**A REPORT**

**ON**

**Heart Attack Risk Prediction**

## **By**

Name of the student Registration No.

Shaik Ansar Subhani AP22110010088

***Machine Learning Project***

5th Semester

### AT



#### SRM UNIVERSITY, AP

**(Nov, 2024)**

* ABSTRACT:

This project focuses on predicting the risk of heart attacks using a machine learning model. By analyzing patient-specific attributes like age, blood pressure, cholesterol levels, and lifestyle factors, the system evaluates whether a person is at high or low risk of a heart attack. The model is implemented using the Random Forest Classifier in Python, leveraging a preprocessed dataset to train and test the system. The results aim to assist in preventive healthcare.

* INTRODUCTION:

Heart attacks are one of the leading causes of death worldwide. Early identification of individuals at risk can save lives through preventive measures. This project aims to create a machine learning-based system to predict heart attack risk using patient data. The Random Forest algorithm is utilized to analyze features such as age, cholesterol, and blood pressure to classify patients into high or low-risk categories.

* DATA-SET DESCRIPTION:
* SOURCE: <https://www.kaggle.com/datasets/iamsouravbanerjee/heart-attack-prediction-dataset/data>
* **Key Features:**

Age, Sex, Cholesterol, Blood Pressure, Heart Rate, Diabetes, Smoking, BMI, and other lifestyle and health attributes.

* **Target Variable:** Heart Attack Risk (Binary: 1 = High Risk, 0 = Low Risk).
* **Preprocessing Steps:**

1. Dropped unnecessary columns: Patient ID, Country, Continent, Hemisphere, and Income.
2. Encoded categorical variables like Sex and Diet using Label Encoding.
3. Converted Blood Pressure values (e.g., "120/80") to averages.
4. Removed rows with missing or invalid data.

* METHODOLOGY
* **Model Used:** Random Forest Classifier, a tree-based ensemble learning algorithm that improves classification accuracy by combining multiple decision trees.
* **Steps:**

1. **Data Cleaning and Preprocessing:** Dropping irrelevant columns, encoding non-numeric values, and handling missing data.
2. **Feature Selection:** Selected features based on relevance to heart attack risk.
3. **Data Splitting:** Used an 80:20 split for training and testing datasets.
4. **Model Training:** Trained the Random Forest Classifier using the preprocessed training data.
5. **Prediction Function:** A user-input-based function takes new patient details and predicts their risk using the trained model.

* IMPLEMENTATION:

**Code Summary:**

* **Data Preprocessing:** Dropped unnecessary columns and encoded categorical data.
* **Training the Model:** Split data into training and testing sets and trained the Random Forest Classifier.
* **Prediction Function:** Accepts inputs like age, cholesterol, blood pressure, and lifestyle factors. Processes the inputs and uses the trained model to predict risk.
* Key Snippets:

Encoding blood pressure as an average

A computer screen with text and numbers

Description automatically generated

Prediction function:

A black screen with white text

Description automatically generated

* RESULT:
* **Model Performance:** Evaluated using the test dataset. Achieved [specify accuracy, e.g., 85% accuracy].
* **Example Prediction:**
* **Input:** Age = 45, Cholesterol = 220, Blood Pressure = "130/85", Smoking = 1
* **Output:** High Risk of Heart Attack.

The prediction function successfully classified patients into high or low-risk categories based on the inputs.

* DISCUSSION:
* **Strengths:**
* Random Forest's ability to handle diverse datasets and avoid overfitting.
* Comprehensive feature set representing health and lifestyle factors.
* **Limitations:**
* Dataset size and quality influence model performance.
* Assumes data preprocessing is accurate, e.g., valid blood pressure inputs.
* **Improvements:**
* Add more diverse patient data to improve model generalization.
* Test with other algorithms like Gradient Boosting for comparison.
* CONCLUSION:

This project successfully demonstrates the use of machine learning to predict heart attack risk based on health and lifestyle features. The Random Forest model provides reliable predictions and can potentially assist in preventive healthcare by identifying high-risk individuals. Future work could include integrating real-time patient monitoring data for improved prediction accuracy.

APPENDIX:

FULL CODE:

import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

dataset\_path = r'C:\Users\ansar\Downloads\archive (5)\heart\_attack\_prediction\_dataset.csv'

data = pd.read\_csv(dataset\_path)

# Drop unnecessary columns

data.drop(['Patient ID', 'Country', 'Continent', 'Hemisphere', 'Income'], axis=1, inplace=True)

# Encode non-numeric columns

non\_numeric\_columns = data.select\_dtypes(include=['object']).columns

for column in non\_numeric\_columns:

if column == 'Blood Pressure': # Handle separately if needed

# Convert "120/80" to average BP (120 + 80) / 2

data['Blood Pressure'] = data['Blood Pressure'].apply(

lambda x: (int(x.split('/')[0]) + int(x.split('/')[1])) / 2

if '/' in str(x) else None

)

else:

# Encode other non-numeric columns

le = LabelEncoder()

data[column] = le.fit\_transform(data[column].astype(str))

# Drop rows with missing or invalid data

data.dropna(inplace=True)

# Separate features and target variable

X = data.drop('Heart Attack Risk', axis=1)

y = data['Heart Attack Risk']

# Split dataset

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

# Train Random Forest Model

rf\_model = RandomForestClassifier(random\_state=42)

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

# Function to predict heart attack risk based on user input

def predict\_heart\_attack():

print("\nEnter the patient's details:")

age = int(input("Age: "))

sex = input("Sex (Male/Female): ")

cholesterol = int(input("Cholesterol: "))

blood\_pressure = input("Blood Pressure (e.g., 120/80): ")

heart\_rate = int(input("Heart Rate: "))

diabetes = int(input("Diabetes (0 = No, 1 = Yes): "))

family\_history = int(input("Family History of Heart Disease (0 = No, 1 = Yes): "))

smoking = int(input("Smoking (0 = No, 1 = Yes): "))

obesity = int(input("Obesity (0 = No, 1 = Yes): "))

alcohol\_consumption = int(input("Alcohol Consumption (0 = No, 1 = Yes): "))

exercise\_hours = float(input("Exercise Hours Per Week: "))

diet = input("Diet (Healthy/Average/Unhealthy): ")

previous\_heart\_problems = int(input("Previous Heart Problems (0 = No, 1 = Yes): "))

medication\_use = int(input("Medication Use (0 = No, 1 = Yes): "))

stress\_level = int(input("Stress Level (0 = Low, 1 = High): "))

sedentary\_hours = float(input("Sedentary Hours Per Day: "))

bmi = float(input("BMI: "))

triglycerides = int(input("Triglycerides: "))

physical\_activity\_days = int(input("Physical Activity Days Per Week: "))

sleep\_hours = int(input("Sleep Hours Per Day: "))

# Preprocess the input

blood\_pressure\_avg = (int(blood\_pressure.split('/')[0]) + int(blood\_pressure.split('/')[1])) / 2

diet\_encoded = {'Healthy': 0, 'Average': 1, 'Unhealthy': 2}.get(diet, 1) # Default to 'Average'

# Create a DataFrame for the input

input\_data = pd.DataFrame({

'Age': [age],

'Sex': [1 if sex.lower() == 'male' else 0],

'Cholesterol': [cholesterol],

'Blood Pressure': [blood\_pressure\_avg],

'Heart Rate': [heart\_rate],

'Diabetes': [diabetes],

'Family History': [family\_history],

'Smoking': [smoking],

'Obesity': [obesity],

'Alcohol Consumption': [alcohol\_consumption],

'Exercise Hours Per Week': [exercise\_hours],

'Diet': [diet\_encoded],

'Previous Heart Problems': [previous\_heart\_problems],

'Medication Use': [medication\_use],

'Stress Level': [stress\_level],

'Sedentary Hours Per Day': [sedentary\_hours],

'BMI': [bmi],

'Triglycerides': [triglycerides],

'Physical Activity Days Per Week': [physical\_activity\_days],

'Sleep Hours Per Day': [sleep\_hours]

})

# Predict

prediction = rf\_model.predict(input\_data)

print("\nPrediction: ", "High Risk of Heart Attack" if prediction[0] == 1 else "Low Risk of Heart Attack")

# Function to visualize feature importance

def visualize\_feature\_importance():

# Get feature importance from the model

feature\_importances = rf\_model.feature\_importances\_

features = X.columns

# Create a DataFrame for better visualization

importance\_df = pd.DataFrame({

'Feature': features,

'Importance': feature\_importances

}).sort\_values(by='Importance', ascending=False)

# Plot the feature importance

plt.figure(figsize=(12, 8))

sns.barplot(x='Importance', y='Feature', data=importance\_df, palette='viridis')

plt.title('Feature Importance in Predicting Heart Attack Risk')

plt.xlabel('Importance Score')

plt.ylabel('Features')

plt.tight\_layout()

plt.show()

# Run the prediction and visualize feature importance

predict\_heart\_attack()

visualize\_feature\_importance()

OUTPUT:

For this input:

A screenshot of a computer

Description automatically generated

This is the output I got:

A graph showing a number of different colored bars

Description automatically generated with medium confidence

GITHUB LINK:

<https://github.com/Ansarshaik02/ML---PROJECT>