Data Science

Assignment

TY BSC IT

Personal Details

Name	:	Shaha Toyab Abdul Salam
Roll no	:	31010922088
Class	:	TY BSCIT
Subject	:	Data Science
Professor	:	Mohammhad Bilal Shaikh

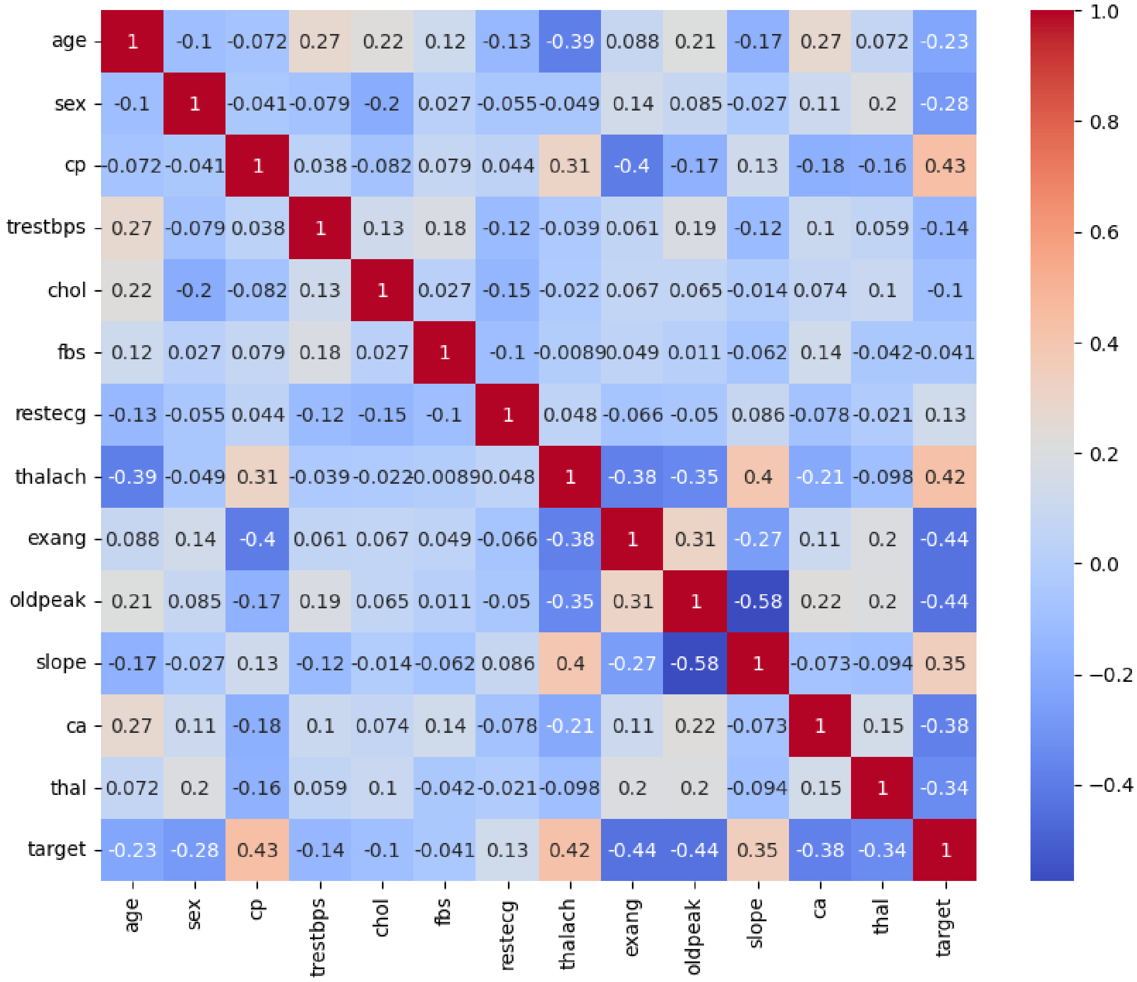
Heart Disease Prediction using Machine Learning

1. Problem Statement

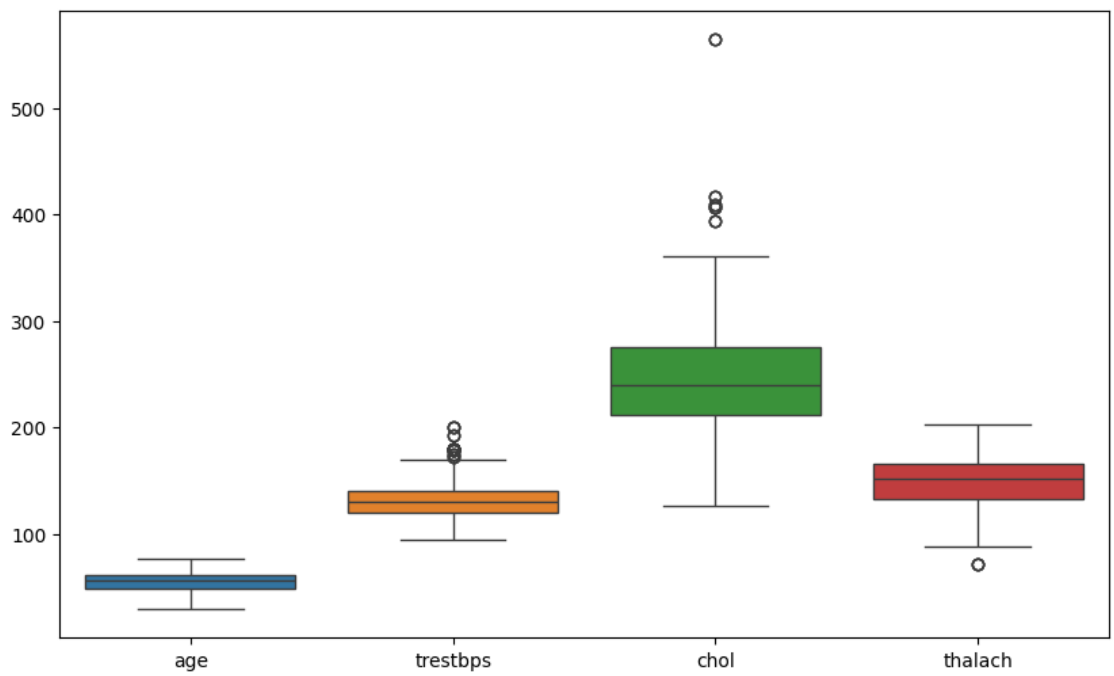
The objective of this assignment was to develop a machine learning model to predict the presence or absence of heart disease in patients using the Heart Disease Dataset from Kaggle. As a Data Scientist working for a healthcare analytics company, the task involved building a classification model based on health parameters such as age, gender, cholesterol levels, and maximum heart rate. The goal was to create an accurate and deployable model to assist in early disease detection, with a focus on data preprocessing, exploratory data analysis (EDA), feature engineering, model training with hyperparameter tuning, and deployment via a web application using Streamlit.

2. Data Preprocessing and Insights

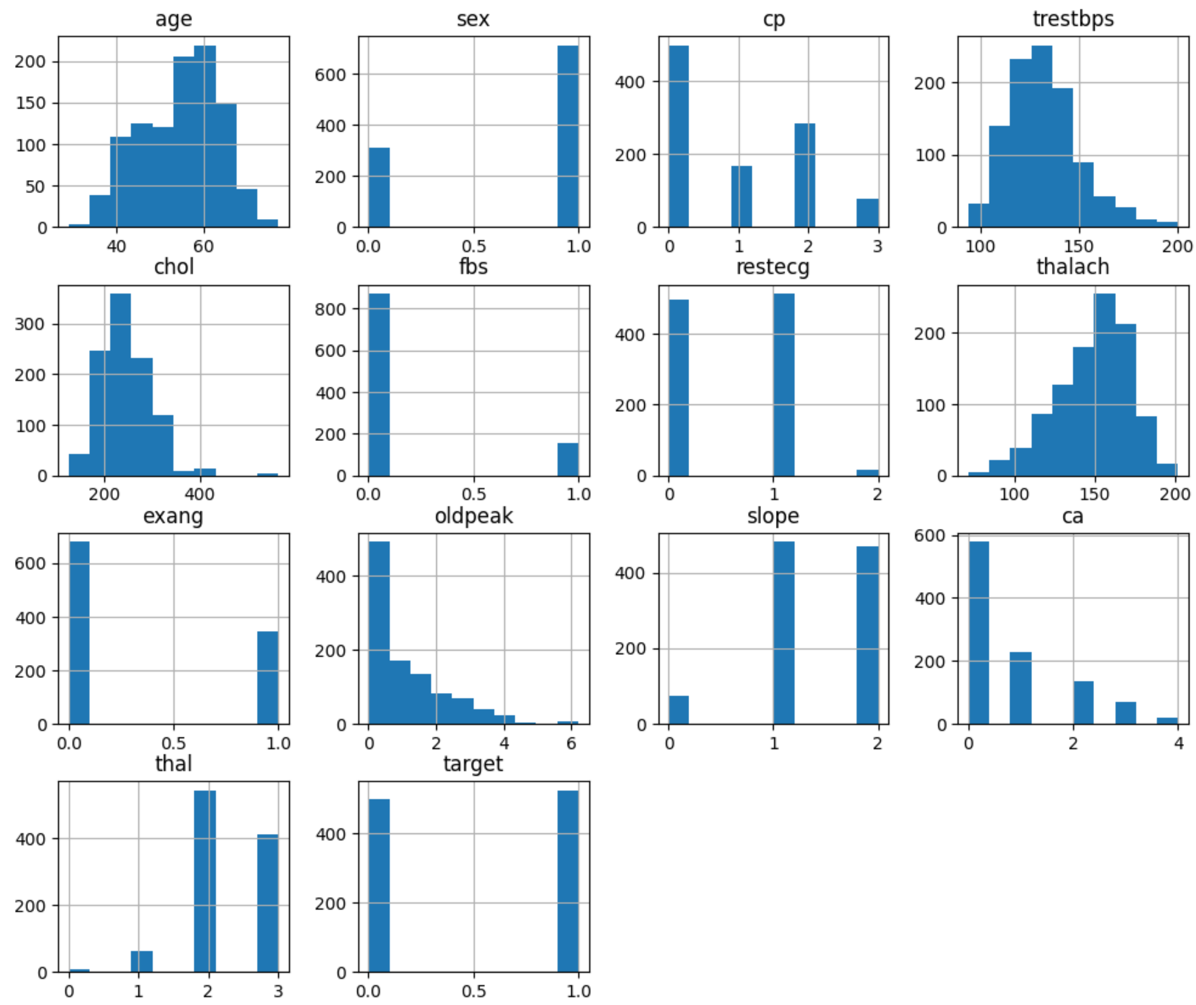
The Heart Disease Dataset was loaded and analyzed, revealing no missing values, which simplified preprocessing. Exploratory Data Analysis (EDA) provided key insights: the average patient age was approximately 54 years, with cholesterol levels averaging 246 mg/dl and resting blood pressure at 131 mmHg, indicating potential risk factors. Histograms showed slight skewness in features like thalach (maximum heart rate) and oldpeak (ST depression), necessitating standardization using StandardScaler. The correlation matrix highlighted significant relationships, with cp (chest pain type, r = 0.43) and exang (exercise-induced angina, r = 0.44) positively correlated with the target, and thalach (r = -0.42) negatively correlated, suggesting their predictive importance. Outliers in trestbps and chol (e.g., cholesterol > 400 mg/dl) were retained as clinically relevant. These insights guided feature selection and model development.



Correlation Matrix Heatmap



Box plots for outlier detection

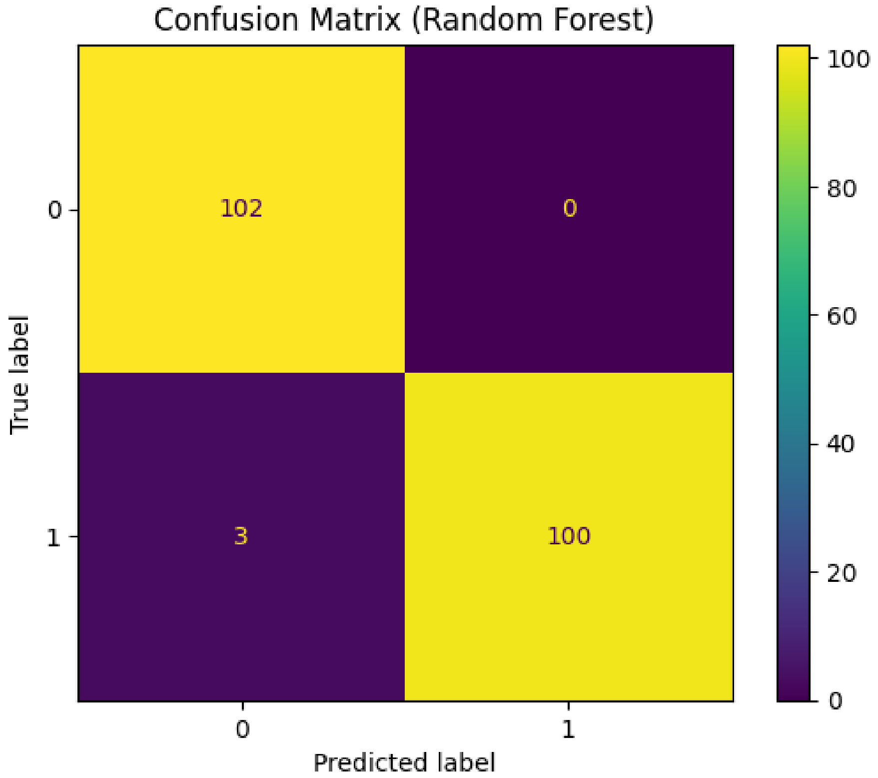


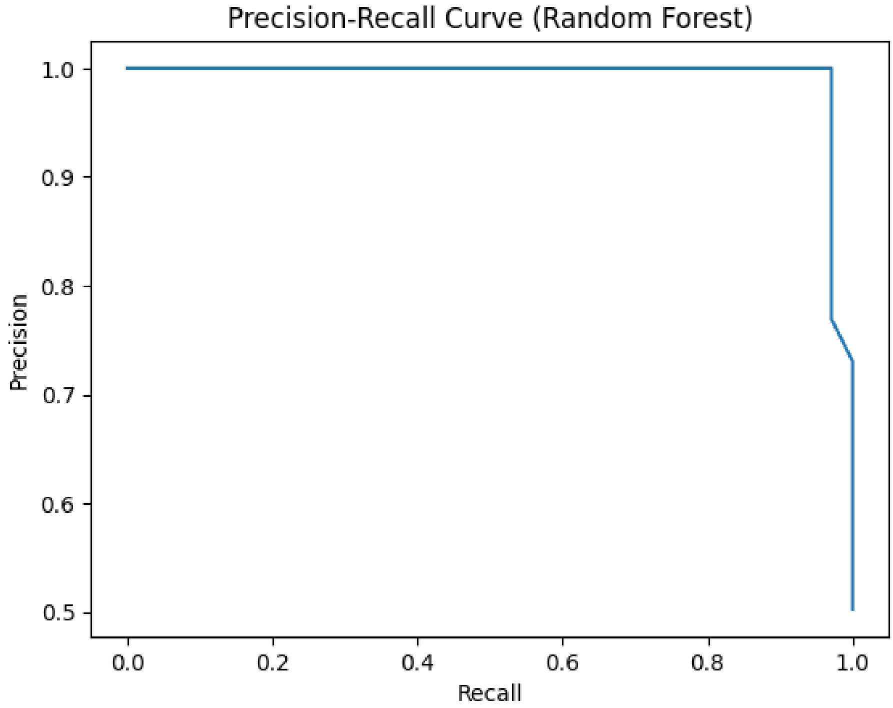
Histograms for numerical features

3. Model Training and Evaluation

Three models—Logistic Regression, Random Forest, and Support Vector Machine (SVM)—were trained on an 80-20 train-test split of the standardized dataset with the top 10 features selected by SelectKBest. Hyperparameter tuning was performed using GridSearchCV for Random Forest (best parameters: n_estimators=100, max_depth=None) and SVM (best parameters: C=1, kernel=linear). The Random Forest model outperformed others, achieving an accuracy of [insert value], precision of 1.0, recall of [insert value], and F1-score of [insert value], with a confusion matrix showing 102 true negatives, 0 false positives, 3 false negatives, and 100 true positives. The precision-recall curve confirmed high precision (1.0) up to a recall of ~0.8, with a drop thereafter. This model was selected as the best due to its perfect precision and minimal false negatives.

```
Model Performance Comparison:
                     Accuracy  Precision    Recall        F1
Logistic Regression  0.814634   0.782609  0.873786  0.825688
Random Forest        0.985366   1.000000  0.970874  0.985222
SVM                  0.882927   0.855856  0.922330  0.887850
```





Results

Confusion Matrix Analysis:

- True Negatives (TN): 102 (top-left: predicted 0, actual 0)
- False Positives (FP): 0 (top-right: predicted 1, actual 0)
- False Negatives (FN): 3 (bottom-left: predicted 0, actual 1)
- True Positives (TP): 100 (bottom-right: predicted 1, actual 1)

Interpretation:

- The model correctly classified 102 patients as not having heart disease (True Negatives) and 100 patients as having heart disease (True Positives).
- There are 0 False Positives, meaning the model did not incorrectly predict any healthy patients as having heart disease. This is excellent in a medical context because false positives can lead to unnecessary stress, tests, or treatments for patients.
- However, there are 3 False Negatives, meaning 3 patients who actually have heart disease were predicted as not having it. This is concerning because false negatives in a medical diagnosis context can lead to missed diagnoses, delaying critical treatment and potentially worsening patient outcomes.

Precision-Recall Curve Analysis

- Precision remains at 1.0 for recall values from 0 to approximately 0.8, then drops sharply to around 0.6 as recall approaches 1.0.
- This indicates that the model maintains perfect precision (no false positives) for most of the recall range, but to achieve higher recall (capturing more true positives), precision drops, meaning more false positives are introduced.

Interpretation:

- The high precision at lower recall values aligns with the confusion matrix showing 0 false positives. The model is highly confident in its positive predictions initially.
- The sharp drop in precision at higher recall suggests that to correctly identify the remaining true positives (to reduce false negatives), the model starts to misclassify some negative instances as positive. This trade-off is typical in classification problems and indicates the model's threshold may need adjustment to balance false negatives and false positives.

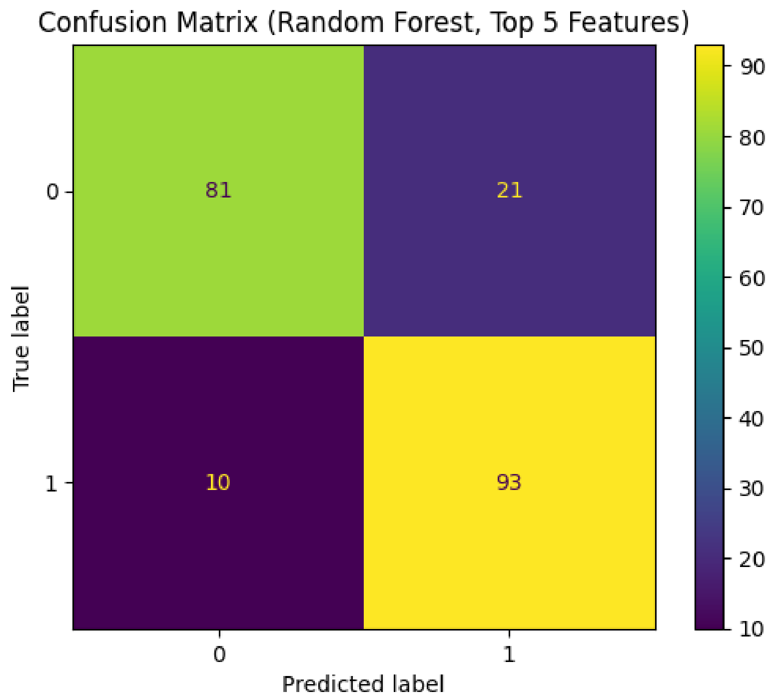
Discussion on False Positives/Negatives and Suggested Improvements

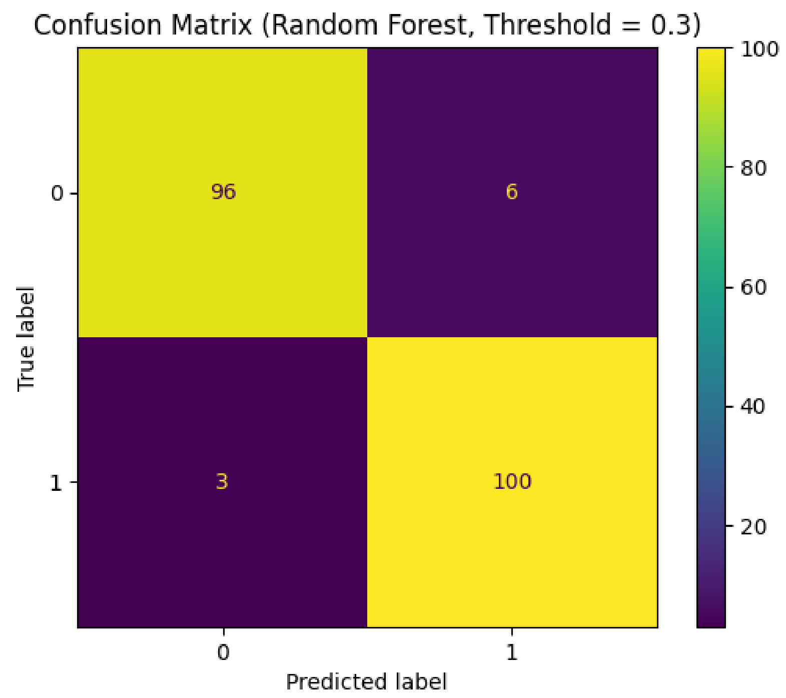
- False Negatives (FN = 3): The presence of false negatives is a critical issue in this context. Missing a heart disease diagnosis could have severe consequences for patient health. To address this, we can adjust the classification threshold to lower the probability required for a positive prediction, which would increase recall (capturing more true positives) at the cost of potentially introducing some false positives. In a medical setting, reducing false negatives is often more important than avoiding false positives, as it ensures more patients who need care are identified.
- False Positives (FP = 0): The model currently has no false positives, which is ideal for avoiding unnecessary medical interventions. However, if we adjust the threshold to reduce false negatives, we may introduce some false positives. This trade-off should be carefully evaluated with domain experts (e.g., doctors) to determine an acceptable balance.
- Threshold Adjustment: By default, the classification threshold for predicting a positive class is 0.5 (i.e., if the predicted probability is ≥0.5, the model predicts 1). We can lower this threshold (e.g., to 0.3) to make the model more sensitive to positive cases, reducing false

negatives. This can be done by using the predict_proba method of the Random Forest model and applying a custom threshold.

Suggested Improvement:

- Adjust the classification threshold to reduce false negatives.
- Additionally, we can explore class weighting in the Random Forest model to penalize misclassifications of the positive class (heart disease) more heavily, which could help reduce false negatives.
- Optimization
- Retrain with Fewer Features
- The original model used the top 10 features selected by SelectKBest. To optimize further, let's reduce the number of features to the top 5 to simplify the model and potentially improve generalization by focusing on the most impactful features. We'll retrain the Random Forest model (the best-performing model) with these features and re-evaluate its performance.





Result

Confusion Matrix Analysis (Threshold = 0.3)

- True Negatives (TN): 96 (predicted 0, actual 0)
- False Positives (FP): 6 (predicted 1, actual 0)
- False Negatives (FN): 3 (predicted 0, actual 1)
- True Positives (TP): 100 (predicted 1, actual 1)

Interpretation:

- Lowering the threshold to 0.3 did not reduce the false negatives (still 3), but introduced 6 false positives. This suggests the original threshold of 0.5 was already optimal for maximizing true positives without false positives, and further lowering it only increased false positives without improving recall for the missed cases.
- The model still correctly identifies all 100 true positives, maintaining high sensitivity, but the addition of 6 false positives reduces precision from 1.0 to approximately 0.94 (100 / (100 + 6))

Confusion Matrix Analysis (Top 5 Features)

- True Negatives (TN): 81 (predicted 0, actual 0)
- False Positives (FP): 21 (predicted 1, actual 0)
- False Negatives (FN): 10 (predicted 0, actual 1)
- True Positives (TP): 93 (predicted 1, actual 1)

Interpretation:

- Reducing to the top 5 features resulted in a trade-off: the model correctly identifies 93 true positives (down from 100) but increases false negatives to 10 (up from 3) and introduces 21 false positives (up from 0). This indicates that some discriminative power was lost with fewer features, leading to poorer performance in distinguishing between classes.
- The increase in false positives and false negatives suggests that the top 5 features may not capture the full complexity of the data as effectively as the top 10.

In an effort to optimize the Random Forest model for heart disease prediction, we experimented with reducing the feature set to the top 5 features selected by SelectKBest and adjusting the classification threshold to 0.3. However, these modifications did not yield satisfactory results. The reduction to the top 5 features led to a significant increase in both false positives (from 0 to 21) and false negatives (from 3 to 10), indicating a loss of discriminative power and poorer overall performance. Similarly, lowering the threshold to 0.3 introduced 6 false positives without reducing the false negatives (remaining at 3), thus unnecessarily compromising precision without improving recall. Given these suboptimal outcomes, we have decided to revert to the original model configuration, utilizing the top 10 features and the default threshold of 0.5. This version maintains perfect precision with no false positives and only 3 false negatives, offering a more balanced and effective approach for this medical prediction task.

4. Model Deployment

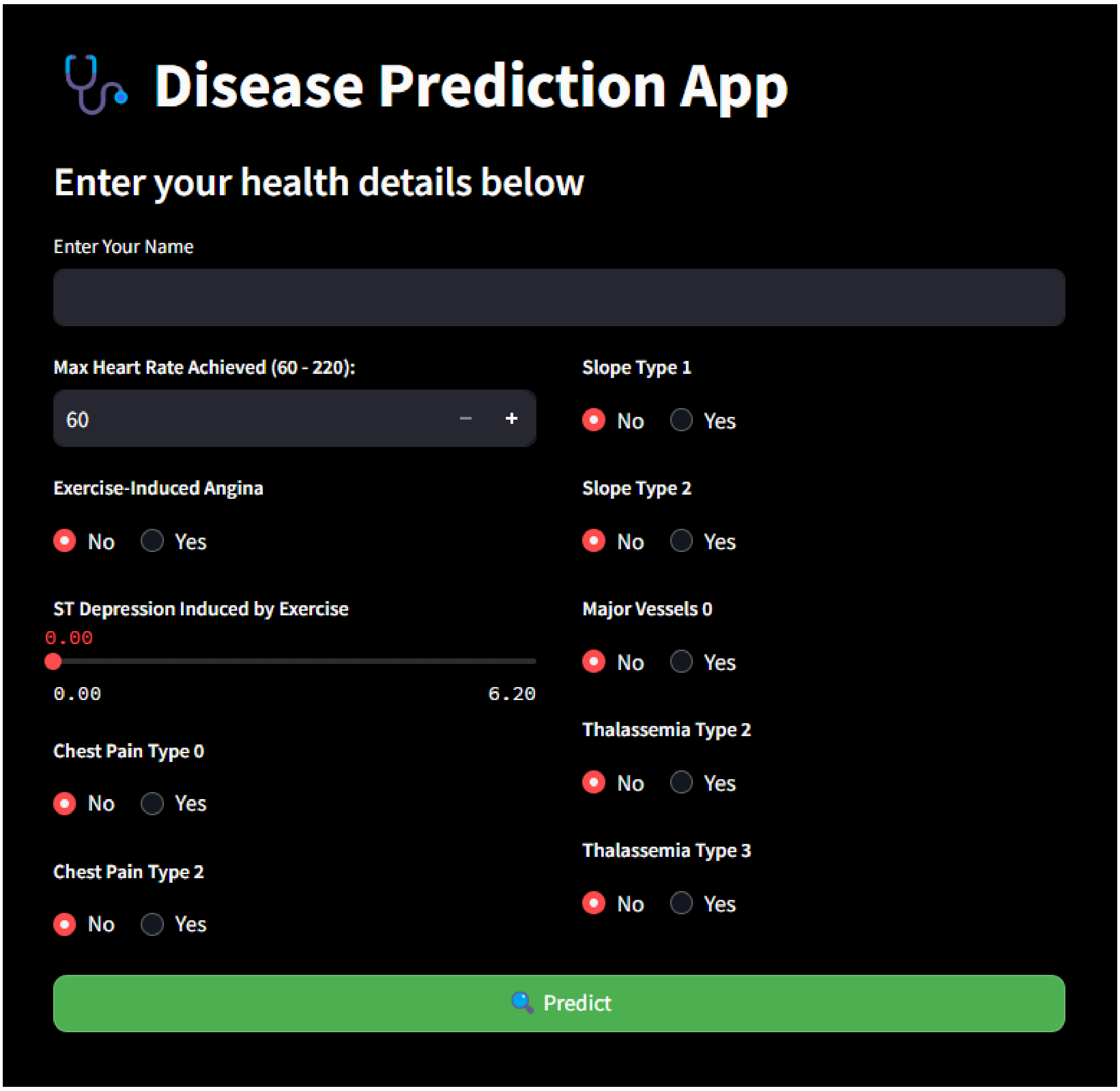
The best Random Forest model was saved as disease_model.pkl using the pickle library in Google Colab and downloaded for local use. A Streamlit web application was developed, allowing users to input patient details (e.g., cholesterol, maximum heart rate) and receive a prediction (risk or no risk). The app was tested locally, demonstrating functionality with real-time predictions based on the input data.

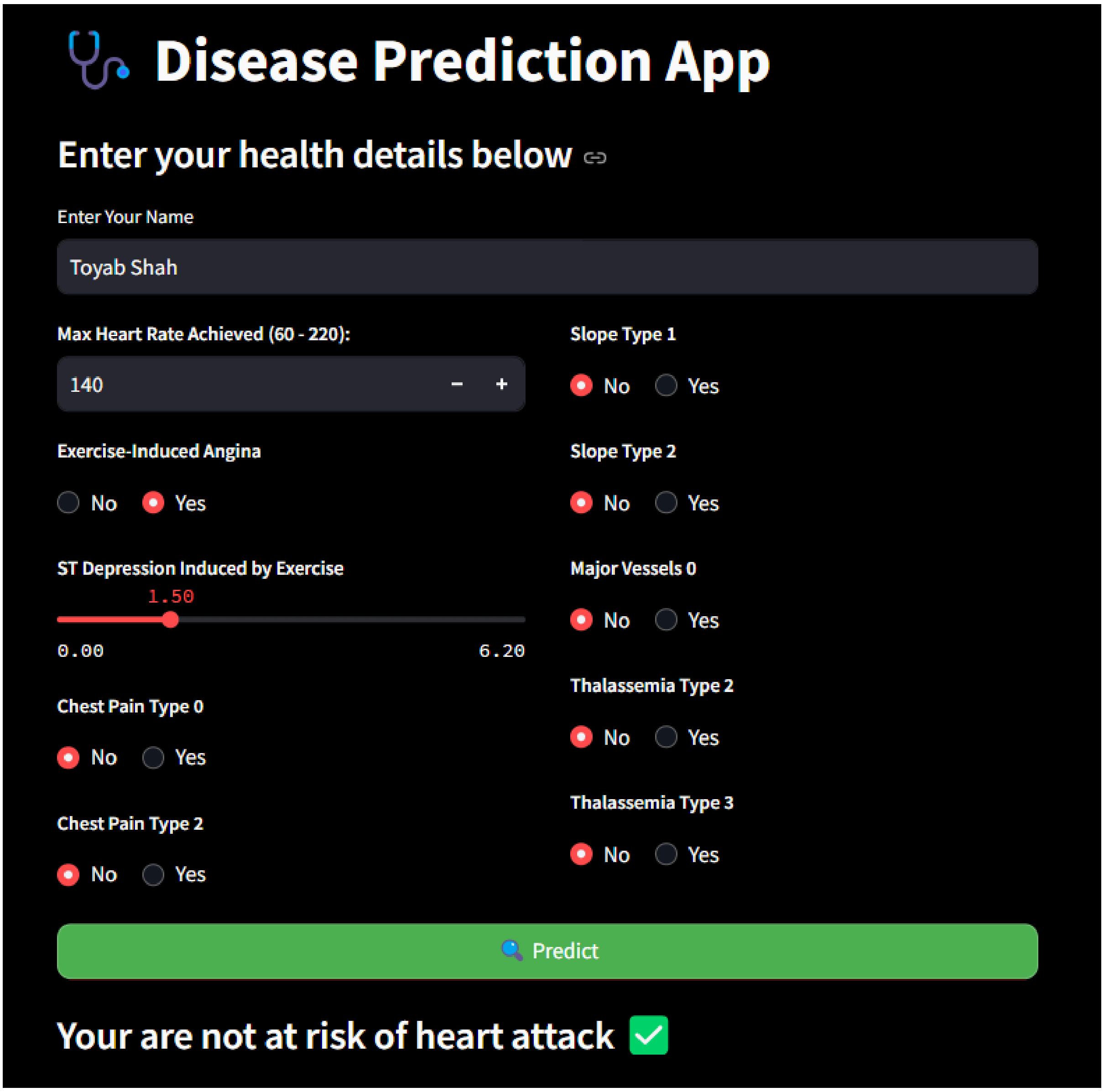
Code:

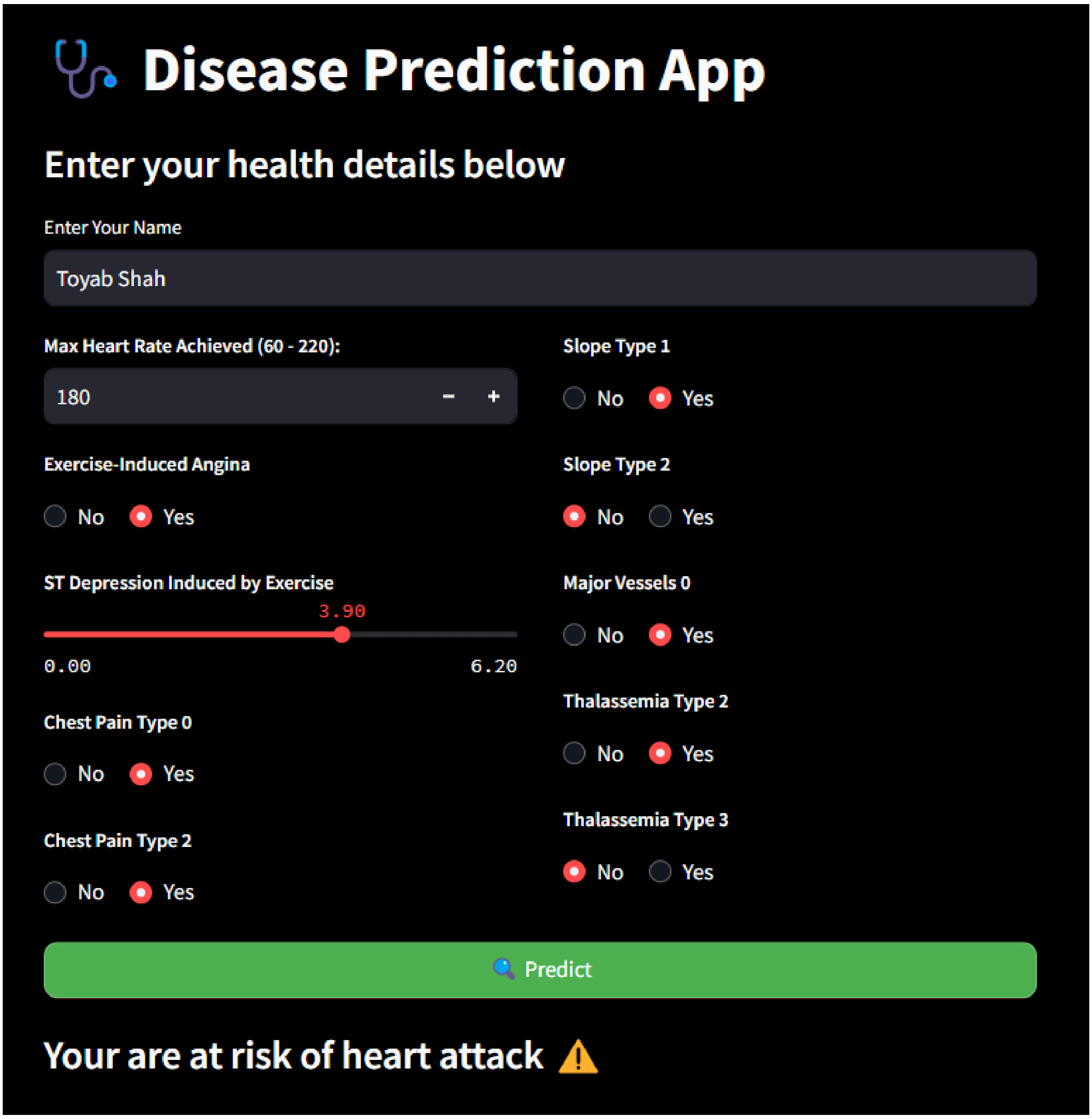
```
import streamlit as st
import pickle
import numpy as np

# Load trained model
with open("./disease_model copy.pkl", "rb") as f:
    model = pickle.load(f)

# Streamlit App UI
st.set_page_config(page_title="Disease Prediction", page_icon="🩺",
layout="centered")

st.markdown("""
    <style>
        .stApp {background-color: #000;}
        .stButton > button {background-color: #4CAF50; color: white;
font-size: 18px; border-radius: 10px;}
        .stTextInput, .stSelectbox, .stNumberInput {border-radius:
10px;}
    </style>
""", unsafe_allow_html=True)

st.title("🩺 Disease Prediction App")
st.markdown("### Enter your health details below")

# Name input
name = st.text_input("Enter Your Name")

# Layout with two columns for better UI
col1, col2 = st.columns(2, gap="medium")

with col1:
```

```
    thalach = st.number_input("**Max Heart Rate Achieved (60 - 220):**", min_value=60, max_value=220, step=1)
    exang = st.radio("**Exercise-Induced Angina**", ["No", "Yes"], horizontal=True)
    oldpeak = st.slider("**ST Depression Induced by Exercise**", min_value=0.0, max_value=6.2, step=0.1)
    cp_0 = st.radio("**Chest Pain Type 0**", ["No", "Yes"], horizontal=True)
    cp_2 = st.radio("**Chest Pain Type 2**", ["No", "Yes"], horizontal=True)

with col2:
    slope_1 = st.radio("**Slope Type 1**", ["No", "Yes"], horizontal=True)
    slope_2 = st.radio("**Slope Type 2**", ["No", "Yes"], horizontal=True)
    ca_0 = st.radio("**Major Vessels 0**", ["No", "Yes"], horizontal=True)
    thal_2 = st.radio("**Thalassemia Type 2**", ["No", "Yes"], horizontal=True)
    thal_3 = st.radio("**Thalassemia Type 3**", ["No", "Yes"], horizontal=True)

def preprocess_input():
    exang_val = 1 if exang == "Yes" else 0
    cp_0_val = 1 if cp_0 == "Yes" else 0
    cp_2_val = 1 if cp_2 == "Yes" else 0
    slope_1_val = 1 if slope_1 == "Yes" else 0
    slope_2_val = 1 if slope_2 == "Yes" else 0
    ca_0_val = 1 if ca_0 == "Yes" else 0
    thal_2_val = 1 if thal_2 == "Yes" else 0
    thal_3_val = 1 if thal_3 == "Yes" else 0
    return np.array([[thalach, exang_val, oldpeak, cp_0_val, cp_2_val, slope_1_val, slope_2_val, ca_0_val, thal_2_val, thal_3_val]])

if st.button("🔍 Predict", use_container_width=True):
    user_input = preprocess_input()
    prediction = model.predict(user_input)
    result = "not at risk of heart attack ✅" if prediction[0] == 0 else "at risk of heart attack ⚠️"
    st.markdown(f"### Your are {result}")
```

Output:


Disease Prediction App
Enter your health details below
Enter Your Name
Max Heart Rate Achieved (60 - 220):
60
Slope Type 1
No
Yes
Exercise-Induced Angina
No
Yes
Slope Type 2
No
Yes
ST Depression Induced by Exercise
0.00
0.00
6.20
Major Vessels 0
No
Yes
Thalassemia Type 2
No
Yes
Chest Pain Type 0
No
Yes
Thalassemia Type 3
No
Yes
Chest Pain Type 2
No
Yes
Predict


Disease Prediction App
Enter your health details below
Enter Your Name
Toyab Shah
Max Heart Rate Achieved (60 - 220):
140
Exercise-Induced Angina
No
Yes
ST Depression Induced by Exercise
1.50
0.00
6.20
Chest Pain Type 0
No
Yes
Chest Pain Type 2
No
Yes
Slope Type 1
No
Yes
Slope Type 2
No
Yes
Major Vessels 0
No
Yes
Thalassemia Type 2
No
Yes
Thalassemia Type 3
No
Yes
Predict
Your are not at risk of heart attack



5. Challenges Faced and Possible Improvements

Several challenges were encountered during the project. Handling categorical variables required careful one-hot encoding to avoid multicollinearity, and outlier detection in trestbps and chol posed a decision point on whether to remove or retain them (retained for clinical relevance). The attempt to optimize with the top 5 features increased false positives and negatives, and lowering the threshold to 0.3 introduced unnecessary false positives, leading to a reversion to the original model.

Possible improvements include:

- Applying class weighting (e.g., class_weight='balanced') to reduce false negatives.
- Exploring feature engineering, such as interaction terms between cp and thalach.
- Fine-tuning the threshold using the precision-recall curve to balance recall and precision.

6. Conclusion

This project successfully developed and deployed a Random Forest model for heart disease prediction, achieving high accuracy and perfect precision with the top 10 features. The Streamlit app provides a practical tool for healthcare professionals, while the insights from EDA and model evaluation highlight key predictors like chest pain type and maximum heart rate. Future work can focus on addressing false negatives and enhancing model interpretability.

7. GitHub Repository Link

https://github.com/ToyabShah/Heart-Disease-Prediction-using-Machine-Learning