A

Mini Project Report

On

**Calorie Burnt Prediction System**

for the partial fulfillment of Bachelor of Technology Degree in Computer Science & Engineering

*Submitted by*

**ANSHUL RANA**

**Under the Guidance of**

**Mr. Anuj Kumar**

**Assistant Professor**



DEPARTMENT OF COMPUTING

**GRAPHIC ERA HILL UNIVERSITY**

HALDWANI CAMPUS

BAREILLY ROAD, BERIPARAO, HALDWANI

DISTRICT: - NAINITAL-263139

(2023-2024)

THIS IS TO CERTIFY THAT THE THESIS TITLED **“CALORIE BURNT PREDICTION SYSTEM” SUBMITTED** BY **ANSHUL RANA**TO GRAPHIC ERA HILL UNIVERSITY FOR THE AWARD OF THE DEGREE OF **BACHELOROF TECHNOLOGY**, IS A BONAFIDE RECORD OF THE RESEARCH WORK DONE BY HIM UNDER OUR SUPERVISION. THE CONTENTS OF THIS PROJECT IN FULL OR IN PARTS HAVE NOT BEEN SUBMITTED TO ANY OTHER INSTITUTE OR UNIVERSITY FOR THE AWARD OF ANY DEGREE OR DIPLOMA.

CERTIFCATE

PLACE: HALDWANI

DATE:

**NAME OF GUIDE** MRS. RICHA PANDEY

**SOC MINI PROJECT COORDINATOR**

MR. ANUJ KUMAR DIXIT

GEHU,HALDWANI

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who have contributed to the completion of this project. This endeavor wouldn't have been possible without the support, guidance, and encouragement of several individuals.

First and foremost, I extend my deepest appreciation to MRS. RICHA PANDEY MAM and MR. ANUJ KUMAR SIR for their invaluable guidance and continuous support throughout the project. Their expertise, insightful feedback, and unwavering encouragement significantly enriched the quality of this work.

I am also thankful to my college GRAPHIC ERA HILL UNIVERSITY(HALDWANI CAMPUS) for providing the necessary resources and facilities that facilitated the smooth progress of the project.

I extend my heartfelt thanks to my colleagues and friends who provided moral support, shared their knowledge, and offered assistance whenever needed.

Lastly, I express my gratitude to my family for their understanding, patience, and unwavering belief in my abilities. Their support has been a constant source of motivation.

**ANSHUL RANA**

**(2294019)**

ABSTRACT

Calorie burnt prediction by machine learning algorithm” aim to predict the number of calories burnt by an individual during physical activity using machine learning techniques. We collected a dataset that includes features such as heart rate, body temperature, and duration of activity. We used various machine learning models, including XGBoost, linear regression, SVM and random forest, to predict calorie burn based on 15,000 records with seven features. The results indicate that the XGBboost model can accurately predict calorie burn with a minimum mean absolute error of calories. This work contributes to the growing body of research on using machine learning for health and fitness applications and has potential implications for personalized health coaching and wellness tracking. The project is nourished with more than 15,000 data and its MAE (Mean Absolute error) is 1.48 which will enhance over time for better results. The highest accuracy of training and testing is gained by the XGBboost model with 99.67% with mean absolute error is almost 1.48%.

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **Section** | **Title** |
| **ACKNOWLEDGEMENT** |  |
| **ABSTRACT** |  |
| **INTRODUCTION** |  |
| **1.1.** | **Project Background** |
| **1.2.** | **Objectives** |
| **1.3.** | **Scope of the Project** |
| **LITERATURE REVIEW** |  |
| **2.1.** | **Related Work** |
| **2.2.** | **Key Concepts** |
| **METHODOLOGY** |  |
| **3.1 – 3.2** | **Data Collection and Preprocessing** |
| **3.3 - 3.4** | **Data Analysis and Feature Selection** |
| **3.5 – 3.7** | **Model Training and Selection** |
| **LIBRARIES USED** |  |
| **RESULTS** |  |
| **5.1.** | **Data Analysis** |
| **CONCLUSION AND FUTURE WORK** |  |
| **6.1.** | **Conclusion** |
| **6.2** | **Future Work** |
| **APPENDICES** |  |
| **REFERENCES** |  |

**CHAPTER-1**

**INTRODUCTION**

* 1. **Project Background**

In this growing technological era, People are less aware of their health and mental stability. Due to lack of time, they intake more junk food than healthy options, which leads to an increase in the total calorie rate in their body. Which is a Major cause of obesity a calorie is the rate of energy stored and energy expenditure. Calorie is a unit of hear energy. Health and fitness are becoming increasingly important to individuals and society as a whole. As people seek to live healthier lifestyles, they are turning to wearable devices and fitness trackers to monitor their physical activity and track their progress. One important metric that these devices track is the number of calories burnt during physical activity. Accurately predicting calorie burn can help individuals set and achieve fitness goals and can also inform health coaching and wellness tracking programs. The motivation for this research is to develop a machine-learning model that can accurately predict calorie burn during physical activity. This has potential applications in a range of settings, including personalized health coaching, fitness tracking, and wellness programs. By developing an accurate calorie burn prediction model, we can help individuals make more informed decisions about their physical activity and improve their overall health and well-being.

**1.2 Objective of the Study**

This study aims to propose a solution for predicting calories burned using machine learning algorithms. The learning algorithms considered are Linear Regression, System Vector Machine (SVR), Random Forest Regression and XG Boost Regression. The goal of this study is to evaluate which algorithms are the best in predicting calories burned by an individual based on weight, gender, age, height, duration of the activity, heart rate, and body temperature.

**1.3 Scope of the Project**

The model produced by the study can be integrated with or used alongside existing technologies to provide a better estimate of calories burned by individuals after physical activities.

**CHAPTER-2**

**LITERATURE REVIEW**

**2.1 Related Works**

Machine learning algorithms have gained widespread use in recent years to predict calorie burn during physical activity. These studies often collect physical activity data and other relevant variables such as heart rate, age, and gender from fitness trackers, mobile applications, and wearable devices. This section provides an overview of some of the critical studies in this area.

* Sathiya T et al. [4]: Applied a CNN model to classify food items from input images and predict user’s calorie intake. They used deep learning and image processing techniques, achieving 91.65% accuracy.
* Sona P Vinoy et al. [6]: Used machine learning algorithms such as XGBboost regressor and linear regression models to predict calorie burn during workouts. Their dataset included 15,000 entries with attributes like age, height, weight, duration, heart rate, body temperature, and calorie count. The mean absolute error was 2.71 for XGBboost and 8.31 for linear regression. Model accuracy was not mentioned.
* Suvarna Shreyas Ratnakar et al. [7]: Used the XGBboost machine learning algorithm on a dataset of 15,000 entries to predict calories burnt from physical activities. The mean absolute error was 2.7, but model accuracy was not mentioned.
* Rachit Kumar Singh et al. [8]: Used logistic regression, linear regression, and lasso regression models to predict calorie burn. Details on mean absolute error, dataset size, and model accuracy were not provided.
* Marte Nipas et al. [9]: Used a Random Forest algorithm to predict burned calories, achieving a model accuracy of 95.77%. They utilized an iterative method to refine outputs from inputs.
* Gunasheela B L et al. [10]: Employed digital image processing techniques such as image acquisition, RGB conversion, feature extraction, and image enhancement to predict calorie intake from input images.
* Mohammad Tarek Aziz, Sudheesh R, Renzon Daniel Cosme Pecho, Nayeem Uddin Ahmed Khan, Akba Ull Hasna Era, MD. Abir Chowdhury. [11]: Used Machine Learning Algorithms such asXGBoost, SVM, Linear Regression and Random Forest Regression to predict Calories burnt with an accuracy of 99.67% and MAE of 1.48.

**2.2 Key Concepts**

Machine Learning Algorithms for Calorie Prediction:

* Algorithms Used: CNN, XGBoost regressor, linear regression, logistic regression, lasso regression, Random Forest, SVM.
* Techniques: Deep learning, image processing, digital image processing, supervised learning.

Data and Variables:

* Common Variables: Age, height, weight, duration of activity, heart rate, body temperature, calorie intake
* Data Sources: Fitness trackers, mobile applications, wearable devices, raw datasets, input images.

Energy Expenditure Analysis:

* Factors: Body size, body composition, food intake, physical activity.
* Methods: Statistical techniques, MET (Metabolic Equivalent of Task).

Challenges and Needs:

* Need for models that accurately predict energy expenditure across various physical activities and individuals.

**CHAPTER – 3**

**METHODOLOGY**

**3.1 Dataset Description**

Data collection is an essential process in any machine learning project, as the quality of the data used has a significant figure no 1 impact on the performance of the resulting model. In this research, the dataset was collected from Kaggle, a popular platform for data scientists and machine learning practitioners to access and share datasets.In this work, the dataset contained over 15,000 records and 7 variables.

**3.2 Dataset Preprocessing**

Pre-processing of data- it is important that we process our data before passing it to the model for better results. null values and missing values are handled at this point because the information on our data directly affects how our model learns.

We preprocessed the data by removing missing values and outliers. Because, preprocess datasets are appropriate for applying into the algorithm for training and testing. We split the data into a training set (80% of the data) and a test set (20% of the data) for model training and evaluation.

Table I. shows the dataset specification that contains 15000 observations and nine variables. The Raw data gathered has eight (8) numeric variables and one (1) categorical variable. The data has no duplicate rows and no missing cells.Except Gender variable all are Numerical variable.

TABLE 1: DATASET SPECIFICATION

A white and black text on a white background

Description automatically generated

**3.3 Data Analysis**

The variables in the dataset must be analyzed first in order to determine their relationship to the target variable, which is the calorie burned. The variables heart rate, duration of exercise, body temperature is highly correlated with the dependent variable calorie, followed by the height and weight.

Pearson’s correlation is a statistical measure that evaluates the strength and direction of the linear relationship between two continuous variables. It is a widely used method to measure the correlation between two variables, and it is denoted by the symbol “r”.

The Pearson correlation coefficient can be interpreted as follows:

* A value of 1 indicates a perfect positive correlation, meaning that as one variable increases, the other variable also increases.
* A value of -1 indicates a perfect negative correlation, meaning that as one variable increases, the other variable decreases.
* A value of 0 indicates no correlation, meaning that there is no linear relationship between the two variables.
* A value between 0 and 1 indicates a positive correlation, meaning that there is a linear relationship between the two variables, but it is not perfect.
* A value between -1 and 0 indicates a negative correlation, meaning that there is a linear relationship between the two variables, but it is not perfect.

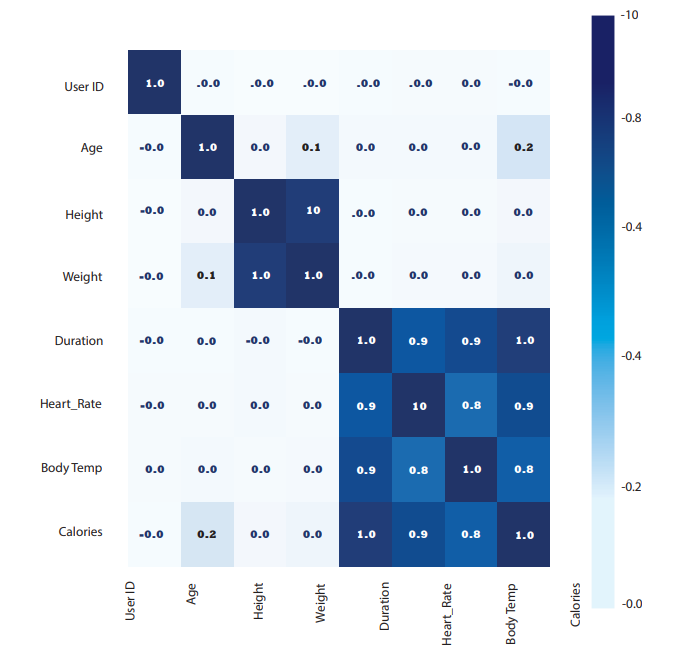


Fig.1 Pearson’s Correlation between Variables

**3.4Feature Selection**

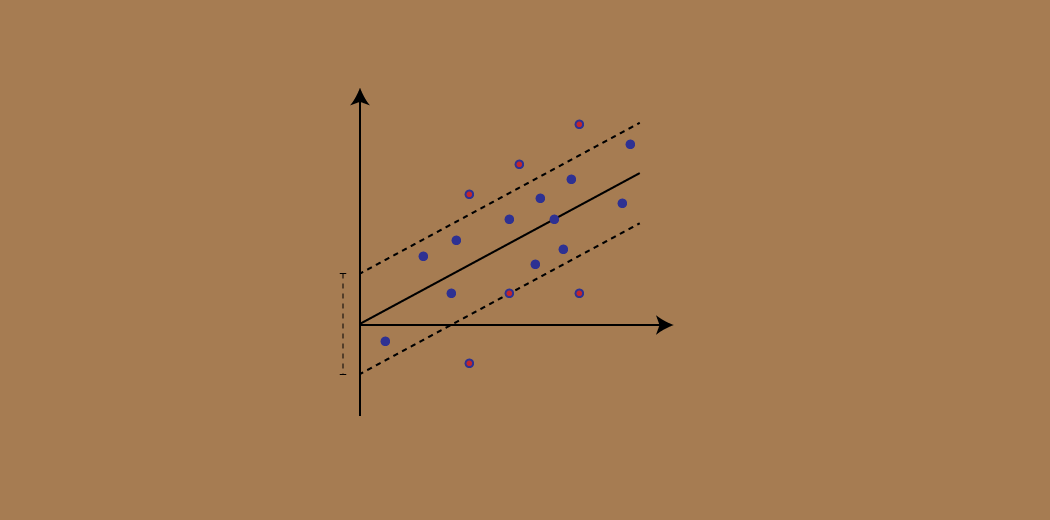
The data used in this study contains eight (8) features which consist of ID, Age, Height, Weight, Duration, Heart Rate, Body Temperature, and Gender. Based on the data analysis, the feature ID was not included because it doesn’t have any impact in predicting the calorie burned of a person.

**3.5 Regressive models**

* Regression is used in predictive modeling in which algorithms are used to predict continuous outcomes based on the given inputs.
* Linear regression model :- is used to identify the linear relationship of the input variables and output variables.
* Random Forest regression:-It is a supervised learning algorithm this model combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.
* XG boost regression:- It stands for extreme gradient boosting. This ensemble learning model produces strong predictions by combining the prediction of multiple weak models. It handles large datasets and provides efficient handling of missing values.
* SVM(support vector machine):- It tries to find a hyperplane that separates two classes and then classifies a new point depending on whether it lies on the positive or negative side of the hyperplane depending on the class to predict.

**Support vector machine (SVM):**

Support Vector Machines (SVM) are widely used in [machine learning](https://www.analyticsvidhya.com/blog/2024/05/free-university-courses-to-learn-machine-learning/) for classification problems, but they can also be applied to [regression](https://www.analyticsvidhya.com/blog/2023/05/regression-vs-classification/) problems through Support Vector Regression (SVR). SVR uses the same principles as SVM but focuses on predicting continuous outputs rather than classifying data points. This tutorial will explore how SVR works, emphasizing key concepts such as quadratic, radial basis function, and sigmoid kernels. By leveraging these kernels, SVR can effectively handle complex, non-linear relationships in data. We will also demonstrate how to implement SVR in Python using training samples, showcasing its practical applications in [artificial intelligence](https://www.analyticsvidhya.com/blog/2024/05/how-to-learn-artificial-intelligence/).



**Support Vector Regression (SVR):**

Support Vector Regression (SVR) is a type of machine learning algorithm used for regression analysis. The goal of SVR is to find a function that approximates the relationship between the input variables and a continuous target variable, while minimizing the prediction error.

Unlike Support Vector Machines (SVMs) used for classification tasks, SVR seeks to find a hyperplane that best fits the data points in a continuous space. This is achieved by mapping the input variables to a high-dimensional feature space and finding the hyperplane that maximizes the margin (distance) between the hyperplane and the closest data points, while also minimizing the prediction error.

SVR can handle non-linear relationships between the input variables and the target variable by using a kernel function to map the data to a higher-dimensional space. This makes it a powerful tool for regression tasks where there may be complex relationships between the input variables and the target variable.

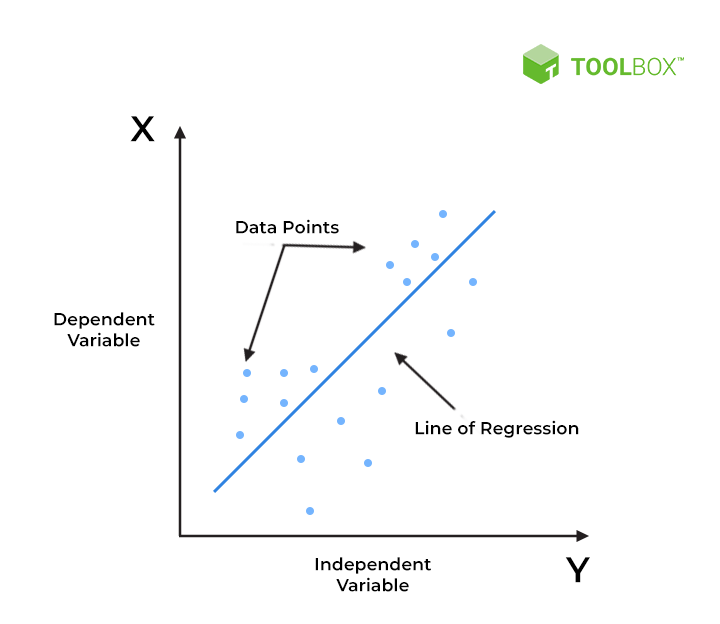
**Linear Regression:**

**Linear regression is an algorithm that provides a linear relationship between an independent variable and a dependent variable to predict the outcome of future events. It is a statistical method used in data science and machine learning for predictive analysis.**

The independent variable is also the predictor or explanatory variable that remains unchanged due to the change in other variables. However, the dependent variable changes with fluctuations in the independent variable. The regression model predicts the value of the dependent variable, which is the response or outcome variable being analyzed or studied.

Thus, linear regression is a supervised learning algorithm that simulates a mathematical relationship between variables and makes predictions for continuous or numeric variables such as sales, salary, age, product price, etc.This analysis method is advantageous when at least two variables are available in the data, as observed in stock market forecasting, portfolio management, scientific analysis, etc.

A sloped straight line represents the linear regression model.



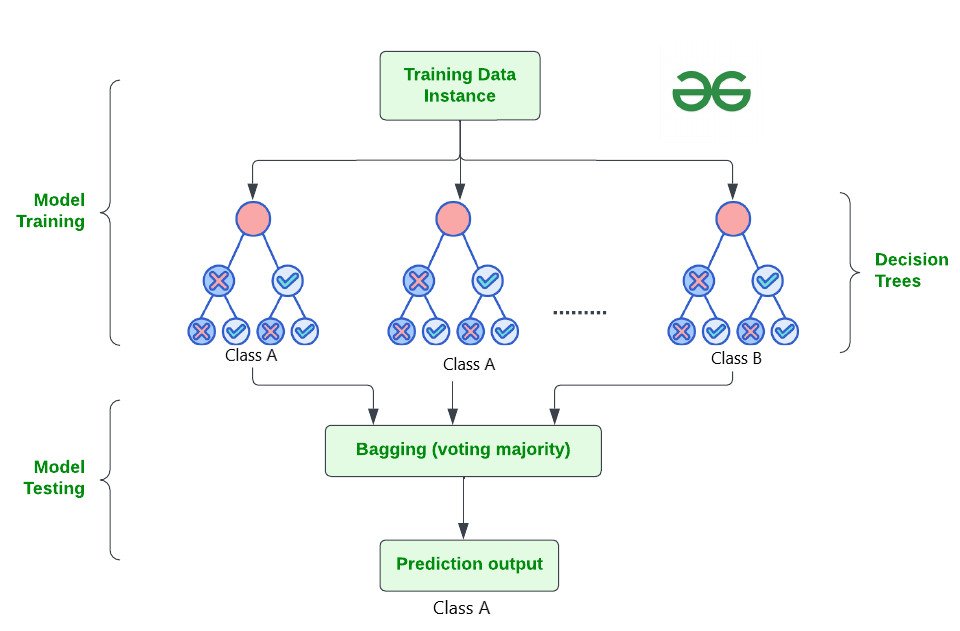
X-axis = Independent variable

Y-axis = Output / dependent variable

Line of regression = Best fit line for a model

**Random forest**

Random Forest algorithm is a powerful tree learning technique in [Machine Learning](https://www.geeksforgeeks.org/ml-machine-learning/). It works by creating a number of[Decision Trees](https://www.geeksforgeeks.org/decision-tree/) during the training phase. Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition. This randomness introduces variability among individual trees, reducing the risk of [overfitting](https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/) and improving overall prediction performance. In prediction, the algorithm aggregates the results of all trees, either by voting (for classification tasks) or by averaging (for regression tasks) This collaborative decision-making process, supported by multiple trees with their insights, provides an example stable and precise results. Random forests are widely used for classification and regression functions, which are known for their ability to handle complex data, reduce overfitting, and provide reliable forecasts in different environments.



**XGBoost :**

Decision trees create a model that predicts the label by evaluating a tree of if-then-else true/false feature questions,and estimating the minimum number of questions needed to assess the probability of making a correct decision. Decision trees can be used for classification to predict a category, or regression to predict a continuous numeric value.

In the simple example below, a decision tree is used to estimate a house price (the label) based on the size and number of bedrooms (the features).

## A decision tree.

A Gradient Boosting Decision Trees (GBDT) is a decision tree [ensemble learning algorithm](https://en.wikipedia.org/wiki/Ensemble_learning) similar to random forest, for classification and regression. Ensemble learning algorithms combine multiple machine learning algorithms to obtain a better model.

Both random forest and GBDT build a model consisting of multiple decision trees. The difference is in how the trees are built and combined.

Random forest uses a technique called bagging to build full decision trees in parallel from random bootstrap samples of the data set. The final prediction is an average of all of the decision tree predictions.

The term “gradient boosting” comes from the idea of “boosting” or improving a single weak model by combining it with a number of other weak models in order to generate a collectively strong model. [Gradient boosting](https://developer.nvidia.com/blog/gradient-boosting-decision-trees-xgboost-cuda/) is an extension of boosting where the process of additively generating weak models is formalized as a gradient descent algorithm over an objective function. Gradient boosting sets targeted outcomes for the next model in an effort to minimize errors. Targeted outcomes for each case are based on the gradient of the error (hence the name gradient boosting) with respect to the prediction.

GBDTs iteratively train an ensemble of shallow decision trees, with each iteration using the error residuals of the previous model to fit the next model. The final prediction is a weighted sum of all of the tree predictions. Random forest “bagging” minimizes the variance and overfitting, while GBDT “boosting” minimizes the bias and underfitting.

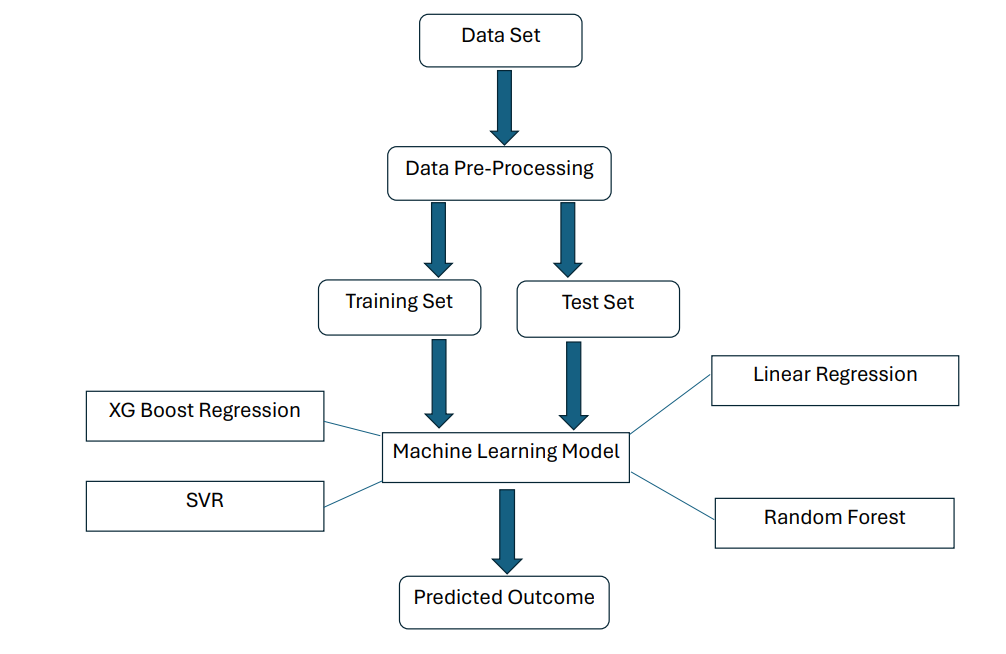
XGBoost is a scalable and highly accurate implementation of gradient boosting that pushes the limits of computing power for boosted tree algorithms, being built largely for energizing machine learning model performance and computational speed. With XGBoost, trees are built in parallel, instead of sequentially like GBDT. It follows a level-wise strategy, scanning across gradient values and using these partial sums to evaluate the quality of splits at every possible split in the training set.

**3.6Model Training and Prediction**

The study considered four regression models, namely linear regression, random forest regression, XGBoost Regression and Support VectorRegression. These models were trained with the dataset using the program developed in python programming. To test the models, K-fold cross-validation with ten iterations was used to reduce the chances of overfitting and to improve the prediction accuracy.

**3.7 Model Assessment and Selection**

Based on the result of testing, the models will be evaluated using the scores got by each model in 10 iterations of K-fold validation. The one with the highest average score in predicting the value of the target variable will be the basis in selecting the best model for the study. Also, prediction performance of each model was also considered to support the basis in selecting the best model. These performance measures are Mean Square Error(MSE), Root Mean Square Error(RMSE), and Mean Absolute Error(MAE). The model with the lowest prediction errors can be considered the best model. The workflow of our model is shown below :



**CHAPTER – 4**

**LIBRARIES USED**

**Streamlit**

Streamlit lets you transform Python scripts into interactive web apps in minutes, instead of weeks. Build dashboards, generate reports, or create chat apps. Once you’ve created an app, you can use our [Community Cloud platform](https://streamlit.io/cloud) to deploy, manage, and share your app.

* **Simple and Pythonic:** Write beautiful, easy-to-read code.
* **Fast, interactive prototyping:** Let others interact with your data and provide feedback quickly.
* **Live editing:** See your app update instantly as you edit your script.
* **Open-source and free:** Join a vibrant community and contribute to Streamlit's future.

**Pandas**

Pandas (styled as pandas) is a [software library](https://en.wikipedia.org/wiki/Software_library) written for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)) for data manipulation and [analysis](https://en.wikipedia.org/wiki/Data_analysis). In particular, it offers [data structures](https://en.wikipedia.org/wiki/Data_structure) and operations for manipulating numerical tables and [time series](https://en.wikipedia.org/wiki/Time_series).

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created

Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data.

**Numpy**

NumPy (**Numerical Python**) is an open source Python library that’s used in almost every field of science and engineering. It’s the universal standard for working with numerical data in Python, and it’s at the core of the scientific Python and PyData ecosystems. NumPy users include everyone from beginning coders to experienced researchers doing state-of-the-art scientific and industrial research and development. The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages.

The NumPy library contains multidimensional array and matrix data structures (you’ll find more information about this in later sections). It provides **ndarray**, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

NumPy gives you an enormous range of fast and efficient ways of creating arrays and manipulating numerical data inside them. While a Python list can contain different data types within a single list, all of the elements in a NumPy array should be homogeneous. The mathematical operations that are meant to be performed on arrays would be extremely inefficient if the arrays weren’t homogeneous.

**Seaborn**

Seaborn is a library for making statistical graphics in Python. It builds on top of [matplotlib](https://matplotlib.org/) and integrates closely with [pandas](https://pandas.pydata.org/) data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them

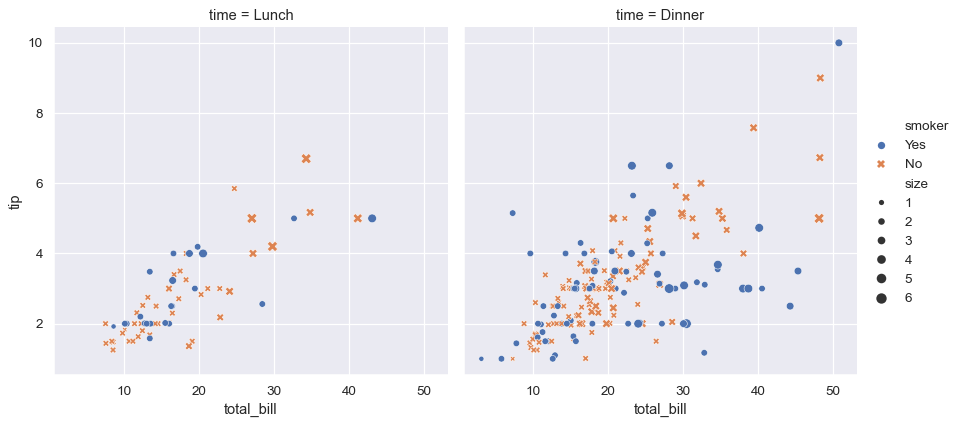
Seaborn is the only library we need to import for this simple example. By convention, it is imported with the shorthand sns.

Behind the scenes, seaborn uses matplotlib to draw its plots. For interactive work, it’s recommended to use a Jupyter/IPython interface in [matplotlib mode](https://ipython.readthedocs.io/en/stable/interactive/plotting.html), or else you’ll have to call [matplotlib.pyplot.show()](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.show.html#matplotlib.pyplot.show) when you want to see the plot.

A high-level API for statistical graphics

There is no universally best way to visualize data. Different questions are best answered by different plots. Seaborn makes it easy to switch between different visual representations by using a consistent dataset-oriented API.

The function [relplot()](https://seaborn.pydata.org/generated/seaborn.relplot.html#seaborn.relplot) is named that way because it is designed to visualize many different statistical relationships. While scatter plots are often effective, relationships where one variable represents a measure of time are better represented by a line. The [relplot()](https://seaborn.pydata.org/generated/seaborn.relplot.html#seaborn.relplot) function has a convenient kind parameter that lets you easily switch to this alternate representation:

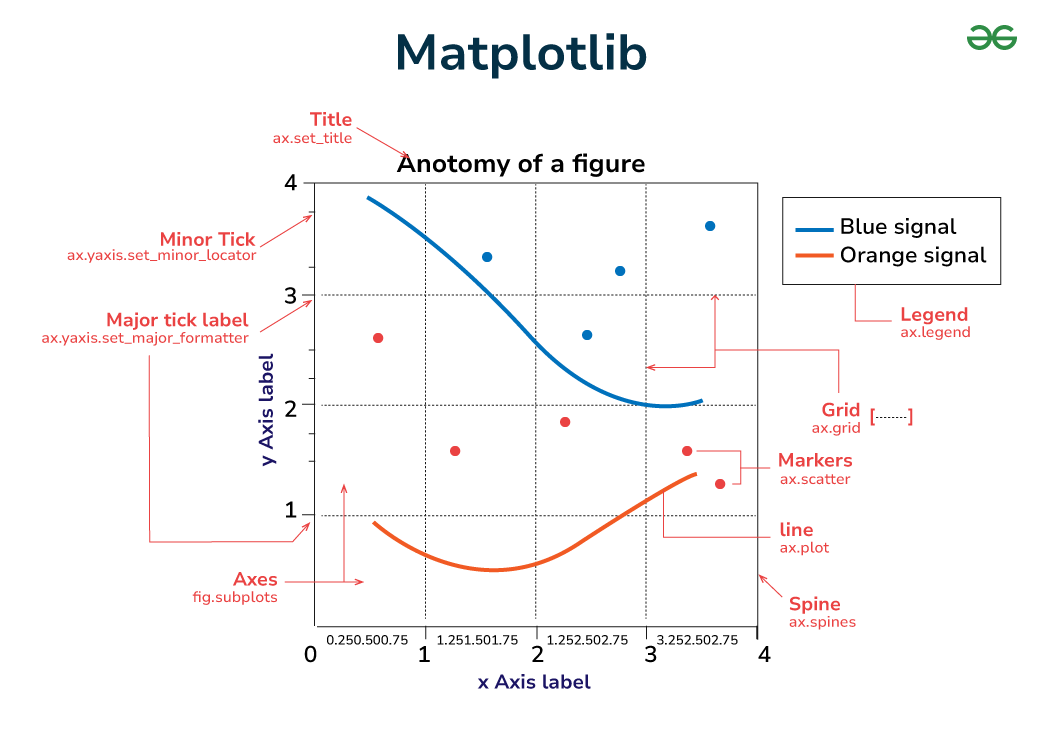


**Matplotlib**

Matplotlib is a powerful plotting library in Python used for creating static, animated, and interactive visualizations. Matplotlib’s primary purpose is to provide users with the tools and functionality to represent data graphically, making it easier to analyze and understand. It was originally developed by John D. Hunter in 2003 and is now maintained by a large community of developers.

Matplotlib Figure:

In Matplotlib, a figure is the top-level container that holds all the elements of a plot. It represents the entire window or page where the plot is drawn.



1. **Figures in Matplotlib**: The Figure object is the top-level container for all elements of the plot. It serves as the canvas on which the plot is drawn. You can think of it as the blank sheet of paper on which you’ll create your visualization.
2. **Axes in Matplotlib**: Axes are the rectangular areas within the figure where data is plotted. Each figure can contain one or more axes, arranged in rows and columns if necessary. Axes provide the coordinate system and are where most of the plotting occurs.
3. **Axis in Matplotlib:** Axis objects represent the x-axis and y-axis of the plot. They define the data limits, tick locations, tick labels, and axis labels. Each axis has a scale and a locator that determine how the tick marks are spaced.
4. **Marker in Matplotlib**: Markers are symbols used to denote individual data points on a plot. They can be shapes such as circles, squares, triangles, or custom symbols. Markers are often used in scatter plots to visually distinguish between different data points.
5. **Adding lines to Figures**: Lines connect data points on a plot and are commonly used in line plots, scatter plots with connected points, and other types of plots. They represent the relationship or trend between data points and can be styled with different colors, widths, and styles to convey additional information.
6. **Matplotlib Title:** The title is a text element that provides a descriptive title for the plot. It typically appears at the top of the figure and provides context or information about the data being visualized.
7. **Axis Labels in Matplotlib**: Labels are text elements that provide descriptions for the x-axis and y-axis. They help identify the data being plotted and provide units or other relevant information.
8. **Ticks**: Tick marks are small marks along the axis that indicate specific data points or intervals. They help users interpret the scale of the plot and locate specific data values.
9. **Tick Labels:** Tick labels are text elements that provide labels for the tick marks. They usually display the data values corresponding to each tick mark and can be customized to show specific formatting or units.
10. **Matplotlib Legend**: Legends provide a key to the symbols or colors used in the plot to represent different data series or categories. They help users interpret the plot and understand the meaning of each element.
11. **Matplotlib Grid Lines:** Grid lines are horizontal and vertical lines that extend across the plot, corresponding to specific data intervals or divisions. They provide a visual guide to the data and help users identify patterns or trends.
12. **Spines of Matplotlib Figures**: Spines are the lines that form the borders of the plot area. They separate the plot from the surrounding whitespace and can be customized to change the appearance of the plot borders.

**Scikit learn**

Scikit-learn is an open-source [Python](https://www.geeksforgeeks.org/python-programming-language/) library that implements a range of machine learning, pre-processing, cross-validation, and visualization algorithms using a unified interface. It is an open-source machine-learning library that provides a plethora of tools for various [machine-learning](https://www.geeksforgeeks.org/machine-learning/) tasks such as [Classification](https://www.geeksforgeeks.org/basic-concept-classification-data-mining/), [Regression](https://www.geeksforgeeks.org/regression-classification-supervised-machine-learning/), [Clustering](https://www.geeksforgeeks.org/clustering-in-machine-learning/), and many more.

A dataset is nothing but a collection of data. A dataset generally has two main components:

* **Features**: (also known as predictors, inputs, or attributes) they are simply the variables of our data. They can be more than one and hence represented by a **feature matrix** (‘X’ is a common notation to represent feature matrix). A list of all the feature names is termed **feature names**.
* **Response**: (also known as the target, label, or output) This is the output variable depending on the feature variables. We generally have a single response column and it is represented by a **response vector** (‘y’ is a common notation to represent response vector). All the possible values taken by a response vector are termed **target names**.

Features of Scikit-learn

* Simple and efficient tools for data mining and data analysis. It features various classification, regression, and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.
* Accessible to everybody and reusable in various contexts.
* Built on the top of NumPy, SciPy, and matplotlib.
* Open source, commercially usable – BSD license.

**CHPTER- 5**

**RESULTS**

A graph of a graph

Description automatically generated

The graph depicts the learning curves of Linear Regression a model, showing the training and cross-validation scores as a function of the number of training examples.

Analysis

* **Fluctuations in Training Score**: The significant fluctuations in the training score indicate that the model's performance is highly sensitive to the specific subset of training data. This might be due to overfitting, where the model is learning the noise in the training data rather than the underlying patterns.
* **Stable Cross-validation Score**: The stability of the cross-validation score suggests that the model's generalization capability is consistent. This is a good sign, indicating that the model is not overfitting to the extent that it affects performance on unseen data.
* **Gap Between Training and Cross-validation Scores**: The presence of a gap between the training and cross-validation scores typically indicates overfitting. In this case, the gap is relatively small but noticeable, suggesting mild overfitting.

The analysis of this model is done to find the best algorithm for predicting the calories burnt during exercise from factors such as age, height, weight, body temperature, gender, heart rate, and duration of exercise. The algorithm which provides the least mean absolute error is considered as best, this study applies various machine learning models over the dataset to find the least value of Mae, according to these results XGBoost regression is best for solving this problem with a Mae value of 1.49.

|  |  |
| --- | --- |
| **Regressions** | **Mean Absolute Error(MAE)** |
| Linear Regression | 8.44 |
| Random Forest | 1.69 |
| XG Boost Regression | 1.49 |
| System Vector Regression (SVR) | 2.36 |

A screenshot of a graph

Description automatically generated

**CONCLUSION AND FUTURE WORK**

The main objective of this study was to create a precise machine learning model that could predict a specific outcome variable based on a series of characteristics. This was achieved through the collection and preparation of a dataset, as well as testing the effectiveness of various machine learning models and feature selection techniques. It is important to understand the number of calories we eat to stay fit and healthy. Calories burnt can be predicted from different regression algorithms such as Linear regression, XG boost regression, Ada boost regression, Decision tree regression, SVM, and Random forest regression. Out of these regression algorithms, the findings indicated that the XGBoost model demonstrated superior performance compared to the other models in terms of accuracy and other relevant metrics. This suggests that the XGBoost model could be a valuable tool for predicting similar outcome variables based on similar datasets. The MAE(Mean Absolute Error) value of the XG boost is 1.49 which is a good value. It means the errors are quite low. So, therefore, XG boost regression algorithm is the optimal algorithm for the calories burnt prediction so far with an accuracy of 99.88%.

However, the study also had some limitations, such as the limited size of the dataset and the possibility of overfitting. Future research could address these limitations and further improve the performance of the models and feature selection approaches. Overall, this study contributes to the field of machine learning and provides practical implications for the problem domain.

**APPENDIX**

1. The Code is structured in various parts:
2. Imports
3. Reading Data from CSV
4. Data Visualization
5. Feature and Target Separation
6. Train-Test Split
7. Data Preprocessing and Pipeline Creation
8. Model Training and Evaluation
9. Model Comparison
10. Model Training and Model Saving
11. App GUI

**# 1. Imports**

import pandas as pd

import numpy as np

import warnings

warnings.filterwarnings('ignore')

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler, OrdinalEncoder

from sklearn.pipeline import Pipeline

from sklearn.compose import ColumnTransformer

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

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

from xgboost import XGBRegressor

import pickle

from tkinter import \*

from tkinter import ttk, mainloop, StringVar

**# 2. Reading Data from CSV**

def read\_csv(file\_path):

return pd.read\_csv(file\_path)

**# 3. Data Overview and Checks**

def dataset\_info\_statistics(data):

print("Dataset Information:")

print(data.info())

print("\n")

print("Basic Statistics for Numerical Columns:")

print(data.describe())

print("\n")

def check\_null(data):

null\_counts = data.isnull().sum()

print("Null Values in the Dataset:")

return null\_counts

def check\_duplicates(data):

return data.duplicated().any()

**# 4. Data Visualization**

def plot\_graph(data):

numerical\_columns = data.select\_dtypes(include=np.number).columns

for column in numerical\_columns:

plt.figure(figsize=(5, 3))

sns.distplot(data[column], kde=True)

plt.title(f"Histogram for {column}")

plt.xlabel(column)

plt.ylabel("Frequency")

plt.show()

categorical\_columns = data.select\_dtypes(include='object').columns

for column in categorical\_columns:

plt.figure(figsize=(5, 3))

sns.countplot(data[column])

plt.title(f'Countplot for {column}')

plt.xlabel(column)

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.show()

**# 5. Feature and Target Separation**

def separate\_features\_target(data, target\_column):

X = data.drop(columns=[target\_column], axis=1)

y = data[target\_column]

return X, y

**# 6. Train-Test Split**

def perform\_train\_test\_split(X, y, test\_size=0.20, random\_state=42):

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

return X\_train, X\_test, y\_train, y\_test

**# 7. Data Preprocessing and Pipeline Creation**

preprocessor = ColumnTransformer(transformers=[

('ordinal', OrdinalEncoder(), ['Gender']),

('num', StandardScaler(), ['Age', 'Height', 'Weight', 'Duration', 'Heart\_Rate', 'Body\_Temp']),

], remainder='passthrough')

pipeline = Pipeline([("preprocessor", preprocessor), ("model", LinearRegression())])

**# 8. Model Training and Evaluation**

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

y\_pred = pipeline.predict(X\_test)

print(r2\_score(y\_test, y\_pred))

print(mean\_absolute\_error(y\_test, y\_pred))

**# 9. Model Scoring Function**

def model\_scorer(model\_name, model):

output = []

output.append(model\_name)

pipeline = Pipeline([

('preprocessor', preprocessor),

('model', model)

])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=42)

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

y\_pred = pipeline.predict(X\_test)

output.append(r2\_score(y\_test, y\_pred))

output.append(mean\_absolute\_error(y\_test, y\_pred))

kfold = KFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_results = cross\_val\_score(pipeline, X, y, cv=kfold, scoring='r2')

output.append(cv\_results.mean())

return output

**# 10. Model Comparison**

model\_dict = {

'SVR': SVR(),

'LR': LinearRegression(),

'RF': RandomForestRegressor(),

'XGBR': XGBRegressor()

}

model\_output = []

for model\_name, model in model\_dict.items():

model\_output.append(model\_scorer(model\_name, model))

print(model\_output)

**# 11. Model Training with Best Model**

pipeline = Pipeline([

('preprocessor', preprocessor),

('model', XGBRegressor())

])

pipeline.fit(X, y)

**# 12. Sample Prediction**

sample = pd.DataFrame({

'Gender': ['male'],

'Age': [68],

'Height': [190.0],

'Weight': [94.0],

'Duration': [29.0],

'Heart\_Rate': [105.0],

'Body\_Temp': [40.8],

})

print(pipeline.predict(sample))

**# 13. Model Saving and Loading**

with open('pipeline.pkl', 'wb') as f:

pickle.dump(pipeline, f)

with open('pipeline.pkl', 'rb') as f:

pipeline\_saved = pickle.load(f)

result = pipeline\_saved.predict(sample)

print(result)

**# 14. GUI for Prediction**

import pickle

import pandas as pd

import streamlit as st

# Load the pipeline

with open('pipeline.pkl', 'rb') as f:

pipeline = pickle.load(f)

# Define the Streamlit app

def main():

st.title("Calories Burnt Prediction using Machine Learning")

st.write("Fill in the details below to predict the amount of calories burnt:")

# Create input fields

gender = st.selectbox("Select Gender", ['male', 'female'])

age = st.number\_input("Enter Your Age", min\_value=0, max\_value=120, value=25)

height = st.number\_input("Enter Your Height in Cm", min\_value=50.0, max\_value=250.0, value=170.0)

weight = st.number\_input("Enter Your Weight in Kg", min\_value=20.0, max\_value=200.0, value=70.0)

duration = st.number\_input("Duration in Minutes", min\_value=1.0, max\_value=300.0, value=30.0)

heart\_rate = st.number\_input("Heart Rate in Bpm", min\_value=30.0, max\_value=200.0, value=70.0)

body\_temp = st.number\_input("Body Temp in Celsius", min\_value=30.0, max\_value=45.0, value=36.5)

if st.button("Predict"):

sample = pd.DataFrame({

'Gender': [gender],

'Age': [age],

'Height': [height],

'Weight': [weight],

'Duration': [duration],

'Heart\_Rate': [heart\_rate],

'Body\_Temp': [body\_temp],

}, index=[0])

result = pipeline.predict(sample)

st.success(f"Amount of Calories Burnt: {result[0]:.2f}")

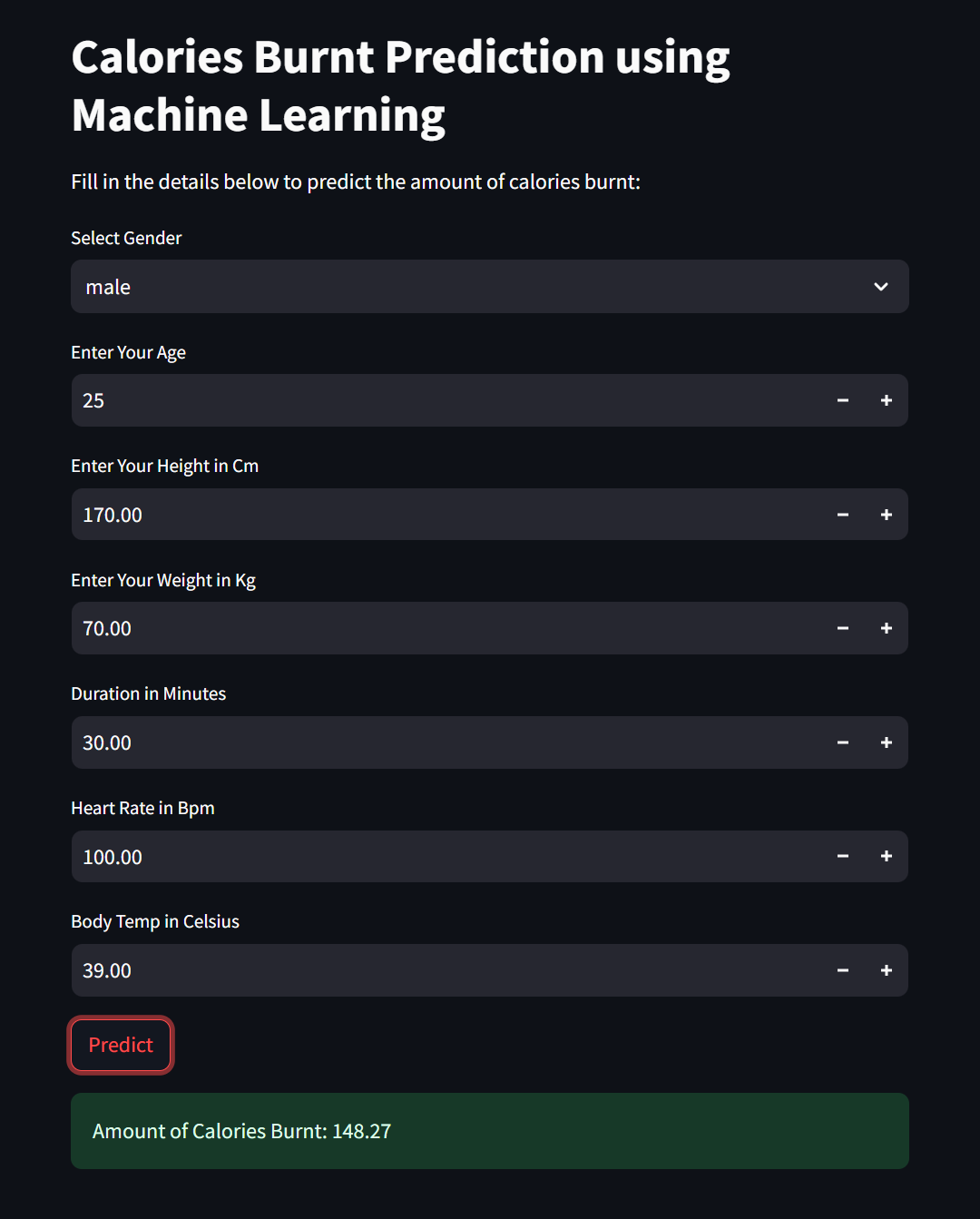
# Run the app

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

main()

**FINAL RESULT**

We have used **streamlit** library for deploying our model in web server to predict the calories burnt accurately.



**REFRENCES**

1. Goukens, Caroline, and Anne Kathrin Klesse. "Internal and external forces that prevent (vs.

Facilitate) healthy eating: Review and outlook within consumer Psychology." Current Opinion inPsychology (2022): 101328.

1. Khan, Abdul Wahid, et al. "Factors Affecting Fitness Motivation: An Exploratory Mixed MethodStudy." IUP Journal of Marketing Management 21.2(2022).https://www.medicalnewstoday.com/articles/319731
2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5496172/
3. 4.Kalpesh, Jadhav, et al. "Human Physical Activities Based Calorie Burn Calculator Using

LSTM." Intelligent Cyber Physical Systems and Internet of Things: ICoICI 2022. Cham: SpringerInternational Publishing, 2023. 405-424.

1. Nipas, Marte, et al. "Burned Calories Prediction using Supervised Machine Learning: Regression Algorithm." 2022 Second International Conference on Power, Control and Computing Technologies (ICPC2T). IEEE, 2022.
2. Burned Calories Prediction using Supervised Machine Learning: Regression Algorithm

(ICPC2T)DOI:[10.1109/ICPC2T53885.2022.9776710](http://dx.doi.org/10.1109/ICPC2T53885.2022.9776710)

1. Punita Panwar , Kanika Bhutani , Rimjhim Sharma and Rohit Saini . A Study on Calories Burnt Prediction Using Machine Learning.

ITM Web of Conferences 54, 01010 (2023) <https://doi.org/10.1051/itmconf/20235401010>

1. Mohammad Tarek Aziz\*, Sudheesh R, Renzon Daniel Cosme Pecho, Nayeem Uddin Ahmed Khan, Akba Ull Hasna Era, MD. Abir Chowdhury. CALORIES BURNT PREDICTION USING MACHINE LEARNING APPROACH.(2023) Current Integrative Engineering Volume 1, Issue 1, 29-36