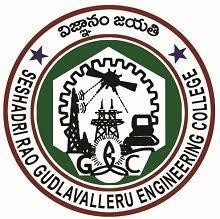
# Heart Attack Risk Prediction

# Machine Learning Project Report Submitted to the Faculty of

# JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA, KAKINADA In partial fulfillment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY

# IN

# INFORMATION TECHNOLOGY



##### Presented by batch B1

1. T. Naga Pavan Reddy 21481A12F2 3. P. Jaswanth 21481A12C6
2. Y. Uday Kiran 21481A12H4 4. P. Mohit Sai 21481A12C5

Under the guidance of:

**Dr. Ch. Kavitha,** M. Tech., PhD.,

Professor & Mentor (AS&A)

DEPARTMENT OF INFORMATION TECHNOLOGY

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

SESHADRI RAO KNOWLEDGE VILLAGE, GUDLAVALLERU - 521 356

ANDHRA PRADESH

2024-2025

**SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE**

**SESHADRI RAO KNOWLEDGE VILLEGE**

**GUDLAVALLERU - 521 356**

**DEPARTMENT OF INFORMATION TECHNOLOGY**



**CERTIFICATE**

This is to certify that the project Report entitled “Heart Attack Risk Prediction” is a bonafide record of work carried out by T.Naga Pavan Reddy(21481A12F2), P.Jaswanth(21481A12C6),Y.Uday Kiran(21481A12H4),P.Mohit Sai(21481A12C5) under the guidance and Supervision in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology of Jawaharlal Nehru Technological University Kakinada, Kakinada during the academic year 2024-2025.

**(Dr. Ch. Kavitha, M. Tech, PhD)**

Project Guide

**INDEX**

**TITLE** **PAGE NO**

1. ABSTRACT 4
2. PROBLEM STATEMENT & OBJECTIVES 5
3. INTRODUCTION 5
4. DATASET DESCRIPTION 6
5. ALGORITHM 7
6. IMPLEMENTATION 10
7. CODE IMPLEMENTATION 11-16
8. ACCURACY RESULTS TABLE 16
9. REASONS FOR ACCURACY 17
10. CONCLUSION 18

**ABSTRACT**

# Predicting heart attack risks is a critical task due to the complexity of medical parameters and variations in individual health conditions. This project, Heart Attack Risk Prediction and Categorization Using Key Attributes, addresses these challenges by leveraging machine learning to classify heart attack risks into Low Risks and High Risk categories. The objective is to assist healthcare providers in identifying high-risk cases, enabling timely interventions and informed decision-making.

# Using a pre-processed dataset containing 100,000 records, the project employs classification algorithms such as Naive Bayes, K-Nearest Neighbors (KNN), Logistic Regression, and Linear Regression to analyze data patterns. Key features include age, gender, blood pressure, alcohol use, stress levels, heart rate, and diet. Data preprocessing involves handling missing values, managing outliers, encoding categorical variables, and feature scaling to ensure data quality.

# The models’ performance is evaluated using metrics such as accuracy and F1 score, ensuring reliable and precise predictions. By integrating robust machine learning algorithms and data analysis techniques, this project provides a scalable and efficient system for heart attack risk assessment. The results aim to reduce heart attack by delivering early risk predictions, enabling healthcare professionals to optimize resources and improve heart care outcomes.

# Heart Attack Risk Prediction

**Problem Statement:** Analyze individual’s health data and medical history to predict the likelihood of a heart attack.

**Objectives:**

• Identify Key Risk Factors.

• Develop a Predictive Model.

• Evaluate Model Performance.

• Optimize Patient Outcomes.

**Introduction:**

Heart disease remains one of the leading causes of mortality worldwide, making early prediction and prevention critical. This project aims to leverage machine learning to predict the likelihood of a heart attack based on an individual’s health data and medical history. By analyzing various features such as age, cholesterol levels, blood pressure, smoking habits, and family history of heart disease, the model aims to classify individuals into risk categories.

The project uses supervised learning algorithms to analyze patterns in historical health data, providing a predictive model that can assist healthcare professionals in identifying high-risk patients early. The key objective is to enhance early detection and facilitate timely interventions, ultimately reducing the number of heart-related emergencies.

Through this project, we aim to improve the accuracy of heart attack predictions, which can be an invaluable tool in preventive healthcare.

### Data Set Description

##### Data set name: Heart Attack Dataset

* **Data set size**: 1,00,000
* **Source**: Kaggle

##### Features (Input Variables):

##### Age: The age of the individual.

##### Sex/Gender: The gender of the individual (male, female).

##### Blood Pressure: Systolic and diastolic blood pressure values.

##### Cholesterol Levels: Total cholesterol, HDL (good cholesterol), and LDL (bad cholesterol) levels.

##### Smoking Habits: Whether the individual is a smoker or not.

##### Diabetes: Whether the individual has diabetes or not.

##### Family History: A history of heart disease in the family.

##### Physical Activity: Whether the individual engages in regular physical exercise.

##### Diet: The individual’s diet, specifically fat or salt intake.

##### Body Mass Index (BMI): A measure of body fat based on height and weight.

##### Blood Sugar Levels: High blood sugar levels can indicate risk factors for heart disease.

##### Stress Levels: High levels of stress or mental health issues.

##### Heart Rate: Resting heart rate or maximum heart rate.

##### Electrocardiogram (ECG): Heart rhythm and activity data, often a part of medical history.

##### Exercise-Induced Angina: Whether the individual experiences chest pain during physical activity.

##### Target Variable (Output):

* Heart Attack Risk:
* 1: Indicates that the individual has had a heart attack or is at high risk of having one.
* 0: Indicates that the individual has not had a heart attack or is not at immediate risk.

#### ALGORITHMS:

We have used the below Algorithms to predict the adaptability levels based on features in samples:

1. Naive Bayes Classifier
2. K-Nearest Neighbours
3. Linear Regression
4. Logistic Regression

#### Naive Bayesian classifier

* Naive Bayes is a probabilistic classification algorithm based on Bayes theorem. It assumes that the features used for classification are conditionally independent, which is often a simplifying but unrealistic assumption.
* Despite this simplification, Naive Bayes can be surprisingly effective in practice, especially for text classification tasks.
* It calculates the probability of a data point belonging to a certain class based on the conditional probabilities of its features.
* Naive Bayes is known for its efficiency and speed.
* Formulae: The simple form of the calculation for Bayes Theorem is as follows

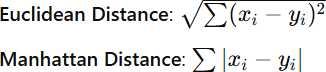
P(A|B) = P(B|A) \* P(A) / P(B)

#### K-Nearest Neighbors (KNN)

* KNN does not explicitly "train" a model. Instead, it stores the entire dataset in memory for reference.
* Calculates the distance between the input instance and all the instances in the training dataset (e.g., using Euclidean, Manhattan, or Minkowski distance).
* Identifies the **K nearest neighbors** to the input instance based on the calculated distances.
* Predicts the output based on the majority class (classification) or the average value (regression) of these neighbors.
* **K (Number of Neighbors)** determines how many nearest neighbors are considered when making a prediction. Smaller values of K focus on local data, while larger values generalize predictions based on broader trends.

##### Distance Metric:

Common metrics to compute the distance between points include:



**Linear Regression**

##### Hypothesis Function:

Represents the relationship between the independent variable(s) (features) and the dependent variable (target). For a single feature: y = β0 + β1x + ϵ.

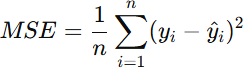
y: Predicted output (dependent variable). β0: Intercept (value of y when x=0).

β1: Slope of the line (rate of change of y with respect to x). x: Independent variable (feature).

ϵ: Error term (difference between actual and predicted values).

##### Cost Function:

Measures the error between predicted and actual values. Linear Regression uses Mean Squared Error (MSE).

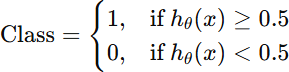


##### Optimization:

* The goal is to minimize the cost function by finding the optimal coefficients (β0,β1,…….βn).
* Methods:
  + **Gradient Descent:** Iteratively updates coefficients to reduce the error.
  + **Normal Equation:** Directly computes coefficients without iteration.

#### Logistic Regression:

* Logistic Regression is a supervised machine learning algorithm used for classification tasks. It predicts the probability of a dependent variable belonging to a specific class, typically binary (e.g., yes/no, 0/1) or multi-class.
* Unlike Linear Regression, Logistic Regression models the relationship between features and the probability of a categorical outcome using a sigmoid function.
* **Hypothesis Function:** Instead of predicting a continuous value, Logistic Regression predicts a probability value between 0 and 1.
* **Decision Boundary:** The output of the sigmoid function is a probability, and a threshold (commonly 0.5) is used to classify instances:



## IMPLEMENTATION

##### Major Parts of Project Implementation

* **Data Collection and Preparation**:

Gather and preprocess the dataset, which includes handling missing data, encoding categorical variables, and scaling features for machine learning.

##### Algorithm Selection:

Choose the machine learning algorithms that will be used for Ad click Prediction such as SVM, Naive Bayes, k-NN and Weighted k-NN.

##### Data Splitting:

Split the dataset into training and testing sets based on different ratios (e.g., 80:20, 70:30, 60:40, 50:50) for model training and evaluation.

##### Model Training:

Train each selected algorithm on the training data to build predictive models for Ad Click Prediction.

##### Model Evaluation:

Assess the performance of each model using evaluation metrics such as accuracy, precision, recall, and F1-score to determine their effectiveness in student placement analysis.

##### Selection of the Highest Accuracy Algorithm:

Identify the algorithm with the highest accuracy across different data split ratios, which will be considered the optimal model for Ad Click Prediction.

##### Results Analysis and Reporting:

Analyze the results, draw conclusions, and create a report or presentation to communicate the findings and the selected algorithm's effectiveness.

## CODE IMPLEMENTATION

##### NAIVE BAYES CLASSIFIER:

import pandas as pd

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

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

# Load the dataset

data = pd.read\_csv('extended\_dataset.csv')

# Separate features and target variable

X = data.drop(['Patient ID', 'Heart Attack Risk'], axis=1)

y = data['Heart Attack Risk']

# Encode categorical variables

label\_encoders = {}

for column in X.select\_dtypes(include=['object']).columns:

    le = LabelEncoder()

    X[column] = le.fit\_transform(X[column])

    label\_encoders[column] = le

# Standardize the features

scaler = StandardScaler()

X = scaler.fit\_transform(X)

# Define different split ratios

split\_ratios = {

    "70:30": 0.3,

    "50:50": 0.5,

    "60:40": 0.4,

    "80:20": 0.2

}

accuracy\_results = {}

for ratio\_name, test\_size in split\_ratios.items():

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

    nb\_model = GaussianNB()

    nb\_model.fit(X\_train, y\_train)

  y\_pred = nb\_model.predict(X\_test)

    accuracy = accuracy\_score(y\_test, y\_pred) \* 100

    accuracy\_results[ratio\_name] = accuracy

accuracy\_df = pd.DataFrame(list(accuracy\_results.items()), columns=["Split Ratio", "Accuracy (%)"])

print("Naive Bayes has been applied to the dataset, and these are the accuracy results for different split ratios:")

print(accuracy\_df)

file\_name = "naive\_bayes\_accuracy\_splits.csv"

accuracy\_df.to\_csv(file\_name, index=False)

print(f'Results saved in {file\_name}')

**K-NN:**

import pandas as pd

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

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

# Load the dataset

data = pd.read\_csv('extended\_dataset.csv')

# Separate features and target variable

X = data.drop(['Patient ID', 'Heart Attack Risk'], axis=1)

y = data['Heart Attack Risk']

# Encode categorical variables

label\_encoders = {}

for column in X.select\_dtypes(include=['object']).columns:

    le = LabelEncoder()

    X[column] = le.fit\_transform(X[column])

    label\_encoders[column] = le

# Standardize the features

scaler = StandardScaler()

X = scaler.fit\_transform(X)

# Define different split ratios

split\_ratios = {

    "70:30": 0.3,

    "50:50": 0.5,

    "60:40": 0.4,

    "80:20": 0.2

}

# Initialize an empty dictionary to store accuracy for each split

accuracy\_results = {}

# Apply KNN model for each split ratio

for ratio\_name, test\_size in split\_ratios.items():

    # Split data based on current test\_size ratio

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

    # Train KNN model

    knn\_model = KNeighborsClassifier(n\_neighbors=5)  # You can adjust n\_neighbors as needed

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

    # Predict and calculate accuracy

    y\_pred = knn\_model.predict(X\_test)

    accuracy = accuracy\_score(y\_test, y\_pred) \* 100  # Convert accuracy to percentage

    accuracy\_results[ratio\_name] = accuracy

# Convert results into a DataFrame for easy visualization

accuracy\_df = pd.DataFrame(list(accuracy\_results.items()), columns=["Split Ratio", "Accuracy (%)"])

# Print the statement

print("K-Nearest Neighbors has been applied to the dataset, and these are the accuracy results for different split ratios:")

print(accuracy\_df)

file\_name = "knn\_accuracy\_splits.csv"

accuracy\_df.to\_csv(file\_name, index=False)

print(f'Results saved in {file\_name}')

**Logistic Regression:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, GridSearchCV

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import StandardScaler, PolynomialFeatures

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import LabelEncoder

# Load the dataset

file\_path = "extended\_dataset.csv"  # Replace with your file path

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

# Prepare data: Select features (X) and target variable (y)

X = data.drop(columns=['Heart Attack Risk', 'Patient ID'])  # Exclude target and ID

y = data['Heart Attack Risk']

# Use LabelEncoder for categorical variables

label\_encoders = {}

X\_encoded\_label = X.copy()

# Apply label encoding to categorical columns

for column in X.select\_dtypes(include='object').columns:

    le = LabelEncoder()

    X\_encoded\_label[column] = le.fit\_transform(X[column])

    label\_encoders[column] = le  # Store the encoder for reference

# Scale the numerical features for better performance

scaler = StandardScaler()

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

# Optionally, create polynomial features (if the relationship is non-linear)

poly = PolynomialFeatures(degree=2)  # You can adjust the degree

X\_poly = poly.fit\_transform(X\_scaled)

# Define the Logistic Regression model with balanced class weight

model = LogisticRegression(max\_iter=1000, class\_weight='balanced')

# Hyperparameter tuning using GridSearchCV for Logistic Regression (tuning 'C' parameter)

param\_grid = {'C': [0.01, 0.1, 1, 10, 100]}  # Regularization parameter (inverse of regularization strength)

grid\_search = GridSearchCV(model, param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_poly, y)

# Get the best model and its hyperparameters

best\_model = grid\_search.best\_estimator\_

# Define split ratios for train-test splits

split\_ratios = [(0.5, 0.5), (0.6, 0.4), (0.7, 0.3), (0.8, 0.2)]

accuracies = {}

# Train and test Logistic Regression on different split ratios

for train\_ratio, test\_ratio in split\_ratios:

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

    # Fit the best model on the training data

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

    # Predict on the test set

    y\_pred = best\_model.predict(X\_test)

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

    accuracies[f"Train:Test = {train\_ratio\*100:.0f}:{test\_ratio\*100:.0f}"] = accuracy \* 100

# Print results as percentages

for split, acc in accuracies.items():

    print(f"{split}: Accuracy = {acc:.2f}%")

**Linear Regression:**

#### # Import necessary libraries

#### import pandas as pd

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

#### from sklearn.linear\_model import LinearRegression

#### from sklearn.preprocessing import StandardScaler, LabelEncoder

#### from sklearn.metrics import accuracy\_score

#### file\_path = 'extended\_dataset.csv'  # Replace with your file path

#### heart\_data = pd.read\_csv(file\_path)

#### # Step 1: Drop irrelevant columns

#### columns\_to\_drop = ["Patient ID", "Country", "Continent", "Hemisphere"]

#### heart\_data\_cleaned = heart\_data.drop(columns=columns\_to\_drop)

#### categorical\_columns = ["Sex", "Diet", "Blood Pressure"]

#### encoder = LabelEncoder()

#### for col in categorical\_columns:

#### heart\_data\_cleaned[col] = encoder.fit\_transform(heart\_data\_cleaned[col])

#### X = heart\_data\_cleaned.drop(columns=["Heart Attack Risk"])

#### y = heart\_data\_cleaned["Heart Attack Risk"]

#### numerical\_columns = X.select\_dtypes(include=["float64", "int64"]).columns

#### scaler = StandardScaler()

#### X[numerical\_columns] = scaler.fit\_transform(X[numerical\_columns])

#### def train\_evaluate\_linear\_regression\_accuracy(X, y, train\_ratio):

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

#### X, y, test\_size=1-train\_ratio, random\_state=42

#### )

#### model = LinearRegression()

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

#### y\_pred = model.predict(X\_test)

#### y\_pred\_rounded = [round(pred) for pred in y\_pred]

#### accuracy = accuracy\_score(y\_test, y\_pred\_rounded) \* 100  # Convert to percentage

#### return accuracy

#### ratios = [0.5, 0.6, 0.7, 0.8]

#### linear\_regression\_accuracies = {

#### f"{int(ratio\*100)}:{int((1-ratio)\*100)}":train\_evaluate\_linear\_regression\_accuracy(X, y, ratio)

#### for ratio in ratios

#### }

#### for split, accuracy in linear\_regression\_accuracies.items():

#### print(f"Split Ratio {split} - Accuracy: {accuracy:.2f}%")

#### ACCURACY RESULTS TABLE

##### Result Table

* The major implementation of our project is applying varieties of classification algorithms while dividing our dataset into a) 80:20 b) 70:30 c) 60:40 d) 50:50 ratios of training and test samples.
* Algorithms used are:
  1. Naive Bayes Classifier
  2. K-NN Classifier
  3. Linear Regression
  4. Logistic Regression

For all the splits of our dataset, we got best accuracy for the Logistic Regression Algorithm.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **TRAIN-TEST RATIO** | **NAIVE BAYES CLASSIFIER** | **KNN** | **LINEAR**  **REGRESSION** | **LOGISTIC**  **REGRESSION** |
| 1 | 50:50 | 64.22 | 96.65 | 64.23 | 64.23 |
| 2 | 60:40 | 64.09 | 98.54 | 64.09 | 64.09 |
| 3 | 70:30 | 64.03 | 99.26 | 64.03 | 64.03 |
| 4 | 80:20 | 64.20 | 99.77 | 64.20 | 64.02 |

**REASONS FOR KNN ACCURACY:**

1. **Nature of the Data:**

* K-NN performs well when the dataset has clear clusters or patterns in the features.
* If your data points are well-separated in the feature space, K-NN can easily classify them based on proximity.

1. **Small Train-Test Split Ratios:**

* Higher train-test ratios (like 70:30 or 80:20) ensure the model gets more training data, improving its ability to classify test data.

1. **Low Dimensionality:**

* K-NN works better in low-dimensional spaces. If your dataset has a manageable number of features, the distance-based calculation becomes more reliable.

1. **Well-Scaled Data:**

* If the dataset features are properly normalized or scaled, K-NN's distance-based approach becomes more effective.

1. **Choice of K Value:**

* The optimal value of k (number of neighbors) could have been chosen, avoiding underfitting or overfitting.

1. **No Model Assumptions:**
   * Unlike Logistic Regression or Naive Bayes, K-NN doesn't assume a specific distribution or relationship in the data, which might align better with your dataset's structure.

**Key Factors for Good K-NN Accuracy:**

* + Well-Defined Clusters: Data with clear and separable patterns enhances proximity-based classification.
  + Feature Scaling: Normalization ensures all features contribute equally to distance calculations.
  + Optimal Hyperparameters: Choosing the right k value and distance metric improves performance.
  + Sufficient Training Data: Higher train-test ratios (e.g., 70:30, 80:20) allow K-NN to model class distributions better.

**Conclusion:**

* A dataset was prepared to predict heart attack likelihood using machine learning, involving health data and medical history.
* Data preprocessing steps included handling missing values, scaling features, and encoding categorical variables.
* Machine learning models like Naive Bayes, K-NN, Linear Regression, and Logistic Regression were applied to classify health risks.
* Various train-test splits (50:50, 60:40, 70:30, 80:20) were used to evaluate model accuracy and adaptability.
* K-NN achieved the highest accuracy due to its ability to classify data with distinct clusters and well-scaled features.
* The K-NN algorithm emerged as the best performer for this project due to its simplicity and effectiveness.
* The project successfully utilized health data to predict heart attack risks, demonstrating its potential for real-world applications.
* This model can support healthcare professionals in identifying high-risk individuals and enabling early intervention.
* Incorporating additional features like lifestyle habits and genetic predisposition can further improve the model's accuracy.
* The system has the potential to reduce heart attack incidents through timely risk detection and care prioritization.