Heart Disease Prediction using Machine Learning Algorithms

**E.P.M.T.S. Ekanayake**

**IT21171406**

BSc (Hons) degree in Information Technology Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Sri Lanka

April 2025

Contents

[**1.** **Introduction** 3](#_Toc196796946)

[**2.** **Problem Statement** 3](#_Toc196796947)

[**3.** **Dataset Description** 3](#_Toc196796948)

[**4.** **Preprocessing** 4](#_Toc196796949)

[**5.** **Methodology** 5](#_Toc196796950)

[**6.** **Model Evaluation** 5](#_Toc196796951)

[**7.** **Results & Comparison** 5](#_Toc196796952)

[**7.1.** **Accuracy Comparison Graph** 6](#_Toc196796953)

[**8.** **Discussion & Future Work** 6](#_Toc196796954)

[**9.** **Conclusion** 7](#_Toc196796955)

[**10.** **Appendix** 7](#_Toc196796956)

# **Introduction**

Cardiovascular disease continues to be one of the leading causes of death globally, so early detection is essential to effective treatment. Current machine learning techniques can help doctors predict the chances of heart disease based on patient data.

In this project, we use machine learning classifiers to predict whether an individual is suffering from a cardiovascular disease or not based on medical and lifestyle conditions. We wish to compare the efficiency of Logistic Regression and Random Forest Classifier for this classification problem.

# **Problem Statement**

Healthcare systems worldwide are under increasing pressure to detect and treat cardiovascular disease (CVD) early. Traditional risk assessment procedures are manual and time-consuming. This project aims to create a predictive model based on machine learning that can be utilized to identify patients with a high likelihood of developing cardiovascular disease from common clinical data. This assists doctors in early detection and preventive care

# **Dataset Description**

* **Source:** [Kaggle - Sulianova]
* **Link:** <https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset>
* **Dataset Name:** Cleveland Heart Disease Dataset
* **Records:** 70,000
* **Features:** 11 input features + 1 target label
* **Target**: cardio (0 = no disease, 1 = has cardiovascular disease)

|  |  |
| --- | --- |
| Feature | Description |
| age | Patient’s age (years) |
| sex | 1= Female, 2 = Male |
| height | Height(cm) |
| weight | Weight(kg) |
| ap\_hi | Systolic blood pressure |
| ap\_lo | Diastolic blood pressure |
| cholesterol | |  | | --- | |  |  |  | | --- | | 1 = normal, 2 = above normal, 3 = well above normal | | |
| gluc | Glucose level (same scale as cholesterol) |
| smoke | 0 = no, 1 = yes |
| alco | 0 = no, 1 = yes |
| active | 0 = no, 1 = yes |

# **Preprocessing**

In this project, careful preprocessing steps were taken to prepare the data for machine learning models:

* Missing Values:

The dataset was checked for missing values across all features. No missing values were found, allowing us to proceed without imputation.

* Age Conversion:

The age feature was originally recorded in days. It was converted into years by dividing by 365 and rounding, making it easier to interpret.

* Target Class Visualization:

A count plot was created to verify the distribution of the target variable (cardio).

This verified that the dataset was quite balanced between patients who did and did not develop cardiovascular disease, which is critical for unbiased model training.

* Outlier Removal:

Blood pressure measurements were cleaned by removing unrealistic records:

Systolic blood pressure (ap\_hi) was bounded between 80 and 200 mmHg.

Diastolic blood pressure (ap\_lo) was bounded between 50 and 150 mmHg.

This eliminated incorrect data entries that would be harmful to model accuracy.

* Feature Scaling:

All numeric features were normalized using StandardScaler from scikit-learn.

This scaling discourages features with larger ranges (like age or blood pressure) from dominating smaller features (like glucose levels).

* Data Splitting:

The data was split into training and testing sets with an 80-20 split.

Stratified sampling was used to preserve the original class distribution in both sets for balanced model testing.

# **Methodology**

Logistic Regression

A linear classifier used for binary responses. It predicts the probability of a dependent variable given independent variables through a logistic function.

Advantages:

* Interpretable and effective for small datasets.
* Suitable for linearly separable data.

Random Forest Classifier

A method where multiple decision trees are combined. Each tree makes a prediction of a class, and the class predicted most often is the model's prediction. Advantages:

* Detects non-linear patterns.
* Eliminates overfitting typical of decision trees.

# **Model Evaluation**

Used the below metrics:

* Accuracy
* Precision
* Recall
* F1- Score
* Confusion Matrix

# **Results & Comparison**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-Score |
| Logistic Regression | |  | | --- | |  |   72.70% | 0.71 / 0.75 | 0.78 / 0.67 | 0.74 / 0.71 |
| Radom Forest | 71.99% | 0.71 / 0.73 | 0.74 / 0.70 | 0.73 / 0.71 |

## **Accuracy Comparison Graph**

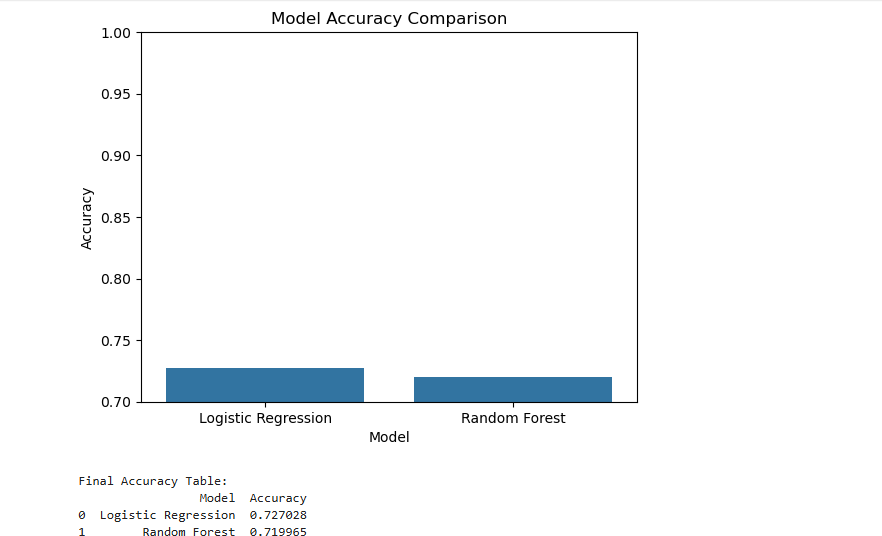


Figure 1: Comparison Graph

# **Discussion & Future Work**

**Analysis**

* Logistic Regression was satisfactory since there were clear linear trends in cholesterol and blood pressure.
* Random Forest provided better support for mixed data types and interactions but may require further fine-tuning.
* Feature importance analysis identified ap\_hi and cholesterol as significant predictors.

**Limitation**

* No hyperparameter tuning performed
* The dataset may contain some noisy or unreported medical variables

**Future Work**

* Apply hyperparameter optimization
* Test with additional models
* Feature selection to eliminate less informative variables

# **Conclusion**

This project shows one of the applications of machine learning to predict cardiovascular risk on real patient data. Logistic regression and random forest both provide useful information. Logistic regression provides interpretability and ease of interpretation, while random forest provides high accuracy and robustness. These models can support doctors in making early diagnoses.

# **Appendix**

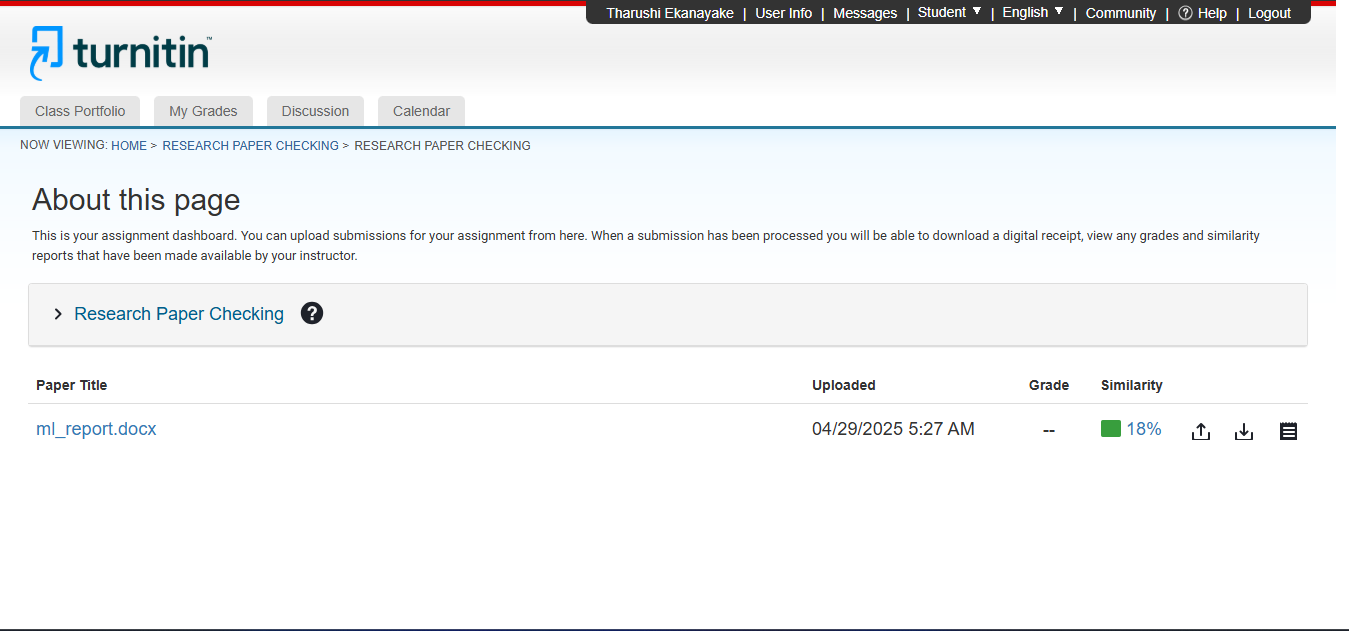


Figure : Plagiarism report

Source Code:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Load the Dataset

df = pd.read\_csv('cardio\_train.csv', sep=';')

print(df.columns)

# Check for missing values

print("Missing values per column:")

print(df.isnull().sum())

df['age'] = (df['age'] / 365).round().astype(int)

#Check Target Distribution

sns.countplot(x='cardio', data=df)

plt.title('Target Class Distribution')

plt.show()

df = df[(df['ap\_hi'] >= 80) & (df['ap\_hi'] <= 200)]

df = df[(df['ap\_lo'] >= 50) & (df['ap\_lo'] <= 150)]

X = df.drop(columns=['cardio'])

y = df['cardio']

#Feature Scaling

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Split into Train and Test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42, stratify=y)

# Model 1: Logistic Regression

lr = LogisticRegression(max\_iter=1000)

lr.fit(X\_train, y\_train)

y\_pred\_lr = lr.predict(X\_test)

#Model 2: Random Forest

rf = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf.fit(X\_train, y\_train)

y\_pred\_rf = rf.predict(X\_test)

def evaluate\_model(name, y\_true, y\_pred):

print(f"\n{name} Evaluation")

print("Accuracy:", accuracy\_score(y\_true, y\_pred))

print(classification\_report(y\_true, y\_pred))

sns.heatmap(confusion\_matrix(y\_true, y\_pred), annot=True, fmt='d', cmap='Blues')

plt.title(f"{name} - Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

evaluate\_model("Logistic Regression", y\_test, y\_pred\_lr)

evaluate\_model("Random Forest", y\_test, y\_pred\_rf)

# Compare Results

results = pd.DataFrame({

'Model': ['Logistic Regression', 'Random Forest'],

'Accuracy': [accuracy\_score(y\_test, y\_pred\_lr), accuracy\_score(y\_test, y\_pred\_rf)]

})

sns.barplot(x='Model', y='Accuracy', data=results)

plt.ylim(0.7, 1)

plt.title('Model Accuracy Comparison')

plt.show()

print("\nFinal Accuracy Table:")

print(results)