Name - Barsha Baibhabi
Roll No - 22053416

Random Forest Model Integrated with Website using FastAPI

1. Overview

This project implements a Random Forest Classifier to predict fitness levels using the Linnerud Dataset from `sklearn.datasets`. The dataset includes physiological attributes, and the model predicts the number of sit-ups an individual can perform based on three key features.

Files in the Project:

- **`RandomForest.ipynb`** – Trains the model and saves it using joblib.
- **`main.py`** – FastAPI backend for handling predictions.
- **`index.html`** – Web frontend for user input and displaying results.
- **RandomForest.py** - Python script version of the Jupyter Notebook.
- **Random_forest_model.pkl** - Saved logistic regression models.

2. Installation & Setup

Prerequisites

Ensure you have Python 3 installed. Install the required dependencies:

```
pip install fastapi uvicorn joblib scikit-learn pandas
```

Running the Application

1. Train the model and generate the necessary files by running `model.ipynb`.
2. Start the FastAPI backend:

uvicorn main:app --host 127.0.0.1 --port 8006 --reload

3. Open `index.html` in a browser and test predictions.

3. Training the Model & Generating Pickle File

The `model.ipynb` notebook performs the following:

- Loads the Linnerud dataset from `sklearn.datasets`.
- Selects three key features:
  - Weight (kg)
  - Waist circumference (cm)
  - Pulse rate (bpm)
- Sets Sit-ups count as the target variable.
- Applies feature scaling using `StandardScaler()`.
- Trains a Random Forest Classifier.
- Saves the trained model (`random_forest_model.joblib`) and scaler (`scaler.joblib`).

4. FastAPI Backend (`main.py`)

The backend is implemented using FastAPI to:

- Load the trained model and scaler from `joblib` files.
- Accept feature inputs from the frontend via a POST request.
- Scale input features and return the predicted sit-ups count.

```
model = joblib.load("random_forest_model.pkl")

@app.post("/predict")

async def predict(features: dict):

    try:

        input_features = np.array(features["features"]).reshape(1, -1)


        if input_features.shape[1] != 3:

            return {"error": "Expected 3 features: Weight, Waist, Pulse"}
```

```
        prediction = model.predict(input_features).tolist()

        return {"prediction": prediction}

    except Exception as e:

        return {"error": str(e)}
```

- Runs on `http://127.0.0.1:8006/`

5. Frontend (`index.html`) with Inline CSS & JavaScript

The frontend:

- Provides an input form with three fields for feature values.
- Uses JavaScript to send AJAX requests to the FastAPI backend.

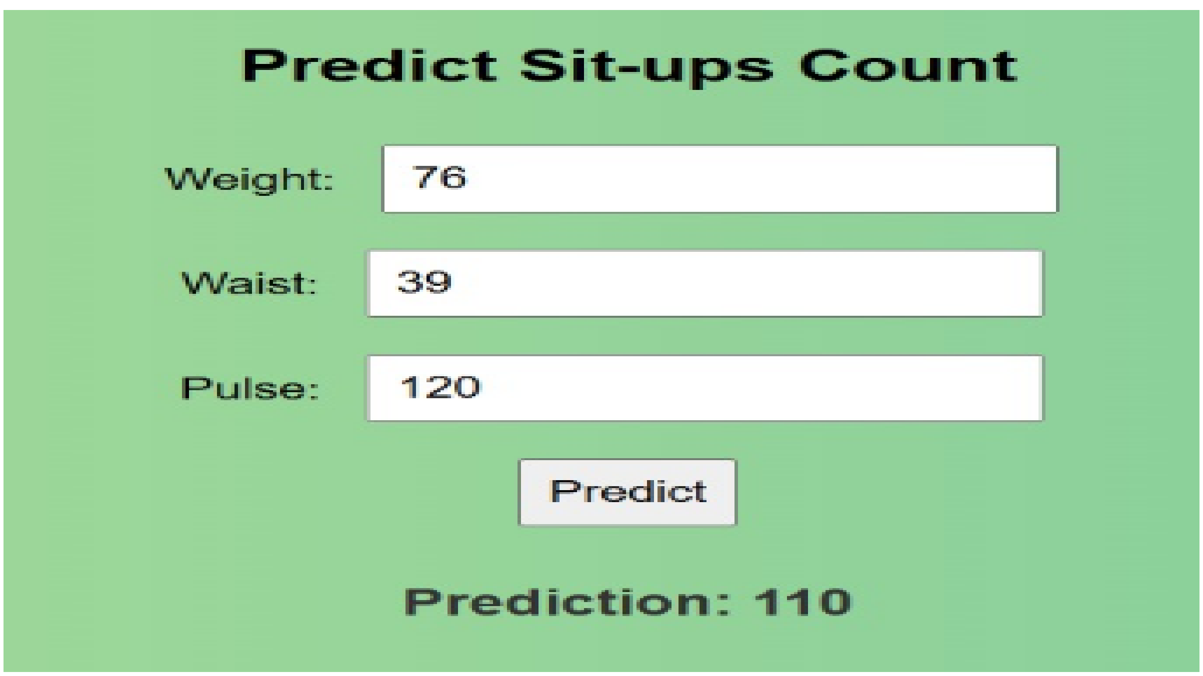
```
 document.getElementById("predictBtn").addEventListener("click",
function() {

            let features = [

                parseFloat(document.getElementById("weight").value),

                parseFloat(document.getElementById("waist").value),

                parseFloat(document.getElementById("pulse").value)

            ];



            fetch("http://127.0.0.1:8084/predict", {

                method: "POST",

                headers: { "Content-Type": "application/json" },

                body: JSON.stringify({ features: features })

            })
```

```
            .then(response => response.json())

            .then(data => {

                document.getElementById("output").innerText = "Prediction: " + data.prediction;

            })

            .catch(error => console.error("Error:", error));

        });
```

- Displays the predicted sit-ups count on the webpage.

6. Testing the Integration

- Enter values for Weight, Waist Circumference, and Pulse in `index.html`.
- Click Predict to send a request to `FastAPI`.
- The API returns the predicted sit-ups count.
- The result is displayed on the webpage.



7. Conclusion

This project successfully integrates a Random Forest model with a FastAPI backend and a simple web frontend. It provides a foundation for further enhancements, such as improved UI, database integration, and additional model tuning.