Heart Stroke Prediction System

Introduction to Heart Stroke Prediction

The system predicts the risk of heart stroke using patient data.

Data is collected and preprocessed for analysis.

A machine learning model is trained and evaluated for accuracy.

Predictions are generated, providing valuable medical recommendations.

Machine Learning Models Used

1. **RandomForestRegressor:**
   An ensemble learning model that uses multiple decision trees to perform regression tasks, averaging their results for improved accuracy.

2. **LogisticRegression:**
   A statistical model used for binary classification tasks that predicts the probability of a binary outcome.

3. **KNeighborsClassifier:**
   A machine learning algorithm that classifies data points based on the majority vote of their k-nearest neighbors.

4. **GaussianNB:**
   A type of Naive Bayes classifier based on Gaussian probability distributions, used for classification tasks with continuous data.

Data Preprocessing

Handled missing values:

- 'education' filled with 'Unknown'
- 'Gender' filled with mode
- 'glucose' forward & backward fill

Encoded categorical features:

- 'Gender', 'Heart_stroke', 'prevalentStroke'
- Dropped 'education' column for model compatibility

Dataset Features

- Patient Demographics: Gender, age, education
- Health Indicators: Hypertension, heart disease, avg_glucose_level, BMI, smoking status
- Target Variable: - Heart_stroke (presence or absence)

Results

RandomForestRegressor:

- Evaluated using accuracy: 86.7%
- MSE:0.11
- R^2:-0.0017

LogisticRegression:

- Evaluated using accuracy:84.98%
- MSE:0.13
- R^2: -0.179

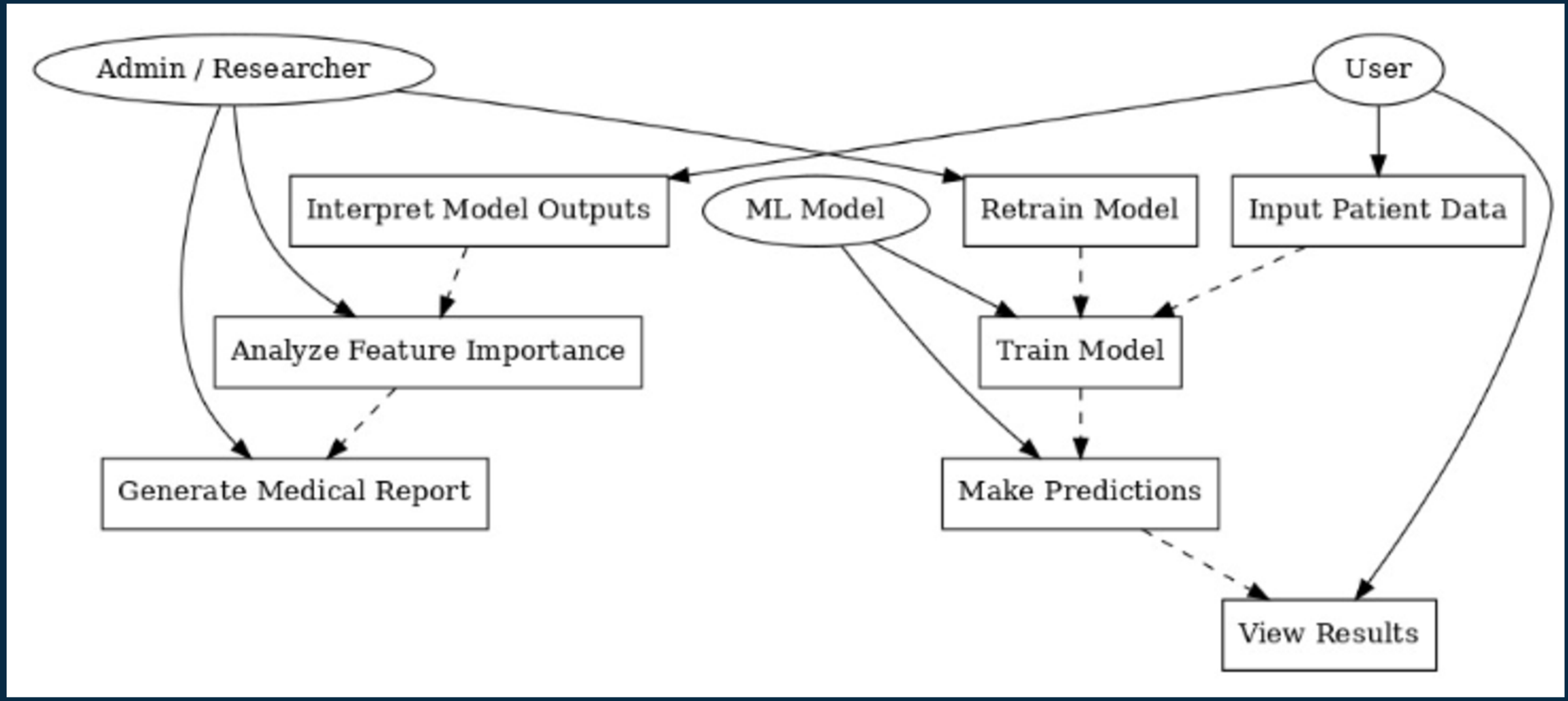
KNeighborsClassifier:

- Evaluated using accuracy:86.3%
- MSE:0.15
- R^2:-0.34

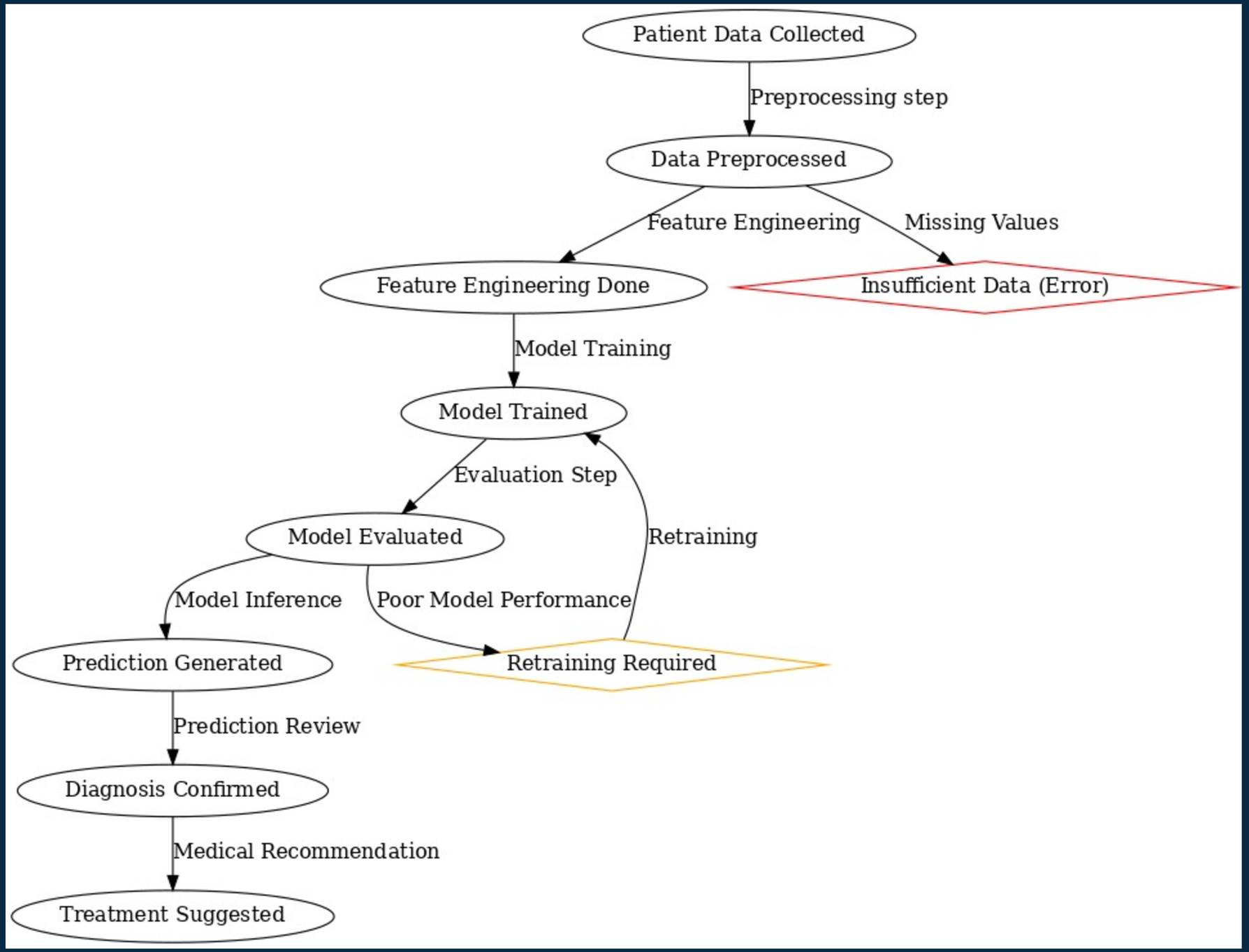
GaussianNB:

- Evaluated using accuracy: 82.5%
- MSE: 0.15
- R^2:-0.34

Use Case Diagram



Diagram


Patient Data Collected
Preprocessing step
Data Preprocessed
Feature Engineering
Missing Values
Feature Engineering Done
Insufficient Data (Error)
Model Training
Model Trained
Evaluation Step
Model Evaluated
Retraining
Model Inference
Poor Model Performance
Prediction Generated
Retraining Required
Prediction Review
Diagnosis Confirmed
Medical Recommendation
Treatment Suggested

Future Scope

- Creating app and website, where we take input from patients and give result.
- Saving the result in a structured database to train the model further. (Containerisation)

Timeline

February (Remaining Days)

✅ Model Completion (Done)

🔹 Test and fine-tune the model for best performance.

🔹 Save/export the model for integration (e.g., TensorFlow SavedModel, ONNX, or TensorFlow Lite).

March (App & Web Development)

🗓️ Week 1 (March 1–7)

🔹 Plan the UI/UX for the web and mobile app.

🔹 Choose tech stack (React/Flutter for frontend, Flask/FastAPI/Django for backend).

🔹 Set up backend API to serve model predictions.

🗓️ Week 2 (March 8–14)

🔹 Implement core functionalities (image upload, model inference, results display).

🔹 Develop frontend UI and integrate with the backend.

Timeline

🗓️ Week 3 (March 15–21)

- 🔹 Optimize the app (performance tuning, UI improvements).
- 🔹 Add additional features (e.g., user authentication, data storage).

🗓️ Week 4 (March 22–31)

- 🔹 Fully test web and mobile applications.
- 🔹 Fix bugs and prepare for deployment.

April (Security & Deployment)

🗓️ Week 1 (April 1–7)

- 🔹 Implement firewall for security (e.g., WAF, network security rules).
- 🔹 Begin containerization (Dockerize the application)

Timeline

🗓️ Week 2 (April 8–14)

- Deploy backend and model API using Docker & Kubernetes.
- Perform security testing (firewall validation, penetration testing).

🗓️ Week 3 (April 15–21)

- Final testing of web and mobile apps in real-world conditions.
- Ensure scalability and reliability (load testing, stress testing).

🗓️ Week 4 (April 22–30)

- Final deployment and launch.
- Documentation and user manual creation.
- Backup strategy and maintenance plan

THANK YOU