Task 1: Iris Dataset Exploration

Objective

To learn how to load, inspect, and visualize the Iris dataset to understand its data trends and distributions.

1. Dataset Loading

- **Library Used**: pandas, seaborn, matplotlib
- **Dataset Source**: Loaded using seaborn.load_dataset('iris')

2. Dataset Inspection

Display:

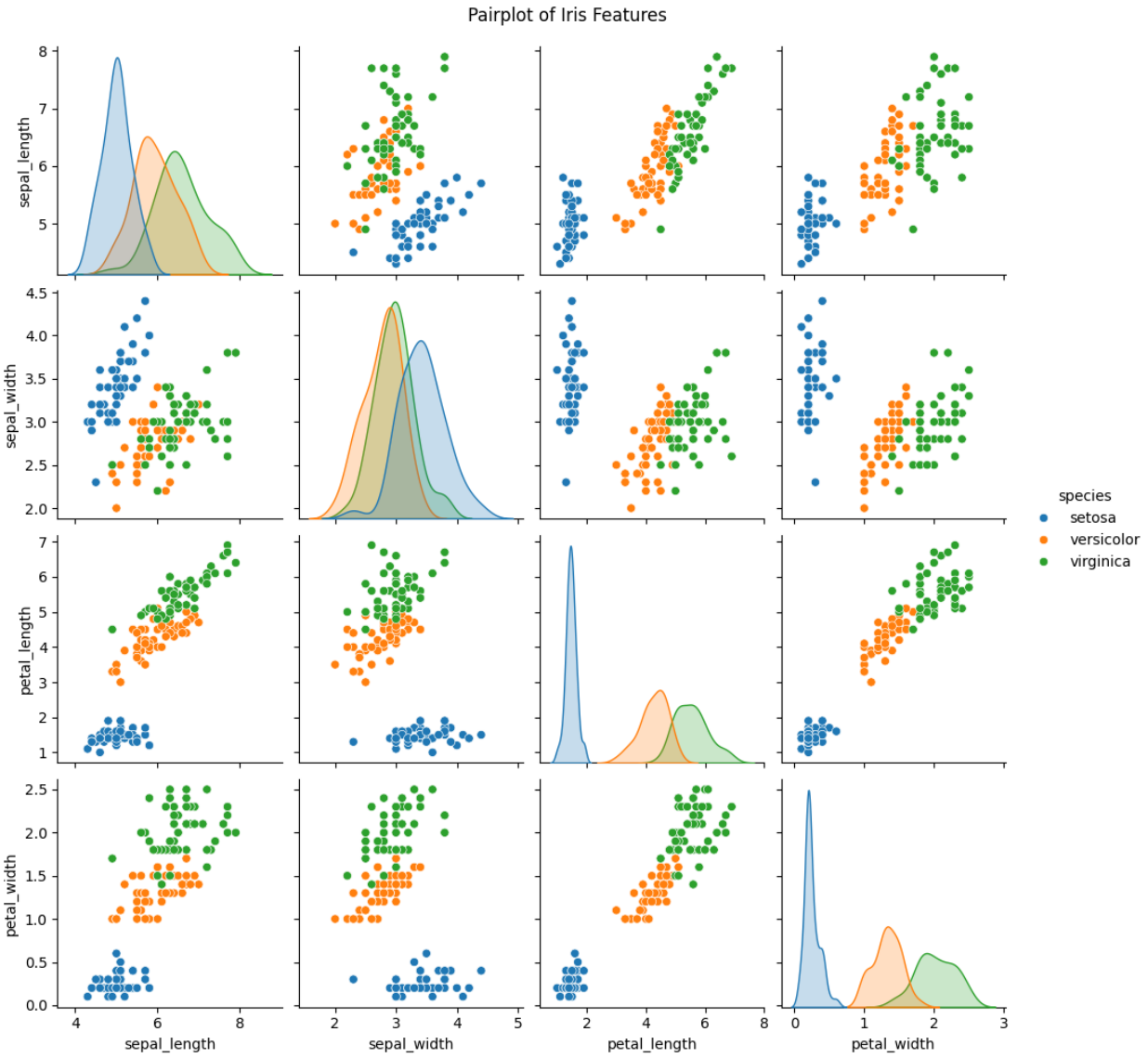
1. Shape and Columns, first five rows.
2. Dataset Information:
    - 150 entries
    - 5 columns (4 numerical, 1 categorical)
    - No missing values
3. Summary Statistics:
    - Mean, std deviation, min, max, and percentiles for all features.
    - Petal dimensions show greater variance between species than sepal dimensions.

3. Data Visualization

a. Scatter Plots (Pairplot)

Insight:

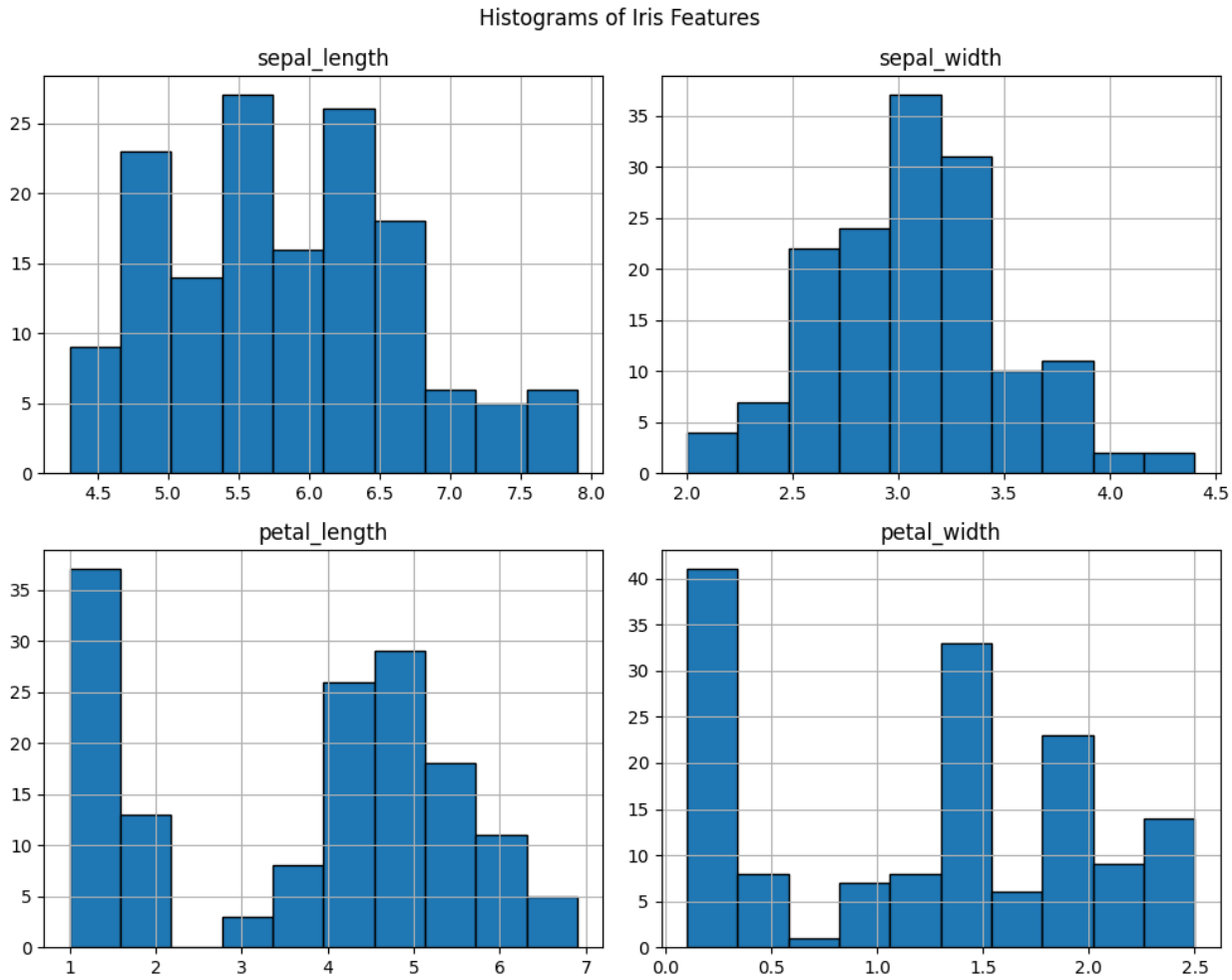
- Clear separation between species based on petal length and width.
- Setosa is most distinguishable from the others.



b. Histograms

Insight:

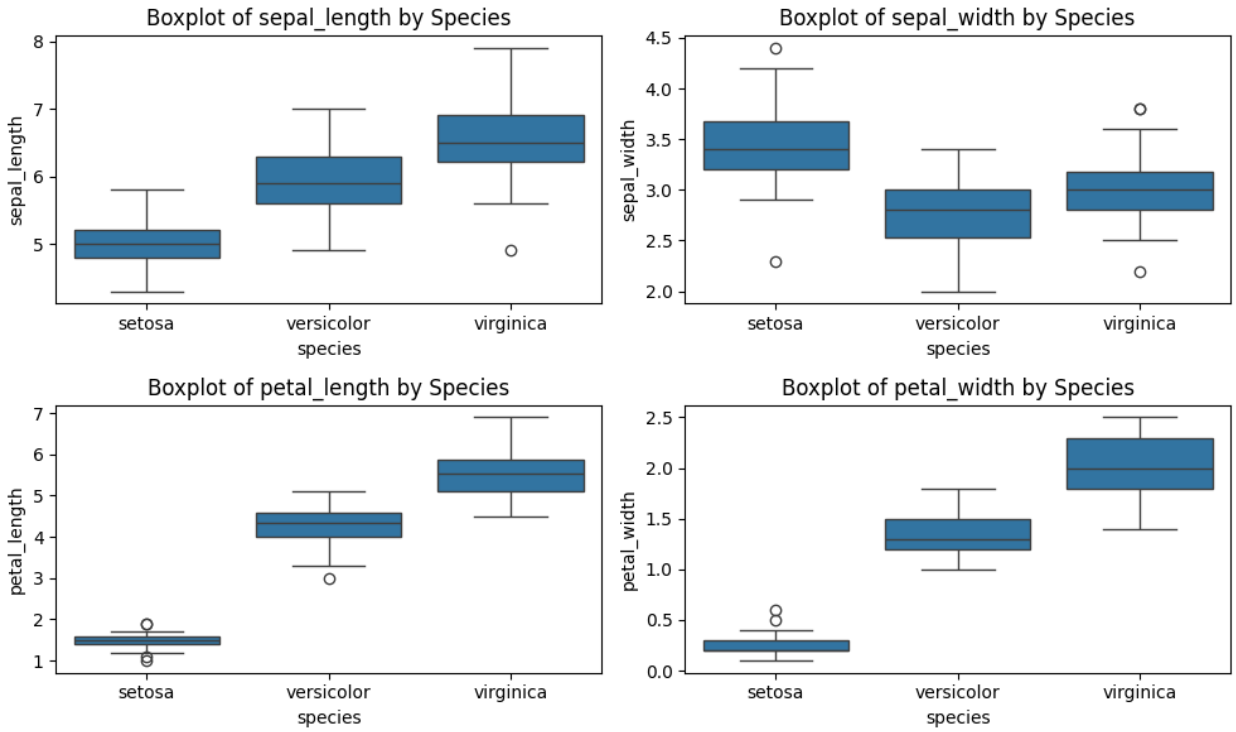
- Petal lengths vary significantly between species.
- Sepal width has a more uniform distribution.



c. Box Plots

Insight:

- Outliers are minimal.
- Petal features are strong indicators of species.
- Sepal width shows overlap but still contributes to classification.



Task 2: Stock Price Prediction

Objective

The objective of this project is to predict the next day's closing price of a stock using historical market data. We use features such as Open, High, Low, and Volume to forecast the closing price with two machine learning models: Linear Regression and Random Forest.

1. Data Source

The historical stock data was retrieved using the yfinance Python library. For this analysis, we selected **Apple Inc. (AAPL)** as the target stock and downloaded daily data from **January 1, 2020 to December 31, 2024**.

2. Data Preparation

The dataset includes the following columns:

- Open: Opening price of the stock on the trading day.
- High: Highest price during the trading day.
- Low: Lowest price during the trading day.

- Close: Closing price of the stock.
- Volume: Number of shares traded.

A new column Next_Close was created by shifting the Close column by one day. This becomes the prediction target.

The features used for prediction:

- Open
- High
- Low
- Volume

3. Modeling Approach

We used two regression models for this task:

- **Linear Regression**: A simple baseline model that assumes a linear relationship between input features and the target.
- **Random Forest Regressor**: An ensemble model that uses multiple decision trees to make more robust predictions and handle nonlinearities in the data.

The data was split into training (80%) and testing (20%) sets, without shuffling, to preserve the time series order.

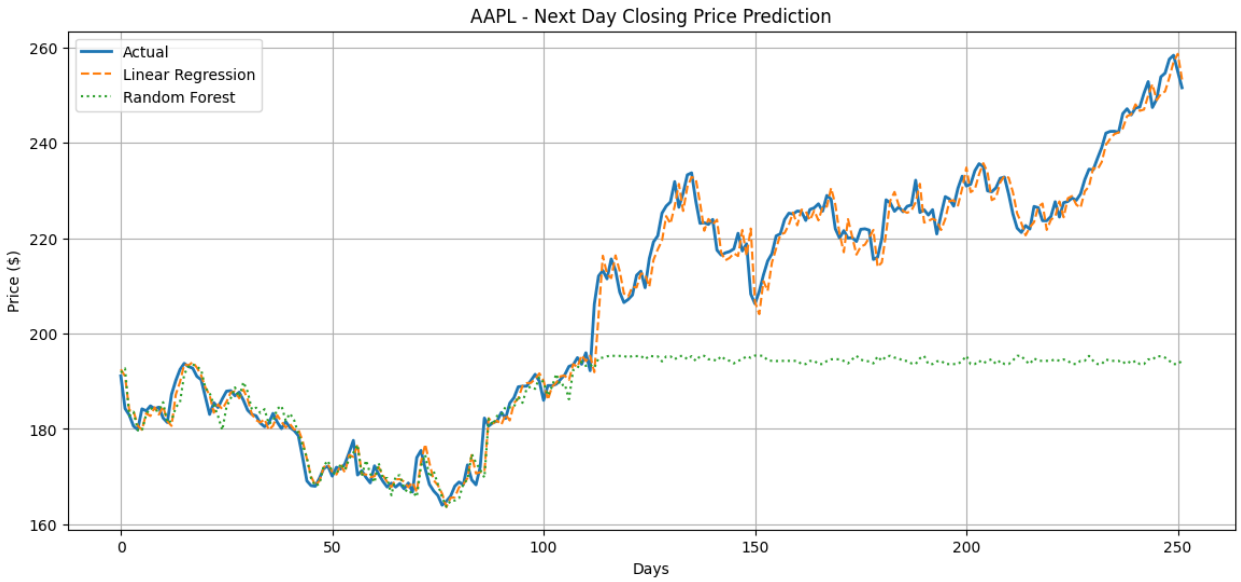
4. Results

The performance of the models was evaluated using **Root Mean Squared Error (RMSE)** on the test set:

Model	RMSE (USD)
Linear Regression	3.11
Random Forest	*25.72*

- **Blue Line**: Actual closing prices.
- **Orange Dashed Line**: Linear Regression predictions.
- **Green Dotted Line**: Random Forest predictions.

The Linear Regression model demonstrated superior performance, capturing more of the complex relationships between features and the target price.



Task 3: Heart Disease Prediction

Objective

To build a classification model that predicts whether a person is at risk of heart disease based on clinical and demographic health data using the UCI Heart Disease dataset.

Dataset Summary

- **Source**: UCI Heart Disease dataset
- **Records**: 920
- **Target Variable**: num (re-coded to binary: 0 = No disease, 1 = Disease)

Key Features

- **Numerical**: age, trestbps, chol, thalch, oldpeak
- **Categorical**: sex, cp (chest pain type), fbs, restecg, exang, slope, dataset

Data Cleaning

- Dropped columns with over 50% missing data: ca, thal
- Imputed:
    - **Numerical features** with median
    - **Categorical features** with mode

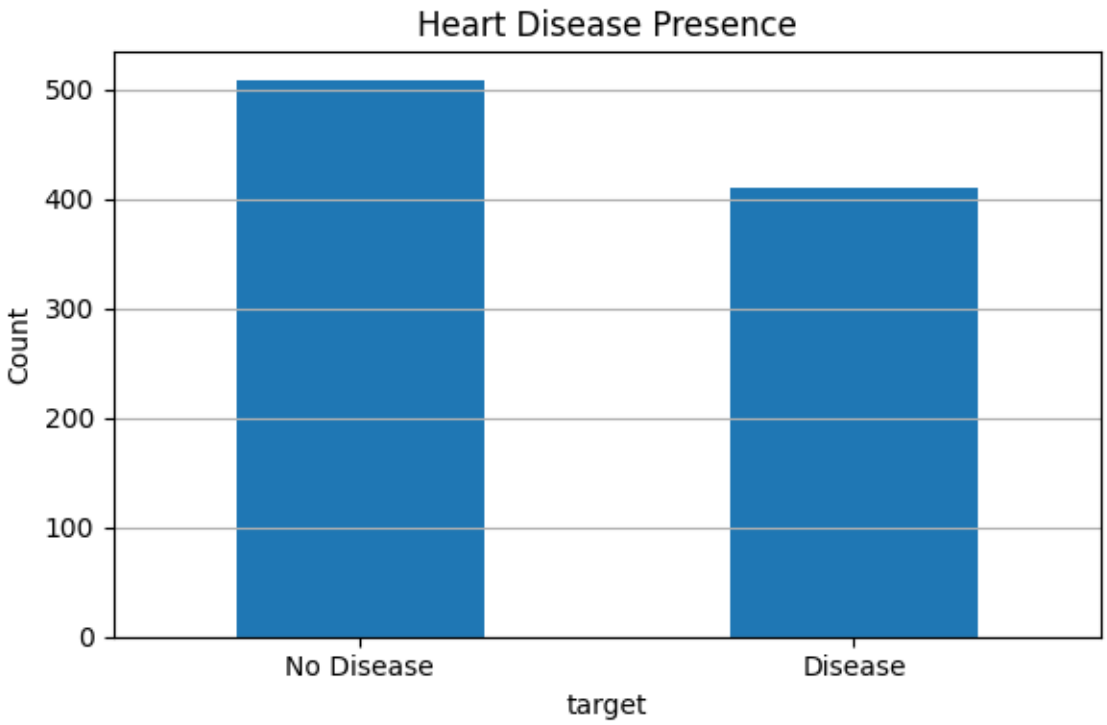
Exploratory Data Analysis (EDA)

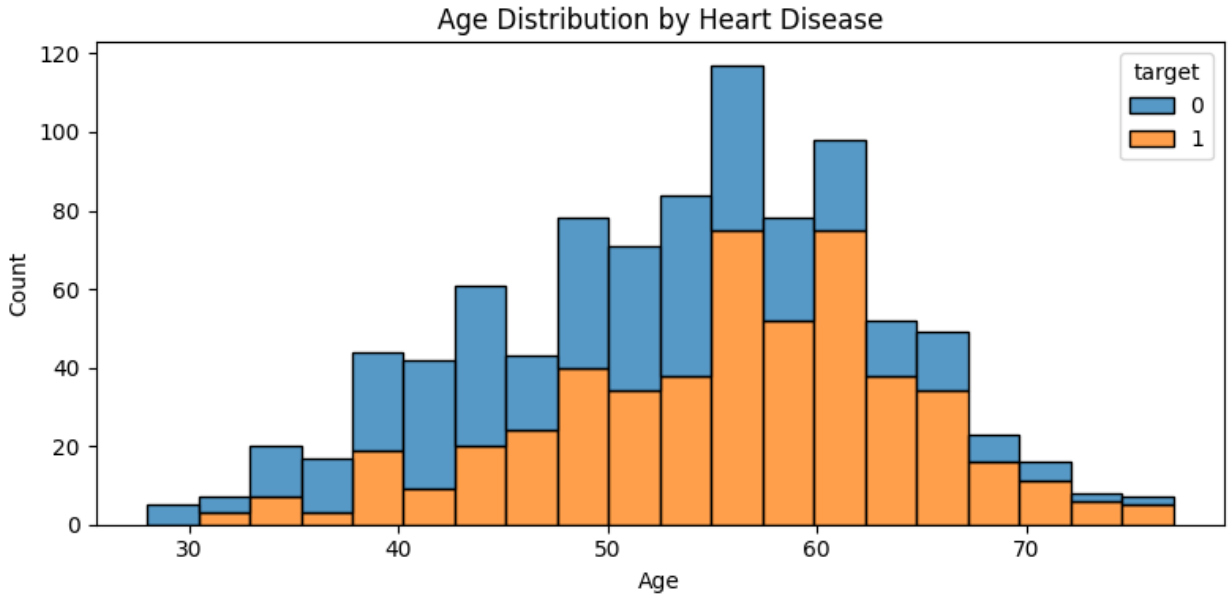
1. Target Distribution

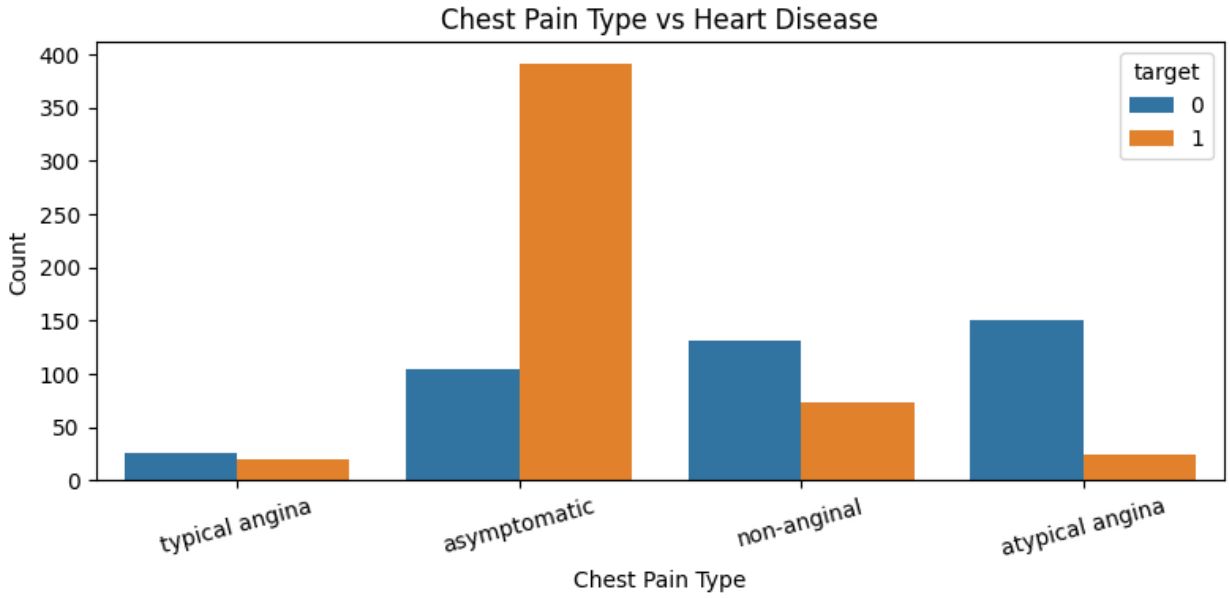
- ~60% of patients were diagnosed with heart disease.
- ~40% had no heart disease.

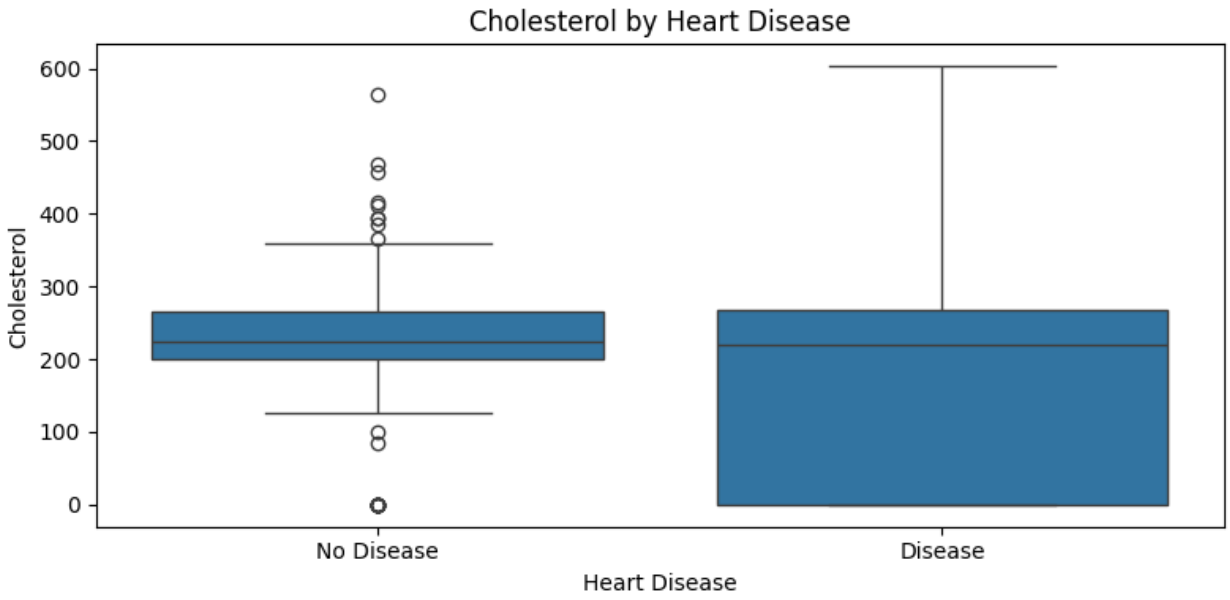
2. Key Findings

- **Age**: Slightly older patients had a higher risk.
- **Chest Pain**: "Asymptomatic" chest pain strongly associated with disease.
- **Cholesterol**: Greater variance in patients with heart disease.




Age Distribution by Heart Disease
target
0
1
Count
Age
0
20
40
60
80
100
120
30
40
50
60
70


Chest Pain Type vs Heart Disease
target
0
1
Count
Chest Pain Type
0
50
100
150
200
250
300
350
400
typical angina
asymptomatic
non-anginal
atypical angina



Modeling

Algorithm: Logistic Regression

- **Data split**: 80% training, 20% testing
- **Preprocessing**: One-hot encoding for categorical features

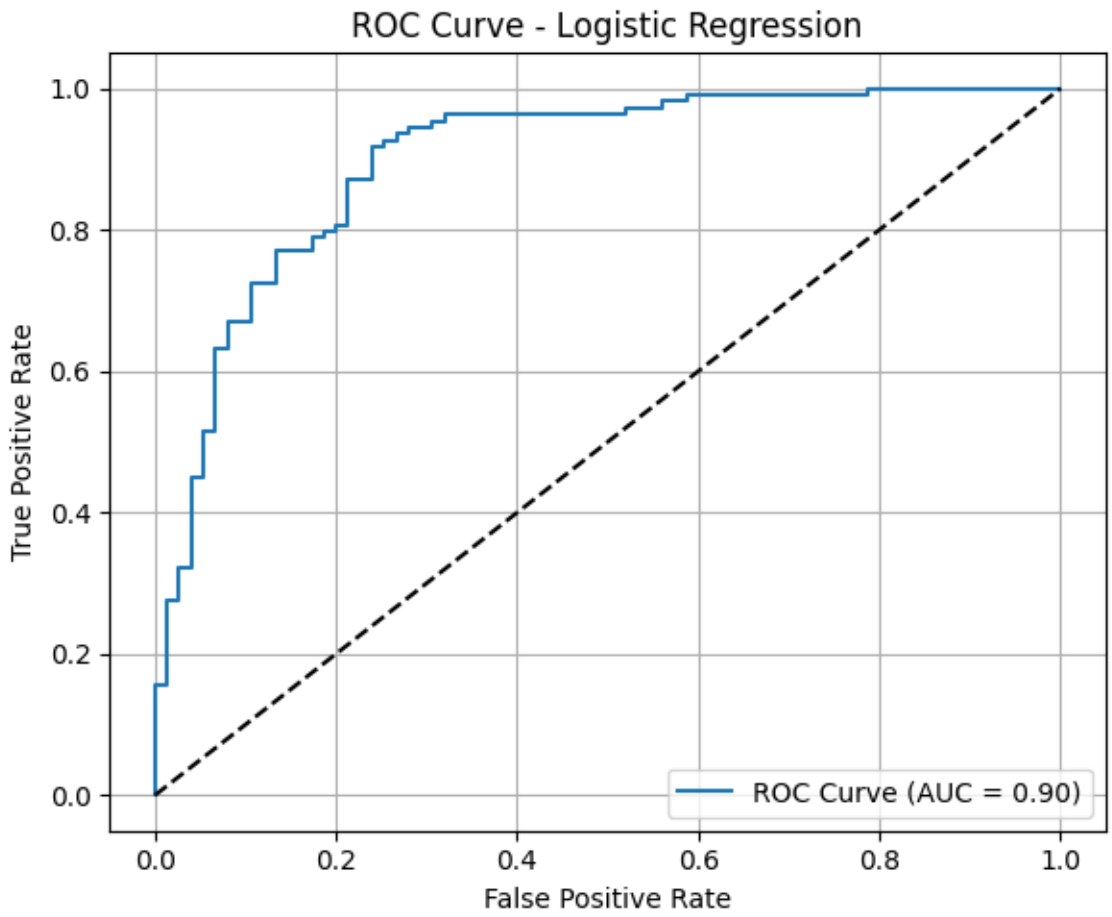
Evaluation Metrics

Metric	***Score***
Accuracy	82.06%
ROC AUC Score	0.8987

Confusion Matrix:

[[59, 16],

[17, 92]]



Important Features

FEATURE	EFFECT
DATASET_SWITZERLAND	↑ Increases risk
CP_ATYPICAL ANGINA	↓ Decreases risk
SEX_MALE	↑ Increases risk
CP_TYPICAL ANGINA	↓ Decreases risk
CP_NON-ANGINAL	↓ Decreases risk

Task 4: General Health Query Chatbot using LLM

1. Project Overview

The goal of this project is to create a **General Health Query Chatbot** that leverages a **Large Language Model (LLM)** to answer common health-related questions in a **safe**, **friendly**, and **easy-to-understand** manner. This chatbot does not replace professional medical advice but provides general health information.

2. Requirements

pip install transformers accelerate

Generate a token for hugging face and go to https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1 and give your contact details to access this LLM.

3. Tools & Technologies Used

- **Model:** Mistral-7B-Instruct-v0.1 (Open-source LLM)
- **Library:** Hugging Face transformers, accelerate
- **Language:** Python
- **Interface:** Command-line based chatbot

4. Key Features

Prompt Engineering

- Custom prompt designed to instruct the model to act as a **friendly medical assistant**.
- Response tone is made clear, educational, and non-alarming.
- Example prompt snippet:

"You are a helpful and friendly medical assistant. Answer general health questions in a simple, safe, and informative way..."

Safety Filters

- A list of red-flag keywords helps identify **risky or sensitive topics** such as:
    - Medication dosages
    - Diagnoses

    - Mental health crises
    - Emergencies
- These queries trigger a default response encouraging the user to seek professional care.

Real-Time Interaction

- Users can interact with the chatbot in a loop through the terminal.
- Responses are generated on-the-fly with transformers text-generation pipeline.

5. Sample Queries and Responses

User Query	Chatbot Response
What causes a sore throat?	A sore throat can be caused by viral infections, dry air, or allergies.
Is paracetamol safe for children?	Paracetamol can be safe if given in the correct dose. Consult dosage guidelines.
Can I take 10 sleeping pills?	This may involve serious health risks. Please speak to a medical professional.

6. Limitations

- The model cannot access real-time medical databases.
- Does not support diagnosis or emergency advice.
- Large models may require significant memory and compute resources (ideally a GPU with ≥12 GB VRAM).

Task 5: Mental Health Support Chatbot using DistilGPT2

Project Overview

This project involves building a **mental health support chatbot** fine-tuned on the **EmpatheticDialogues** dataset using **DistilGPT2**, a lightweight version of GPT-2. The goal was to create a chatbot capable of generating emotionally supportive and empathetic responses to user inputs related to stress, anxiety, and emotional well-being.

Objectives

- Fine-tune a small language model (DistilGPT2) for empathetic dialogue generation.
- Use Facebook AI's **EmpatheticDialogues** dataset for training.
- Create a simple and accessible **Streamlit** interface to interact with the model.
- Ensure the model responds with **emotionally supportive**, **non-judgmental**, and **conversational** tone.

Requirements

!pip install transformers datasets accelerate streamlit

!pip install -U datasets

!pip install fsspec==2023.9.2

Tools & Technologies

Tool	***Purpose***
DistilGPT2	Base language model (light GPT-2)
EmpatheticDialogues	Dataset of human empathetic conversations
Hugging Face Transformers	Model fine-tuning and inference
Streamlit	Lightweight web UI for chatbot interface
Google Colab	Training and testing environment
PyTorch	Backend framework for model training

Dataset Details

- **Name**: EmpatheticDialogues
- **Source**: Facebook AI
- **Size**: ~80,000 multi-turn conversations
- **Usage**: Only context and response (utterance) pairs were used.
- **Goal**: Teach the model how humans respond empathetically in emotionally charged conversations.

Methodology

1. Data Preprocessing

- Combined context and utterance as a single training sequence.
- Tokenized using distilgpt2 tokenizer.
- Applied truncation and max padding (128 tokens).
- Labels were set equal to input_ids (causal language modeling requirement).

2. Fine-Tuning

- Used Hugging Face's Trainer API.
- 3 epochs of training.
- Batch size: 8
- Mixed precision training enabled (fp16).
- Evaluation was skipped during training (for simplicity).

3. Model Saving

- Trained model saved to: empathetic-distilgpt2/
- Tokenizer also saved for consistent inference.

4. Interface (Streamlit)

- Built a chatbot UI using Streamlit.
- Supports multi-turn chat history.
- Generates responses with top-k sampling (top_k=50, top_p=0.9, temperature=0.7) to encourage empathetic variation.

Example Interaction

User: "I've been feeling really alone lately."
Bot: "I'm really sorry you're feeling this way. You're not alone, and it's okay to talk about it."

Results

- The chatbot generates emotionally appropriate responses.
- It avoids blunt, robotic, or generic replies thanks to fine-tuning on human empathetic dialogues.

- Streamlit app provides a friendly interface for non-technical users.

Deployment Options

- **Run locally** with streamlit run app.py on command line interface in the same directory as the app.py file
- Optional: deploy to **Hugging Face Spaces**, **Heroku**, or **Streamlit Cloud**

Task 6: House Price Prediction

Objective

The goal of this project was to develop a machine learning model to predict house prices based on property features such as square footage (area), number of bedrooms, location attributes, and other facilities.

Dataset Overview

- **Source**: Provided CSV dataset from Kaggle (Housing.csv)
- **Records**: 545 rows
- **Features**: 12 input features and 1 target (price)
- **Feature Types**:
    - **Numerical**: area, bedrooms, bathrooms, stories, parking

- **Categorical**: mainroad, guestroom, basement, hotwaterheating, airconditioning, prefarea, furnishingstatus

Data Preprocessing

- **Categorical Encoding**: One-hot encoding (dropping first category to avoid multicollinearity)
- **Numerical Features**: Used as-is (no scaling was required for tree-based models)

Models Trained

1. **Linear Regression**
2. **Gradient Boosting Regressor**

Both models were implemented using pipelines to ensure reproducible preprocessing and training.

Evaluation Metrics

We evaluated the models on the test set using:

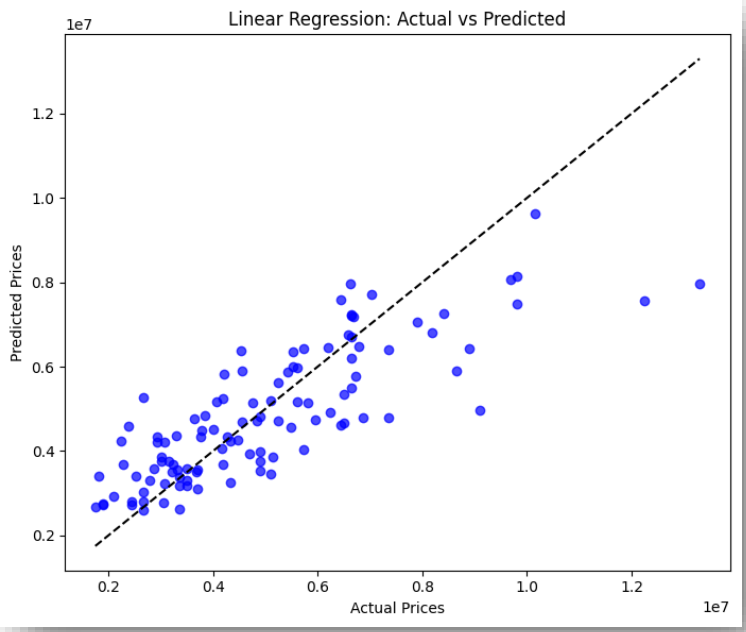
- **Mean Absolute Error (MAE)**: Measures average absolute prediction error
- **Root Mean Squared Error (RMSE)**: Penalizes larger errors more heavily

Model	***MAE (₹)***	***RMSE (₹)***
Linear Regression	₹970,043	₹1,324,507
Gradient Boosting	₹959,749	₹1,299,386

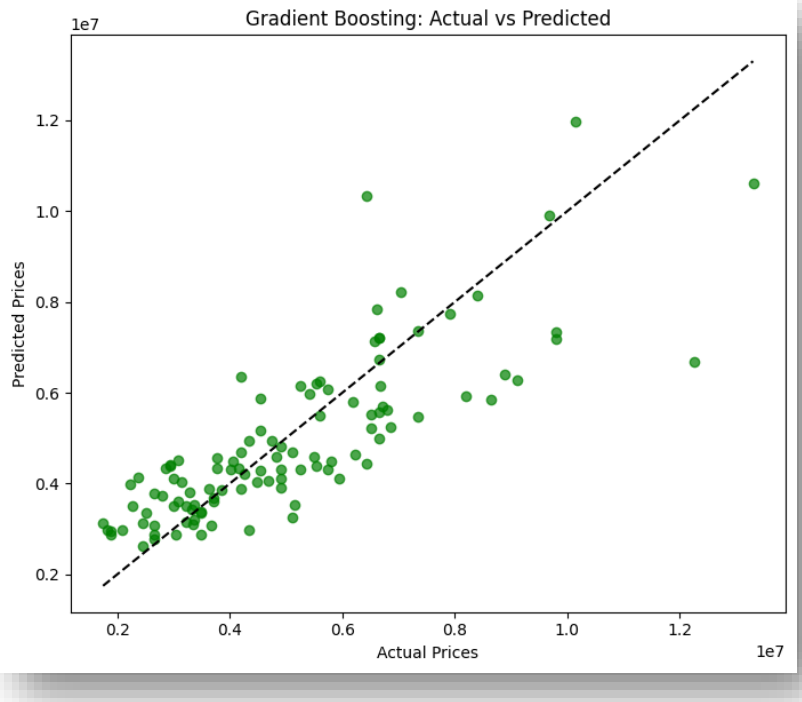
Visualization

Scatter plots comparing predicted prices against actual prices showed that:

- **Linear Regression** had a wider spread of errors, especially at higher price ranges.



- **Gradient Boosting** predictions were closer to the ideal line, indicating more consistent performance.



Conclusion

- **Gradient Boosting** outperformed Linear Regression in both error metrics and visual accuracy.
- It is a better choice for this regression task due to its ability to handle nonlinear relationships and interactions.