CARDIOVASCULAR RISK PREDICTION

REEBAL U

11271214017

# Import necessary libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier # Import RandomForestClassifier

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

# Load the dataset from an Excel file

data = pd.read\_excel('CVD\_cleaned (1).xlsx')

# Preprocessing

# Assume your dataset has features and a target variable. Adjust accordingly.

X = data.drop('General\_Health', axis=1) # Features

y = data['General\_Health'] # Target variable

# Drop non-numeric columns

X\_numeric = X.select\_dtypes(include=[np.number])

# Split the dataset into training and testing sets

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

# Handling missing values

X\_train.fillna(X\_train.mean(), inplace=True)

X\_test.fillna(X\_test.mean(), inplace=True)

# Standardize features by removing the mean and scaling to unit variance

scaler = StandardScaler()

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

X\_test\_scaled = scaler.transform(X\_test)

# Train a Random Forest classifier

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

model.fit(X\_train\_scaled, y\_train)

# Make predictions on the test set

predictions = model.predict(X\_test\_scaled)

# Evaluate the model

accuracy = accuracy\_score(y\_test, predictions)

print("Accuracy:", accuracy)

# Print classification report and confusion matrix

print("\nClassification Report:")

print(classification\_report(y\_test, predictions))

print("\nConfusion Matrix:")

print(confusion\_matrix(y\_test, predictions))

# Train a Random Forest classifier

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

model.fit(X\_train\_scaled, y\_train)

# Get feature importances

importances = model.feature\_importances\_

# Sort feature importances in descending order

indices = np.argsort(importances)[::-1]

# Plot feature importances

plt.figure(figsize=(10, 6))

plt.title("Feature Importances")

plt.bar(range(X\_train\_scaled.shape[1]), importances[indices],

color="b", align="center")

plt.xticks(range(X\_train\_scaled.shape[1]), X\_train.columns[indices], rotation=90)

plt.xlim([-1, X\_train\_scaled.shape[1]])

plt.xlabel('Features')

plt.ylabel('Importance')

plt.tight\_layout()

plt.show()