# ****Heart Disease Prediction System Documentation****

## ****1. System Overview****

The **Heart Disease Prediction System** is a web-based application that leverages **machine learning** to predict the likelihood of heart disease based on patient medical data. It integrates a **stacking classifier** combining **XGBoost** and **Random Forest** algorithms with **Logistic Regression** as the meta-learner.

## ****2. Technical Architecture****

### **2.1 Components**

* **Frontend**: HTML, CSS interface for data input and result display.
* **Backend**: Flask web server.
* **ML Model**: Stacking classifier (XGBoost + Random Forest + Logistic Regression).
* **Data Processing**: Scikit-learn preprocessing pipeline.

### **2.2 Dependencies**

Flask==2.0.1

numpy==1.21.0

pandas==1.3.0

scikit-learn==0.24.2

xgboost==1.4.2

joblib==1.0.1

## ****3. Model Specifications****

### **3.1 Input Features**

| **Feature** | **Type** | **Range/Values** | **Description** |
| --- | --- | --- | --- |
| **Age** | Integer | 28–77 years | Patient's age |
| **Sex** | Binary | 0 = Female, 1 = Male | Biological sex |
| **ChestPainType** | Categorical | 0: Typical Angina, 1: Atypical Angina,  2: Non-anginal Pain, 3: Asymptomatic | Type of chest pain |
| **RestingBP** | Integer | 0–200 mmHg | Resting blood pressure |
| **Cholesterol** | Integer | 0–603 mg/dL | Serum cholesterol |
| **FastingBS** | Binary | 0: ≤120 mg/dL, 1: >120 mg/dL | Fasting blood sugar |
| **RestingECG** | Categorical | 0: Normal, 1: ST-T Abnormality,  2: Left Ventricular Hypertrophy | Resting ECG results |
| **MaxHR** | Integer | 60–202 bpm | Maximum heart rate achieved |
| **ExerciseAngina** | Binary | 0: No, 1: Yes | Exercise-induced angina |
| **Oldpeak** | Float | -2.6 to 6.2 | ST depression during exercise |
| **ST\_Slope** | Categorical | 0: Upsloping, 1: Flat, 2: Downsloping | ST segment slope |

### **3.2 Model Output**

* **Primary Output**: Probability of heart disease (0–100%).
* **Binary Classification**:
  + 0: No heart disease.
  + 1: Heart disease.

### **3.3 Model Performance**

| **Metric** | **Value** |
| --- | --- |
| **Accuracy** | 97.61% |
| **Precision** | 0.98 |
| **Recall** | 0.98 |
| **F1-Score** | 0.98 |
| **Cross-Validation Std Dev** | 0.021 |

## ****4. API Documentation****

### **4.1 Endpoints**

#### **GET /**

* **Description**: Renders the home page with a prediction form.
* **Response**: HTML page.

#### **POST /predict**

* **Description**: Predicts the likelihood of heart disease based on input data.
* **Content-Type**: application/x-www-form-urlencoded.
* **Required Fields**: All input features listed in **Section 3.1**.
* **Response (JSON)**:

json

Copy code

{

"prediction": 0 or 1,

"probability": float (0-100),

"error": string (if applicable)

}

## ****5. Setup Instructions****

### **5.1 Environment Setup**

# Create virtual environment

python -m venv venv

source venv/bin/activate # Linux/Mac

venv\Scripts\activate # Windows

# Install dependencies

pip install -r requirements.txt

### **5.2 Model Deployment**

1. Place model.pkl and scaler.pkl in the root directory.
2. Set up the Flask environment:

export FLASK\_APP=app.py

export FLASK\_ENV=development

1. Run the application:

flask run

## ****6. Usage Guidelines****

### **6.1 Data Preprocessing**

* Ensure all numeric inputs are within specified ranges.
* Categorical inputs must match the defined values.
* Missing values are not supported.

### **6.2 Interpretation of Results**

* **Probability < 50%**: Low risk.
* **Probability ≥ 50%**: High risk.
* **Recommendation**: Consult a medical professional for probabilities >70%.

## ****7. Error Handling****

### **7.1 Common Errors**

* Invalid input ranges.
* Missing required fields.
* Type mismatches.
* Server connection issues.

### **7.2 Error Responses**

Example JSON response for errors:

json

Copy code

{

"error": "Error message",

"code": "ERROR\_CODE",

"details": "Additional error details"

}

## ****8. Security Considerations****

* Input validation for all fields.
* Rate limiting on API endpoints.
* Encryption for data in transit.
* No storage of personal health information.
* Regular security updates.

## ****9. Maintenance****

### **9.1 Model Retraining**

* **Schedule**: Quarterly.
* **Trigger**: Performance drops below 95% accuracy.
* **Validation**: Cross-validation with new data.

### **9.2 System Updates**

* Regular dependency updates.
* Performance monitoring.
* Error log analysis.
* Scheduled backups.

## ****10. Support****

For technical support or bug reports:

* **Email**: nour.hesham.ds@gmail.com
* **Documentation Updates**: Check version control.
* **Emergency Contact**: Refer to deployment guide.