1. Importing Libraries

python Copy code

```python
from flask import Flask, render_template, request
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

- **Flask:** A lightweight web framework to create and manage the web application.
- **NumPy:** To handle numerical data and arrays.
- **LogisticRegression:** A machine learning algorithm to predict whether a patient needs a C-section or not.
- **StandardScaler:** To normalize the input features for better model performance.

---

2. Initializing the Flask Application

python Copy code

```python
app = Flask(__name__)
```

- `Flask(__name__)` initializes the application. The `__name__` helps Flask locate resources (e.g., templates).

3. Dataset Setup

python Copy code

```python
fetal_distress = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
maternal_age = [25, 30, 35, 20, 28, 32, 38, 22, 27, 29]
gestational_age = [38, 39, 40, 37, 38, 41, 39, 36, 40, 38]
previous_c_section = [0, 1, 0, 0, 1, 0, 1, 0, 0, 1]
blood_pressure = [120, 110, 130, 115, 125, 118, 132, 113, 128, 122]
bmi = [22, 27, 24, 23, 26, 28, 30, 22, 25, 27]
c_section = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
```

- Dummy dataset: Features include information about the patient (e.g., `fetal_distress`, `maternal_age`), and the target variable (`c_section`) indicates whether a C-section is needed (1) or not (0).

---

4. Preparing and Standardizing the Data

python Copy code

```python
X = np.column_stack((fetal_distress, maternal_age, gestational_age, previous_c_section, bl
y = c_section

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

- Combines all feature arrays into a single 2D array (`X`).
- Normalizes the data with `StandardScaler`, transforming all features to have zero mean and unit variance. This improves the model's convergence and performance.

5. Training the Logistic Regression Model

python Copy code

```python
model = LogisticRegression(max_iter=500)
model.fit(X_scaled, y)
```

- Initializes a logistic regression model and trains it using the scaled dataset (`X_scaled` and `y`).

---

6. Flask Route: Home Page

python Copy code

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

- Defines the root route (`'/'`), which renders the `index.html` file. This serves as the landing page for the application.

7. Flask Route: Prediction Page

python Copy code

```python
@app.route('/predict', methods=['POST'])
def predict():
    ...
```

- Handles form submissions on the `/predict` route using the POST method.

Extracting Form Data

python Copy code

```python
fetal_distress_input = int(request.form['fetal_distress'])
maternal_age_input = int(request.form['maternal_age'])
gestational_age_input = int(request.form['gestational_age'])
previous_c_section_input = int(request.form['previous_c_section'])
blood_pressure_input = int(request.form['blood_pressure'])
bmi_input = float(request.form['bmi'])
```

- Reads user inputs submitted via the HTML form. These are cast to appropriate data types (e.g., `int`, `float`).

Preprocessing and Prediction

python Copy code

```python
new_patient_data = np.array([[fetal_distress_input, maternal_age_input, gestational_age_in
new_patient_data_scaled = scaler.transform(new_patient_data)
predicted_c_section = model.predict(new_patient_data_scaled)
```

- Converts user inputs into a NumPy array and scales the data using the previously fitted `StandardScaler`.
- Uses the logistic regression model to predict whether a C-section is needed.

Displaying Results

python Copy code

```python
if predicted_c_section[0] == 1:
    prediction = "C-section"
    instructions = "For a C-section procedure, please consult with your healthcare provide
else:
    prediction = "No C-section"
    instructions = "For a natural delivery: Focus on preparing for labor and delivery ..."
```

- Based on the prediction, displays instructions for either a C-section or natural delivery.

Error Handling

python Copy code

```python
except Exception as e:
    prediction = f"Error: {str(e)}"
    instructions = None
```

- Catches and displays any errors during the form submission or prediction process.

---

8. Flask Route: Render Result Page

python Copy code

```python
return render_template('result.html', prediction=prediction, instructions=instructions)
```

- Passes the prediction result and additional instructions to the `result.html` template for rendering.

9. Running the Application

python Copy code

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

- Starts the Flask application in debug mode, which reloads the server automatically when code changes.

Overview of Workflow

1. **User Interaction:**
   - User visits the homepage (`/`) and fills in the form (e.g., fetal distress, maternal age).
   - Submits the form to `/predict`.
2. **Processing and Prediction:**
   - Form data is preprocessed, scaled, and fed into the logistic regression model.
   - The model predicts whether a C-section is required.
3. **Result Display:**
   - The application renders the result page (`result.html`) with a prediction and relevant instructions.

This workflow demonstrates how Flask integrates machine learning models into a user-friendly web interface.