Github Link: https://github.com/suji1207/transforming-healthcare-with-AI-powered-disease-prediction-based-on-patient-data.git

Transforming Healthcare with AI-powered disease prediction based on patient data

Student Name: S.SUJI
Register Number:511023205004
Institution: Jei Mathaajee College of Engineering(5110)
Department: B.Tech(IT)
Date of Submission: 02-05-2025

1. Problem Statement :

The healthcare industry faces persistent challenges in early and accurate disease detection due to overburdened systems, limited access to skilled professionals, and the complexity of interpreting large volumes of patient data. Traditional diagnostic methods are often reactive rather than preventive, leading to delayed treatment and increased healthcare costs. With the rapid growth of electronic health records (EHRs), wearable devices, and other patient data sources, there is a pressing need for intelligent systems that can analyze this data efficiently.

Despite the availability of large datasets, existing predictive models often suffer from low accuracy, limited scalability, or poor generalization across populations due to bias, data imbalance, or insufficient integration of diverse data types (e.g., clinical, demographic, behavioral). There is a gap in effectively leveraging artificial intelligence (AI) to build robust, interpretable, and personalized disease prediction models that support timely clinical decision-making.

2. Project Objectives :

1. **Develop a Predictive AI Model:** Build machine learning or deep learning models capable of analyzing patient data to predict the risk or onset of specific diseases with high accuracy and reliability.

2. **Integrate Multi-source Patient Data :** Collect and preprocess diverse data sources such as electronic health records (EHRs), lab test results, medical imaging, genetic data, and real-time health metrics from wearables to create a comprehensive patient profile.

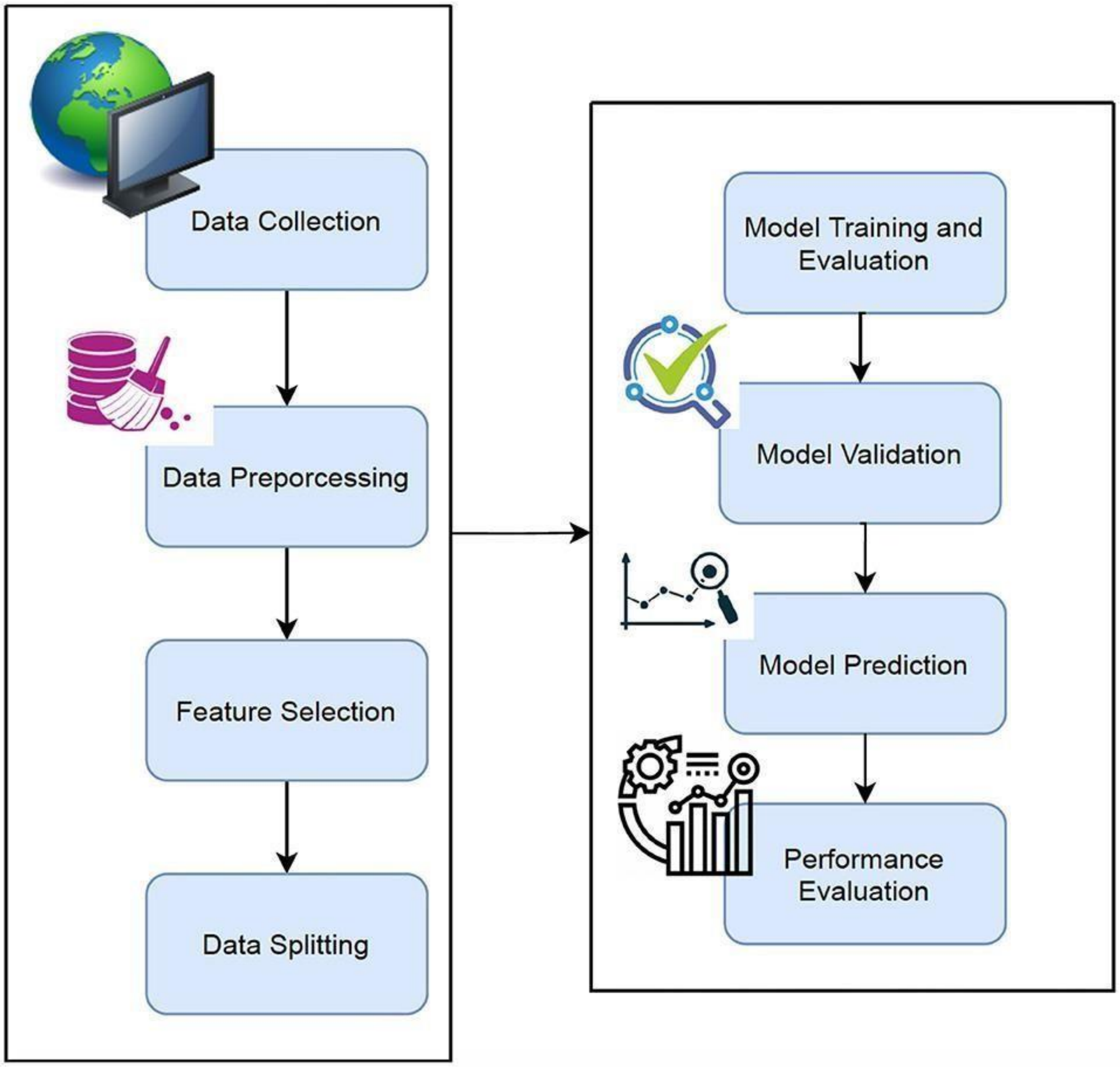
3. **Ensure Model Interpretability:** Incorporate explainable AI (XAI) techniques to ensure transparency in predictions, allowing healthcare providers

4. **Ensure Model Interpretability:** Incorporate explainable AI (XAI) techniques to ensure transparency in predictions, allowing healthcare providers to understand the rationale behind each prediction and support informed clinical decisions.

5. **Enable Early Detection and Prevention:** Design the system to identify early warning signs of chronic and acute conditions (e.g., diabetes, heart disease, cancer) to enable timely intervention and treatment.

6. **Optimize for Scalability and Real-time Use:** Develop a scalable architecture that can be integrated into hospital IT systems and handle real-time data processing for use in clinical settings.

3. Flowchart of the Project Workflow :



4. Data Description :

1. Electronic Health Records (EHRs)

- **Contents**: Patient history, diagnoses, procedures, medications, allergies, vital signs, and hospitalization records.
- **Format**: Structured (tables, ICD codes) and unstructured (clinical notes).
  Use: Basis for understanding patient history and prior clinical events.

2. Laboratory Test Results

- **Contents**: Blood tests, urine tests, liver/kidney function panels, glucose levels, cholesterol, etc.
- **Format**: Numerical/quantitative data with time-stamps.
- **Use**: Key indicators of organ function, disease onset, and progression.

3. Demographic Information

- **Contents**: Age, gender, ethnicity, address, occupation, education level.
- **Format**: Categorical and numerical.
- **Use**: Risk stratification and detecting disease prevalence trends across groups.

4. Medical Imaging Data (optional, for advanced models)

- **Contents**: X-rays, CT scans, MRIs, ultrasound.
- **Format**: DICOM images or pixel arrays.
- **Use**: Image-based diagnostics for diseases like pneumonia, cancer, or stroke.

5. Wearable and Real-Time Sensor Data

- **Contents**: Heart rate, sleep patterns, activity levels, glucose monitors.
- **Format**: Time-series data.
- **Use**: Continuous health monitoring and real-time alerts.

6. Lifestyle and Behavioral Data

- **Contents**: Smoking status, alcohol consumption, exercise frequency, diet patterns.

- **Format**: Self-reported (structured or unstructured).
- **Use**: Personal health risk assessment and behavioral influence on disease.

7. Genomic or Genetic Data (if available)

- **Contents**: SNPs, gene expression profiles, genetic markers.
- **Format**: High-dimensional tabular data.
- **Use**: Personalized risk prediction and pharmacogenomics.

5. Data Preprocessing :

1. Data Cleaning

- **Handle Missing Values**:
  - Techniques: Imputation (mean, median, mode), forward/backward fill, or modelbased imputation. ○ Example: Missing blood pressure values may be filled using a patient's historical data.
- **Remove Duplicates**: ○ Identify and remove repeated entries (especially in EHRs or logs).
- **Outlier Detection**:
  - Use statistical methods (IQR, Z-score) to detect abnormal values (e.g., blood sugar = 1000 mg/dL). ○ Handle with capping, removal, or transformation.

2. Data Integration

- **Merge Data from Multiple Sources**:
  - Combine EHRs, lab reports, and wearable data by aligning them on patient IDs and timestamps.
- **Resolve Inconsistencies**:
  - Standardize medical terminology using ontologies like SNOMED CT or ICD-10 codes.

3. Data Transformation

- **Normalization/Standardization**:
  - Apply Min-Max scaling or Z-score normalization to numerical features to bring them to a common scale.

- **Encoding Categorical Data**:
    - Convert non-numeric data (e.g., gender, smoking status) using one-hot encoding, label encoding, or embedding.
- **Time-Series Alignment**:
    - For sensor or wearable data, align with medical events using timestamps.

4. *Data Balancing (if classification is imbalanced)*

- **Techniques**:
    - Oversampling (SMOTE) ○ Undersampling
    - Class weight adjustment during training
- **Use Case**: Useful when predicting rare diseases where positive cases are limited.

5. *Feature Engineering* **Derived Features**:

- Create new features (e.g., BMI from height and weight, risk score from multiple inputs).
- **Temporal Features**: ○ Add time-related features (e.g., time since last hospital visit).
- **Aggregation**:
    - Summarize data (e.g., average glucose level over 30 days).

6. Exploratory Data Analysis (EDA) :

1. *Data Overview*

- **Check shape**: Number of rows and columns.
- **Types of features**: Categorical, numerical, datetime.
- **Missing values**: Percentage and location.

2. *Univariate Analysis* **Numerical Variables**:

- Histograms, box plots, and summary statistics (mean, median, std dev).
- Example: Age distribution, blood pressure ranges.

- **Categorical Variables**:
    - Bar plots and frequency tables. ○ Example: Distribution of gender, disease outcome counts.

3. *Bivariate Analysis*

- **Correlation Matrix (Heatmap)**:
  - Identify relationships between numerical features.
  - Example: Correlation between BMI and glucose levels.
- **Cross-tabulation / Grouped Analysis**:
  - Compare features by disease presence/absence.
  - Example: Mean cholesterol by heart disease status.
- **Boxplots/Violin plots**:
  - Visualize distributions across disease outcomes.

4. *Target Variable Distribution*

- Analyze class balance for disease prediction:
  - Example: % of patients with vs. without diabetes. ○ Useful to assess if class imbalance handling is needed.

5. *Outlier Detection* □ Use boxplots or Z-score/IQR method.

- Example: Extremely high liver enzyme values indicating possible data error or rare condition.

6. *Time-Series Trends (if applicable)*

- Plot variables like heart rate, glucose, or sleep patterns over time.
- Identify anomalies, patterns, or seasonal trends.

7. **Feature Engineering :**

1. Domain-Specific Derived Features

- BMI (Body Mass Index) = weight (kg) / height^2 (m^2)
- Pulse Pressure = systolic BP – diastolic BP
- Age groups = categorize age into ranges (e.g., 0–18, 19–40, 41–60, 60+) □ Comorbidity Score = count of existing chronic conditions

2. Temporal Feature Extraction

- Time Since Last Visit/Test: Indicates healthcare engagement frequency.
- Trend Features: Slope of glucose level or heart rate over time.
- Rolling Averages: Moving averages over 7 or 30 days for vitals or labs.

3. Statistical Features (especially for time-series or wearable data)

- Mean, Median, Standard Deviation of heart rate, blood glucose, etc.
- Min/Max values over fixed time windows
- Number of spikes or abnormal readings in a period

4. Categorical Feature Encoding

- Label Encoding: For ordinal categories (e.g., pain severity: low, medium, high)
- One-Hot Encoding: For non-ordinal data like smoking status or gender ▯ Frequency Encoding: For high-cardinality features (e.g., zip code)

5. Aggregation Features

- Count of doctor visits per year
- Number of prescriptions in last 6 months
- Average lab value by test type per patient

6. Risk Scores (Clinical Rules-Based Features)

- Derived using medical guidelines: ○ Framingham Risk Score ○ Charlson Comorbidity Index ○ Diabetes risk factors: age, BMI, family history, etc.

7. Text Feature Extraction (if using clinical notes)

- TF-IDF, Bag of Words, or word embeddings from unstructured doctor notes.

- NER (Named Entity Recognition): extract medications, symptoms, or diagnoses.

8. Feature Selection & Reduction

- Techniques:
  - Recursive Feature Elimination (RFE) ○ Feature importance from tree-based models
  - Principal Component Analysis (PCA) for dimensionality reduction

8. Model Building :

1. Define the Prediction Task

- **Type**: Classification (e.g., predict presence of heart disease, diabetes) or Regression (e.g., predict hospital stay length).
- **Target Variable**: Disease status (binary: 0 = No, 1 = Yes) or risk score.

2. Select Algorithms

Based on the dataset and task, consider:

- **Traditional ML Models**:
  - Logistic Regression (baseline) ○ Decision Trees ○ Random Forest ○ Gradient Boosting (XGBoost, LightGBM) ○ Support Vector Machines (SVM) ○ K-Nearest Neighbors (KNN)
- **Deep Learning Models** (for large or complex data): ○ **MLP (Feedforward Neural Networks)** for structured/tabular data.
  - **CNNs** for image-based diagnosis (e.g., X-rays). ○ **LSTMs/RNNs** for time-series patient data (e.g., wearable sensors).

    - **Transformer-based models** for clinical text (e.g., BERT, BioBERT).

3. Train-Test Split

- Split data into:
    - **Training Set**: 70–80% ○ **Validation Set**: 10–15% ○ **Test Set**: 10–15%
- Use stratified sampling if classes are imbalanced.

4. Model Training

- Fit the model to the training data.
- Use **cross-validation (e.g., 5-fold)** to assess stability and prevent overfitting.
- Optimize hyperparameters (e.g., learning rate, depth) using:
    - Grid Search ○ Random Search ○ Bayesian Optimization

5. Model Evaluation

Evaluate performance using:

- **Accuracy**
- **Precision, Recall, F1-Score**

- **AUC-ROC Curve** **Confusion Matrix**
- **PR (Precision-Recall) Curve** (for imbalanced data)

6. *Handle Class Imbalance (if needed)*

- **SMOTE** (Synthetic Minority Oversampling)
- **Class weights** during training
- **Resampling methods** (undersampling or oversampling)

7. *Model Explainability*

- Apply **Explainable AI (XAI)** tools to interpret predictions: ○ SHAP (SHapley Additive exPlanations)
    - LIME (Local Interpretable Model-Agnostic Explanations) ○ Feature Importance from tree-based models

8. *Model Comparison & Selection*

- Train and compare multiple models on validation set.
- Select the model with the best trade-off between accuracy and interpretability.

9. Visualization of Results & Model Insights :

1. *Performance Metrics Visualizations*

- **Confusion Matrix**:
    - Visualize the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).
    - Helps assess the accuracy, precision, recall, and F1-score of the model.

○

python CopyEdit

Example: A confusion matrix for predicting whether a patient has diabetes or not.

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["No Diabetes", "Diabetes"]) disp.plot()

- **ROC Curve (Receiver Operating Characteristic)**:
  - Plots the True Positive Rate (Recall) against the False Positive Rate.
  - **AUC (Area Under the Curve)** measures the overall ability of the model to distinguish between classes.

python
CopyEdit
from sklearn.metrics import roc_curve, roc_auc_score
fpr, tpr, _ = roc_curve(y_true, y_prob) auc =
roc_auc_score(y_true, y_prob) plt.plot(fpr, tpr,
label=f'AUC = {auc:.2f}') **Precision-Recall Curve**:

  - Useful for imbalanced datasets, where precision and recall are more informative than accuracy. ○ Shows the trade-off between precision and recall at different thresholds.

python
CopyEdit
from sklearn.metrics import precision_recall_curve
precision, recall, _ = precision_recall_curve(y_true, y_prob)
plt.plot(recall, precision, label='Precision-Recall curve')

2. Feature Importance Visualization

- **Feature Importance (for Tree-based models)**:
  - Visualize which features are most important in the model's decisionmaking.

    Example: A bar plot showing the importance of features like age, BMI, or cholesterol in predicting heart disease.

import matplotlib.pyplot as plt

○

python CopyEdit

feature_importances =
model.feature_importances_ features =
data.columns plt.barh(features,
feature_importances) plt.xlabel('Feature
Importance')
plt.title('Feature Importance of Different Variables')

 SHAP (SHapley Additive exPlanations):

- Provides a unified measure of feature importance and shows how each feature contributes to individual predictions. ○ **Example**: Visualize how variables like glucose level, BMI, or age affect the predicted risk of diabetes.

python CopyEdit import shap explainer =
shap.TreeExplainer(model) shap_values =
explainer.shap_values(X_train)
shap.summary_plot(shap_values, X_train)

3. Model Predictions Visualizations

- **Prediction Distribution**:
    - Visualize the distribution of predicted values (probabilities) for the disease, e.g., a histogram of predicted probabilities for heart disease presence.

python
CopyEdit
plt.hist(y_prob, bins=50) plt.title('Prediction
Distribution') plt.xlabel('Predicted
Probability of Disease')
plt.ylabel('Frequency')

- **Individual Prediction Explanations (e.g., Local Interpretability)**: Use LIME or SHAP to show why a particular patient was predicted to have a certain disease or not, focusing on individual predictions.

○

python CopyEdit

```
import lime
explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values,
mode="classification", training_labels=y_train)
explanation = explainer.explain_instance(X_test.iloc[0].values,
model.predict_proba)
explanation.show_in_notebook()
```

4. Model Calibration Visualization ▯ **Calibration Curve**:

- Shows how well the predicted probabilities match the actual outcomes. A good model should have a calibration curve close to the diagonal.

python
CopyEdit

```
from sklearn.calibration import calibration_curve
fraction_of_positives, mean_predicted_value = calibration_curve(y_true,
y_prob, n_bins=10)
plt.plot(mean_predicted_value, fraction_of_positives, "s-", label="Model
Calibration")
```

5. Class Distribution Visualization

- **Class Distribution**:
  - Visualize the distribution of target classes (e.g., number of patients with and without disease) to understand data balance.

python
CopyEdit

```
plt.bar(["No Disease", "Disease"], [np.sum(y_train == 0), np.sum(y_train
== 1)]) plt.title("Class
Distribution")
plt.ylabel("Number of Patients")
```

6. Time-Series Predictions (if applicable)

- **Prediction Over Time**:
    - For time-series data (e.g., wearable device data), plot the predicted vs. actual values of health metrics like heart rate, blood glucose, etc.

```
python
CopyEdit
plt.plot(time_stamps, actual_values, label="Actual")
plt.plot(time_stamps, predicted_values, label="Predicted")
plt.xlabel("Time") plt.ylabel("Glucose Levels")
plt.legend()
```

10. Tools and Technologies Used :

- **ProgrammingLanguage**:Python3
- **Notebook Environment**:
GoogleColab
- **Key Libraries**:
    - pandas, numpy for data handling
    - matplotlib, seaborn, plotly for visualizations
    - scikit-learn for preprocessing and modeling
    - Gradio for interface deployment

11. Team Members and Contributions :

- *Data cleaning* –**K. KOMALA**
- *EDA, Feature engineering*- **J. KUTTI**
- *Model development, Documentation and reporting*- D. ANUSUYA