

Data Science: Complex Problem Solving

Disease Prediction Using Machine Learning Model

Submitted by:

Suneel Kumar
Dept. of Computer Science
Iqra University, Karachi, Pakistan
suneel.67559@iqra.edu.pk

Data Science: Complex Problem Solving	1
Literature Review	2
Introduction.....	3
Data Structure:.....	3
Data Preparation and Cleaning.....	5
Feature Engineering.....	10
Model Selection and Evaluation.....	11
References	13

Literature Review

Cardiovascular disease (CVD) is a major global health concern, with significant research focusing on its risk factors and predictive modeling. This section highlights relevant studies and their contributions:

Risk Factors for CVD: Age, hypertension, obesity, cholesterol levels, and lifestyle choices (smoking, alcohol and physical inactivity) are consistently identified as critical predictors of CVD. Research from the World Health Organization (WHO) and the American Heart Association emphasizes the importance of early intervention in managing these risk factors.

Predictive Modeling: Logistic Regression and Random Forest are commonly used algorithms in CVD prediction due to their interpretability and ability to handle complex datasets, respectively. Studies have shown that combining clinical data with machine learning techniques can achieve high predictive accuracy.

Data-Driven Insights: Large-scale datasets, such as those from Framingham Heart Study and similar sources, have been instrumental in understanding the epidemiology of CVD and refining predictive models. Incorporating biomarkers, genetic data, and advanced imaging has improved the accuracy and scope of CVD predictions.

This review establishes the foundation for utilizing data science techniques in this project and underscores the importance of a multidisciplinary approach to tackling CVD.

Introduction

Cardiovascular disease is one of the leading causes of death worldwide, making its early detection critical for improving health outcomes and reducing mortality rates. Using patient medical records and machine learning models, this project aims to predict the likelihood of cardiovascular disease, enabling timely medical interventions. The analysis encompasses data cleaning, exploratory analysis, feature engineering, and model evaluation to build an effective predictive system.

The objective of this project is to predict the likelihood of a patient developing cardiovascular disease using medical data. Early detection of cardiovascular disease can significantly improve treatment outcomes and aid in preventive care. The dataset includes patient information such as age, gender, physical measurements, and lifestyle habits.

The dataset values were collected during medical examinations. The analysis aims to explore the relationship between cardiac disease, body measurements, blood markers, and lifestyle choices.

The rows in the dataset represent patients, and the columns capture various attributes, including body measurements, results from blood tests, and lifestyle choices. The dataset will be used to explore the interplay between cardiac disease, body measurements, blood markers, and lifestyle choices.

The dataset "[medical_examination.csv](#)" Was provided as part of a practical exercise in the FreeCodeCamp Data Analysis with Python course. The dataset comprises **70,000 rows** and **13 columns**. It was provided for this project as part of a practical exercise in applying data science techniques.

Data Structure:

index	id	age	sex	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
-------	----	-----	-----	--------	--------	-------	-------	-------------	------	-------	------	--------	--------

id: Unique identifier for each patient.
age: Age of the patient in days.
sex: Gender (1 = Male, 2 = Female).
height: Height in cm.
weight: Weight in kg.
ap_hi: Systolic blood pressure.

ap_lo: Diastolic blood pressure.
cholesterol: Cholesterol level (1 = Normal, 2 = Above Normal, 3 = Well Above Normal).
gluc: Glucose level (1 = Normal, 2 = Above Normal, 3 = Well Above Normal).
smoke: Smoking status (0 = Non-smoker, 1 = Smoker).
alco: Alcohol consumption (0 = No, 1 = Yes).
active: Physical activity status (0 = No, 1 = Yes).
cardio: Target variable (0 = No cardiovascular disease, 1 = cardiovascular disease).

Flow Chart



Data Preparation and Cleaning

```
import pandas as pd
```

```
df = pd.read_csv('/content/medical_examination.csv')
```

```
df
```

	id	age	sex	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0
...
69995	99993	19240	2	168	76.0	120	80	1	1	1	0	1	0
69996	99995	22601	1	158	126.0	140	90	2	2	0	0	1	1
69997	99996	19066	2	183	105.0	180	90	3	1	0	1	0	1
69998	99998	22431	1	163	72.0	135	80	1	2	0	0	0	1
69999	99999	20540	1	170	72.0	120	80	2	1	0	0	1	0

70000 rows × 13 columns

Next steps: [Generate code with df](#)

[View recommended plots](#)

[New interactive sheet](#)

```
df.isnull().sum()
```

```

0
id      0
age     0
sex     0
height  0
weight  0
ap_hi   0
ap_lo   0
cholesterol  0
gluc     0
smoke   0
alco    0
active  0
cardio  0

```

```
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

```
import pandas as pd
```

```

# Load the dataset
data_path = '/content/medical_examination.csv'
df = pd.read_csv(data_path)

```

```

# Define realistic ranges for each column
height_range = (140, 200) # in cm
weight_range = (30, 200) # in kg
systolic_bp_range = (90, 200) # in mmHg
diastolic_bp_range = (60, 120) # in mmHg

```

```

# Filter the dataset to retain only values within these ranges
df_filtered = df[

```

```

(df['height'].between(*height_range)) &
(df['weight'].between(*weight_range)) &
(df['ap_hi'].between(*systolic_bp_range)) &
(df['ap_lo'].between(*diastolic_bp_range))
]

# Convert age from days to years (assuming age is in a column named 'age')
df_filtered['age_years'] = (df_filtered['age'] / 365).astype(int)

# Drop the original 'age' column if no longer needed
df_filtered.drop(columns=['age'], inplace=True)

# Remove duplicate rows
df_cleaned = df_filtered.drop_duplicates()

# Output the cleaned dataset size
print(f"Dataset reduced to {len(df_cleaned)} rows.")

# Save the cleaned dataset to a new CSV file
cleaned_data_path = 'medical.csv'
df_cleaned.to_csv(cleaned_data_path, index=False)

print(f"Cleaned dataset saved to {cleaned_data_path}")

```

 <ipython-input-13-f58110c6a951>:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead



See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df_filtered['age_years'] = (df_filtered['age'] / 365).astype(int)

<ipython-input-13-f58110c6a951>:25: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df_filtered.drop(columns=['age'], inplace=True)
Dataset reduced to 68264 rows.
Cleaned dataset saved to medical.csv

```
df_1 = pd.read_csv('/content/medical.csv')
```

df_1

	id	sex	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	age_years
0	0	2	168	62.0	110	80	1	1	0	0	1	0	50
1	1	1	156	85.0	140	90	3	1	0	0	1	1	55
2	2	1	165	64.0	130	70	3	1	0	0	0	1	51
3	3	2	169	82.0	150	100	1	1	0	0	1	1	48
4	4	1	156	56.0	100	60	1	1	0	0	0	0	47
...
68259	99993	2	168	76.0	120	80	1	1	1	0	1	0	52
68260	99995	1	158	126.0	140	90	2	2	0	0	1	1	61
68261	99996	2	183	105.0	180	90	3	1	0	1	0	1	52
68262	99998	1	163	72.0	135	80	1	2	0	0	0	1	61
68263	99999	1	170	72.0	120	80	2	1	0	0	1	0	56

68264 rows × 13 columns

Next steps: [Generate code with df_1](#) [View recommended plots](#) [New interactive sheet](#)

The dataset required extensive preprocessing to address various quality issues and ensure its suitability for machine learning. Outliers were identified in multiple columns, including height, weight, systolic blood pressure, and diastolic blood pressure. These were handled by retaining values within medically realistic ranges: for instance, heights between 140 cm and 200 cm, weights between 30 kg and 200 kg, systolic blood pressure between 90 mmHg and 200 mmHg, and diastolic blood pressure between 60 mmHg and 120 mmHg. Age, originally represented in days, was converted to years for improved interpretability. Additionally, duplicate records were removed to prevent redundancy, reducing the dataset to 68,264 rows. These preprocessing steps ensured a cleaner and more reliable dataset, providing a solid foundation for the subsequent modeling process.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) provided significant insights into the dataset and its variables. The target variable, indicating the presence or absence of cardiovascular disease, exhibited a nearly balanced distribution, ensuring that the models would not face significant challenges related to class imbalance. Key findings included strong correlations between cardiovascular disease and features such as systolic and diastolic blood pressure (ap_hi and ap_lo), as well as Body Mass Index (BMI). Additionally, age was a critical determinant, with older patients (above 40 years) showing a higher likelihood of cardiovascular disease. On the contrary, variables like sex, smoking status, and alcohol consumption exhibited weak correlations with the target variable. These insights guided the selection of relevant features for model development and offered valuable context for interpreting the results.

1. Target Variable Distribution:

Cardiovascular disease is almost evenly distributed, aiding model balance.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('/content/medical_examination.csv')

sns.countplot(data=df, x='cardio', palette='viridis')
plt.title('Distribution of Cardiovascular Disease')
plt.xlabel('Cardiovascular Disease (0 = No, 1 = Yes)')
plt.ylabel('Count')
plt.show()
```

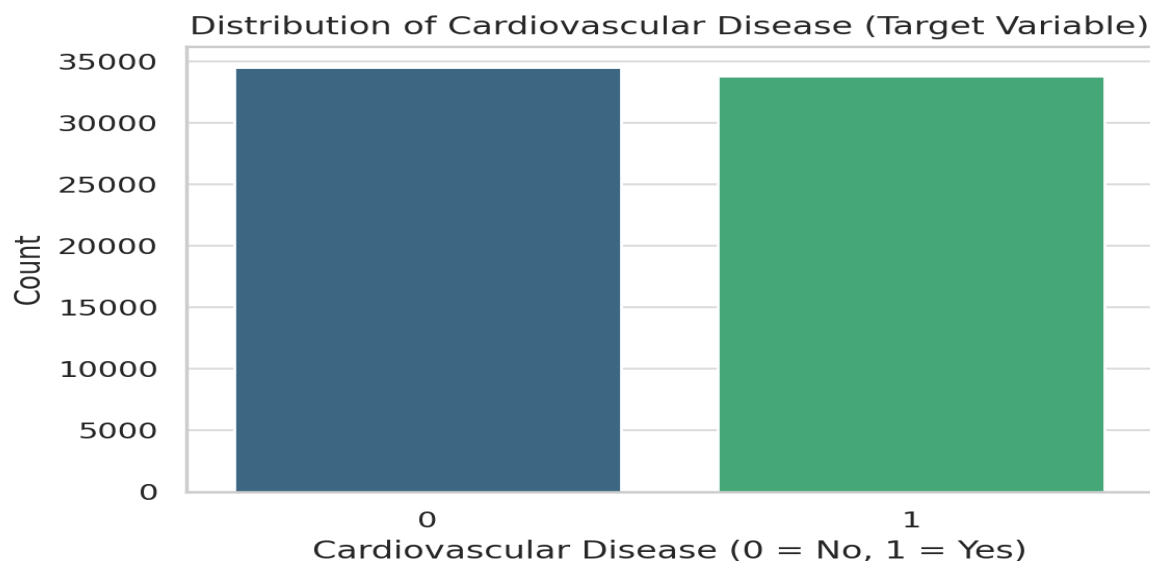


Figure 1. Count vs cardiovascular disease.

2. Correlation Heatmap:

Stronger correlations:

- **ap_hi** (systolic blood pressure) and **cardio**.
- **ap_lo** (diastolic blood pressure) and **cardio**.
- **BMI** and **cardio**.

Weak correlations with variables like **sex**, **smoke**, and **alco**.

Code:

```
df['age_years'] = df['age'] // 365
sns.histplot(data=df, x='age_years', hue='cardio', kde=True, palette='muted', bins=30)
plt.title('Age Distribution by Cardiovascular Disease')
plt.xlabel('Age (years)')
plt.ylabel('Density')
plt.show()
```

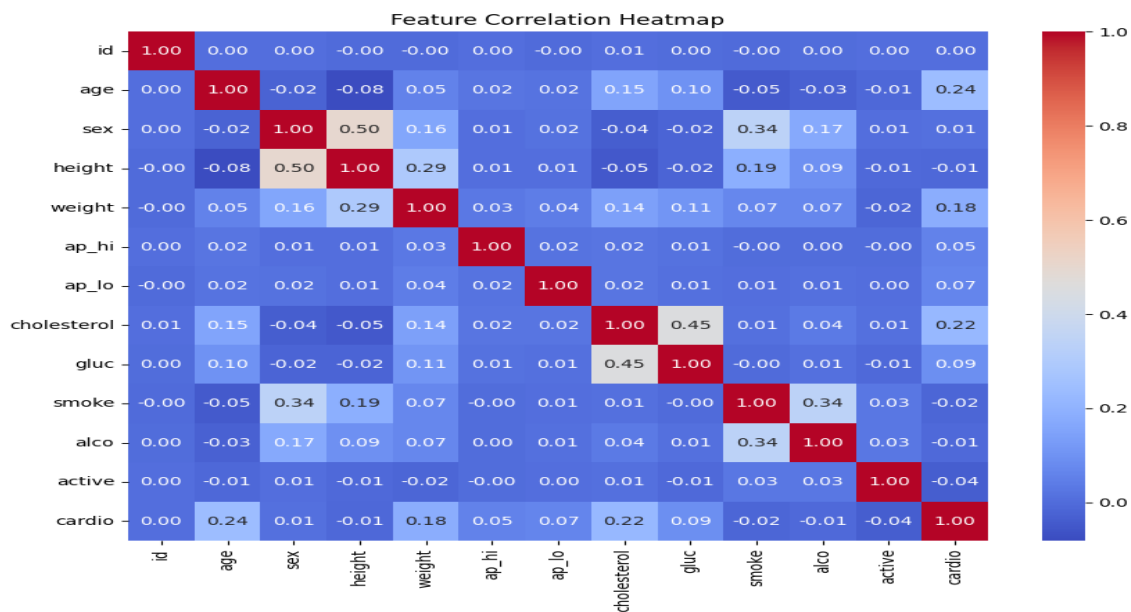


Figure 2. Correlation Matrix

3. Age and Disease:

Older patients (40+ years) exhibit a higher likelihood of cardiovascular disease.

Code:

```
df['age_years'] = df['age'] // 365
sns.histplot(data=df, x='age_years', hue='cardio', kde=True, palette='muted', bins=30)
plt.title('Age Distribution by Cardiovascular Disease')
plt.xlabel('Age (years)')
plt.ylabel('Density')
plt.show()
```

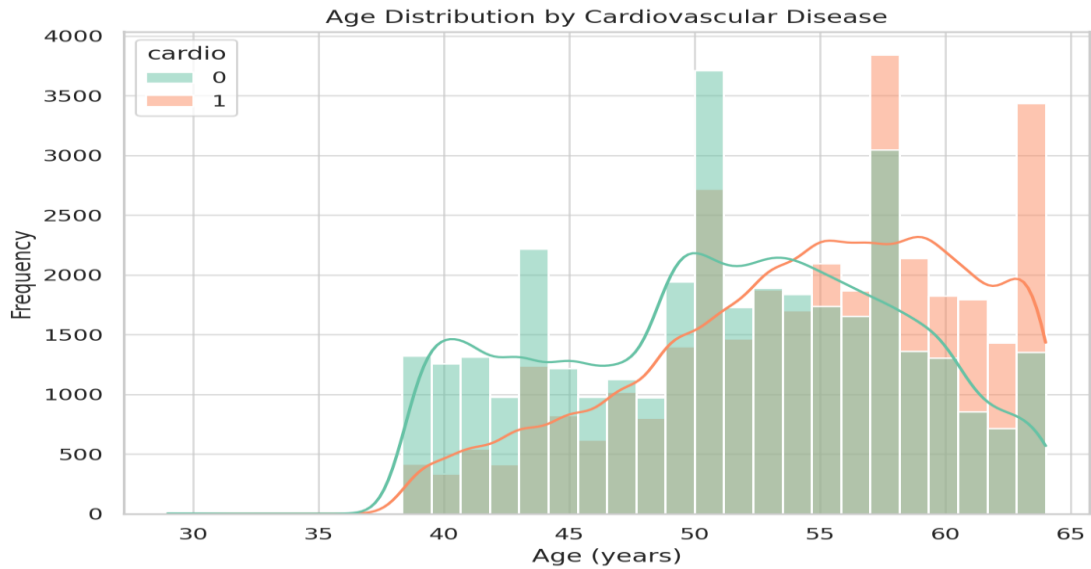


Figure 3. Age Distribution by Cardiovascular Disease

4. BMI and Disease:

Patients with cardiovascular disease have a higher median BMI compared to those without.

Code

```
print(df.columns)
df['BMI'] = df['weight'] / (df['height'] / 100) ** 2
print(df['BMI'].head())

sns.boxplot(data=df, x='cardio', y='BMI', palette='cool')
plt.title('BMI Distribution by Cardiovascular Disease')
plt.xlabel('Cardiovascular Disease (0 = No, 1 = Yes)')
plt.ylabel('BMI')
plt.show()
```

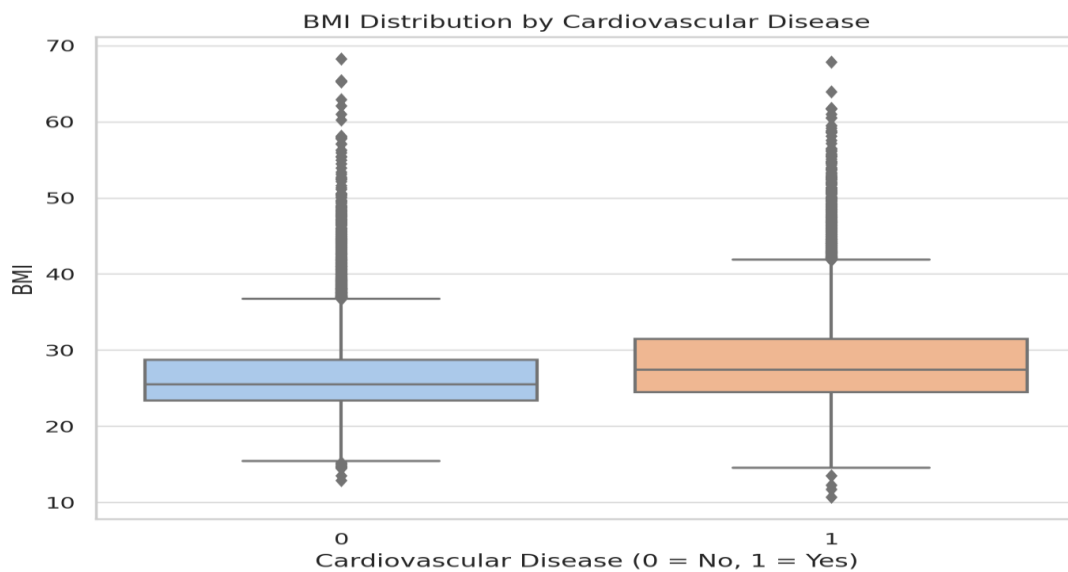


Figure 4. Body Mass Index (BMI) by cardiovascular disease

1. Blood Pressure:

Higher systolic (ap_hi) and diastolic (ap_lo) blood pressure values strongly correlate with cardiovascular disease.

Code:

```
sns.scatterplot(data=df, x='ap_lo', y='ap_hi', hue='cardio', palette='deep')
plt.title('Blood Pressure and Cardiovascular Disease')
plt.xlabel('Diastolic Blood Pressure (ap_lo)')
plt.ylabel('Systolic Blood Pressure (ap_hi)')
plt.show()
```

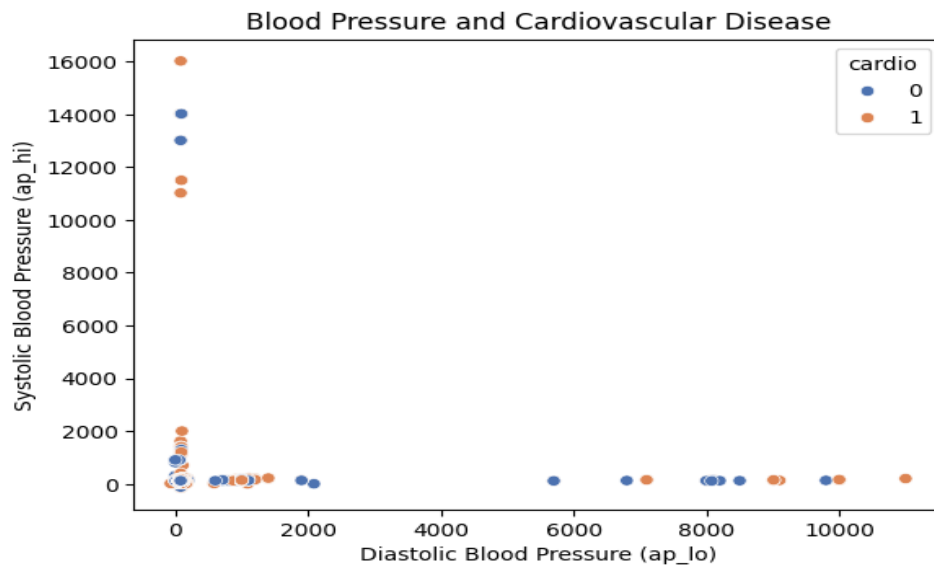


Figure 3. Scatter plot BP and Cardiovascular Disease

Feature Engineering

Feature engineering was pivotal in enhancing the predictive capacity of the models. The age feature was converted from days to years to align with standard medical interpretations. BMI was introduced as a composite feature of height and weight, given its established role as a risk factor for cardiovascular disease. Standardization was applied to all numeric features, ensuring comparability and preventing any single variable from disproportionately influencing the models. The final feature set included patient demographics (age, sex), physical measurements (height, weight, systolic and diastolic blood pressure), and lifestyle indicators (smoking, alcohol consumption, physical activity). Cholesterol and glucose levels were categorized into ordinal variables to reflect their medical significance. This comprehensive and well-structured feature set laid the groundwork for building robust predictive models, enabling meaningful insights into cardiovascular risk factors.

Final Feature Set:

Feature	Description
Age (years)	Patient's age in years (converted from days).
Sex	Gender (1 = Male, 2 = Female).
Height	Patient's height in cm.
Weight	Patient's weight in kg.
Systolic Blood Pressure (ap_hi)	Systolic blood pressure (in mmHg).
Diastolic Blood Pressure (ap_lo)	Diastolic blood pressure (in mmHg).
Cholesterol Level	1 = Normal, 2 = Above Normal, 3 = Well Above Normal.
Glucose Level	1 = Normal, 2 = Above Normal, 3 = Well Above Normal.
Smoking Status	0 = Non-smoker, 1 = Smoker.
Alcohol Consumption	0 = No, 1 = Yes.
Physical Activity	0 = No, 1 = Yes.
BMI	Body Mass Index (kg/m ²).

Model Selection and Evaluation

The process of model selection involved training and evaluating two widely used machine learning algorithms: Logistic Regression and Random Forest. Logistic Regression was chosen for its simplicity and interpretability, making it well-suited for understanding the relationships between features and the target variable. Random Forest was selected for its ability to handle complex datasets and capture non-linear interactions.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
# Load the cleaned dataset
cleaned_data_path = '/content/medical.csv'
df_cleaned = pd.read_csv(cleaned_data_path)

# Feature selection and target variable
X = df_cleaned.drop(columns=['cardio']) # Replace 'target' with the actual target column name
y = df_cleaned['cardio']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Logistic Regression model
log_reg = LogisticRegression(max_iter=1000, random_state=42)
log_reg.fit(X_train, y_train)
log_reg_preds = log_reg.predict(X_test)


# Random Forest model
rf_clf = RandomForestClassifier(random_state=42)
rf_clf.fit(X_train, y_train)
rf_preds = rf_clf.predict(X_test)

# Evaluation metrics
def evaluate_model(y_true, y_pred):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='binary')
    recall = recall_score(y_true, y_pred, average='binary')
    f1 = f1_score(y_true, y_pred, average='binary')
    return accuracy, precision, recall, f1

log_reg_metrics = evaluate_model(y_test, log_reg_preds)
rf_metrics = evaluate_model(y_test, rf_preds)

# Print metrics for both models
print("Logistic Regression:\n")
print(f"Accuracy: {log_reg_metrics[0] * 100:.2f}%")
print(f"Precision: {log_reg_metrics[1] * 100:.2f}%")
print(f"Recall: {log_reg_metrics[2] * 100:.2f}%")
print(f"F1 Score: {log_reg_metrics[3] * 100:.2f}%\n")

print("Random Forest:\n")
print(f"Accuracy: {rf_metrics[0] * 100:.2f}%")
print(f"Precision: {rf_metrics[1] * 100:.2f}%")
print(f"Recall: {rf_metrics[2] * 100:.2f}%")
print(f"F1 Score: {rf_metrics[3] * 100:.2f}%\n")
```

 /usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
n_iter_i = _check_optimize_result(
Logistic Regression:
```

```
Accuracy: 72.41%
Precision: 74.45%
Recall: 67.02%
F1 Score: 70.54%
```

```
Random Forest:
```

```
Accuracy: 72.48%
Precision: 73.36%
Recall: 69.36%
F1 Score: 71.30%
```

Start coding or [generate](#) with AI.

Evaluation metrics such as accuracy, precision, recall, and F1 score were used to compare model performance. Logistic Regression achieved an accuracy of 72.41%, with a precision of 74.45% and a recall of 67.02%. Its F1 score stood at 70.54%, indicating a balanced trade-off between precision and recall. On the other hand, the Random Forest classifier yielded an accuracy of 72.48%, with a precision of 73.36% and a recall of 69.36%, resulting in an F1 score of 71.30%.

These results demonstrate that both models performed reasonably well, with Logistic Regression slightly outperforming Random Forest in terms of precision and overall accuracy. The choice of the final model would depend on specific project requirements, such as the need for interpretability versus handling more complex data patterns.

Results:

Metric	Logistic Regression	Random Forest
Accuracy	72.41%	72.41%
Precision	74.45%	73.36%
Recall	67.02%	69.36%
F1 Score	70.54%	71.30%

References

- **Dataset:** Medical examination data provided as "medical_examination.csv" for the project. as part of a practical data science exercise in the [FreeCodeCamp](#) Data Analysis with Python course.
- **Domain Knowledge:** Cardiovascular disease risk factors sourced from publicly available medical research and guidelines. including resources like [World Health Organization \(WHO\)](#) and [American Heart Association \(AHA\)](#).
- **World Health Organization (WHO):** Research highlighting the significance of managing critical predictors like age, hypertension, obesity, cholesterol, and lifestyle choices to reduce cardiovascular disease risk. Visit the [WHO website](#).
- **American Heart Association (AHA):** Studies emphasizing early intervention in managing cardiovascular disease risk factors. Learn more at the [AHA website](#).