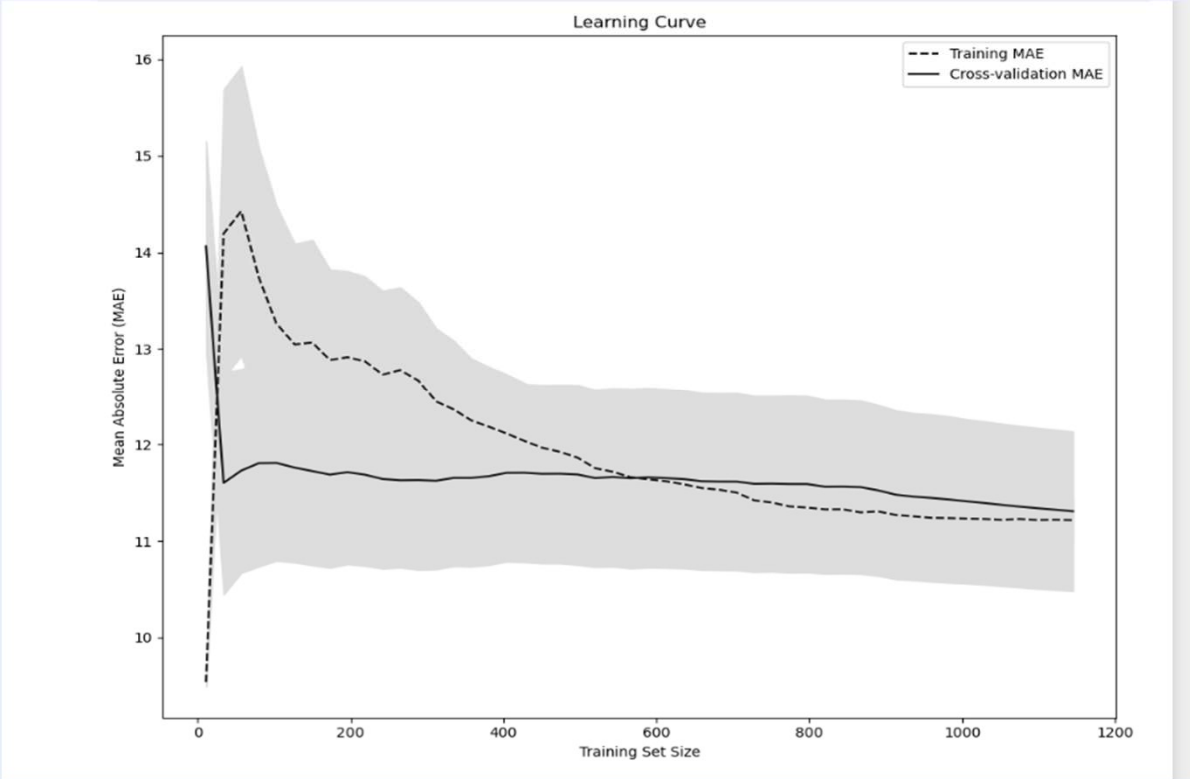
```
In [11]: # Make a prediction
         prediction = model.predict(input_data).tolist()

In [12]: # Predict 'RequestedQuantity' on the test set
         prediction

Out[12]: [37.79863989744041]

In [13]: prediction

Out[13]: [37.79863989744041]

In [14]: # model.fit(X_train, y_train)
         model.score(X_train, y_train)

Out[14]: 0.36888600431470053
```

Linear Regression





Linear Regression



Decision Tree Regression

Training Set Size

In [20]:
```
# Decision Tree Regression Algorithm
```

In [21]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor  # Import DecisionTreeRegressor
import matplotlib.pyplot as plt
```

In [22]:
```
# Load the dataset
df = pd.read_csv('newdataset.csv')

# Preprocess the Data
# (You can keep the LabelEncoder for ItemName)
from sklearn.preprocessing import LabelEncoder
```

In [23]:
```
le_ItemName = LabelEncoder()
df["ItemName"] = le_ItemName.fit_transform(df["ItemName"])

# Define independent variables (features) and the target variable
X = df[["ItemName", "ItemID", "ItemCategory", "UsageHistory"]]
y = df["RequestedQuantity"]
```

In [24]:
```
# 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)
```

In [25]:
```
# Print out the shape of the training and testing sets
print("Training set shape:", X_train.shape, y_train.shape)
print("Testing set shape:", X_test.shape, y_test.shape)
```

```
Training set shape: (1145, 4) (1145,)
Testing set shape: (287, 4) (287,)
```

In [26]:
```
# Initialize Decision Tree Regression model
model = DecisionTreeRegressor(random_state=42)  # You can specify other hyperparameters if needed
```

In [27]:
```
# Train the Decision Tree model
model.fit(X_train, y_train)
```

Out[27]:
```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

DecisionTreeRegressor(random_state=42)

In [28]:
```
# Test Set prediction
data = {
    "ItemName": "FaceMask",
    "ItemID": 100,
    "ItemCategory": 1,
    "UsageHistory": 45,
}

# Convert "ItemName" to numerical format using the LabelEncoder
if "ItemName" in data:
    data["ItemName"] = le_ItemName.transform([data["ItemName"]])[0]

# Ensure the data is in the correct format
input_data = pd.DataFrame(
    [data],
    columns=[
        "ItemName",
        "ItemID",
        "ItemCategory",
        "UsageHistory",
    ],
)
```

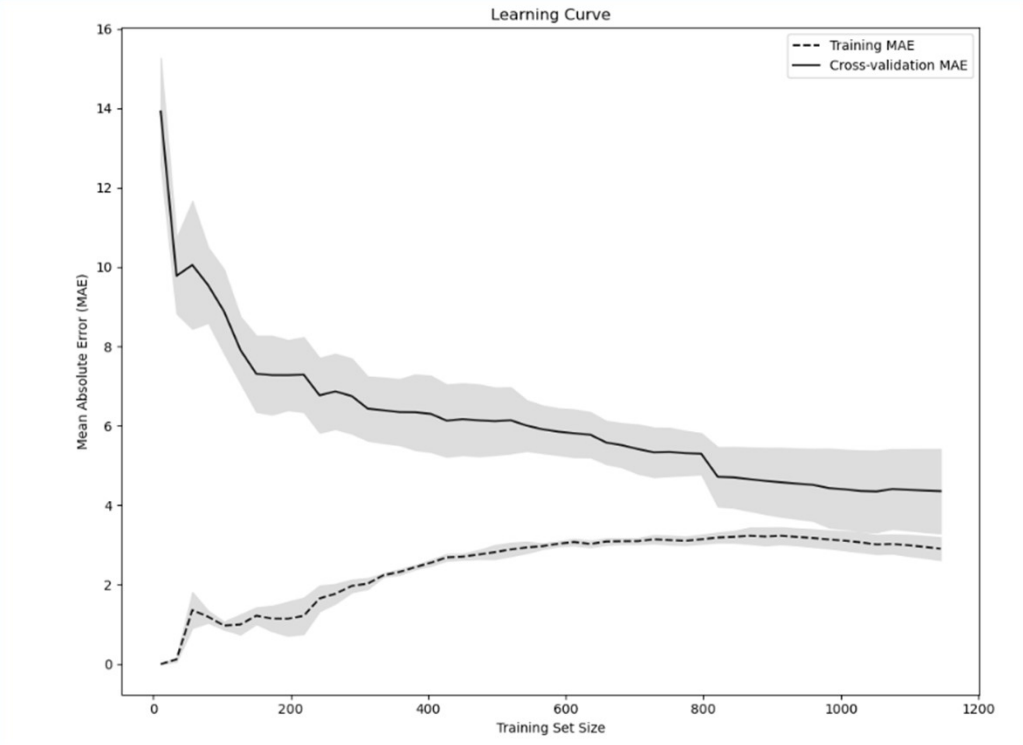
In [29]:
```
# Make a prediction using the Decision Tree model
prediction = model.predict(input_data).tolist()

# Print the predicted 'RequestedQuantity' on the test set
print("Prediction:", prediction)
```

```
Prediction: [46.666666666666664]
```

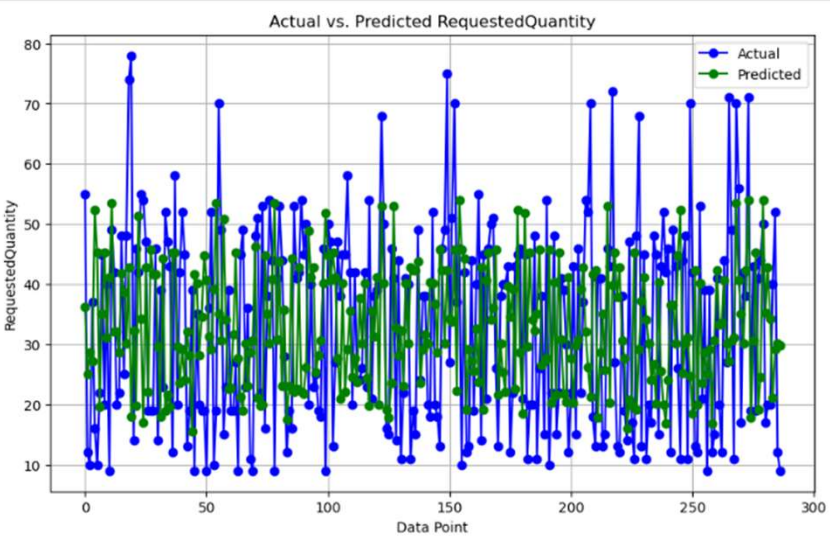
In [30]:
```
model.score(X_train, y_train)
```

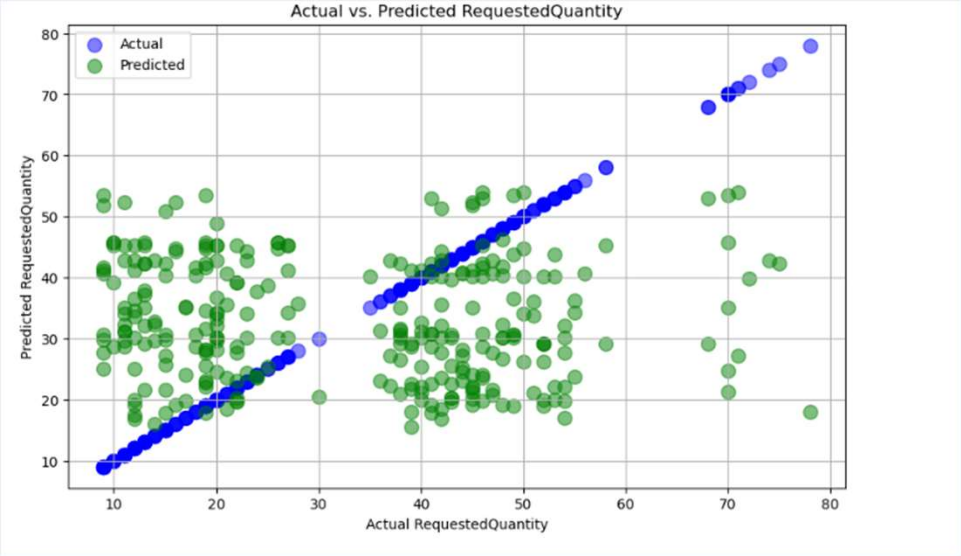
Out[30]:
```
0.8945812178633263
```

Decision Tree Regression



Decision Tree Regression





IT20665098
Prabodha K.W.D.S

Bachelor of Science (Hons) in Information Technology
Specializing in Software Engineering

Critical Medication Priority Recommender System

Introduction

- **Research Problem**

We're addressing medication supply challenges at Apeksha Hospital with a 'Critical Medication Priority Recommender System.' Using machine learning and optimization, we aim to identify crucial medications, predict shortages, optimize procurement, and elevate patient care quality

- **Proposed Solution**

The system uses machine learning to predict shortages, identify critical medications, and optimize recommendations at Apeksha Hospital, improving patient care. It consists of a Shortage Prediction Model and a Critical Medication Identification Model.

Critical Medication Priority Recommender System

Introduction

- **Main Objective**

"Developing an intelligent system for Apeksha Hospital using machine learning to predict shortages, identify critical medications, and optimize their procurement, thus enhancing patient care."

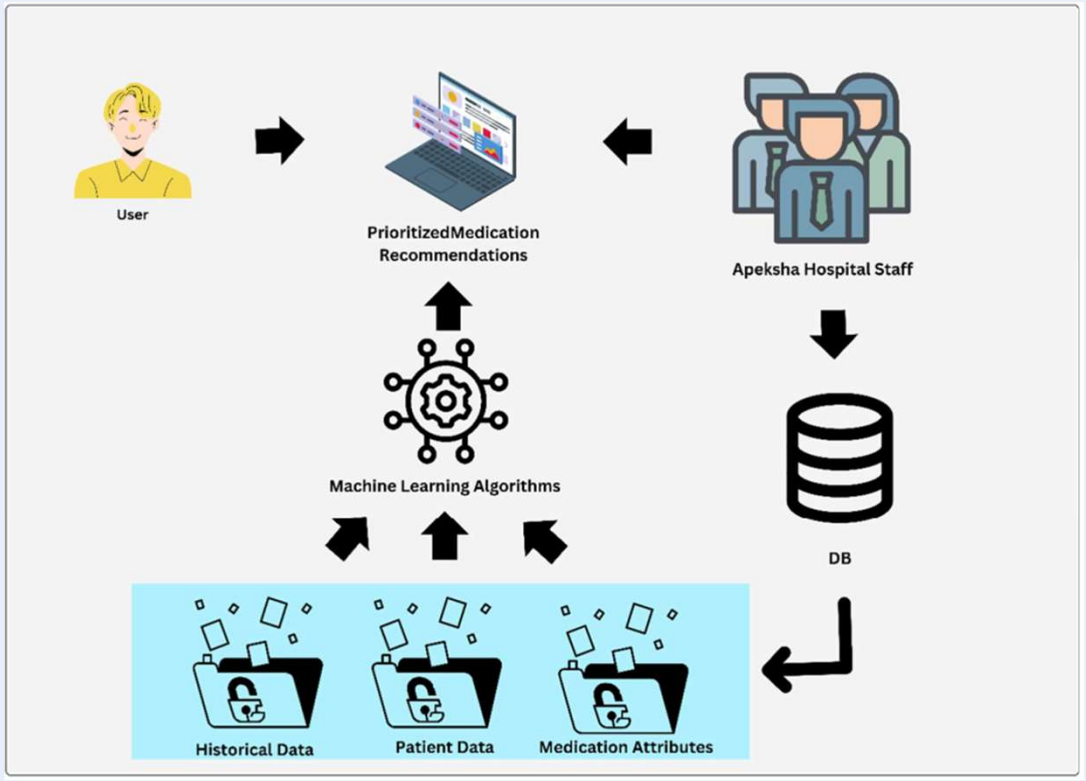
- **Specific Objectives**

1. Critical Medication Identification Model
2. Medication Shortage Prediction Model
3. Optimization Algorithms Integration
4. User-Friendly Interface

Critical Medication Priority Recommender System

Methodology

- **Component Diagram**



Critical Medication Priority Recommender System

Methodology

- **Tools And Technologies**

Programming Language

- Python

Tools

- Jupyter Notebook
- Anaconda Navigator
- Scikit Learn

Version Controlling

- GitHub

Algorithms

- Logistic Regression Algorithm
- Random Forest Classification

Critical Medication Priority Recommender System

Methodology

Sub objective 01 - Critical Medication Identification Model

Data Collection

- Collected data from Internet open data Platforms.

Data Pre-Processing

- Conversion of Categorical Variables to Numerical Format
- Date Column Conversion
- Date Feature Extraction

Implement Prediction Model

- Implement Machine Learning Model using "**Random Forest Classifier**" Algorithm

- Implement Machine Learning Model using "**Logistic Regression** " Algorithm

Critical Medication Priority Recommender System

Methodology

Sub objective 01 - Critical Medication Identification Model – Sample Codes

Random Forest Classifier **Accuracy- 56%**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Read the CSV file into a DataFrame
data = pd.read_csv('critical_medication.csv')

data['Shelf_Life'] = pd.to_datetime(data['Shelf_Life'])
data['Shelf_Life_Day'] = data['Shelf_Life'].dt.day
data['Shelf_Life_Month'] = data['Shelf_Life'].dt.month
data['Shelf_Life_Year'] = data['Shelf_Life'].dt.year

data.drop(['Shelf_Life'], axis=1, inplace=True)

# Define the features (predictors) and the target variable
X = data[['Patients', 'Usage','Emergency_Usage','Shelf_Life_Day','Shelf_Life_Month','Shelf_Life_Year']]
y = data['Importance_Level']

# 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)

# Create and train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)
print(f'Critical Medicines: {y_pred}')

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Generate a classification report
report = classification_report(y_test, y_pred)
print(report)
```

Logistic Regression Algorithm **Accuracy- 60%**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.linear_model import LogisticRegression

# Read the CSV file into a DataFrame
data = pd.read_csv('critical_medication.csv')

data['Shelf_Life'] = pd.to_datetime(data['Shelf_Life'])
data['Shelf_Life_Day'] = data['Shelf_Life'].dt.day
data['Shelf_Life_Month'] = data['Shelf_Life'].dt.month
data['Shelf_Life_Year'] = data['Shelf_Life'].dt.year

data.drop(['Shelf_Life'], axis=1, inplace=True)

# Define the features (predictors) and the target variable
X = data[['Patients', 'Usage','Emergency_Usage','Shelf_Life_Day','Shelf_Life_Month','Shelf_Life_Year']]
y = data['Importance_Level']

# 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)

log_reg=LogisticRegression(random_state=0).fit(X_train,y_train)

log_reg.predict(X_train)
```

Critical Medication Priority Recommender System

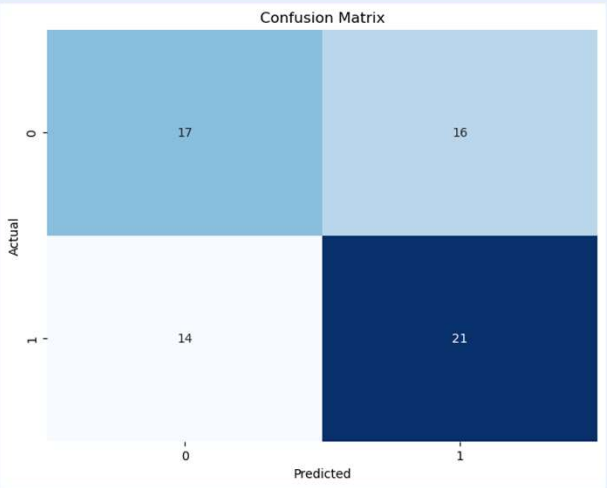
Methodology

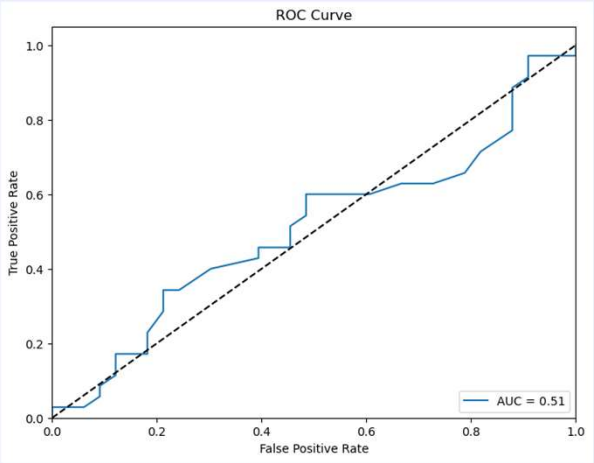
Sub objective 01 - Critical Medication Identification Model

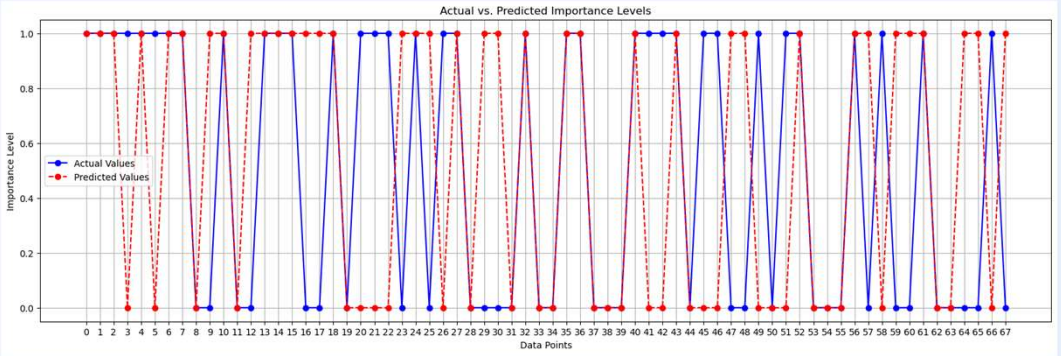
Random Forest Classifier

Accuracy: 0.56

	precision	recall	f1-score	support
0	0.55	0.52	0.53	33
1	0.57	0.60	0.58	35
accuracy			0.56	68
macro avg	0.56	0.56	0.56	68
weighted avg	0.56	0.56	0.56	68







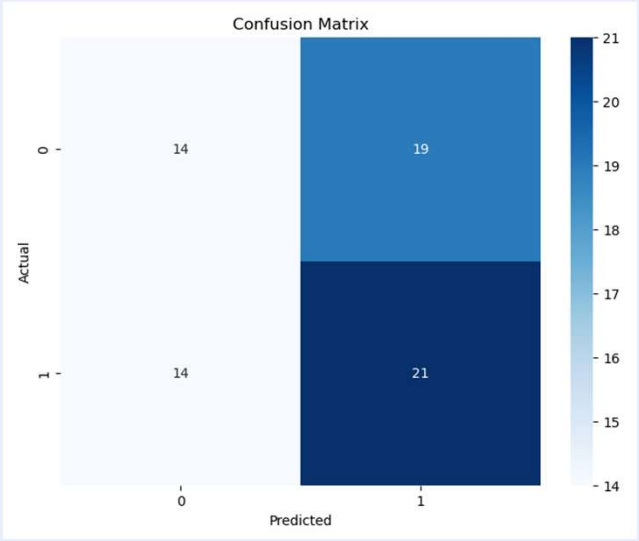
Critical Medication Priority Recommender System

Methodology

Sub objective 01 - Critical Medication Identification Model

Logistic Regression Algorithm

```
Training Accuracy: 0.6007462686567164
Testing Accuracy: 0.5147058823529411
Classification Report for Testing Data:
              precision    recall  f1-score   support

           0       0.50      0.42      0.46        33
           1       0.53      0.60      0.56        35

    accuracy                           0.51        68
   macro avg       0.51      0.51      0.51        68
weighted avg       0.51      0.51      0.51        68
```



Critical Medication Priority Recommender System

Methodology

Sub objective 02 - Medication Shortage Prediction Model

Data Collection

- Collected data from Internet open data Platforms.

Data Pre-Processing

- Conversion of Categorical Variables to Numerical format
- Date Column Conversion
- Date Feature Extraction
- Applied feature scaling using the StandardScaler from scikit-learn

Implement Prediction Model

- Implement Machine Learning Model using "**Logistic Regression** " Algorithm

Critical Medication Priority Recommender System

Methodology

Sub objective 02 - Medication Shortage Prediction Model – Sample Code

Logistic Regression Algorithm

Accuracy- 78%

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

df=pd.read_csv("shortage_data.csv")

df['Shelf_Life'] = pd.to_datetime(df['Shelf_Life'])
df['Admin_Timestamp'] = pd.to_datetime(df['Admin_Timestamp'])

df['Shelf_Life_Day'] = df['Shelf_Life'].dt.day
df['Shelf_Life_Month'] = df['Shelf_Life'].dt.month
df['Shelf_Life_Year'] = df['Shelf_Life'].dt.year

df['Admin_Day'] = df['Admin_Timestamp'].dt.day
df['Admin_Month'] = df['Admin_Timestamp'].dt.month
df['Admin_Year'] = df['Admin_Timestamp'].dt.year

df.drop(['Shelf_Life', 'Admin_Timestamp'], axis=1, inplace=True)
X=df.drop(columns='Medication Shortage')

y=df['Medication Shortage']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=21)
X_train

scaler=StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled=scaler.transform(X_test)
X_train_scaled

log_reg=LogisticRegression(random_state=0).fit(X_train_scaled,y_train)
log_reg.predict(X_train_scaled)
log_reg.score(X_train_scaled, y_train)
```

Critical Medication Priority Recommender System

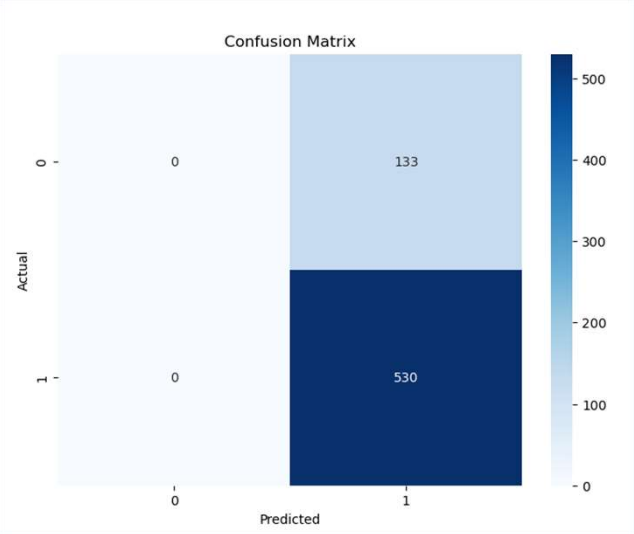
Methodology

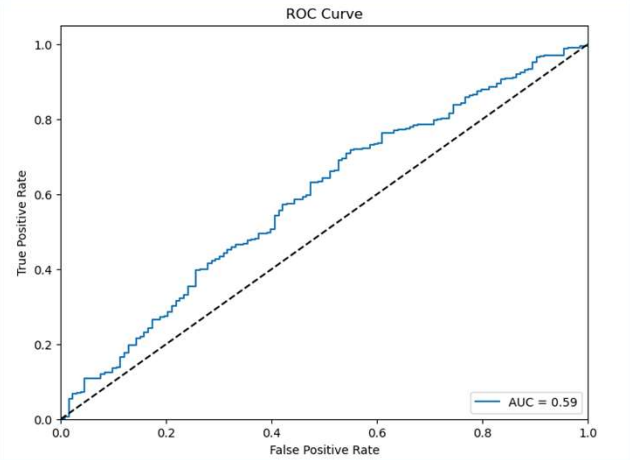
Sub objective 02 - Medication Shortage Prediction Model

Logistic Regression Algorithm

```
Training Accuracy: 0.78409825468649
Testing Accuracy: 0.799396681749623
Classification Report for Testing Data:
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	133
1	0.80	1.00	0.89	530
accuracy			0.80	663
macro avg	0.40	0.50	0.44	663
weighted avg	0.64	0.80	0.71	663





Critical Medication Priority Recommender System

Completion of the components

- ✓ Train Critical Medication Identification Model
- ✓ Train Medication Shortage Prediction Model

50%

- Optimization Algorithms Integration
- Implement User-Friendly Interfaces

50%

IT20662028
Bandara H.R.H.S

Bachelor of Science (Hons) in Information Technology
Specializing in Software Engineering

Introduction

Blood Donation Campaign Organizing Challenges

- There is High Demand for blood in Apeksha hospital.
- Organize donation Campaigns without Prediction.
- Food Wastage and Crowd Management Issues.
- Waste campaign marketing cost and another cost.

Research Problem & Solution

- Problem:

Apeksha Hospital's blood donation campaigns face challenges due to low donor participation and resource inefficiencies.

- Solution:

Using predictive analytics and machine learning, Apeksha Hospital can boost campaign effectiveness by predicting success factors and optimizing resource allocation.

Specific and Sub Objective

- **Main Objective**

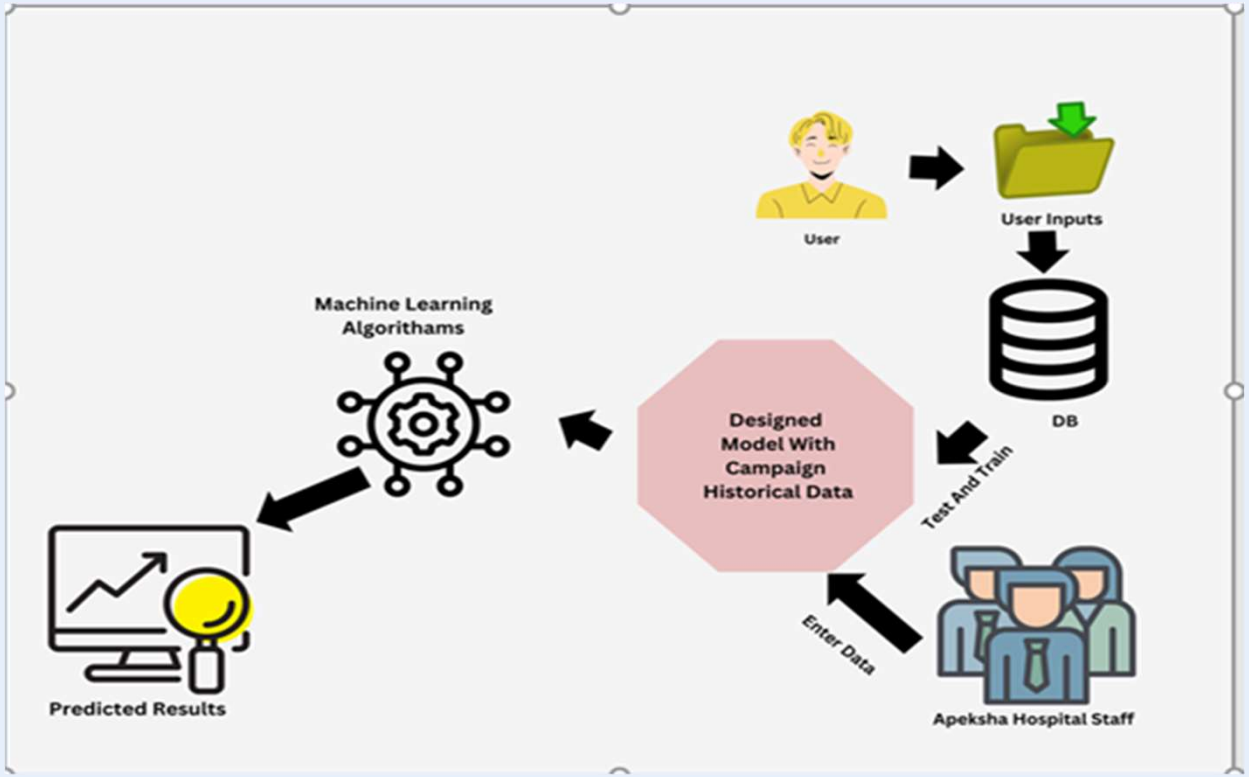
Implement machine learning to enhance the efficiency of blood donation campaigns.

- **Specific Objectives**

1.Optimization Algorithms Integration
2.Predict People amount
3.User-Friendly Interface

Methodology

- **Component Diagram**



Methodology

- **Tools And Technologies**

Programming Language

- ❖ Python
- ❖ React native

Tools

- ❖ Jupyter Notebook
- ❖ Anaconda Navigator
- ❖ Scikit Learn
- ❖ Joblib

Version Controlling

- ❖ GitHub

Algorithm

- ❖ `RandomForestRegressor`

Methodology
Evidence of Completion

Data collection

Data pre-processing

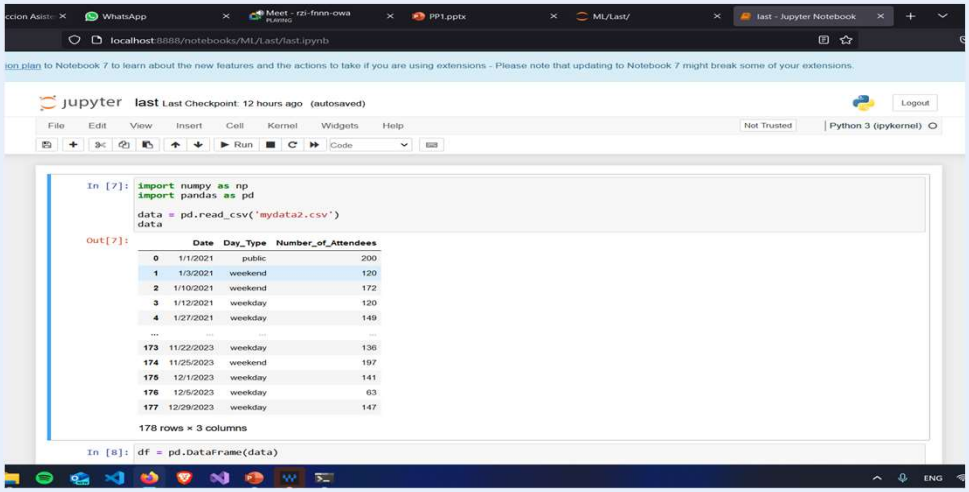
Used Random Forest Regressor Algorithm for training models

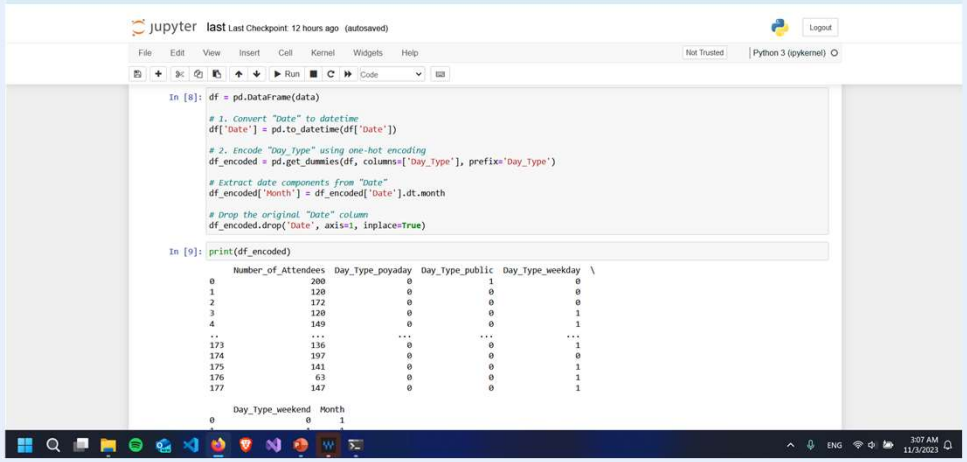
Data Visualization

Host the Current model in the flask server and Azure cloud

Displayed the prediction in mobile application

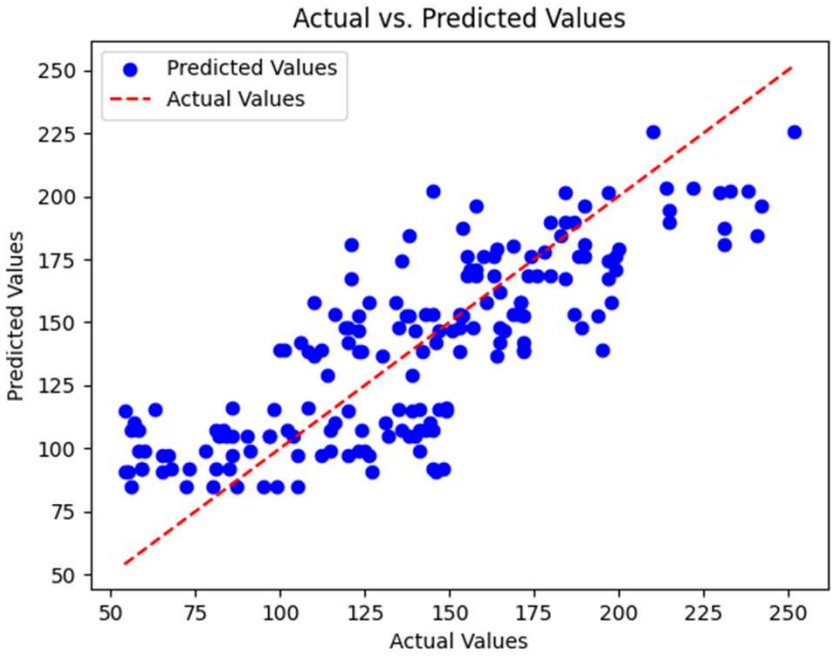
Data Collection and Pre-processing

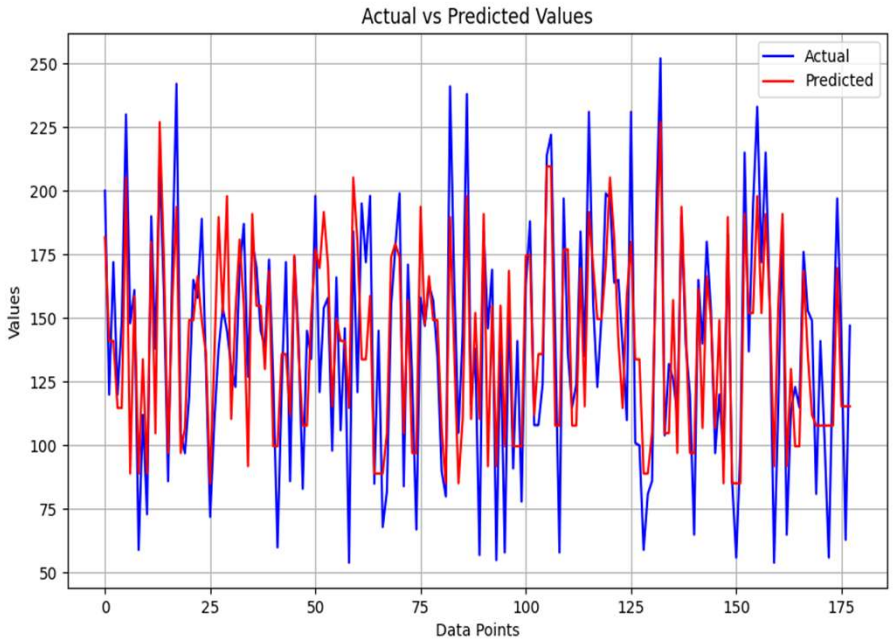




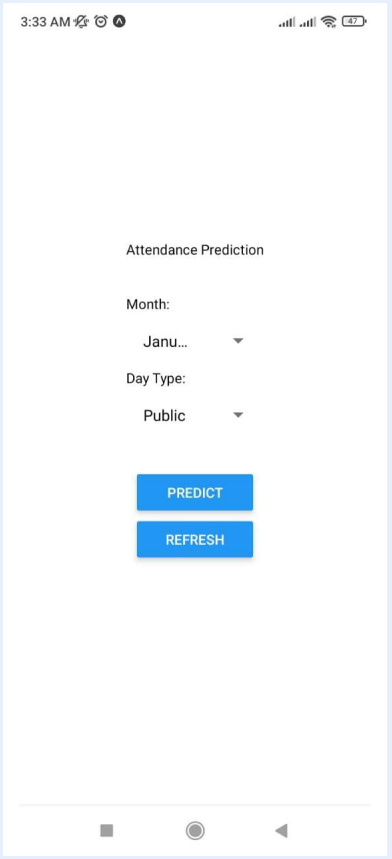
Random Forest Regressor Accuracy

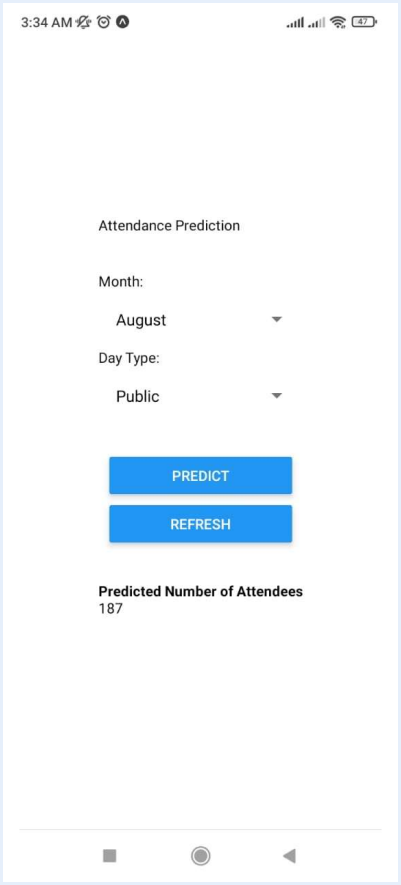
```
Enter the month (1-12): 1
Is it a Poyaday? (0 or 1): 0
Is it a public holiday? (0 or 1): 1
Is it a weekday? (0 or 1): 0
Is it a weekend? (0 or 1): 0
Predicted Number of Attendees: 178.09783333333334
R-squared (R²) on the entire dataset: 0.6474175991009472
```





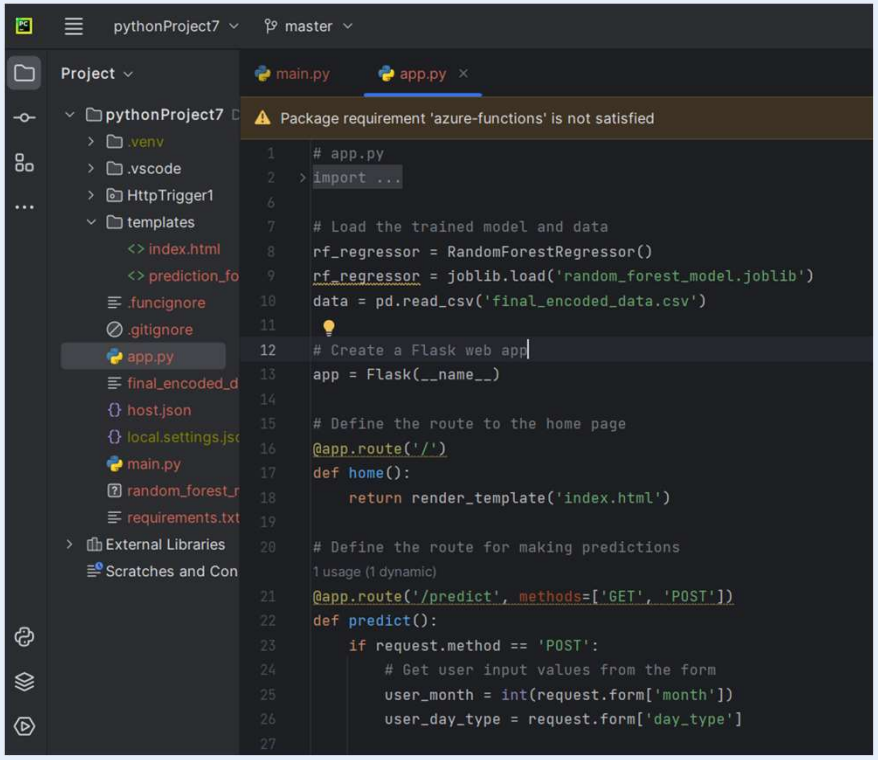
Display the results in mobile applications



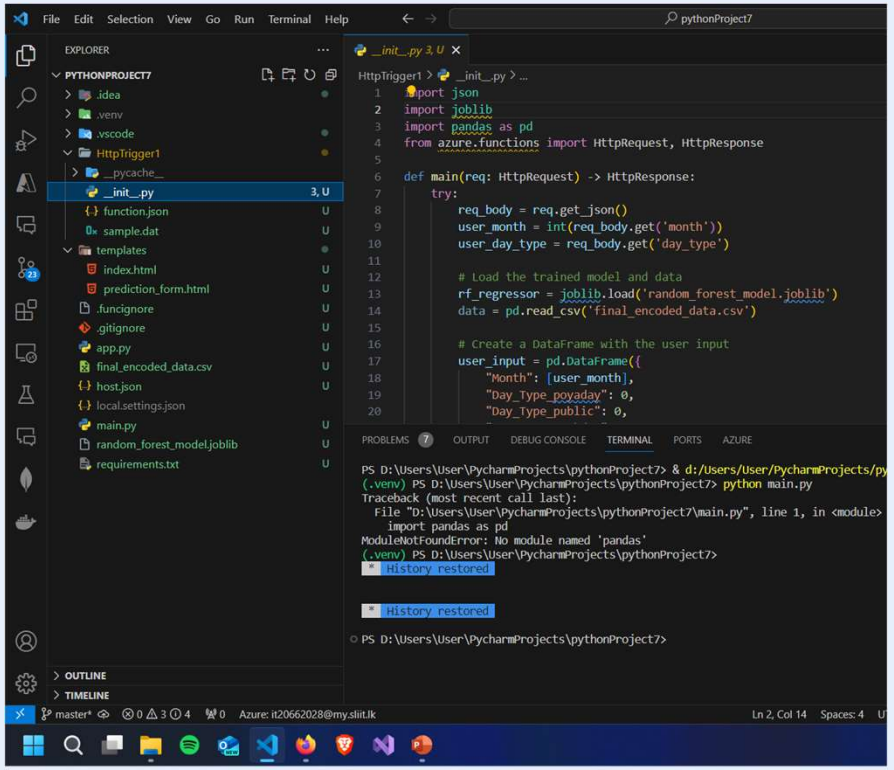


Host the Current model in the flask server and Azure cloud

Flask server



Azure cloud



Host the Current model in the flask server and Azure cloud

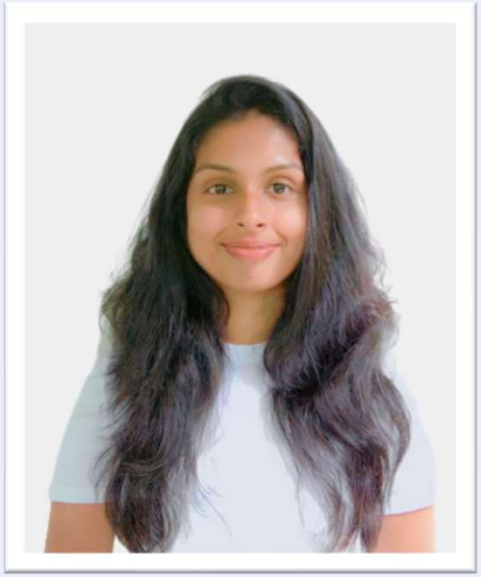
Completion of the components

- Identification of best algorithm for people amount prediction
- Completed the show result part in mobile application
- Host the trained model in the azure cloud

Future Implementations

- Collecting test dataset with a variety
- Implement the web application for hospital users

IT20660352
Wijesooriya P.L.P.G.D.S

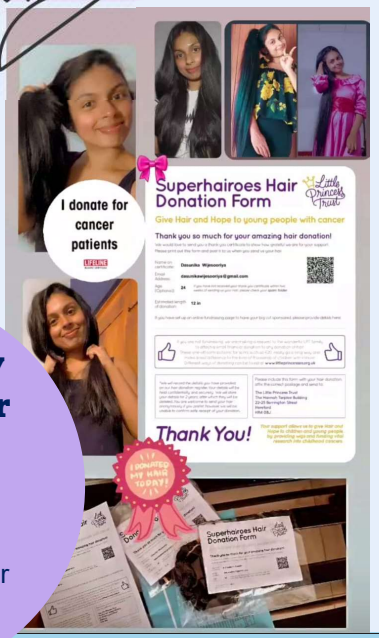


Bachelor of Science (Hons) in Information Technology
Specializing in Software Engineering

Promoting Quality Hair Donation for Cancer Patients

Introduction
Background

Apeksha hospital-based hair donation process currently has no specific method. Applying to build the system to obtain hair from hair donors according to the standards recommended by the hospital and to eliminate the distance between the hospital and the donor. Using CNN Architecture in image processing.



Research
Problem

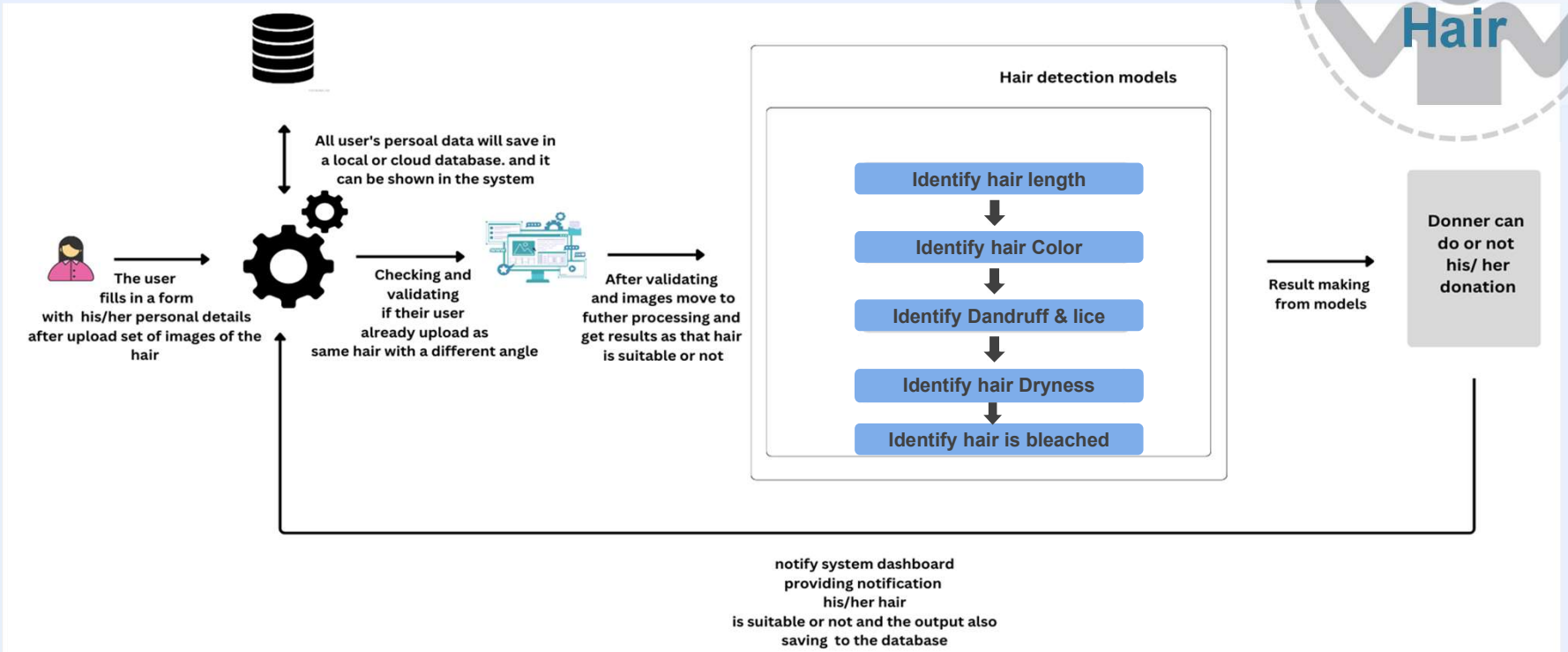
How to determine whether hair donated by hair donors meets the standards and qualities of hair recommended by the Apeksha hospital?

I donated my hair to cancer patients.

But,

Apeksha Hospital could not donate hair due to the informal procedure there.

System Architecture



Methodology

- Data collection
  - Data Pre Processing and Create Data Set With Data Augmentation
- Used transfer learning based Convolutional Neural Network(CNN) architectures for training models
  - Data augmentation and selected the best technique to retrain models to achieve good fitting
- Tuning hyper-parameters and retrained models to select the best architecture
  - Data Pre Processing and Create Data Set With Data Augmentation
- Data Visualization
  - Host the finalized model in the flask server
- Displayed the results in web application

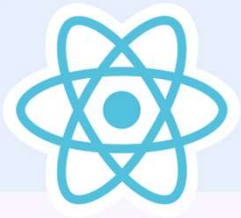
Used Techniques and Technologies

Techniques

- Transfer learning
- Data Augmentation
- Normalization

Technologies

- React
- Python
- TensorFlow
- Anaconda Environment
- Node Server
- Jupyter Notebook
- Google Colab
- Visual Studio Code
- Keras






Specific and Sub Objective

Specific Objective

Implementation of Deep Learning Algorithm (CNN) with online platform to examine key quality factors of hair. – main objective

- Model Implemented to identify of Hair type - 1.1
- Model Implemented to identify Color of Hair -1.2
- Model Implemented to identify bleached of Hair – 1.3
- Model Implemented to identify Dryness of Hair – 1.4
- Model Implemented to identify lice & dandruff of Hair -1.5

Sub Objectives

01 - Collected data which needs to train the prediction models

02 - Combine all the models to implement the final model

03 - Implement the Web Application with User-friendly Interface.

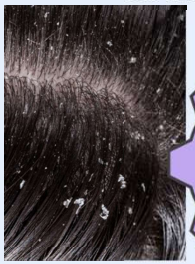
Implementation of the model

Training Data Set

Test Data Set

Model training to detect Hair Color (White, Black)

Model training to detect Hair Type (Long, Short)



Model training to detect dandruff and Lice

Final Model that will predict Quality with Hair

Preprocessing for the image data sets

In [4]:
```
import cv2
import imghdr
```

In [6]:
```
image_exts = ['jpeg','jpg', 'bmp', 'png']
```

In [7]:
```
data_dir='C:/Users/Ridma/Downloads/Hairs'
for image_class in os.listdir(data_dir):
    for image in os.listdir(os.path.join(data_dir, image_class)):
        image_path = os.path.join(data_dir, image_class, image)
        try:
            img = cv2.imread(image_path)
            tip = imghdr.what(image_path)
            if tip not in image_exts:
                print('Image not in ext list {}'.format(image_path))
                os.remove(image_path)
        except Exception as e:
            print('Issue with image {}'.format(image_path))
            # os.remove(image_path)
```

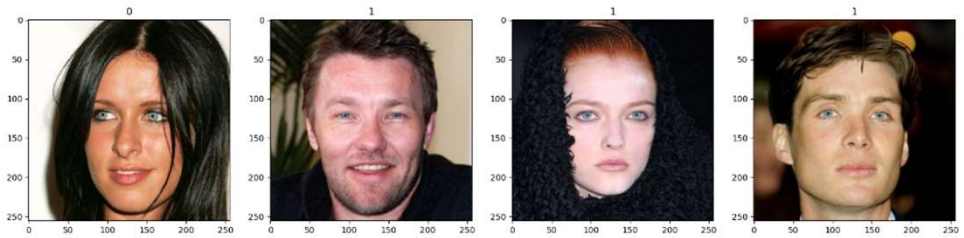
In [ ]:
```
datagen = preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
```

01.Model for the Hair Type Detection

According to the hair donation rules of Apeksha Hospital, they request hair longer than 12 inches. Because this system is not successful in measuring the inch size in practice, a model is used to identify the donor's hair type as short hair and long hair. Only long hair can be donated here.

- To implement this model the data was collected through a Kaggle data set.

```
In [16]: data_iterator = data.as_numpy_iterator()

In [17]: batch = data_iterator.next()

In [18]: fig, ax = plt.subplots(ncols=4, figsize=(20,20))
         for idx, img in enumerate(batch[0][:4]):
             ax[idx].imshow(img.astype(int))
             ax[idx].title.set_text(batch[1][idx])
```



Proposed Model

Classes	Train, Validation and Test data
Short Hair	Train - 478 Validation - 67
Long Hair	Train - 430 Validation - 76

CNN Models

CNN Architectures - VGG16

In [4]:
```
train_generator = datagen.flow_from_directory(
    'C:/Users/Ridma/Downloads/Hairs',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)
```

```
Found 858 images belonging to 2 classes.
```

In [5]:
```
val_generator = datagen.flow_from_directory(
    'C:/Users/Ridma/Downloads/Hairs',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 213 images belonging to 2 classes.
```

In [6]:
```
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss', patience=10, verbose=1, restore_best_weights=True)
```

In [7]:
```
base_model = VGG16(input_shape=(160, 160, 3), include_top=False, weights='imagenet')
base_model.trainable = False

vgg16_model = models.Sequential([
    base_model,
    layers.Flatten(),
    layers.Dense(4096, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(4096, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(2, activation='softmax')
])

vgg16_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history=vgg16_model.fit(train_generator, validation_data=val_generator, epochs=20 , callbacks=[early_stop])
```

```
Epoch 1/20
27/27 [==============================] - 163s 6s/step - loss: 5.3365 - accuracy: 0.6084 - val_loss: 0.3546 - val_accuracy: 0.87
```

CNN Architectures - MobileNetV2

In [38]:
```
train_generator = datagen.flow_from_directory(
    'C:/Users/Ridma/Downloads/Hairs',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)
```

```
Found 858 images belonging to 2 classes.
```

In [39]:
```
val_generator = datagen.flow_from_directory(
    'C:/Users/Ridma/Downloads/Hairs',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 213 images belonging to 2 classes.
```

In [40]:
```
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss', patience=10, verbose=1, restore_best_weights=True)
```

In [10]:
```
base_model = MobileNetV2(input_shape=(160, 160, 3), include_top=False, weights='imagenet')
base_model.trainable = False

mobilenet_model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(2, activation='softmax')
])

mobilenet_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history=mobilenet_model.fit(train_generator, validation_data=val_generator, epochs=20 ,callbacks=[early_stop])
```

```
Epoch 1/20
14/14 [==============================] - 26s 1s/step - loss: 0.2733 - accuracy: 0.9021 - val_loss: 0.1966 - val_accuracy: 0.915
5
Epoch 2/20
14/14 [==============================] - 16s 1s/step - loss: 0.1027 - accuracy: 0.9732 - val_loss: 0.1247 - val_accuracy: 0.962
4
```

Transfer learning Architectures

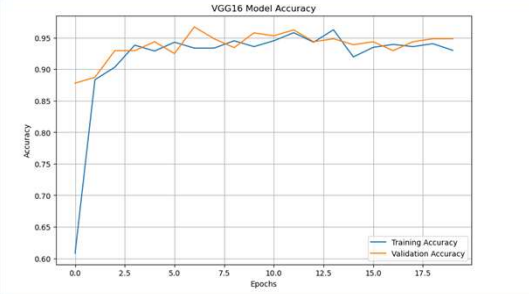
Evidence of Completion

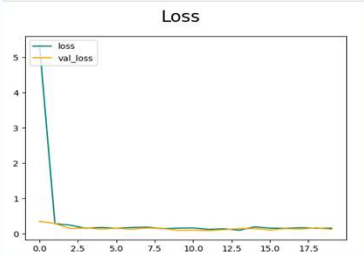
CNN Architectures

- **VGG16**
- **MobileNetV2**

VGG16

Best Architecture

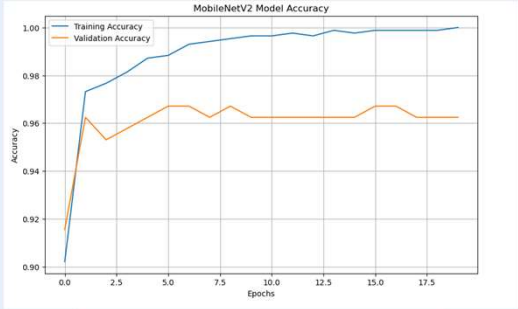


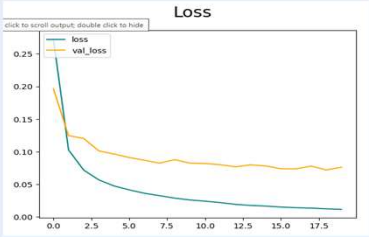


Loss: 0.2021

Accuracy: 0.9345

MobileNetV2





Loss: 0.0105

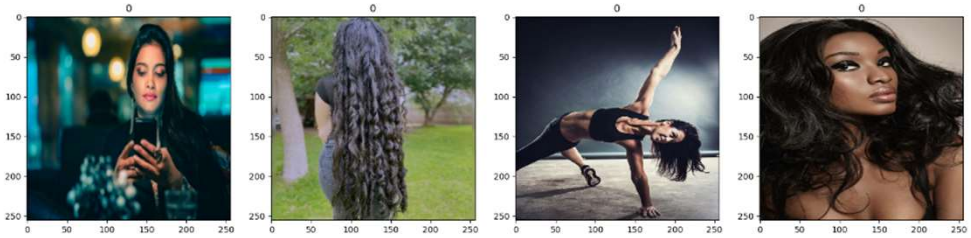
Accuracy: 0.9968

02.Model for the Hair Color Detection

According to the hair donation rules of Apeksha Hospital, currently they only receive black hair. Gray hair or dyed hair is not accepted. This system uses a model to identify the color of the donor's hair. Only black color hair can be donated here.

- To implement this model the data was collected through istockphoto data set.

```
In [5]: data = tf.keras.utils.image_dataset_from_directory('C:/Users/Ridma/Downloads/newimages')
        Found 989 files belonging to 2 classes.

In [6]: data_iterator = data.as_numpy_iterator()

In [7]: batch = data_iterator.next()

In [8]: fig, ax = plt.subplots(ncols=4, figsize=(20,20))
        for idx, img in enumerate(batch[0][:4]):
            ax[idx].imshow(img.astype(int))
            ax[idx].title.set_text(batch[1][idx])
```



Proposed Model

Classes	**Train and Validation data**
Black Hair	**Train - 460** **Validation -110**
White Hair	**Train - 440** **Validation - 108**

CNN Models

CNN Architectures - VGG16

In [6]:
```
train_generator = datagen.flow_from_directory(
    'C:/Users/Ridma/Downloads/newimages',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)
```

```
Found 792 images belonging to 2 classes.
```

In [7]:
```
val_generator = datagen.flow_from_directory(
    'C:/Users/Ridma/Downloads/newimages',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 197 images belonging to 2 classes.
```

In [8]:
```
base_model = VGG16(input_shape=(160, 160, 3), include_top=False, weights='imagenet')
base_model.trainable = False

vgg16_model = models.Sequential([
    base_model,
    layers.Flatten(),
    layers.Dense(4096, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(4096, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(2, activation='softmax')  # Assuming 2 hair colors
])

vgg16_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history=vgg16_model.fit(train_generator, validation_data=val_generator, epochs=20)
```

```
Epoch 1/20
13/13 [==============================] - 209s 16s/step - loss: 8.1964 - accuracy: 0.5139 - val_loss: 0.7592 - val_accuracy: 0.5
```

CNN Architectures - MobileNetV2

In [13]:
```
train_generator = datagen.flow_from_directory(
    'C:/Users/Ridma/Downloads/newimages',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)
```

```
Found 792 images belonging to 2 classes.
```

In [14]:
```
val_generator = datagen.flow_from_directory(
    'C:/Users/Ridma/Downloads/newimages',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 197 images belonging to 2 classes.
```

In [15]:
```
base_model = MobileNetV2(input_shape=(160, 160, 3), include_top=False, weights='imagenet')
base_model.trainable = False

mobilenet_model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(2, activation='softmax')  # Assuming 2 hair colors
])

mobilenet_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history=mobilenet_model.fit(train_generator, validation_data=val_generator, epochs=10)
```

```
Epoch 1/10
13/13 [==============================] - 42s 3s/step - loss: 0.6384 - accuracy: 0.6616 - val_loss: 0.3875 - val_accuracy: 0.852
8
```

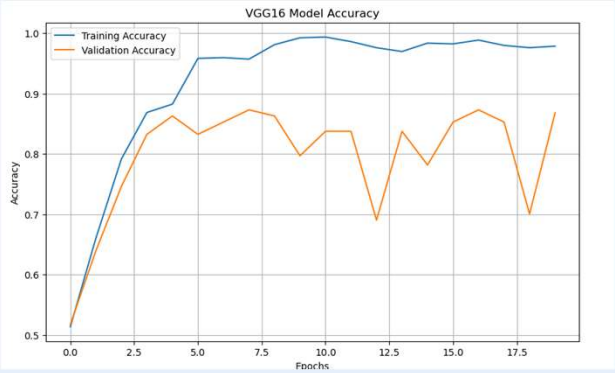
Transfer learning Architectures

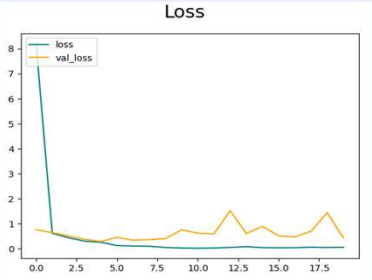
Evidence of Completion

CNN Architectures

- **VGG16**
- **MobileNetV2**

VGG16



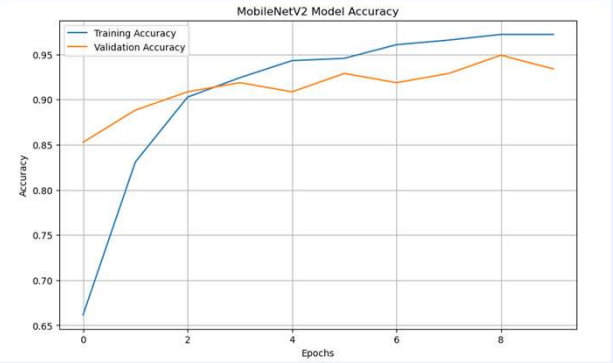


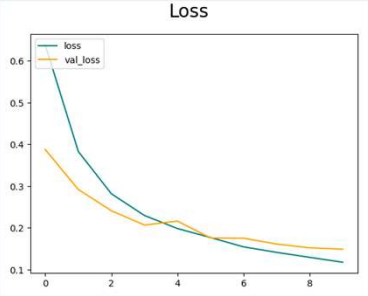
Loss: 0.0576

Accuracy: 0.9785

MobileNetV2

Best Architecture





Loss: 0.1177

Accuracy: 0.9722

03.Model for the Dandruff & Lice Detection

As per hair donation rules at Apeksha Hospital, they require clean, healthy hair free from dandruff and lice infestation. This system checks whether the donor's hair is healthy and free from dandruff and lice.

- To implement this model the data was collected through istockphoto data set.

```
[12]: data_iterator = data.as_numpy_iterator()

[13]: batch = data_iterator.next()

[14]: fig, ax = plt.subplots(ncols=4, figsize=(20,20))
      for idx, img in enumerate(batch[0][:4]):
          ax[idx].imshow(img.astype(int))
          ax[idx].title.set_text(batch[1][idx])
```

Proposed Model

Classes	Train, Validation and Test data
Lice & Dandruff Hair	Train - 160 Validation -48
Clean Hair	Train - 150 Validation - 40

CNN Models

CNN Architectures - VGG16

```
In [5]: train_generator = datagen.flow_from_directory(
            'C:/Users/Ridma/Downloads/Dandruff',
            target_size=img_size,
            batch_size=batch_size,
            class_mode='categorical',
            subset='training'
        )

Found 138 images belonging to 2 classes.

In [6]: val_generator = datagen.flow_from_directory(
            'C:/Users/Ridma/Downloads/Dandruff',
            target_size=img_size,
            batch_size=batch_size,
            class_mode='categorical',
            subset='validation'
        )

Found 34 images belonging to 2 classes.

In [7]: from tensorflow.keras.callbacks import EarlyStopping

        early_stop = EarlyStopping(monitor='val_loss', patience=10, verbose=1, restore_best_weights=True)

In [8]: base_model = VGG16(input_shape=(160, 160, 3), include_top=False, weights='imagenet')
        base_model.trainable = False

        vgg16_model = models.Sequential([
            base_model,
            layers.Flatten(),
            layers.Dense(4096, activation='relu'),
            layers.Dropout(0.5),
            layers.Dense(4096, activation='relu'),
            layers.Dropout(0.5),
            layers.Dense(2, activation='softmax')
        ])

        vgg16_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

        history=vgg16_model.fit(train_generator, validation_data=val_generator, epochs=20 , callbacks=[early_stop])

        Epoch 1/20
```

CNN Architectures - MobileNetV2

```
In [25]: train_generator = datagen.flow_from_directory(
             'C:/Users/Ridma/Downloads/Dandruff',
             target_size=img_size,
             batch_size=batch_size,
             class_mode='categorical',
             subset='training'
         )

Found 138 images belonging to 2 classes.

In [26]: val_generator = datagen.flow_from_directory(
             'C:/Users/Ridma/Downloads/Dandruff',
             target_size=img_size,
             batch_size=batch_size,
             class_mode='categorical',
             subset='validation'
         )

Found 34 images belonging to 2 classes.

In [27]: from tensorflow.keras.callbacks import EarlyStopping

         early_stop = EarlyStopping(monitor='val_loss', patience=10, verbose=1, restore_best_weights=True)

In [28]: base_model = MobileNetV2(input_shape=(160, 160, 3), include_top=False, weights='imagenet')
         base_model.trainable = False

         mobilenet_model = models.Sequential([
             base_model,
             layers.GlobalAveragePooling2D(),
             layers.Dense(2, activation='softmax')
         ])

         mobilenet_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
         history=mobilenet_model.fit(train_generator, validation_data=val_generator, epochs=20, callbacks=[early_stop])

         Epoch 1/20
         18/18 [==============================] - 18s 657ms/step - loss: 0.5687 - accuracy: 0.7101 - val_loss: 0.1855 - val_accuracy: 0.
         9412
         Epoch 2/20
```

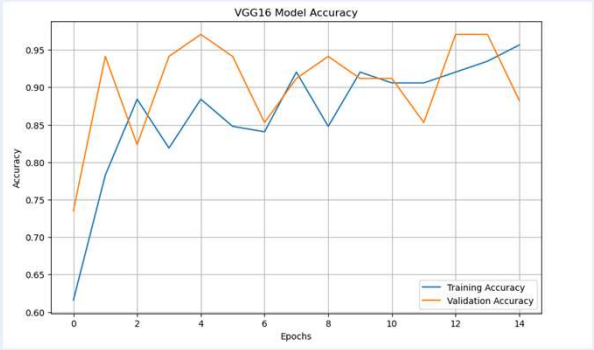
Transfer learning Architectures

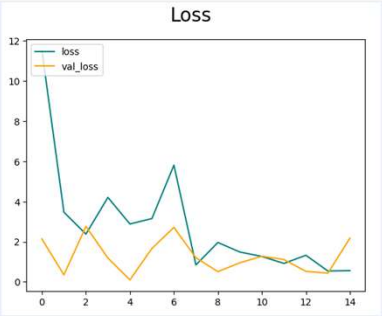
Evidence of Completion

CNN Architectures

- **VGG16**
- **MobileNetV2**

VGG16



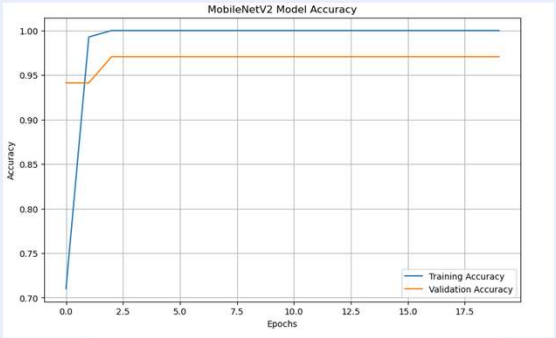


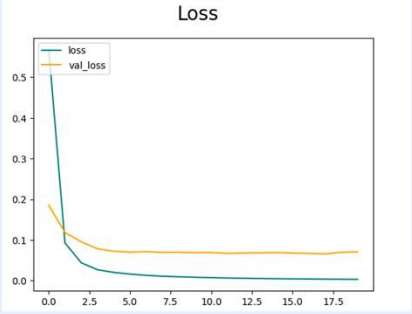
Loss: 0.5594

Accuracy: 0.9565

MobileNetV2

Best Architecture





Loss:0.0032

Accuracy: 1.0000

Completion and Future works

Completion of the components

50%

- Collected data which needs to train the prediction models - Sub Objective 1
- Identification of best architecture for transfer learning
- Model Implemented to identify Color of Hair – Main Objective 1.2
- Model Implemented to identify length type of Hair - Main Objective 1.1
- Model Implemented to identify Dandruff & lice of Hair - Main Objective 1.5

Future Implementations

50%

- Model Implemented to identify Dryness of Hair - Main Objective 1.4
- Model Implemented to identify **bleached** of Hair - Main Objective 1.3
- Combine all the models to implement the final model - Sub Objective 2
- Implement the Web Application with User-friendly Interface - Sub Objective 3

References

[01]R. Mathew, A. Saripella, J. Faig, and JM. Fagan, "Increasing the Efficiency Within the Charitable Hair Donation Process," 2010. [Online]. Available: http://rucore.libraries.rutgers.edu/

[02]E.C. Herbertson, C.D. Lahiri, J.N. Nwogu, et al., "High acceptability of donating hair and other biological samples for research among people living with HIV in an outpatient clinic in Lagos, Nigeria," AIDS Research and Therapy, 2021. [Online]. Available: https://www.liebertpub.com/

[03]D.B. Burfeind, "Locks of Love seeks hair donations to make wigs for children with hair loss," Dermatology Nursing, 2007. [Online]. Available: https://go.gale.com/

[04]N.B. Silverberg, "Helping children cope with hair loss," Cutis, 2006. [Online]. Available: https://cdn.mdedge.com/

[05]LSG Kovásznay and HM Joseph, "Image processing," Proceedings of the IRE, 1955. [Online]. Available: https://ieeexplore.ieee.org/