

# **CALORIES BURNT PREDICTION USING MACHINE LEARNING**

**A MINI-PROJECT REPORT**

*Submitted by*

**SWETHA SARAVANAN (922319104038)**

**RANJITH KUMAR P (922319104029)**

**VIMAL NISANTH V (922319104046)**

**NANDHA KUMAR S (922319104022)**

*in partial fulfilment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**UNIVERSITY COLLEGE OF ENGINEERING DINDIGUL**

**ANNA UNIVERSITY: CHENNAI 600 025**

**APRIL/MAY 2022**

# **ANNA UNIVERSITY: CHENNAI 600 025**

## **BONAFIDE CERTIFICATE**

Certified that this project report “**CALORIES BURNT PREDICTION USING MACHINE LEARNING**” is the work of “**SWETHA SARAVANAN (922319104038), RANJITH KUMAR P (922319104029), VIMAL NISANTH (922319104046), NANDHA KUMAR S (922319104022)**” who carried out the project work under my supervision.

### **SIGNATURE**

Dr. M. VINOTH KUMAR M.E., Ph. D

### **HEAD OF THE DEPARTMENT**

Assistant professor

Computer Science and  
Engineering

University College of  
Engineering Dindigul-624 622

### **SIGNATURE**

Dr. P. Ramya M.E., Ph. D

### **SUPERVISOR**

Teaching Fellow

Computer Science and  
Engineering

University College of  
Engineering Dindigul-624 622

Submitted for the mini-project (CS8611) VIVA-VOCE held at University  
College of Engineering Dindigul on .....

INTERNAL EXAMINER

EXTERNAL EXAMINER

## **ACKNOWLEDGEMENT**

Firstly, we would like to thank the Lord Almighty for his benevolence and blessings that has given us the strength to complete this project.

We express our sincere gratitude to our Dean Dr. Sutha M.E., Ph.D., for providing us with plethora of resources, which enhances the quality of learning.

We express our sincere thanks to Dr. M. Vinoth Kumar, M.E, M.B.A., PhD, Assistant professor Sr. Gr and Head of the Department, Computer Science and Engineering, for providing necessary facilities to carry out the project work.

We would like to express our deep sense of gratitude to our guide P. Ramya M.E. PhD and Project Coordinator Dr. M. Vinoth Kumar M.E., M.B.A., PhD Assistant Professor, Senior Grade Department of Computer Science and Engineering, for his stimulating guidance, help and encouragement in the study, design and implementation of our project which enabled us to complete it successfully.

Also, we wish to thank all the faculty members of our department and our friends for rendering their support.

Swetha Saravanan

Ranjith Kumar P

Vimal Nisanth V

Nandha Kumar S

## **TABLE OF CONTENTS**

| <b>CHAPTER</b> | <b>TITLE</b>                     | <b>PAGE NO</b> |
|----------------|----------------------------------|----------------|
|                | <b>ABSTRACT</b>                  | <b>i</b>       |
|                | <b>LIST OF FIGURES</b>           | <b>ii</b>      |
|                | <b>LIST OF TABLES</b>            | <b>iii</b>     |
| <b>1.</b>      | <b>INTRODUCTION</b>              | <b>1</b>       |
|                | 1.1. Objective                   | 2              |
|                | 1.2. Machine Learning            | 3              |
|                | 1.2.1. Types of Machine learning | 4              |
|                | 1.3. Language Description        | 7              |
| <b>2.</b>      | <b>LITERATURE SURVEY</b>         | <b>10</b>      |
| <b>3.</b>      | <b>SYSTEM ANALYSIS</b>           |                |
|                | 3.1. Problem Definition          | 12             |
|                | 3.2. Existing system             | 12             |
|                | 3.3. Proposed system             | 13             |
| <b>4.</b>      | <b>SYSTEM REQUIREMENTS</b>       |                |
|                | 4.1. Software requirements       | 14             |
|                | 4.2. Software Description        | 14             |
|                | 4.3. Hardware requirements       | 16             |
| <b>5.</b>      | <b>SYSTEM DESIGNS</b>            |                |
|                | 5.1. Work flow                   | 17             |
|                | 5.2. UML Diagrams                | 17             |

|           |                                      |    |
|-----------|--------------------------------------|----|
| <b>6.</b> | <b>SYSTEM IMPLEMENTATION</b>         |    |
|           | 6.1. Architecture Diagram            | 20 |
|           | 6.2. Modules and Module Descriptions | 21 |
|           | 6.3. Algorithms                      | 22 |
|           | 6.4. Python modules                  | 24 |
| <b>7.</b> | <b>EXPERIMENTAL RESULTS</b>          |    |
|           | 7.1. Data Source                     | 32 |
|           | 7.2. Screenshots                     | 35 |
| <b>8.</b> | <b>RESULTS AND FINDINGS</b>          | 47 |
| <b>9.</b> | <b>CONCLUSION</b>                    | 48 |
|           | <b>APPENDIX (Source code)</b>        | 49 |
|           | <b>REFERENCES</b>                    | 52 |

## **ABSTRACT**

As we know that running is the effective workout for most calories burnt per hour. Stationary bicycling, running, and swimming are fabulous choices as well. HIIT (High Intensity Interval Training) works out are too incredible for burning calories. After a HIIT workout, your body will proceed to burn calories for up to 24 hours. Forecasting the workout burnt calories still remains an open challenge as the changes in the environmental calamity and body health. The machine learning strategies can predict the burnt out calories for the course of exercise done by a body. With this background, we have utilized Exercise dataset extracted from Kaggle Machine Learning repository for predicting the workout burnt calories. The forecasting of burnt calories rate are achieved using Linear Regressor, XGB Regressor and Random Forest Regressor algorithms.

The execution is done using python language under google colab. Experimental results shows that the Linear Regressor, XGB Regressor and Random Forest Regressor tends to retain 96%, 99.6%, 99.9%. The mean absolute error value that is getting in Linear Regressor, XGB Regressor and Random Forest Regressor is 8.385, 2.715, 1.700. The error value is very less in Random Forest Regressor. Therefore we can conclude that the best model for the calorie burn prediction is Random Forest Regressor.

## LIST OF FIGURES

| CHAPTER | TITLE                                    | PAGE NO |
|---------|--|---------|
| 1.1     | Machine Learning diagram                 | 3       |
| 1.2     | Supervised Learning                      | 4       |
| 1.3     | Classification and Regression difference | 5       |
| 1.4     | Unsupervised Learning                    | 6       |
| 1.5     | Semi-Supervised learning                 | 6       |
| 1.6     | Reinforcement Learning                   | 7       |
| 1.7     | Python IDEs                              | 9       |
| 4.1.    | Google Colaboratory welcome page         | 15      |
| 5.1     | Work Flow Diagram (XGBoost)              | 17      |
| 5.2.    | Use case Diagram                         | 17      |
| 5.3.    | Activity Diagram                         | 18      |
| 6.1.    | Architecture of proposed work            | 20      |
| 6.2     | XGBoost Regressor                        | 22      |
| 6.3     | Linear Regression                        | 23      |
| 6.4     | Random Forest Regression                 | 24      |
| 6.5.    | pyplot graphs                            | 27      |
| 6.6.    | sklearn.metrics sample graph             | 31      |
| 7.1.    | Importing modules                        | 35      |
| 7.2.    | Checking number of rows and columns      | 37      |
| 7.3.    | Data information                         | 37      |
| 7.4.    | Checking Missing values                  | 38      |
| 7.5.    | Plotting Gender                          | 39      |
| 7.6.    | Plotting age                             | 39      |
| 7.7.    | Plotting height                          | 40      |
| 7.8.    | Plotting weight                          | 40      |

|       |                             |    |
|-------|-----------------------------|----|
| 7.9.  | Correlation graph           | 41 |
| 7.10. | Separated X values          | 42 |
| 7.11. | Separated Y values          | 43 |
| 7.12. | Testing and Training        | 43 |
| 7.13. | Linear Regression MAE       | 44 |
| 7.14. | XGB Regressor MAE           | 45 |
| 7.15. | Random Forest Regressor MAE | 46 |
| 8.1.  | Comparison graph            | 47 |

### LIST OF TABLES

| CHAPTER | TITLE                                | PAGE NO |
|---------|--------------------------------------|---------|
| 3.1.    | Existing system MAE                  | 13      |
| 7.1     | Exercise Dataset                     | 33      |
| 7.2     | Calories Dataset                     | 34      |
| 7.3     | First 5 rows of dataframe (calories) | 35      |
| 7.4.    | First 5 rows of dataframe (exercise) | 36      |
| 7.5.    | First 5 rows of Concatenated dataset | 36      |
| 7.6.    | Statistical measurement              | 38      |
| 7.7.    | Replaced (Gender) table              | 42      |



## 1. INTRODUCTION

In this fast and busy schedule life, people are not giving importance to the quality of food they are eating. They tend to neglect their eating patterns and habits. The fast-food consumption rate is alarmingly high and this consequently has led to the intake of unhealthy food. This leads to various health issues such as obesity, diabetes, an increase in blood pressure etc. Hence it has become very essential for people to have a good balanced nutritional healthy diet. There are many applications which are booming to help people so that they can have control over their diet and hence can reduce weight or they can help them to keep them fit and healthy.

Calories in the foods we eat provide energy in the form of heat so that our bodies can function. This means that we need to eat a certain amount of calories just to sustain life. But if we take in too many calories, then we risk gaining weight. So there is need to burn Calories, for burning calories we doing exercises and more. for know how much calories we have burn Today we are going to build a machine learning model that predict calories based on some data. Calories Burned in Physical Activity, physical activity burns calories beyond the basal metabolic rate. Your muscles use both readily available and stored energy sources in your body. The exercise calories burned during cardiovascular activities such as walking, running, swimming, and cycling depend on the intensity of the exercise, your body weight, and the amount of time you spend exercising. Moderate-intensity exercises such as brisk walking burn fewer calories per minute than more vigorous-intensity exercises such as running.

For example, you can use a walking calorie chart to find out how many calories you can burn per mile based on your weight and speed. Walking burns approximately 90 calories per mile for a 160-pound person.

Depending on the duration and intensity of exercise, your body burns available blood sugar, glycogen stored in the muscles and liver, fat and, if required, even begins to burn muscle protein.

Some people aim to exercise at 60% to 70% of their maximum heart rate to burn body fat. In that fat-burning zone, 85% of the calories you burn are from fat. However, you'll burn more total calories per minute if you exercise at a higher intensity.

The "fat-burning" zone is more tolerable for many people and might allow you to exercise for a longer period of time. But if you work out for a shorter period of time, a higher intensity session will help you to burn more calories. Fitness monitors and pedometers often estimate calories burned based on your weight, number of steps taken, speed, pace, and intensity. It is generally more accurate if the exercise intensity is measured by the heart rate during exercise. You may use hand grip pulse monitors on a treadmill or elliptical trainer for a more accurate estimate. More and more fitness bands and smartwatches have pulse detectors built in to monitor your exercise intensity. A chest strap heart rate monitor is considered the most accurate.

This project focuses on the calories burned in accordance with the duration provided and heart rate during the exercise period. It introduces the topic of linear regression and its predicting capability with the effectiveness from the data provided. This research helps in providing the benefits of a machine learning algorithm over predicting the calories burned.

A machine learning XGBoost regressor algorithm, linear regression algorithm and Random Forest Regressor algorithms are used to predict calories burned depends on the workout duration, body temperature, height, weight and age of the person. Then load the model XGB Regressor and evaluate the prediction on test data. The model goes through this test data and calories burned for the `X_test`. And compare the values predicted by our model with original values. For this we use the metrics-mean absolute blunders which tells what is the magnitude of mistakes the version is making metrics.

### **1.1. Objective:**

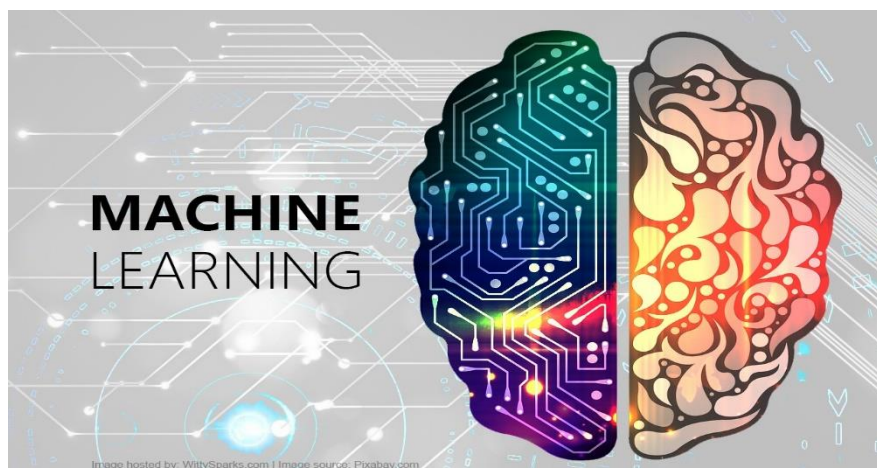
This project is to predict the calorie burned during the workout of different people and compare the two algorithms in machine learning by looking through the data sets. The dataset used in this study has 7 features, one target variable, and 15000 instances. We are using this data sets to train a dataset and find out the accurate algorithms and its mean absolute error and find the best model.

## 1.2. Machine Learning

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

Machine learning is a dimension of artificial intelligence that's in certain mark out as the software applications turn out to be more correct at predicting outcomes without being explicitly programmed to accomplish that.

- To predict the new output values historical data is used by these algorithms.
- Machine learning let the user to feed a computer algorithm, massive amount of information and then the computer will examine and make data-driven suggestions and decisions focusing on only the input data.
- Recommendation engines are a common use case for machine learning.
- Other popular uses include fraud detection, spam filtering, malware threat detection, business process automation (BPA) and Predictive maintenance.



**Fig. 1.1. Machine Learning**

### 1.2.1. Types of Machine Learning

Classical machine learning is often categorized by how an algorithm learns to become more accurate in its predictions. There are four basic approaches:

- Supervised learning,
- Unsupervised learning,
- Semi-supervised learning and
- Reinforcement learning.

The type of algorithm data scientists choose to use depends on what type of data they want to predict.

#### Supervised learning:

In this type of machine learning, data scientists supply algorithms with labelled training data and define the variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified.

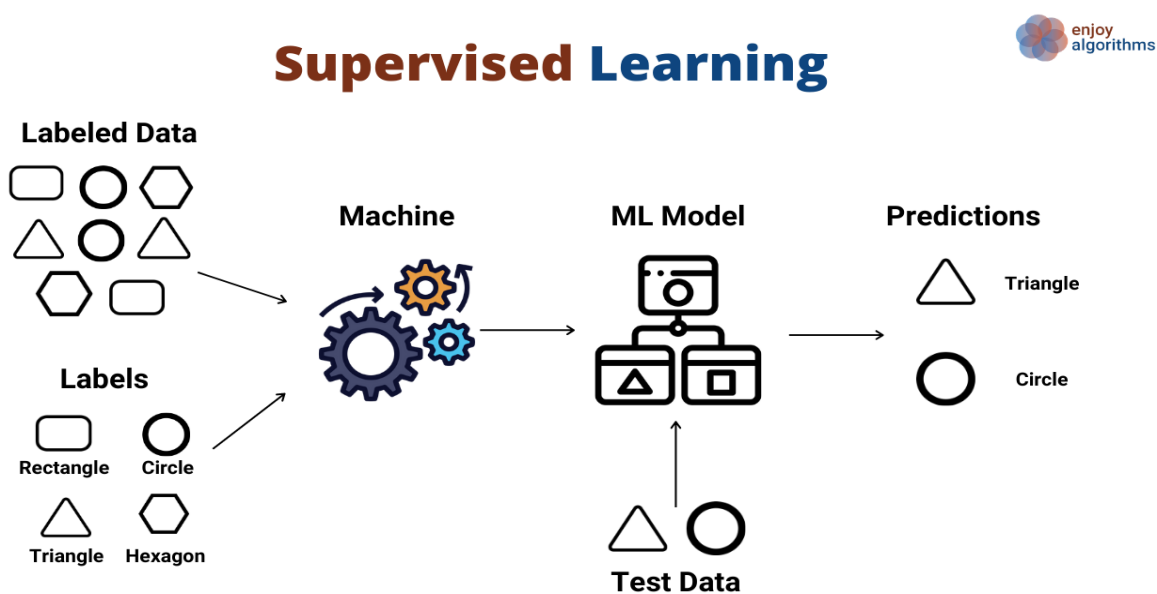


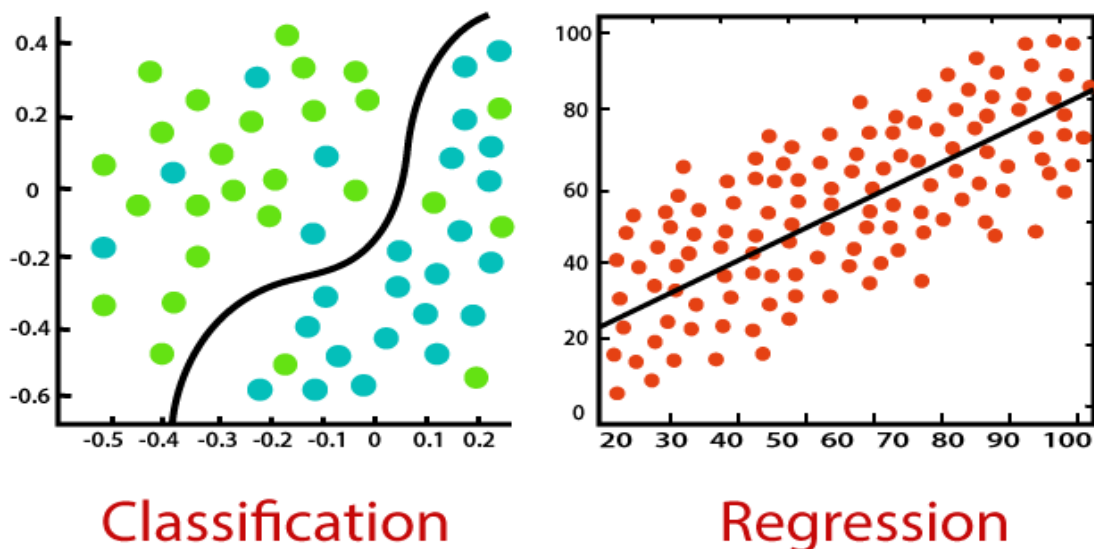
Fig. 1.2. Supervised Learning

Supervised learning can be divided into two categories:

- ❖ Classification and
- ❖ Regression.

**Classification**: Classification is a technique for determining which class the dependent belongs to based on one or more independent variables.

**Regression**: Regression is a statistical method used in finance, investing, and other disciplines that attempts to determine the strength and character of the relationship between one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables).



**Fig. 1.3. Classification and Regression difference**

**Unsupervised learning:**

This type of machine learning involves algorithms that train on unlabelled data. The algorithm scans through data sets looking for any meaningful connection. The data that algorithms train on as well as the predictions or recommendations they output are predetermined.

# Unsupervised Learning

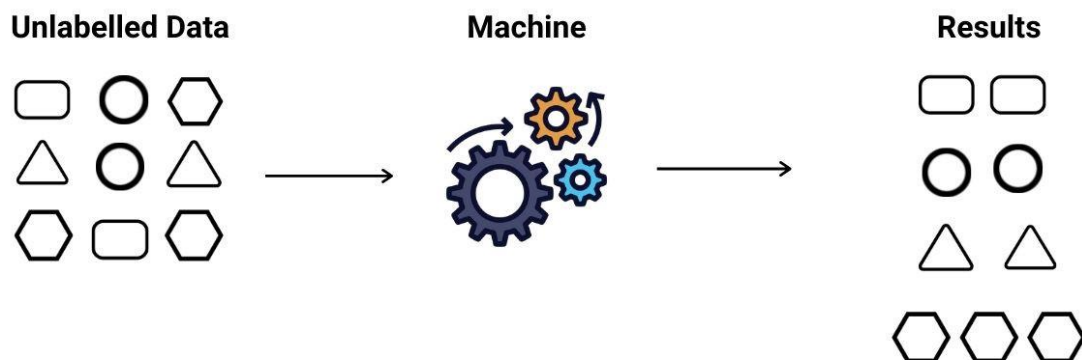


Fig. 1.4. Unsupervised Learning

## Semi-supervised learning:

This approach to machine learning involves a mix of the two preceding types. Data scientists may feed an algorithm mostly labelled training data, but the model is free to explore the data on its own and develop its own understanding of the data set

## Semi-supervised learning use-case

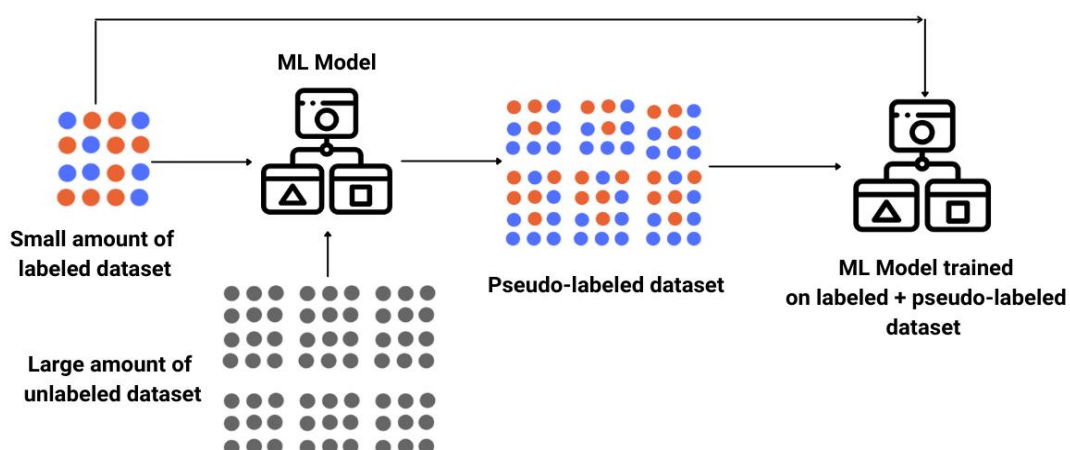
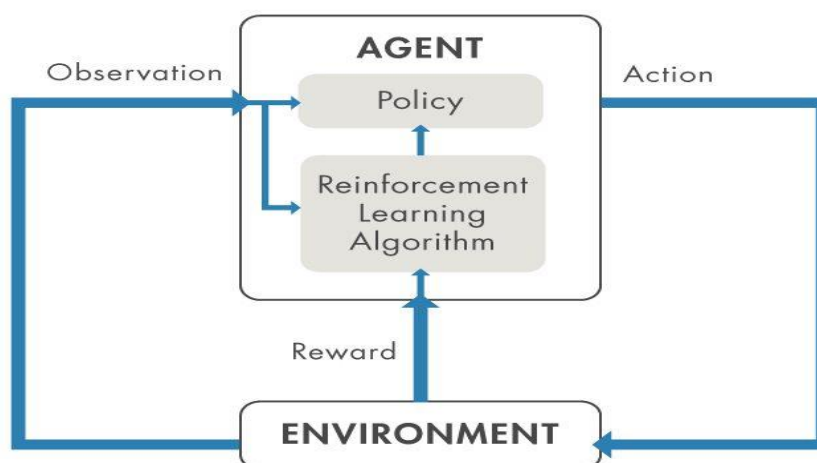


Fig. 1.5. Semi-Supervised learning

### Reinforcement learning:

Data scientists typically use reinforcement learning to teach a machine to complete a multi-step process for which there are clearly defined rules. Data scientists program an algorithm to complete a task and give it positive or negative cues as it works out how to complete a task. But for the most part, the algorithm decides on its own what steps to take along the way.



**Fig. 1.6. Reinforcement Learning**

### **1.3. Language Description:**

#### **PYTHON**

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today. A survey conducted by industry analyst firm RedMonk found that it was the second-most popular programming language among developers in 2022.

Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- Python is Interactive – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

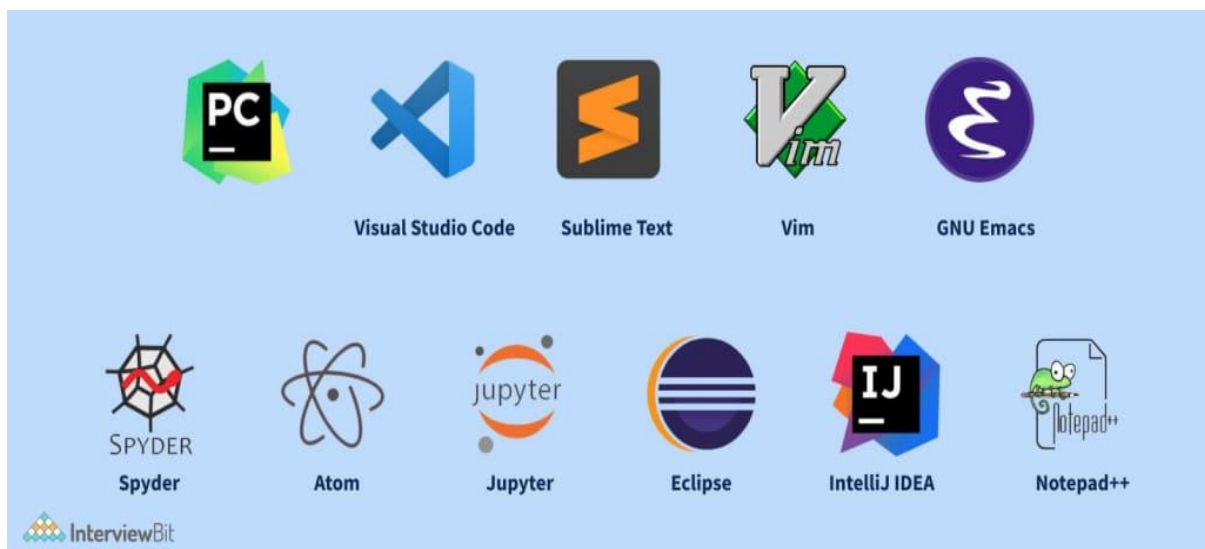
### **What makes Python the best programming language for machine learning?**

- Python offers concise and readable code.
- While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems.
- Developers get to put all their effort into solving an ML problem instead of focusing on the technical nuances of the language.
- Additionally, Python is appealing to many developers as it's easy to learn.
- Python code is understandable by humans, which makes it easier to build models for machine learning.
- Many frameworks, libraries, and extensions that simplify the implementation of different functionalities.
- It's generally accepted that Python is suitable for collaborative implementation when multiple developers are involved
- Since Python is a general-purpose language, it can do a set of complex machine learning tasks and enable you to build prototypes quickly that allow you to test your product for machine learning purposes.



**Some of the python IDEs are:**

- Atom, an open source cross-platform IDE with autocomplete, help and more Python features under package extensions.
- Eclipse ,with the Pydev plug-in. Eclipse supports many other languages as well.
- Jupyter Notebook, an IDE that supports markdown, Python, Julia, R and several other languages.
- Visual Studio Code, an Open Source IDE for various languages, including Python.
- PyCharm, a proprietary and Open Source IDE for Python development.
- Geany, IDE for Python development and other languages.



**Fig. 1.7**

**Python IDEs**

## **2. Literature Survey**

### **2.1. Ingmar, “Forecasting Workout Burnt Calories Using Machine Learning,” in Qatar Computing Research Institute, January 2016.**

In this study, the author attempted to find the performance analysis of the exercise dataset in forecasting the workout burnt calories by applying various feature selection methods with High correlation, High Variance, Anova Test analysis and KBest Feature analysis. The MI value for the Kbest feature for feature regression and mutual information regression also examined and visualized. The correlation of each features in the dataset is extricated and the distribution of target variable with respect to gender and duration is analysed. Experimental results shows that the Decision Tree and Gradient Boosting regressors tends to retain 99% before and after feature scaling for the Anova test, Correlated Feature, Variance Based and KBest Feature based methods.

### **2.2. Manal Chokr, and Shady Elbassuoni, “Calories prediction from food images,” in Innovative Applications of Artificial Intelligence, 2017.**

The purpose of this paper was to calculating the amount of calories in a given food item is now a common task. We propose a machine-learning-based approach to predict the amount of calories from food images. Our system does not require any input from the user, except from an image of the food item. We take a pragmatic approach to accurately predict the amount of calories in a food item and solve the problem in three phases. First, we identify the type of the food item in the image. Second, we estimate the size of the food item in grams. Finally, by taking into consideration the output of the first two phases, we predict the amount of calories in the photographed food item. All these three phases are based purely on supervised machinelearning. We show that this pipelined approach is very effective in predicting the amount of calories in a food item as compared to baseline approaches which directly predicts the amount of calories from the image.

**2.3. Trevon D. Logan NBER, “The Transformation of Hunger: The Demand for Calories Past and Present,” in National Bureau of Economic Research, Working Paper No. 11754, November 2005.**

According to conventional income measures, nineteenth century American and British industrial workers were two to four times as wealthy as poor people in developing countries today. Surprisingly, however, today's poor are less hungry than yesterday's wealthy industrial workers. I estimate the demand for calories of American and British industrial workers using the 1888 Cost of Living Survey and find that the estimated calorie elasticities for both American and British households are greater than calorie elasticity estimates for households in present day developing countries. The results are robust to measurement error, unreported food consumption, and indirect estimation bias. This finding implies substantial nutritional improvements among the poor in the twentieth century. Using the Engel curve implied by the historical calorie elasticities, I derive new income estimates for developing countries which yield income estimates that are six to ten times greater than those derived using purchasing power parity or GDP deflators.

**2.4. R.N. Dickerson, J. J. Patel, and C. J. McClain, “Protein and Calorie Requirements Associated With the Presence of Obesity. in Nutr Clin Pract. vol. 32, no. 1, 2017.**

Obesity compounds the metabolic response to critical illness and increases the risk for overfeeding complications due to its comorbidities. Hypocaloric, high-protein nutrition therapy affords the hospitalized patient with obesity the opportunity to achieve net protein anabolism with a reduced risk of overfeeding complications. The intent of this review is to provide the theoretical framework for development of a hypocaloric high-protein regimen, scientific evidence to support this mode of therapy, and unique considerations for its use in specialized subpopulations. Macronutrient goals and practical suggestions for patient monitoring are given.

### **3. SYSTEM ANALYSIS**

#### **3.1. Problem Definition:**

The variety of energy burned each day is immediately connected to weight loss, weight gain, or weight maintenance. To shed pounds, a person ought to burn greater calories than they take in, developing a calorie deficit. but, to do that they want to recognize what number of calories they burn each day. Most people think about calories as most effective having to do with food and weight reduction Calorie, a unit of energy or heat variously defined. Calorie may be defined as the amount of energy that is vital to increase 1 gram(g)of water by means of 1°C.This measurement can be carried out to lots of different strength releasing mechanisms outdoor of the human body.

Therefore, the goal of our project is to apply algorithms of machine learning over large existing data sets to effectively predict the calories burnt while doing exercise and to find the best algorithm that provides good accuracy.

#### **3.2. Existing system**

In the Existing system they met with a conclusion that the XGB Regressor has more accurate results than the Linear regression model. Mean absolute error imply absolute error ought to be as low as viable. it is not anything but the difference between the actual and predicted values through the models. The mean absolute error value that is getting in XGB Regressor is 2.71 which is a good value. The error value is very less. Therefore they conclude that the XGBoost Regressor is better than Linear regression.

| <b>Machine Learning Model</b> | <b>Mean Absolute Error</b> |
|-------------------------------|----------------------------|
| XGB Regressor                 | 2.71                       |
| Linear Regression             | 8.38                       |

**Table. 3.1. Existing system MAE**

### **3.3. Proposed system**

This paper is all about the collection of appropriate set to teach our machine learning models in order that it will find out what is the amount of calories that the individual goes to burn. Before feeding procedure the statistics via records pre-processing need to be done. After that data analysis is carry out where we use some visualization techniques to arrange the data in plots and graphs. Afterwards divide the data set into training and test set. Here we use Random Forest Regressor, XGBoost regressor and linear regression as machine learning models for comparison and then evaluate this models. The tool used is Google Colaboratory or Google Colab is a web based tool and a cloud-based service.

## 4. SYSTEM REQUIREMENTS

### 4.1. Software requirements:

The tool used is Google Colaboratory or Google Colab is a web based tool and a cloud-based service.

 Google Colaboratory

Operating System: Windows 10

### 4.2. Software Description:

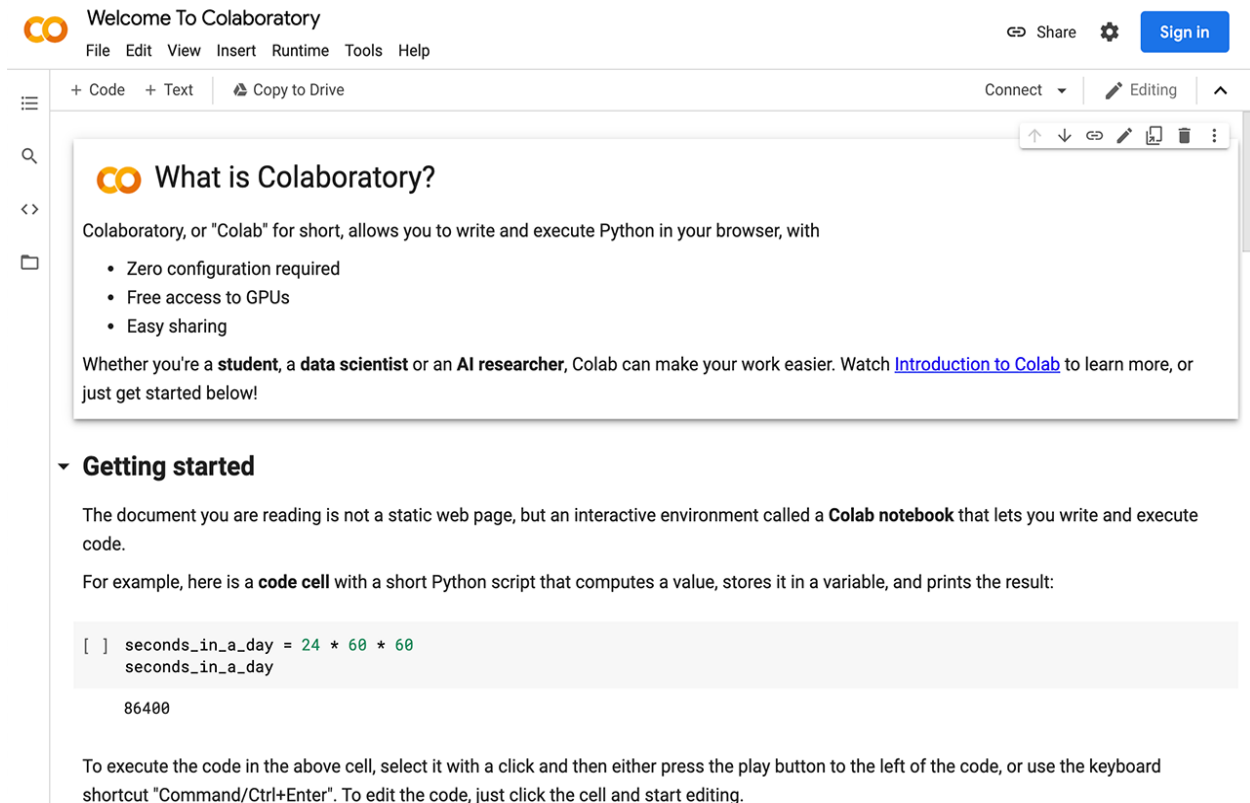
#### Google Colaboratory

Colab “Colab” short for Colaboratory, is a product from Google Research that runs completely in the cloud. Colab allows us to execute python code in the browser, and is mainly well suited for machine learning, data analysis and algorithms. On the subject of technical, Colab is a Jupyter notebook offers a hosted service which want no setup to use, at the same time as put forward loose get right of entry to computing resources comprising GPUs. Colab may be used perform the full strength of well-known Python libraries to investigate and visualize facts. With Colab you may import an image dataset, train an image classifier on it, and examine the model, all in only some lines of code. Colab notebooks execute code on Googles cloud servers, which means you have an advantage of Google hardware, as well as GPUs and TPUs, nevertheless of the power of your machine. All you need is a browser.

A programmer can perform the using Google Colab. We can write and execute code in Python in colab.

- Document your code that supports mathematical equations
- Create, Upload, Share notebooks
- Import and save notebook from or to Google Drive
- Import or Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV.

- Free Cloud service with free.



**Fig. 4.1. Google Colaboratory welcome page**

### Benefits of Google colab:

- 1) **Sharing:** You can share your Google Colab notebooks very easily. Thanks to Google Colab everyone with a Google account can just copy the notebook on his own Google Drive account. No need to install any modules to run any code, modules come preinstalled within Google Colab.
- 2) **Code snippets:** Google Colab has a great collection of snippets you can just plug in on your code. E.g. if you want to write data to a Google Sheet automatically, there's a snippet for it in the Google Library
- 3) **Performance:** Use the computing power of the Google servers instead of your own machine. Running python scripts requires often a lot of computing power and can take time. By running scripts in the cloud, you don't need to worry. Your local machine performance won't drop while executing your Python scripts.

### 4.3. Hardware Requirement:

PROCESSOR : INTEL CORE

SPEED : 2.49 GHz

RAM : 8GB DDR4 MEMORY

HDD : 250GB HDD

### HARDWARE SPECIFICATIONS

#### Processors:

Intel® Core™ i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2 cores, 2 threads per core), 8 GB of DRAM  
 Intel® Xeon® processor E5-2698 v3 at 2.30 GHz (2 sockets, 16 cores each, 1 thread per core), 64 GB of DRAM  
 Intel® Xeon Phi™ processor 7210 at 1.30 GHz (1 socket, 64 cores, 4 threads per core), 32 GB of DRAM, 16 GB of MCDRAM (flat mode enabled)

#### Minimum:

Intel Atom® processor or Intel® Core™ i3 processor

#### Disk space:

1 GB

#### Operating systems:

Windows® 10, macOS\*, and Linux\*

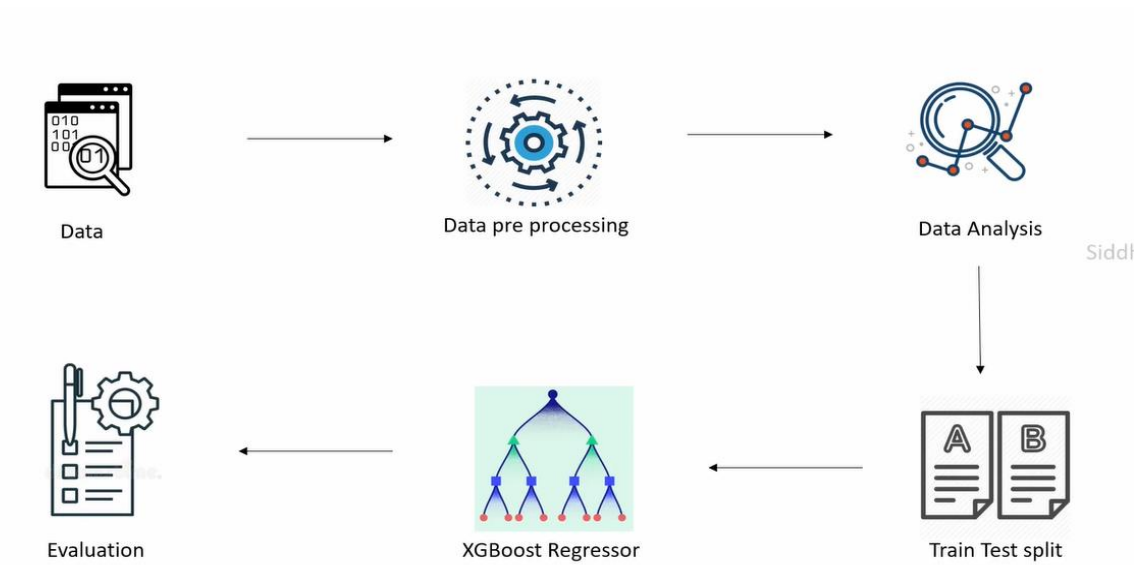
#### Minimum:

Linux, Windows 7 or Later



## 5. SYSTEM DESIGNS

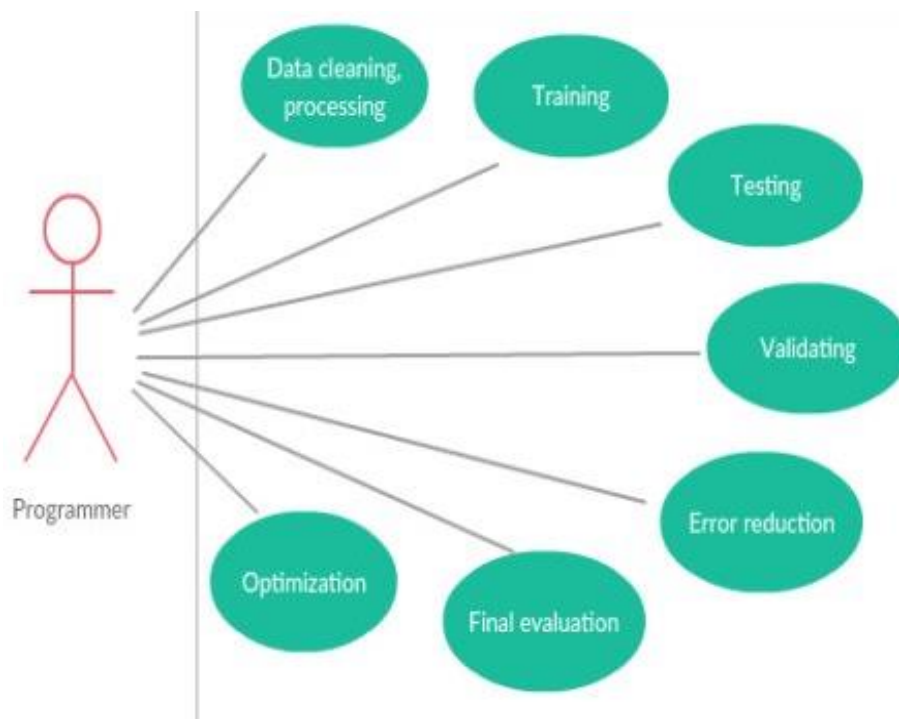
### 5.1. Work flow:



**Fig. 5.1 Work Flow Diagram (XGBoost)**

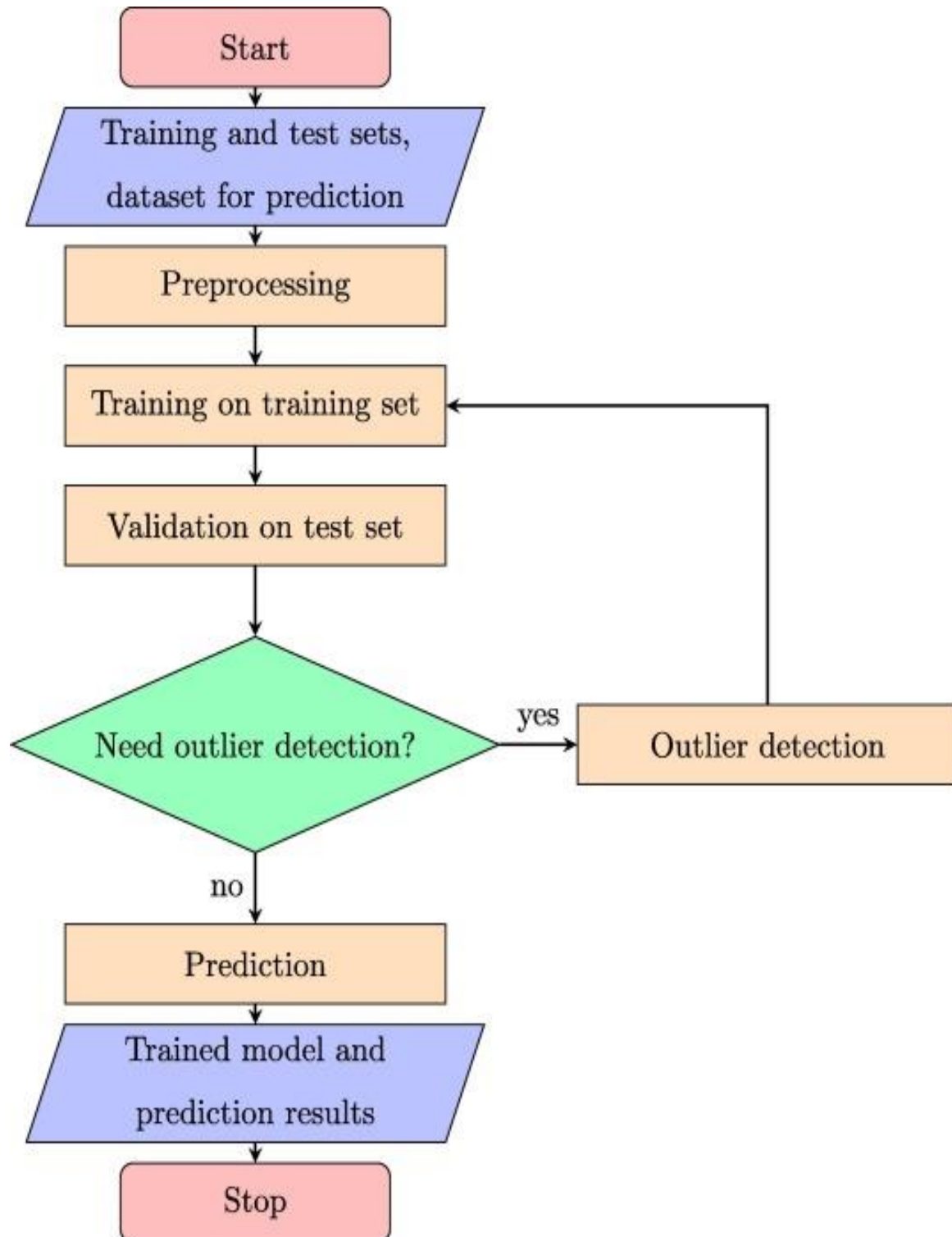
### 5.2. UML Diagrams:

Use-case diagram:



**Fig. 5.2. Use case Diagram**

Activity diagram:



**Fig. 5.3. Activity Diagram**

The variety of energy burned each day is immediately connected to weight loss, weight gain, or weight maintenance. To shed pounds, a person ought to burn greater calories than they take in, developing a calorie deficit. but, to do that, they want to recognize what number of calories they burn each day. Most people think about calories as most effective having to do with food and weight reduction. Calorie, a unit of energy or heat variously defined. Calorie may be defined as the amount of energy that is vital to increase 1 gram(g) of water by means of 1° C. This measurement can be carried out to lots of different strength releasing mechanisms outdoor of the human body. In the case of human body, calories are measure of how much energy the body requires to function. In order to be able to exercise how a whole lot of calories are burned every day is important to any man or woman trying to preserve, lose, or maintain weight. Understanding what elements contribute to calorie burning can help a person regulate their diet or workout program to deal with the aim. There are many factors that affect how much calories a person burns each day. some of the elements that effect day by day calorie burn aren't in a person's manage at the same time as others may be changed. These factors include:

In case of age, the older a person is, the fewer calories burned per day.

Gender: men burn greater energy than women.

Quantity of daily activity:

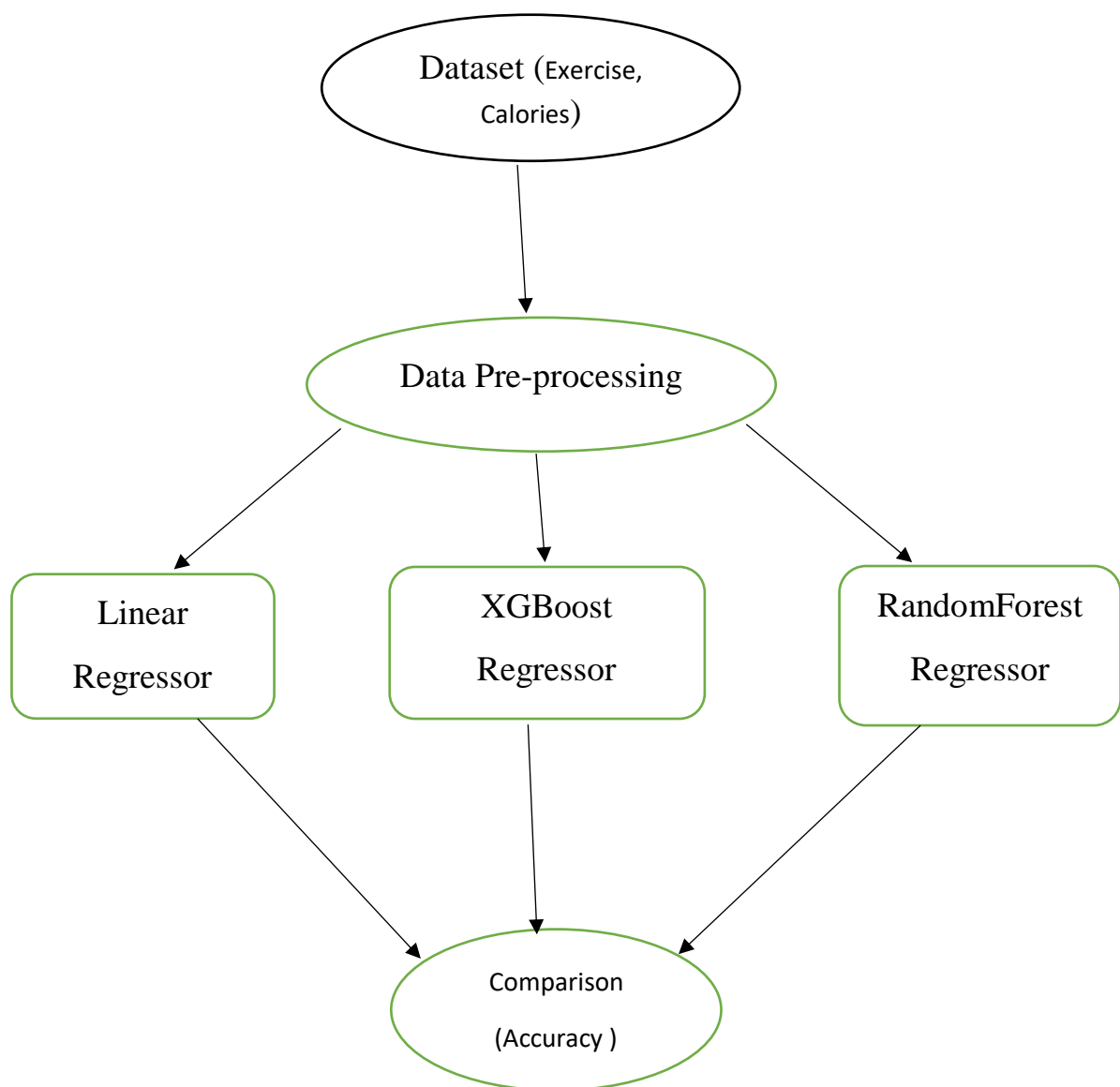
Body composition: those with more muscular tissues, burn more calories than people who've much less muscle.

Body size: larger people burn greater calories than smaller human beings, even at relaxation. Thermogenesis: that is the amount of strength our body uses to break down meals.

## 6. SYSTEM IMPLEMENTATION

The Exercise dataset with 8 independent variables and 1 dependent variable has been used for implementation. The prediction of burnt out calories is done with the following contributions. Firstly, the data set is pre-processed with Feature Scaling and missing values. Secondly, exploratory feature examination is done and the scattering of target highlight is visualized. Thirdly, the raw data set is fitted to all the regressors and the execution is dissected. Performance analysis is done using metrics like MAE.

### 6.1. Architecture Diagram:



**Fig. 6.1. Architecture of proposed work**

## **6.2. Modules and Module Descriptions:**

### **Dataset:**

The Exercise dataset with 8 independent variables and 1 dependent variable has been used for implementation. The two csv files are uploaded in colab which is used for processing. We use data frames for analysis and processing. It obtain some statistical measures about the data. The datasets that are used in this project are Exercise and Calories.

### **Data Pre-Processing:**

Two datasets are Concatenated initially. And then the number of rows and columns are checked in the concatenated dataset. After this, missing values are listed in order to find the null values in a dataset.

And the data is visualized according to age, height, weight. Then we study the correlation the various records and there are two types of correlation, positive and negative correlation.

### **Evaluation:**

Model evaluation is the process of using different evaluation metrics to understand a machine learning model's performance, as well as its strengths and weaknesses. Model evaluation is important to assess the efficacy of a model during initial research phases, and it also plays a role in model monitoring.

Now load the model Linear Regression and evaluate the prediction in test data. Similarly evaluate the prediction using XGB Regressor and Random Forest Regressor. Performance analysis is done using metrics like MAE.

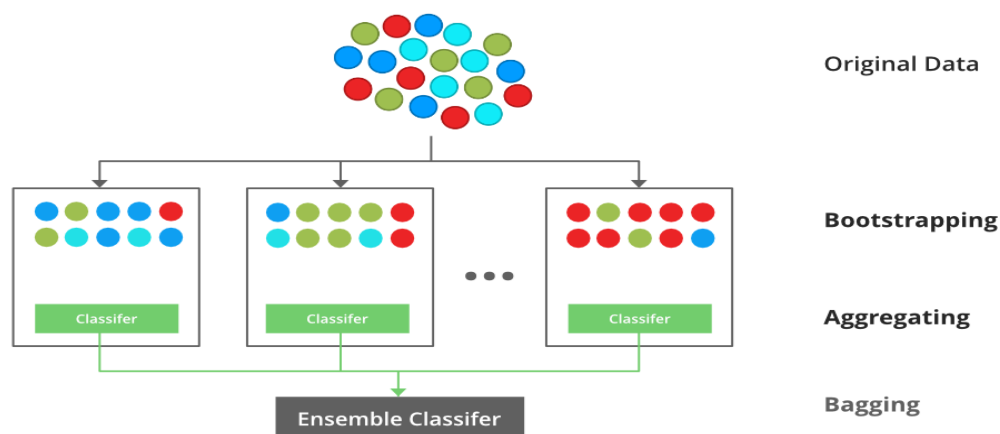
### **Comparison:**

From the performance analysis compare the error values and the accuracy of the three models to find the best algorithm.

### 6.3. Algorithms:

#### XGBoost Regressor:

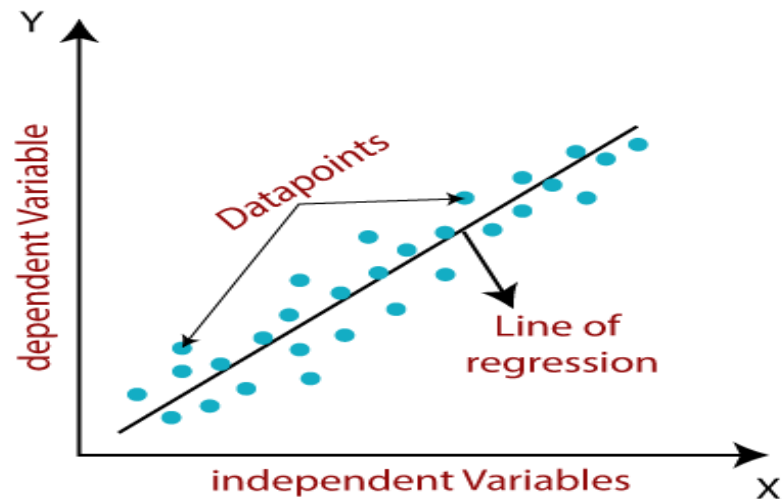
XBoost Regressor is a regression algorithm that is an analysis which is a statistical technique to model the connection between a dependent (target) and independent (predictor) variables with one or more unbiased variables. In machine learning the XGBoost algorithm performs well since it has robust handling of many variety of data types, relationships, distributions, and the many hyperparameters that you can fine-tune. XGBoost regressor can be used for regression, classification for both binary and multiclass, and ranking problems.



**Fig. 6.2 XGBoost Regressor**

#### Linear Regression:

In Machine Learning, Linear Regression is a supervised learning algorithm. In this the linear equation specify the relationship amongst one or extra predictor variables and one outcome variable and combines a specific set of input values (x) to which is the predicted output for that set of input values (y). The equation can be written as  $Y = a + bX$ . Steps: A. Collect Dataset B Data Pre-processing C Data Analysis D. Machine learning model E. Evaluation



**Fig. 6.3 Linear Regression**

Mathematically, we can represent a linear regression as

$$y = a_0 + a_1x + \varepsilon$$

Y= Dependent Variable(Target Variable)

X= Independent Variable(Predictor Variable)

$a_0$ = Intercept of the line(Gives an addition degree of freedom)

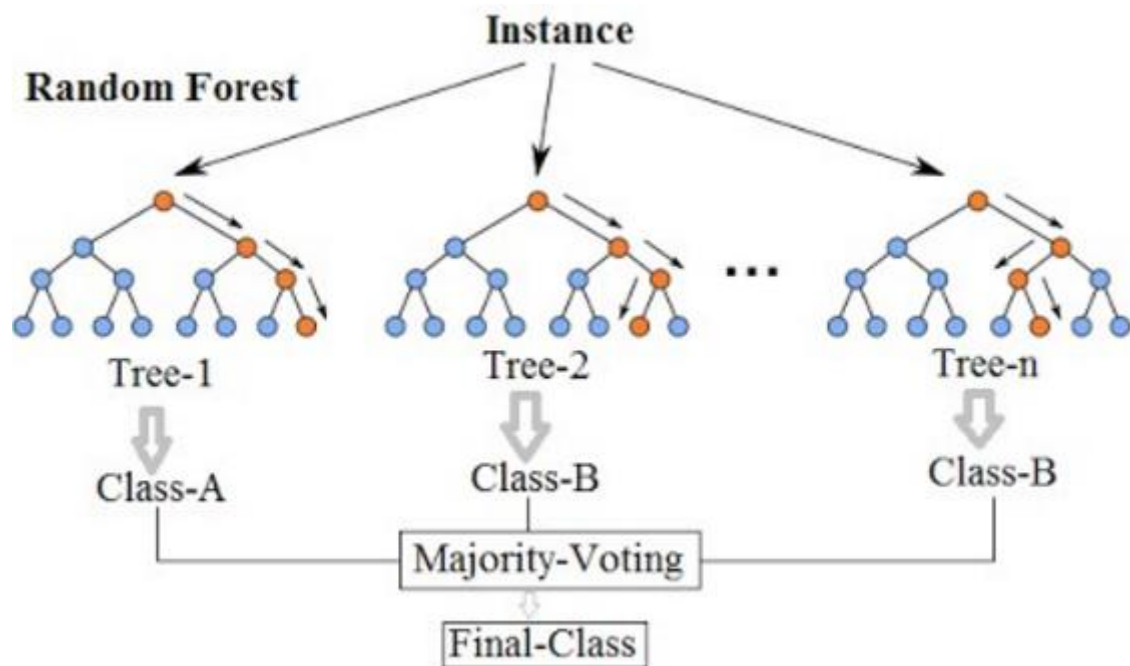
$a_1$ = Linear regression coefficient(scale factor to each input value)

$\varepsilon$ = random error

The values for x and y variables are training datasets for Linear Regression model representation.

#### Random Forest Regression:

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.



**Fig. 6.4 Random Forest Regression**

## 6.4. Python modules:

Modules that are imported to predict the calories burned from the data sets are

- ✓ numpy
- ✓ pandas
- ✓ matplotlib.pyplot
- ✓ seaborn
- ✓ sklearn.model\_selection. train\_test\_split
- ✓ sklearn.ensemble. RandomForestRegressor
- ✓ sklearn.linear\_model. LinearRegression
- ✓ xgboost. XGBRegressor
- ✓ Sklearn. metrics



## **Python Modules Description:**

### **Numpy:**

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. **Numeric**, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities.

### **Operations using NumPy:**

Using NumPy, a developer can perform the following operations –

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy – A Replacement for MatLab

### **Pandas:**

Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labelled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way towards this goal.

Main Features

Here are just a few of the things that pandas does well:

- Easy handling of missing data (represented as NaN, NA, or NaT) in floating point as well as non-floating point data.

- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects.
- Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations.
- Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data.
- Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into Data Frame objects.
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets
- Intuitive merging and joining data sets.
- Flexible reshaping and pivoting of data sets.
- Hierarchical labelling of axes (possible to have multiple labels per tick).
- Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving/loading data from the ultrafast HDF5 format.
- Time series-specific functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging.

### **matplotlib.pyplot:**

Pyplot is an API (Application Programming Interface) for Python's matplotlib that effectively makes matplotlib a viable open source alternative to MATLAB. Matplotlib is a library for data visualization, typically in the form of plots, graphs and charts.

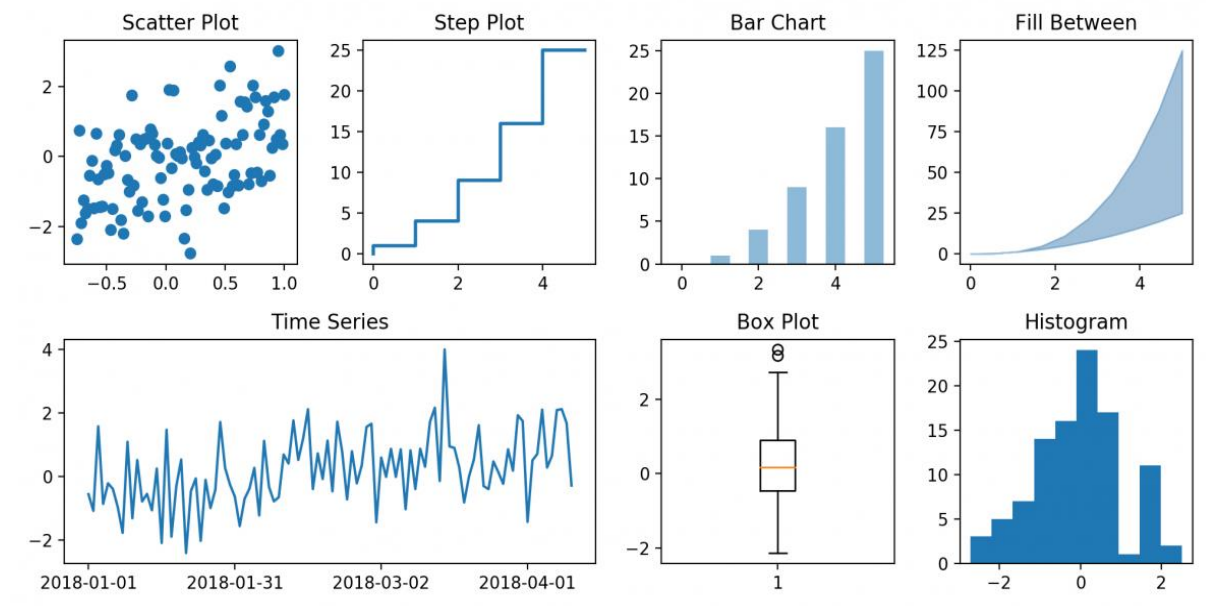
Pyplot API Structure:

Pyplot provides matplotlib with two key features:

- A MATLAB-style interface, which allows those familiar with MATLAB to adopt Python more easily
- Statefulness, which means that pyplot stores the state of an object when you first plot it. This is essential for use in the same loop or session state until `plt.close()` is encountered in the code. State can also be important when creating several plots continuously.

The pyplot API consists of a hierarchy of Python code objects, and includes numerous functions topped by `matplotlib.pyplot`. This stack can be viewed as having three interdependent layers:

- Scripting layer – used to define a figure, which contains one or more plots, which consist of axes (i.e., x axis ,y axis, and possibly z axis)
- Artist Layer – used to manipulate elements of a plot, such as adding labels, drawing lines, etc.
- Backend Layer – used to format the plot for display in a specific target application, such as a Jupyter Notebook.



**Fig. 6.5. pyplot graphs**

### **Seaborn:**

Seaborn is a Python data visualization library based on [matplotlib](#). It provides a high-level interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the introductory notes or the paper. Visit the installation page to see how you can download the package and get started with it. You can browse the example gallery to see some of the things that you can do with seaborn, and then check out the tutorial or API reference to find out how.

To see the code or report a bug, please visit the GitHub repository. General support questions are most at home on stack overflow or discourse, which have dedicated channels for seaborn. Seaborn helps you explore and understand your data. Its plotting functions operate on data frames 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.

### **sklearn.model\_selection.train\_test\_split:**

Split arrays or matrices into random train and test subsets.

Quick utility that wraps input validation and `next(ShuffleSpilt().split(X,y))` and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

Read more in the User Guide.

Parameters:

- `*arrays`: sequence of indexables with same length / `shape[0]`-Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.
- `test_size`: float or int, default=None- If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is set to the complement of the train size. If `train_size` is also None, it will be set to 0.25.
- `train_size`: float or int, default=None-If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size.

- `random_state`: int, `RandomState` instance or `None`, `default=None`- Controls the shuffling applied to the data before applying the split. Pass an int for reproducible output across multiple function calls. See Glossary.
- `Shuffle` : bool, `default=True`- Whether or not to shuffle the data before splitting. If `shuffle=False` then `stratify` must be `None`.
- `Stratify`: array-like, `default=None`-If not `None`, data is split in a stratified fashion, using this as the class labels. Read more in the User Guide.

Returns:

Splitting: list, `length=2 * len(arrays)`-List containing train-test split of inputs. New in version 0.16: If the input is sparse, the output will be a `scipy.sparse.csr_matrix` . Else, output type is the same as the input type.

### **sklearn.ensemble.RandomForestRegressor:**

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True`(default), otherwise the whole dataset is used to build each tree.

### **sklearn.linear\_model.LinearRegression:**

```
class sklearn.linear_model.LinearRegression(*, fit_intercept=True, normalize='deprecated',
copy_X=True, n_jobs=None, positive=False) [source]
```

Ordinary least squares Linear Regression.

Linear Regression fits a linear model with coefficients  $w = (w_1, \dots, w_p)$  to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. Linear regression algorithm shows a linear relationship between a dependent ( $y$ ) and one or more independent ( $x$ ) variables, hence called as linear regression.

Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable. Linear regression model provides a sloped straight line representing the relationship between the variables.

### **XGBoost:**

Extreme Gradient Boosting (XGBoost) is an open-source library that provides an efficient and effective implementation of the gradient boosting algorithm.

Shortly after its development and initial release, XGBoost became the go-to method and often the key component in winning solutions for a range of problems in machine learning competitions.

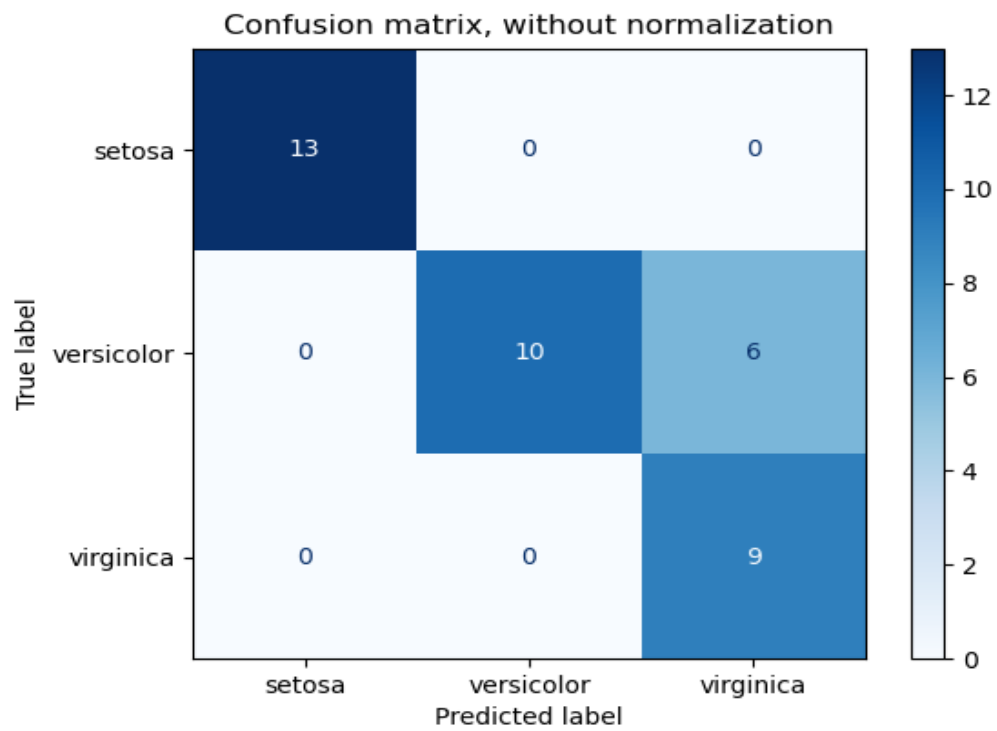
Regression predictive modelling problems involve predicting a numerical value such as a dollar amount or a height. XGBoost can be used directly for regression predictive modelling.

In this tutorial, you will discover how to develop and evaluate XGBoost regression models in Python.

- XGBoost is an efficient implementation of gradient boosting that can be used for regression predictive modelling.
- How to evaluate an XGBoost regression model using the best practice technique of repeated k-fold cross-validation.
- How to fit a final model and use it to make a prediction on new data.

### **sklearn.metrics:**

The sklearn.metrics module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values. Most implementations allow each sample to provide a weighted contribution to the overall score, through the `sample_weight` parameter.



**Fig. 6.6. sklearn.metrics sample graph**

## 7. EXPERIMENTAL RESULTS

Here we are using 3 kinds of algorithms on given data set in order make out a useful model for predicting the calories burned during the workout based on the

- workout duration,
- age,
- gender,
- height and
- weight of the person.
- Heart rate
- Body temperature

### 7.1. Data Source

The repository that we used for dataset is Kaggle. There are two csv files which contains 15000 instances and 7 attributes. The data set from Kaggle repository contains attributes of each person's details including their gender, age, workout duration, heart rate, body temperature, height and weight. This dataset is taken as the training data. And the second calories dataset contains target class which have the calories burned by corresponding person.

#### **Input Attributes: Function**

Gender: gender (male : 0,female : 1)

Age: age mentioned in years

Height: height of the person

Weight: weight of the person duration The time taken to complete the exercising in minutes.

heart\_rate: Average heart rate during the workout (more than normal rate 75 beats/min)

body\_temp:Body temperature in the course of the workout(greater than 37 degree celsius)

calories:Total calories burned during the exercise.



Exercise dataset:

|    | A        | B      | C   | D      | E      | F        | G         | H         | I |
|----|----------|--------|-----|--------|--------|----------|-----------|-----------|---|
| 1  | User_ID  | Gender | Age | Height | Weight | Duration | Heart_Rat | Body_Temp |   |
| 2  | 14733363 | male   | 68  | 190    | 94     | 29       | 105       | 40.8      |   |
| 3  | 14861698 | female | 20  | 166    | 60     | 14       | 94        | 40.3      |   |
| 4  | 11179863 | male   | 69  | 179    | 79     | 5        | 88        | 38.7      |   |
| 5  | 16180408 | female | 34  | 179    | 71     | 13       | 100       | 40.5      |   |
| 6  | 17771927 | female | 27  | 154    | 58     | 10       | 81        | 39.8      |   |
| 7  | 15130815 | female | 36  | 151    | 50     | 23       | 96        | 40.7      |   |
| 8  | 19602372 | female | 33  | 158    | 56     | 22       | 95        | 40.5      |   |
| 9  | 11117088 | male   | 41  | 175    | 85     | 25       | 100       | 40.7      |   |
| 10 | 12132339 | male   | 60  | 186    | 94     | 21       | 97        | 40.4      |   |
| 11 | 17964668 | female | 26  | 146    | 51     | 16       | 90        | 40.2      |   |
| 12 | 13723164 | female | 36  | 177    | 76     | 1        | 74        | 37.8      |   |
| 13 | 13681290 | female | 21  | 157    | 56     | 17       | 100       | 40        |   |
| 14 | 15566424 | male   | 66  | 171    | 79     | 11       | 90        | 40        |   |
| 15 | 12891699 | female | 32  | 157    | 54     | 18       | 93        | 40.4      |   |
| 16 | 13823829 | male   | 53  | 182    | 85     | 2        | 82        | 38.1      |   |
| 17 | 17557348 | female | 39  | 156    | 62     | 28       | 104       | 40.8      |   |
| 18 | 12198133 | male   | 39  | 182    | 82     | 4        | 82        | 38.6      |   |
| 19 | 15236104 | male   | 46  | 169    | 67     | 11       | 89        | 40.2      |   |
| 20 | 11042324 | female | 27  | 171    | 65     | 4        | 85        | 38.6      |   |
| 21 | 16864285 | male   | 50  | 188    | 86     | 14       | 94        | 40.2      |   |

<
>
exercise
+

Table. 7.1

Exercise Dataset

Calories Dataset:

|    | A        | B        | C |
|----|----------|----------|---|
| 1  | User_ID  | Calories |   |
| 2  | 14733363 | 231      |   |
| 3  | 14861698 | 66       |   |
| 4  | 11179863 | 26       |   |
| 5  | 16180408 | 71       |   |
| 6  | 17771927 | 35       |   |
| 7  | 15130815 | 123      |   |
| 8  | 19602372 | 112      |   |
| 9  | 11117088 | 143      |   |
| 10 | 12132339 | 134      |   |
| 11 | 17964668 | 72       |   |
| 12 | 13723164 | 3        |   |
| 13 | 13681290 | 92       |   |
| 14 | 15566424 | 58       |   |
| 15 | 12891699 | 88       |   |
| 16 | 13823829 | 7        |   |
| 17 | 17557348 | 170      |   |
| 18 | 12198133 | 11       |   |
| 19 | 15236104 | 43       |   |
| 20 | 11042324 | 15       |   |
| 21 | 16864285 | 74       |   |
|    |          | calories | + |

Table. 7.2

Calories Dataset

## 7.2. Screenshots:

Packages:

Importing all the required packages and modules.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn import metrics
```

**Fig. 7.1. Importing modules**

Dataset:

There are two dataset csv files which should be uploaded to colab which is used for processing. We use data frames for analysis and processing. It obtain some statistical measures about the data

```
# LOADING THE DATA FROM CSV FILE TO A PANDAS DATAFRAME
calories=pd.read_csv('/content/calories.csv')
```

```
# print the first 5 rows of the dataframe
calories.head()
```


|   | User_ID  | Calories |
|---|----------|----------|
| 0 | 14733363 | 231.0    |
| 1 | 14861698 | 66.0     |
| 2 | 11179863 | 26.0     |
| 3 | 16180408 | 71.0     |
| 4 | 17771927 | 35.0     |

**Table. 7.3 First 5 rows of dataframe (calories)**

To print the first 5 values of the table head() method is used.

```
exercise_data=pd.read_csv('/content/exercise.csv')
```

```
exercise_data.head()
```




|   | User_ID  | Gender | Age | Height | Weight | Duration | Heart_Rate | Body_Temp |
|---|----------|--------|-----|--------|--------|----------|------------|-----------|
| 0 | 14733363 | male   | 68  | 190.0  | 94.0   | 29.0     | 105.0      | 40.8      |
| 1 | 14861698 | female | 20  | 166.0  | 60.0   | 14.0     | 94.0       | 40.3      |
| 2 | 11179863 | male   | 69  | 179.0  | 79.0   | 5.0      | 88.0       | 38.7      |
| 3 | 16180408 | female | 34  | 179.0  | 71.0   | 13.0     | 100.0      | 40.5      |
| 4 | 17771927 | female | 27  | 154.0  | 58.0   | 10.0     | 81.0       | 39.8      |

**Table. 7.4. First 5 rows of dataframe(exercise)**

Concatenating datasets:

```
calories_data=pd.concat([exercise_data,calories['Calories']],axis=1)
```

```
calories_data.head()
```



|   | User_ID  | Gender | Age | Height | Weight | Duration | Heart_Rate | Body_Temp | Calories |
|---|----------|--------|-----|--------|--------|----------|------------|-----------|----------|
| 0 | 14733363 | male   | 68  | 190.0  | 94.0   | 29.0     | 105.0      | 40.8      | 231.0    |
| 1 | 14861698 | female | 20  | 166.0  | 60.0   | 14.0     | 94.0       | 40.3      | 66.0     |
| 2 | 11179863 | male   | 69  | 179.0  | 79.0   | 5.0      | 88.0       | 38.7      | 26.0     |
| 3 | 16180408 | female | 34  | 179.0  | 71.0   | 13.0     | 100.0      | 40.5      | 71.0     |
| 4 | 17771927 | female | 27  | 154.0  | 58.0   | 10.0     | 81.0       | 39.8      | 35.0     |

**Table. 7.5. First 5 rows of Concatenated dataset**

Checking number of rows and columns:

```
#checking number of rows and columns
calories_data.shape

(15000, 9)
```

**Fig. 7.2. Checking number of rows and columns**

Here, 15000 rows are available in the table. And 9 columns are available in the concatenated table.

Some information about data:

```
#getting some information about the data
calories_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 9 columns):
#   Column      Non-Null Count  Dtype
---  -
0   User_ID     15000 non-null  int64
1   Gender      15000 non-null  object
2   Age         15000 non-null  int64
3   Height      15000 non-null  float64
4   Weight      15000 non-null  float64
5   Duration    15000 non-null  float64
6   Heart_Rate  15000 non-null  float64
7   Body_Temp   15000 non-null  float64
8   Calories    15000 non-null  float64
dtypes: float64(6), int64(2), object(1)
memory usage: 1.0+ MB
```

**Fig. 7.3. Data Information**

The info() returns the number of null values and the total values and the type of the each column.

Checking for missing values:

```
#checking for missing values
calories_data.isnull().sum()
```

```
User_ID      0
Gender       0
Age          0
Height       0
Weight       0
Duration     0
Heart_Rate   0
Body_Temp    0
Calories     0
dtype: int64
```

**Fig. 7.4. Checking Missing values**

Here every row and column contain a value. So, no column or row is null.

Statistical measures about data:

```
#get some statistical measures about the data
calories_data.describe()
```

|              | User_ID      | Age          | Height       | Weight       | Duration     | Heart_Rate   | Body_Temp    | Calories     |
|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| <b>count</b> | 1.500000e+04 | 15000.000000 | 15000.000000 | 15000.000000 | 15000.000000 | 15000.000000 | 15000.000000 | 15000.000000 |
| <b>mean</b>  | 1.497736e+07 | 42.789800    | 174.465133   | 74.966867    | 15.530600    | 95.518533    | 40.025453    | 89.539533    |
| <b>std</b>   | 2.872851e+06 | 16.980264    | 14.258114    | 15.035657    | 8.319203     | 9.583328     | 0.779230     | 62.456978    |
| <b>min</b>   | 1.000116e+07 | 20.000000    | 123.000000   | 36.000000    | 1.000000     | 67.000000    | 37.100000    | 1.000000     |
| <b>25%</b>   | 1.247419e+07 | 28.000000    | 164.000000   | 63.000000    | 8.000000     | 88.000000    | 39.600000    | 35.000000    |
| <b>50%</b>   | 1.499728e+07 | 39.000000    | 175.000000   | 74.000000    | 16.000000    | 96.000000    | 40.200000    | 79.000000    |
| <b>75%</b>   | 1.744928e+07 | 56.000000    | 185.000000   | 87.000000    | 23.000000    | 103.000000   | 40.600000    | 138.000000   |
| <b>max</b>   | 1.999965e+07 | 79.000000    | 222.000000   | 132.000000   | 30.000000    | 128.000000   | 41.500000    | 314.000000   |

**Table. 7.6. Statistical measurement**

Here the mean value of age is 42.7 and we have the standard deviation and the percentile information. The body temperature is about 40.

The body temperature will be more in those people who are doing workout. The crucial insights for this analysis is coronary heart rate and temperature.

Subsequently we need to visualize the data via using a few plots and graphs.

## Data visualization

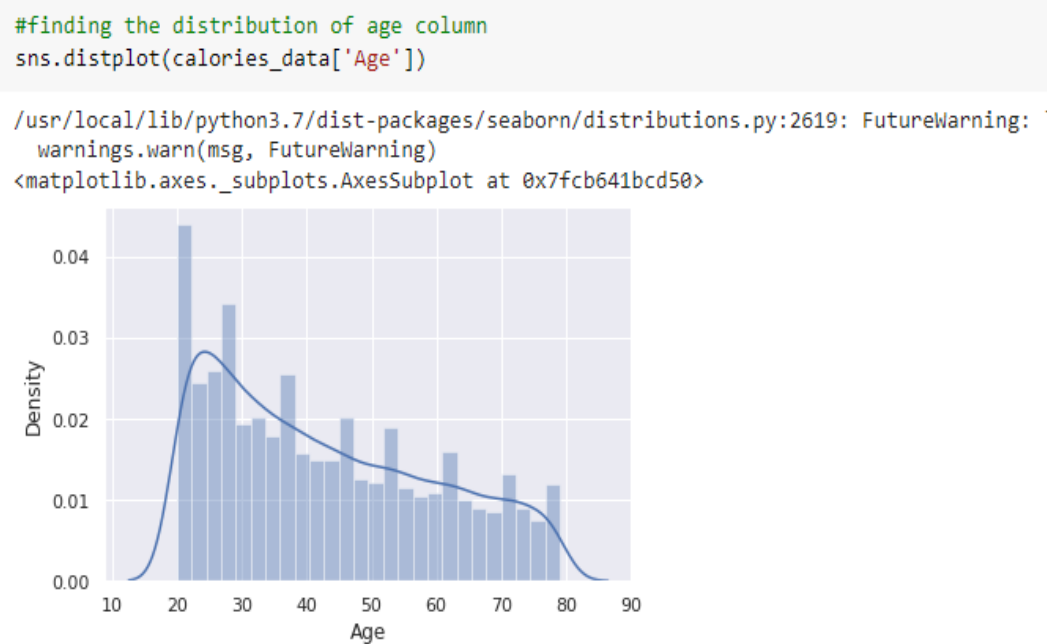
### Age:



**Fig. 7.5. Plotting Gender**

The Gender column in the calories\_data is plotted using the countplot() in Seaborn.

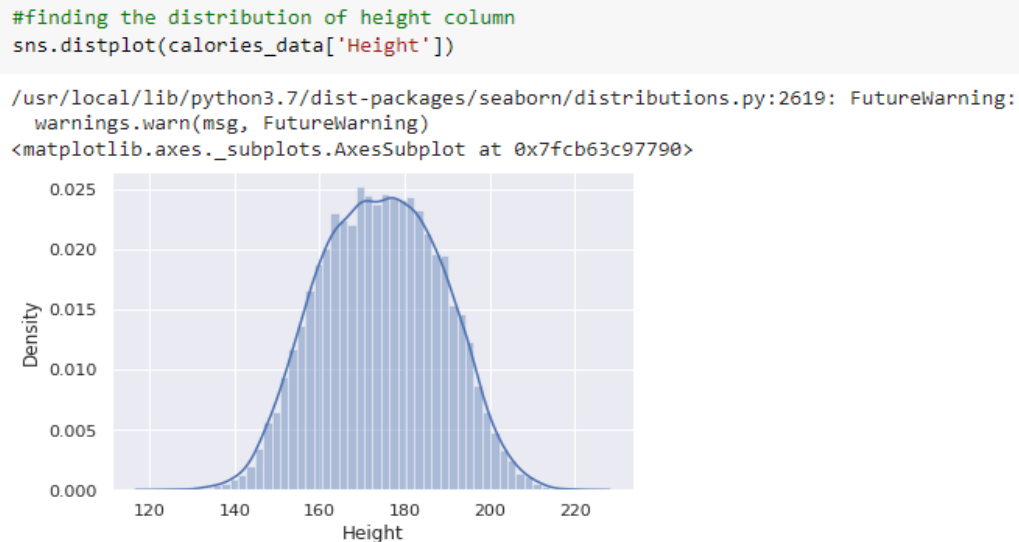
### Distribution of age column:



**Fig. 7.6. Plotting age**

For age, the more values in the range of 20 and 30 where we can see a peak in the curve means from 15000 instances more are present in this range. Then there is a decreasing means less people tend to do workout at older age.

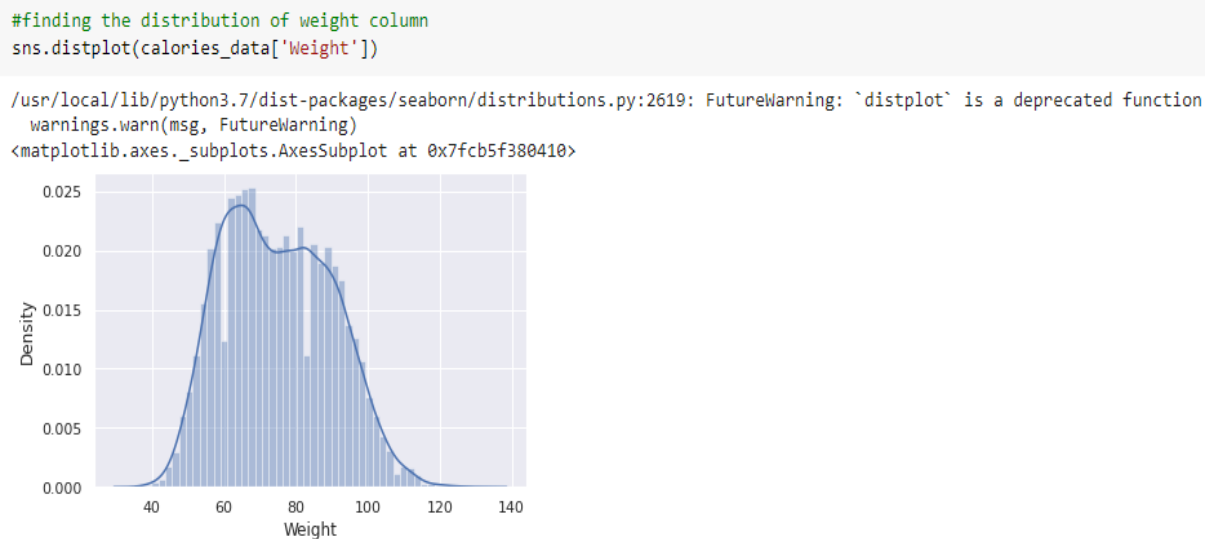
#### Distribution of height column:



**Fig. 7.7. Plotting height**

For height, the more values in the range of 0.020 and 0.025 where we can see a peak in the curve means from 15000 instances more are present in this range. Then there is a decreasing means less people tend to do workout at lesser height.

#### Distribution of weight column:

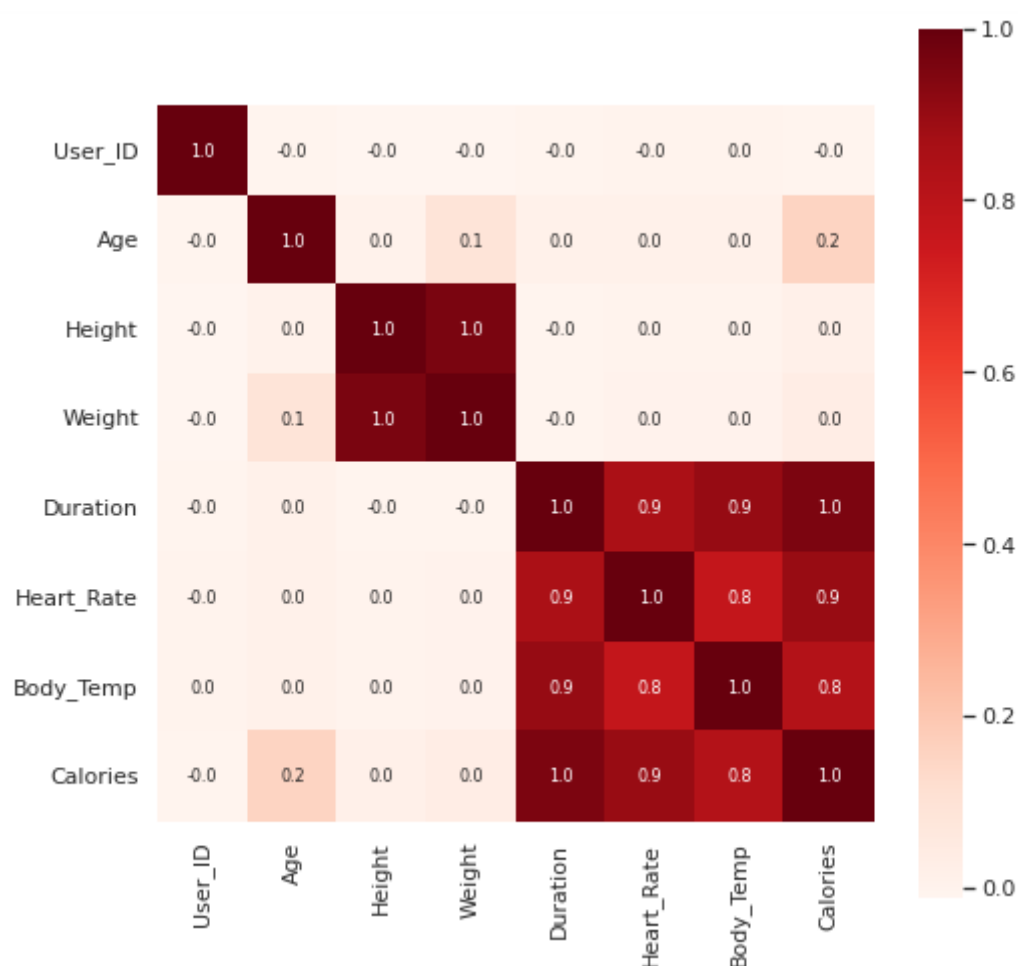


**Fig. 7.8. Plotting weight**



For weight, the more values in the range of 0.020 and 0.025 where we can see a peek in the curve means from 15000 instances more are present in this range.

Then we study the correlation the various records and there are two types of correlation , positive and negative correlation. The duration for which the person doing the workout is more then the number of calories the person burnt will also be more. So those values are immediately proportional which is inside the equal direction are undoubtedly correlated. need to visualize the data via using a few plots and graphs.



**Fig. 7.9. Correlation graph**

From this we get that the weight and weight are positively correlated and the duration, heart rate and body temperature are highly positively correlated with calories. Now split the

data into training and test data by taking the two variables X and Y that is to separate for features and target.

Replacing some data:

```
calories_data.replace({'Gender':{'male':0,'female':1}},inplace=True)
```

```
calories_data.head()
```

|   | User_ID  | Gender | Age | Height | Weight | Duration | Heart_Rate | Body_Temp | Calories |
|---|----------|--------|-----|--------|--------|----------|------------|-----------|----------|
| 0 | 14733363 | 0      | 68  | 190.0  | 94.0   | 29.0     | 105.0      | 40.8      | 231.0    |
| 1 | 14861698 | 1      | 20  | 166.0  | 60.0   | 14.0     | 94.0       | 40.3      | 66.0     |
| 2 | 11179863 | 0      | 69  | 179.0  | 79.0   | 5.0      | 88.0       | 38.7      | 26.0     |
| 3 | 16180408 | 1      | 34  | 179.0  | 71.0   | 13.0     | 100.0      | 40.5      | 71.0     |
| 4 | 17771927 | 1      | 27  | 154.0  | 58.0   | 10.0     | 81.0       | 39.8      | 35.0     |

**Table. 7.7. Replaced (Gender) table**

Here the male and female values are replaced by the 0 and 1 to have easy calculation.

Male =0 and Female=1

Separating the features and targets:

```
X=calories_data.drop(columns=['User_ID','Calories'],axis=1)
Y=calories_data['Calories']
```

```
print(X)
```

|       | Gender | Age | Height | Weight | Duration | Heart_Rate | Body_Temp |
|-------|--------|-----|--------|--------|----------|------------|-----------|
| 0     | 0      | 68  | 190.0  | 94.0   | 29.0     | 105.0      | 40.8      |
| 1     | 1      | 20  | 166.0  | 60.0   | 14.0     | 94.0       | 40.3      |
| 2     | 0      | 69  | 179.0  | 79.0   | 5.0      | 88.0       | 38.7      |
| 3     | 1      | 34  | 179.0  | 71.0   | 13.0     | 100.0      | 40.5      |
| 4     | 1      | 27  | 154.0  | 58.0   | 10.0     | 81.0       | 39.8      |
| ...   | ...    | ... | ...    | ...    | ...      | ...        | ...       |
| 14995 | 1      | 20  | 193.0  | 86.0   | 11.0     | 92.0       | 40.4      |
| 14996 | 1      | 27  | 165.0  | 65.0   | 6.0      | 85.0       | 39.2      |
| 14997 | 1      | 43  | 159.0  | 58.0   | 16.0     | 90.0       | 40.1      |
| 14998 | 0      | 78  | 193.0  | 97.0   | 2.0      | 84.0       | 38.3      |
| 14999 | 0      | 63  | 173.0  | 79.0   | 18.0     | 92.0       | 40.5      |

```
[15000 rows x 7 columns]
```

**Fig. 7.10. Separated X values**

The value X has the dataset values in the exercise.csv.

```
| print(Y)
0      231.0
1       66.0
2       26.0
3       71.0
4       35.0
...
14995   45.0
14996   23.0
14997   75.0
14998   11.0
14999   98.0
Name: Calories, Length: 15000, dtype: float64
```

**Fig. 7.11. Separated Y values**

The value Y has the dataset values in the calories.csv

Now the data is Split into training data and test data

```
X_train, X_test, Y_train, Y_test= train_test_split(X, Y, test_size=0.2, random_state=2)

print(X.shape,X_train.shape,X_test.shape)

(15000, 7) (12000, 7) (3000, 7)
```

**Fig. 7.12. Testing and Training**

The dataset is splitted with 7:3 ratio for training and testing dataset

- Total number of row and column values are 15000 and 7.
- The number of values that are taken for training is 12000 and 7.
- The number of values that are taken for testing is 3000 and 7.

## Evaluation:

### Linear Regression:

```

▶ model1=LinearRegression()

[ ] model1.fit(X_train, Y_train)

    LinearRegression()

[ ] test_data_prediction= model1.predict(X_test)

[ ] print(test_data_prediction)

    [137.49241057 182.18166512  50.15864741 ... 157.56637167  16.54924422
    100.21047087]

[ ] mae=metrics.mean_absolute_error(Y_test, test_data_prediction)

[ ] print("Mean Absolute Error", mae)

    Mean Absolute Error 8.385188053147187

```

**Fig. 7.13. Linear Regression MAE**

Then load the model Linear Regression and evaluate the prediction on test data. The model goes through this test data and calories burned for the X\_test. And compare the values predicted by our model with original values. For this we use the metrics-mean absolute blunders which tells what is the magnitude of mistakes the version is making `metrics.mean_absolute_error(Y_test,test_data_prediction)`.

The mean absolute error is **8.38**.

## XGB Regressor:

```
[ ] #loading the model
    model2=XGBRegressor()

[ ] #traing the model with X_train
    model2.fit(X_train, Y_train)

[ ] test_data_prediction= model2.predict(X_test)

[ ] print(test_data_prediction)

    [129.06204  223.79721   39.181965 ... 145.59767   22.53474   92.29064 ]

[ ] mae=metrics.mean_absolute_error(Y_test, test_data_prediction)

[ ] print("Mean Absolute Error", mae)

    Mean Absolute Error 2.7159012502233186

[ ] model2.score(X_train, Y_train)

    0.9966777021480265
```

**Fig. 7.14. XGB Regressor MAE**

Then load the model XGB Regressor and evaluate the prediction on test data. The model goes through this test data and calories burned for the X\_test. And compare the values predicted by our model with original values. For this we use the metrics-mean absolute blunders which tells what is the magnitude of mistakes the version is making `metrics.mean_absolute_error(Y_test,test_data_prediction)`.

The mean absolute error is **2.71**

## Random Forest Regressor:

```
[ ] model3= RandomForestRegressor()

[ ] model3.fit(X_train, Y_train)

[ ] test_data_prediction= model3.predict(X_test)

[ ] print(test_data_prediction)
[128.97 223.19 36.72 ... 147.74 23.91 88.8 ]

[ ] mae=metrics.mean_absolute_error(Y_test, test_data_prediction)

[ ] print("Mean Absolute Error", mae)
Mean Absolute Error 1.7007933333333332

[ ] m3=model3.score(X_train, Y_train)

[ ] model3.score(X_train, Y_train)
0.9996839231056824
```

**Fig. 7.15. Random Forest Regressor MAE**

Then load the model XGB Regressor and evaluate the prediction on test data. The model goes through this test data and calories burned for the X\_test. And compare the values predicted by our model with original values. For this we use the metrics-mean absolute blunders which tells what is the magnitude of mistakes the version is making  
`metrics.mean_absolute_error(Y_test,test_data_prediction)`

The mean absolute error is **1.700**

## Algorithms accuracy:

### ➤ Linear Regressor:

Accuracy:96%

### ➤ XGB Regressor:

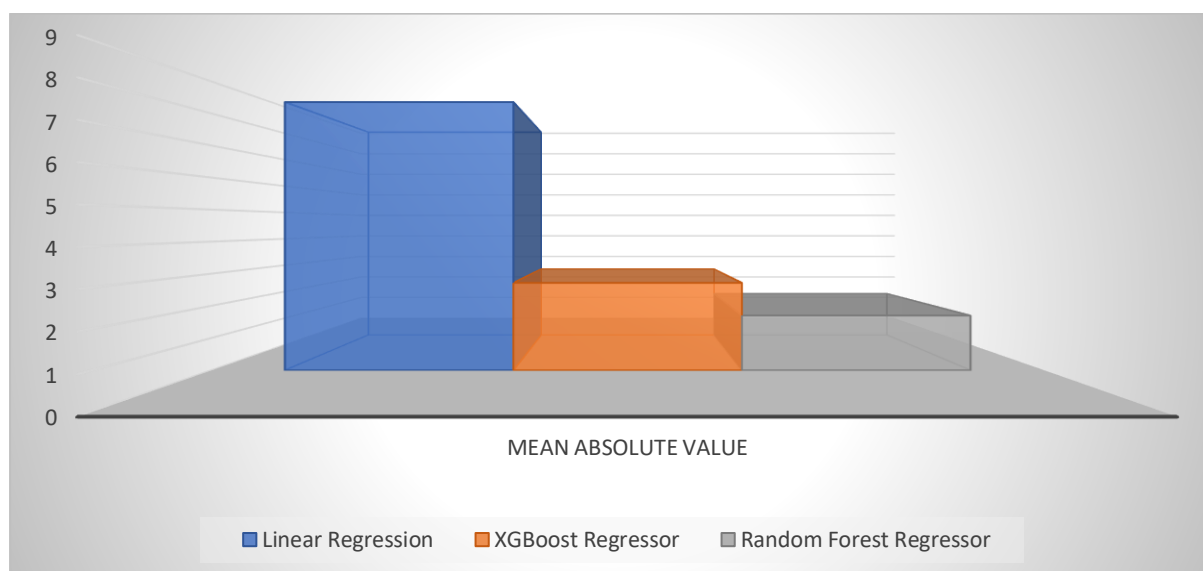
Accuracy:99.6%

### ➤ Random Forest Regressor:

Accuracy:99.9%

## 8. RESULTS AND FINDINGS

The analysis of this dataset was done to predict the calories burned depends on the duration of workout and also based on the gender, age, body temperature and heart rate at some stage in the exercise. By using these machine learning algorithms we are looking for a machine learning model with less mean absolute error, which gives more accurate results. By comparing the three models, XGB regressor, Random Forest Regressor and Linear regression we get that the Random Forest Regressor gives the more accurate results of the calories burned with a mean absolute error of 1.700 than the linear regressor.



**Fig. 7.1. Comparison graph**

## **9. CONCLUSION:**

From the analysis we met with a conclusion that the Random Forest Regressor has more accurate results than the Linear regression model and XGB regressor. Mean absolute error imply absolute error ought to be as low as viable. It is not anything but the difference between the actual and predicted values through the models. The mean absolute error value that is getting in Random Forest Regressor is 1.700 which is a good value. The error value is very less. Therefore, we can conclude that the best model for the calories burn prediction is Random Forest Regressor.



## APPENDIX (Source code):

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn import metrics

calories=pd.read_csv('/content/calories.csv')
calories.head()

exercise_data=pd.read_csv('/content/exercise.csv')
exercise_data.head()

calories_data=pd.concat([exercise_data,calories['Calories']], axis=1)
calories_data.head()

calories_data.shape

calories_data.info()

calories_data.isnull().sum()

calories_data.describe()

sns.set()
sns.countplot(calories_data['Gender'])
```

```

sns.distplot(calories_data['Age'])
sns.distplot(calories_data['Height'])
sns.distplot(calories_data['Weight'])

correlation=calories_data.corr()

plt.figure(figsize=(8,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size':8
}, cmap='Reds')

calories_data.replace({'Gender':{'male':0,'female':1}},inplace=True)
calories_data.head()

X=calories_data.drop(columns=['User_ID','Calories'],axis=1)
Y=calories_data['Calories']

print(X)
print(Y)

X_train, X_test, Y_train, Y_test= train_test_split(X, Y, test_size=0.2, random_state=2)
print(X.shape,X_train.shape,X_test.shape)

model1=LinearRegression()
model1.fit(X_train, Y_train)
test_data_prediction= model1.predict(X_test)
print(test_data_prediction)
mae=metrics.mean_absolute_error(Y_test, test_data_prediction)
print("Mean Absolute Error", mae)
model1.score(X_train, Y_train)

model2=XGBRegressor()
model2.fit(X_train, Y_train)
test_data_prediction= model2.predict(X_test)
print(test_data_prediction)

```

```
mae=metrics.mean_absolute_error(Y_test, test_data_prediction)
print("Mean Absolute Error", mae)
model2.score(X_train, Y_train)
```

```
model3= RandomForestRegressor()
model3.fit(X_train, Y_train)
test_data_prediction= model3.predict(X_test)
print(test_data_prediction)
mae=metrics.mean_absolute_error(Y_test, test_data_prediction)
print("Mean Absolute Error", mae)
m3=model3.score(X_train, Y_train)
model3.score(X_train, Y_train)
```

## REFERENCES:

- Herman Pontzer, “Burn: New Research Blows the Lid Off How We Really Burn Calories, Lose Weight, and Stay Healthy,” April 2015.
- Ingmar, “Forecasting Workout Burnt Calories Using Machine Learning,” in Qatar Computing Research Institute, January 2016.
- R.N. Dickerson, J. J. Patel, and C. J. McClain, “Protein and Calorie Requirements Associated With the Presence of Obesity. in Nutr Clin Pract. vol. 32, no. 1, 2017.
- Trevon D. Logan NBER, “The Transformation of Hunger: The Demand for Calories Past and Present,” in National Bureau of Economic Research, Working Paper No. 11754, November 2005.
- Manal Chokr, and Shady Elbassuoni, “Calories prediction from food images,” in Innovative Applications of Artificial Intelligence, 2017.
- J. Litta, Sumam Mary Idicula, and U. C. Mohanty, “Artificial Neural Network Model in Prediction of Meteorological Parameters during Premonsoon Thunderstorms,” in Hindawi, 2013.
- Tomoaki Kashiwaoa, Koichi Nakayamaa, Shin Andoc, Kenji Ikeda, Moonyong Lee, and Alireza Bahadori, ““A neural network-based local rainfall prediction system using meteorology.