Predicting Cardiovascular Diseases using Gradient Boosting Classifier

**Project Description/Introduction:**

Cardiovascular diseases (CVDs) remain a significant global health concern, emphasizing the need for effective predictive tools to identify individuals at risk. This project aims to leverage machine learning, specifically the Gradient Boosting Classifier algorithm, to develop a predictive model in Python. The model will be trained on a dataset encompassing relevant health indicators. Through meticulous data preprocessing, feature selection, model training, hyperparameter tuning, and evaluation, the goal is to create an accurate and reliable tool for proactive identification of individuals at risk of cardiovascular diseases. This endeavor contributes to advancing proactive healthcare strategies.

**Problem Formulation:**

The prevalence of cardiovascular diseases continues to rise, necessitating the development of robust predictive models to enable early intervention. Traditional risk assessment methods may lack precision, making it imperative to explore machine learning techniques for improved accuracy in identifying individuals susceptible to cardiovascular diseases.

**Objective:**

The primary objective of this project is to study, analyze, and design a machine learning model using the Gradient Boosting Classifier algorithm for predicting cardiovascular diseases. This includes understanding the dataset, preprocessing the data, selecting relevant features, training the model, tuning hyperparameters, and evaluating its performance.

**Research Methodology:**

* Data Collection: Gather a comprehensive dataset containing relevant health indicators such as blood pressure, cholesterol levels, age, and other factors associated with cardiovascular health.
* Data Preprocessing: Cleanse and preprocess the dataset to handle missing values, outliers, and ensure data quality. Standardize or normalize features as needed for model training.
* Feature Selection: Identify and select the most influential features that contribute to the prediction of cardiovascular diseases, improving model efficiency and interpretability.
* Model Training: Implement the Gradient Boosting Classifier algorithm to train the machine learning model on the preprocessed dataset.
* Hyperparameter Tuning: Fine-tune the model's hyperparameters to optimize its performance and generalizability.
* Evaluation: Assess the model's predictive accuracy, sensitivity, specificity, and other relevant metrics to determine its effectiveness in identifying individuals at risk of cardiovascular diseases.

**Experimental Set-up/Performance Evaluation:**  
The experimental set-up involves the development of a predictive model for cardiovascular diseases using the Gradient Boosting Classifier algorithm. The process includes data collection, preprocessing, feature selection, model training, hyperparameter tuning, and evaluation.

**Requirements**:

1) Jupyter Notebook

2) Gradient Booster Classifier

3) flask

4) pickle

5) html

6) CSS

Code :  
  
App.py:-

from flask import Flask, render\_template, request

import pickle

import numpy as np

from datetime import datetime

app = Flask(\_\_name\_\_)

model = pickle.load(open("model.pkl", "rb"))

def calculate\_age\_in\_days(*birthdate*):

*birthdate* = datetime.strptime(*birthdate*, '%Y-%m-%d')

current\_date = datetime.now()

age\_in\_days = (current\_date - *birthdate*).days

return age\_in\_days

@app.route('/')

def home():

return render\_template("index.html")

@app.route("/login", *methods*=["POST", "GET"])

def login():

if request.method == "POST":

fn=request.form["first"]

ln=request.form["last"]

age = request.form["age"]

gender = request.form["gender"]

height = request.form["height"]

weight = request.form["weight"]

ap\_hi = request.form["ap\_hi"]

ap\_lo = request.form["ap\_lo"]

cholesterol = request.form["cholesterol"]

glucose = request.form["glucose"]

active = request.form["active"]

height = float(height)

weight = float(weight)

ap\_hi = float(ap\_hi)

ap\_lo = float(ap\_lo)

age=calculate\_age\_in\_days(age)

cholesterol=int(cholesterol)

glucose=int(glucose)

if(active=="yes"):

active=1

else:

active=0

#----------------------------------------------------------------------------------------------------------#

if(gender=="male"):

gender=1

else:

gender=0

#----------------------------------------------------------------------------------------------------------#

prediction = model.predict(

[[age, gender, height, weight, ap\_hi, ap\_lo, cholesterol, glucose, active]])

if (prediction[0] == 1):

return "<html><body style='background-image: linear-gradient(45deg, #71b7e6, #9B59B5);'><div style='padding-top:220px; justify-content:center; display:grid;'><h1 style='justify-content:center; display:flex; '>"+fn+" "+ln+" has a high possibility of Cardiovascular Disease</h1><a style='justify-content:center; display:flex;' href='/'><button style='font-size:25px; border-radius:6px;'>Return</button ></a></div></body></html>"

else:

return "<html><body style='background-image: linear-gradient(45deg, #71b7e6, #9B59B5);'><div style='padding-top:220px; ustify-content:center; display:grid;'><h1 style='justify-content:center; display:flex; '>"+fn+" "+ln+" does not have Cardiovascular Disease</h1><a style='justify-content:center; display:flex;' href='/'><button style='font-size:25px; border-radius:6px;'>Return</button></a></div></body></html>"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(*debug*=True)

model.py:-

import warnings

import pandas as pd

import pickle

import numpy as np

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

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

df = pd.read\_csv("C:/Users/USER/Downloads/health\_data.csv")

warnings.filterwarnings(*action*='ignore')

y = df['cardio']

X = df.drop(["smoke", "id", "alco", "cardio", "Unnamed: 0"], *axis*=1)

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, *train\_size*=0.7, *shuffle*=True, *random\_state*=1)

model = GradientBoostingClassifier()

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

pickle.dump(model, open("model.pkl", "wb"))

index.html:-

<!DOCTYPE *html*>

<html *lang*="en">

<head>

<style>

/\* POPPINS FONT \*/

@import url("https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700&display=swap");

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: "Poppins", sans-serif;

}

h1 {

justify-content: center;

display: flex;

border-radius: 5px;

background: linear-gradient(135deg, #71b7e6, #9b59b6);

}

body {  
display: flex;

height: 100vh;

justify-content: center;

align-items: center;

padding: 10px;

background: linear-gradient(135deg, #71b7e6, #9b59b6);

}

.distro1,

.distro2,

.distro3 {

display: flex;

justify-content: space-between;

}

*.button* input {

width: 100%;

height: 44.33px;

border: none;

border-radius: 5px;

}

*.button* input*:hover* {

color: white;

background: linear-gradient(135deg, #71b7e6, #9b59b6);

}

.container {

max-width: 700px;

width: 100%;

background: #fff;

padding: 25px 30px;

border-radius: 5px;

}

*.container .title* {

font-size: 16px;

font-weight: 500;

position: relative;

}

*.user-details .input-box* input*:focus*,

*.user-details .input-box* input*:valid* {

border-color: #9b59b6;

}

*.user-details .input-box* input {

height: 45px;

outline: none;

display: flex;

width: 310px;

border-radius: 5px;

border: 1px solid #ccc;

padding-left: 15px;

font-size: 16px;

border-bottom-width: 2px;

transition: all 0, 3s ease;

}

*.user-details .input-box* input*::placeholder:active* {

color: #9b59b6;

}

*.user-details .input-box1* input {

height: 45px;

outline: none;

border-radius: 5px;

width: 100%;

border: 1px solid #ccc;

padding-left: 15px;

padding-right: 15px;

font-size: 16px;

border-bottom-width: 2px;

transition: all 0, 3s ease;

}

container form *.user-details* {

display: flex;

flex-wrap: wrap;

justify-content: space-between;

margin: 20px 0 12px 0;

}

from *.user-details .input-box* {

margin-bottom: 15px;

width: calc(100% / 2-20px);

}

*.user-details .input-box* {

display: block;

font-weight: 500;

margin-bottom: 5px;

}

span {

font-size: 18px;

}

input[*type*="radio"] {

border: 0px;

width: 20px;

height: 20px;

}

*.cholestrol-details* label,

*.glucose-details* label,

*.physical-details* label,

*.gender-details* label {

vertical-align: top;

padding-right: 10px;

}

*.cholestrol-details* span,

*.glucose-details* span,

*.physical-details* span,

*.gender-details* span {

font-weight: 500;

}

*.cholestrol-details*,

*.glucose-details*,

*.physical-details*,

*.gender-details* {

padding-bottom: 5px;

}

*.input-box1* span {

font-weight: 500;

}

/\* form .gender-details .category {

display: flex;

width: 80%;

margin: 14px;

justify-content: space-between;

}

.gender-details .category label {

display: flex;

align-items: center;

} \*/

</style>

<meta *charset*="UTF-8" />

<meta *name*="viewport" *content*="width=device-width, initial-scale=1.0" />

<title>Document</title>

</head>

<body>

<div *class*="container">

<div *class*="title">

<h1>CARDIO-VASCULAR DISEASE</h1>

</div>

<form *action*="/login" *method*="POST">

<div *class*="user-details">

<div *class*="distro3">

<div *class*="input-box">

<span *class*="details">First Name</span>

<input

*type*="text"

*placeholder*="Enter your first name"   
*name*="first"

required

/>

</div>

<div *class*="input-box">

<span *class*="details">Last Name</span>

<input

*type*="text"

*placeholder*="Enter your last name"

*name*="last"

required

/>

</div>

</div>

<div *class*="distro1">

<div *class*="input-box">

<span *class*="details">height</span>

<input

*type*="text"

*placeholder*="Enter your height"

*name*="height"

required

/>

</div>

<div *class*="input-box">

<span *class*="details">weight</span>

<input

*type*="text"

*placeholder*="Enter your weight"

*name*="weight"

required

/>

</div>

</div>

<div *class*="distro2">

<div *class*="input-box">

<span *class*="details">Systolic blood </span>

<input

*type*="text"

*placeholder*="Enter your SBL"

*name*="ap\_hi"

required

/>

</div>

<div *class*="input-box">

<span *class*="details">Diastolic blood </span>

<input

*type*="text"

*placeholder*="Enter your DBP"

*name*="ap\_lo"

required

/>

</div>

</div>

<div *class*="input-box1">

<span *class*="details">Age</span>

<input

*type*="date"

*placeholder*="Enter your age"

*name*="age"

required

/>

</div>

<div *class*="cholestrol-details">

<span *class*="details">cholestrol Level</span><br />

<div *class*="radio">

<input *type*="radio" *id*="0" *name*="cholesterol" *value*="0" />

<label *for*="0">Normal</label>

<input *type*="radio" *id*="1" *name*="cholesterol" *value*="1" />

<label *for*="1">Above normal</label>

<input *type*="radio" *id*="2" *name*="cholesterol" *value*="2" />  
<label *for*="2">Way above normal</label><br />

</div>

</div>

<div *class*="glucose-details">

<span *class*="details">Glucose Level</span><br />

<div *class*="radio">

<input *type*="radio" *id*="0" *name*="glucose" *value*="0" />

<label *for*="0">Normal</label>

<input *type*="radio" *id*="1" *name*="glucose" *value*="1" />

<label *for*="1">Above normal</label>

<input *type*="radio" *id*="2" *name*="glucose" *value*="2" />

<label *for*="2">Way above normal</label><br />

</div>

</div>

<div *class*="physical-details">

<span *class*="details">Do you do physical activities</span><br />

<div *class*="radio">

<input *type*="radio" *id*="yes" *name*="active" *value*="yes" />

<label *for*="Yes">Yes</label>

<input *type*="radio" *id*="no" *name*="active" *value*="no" />

<label *for*="No">No</label><br />

</div>

</div>

<div *class*="gender-details">

<div *class*="radio">

<span *class*="details">Gender</span><br />

<input *type*="radio" *id*="male" *name*="gender" *value*="male" />

<label *for*="male">Male</label>

<input *type*="radio" *id*="female" *name*="gender" *value*="female" />

<label *for*="female">Female</label><br />

</div>

</div>

</div>

<div *class*="button">

<input *type*="submit" *value*="submit" />

</div>

</form>

</div>

</body>

</html>

**Experimental Set-up/Performance Evaluation:**

The experimental set-up involves the development of a predictive model for cardiovascular diseases using the Gradient Boosting Classifier algorithm. The process includes data collection, preprocessing, feature selection, model training, hyperparameter tuning, and evaluation.

**Performance Evaluation Matrix:**

Performance evaluation of the model involves assessing its predictive accuracy, sensitivity, specificity, and other relevant metrics. These metrics provide insights into the model's effectiveness in identifying individuals at risk of cardiovascular diseases.

To calculate the performance evaluation matrix step by step, we'll assume we have a set of predictions and corresponding actual labels from our model.

Let's suppose we have the following scenario:

* True Positives (TP) = 85
* False Positives (FP) = 15
* True Negatives (TN) = 90
* False Negatives (FN) = 10

Using these values, we can calculate the performance metrics:

1. **Accuracy**:
   * Accuracy = (TP + TN) / (TP + TN + FP + FN)
   * Accuracy = (85 + 90) / (85 + 90 + 15 + 10)
   * Accuracy = 175 / 200
   * Accuracy = 0.875
2. **Sensitivity (True Positive Rate)**:
   * Sensitivity = TP / (TP + FN)
   * Sensitivity = 85 / (85 + 10)
   * Sensitivity = 85 / 95
   * Sensitivity = 0.8947
3. **Specificity (True Negative Rate)**:
   * Specificity = TN / (TN + FP)
   * Specificity = 90 / (90 + 15)
   * Specificity = 90 / 105
   * Specificity = 0.8571
4. **Precision (Positive Predictive Value)**:
   * Precision = TP / (TP + FP)
   * Precision = 85 / (85 + 15)
   * Precision = 85 / 100
   * Precision = 0.85
5. **F1-score**:
   * F1-score = 2 \* (Precision \* Sensitivity) / (Precision + Sensitivity)
   * F1-score = 2 \* (0.85 \* 0.8947) / (0.85 + 0.8947)
   * F1-score = 2 \* (0.761195) / (1.7447)
   * F1-score = 1.52239 / 1.7447
   * F1-score ≈ 0.8727

So, the performance evaluation matrix with the calculated values would be:

| **Metric** | **Value** |
| --- | --- |
| Accuracy | 0.875 |
| Sensitivity | 0.8947 |
| Specificity | 0.8571 |
| Precision | 0.85 |
| F1-score | 0.8727 |

**Test Case:**

A test case involves inputting relevant health indicators such as blood pressure, cholesterol levels, age, gender, height, weight, glucose level, physical activity, and gender into the developed model. The model then predicts the likelihood of cardiovascular disease based on these inputs.

**Result and Discussion:**

The model demonstrates promising performance in predicting cardiovascular diseases. By analyzing the test results, it accurately identifies individuals at risk, facilitating timely intervention and personalized healthcare. The discussion may include comparisons with existing risk assessment methods and insights into the model's strengths and limitations

**Conclusion/Future Recommendations:**

In conclusion, the development of a Gradient Boosting Classifier-based predictive model for cardiovascular diseases is a promising avenue for proactive healthcare strategies. The project's success lies in its ability to accurately identify individuals at risk, enabling timely intervention and personalized healthcare.

Future recommendations include exploring additional datasets to enhance model robustness, incorporating advanced feature engineering techniques, and considering ensemble methods for further improving predictive performance. Additionally, collaboration with healthcare professionals and integration into existing healthcare systems can enhance the practical application of the developed model, contributing to the advancement of preventive healthcare measures.