**Phase-2 Submission Template**

**Student Name:** V VARUN

**Register Number:** 732323104051

**Institution:** SSM COLLEGE OF ENGINEERING

**Department:** COMPUTER SCIENCE AND ENGINEERING

**Date` of Submission:** 24-04-2025

**Github Repository Link:** [https://github.com/Harish19606/Group12/blob/main/Harish.py]

### **1.Problem Statement**

Transforming healthcare with Al-powered disease prediction based on patient data give the model building

### **2. Project Objectives**

To develop and implement an AI-powered system that analyzes patient data to accurately predict the onset and progression of diseases. This system aims to support early diagnosis, enable proactive treatment strategies, and enhance overall healthcare outcomes by leveraging machine learning algorithms and real-time data analysis.

### **3. Flowchart of the Project Workflo****w**

### Start

### ↓

### 1. Data Collection

### - Patient demographics

### - Medical history

### - Lab reports

### - Imaging data (optional)

### - Wearable device data (if available)

### ↓

### 2. Data Preprocessing

### - Data cleaning (missing values, outliers)

### - Data normalization/scaling

### - Feature extraction/selection

### - Encoding categorical variables

### ↓

### 3. Exploratory Data Analysis (EDA)

### - Statistical summaries

### - Correlation analysis

### - Visualization of data distributions and trends

### ↓

### 4. Model Selection

### Choose suitable algorithms (e.g., Random Forest, XGBoost, Neural Networks)

### - Define performance metrics (Accuracy, AUC, Precision, Recall, F1)

### ↓

### 5. Model Training

### - Train model(s) using training data

### - Perform cross-validation

### - Tune hyperparameter

### ↓

### 6. Model Evaluation

### - Test on validation/test dataset

### - Analyze confusion matrix, ROC curve, etc.

### - Compare model performance

### ↓

### 7. Model Interpretation

### - Use SHAP, LIME, or feature importance plots

### - Validate predictions with domain experts

### ↓

### 8. Deployment

### - Integrate into clinical decision support system (CDSS)

### - Web/mobile interface for healthcare providers.

### **↓**

### 9. Monitoring & Maintenance

### - Continuous performance tracking

- Retraining with new data

### 4. Data Description

1.Patient Demographics:

| * ***Feature Name*** | * ***Description*** | * ***Data Type*** | * ***Example Value*** |
| --- | --- | --- | --- |
| * *Patient\_ID* | * *Unique identifier for each patient* | * *Categorical* | * *P12345* |
| * *Age* | * *Age of the patient* | * *Numerical* | * *52* |
| * *Gender* | * *Biological sex of the patient* | * *Categorical* | * *Male/Female/Other* |
| * *Ethnicity* | * *Ethnic background* | * *Categorical* | * *Asian, Hispanic, etc.* |

* *Static or dynamic dataset.*
* *Target variable (if supervised learning).]*

*2.Lifestyle and Social History:*

| ***Feature Name*** | ***Description*** | ***Data Type*** | ***Example Value*** |
| --- | --- | --- | --- |
| *Smoking\_Status* | *Current or past smoker* | *Categorical* | *Never, Former, Current* |
| *Alcohol\_Consumption* | *Frequency of alcohol intake* | *Categorical* | *None, Moderate* |
| *Physical\_Activity* | *Hours of physical activity per week* | *Numerical* | *3.5* |
| *Diet\_Score* | *Nutrition-based scoring (e.g., from surveys)* | *Numerical* | *7.2* |
| *Feature Name* | *Description* | *Data Type* | *Example Value* |
| *Past\_Conditions* | *History of diseases (e.g., diabetes, hypertension)* | *Categorical* | *Diabetes, Hypertension* |
|  |  |  |  |
|  |  |  |  |

### **5. Data Preprocessing**

*Preprocessing is critical in ensuring high-quality, reliable input for AI models. Here's a detailed step-by-step guide for preprocessing patient data:*

***1. Data Collection***

* *Source data from Electronic Health Records (EHRs), lab test results, wearable devices, imaging reports, genomics, and patient history.*
* *Ensure diverse, representative data across age, gender, ethnicity, and medical conditions.*

***2. Data Cleaning***

* ***Handle missing values:*** *Impute (mean/median/mode), remove, or flag incomplete entries.*
* ***Remove duplicates:*** *Ensure each patient record is unique.*
* ***Correct errors:*** *Fix inconsistencies in dates, units, or misentered values (e.g., "Mlae" → "Male").*

***3. Data Normalization/Standardization***

* *Normalize continuous variables (e.g., glucose level, blood pressure) using Min-Max scaling or Z-score normalization.*
* *Standardize units (e.g., mg/dL vs. mmol/L).*

***4. Categorical Encoding***

* *Convert categorical variables (e.g., gender, diagnosis codes) into numerical values using:*
  + ***One-hot encoding*** *for non-ordinal categories.*
  + ***Label encoding*** *for ordinal variables.*

***5. Outlier Detection and Treatment***

* *Use statistical methods (IQR, Z-score) or visualization (box plots) to detect outliers.*
* *Treat based on domain knowledge—either remove or cap/floor them.*

***6. Feature Engineering***

* *Derive new features like* ***BMI****,* ***disease risk scores****,* ***time since last diagnosis****, etc.*
* *Aggregate historical patient visits or lab results for trend analysis.*

***7. Dimensionality Reduction (if needed)***

* *Use PCA, t-SNE, or feature selection techniques to reduce dimensionality without losing important information.*

***8. Data Splitting***

* *Divide data into* ***training****,* ***validation****, and* ***test*** *sets (e.g., 70/15/15).*
* *Use stratified sampling if working with imbalanced disease classes.*

***9. Handling Imbalanced Data***

* *If disease cases are rare, use techniques like:*
  + ***SMOTE (Synthetic Minority Oversampling Technique)***
  + ***Class weighting*** *during model training*
  + ***Undersampling*** *majority class*

***10. Ensure Privacy & Security***

* *De-identify or anonymize data.*
* *Apply data encryption and comply with regulations (e.g., HIPAA, GDPR).*

### 

### **6. Exploratory Data Analysis (EDA)**

**1. Understand the Dataset**

* **Common Features in Healthcare Data**:
  + Patient ID
  + Age, Gender
  + Vitals: Blood pressure, Heart rate, BMI
  + Lab Results: Glucose, Cholesterol, Hemoglobin, etc.
  + Lifestyle: Smoking, Alcohol, Exercise
  + History: Diagnosed conditions, Medications
  + Outcome variable: Presence of disease (e.g., Diabetes, Heart Disease)

📌 **Goal**: Predict disease presence (classification problem) using patient attributes.

**2. Data Overview**

python

df.shape

df.info()

df.describe()

* Check for:
  + Missing values
  + Data types (numeric, categorical, date)
  + Unique values per column

**3. Missing Values & Data Quality**

python

df.isnull().sum().sort\_values(ascending=False)

* Visualize with heatmap or bar plot
* Strategies:
  + Impute with mean/median/mode
  + Use domain knowledge
  + Drop if not critical

**4. Univariate Analysis**

**🧬 Numerical Features**

python

df['age'].hist()

df['bmi'].plot(kind='box')

* Look for outliers and skewed distributions

**🧬 Categorical Features**

python

df['gender'].value\_counts().plot(kind='bar')

* Distribution of categories (e.g., gender, smoker)

**5. Bivariate Analysis**

**📈 Numeric vs Outcome**

python

sns.boxplot(x='disease', y='age', data=df)

* See how features differ by disease status

**📊 Categorical vs Outcome**

python

pd.crosstab(df['smoker'], df['disease']).plot(kind='bar', stacked=True)

* Association between lifestyle habits and disease

**6. Correlation Analysis**

python

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

* Detect multicollinearity
* Identify features most correlated with disease

**7. Dimensionality Reduction (Optional)**

* PCA or t-SNE for visual clustering
* Useful for high-dimensional datasets

**8. Target Distribution**

python

df['disease'].value\_counts(normalize=True).plot(kind='bar')

* Check for class imbalance
* May need resampling techniques (SMOTE, etc.)

**9. Initial Insights**

* Summarize:
  + Risk factors correlated with disease
  + Feature distributions and patterns
  + Data quality issues to address before modeling

### **7. Feature Engineering**

***1. Handling Missing Values***

* ***Numerical columns*** *→ Impute with* ***median*** *or* ***mean***
* ***Categorical columns*** *→ Impute with* ***mode*** *or create* ***"Unknown"*** *category*

*python*

*df['bmi'].fillna(df['bmi'].median(), inplace=True)*

*df['smoking\_status'].fillna('Unknown', inplace=True)*

***2. Encoding Categorical Variables***

* *Label encode binary categories (e.g., gender)*
* *One-hot encode multi-category features (e.g., smoking status)*

*python*

*df['gender'] = df['gender'].map({'Male': 0, 'Female': 1})*

*df = pd.get\_dummies(df, columns=['smoking\_status'], drop\_first=True)*

***3. Create New Features***

*Here’s where AI can shine through* ***domain-informed features****:*

| ***New Feature*** | ***Logic*** | ***Reason*** |
| --- | --- | --- |
| *age\_group* | *Binned age into decades* | *Risk varies by age* |
| *bmi\_category* | *Underweight, Normal, Overweight, Obese* | *Links to heart/diabetes* |
| *risk\_score* | *Composite of glucose, cholesterol, BMI* | *Early risk assessment* |
| *has\_family\_history* | *From patient history* | *Strong predictor* |

*python*

*df['age\_group'] = pd.cut(df['age'], bins=[0, 30, 45, 60, 75, 100], labels=['<30', '30-45', '45-60', '60-75', '75+'])*

*df['bmi\_category'] = pd.cut(df['bmi'], bins=[0, 18.5, 25, 30, 100], labels=['Underweight', 'Normal', 'Overweight', 'Obese'])*

*df['risk\_score'] = df['glucose']\*0.3 + df['cholesterol']\*0.3 + df['bmi']\*0.4*

***4. Outlier Detection & Treatment***

* *Remove or cap extreme outliers (important for tree-based or linear models)*

*python*

*Q1 = df['glucose'].quantile(0.25)*

*Q3 = df['glucose'].quantile(0.75)*

*IQR = Q3 - Q1*

*df = df[(df['glucose'] >= Q1 - 1.5 \* IQR) & (df['glucose'] <= Q3 + 1.5 \* IQR)]*

***5. Feature Scaling (if needed)***

* *For algorithms like* ***Logistic Regression, SVM, KNN****, apply scaling.*

*python*

*from sklearn.preprocessing import StandardScaler*

*num\_cols = ['age', 'glucose', 'bmi', 'cholesterol']*

*df[num\_cols] = StandardScaler().fit\_transform(df[num\_cols])*

***✅ Final Steps Before Modeling***

* *Define* ***X (features)*** *and* ***y (target variable)****:*

*python*

*X = df.drop('disease', axis=1)*

*y = df['disease']*

### 

### **8. Model Building**

### you can use various ML models depending on complexity and interpretability

### Baseline Models:

### Python

### from sklearn.linear\_model import LogisticRegression

### from sklearn.tree import DecisionTreeClassifier

### from sklearn.ensemble import RandomForestClassifier

### model = RandomForestClassifier()

### model.fit(X\_train, y\_train)

### Advanced Models:

### Gradient Boosting (XGBoost, LightGBM)

### Deep Learning (for time series, EHR, imaging, etc.)

### AutoML frameworks (e.g., AutoKeras, H2O)

### 

### **9. Visualization of Results & Model Insights**

**1. Data & Model Training Setup**

Python

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, roc\_auc\_score, roc\_curve

import matplotlib.pyplot as plt

import seaborn as sns

# Load sample dataset (replace with your own EHR data)

df = pd.read\_csv("heart.csv") # UCI Heart dataset

# Preprocessing

X = df.drop("target", axis=1)

y = df["target"]

### X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

### # Train model

### model = RandomForestClassifier()model.fit(X\_train, y\_train)

### y\_pred = model.predict(X\_test)

### **10. Tools and Technologies Used**

**🔹 1. Data Collection & Integration**

| Tool/Technology | Purpose |
| --- | --- |
|  |  |
| Electronic Health Records (EHR) | Patient data source (e.g., Epic, Cerner, MIMIC-III) |
| FHIR / HL7 APIs | Standard formats for health data exchange |
| Pandas / SQL | Data manipulation and querying |
| Apache NiFi / Airflow | Data pipelines and automation |
| FHIR Server (e.g., HAPI FHIR) | To access standardized patient records |

**🔹 2. Data Preprocessing & Exploration**

| Tool/Technology | Purpose |
| --- | --- |
| Python (Pandas, NumPy) | Data cleaning, transformation, imputation |
| Jupyter Notebooks | Interactive development environment |
| Matplotlib / Seaborn / Plotly | Data visualization |
| Scikit-learn / SciPy | Feature scaling, encoding, imbalanced class handling |
| AutoViz / Sweetviz | Automated EDA (Exploratory Data Analysis) |

**🔹 3. Model Building & Training**

| Tool/Technology | Purpose |
| --- | --- |
| Scikit-learn | Classical ML models (e.g., Logistic Regression, Random Forest) |
| XGBoost / LightGBM / CatBoost | Boosted tree models (fast & accurate) |
| TensorFlow / PyTorch | Deep learning (e.g., CNNs, RNNs for imaging or time-series) |
| AutoML (e.g., H2O.ai, AutoKeras) | Model selection and hyperparameter tuning |
| Google Colab / Kaggle Kernels | Free GPUs for training deep models |

**🔹 4. Model Evaluation & Explainability**

| Tool/Technology | Purpose |
| --- | --- |
| Sklearn.metrics | Confusion matrix, ROC-AUC, accuracy, precision/recall |
|  |  |
| SHAP / LIME | Explainable AI: global and local prediction explanations |
| Yellowbrick / MLxtend | Visualization of learning curves, classification reports |
| TensorBoard | Deep learning model monitoring |

**🔹 5. Model Deployment**

| Tool/Technology | Purpose |
| --- | --- |
| Flask / FastAPI | Build REST APIs to serve models |
| Docker | Containerize ML model for scalable deployment |
| Kubernetes | Orchestration of containerized ML services |
| MLflow / DVC | Model tracking, versioning, reproducibility |
| AWS SageMaker / Azure ML / GCP Vertex AI | Managed ML deployment platforms |

**🔹 6. Visualization of Results & Insights**

| Tool/Technology | Purpose |
| --- | --- |
| Matplotlib / Seaborn | Static plots (confusion matrix, ROC curves) |
| Plotly / Dash | Interactive dashboards |
| Streamlit | Quick app prototyping for ML models |
| Power BI / Tableau | Enterprise-grade health data visualization |
| SHAP Force / Summary Plots | Interpretability at patient and population levels |

**🔹 7. Security, Compliance & Privacy**

| Tool/Technology | Purpose |
| --- | --- |
| HIPAA / GDPR Compliance Tools | Ensure patient data privacy |
| Data Anonymization Libraries | De-identification of sensitive data |
| Vault / AWS KMS | Secure storage of secrets and keys |
| OAuth 2.0 / JWT | Secure API authentication for accessing patient data |

**Example Stack for End-to-End AI Health Prediction App**

| Layer | Tool/Tech |
| --- | --- |
| Frontend | Streamlit / Dash |
| API Layer | FastAPI |
| Model | XGBoost / TensorFlow |
| Serving | Docker + Flask API |
| Data Storage | PostgreSQL / FHIR Server |
| Security | HTTPS + JWT + HIPAA Controls |

**11.REPORT**

Impact & Future Work

**Benefits:**

* Early detection of chronic disease
* Personalized, data-driven treatment plans
* Reduced burden on healthcare systems

**Future Enhancements:**

* Use of real-time data from wearables (IoT)
* Deep learning for time-series health records
* Integration with hospital systems (FHIR APIs)

**Conclusion:**

This project demonstrates the potential of AI to transform healthcare from reactive to proactive. Our disease prediction model provides clinicians with an explainable, high-accuracy tool to detect high-risk patients before symptoms emerge — opening the door to early intervention, better outcomes, and smarter healthcare delivery.

### 

### **12. Team Members and Contributions**

***1. HARISH(data cleaning,reporting)  
2.G DIVYA DHARSHINI(EDA,model development)  
3.V VARUN(feature engineering)***

***4.S YALARASU(documentation,data cleaning)***