🧠 Project Title: Heart Disease Risk Prediction Using Machine Learning

---

🎯 Objective:

Build a robust classification model to predict whether a patient is at risk of heart disease based on medical and lifestyle indicators. The goal is to apply the full ML pipeline: data cleaning, EDA, preprocessing, statistical testing, model building, evaluation, and performance tuning.

📁 Dataset Overview: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

- 📌 **Total Rows**: 918
- 📌 **Columns (Features)**: 12 + 1 target (`HeartDisease`)
- ✅ Already cleaned — no missing values
- 🏷️ **Target column**: `HeartDisease` (1 = heart disease present, 0 = not)

🔢 Features:

Feature	Description
`Age`	Age of the patient
`Sex`	Sex (M or F)
`ChestPainType`	Type of chest pain
`RestingBP`	Resting blood pressure
`Cholesterol`	Serum cholesterol
`FastingBS`	Fasting blood sugar > 120 mg/dl (1 = true, 0 = false)
`RestingECG`	Resting electrocardiogram results
`MaxHR`	Maximum heart rate achieved
`ExerciseAngina`	Exercise-induced angina (Y/N)
`Oldpeak`	ST depression
`ST_Slope`	Slope of the peak exercise ST segment
`HeartDisease`	Target (1 or 0)

🔧 Project Workflow

1. 📊 Data Loading & Inspection

- Load data using `pandas`
- Check data types, shape, and unique values
- Check for nulls (none exist here, but good practice)

2. 🧹 Data Cleaning & Preprocessing

- Convert categorical columns (`Sex`, `ChestPainType`, etc.) using:
  - One-Hot Encoding (preferred) or Label Encoding
- Scale numeric columns (`RestingBP`, `Cholesterol`, etc.) using:
  - `StandardScaler` or `MinMaxScaler`
- Handle class imbalance if observed (check `value_counts` of `HeartDisease`)
- Train-test split (80:20)

Tools: `pandas`, `sklearn.preprocessing`, `train_test_split`

---

3. 📈 Exploratory Data Analysis (EDA)

- Univariate & bivariate plots
- Analyze:
  - Age vs HeartDisease
  - ChestPainType distribution
  - Cholesterol by HeartDisease status
- Correlation heatmap
- Pairplot for key variables

Tools: `seaborn`, `matplotlib`

---

4. 🧪 Statistical Testing (Optional/Advanced)

- **Chi-square test** for categorical features vs target
- **t-test** for numerical features vs target
- Identify features with strong relationships to target

Tools: `scipy.stats`, `statsmodels`

---

5. 🤖 Model Development

Train multiple models:

- Logistic Regression

- Decision Tree
- Random Forest
- XGBoost
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)

Use:

- Cross-validation
- GridSearchCV or RandomizedSearchCV for hyperparameter tuning

Tools: `sklearn`, `xgboost`

---

6. 📊 Model Evaluation

Evaluate using:

- Accuracy
- Precision, Recall, F1-score
- Confusion Matrix
- ROC-AUC
- Classification Report

Visualizations:

- ROC curves
- Precision-recall curves

---

7. 🚀 Performance Improvement

- Feature selection using `SelectKBest`, `RFE`, or model importance
- Try ensemble models (e.g., VotingClassifier)
- Handle overfitting via regularization or pruning

---

8. 🔍 Model Interpretation (Optional/Advanced)

- Feature importance plots from RandomForest or XGBoost
- SHAP or LIME explanations for predictions

Tools: `shap`, `lime`, `eli5`

---

9. 📝 Reporting

Deliverables:

- Jupyter Notebook with full code and markdowns
- Summary PDF/PPT with:
  - Insights from EDA
  - Model comparison and justification
  - Final model evaluation and business interpretation
