# **Heart Disease Detection Capstone Project**

## **Data Science Classification Challenge**

### **Project Overview**

This capstone project challenges students to develop and optimize machine learning models for predicting heart disease based on diagnostic test results. Students will work with clinical diagnostic data to build classification models that can assist healthcare professionals in early detection and risk assessment.

## **Problem Statement**

**Primary Question**: Can we accurately predict the presence of heart disease in patients based on their diagnostic test results and clinical measurements?

**Business Context**: Heart disease remains one of the leading causes of death globally. Early detection through diagnostic screening can significantly improve patient outcomes and reduce healthcare costs. This project simulates a real-world scenario where data scientists work with medical professionals to develop predictive models for clinical decision support.

**Challenge**: Given a patient's diagnostic test results (blood pressure, cholesterol levels, ECG results, exercise capacity, etc.), predict whether the patient has heart disease or not.

## **Project Goals**

### **Primary Objectives**

1. **Model Development**: Build and compare multiple classification algorithms (Decision Trees, Random Forest, Logistic Regression, Support Vector Machines)
2. **Performance Optimization**: Use grid search with k-fold cross-validation to optimize model hyperparameters
3. **Model Evaluation**: Assess models using appropriate metrics for medical diagnosis (accuracy, precision, recall, F1-score, ROC-AUC)
4. **Feature Analysis**: Identify the most important diagnostic indicators for heart disease prediction

### **Learning Outcomes**

* Master classification algorithms and their appropriate use cases
* Understand hyperparameter tuning and cross-validation techniques
* Learn to evaluate models in high-stakes domains (healthcare)
* Practice feature importance analysis and model interpretation
* Develop skills in presenting analytical findings to stakeholders

## **Dataset Description**

### **Target Variable**

* **heart\_disease**: Binary classification (0 = No heart disease, 1 = Heart disease present)

### **Feature Categories**

#### **Demographic Information**

* **age**: Patient age (years)
* **sex**: Gender (0 = Female, 1 = Male)

#### **Clinical Measurements**

* **chest\_pain\_type**: Type of chest pain (0-3: Typical angina, Atypical angina, Non-anginal pain, Asymptomatic)
* **resting\_blood\_pressure**: Resting blood pressure (mm Hg)
* **cholesterol**: Serum cholesterol level (mg/dl)
* **fasting\_blood\_sugar**: Fasting blood sugar > 120 mg/dl (0 = False, 1 = True)

#### **Diagnostic Test Results**

* **resting\_ecg**: Resting electrocardiographic results (0-2)
* **max\_heart\_rate**: Maximum heart rate achieved during exercise
* **exercise\_induced\_angina**: Exercise-induced angina (0 = No, 1 = Yes)
* **st\_depression**: ST depression induced by exercise relative to rest
* **st\_slope**: Slope of peak exercise ST segment (0-2)
* **num\_major\_vessels**: Number of major vessels colored by fluoroscopy (0-3)
* **thalassemia**: Thalassemia test result (0-3)

## **Project Steps**

### **Phase 1: Data Exploration and Preprocessing (Week 1)**

#### **1.1 Initial Data Analysis**

* Load and examine the synthetic dataset
* Perform descriptive statistics analysis
* Check data types and missing values
* Create visualizations to understand feature distributions

#### **1.2 Exploratory Data Analysis (EDA)**

* Analyze target variable distribution
* Create correlation matrices and heatmaps
* Generate pair plots for continuous variables
* Examine relationships between features and target variable
* Identify potential outliers

#### **1.3 Data Preprocessing**

* Handle missing values (if any)
* Encode categorical variables appropriately
* Scale/normalize features for algorithms that require it
* Split data into training and testing sets (80/20 split)

**Deliverable**: EDA Report with visualizations and preprocessing decisions

### **Phase 2: Baseline Model Development (Week 2)**

#### **2.1 Model Implementation**

Implement four classification algorithms with default parameters:

1. **Decision Tree Classifier**
   * Understand tree structure and interpretability
   * Analyze feature importance
2. **Random Forest Classifier**
   * Leverage ensemble learning
   * Compare with single decision tree
3. **Logistic Regression**
   * Understand linear decision boundaries
   * Interpret coefficients
4. **Support Vector Machine (SVM)**
   * Experiment with different kernels
   * Understand margin maximization

#### **2.2 Initial Model Evaluation**

* Use 5-fold cross-validation for initial assessment
* Calculate baseline metrics for each model:
  + Accuracy
  + Precision
  + Recall
  + F1-Score
  + ROC-AUC Score
* Create confusion matrices
* Plot ROC curves for comparison

**Deliverable**: Baseline model comparison report

### **Phase 3: Hyperparameter Optimization (Week 3)**

#### **3.1 Grid Search Implementation**

For each algorithm, define hyperparameter grids:

**Decision Tree**:

dt\_param\_grid = {

'max\_depth': [3, 5, 7, 10, None],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

'criterion': ['gini', 'entropy']

}

**Random Forest**:

rf\_param\_grid = {

'n\_estimators': [50, 100, 200],

'max\_depth': [3, 5, 7, None],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

'max\_features': ['sqrt', 'log2', None]

}

**Logistic Regression**:

lr\_param\_grid = {

'C': [0.01, 0.1, 1, 10, 100],

'penalty': ['l1', 'l2'],

'solver': ['liblinear', 'saga'],

'max\_iter': [1000, 2000]

}

**Support Vector Machine**:

svm\_param\_grid = {

'C': [0.1, 1, 10, 100],

'kernel': ['linear', 'rbf', 'poly'],

'gamma': ['scale', 'auto', 0.001, 0.01, 0.1, 1]

}

#### **3.2 Cross-Validation Strategy**

* Implement stratified k-fold cross-validation (k=5 or k=10)
* Use appropriate scoring metrics (consider using F1-score or ROC-AUC for imbalanced datasets)
* Track computational time for each model

#### **3.3 Optimization Process**

1. Run grid search with cross-validation for each algorithm
2. Identify best hyperparameters for each model
3. Retrain models with optimal parameters
4. Compare optimized models against baselines

**Deliverable**: Hyperparameter optimization report with best parameters and performance improvements

## **Technical Requirements**

### **Required Libraries**

# Data manipulation and analysis

import pandas as pd

import numpy as np

# Visualization

import matplotlib.pyplot as plt

import seaborn as sns

# Machine learning

from sklearn.model\_selection import train\_test\_split, GridSearchCV, cross\_val\_score, StratifiedKFold

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score

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

from sklearn.inspection import permutation\_importance

# Statistical analysis

from scipy import stats

### **Evaluation Metrics Focus**

Given the medical context, emphasize:

* **Sensitivity (Recall)**: Correctly identifying patients with heart disease
* **Specificity**: Correctly identifying healthy patients
* **Precision**: Minimizing false positive diagnoses
* **F1-Score**: Balanced measure for potentially imbalanced dataset
* **ROC-AUC**: Overall discriminative ability

## **Success Metrics**

### **Minimum Viable Product**

* All four algorithms implemented and evaluated
* Grid search optimization completed
* Cross-validation properly implemented
* Clear model comparison and recommendation

### **Excellence Indicators**

* ROC-AUC > 0.85 on test set
* Comprehensive feature importance analysis
* Professional-quality presentation
* Actionable business recommendations
* Thoughtful discussion of medical implications

# **Additional Steps: Model Deployment Pipeline**

Here are the additional steps to transform your machine learning model into a complete production-ready application:

## **Step 10: Model Serialization and Persistence**

**Goal**: Save the trained model and preprocessing components for deployment

**What to do**:

* Save the trained model using pickle
* Save any preprocessing objects (scalers, encoders)
* Create a model metadata file with performance metrics
* Implement model loading functions

## **Step 11: FastAPI Backend Development**

**Goal**: Create a REST API to serve model predictions

**What to do**:

* Create FastAPI application with prediction endpoints
* Implement input validation using Pydantic models
* Add health checks and model information endpoints
* Handle errors gracefully

## **Step 12: Docker Configuration**

**Goal**: Containerize the FastAPI application for easy deployment

**What to do**:

* Create Dockerfile for the FastAPI application
* Set up requirements.txt with all dependencies
* Create docker-compose.yml for multi-service deployment
* Add .dockerignore for optimization