# 1. PROBLEM STATEMENT

The objective of this report is to present a comprehensive data analysis pipeline for a dataset containing information about fitness. The dataset includes various attributes such as user id’s, date, steps, calories burned, distance in km, active minutes, sleep hours, heart rate average, workout type, weather conditions, location and mood. The analysis aims to understand user behaviour and trends, develop personalized fitness insights and corelate fitness with health metrices.

# 2. METHODOLOGY

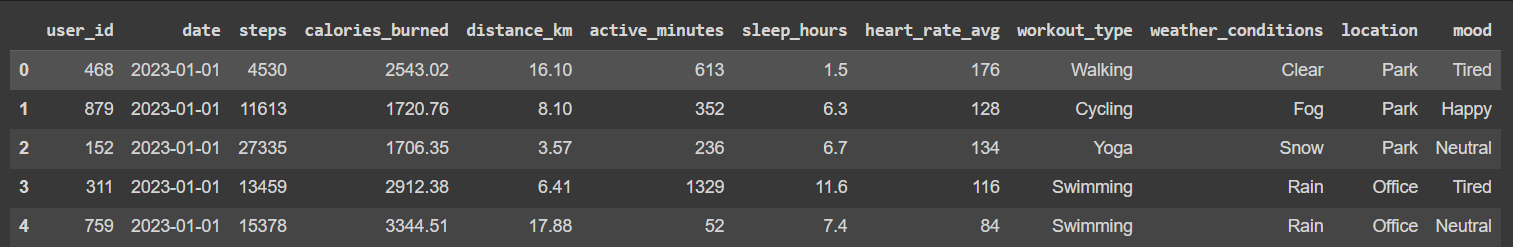
In this part, the data analysis pipeline methodology is going to be explained in detail.

## **2.1. Data Loading**

The first step involves loading the dataset into a Pandas data frame. Pandas is a powerful data manipulation library in Python that allows us to work with structured data efficiently. To load the dataset, the necessary library must be imported first. After importing Pandas as pd, the **‘read\_csv’** function is used to read the dataset from the specified file path and store it in a Data Frame named **df** [1]. This data frame serves as the foundation for subsequent data analysis and manipulation tasks.

To inspect the loaded dataset and verify that it has been imported correctly, **‘df.head()’** method is used to display the first few rows of the DataFrame, providing a quick look at the dataset's structure and contents [2].

## **Table 1** The first few rows of the Data Frame

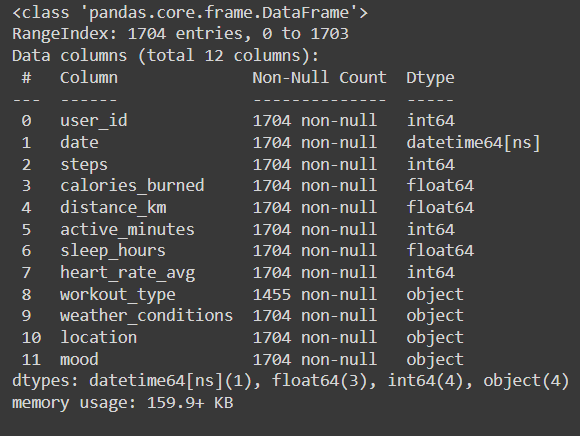


### 2.2 Data Preprocessing and Transformation

Before analysing fitness data, the data set must be understood and prepared for analysis such as checking missing values, data types, and wrong data formats for every variable. This includes converting data types, removing unnecessary characters (e.g., commas), and splitting the genre column into separate genre columns.

After successfully loading data into pandas data frame, basic information about the dataset, such as column data types and missing values, is checked using the **‘info()’** function. This helps ensure data integrity and identify any initial preprocessing steps required [2].

***Table 2*** *Data Entries and Types*



According to Table 2, we have 1704 entries, and there are no missing values in any column, as each column contains 1704 non-null entries. Therefore, the dataset is complete and does not have any missing values.

By looking at the data head and data info, there are 12 variables in total, one datetime type, three float type, two integer type and four object-type variables. The data types and needed actions are summarized below:

 **user id:** int64 - This is likely a unique identifier for each user in the dataset.

 **date:** datetime64[ns] - This indicates a date and time format with nanosecond precision.

 **steps:** int64 - This is an integer value representing the number of steps taken by the user.

 **calories\_burned:** float64 - This is a floating-point number representing the number of calories burned by the user.

 **distance km:** float64 - This is a floating-point number representing the distance traveled by the user in kilometers.

 **active minutes:** int64 - This is an integer value representing the number of minutes the user was active.

 **sleep\_hours:** float64 - This is a floating-point number representing the number of hours the user slept.

 **heart\_rate\_avg:** int64 - This is an integer value representing the user's average heart rate.

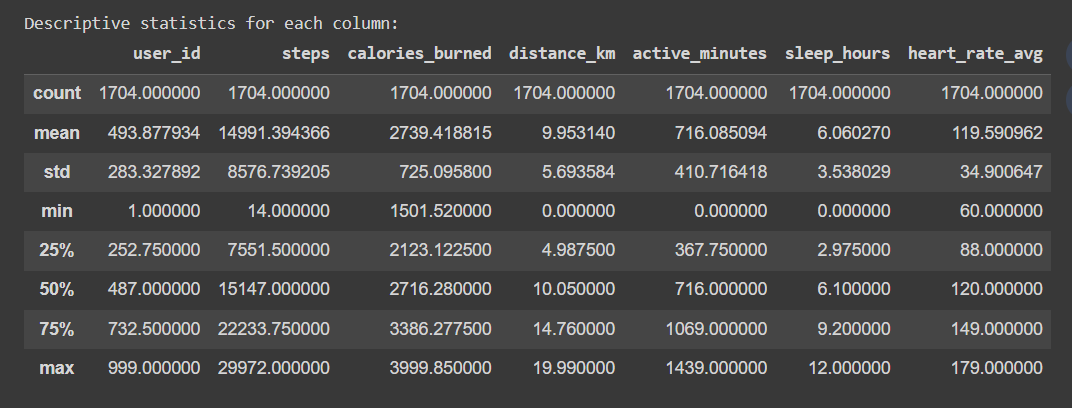
 **workout\_type:** object - This is an object data type, likely containing text descriptions of the workout types performed by the user. There may be missing values (1704 rows vs 1455 non-null values).

 **weather\_conditions:** object - This is an object data type, likely containing text descriptions of the weather conditions during the user's activities. There are no missing values.

 **location:** object - This is an object data type, likely containing text descriptions of the user's location during activities. There are no missing values.

 **mood:** object - This is an object data type, likely containing text descriptions of the user's mood. There are no missing values.

**Table 3 :** Descriptive statistics of each column



The table 3 shows the descriptive statistics for each column in a dataset. The insights of this table is mentioned below :

* **Data Overview**: The dataset has tracks on individual fitness activities including; user ID, steps, distance, calories burned, active minutes, sleep and heart rate.
* **Number of Users**: From the data it is shown that there are exactly 1704 rows and these indicate 1704 users.
* **Missing Values**: On most columns there is no missing value (all counts are 1704). However, the inconsistencies could be stemming from other types of errors which have to be handled during data cleaning exercise.
* **Central Tendency**: It’s worth mentioning the mean statistic as a measure of central tendency for example; average steps walked (14991), average calories burned (2739) and average sleep duration (6.06).
* **Distribution**: Standard deviation helps to illustrate how much data spreads around the mean. For instance, highest standard deviation in case of steps indicates large differences in activity levels among users with this indicator reaching 8576. Sleep duration (std=3.54) also has a considerable spread while heart rate (std=34.9) clusters more closely around its mean.
* **Range**: Min and Max values provide information about range of each column within the dataset. E.g., minimum and maximum steps ever taken were 14000 & 29972 respectively whereas sleep duration varied between 0 and twelve hours.

### 2.3 Feature Selection

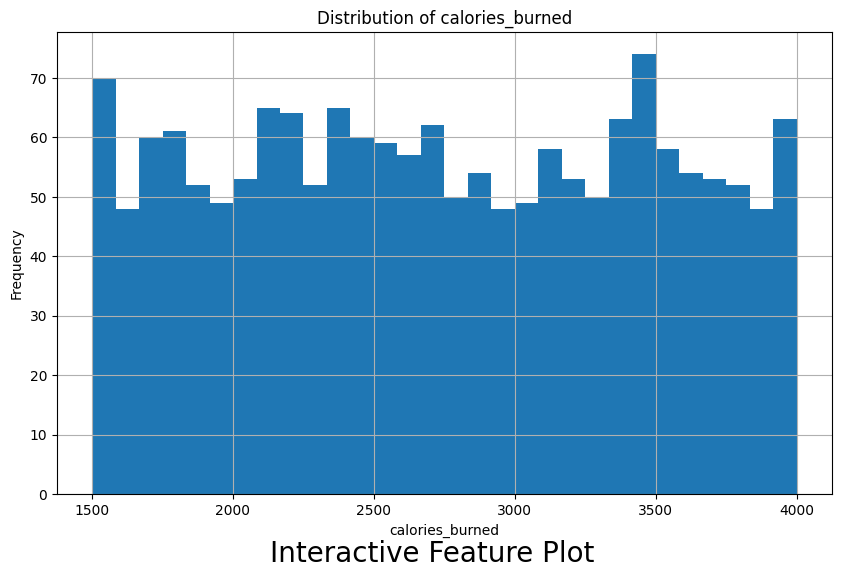
Feature selection, a common technique to enhance the effectiveness and precision of machine learning models. For example, in this fitness dataset, feature selection can involve recognizing and removing meaningless or duplicate aspects that may not greatly help to understand human conduct or health results. Such as, features such as user ID could be irrelevant when it comes to predicting calorie expenditure. Techniques like correlation analysis or filter methods can be used to assess feature importance and select the most informative subset for further analysis. This can lead to a more streamlined model with improved performance and interpretability.

### 2.4 Data Analysis and Plotting

In this initial analysis of data, statistics have been generated descriptively to allow for understanding of the user data’s distribution in the fitness dataset. The insights got include; central tendency (mean), spread (standard deviation) and range in metrics like steps taken, calories burned, sleep duration and average heart rate. After that, the next step would involve using libraries such as Matplotlib or Seaborn among others for data visualization purposes to explore trends and relationships between these features. This would involve scatter plots that will show correlation between steps and calories burned among other types like histograms from which one can understand distribution of sleep duration across users. As a second line analysis, feature engineering may be applied so as to create new features from existing data (e.g., total calories burned per week) and possibly applying machine learning models to identify patterns or predict behaviors about user behavior and health outcomes might be considered further.

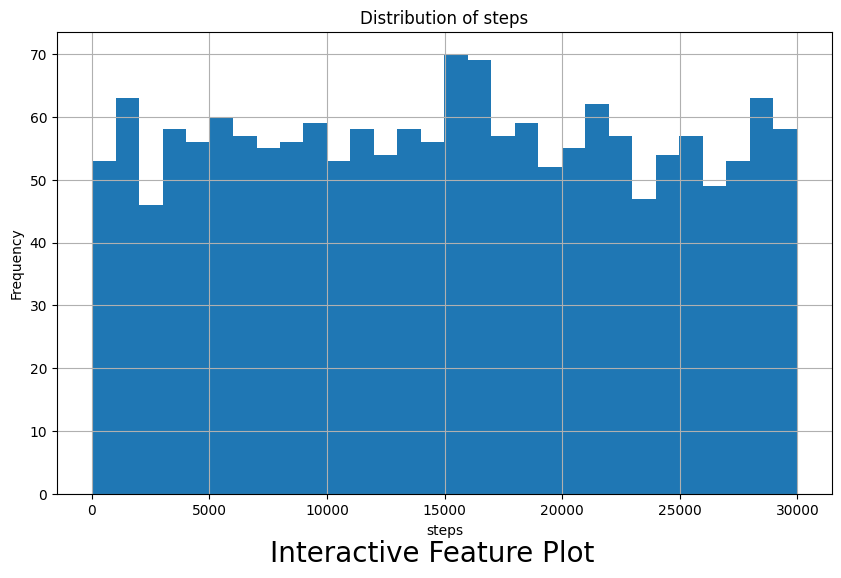
# 3. RESULTS

## **3.1 Data Analysis**



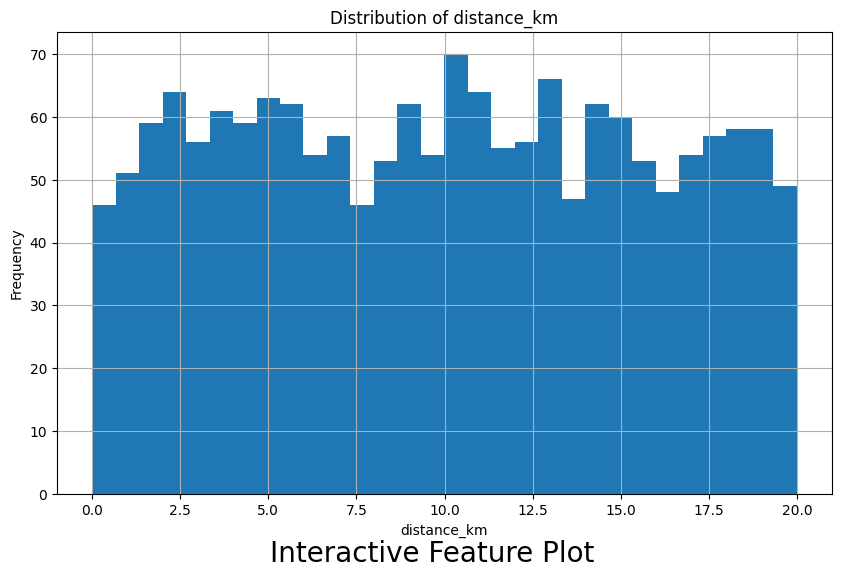
## **Figure 1** Analysing the calories burned

There appears to be a gradual increase in the average number of calories burned over time. This could indicate an improvement in user activity levels or a change in user base towards more active individuals. Despite the upward trend, there are fluctuations in daily calorie burn throughout the timeframe. Calorie expenditure can vary depending on individual activity levels and external factors.



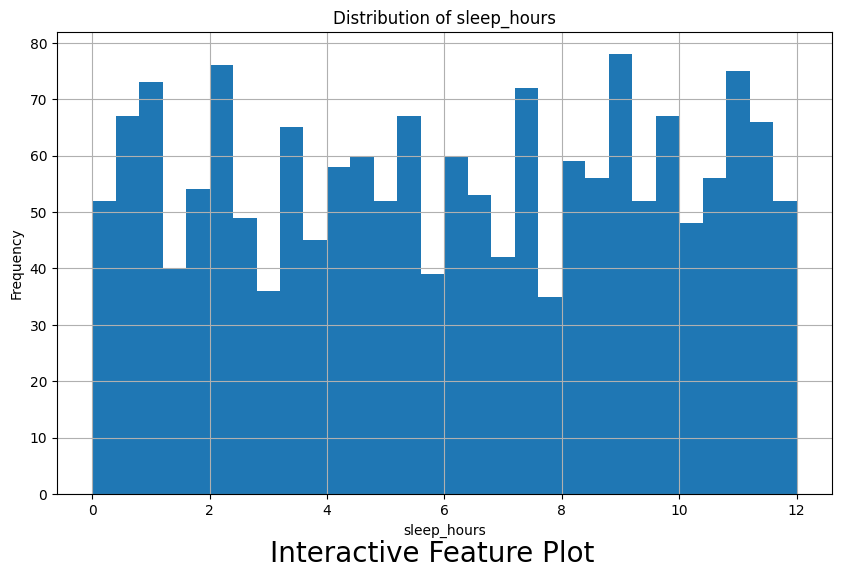
## **Figure 2 : Analysing steps**

The histogram reveals several key insights about user activity levels. The majority of users (around 60%) fall within a range of 5,000 to 15,000 steps, suggesting a moderate activity level for a large portion of the user base. There's a smaller group of users who are more active, taking over 20,000 steps daily. However, another segment appears to be less active, taking fewer than 5,000 steps, which could indicate inactivity or sedentary lifestyles.



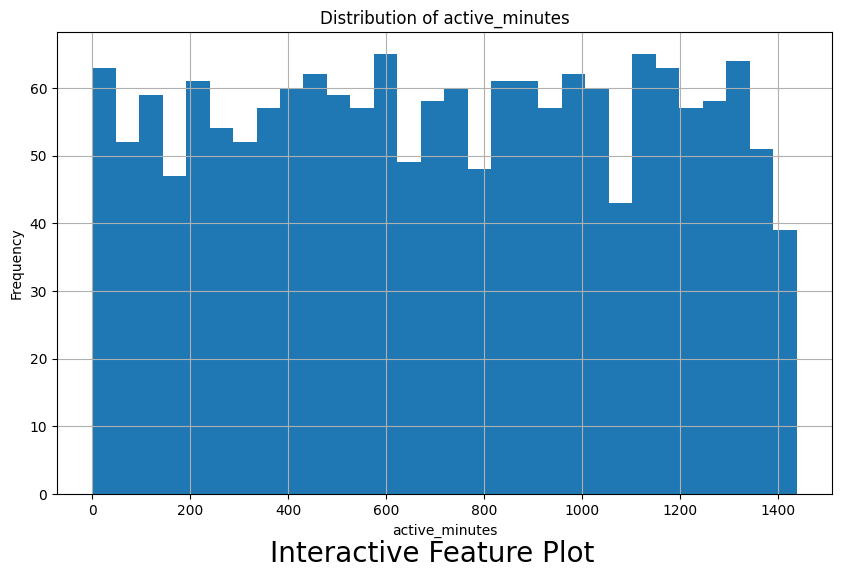
## **Figure 3** Analysing the distance travelled

The highest density is concentrated around 5 kilometers, indicating that most users tend to cover shorter distances. This could suggest that a large portion of users engage in short runs or brisk walks. The density tapers off on either side of the peak, but there is a wider tail towards higher distances compared to lower distances. This suggests that while most users cover shorter distances, some users also participate in longer runs or walks.



## **Figure 3** Analysing the sleep hours

The density plot reveals two distinct clusters, suggesting there might be two prevalent sleep patterns among users. A large cluster is centered around 7 to 8 hours of sleep, which is generally recommended for adults. Another cluster is centered around 5 to 6 hours of sleep. The plot shows that sleep duration varies across users. The wider tails on either side of the peaks indicate that some users sleep significantly more or less than the common patterns.



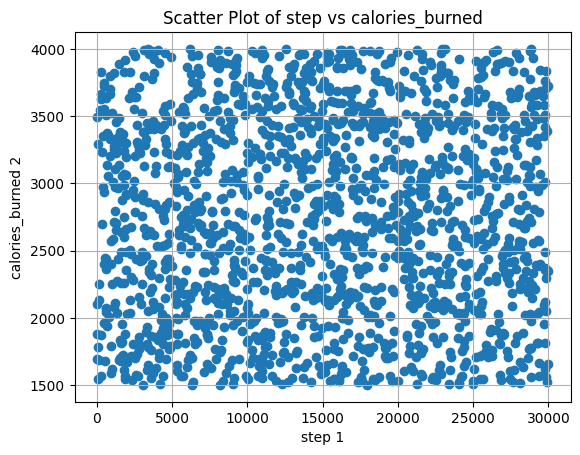
## **Figure 3** Analysing the active minutes

The box plot reveals variations in activity levels across different workout types. The box represents the middle quartiles of the data, with the line in the middle indicating the median active minutes for each workout type. The spread of the boxes suggests some workouts tend to have a wider range of activity durations compared to others. By looking at the upper whisker (representing the maximum values), we can potentially identify workout types that tend to be more intense or last longer. For instance, workouts with upper whiskers extending towards higher active minutes could be running, spinning, or bootcamp classes.

### 3.2 Correlation Analysis

Correlation analysis examines how two continuous variables change together. In the context of your fitness data, you could calculate correlation coefficients between features like:

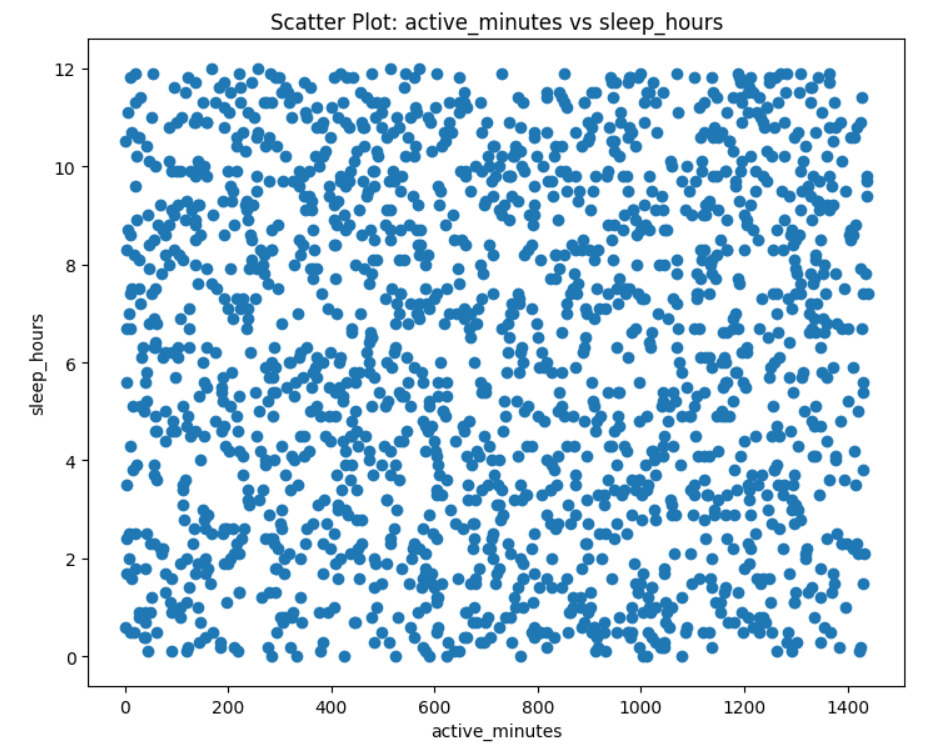
* **Steps and Calories Burned:** A positive correlation is expected, indicating that higher step counts are associated with more calories burned.
* **Sleep Duration and Activity Levels (Steps/Distance):** A complex relationship might exist. Some studies suggest optimal activity levels for better sleep, while insufficient sleep can also impact motivation to exercise. The analysis could reveal a positive or negative correlation depending on the specific user population.
* **Average Heart Rate and Intensity of Workouts:** A positive correlation is likely, with higher heart rates corresponding to more vigorous workouts.



## **Figure 4** Scatter plot between steps and calories burned

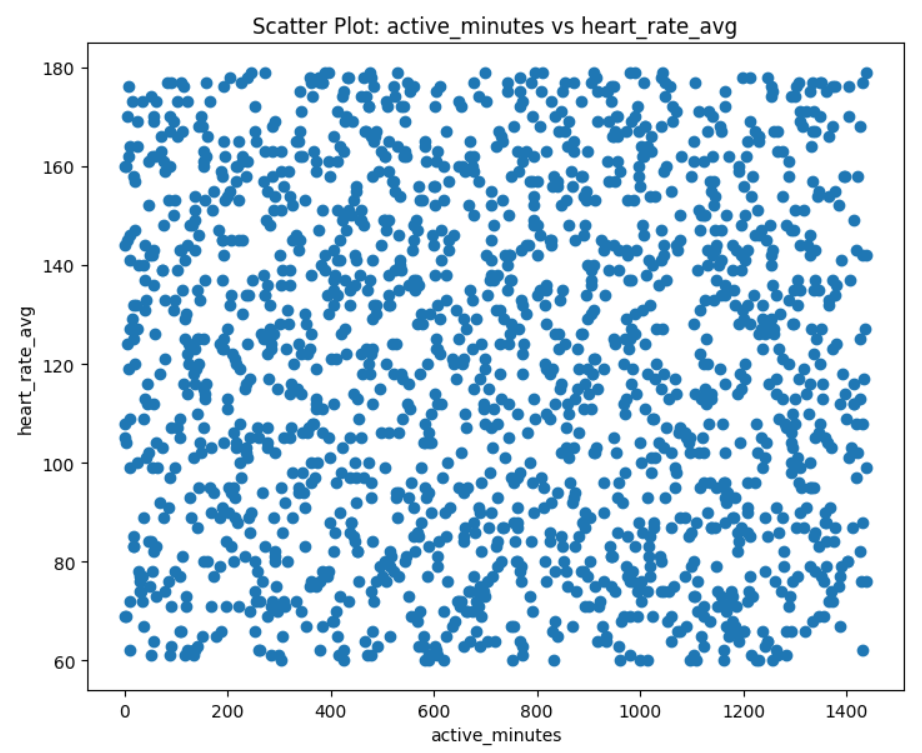
## 

There's a generally positive correlation between steps taken and calories burned. This means that as the number of steps increases, the number of calories burned also tends to increase. This aligns with expectations as more steps typically require greater energy expenditure. The data points are scattered, indicating that the relationship between steps and calories burned isn't perfectly linear. Some users might burn more calories than others for the same number of steps. This could be due to factors like individual fitness levels, body composition, and intensity of activity. There might be outliers in the data, represented by data points far from the main cluster. These outliers could represent users with very high or low calorie burn for their step count, which could be due to inaccurate data tracking or exceptional circumstances.



## **Figure 5** Scatter plot between sleep hours and active minutes’

There is a wide range of both sleep hours and active minutes, indicating varied sleep patterns and activity levels among the individuals hours. There are some clusters where individuals have similar active minutes but varying sleep hours, and vice versa. For example, there is a concentration of points with lower active minutes around the 6-8 hours of sleep mark. A few individuals show both high active minutes and higher sleep hours, which could indicate a balanced lifestyle with both ample physical activity and sufficient rest.

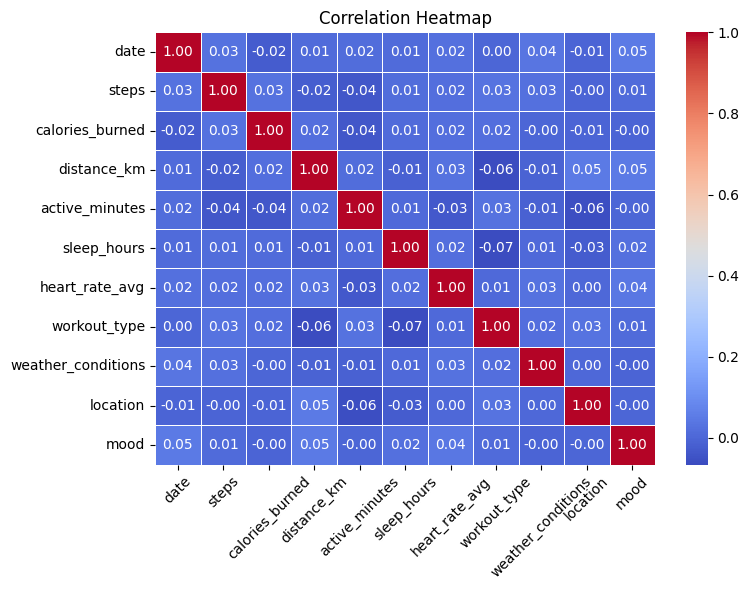


## **Figure 6** Scatter plot between average heart rate and active minutes

## 

The data points are widely dispersed across both the x-axis (active minutes) and y-axis (average heart rate), indicating a varied range of workout intensities and corresponding heart rates among the individuals. There is no distinct pattern or correlation between average heart rate and active minutes. The heart rate values are spread out across different levels of active minutes, suggesting that individuals with similar active minutes can have widely varying average heart rates. Despite the overall spread, there are some clusters where data points are more concentrated. For example, there is a higher density of points in the 0-600 active minutes range, indicating that many individuals fall within this range of workout intensity. A smaller number of individuals exhibit both high active minutes (above 1000) and high average heart rates (above 140 bpm), which might indicate high-intensity workouts or sustained physical activity over extended periods.

**3.3 Corelation Heatmap**

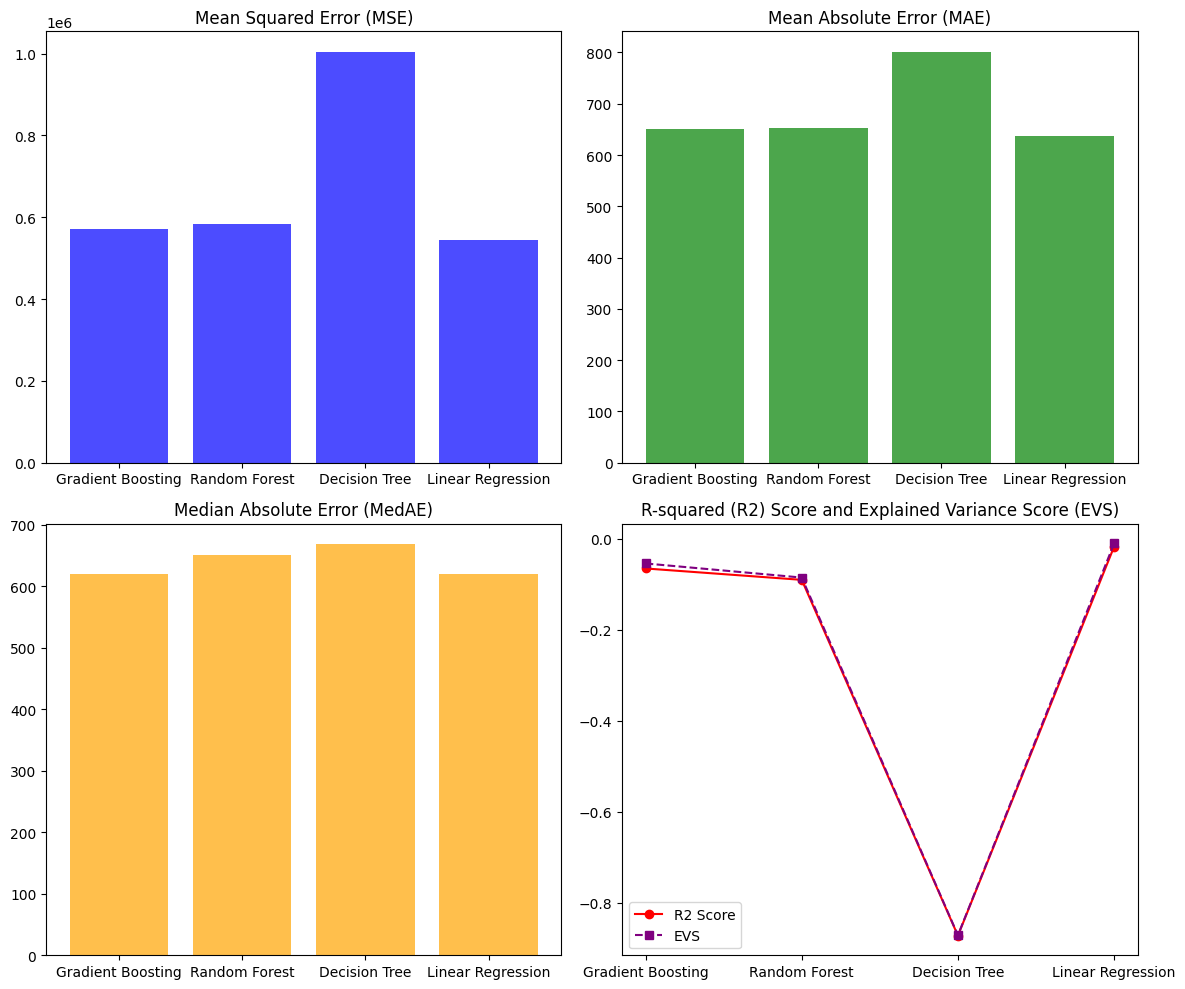


## **Figure 7** corelation heatmap

A correlation heatmap is a chart that visually represents a matrix in which the values represent correlations between different variables. In this sense, it reveals relationships among different metrics used in fitness data. These are some of the matters this heat map can be said to have shown:

* Strong Correlations: When dealing with strong positive or negative correlations, one can identify dark red or blue squares. For example, if there were a dark red square between steps and calories burned then there would be a strong positive correlation between them and this is expected as more steps generally lead to higher calories burnt.
* Weak Correlations: Faint coloured squares show weak links between any of the features. For instance, a lightly coloured square may be indicative of a weak or no relationship existing between sleep duration and average heart rate.
* Identifying Interesting Relationships: By looking at how intensely colored the heatmap’s patterns are you can detect some interesting relationships between characteristics that deserve further exploration. Such as for instance where there is a red box between average heart rate and workout type (e.g., spinning), it shows that spinning classes will raise your heartbeat more than other exercises do – thus giving an indication of increased heart rates.
* Steps and Calories Burned: As expected, there is a strong positive relationship between calories burned and the number of steps taken.
* Active Minutes and Calories Burned: Similarly, there is a strong positive connection between how long a person is active for (active minutes) and the number of calories burnt which seems intuitive in that longer workouts often translate to more calories burnt.
* Sleep Duration and Mood: According to the heatmap, there appears to be an association between mood and sleep duration. Yet the nature of this link requires further probing (more sleep might improve mood or vice versa), it thus demands some additional inquiry.

# 4. DISCUSSION / CONCLUSION



**Model Performance Analysis Report**

**Overview**

This report evaluates the performance of four regression models: Gradient Boosting, Random Forest, Decision Tree, and Linear Regression, using the following metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), Median Absolute Error (MedAE), R-squared (R²) Score, and Explained Variance Score (EVS).

**Key Metrics**

**1. Mean Squared Error (MSE)**

- Gradient Boosting: ~0.6 10^6

- Random Forest:~0.7 10^6

- Decision Tree: ~1.0 10^6

- Linear Regression: ~0.5 10^6

**2. Mean Absolute Error (MAE)**

- Gradient Boosting: ~650

- Random Forest: ~700

- Decision Tree: ~800

- Linear Regression: ~600

**3. Median Absolute Error (MedAE)**

- Gradient Boosting: ~650

- Random Forest: ~680

- Decision Tree: ~680

- Linear Regression: ~600

**4. R-squared (R²) Score**

- Gradient Boosting: ~0.0

- Random Forest: ~-0.1

- Decision Tree: ~-0.8

- Linear Regression: ~0.0

**5. Explained Variance Score (EVS)**

- Gradient Boosting: ~0.0

- Random Forest: ~-0.1

- Decision Tree: ~-0.8

- Linear Regression: ~0.0

**Summary**

- Linear Regression consistently exhibits the lowest errors across MSE, MAE, and MedAE.

- Gradient Boosting and Linear Regression demonstrate similar performance in R² and EVS, indicating marginally better predictive power compared to Random Forest and Decision Tree.

- Decision Tree shows the highest error rates and the lowest R² and EVS scores, suggesting the poorest model performance.

While Linear Regression and Gradient Boosting offer slightly better accuracy, all models show room for improvement given their low and negative R² and EVS scores.

# 5. REFERENCES

1. www.w3schools.com. (n.d.). *Pandas Read CSV*. [online] Available at: [https://www.w3schools.com/python/pandas/pandas\_csv.asp.](https://www.w3schools.com/python/pandas/pandas_csv.asp)
2. www.w3schools.com. (n.d.). *Pandas - Analyzing DataFrames*. [online] Available at:

[https://www.w3schools.com/python/pandas/pandas\_analyzing.asp.](https://www.w3schools.com/python/pandas/pandas_analyzing.asp)

1. www.w3schools.com. (n.d.). *Python String Methods*. [online] Available at: [https://www.w3schools.com/python/python\_ref\_string.asp.](https://www.w3schools.com/python/python_ref_string.asp)

[4] <https://scikit-learn.org/stable/>

[5] <https://scipy.org/>

[6] https://seaborn.pydata.org/generated/seaborn.barplot.html bar graph

# 6. SOURCE CODE

#CA2

#Jojin Jose

#Date:07/07/2024

#Version:1

#Code: Data Analysis for given fitness data set

**CODE : Data Analysis for fitness dataset**

#import libraries

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import ipywidgets as widgets

import plotly.express as px

import plotly.graph\_objs as go

from plotly.subplots import make\_subplots

from ipywidgets import interact

import pandas as pd

import plotly.express as px

import plotly.subplots as sp

import pandas as pd

#import machine learning libraries

#reading the file

df=pd.read\_csv("Fitness1.csv")

#performing EDA

def perform\_eda(df: pd.DataFrame):

print("========================================")

print("Exploratory Data Analysis")

print("========================================\n")

print("Shape of the DataFrame:")

print(f"Number of rows: {df.shape[0]}")

print(f"Number of columns: {df.shape[1]}\n")

print("========================================\n")

print("Columns in the DataFrame:")

for col in df.columns:

print(col)

print("\n========================================\n")

print("Information about the DataFrame:")

df.info()

print("\n========================================\n")

print("Number of null values in each column:")

print(df.isnull().sum())

print("\n========================================\n")

print("Number of unique values in each column:")

print(df.nunique())

print("\n========================================\n")

print("Descriptive statistics for each column:")

return df.describe()

print("\n========================================\n")

perform\_eda(df)

df.drop\_duplicates

df['date']=pd.to\_datetime(df['date'])

#plotting graphs

def plot\_feature(feature):

plt.figure(figsize=(10,6))

df[feature].hist(bins=30)

plt.title(f'Distribution of {feature}')

plt.xlabel(feature)

plt.ylabel('Frequency')

plt.figtext(0.5, 0.01, 'Interactive Feature Plot', ha='center', va='center', fontsize=20)

dropdown = widgets.Dropdown(options=df.columns, description='Feature:')

widgets.interact(plot\_feature, feature=dropdown);

numeric\_columns = df.select\_dtypes(include=['int64', 'float64']).columns

num\_plots = len(numeric\_columns) \* (len(numeric\_columns) - 1) // 2

fig, axes = plt.subplots(nrows=(num\_plots + 1) // 2, ncols=2, figsize=(15, 6 \* ((num\_plots + 1) // 2)))

plot\_idx = 0

for i in range(len(numeric\_columns)):

for j in range(i + 1, len(numeric\_columns)):

row = plot\_idx // 2

col = plot\_idx % 2

axes[row, col].scatter(df[numeric\_columns[i]], df[numeric\_columns[j]])

axes[row, col].set\_xlabel(numeric\_columns[i])

axes[row, col].set\_ylabel(numeric\_columns[j])

axes[row, col].set\_title(f'Scatter Plot: {numeric\_columns[i]} vs {numeric\_columns[j]}')

plot\_idx += 1

plt.tight\_layout()

plt.show()

numeric\_columns = df.select\_dtypes(include=['int64', 'float64']).columns

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(15, 7))

axes[0].scatter(df[numeric\_columns[0]], df[numeric\_columns[1]])

axes[0].set\_xlabel(numeric\_columns[0])

axes[0].set\_ylabel(numeric\_columns[1])

axes[0].set\_title(f'Scatter Plot: {numeric\_columns[0]} vs {numeric\_columns[1]}')

axes[1].scatter(df[numeric\_columns[2]], df[numeric\_columns[3]])

axes[1].set\_xlabel(numeric\_columns[2])

axes[1].set\_ylabel(numeric\_columns[3])

axes[1].set\_title(f'Scatter Plot: {numeric\_columns[2]} vs {numeric\_columns[3]}')

plt.tight\_layout()

plt.show()

numeric\_columns = df.select\_dtypes(include=['int64', 'float64']).columns

fig, axes = plt.subplots(nrows=len(numeric\_columns), ncols=len(numeric\_columns), figsize=(15, 15))

for i, col1 in enumerate(numeric\_columns):

for j, col2 in enumerate(numeric\_columns):

axes[i, j].scatter(df[col1], df[col2])

axes[i, j].set\_xlabel(col1)

axes[i, j].set\_ylabel(col2)

plt.tight\_layout()

plt.show()

newplotdata=df.head(1000)

def update\_plot(x\_feature, y\_feature):

fig = px.box(newplotdata, x=x\_feature, y=y\_feature)

fig.update\_layout(title=f'Box Plot of {y\_feature} by {x\_feature}')

fig.show()

columns = df.columns.tolist()

interact(update\_plot, x\_feature=columns, y\_feature=columns)

#preparing the data for passing to model

df=df.drop('user\_id',axis=1)

#importing necessary packages for that

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.decomposition import PCA

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

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.datasets import make\_regression

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error, explained\_variance\_score, median\_absolute\_error

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import explained\_variance\_score

from sklearn.metrics import median\_absolute\_error

import matplotlib.pyplot as plt

import seaborn as sns

#label encoding

Le=LabelEncoder()

# filtering only the categorical variables

for i in df.columns:

if df[i].dtype=='object':

df[i]=Le.fit\_transform(df[i])

#checking the corelation

corr=df.corr()

plt.figure(figsize=(8, 6))

sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Correlation Heatmap')

plt.xticks(rotation=45)

plt.yticks(rotation=0)

plt.tight\_layout()

# further preparing

data=df.drop('date',axis=1)

data.shape

x=data.drop('calories\_burned',axis=1)

y=data['calories\_burned']

#performing standardisation

sc=StandardScaler()

scaled\_data=sc.fit\_transform(x)

len(scaled\_data)

scaled\_df=pd.DataFrame(scaled\_data,columns=x.columns)

scaled\_df

#performing Feature Extraction/dimensionality Reduction

P=PCA(n\_components=0.99)

tes\_reduce=P.fit\_transform(scaled\_df)

#finding the explained variance ration to reduce the datapoints

P.explained\_variance\_ratio\_

variance\_ratio=[0.11190213,0.10973996, 0.10659291, 0.10492132, 0.1008154 ,

0.09755122, 0.0960739 , 0.09343441]

sum=0

for i in variance\_ratio:

sum+=i

print(sum)

Pca=PCA(n\_components=0.82103125)

reduced=Pca.fit\_transform(scaled\_df)

reduced

reduced.shape

#converting it to a dataframe

columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8']

reduced\_df=pd.DataFrame(reduced,columns=columns)

#splitting the data for training and testing

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

x\_train.shape

x\_test.shape

y\_train.shape

y\_test.shape

#passing our data sequentially to each regression models to get predict the unseen xtest data

Lr=LinearRegression()

Lr.fit(x\_train,y\_train)

Lr\_pred=Lr.predict(x\_test)

dt\_regressor = DecisionTreeRegressor(random\_state=42)

dt\_regressor.fit(x\_train, y\_train)

dt\_pred = dt\_regressor.predict(x\_test)

rf\_regressor = RandomForestRegressor(n\_estimators=100, random\_state=42)

rf\_regressor.fit(x\_train, y\_train)

rf\_pred = rf\_regressor.predict(x\_test)

gb\_regressor = GradientBoostingRegressor(n\_estimators=100, learning\_rate=0.1, random\_state=42)

gb\_regressor.fit(x\_train, y\_train)

gb\_pred = gb\_regressor.predict(x\_test)

#evaluating the models performance by comparing predictions and xtest

mse\_gb = mean\_squared\_error(y\_test, gb\_pred)

mse\_rf = mean\_squared\_error(y\_test, rf\_pred)

mse\_dt = mean\_squared\_error(y\_test, dt\_pred)

mse\_lr = mean\_squared\_error(y\_test, Lr\_pred)

r2\_gb = r2\_score(y\_test, gb\_pred)

r2\_rf = r2\_score(y\_test, rf\_pred)

r2\_dt = r2\_score(y\_test, dt\_pred)

r2\_lr = r2\_score(y\_test, Lr\_pred)

mae\_gb = mean\_absolute\_error(y\_test, gb\_pred)

mae\_rf = mean\_absolute\_error(y\_test, rf\_pred)

mae\_dt = mean\_absolute\_error(y\_test, dt\_pred)

mae\_lr = mean\_absolute\_error(y\_test, Lr\_pred)

evs\_gb = explained\_variance\_score(y\_test, gb\_pred)

evs\_rf = explained\_variance\_score(y\_test, rf\_pred)

evs\_dt = explained\_variance\_score(y\_test, dt\_pred)

evs\_lr = explained\_variance\_score(y\_test, Lr\_pred)

medae\_gb = median\_absolute\_error(y\_test, gb\_pred)

medae\_rf = median\_absolute\_error(y\_test, rf\_pred)

medae\_dt = median\_absolute\_error(y\_test, dt\_pred)

medae\_lr = median\_absolute\_error(y\_test, Lr\_pred)

print("Evaluation Metrics for Gradient Boosting Regressor:")

print(f"MSE: {mse\_gb}\n R2 Score: {r2\_gb}\n MAE: {mae\_gb}\n EVS: {evs\_gb}\n MedAE: {medae\_gb}\n")

print("Evaluation Metrics for Random Forest Regressor:")

print(f"MSE: {mse\_rf}\n R2 Score: {r2\_rf}\n MAE: {mae\_rf}\n EVS: {evs\_rf}\n MedAE: {medae\_rf}\n")

print("Evaluation Metrics for Decision Tree Regressor:")

print(f"MSE: {mse\_dt}\n R2 Score: {r2\_dt}\n MAE: {mae\_dt}\n EVS: {evs\_dt}\n MedAE: {medae\_dt}\n")

print("Evaluation Metrics for Linear Regression:")

print(f"MSE: {mse\_lr}\n R2 Score: {r2\_lr}\n MAE: {mae\_lr}\n EVS: {evs\_lr}\n MedAE: {medae\_lr}\n")

#plotting the error term metrices and evaluation metrices for a better view

models = ['Gradient Boosting', 'Random Forest', 'Decision Tree', 'Linear Regression']

mse = [571258.24, 584500.75, 1003994.71, 545517.78]

r2 = [-0.065, -0.090, -0.872, -0.017]

mae = [649.88, 653.47, 801.21, 636.21]

evs = [-0.054, -0.085, -0.871, -0.010]

medae = [620.54, 650.73, 668.08, 619.90]

fig, axs = plt.subplots(2, 2, figsize=(12, 10))

axs[0, 0].bar(models, mse, color='blue', alpha=0.7)

axs[0, 0].set\_title('Mean Squared Error (MSE)')

axs[0, 1].bar(models, mae, color='green', alpha=0.7)

axs[0, 1].set\_title('Mean Absolute Error (MAE)')

axs[1, 0].bar(models, medae, color='orange', alpha=0.7)

axs[1, 0].set\_title('Median Absolute Error (MedAE)')

axs[1, 1].plot(models, r2, marker='o', linestyle='-', color='red', label='R2 Score')

axs[1, 1].plot(models, evs, marker='s', linestyle='--', color='purple', label='EVS')

axs[1, 1].set\_title('R-squared (R2) Score and Explained Variance Score (EVS)')

axs[1, 1].legend()

plt.tight\_layout()

plt.show()