Transforming healthcare with AI-powered disease prediction based on patient data.

Student Name: **J.Jabanesh**

Register Number: **412323205015**

Institution: **Sri Ramanujar Engineering College**

Department: **B.Tech / IT**

Date of Submission: **10/05/2025**

Github Repository Link: **https://github.com/karuppan024/S.tamilselvan.git**
your Github Repository]

1.project statement

1.Late Diagnosis: Diseases are often detected too late, limiting treatment effectiveness and increaseing cost.

2.Data Underutilization:patient data is not fully levaraged for early disease prediction, missing crucial insights.

3.Resource misallocation: Healthcare resources are are not optimized due to inability to predict at-risk patients early.

4.Generic Care:lack of personalized predictions leads to one-size-fits- all treatment plans, reducing treatment efficacy.

2.project objectives

1. Develop an AI-based model capable of accurately predicting the onset of specific diseases using historical and real-time patient data.

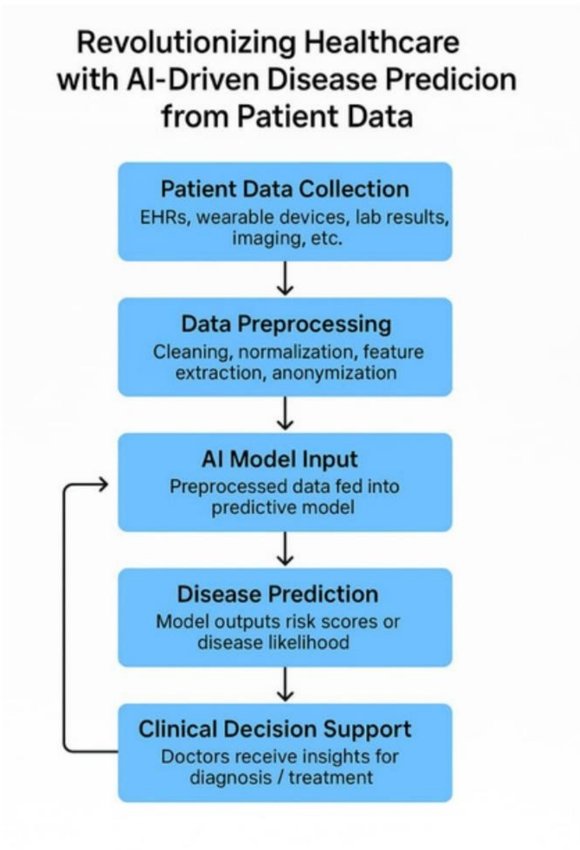
2. Integrate diverse data sources, including electronic health records (EHR), lab results, medical imaging, and wearable device data, to enhance prediction accuracy.

3. Ensure patient data privacy and security by implementing strong encryption and compliance with healthcare regulations (e.g., HIPAA, GDPR).

4. Improve early diagnosis rates for targeted diseases, reducing the time from symptom onset to clinical intervention.

5. Deploy a user-friendly interface for healthcare providers to interpret AI predictions and integrate them into clinical workflows.

3.Flowchart of the project work flow



4.Data description

1. Patient Demographics

Attributes: Age, gender, ethnicity, weight, height, BMI, smoking status

Purpose: Establish baseline risk factors and personalize predictions

2. Medical History

Attributes: Chronic conditions (e.g., diabetes, hypertension), previous surgeries, allergies, family history of diseases

Purpose: Identify long-term risk patterns and hereditary influences

3. Clinical Data

Attributes: Lab test results (blood tests, cholesterol levels), imaging results (X-rays, MRIs), vital signs (blood pressure, heart rate, temperature)

Purpose: Provide measurable indicators of current health status

4. Medications

Attributes: Current prescriptions, dosage, frequency, past medications

Purpose: Analyze potential side effects or interactions affecting disease risk

5. Lifestyle Data

Attributes: Physical activity, diet, alcohol consumption, sleep patterns

Source: Self-reported surveys, fitness trackersSure! Here's a single key point:

5.Data preprocessing

1. Data Collection:

Sources: EHRs (Electronic Health Records), lab results, wearable devices, imaging data, etc.

2. Data CleanHandling Missing :

Values:Drop rows/columns with excessive missingness

Impute with mean/median/mode or use model-based imputation

Removing Duplicates

Outlier Detection:

Use Z-score, IQR, or domain-specific rules (e.g., BP > 300 is invalid)

3. Data Transformation:

Encoding Categorical Variables:

One-hot encoding (for nominal data)

Label encoding (for ordinal data)

Feature Scaling:

Standardization (Z-score)

Normalization (Min-Max scaling)

Date and Time Features:

Extract age, day of week, time since diagnosis, etc

4. Feature Engineering:

Combine Features: E.g., BMI = weight / $height^2$

Domain-Specific Ratios: Lab value ratios, risk scores

Interaction Features: Age × chronic condition statement

5. Dimensionality Reduction (if needed):

PCA, t-SNE (for visualization), or domain-driven feature selection

6. Exploratory Data Analysis

1. Understand the Dataset :

Load the data: Use tools like pandas to read CSV, Excel, or database files.
Preview: Use .head(), .info(), .describe() to inspect structure and basic statistics.
Data types: Check for categorical, numerical, boolean, and datetime features.

```
import pandas as pd
df = pd.read_csv('patient_data.csv')
print(df.head())
print(df.info())
print(df.describe())
```

2. Clean the Data :

Missing values: Use df.isnull().sum() to detect, and impute or drop as needed.
Duplicates: Use df.duplicated().sum() and df.drop_duplicates() if necessary.
Outliers: Use box plots or IQR method.

3. Univariate Analysis :

Numerical features: Histograms, box plots.
Categorical features: Bar plots, value counts.

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.histplot(df['age'], kde=True)
sns.countplot(x='gender', data=df)
```

4. Bivariate/Multivariate Analysis :

Correlations: Heatmap for numeric features.
Target relationship: Analyze how each feature relates to disease presence or severity.

```
# Correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
# Boxplot of age by disease outcome
sns.boxplot(x='disease', y='age', data=df)
```

5.Summary Insights

Highlight patterns, such as risk factors.

Identify which features are most associated with disease.

7.Feature Engineering

1. Handling Categorical Variables :

Convert non-numeric features into numbers so machine learning models can use them.

One-Hot Encoding: For unordered categories (e.g., gender).

Creates binary columns like gender_male = 1/0.

Label Encoding: For ordered categories (e.g., disease stage).

Assigns numbers based on rank (e.g., mild = 0, severe = 2)

2. Creating New Features :

Derive meaningful new variables from existing data to improve prediction.

BMI:

BMI = weight_kg / (height_m)^2 — useful for identifying obesity-related risks.

Age Groups:

Group ages into categories like young, middle-aged, senior.

Risk Flags:

Create binary indicators like:

high_bp = 1 if systolic BP > 130 or diastolic BP > 80

high_glucose = 1 if glucose > 126

8.Model Building

1. Data Preprocessing

Clean Data: Handle missing values, remove outliers, and encode categorical variables (e.g., One-Hot or Label Encoding).
Feature Scaling: Normalize or standardize numerical features if using algorithms like SVM or KNN.
Split Data: Divide into training and test sets (e.g., 80/20 split).

2. Model Selection

Algorithm Choice: Start with simpler models like Logistic Regression or Decision Trees, then move to more complex ones like Random Forest or XGBoost.
Cross-validation: Use k-fold cross-validation to evaluate model stability.
Hyperparameter Tuning: Tune models using grid search or random search.

3. Model Evaluation

Metrics: Use appropriate evaluation metrics like accuracy, precision, recall, F1-score, ROC-AUC (especially for imbalanced datasets).
Confusion Matrix: Helps visualize true positives, false positives, true negatives, and false negatives.

9.visualization Of Results & model insight

1. Confusion Matrix

Shows how well the model classifies:

True Positives (TP), False Positives (FP), etc.

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_estimator(model, X_test, y_test)
```

2. ROC Curve & AUC Score

Visualizes the trade-off between true positive rate and false positive rate.

```
from sklearn.metrics import roc_curve, auc
fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:,1])
plt.plot(fpr, tpr, label=f"AUC = {auc(fpr, tpr):.2f}"
```

3. Feature Importance

Shows which features contributed most to predictions (for tree-based models).

```
import seaborn as sns
importances = model.feature_importances_
sns.barplot(x=importances, y=feature_names)
```

10.Tools and technologies used

1. Programming Language

Python – Widely used for data science and machine learning.

2. Data Handling & Analysis

Pandas – For data manipulation and analysis.

NumPy – For numerical operations.

3. Visualization

Matplotlib & Seaborn – For data and result visualization.

Plotly – For interactive visualizations.

4. Machine Learning

Scikit-learn – For model building, evaluation, and preprocessing.

XGBoost / LightGBM – For high-performance gradient boosting models.

5. Model Tuning & Evaluation

GridSearchCV / RandomizedSearchCV – For hyperparameter tuning.

SHAP / LIME – For model explainability and insights.

6. Environment & Tools

Jupyter Notebook / Google Colab – For interactive coding and experiments.

Anaconda – For managing packages and environments.

11.Team members and contribution

S.Tamilselvan - (Data cleaning, EDA)

J.Jabanesh - (Feature engineering, Model development)

L.Vijay - Documention and reporting