

# FITNESS TRACKER SYSTEM

NAME: PALLA BHARATH KUMAR

INTERN ID: MST03-0089

DOMAIN: DATA SCIENCE TRAINING

META SCIFOR TECHNOLOGIES

UNDER THE GUIDANCE OF  
UROOJ KHAN

# AGENDA

- ▶ Introduction
- ▶ Project Overview
- ▶ Key Features
- ▶ Methodology
- ▶ Implementation
- ▶ Results
- ▶ Conclusion

# INTRODUCTION

- ▶ The Fitness Tracker System is a data-driven approach to monitoring and analysing fitness metrics, designed to assist users in achieving their health goals. By leveraging advanced data science techniques, the system processes key metrics such as calories burned, BMI, and workout frequency to provide actionable insights. Using a robust dataset of over 1,000 records, it employs machine learning models to predict outcomes and recommend personalized fitness routines. The system integrates exploratory data analysis, feature engineering, and visualization tools to uncover trends and correlations, making it a comprehensive solution for enhancing personal health and fitness.

# Project Overview

- ▶ The Fitness Tracker System is a comprehensive application that utilizes data science to transform raw fitness data into meaningful insights. It analyses user-specific metrics such as age, weight, workout frequency, session duration, and calories burned to provide personalized recommendations for optimizing fitness routines. By combining techniques like data pre-processing, feature engineering, and machine learning, the system achieves high prediction accuracy and ensures actionable outputs. The project highlights the integration of technology in promoting healthier lifestyles and serves as a scalable framework for personal fitness tracking and analysis.

# Key Features

- ▶ BMI Calculation: Automatically calculates and categorizes Body Mass Index
- ▶ (BMI).Workout Insights: Analyses session duration, frequency, and caloric expenditure.
- ▶ Machine Learning Models: Utilizes Logistic Regression, Decision Trees, and Random Forest for predictions.
- ▶ Data Visualization: Generates heat-maps, scatter plots, and histograms for fitness trends.
- ▶ One-Hot Encoding: Processes categorical data like gender and workout type.
- ▶ Scalable Framework: Handles datasets with over 1,000 records for robust analysis.
- ▶ Model Deployment: Supports saving and loading models for reuse using pickle.

# METHODOLOGY

- ▶ Data Understanding: Examine the dataset, its attributes, and structure.
- ▶ Data Cleaning: Handle missing values, remove duplicates, and ensure consistent data types.
- ▶ Feature Engineering: Derive new metrics like BMI and categorize variables (e.g., age groups, BMI categories).
- ▶ Data Transformation: Apply scaling and one-hot encoding for pre-processing.
- ▶ Dataset Splitting: Split data into training and testing sets (80-20 ratio).
- ▶ Model Training: Train machine learning models like Logistic Regression, Decision Trees, and Random Forest.
- ▶ Model Evaluation: Assess models using accuracy, precision, recall, and F1-score metrics.
- ▶ Insights & Recommendations: Provide actionable results and fitness optimizations.

# Implementation

```
[ ] #importing necessary libraries

import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

## Load the Dataset

```
[ ] #dataset
df = pd.read_csv('Gym_data.csv')

[ ] #display 20 rows of dataset
df.head(20)
```



Weight Height

Session Duration

Water Intake Workout Frequency

## Data Cleaning:

- Ensure the dataset is clean and ready for analysis by handling missing values, removing duplicates, and ensuring consistent data types.

```
▶ #check for missing values in each column
```

```
missing_values = df.isnull().sum()
print(missing_values)
```



```
Age          10
Gender       71
Weight (kg)   22
Height (m)    26
Max_BPM      21
Avg_BPM      30
Resting_BPM  19
Session_Duration (hours) 23
Calories_Burned 23
Workout_Type  61
Fat_Percentage 16
Water_Intake (liters) 24
Workout_Frequency (days/week) 58
Experience_Level 57
BMI          30
dtype: int64
```

## Handle missing values

```
[ ] df.dropna(inplace=True) #remove rows with missing values
```

## Check for Duplicates

- Identify and remove duplicate rows using .duplicated()

```
[ ] #check for duplicate rows
duplicates = df.duplicated().sum()
print("Number of duplicate rows:", duplicates)
```





## Exploratory Data Analysis (EDA):

- Gain insights from the dataset through summary statistics, visualizations, and relationship analysis.

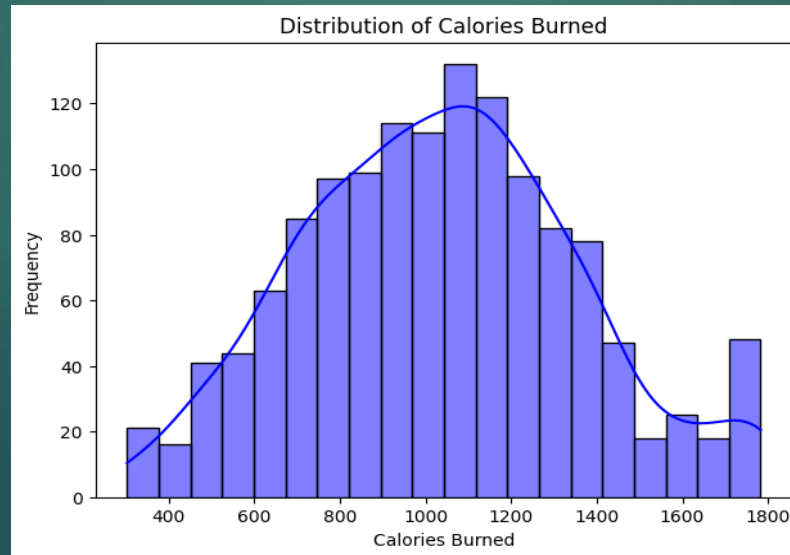
# Summary statistics for numerical features

```
print(df.describe())
```

### 🎮 #Visualize Distributions of Key Features

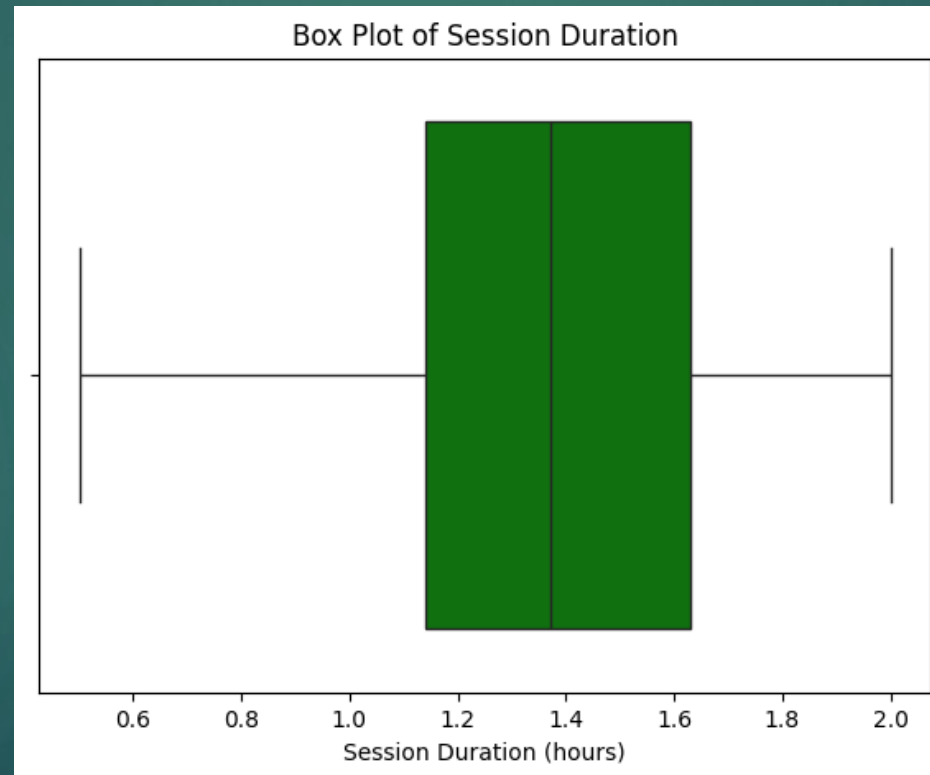
```
import matplotlib.pyplot as plt
import seaborn as sns

# Histogram for a key feature(Calories_Burned)
plt.figure(figsize=(7, 5))
sns.histplot(data['Calories_Burned'], kde=True, bins=20, color='blue')
plt.title('Distribution of Calories Burned')
plt.xlabel('Calories Burned')
plt.ylabel('Frequency')
plt.show()
```





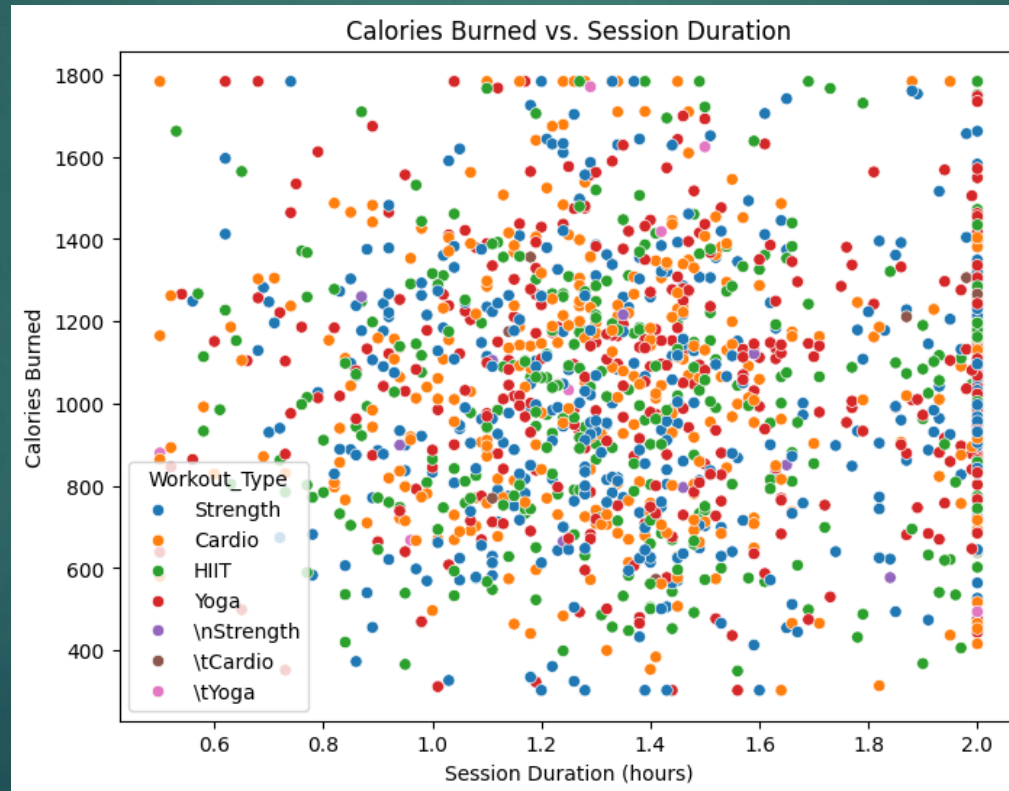
```
▶ # Box plot for a key feature(Session_Duration (hours))  
plt.figure(figsize=(7, 5))  
sns.boxplot(x=data['Session_Duration (hours)'], color='green')  
plt.title('Box Plot of Session Duration')  
plt.xlabel('Session Duration (hours)')  
plt.show()
```



## Analyze Relationships Between Variables

- Visualize relationships using scatter plots and correlation heatmaps.

```
# Scatter plot: Relationship between Calories_Burned and Session_Duration
plt.figure(figsize=(8, 6))
sns.scatterplot(x=data['Session_Duration (hours)'], y=data['Calories_Burned'], hue=data['Workout_Type'])
plt.title('Calories Burned vs. Session Duration')
plt.xlabel('Session Duration (hours)')
plt.ylabel('Calories Burned')
plt.show()
```



```

# Correlation heatmap

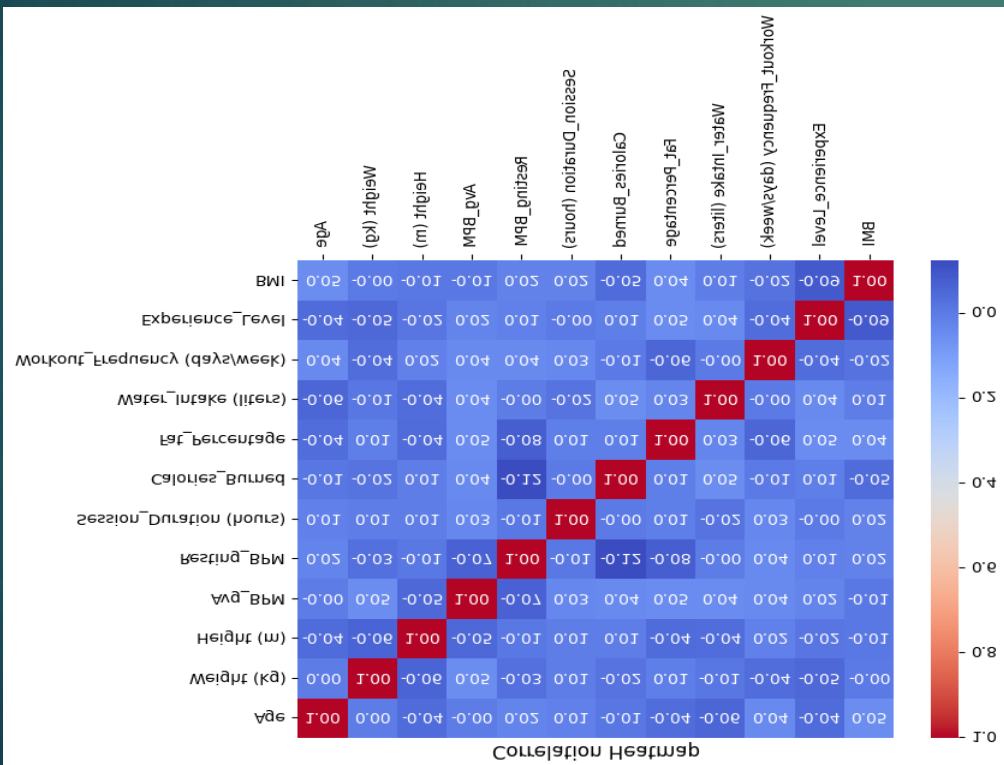
#Select only numerical columns
numerical_data = data.select_dtypes(include=['float64', 'int64'])

#Fill missing values (if any)
numerical_data.fillna(numerical_data.mean(), inplace=True)

#Calculate the correlation matrix
correlation_matrix = numerical_data.corr()

#plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()

```



## Dataset Splitting

- ▶ Split the dataset into training and testing subsets to evaluate machine learning models effectively.
- ▶ Identify Features and Target
- ▶ Separate the independent variables (features) from the dependent variable (target)

```
# Define features (X) and target (y)
```

```
X = scaled_data.drop(columns=['Calories_Burned'])
```

```
y = scaled_data['Calories_Burned']
```

Split the Dataset \*Use train\_test\_split from sklearn to divide the dataset

```
from sklearn.model_selection import train_test_split
```

```
# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

## ▶ Model Training

Train the selected models on the training dataset and optimize performance through hyper-parameter tuning

```
#check for categorical columns
```

```
#identify columns in X_train are non-numeric
```

```
#check for non-numeric columns
```

```
non_numeric_cols = X_train.select_dtypes(include=['object']).columns
```

```
print("Non-numeric columns in X_train:\n", non_numeric_cols)
```

Encode Categorical Variables:

- Convert categorical columns into numerical format using encoding methods such as One-Hot Encoding or Label Encoding.

One-Hot Encoding

- Recommended when the categorical column has no ordinal relationship (Gender)

```
[ ] # Use one-hot encoding for non-numeric columns
X_train = pd.get_dummies(X_train, columns=non_numeric_cols, drop_first=True)
X_test = pd.get_dummies(X_test, columns=non_numeric_cols, drop_first=True)

# Ensure both train and test datasets have the same columns
X_train, X_test = X_train.align(X_test, join='left', axis=1, fill_value=0)
```


```
from sklearn.linear_model import LogisticRegression
import numpy as np

# Convert continuous target to binary classes for Logistic Regression
y_train_binary = np.where(y_train > y_train.median(), 1, 0) # Example: High/Low classes

# Initialize Logistic Regression model
logistic_model = LogisticRegression(random_state=42)

# Train the model on the training data
logistic_model.fit(X_train, y_train_binary)

print("Logistic Regression model trained successfully.")
```

 Logistic Regression model trained successfully.



```
# Summarize model evaluation metrics
```

```
print(f'Metrics:')

```

```
print(f'Accuracy: {accuracy:.2f}')

```

```
print(f'Precision: {precision:.2f}')

```

```
print(f'Recall: {recall:.2f}')

```

```
print(f'F1 Score: {f1:.2f}')

```

## ► Save Model Using pickle

- I. Pickle is a Python library for serializing and deserializing objects.

```
import pickle

```

```
# Save the model

```

```
with open('decision_tree_model.pkl', 'wb') as file:

```

```
    pickle.dump(classifier, file)

```

```
print("Model saved as decision_tree_model.pkl")

```

# Results

## Model Performance:

- ▶ Random Forest Classifier achieved an accuracy of 98%, demonstrating robust prediction capability for caloric expenditure and other fitness metrics.

## Feature Insights:

- ▶ BMI, workout frequency, and session duration emerged as the strongest predictors of calories burned.

## Visual Trends:

- ▶ Scatter Plots: Highlighted a positive correlation between session duration and calories burned.
- ▶ Heatmaps: Revealed significant relationships among numerical features like BMI and caloric expenditure.

## Evaluation Metrics:

- ▶ Precision, recall, and F1-scores were consistently high, confirming the reliability of the model.



# Conclusion

- ▶ The Fitness Tracker System successfully leverages data science to provide accurate and actionable insights into user fitness metrics. By integrating robust data pre-processing techniques, feature engineering, and machine learning models, the system achieves high prediction accuracy (98%) for caloric expenditure and related fitness variables. Key insights, such as the impact of BMI and workout frequency on fitness outcomes, demonstrate the system's ability to guide personalized routines. This project highlights the potential of technology in promoting healthier lifestyles and sets a strong foundation for future enhancements, such as real-time data integration and advanced predictive modelling.



**THANK YOU!**

---