FITNESS TRACKER SYSTEM

(AN OVERVIEW OF MACHINE LEARNING AND STREAMLIT INTEGRATION)

META SCIFOR TECHNOLOGIES
2024

Meta
Scifor Technologies

Script. Sculpt. Socialize

Guided by:

UROOJ KHAN

Submitted by:

HARITHA P V
MST03-0058

TABLE OF CONTENT

1 ABSTRACT 3

2 INTRODUCTION 4

3 FITNESS TRACKER SYSTEM 6

4 TECHNOLOGY USED 10

5 DATASET INFORMATION 13

5 METHODOLOGY 16

6 CODE SNIPPET 20

7 DATA VISUALIZATION 27

8 RESULTS 32

9 CONCLUSION 36

10 REFERENCES 37

ABSTRACT

This project presents the development of an innovative Fitness Tracker System that combines machine learning models with a user-friendly web interface to monitor and predict physical activity. With the growing emphasis on health and wellness, fitness tracking has become increasingly important for individuals seeking to improve their lifestyle. The primary objective of this project was to design and implement a system capable of predicting the total number of steps a user might take in a day, based on historical fitness data.

To achieve this, we utilized two machine learning models: Polynomial Regression and Random Forest Regressor. These models were trained on a comprehensive dataset containing various features related to physical activity, such as step count, calories burned, and distance covered. The models were evaluated using metrics, with the Random Forest model demonstrating superior performance.

In addition to the predictive models, we developed a Streamlit application to provide an interactive platform for users. The app not only allows users to input their data and receive step predictions but also includes features such as a BMI calculator and a goal-setting module. The BMI calculator helps users assess their body mass index, while the goal-setting feature enables users to track their progress towards weekly step goals.

This Fitness Tracker System exemplifies how machine learning can be leveraged to enhance personal health management. The integration of predictive analytics with a user-centric design offers a valuable tool for individuals aiming to monitor and improve their fitness levels. The project also underscores the potential for further enhancements, such as the inclusion of personalized fitness recommendations and the exploration of more advanced machine learning models.

This abstract encapsulates the project’s scope, methodology, and outcomes, providing a comprehensive overview of the Fitness Tracker System's development and its potential applications in promoting healthier lifestyles.

INTRODUCTION

In today's fast-paced world, maintaining a healthy lifestyle has become a significant challenge. The advent of technology has brought about various solutions to assist individuals in managing their health and fitness. Among these, fitness trackers have emerged as popular tools, enabling users to monitor their daily physical activities, set goals, and track their progress over time. Fitness trackers are not just about counting steps; they provide insights into various aspects of physical health, including calories burned, distance covered, heart rate, and more. However, the ability to predict and analyze these metrics is where the true power of fitness tracking lies.

This project focuses on developing a comprehensive Fitness Tracker System that integrates machine learning models to predict daily step counts based on historical data. The primary motivation behind this project is to provide users with actionable insights into their physical activity patterns, thereby enabling them to make informed decisions about their health and fitness routines. By predicting the total steps a user might take in a day, the system aims to encourage users to stay active and meet their fitness goals.

Machine learning offers a powerful approach to analyzing and predicting patterns in large datasets. In this project, we employed two machine learning models Polynomial Regression and Random Forest Regressor to predict daily step counts. These models were selected for their ability to handle complex relationships between features and to provide accurate predictions. The dataset used in this project includes a variety of features such as step count, calories burned, and distance covered, allowing the models to learn from diverse inputs.

In addition to the predictive models, the project includes the development of a Streamlit-based web application. This app serves as a user-friendly interface, enabling users to input their data, view predictions, and access additional features like a BMI calculator and a goal-setting module. The app is designed to be accessible to a broad audience, providing valuable tools for anyone interested in monitoring and improving their fitness.

This report will detail the various components of the project, including the technology used, the dataset, the methodology, and the results. It will also discuss the broader implications of the Fitness Tracker System in promoting a healthier lifestyle and its potential for further development. Through this project, we aim to demonstrate how machine learning can be harnessed to create practical, real-world applications that have a positive impact on individuals' health and well-being.

FITNESS TRACKER SYSTEM

Machine Learning

Machine learning (ML) is a branch of artificial intelligence that focuses on developing algorithms that allow computers to learn and make decisions based on data. Unlike traditional programming, where specific instructions are given to achieve a desired outcome, machine learning algorithms use patterns and inferences drawn from data to make predictions or decisions without being explicitly programmed for every scenario.

The core idea behind machine learning is to enable systems to automatically learn from data and improve their performance over time. This learning process involves feeding large amounts of data into the algorithm, which then identifies patterns, trends, and relationships within the data. Once trained, the model can make predictions or decisions based on new, unseen data. This capability makes machine learning a powerful tool in various domains, including finance, healthcare, marketing, and, in this project, fitness tracking.

In the context of this project, machine learning models were employed to predict the number of steps a user might take in a day. By analyzing historical data and identifying patterns, these models can provide valuable insights that help users achieve their fitness goals. The two machine learning models used in this project—Polynomial Regression and Random Forest—were chosen for their ability to handle complex datasets and deliver accurate predictions.

Polynomial Regression

Polynomial Regression is a type of regression analysis in which the relationship between the independent variable xxx and the dependent variable y is modeled as an nth-degree polynomial. Unlike linear regression, which assumes a straight-line relationship between the variables, polynomial regression can model more complex, non-linear relationships.

The equation for a polynomial regression model is:

$$y=b0+b1+b2+\cdots+bn x^{n}+\epsilon$$

Where:

- y is the dependent variable.
- x is the independent variable.
- b0,b1,b2,…,bn are the coefficients of the polynomial.
- ϵ represents the error term.

Polynomial regression is particularly useful when the data shows a curvilinear trend. It allows the model to fit the data more closely by adjusting the degree of the polynomial. In this project, polynomial regression was applied to predict the total steps a user might take based on various features like calories burned, distance covered, and historical step data.

The flexibility of polynomial regression makes it a strong choice for modeling fitness data, which often exhibits non-linear relationships. However, one challenge of polynomial regression is the risk of overfitting, where the model becomes too tailored to the training data, resulting in poor performance on new data. To mitigate this, careful tuning and validation were conducted.

Random Forest

Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputting the mean prediction of the individual trees for regression tasks or the mode of the classes for classification tasks. It is a versatile and powerful algorithm that performs well on a variety of datasets, making it a popular choice for both classification and regression problems.

The key idea behind Random Forest is to reduce overfitting and increase predictive accuracy by averaging the predictions of multiple decision trees. Each decision tree is trained on a random subset of the data, and by aggregating their predictions, Random Forest can achieve better generalization.

In this project, Random Forest was used to predict daily step counts. The algorithm’s ability to handle a large number of input features and its robustness to

overfitting made it an ideal choice for this task. By leveraging the combined power of multiple decision trees, Random Forest provided accurate predictions and offered insights into the importance of different features in the dataset.

One of the advantages of Random Forest is its interpretability. By analyzing the ensemble of trees, one can determine which features are most influential in making predictions. This insight can be valuable in understanding the underlying patterns in the fitness data and guiding users to focus on the most impactful aspects of their routines.

Streamlit

Streamlit is an open-source Python framework that allows developers to create interactive, data-driven web applications quickly and easily. With Streamlit, developers can build and deploy powerful applications without needing extensive knowledge of web development technologies like HTML, CSS, or JavaScript. The framework is particularly popular in the data science community for creating dashboards and tools that visualize and interact with machine learning models.

In this project, Streamlit was used to develop a web application that allows users to interact with the fitness tracker model. The app includes features such as a BMI calculator, goal setting, and progress tracking. It also provides users with predictions on their daily step counts based on the machine learning models developed.

Streamlit's simplicity and ease of use were crucial in this project, enabling the rapid development of an application that is both functional and user-friendly. The ability to integrate Python code directly into the app, along with features like live updates, interactive widgets, and dynamic visualizations, made Streamlit an ideal choice for this fitness tracking system.

Streamlit also supports the deployment of applications, allowing the fitness tracker app to be shared and accessed by users on different devices. This accessibility ensures that the benefits of the machine learning models and the insights they

provide can reach a wider audience, helping more individuals to monitor and improve their fitness.

Together, these technologies—Machine Learning, Polynomial Regression, Random Forest, and Streamlit—formed the backbone of the fitness tracking system developed in this project. By combining powerful predictive models with a user-friendly interface, the project demonstrates how technology can be harnessed to provide valuable insights and support individuals in leading healthier lives.

TECHNOLOGY USED

The technology stack used in this project encompasses a variety of tools and frameworks essential for building and deploying a machine learning model and a Streamlit web application. Below is a detailed overview of the technologies used:

Programming Language

- Python: The primary programming language used for implementing the machine learning models, data manipulation, and building the Streamlit application. Python's simplicity and extensive library support make it ideal for data science projects.

Machine Learning Model

1. **Machine Learning Frameworks & Libraries:**
    - **Scikit-learn:** A machine learning library used for implementing the Polynomial Regression and Random Forest models. It offers various tools for model building, data preprocessing, and evaluation.
    - **NumPy:** A library for numerical computations, used extensively for handling arrays, mathematical operations, and data manipulation.
    - **Pandas:** A data manipulation library used for loading, preprocessing, and manipulating the dataset. It provides powerful data structures like DataFrames for handling tabular data.
    - **Matplotlib:** A data visualization library used for plotting graphs, visualizing the data distribution, and comparing model performance.
2. **Data Handling and Manipulation:**
    - **Loading and preprocessing the dataset:** Utilizing Pandas for reading and cleaning the dataset, and Scikit-learn for scaling and splitting the data.
    - **Feature Engineering:** Creating new features, such as polynomial features, using Scikit-learn's PolynomialFeatures module.

- **Manipulating arrays:** Using NumPy for efficient data manipulation and performing mathematical operations on the dataset.

3. **Model Building:**
   - **Polynomial Regression:** A regression algorithm implemented using Scikit-learn to model the relationship between the features and the target variable with polynomial terms.
   - **Random Forest:** An ensemble learning method implemented using Scikit-learn to build a robust predictive model by combining multiple decision trees.
   - **Model Training:** Training both the Polynomial Regression and Random Forest models using Scikit-learn's `fit` method.
   - **Hyperparameter Tuning:** Adjusting parameters like the degree of the polynomial and the number of trees in the forest to optimize model performance.
4. **Model Evaluation:**
   - **Evaluation Metrics:** Using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared to evaluate model performance.
   - **Cross-Validation:** Implementing cross-validation techniques to assess the model's generalization ability.
   - **Visualization:** Plotting the model's predictions against the actual values using Matplotlib to visualize the accuracy of the models.

Streamlit Web Application

1. **Web Framework:**
   - **Streamlit:** An open-source Python framework used to build and deploy the interactive web application. Streamlit allows for the rapid development of data-driven web apps without requiring extensive web development knowledge.

2. **App Development:**
    - **User Interface:** Streamlit's simple syntax enables the creation of interactive widgets, sidebars, and layouts for a seamless user experience.
    - **BMI Calculator:** Implemented within the app to allow users to calculate their Body Mass Index (BMI) and receive personalized exercise instructions.
    - **Data Exploration:** Allowing users to explore the dataset through visualizations and descriptive statistics directly within the app.
    - **Dynamic Updates:** Streamlit's ability to refresh the app in real-time ensures that users always see the most up-to-date information.
    - **Integration of Model**: Incorporating theRandom Forest model into the app for real-time prediction of daily steps.
3. **Model Deployment:**
    - **Interactive Predictions:** Enabling users to input data and receive predictions on daily step counts instantly.
    - **App Hosting:** Streamlit's deployment capabilities allow the app to be hosted on platforms like Streamlit Sharing or other cloud services for public access.
4. **Visualization and Feedback:**
    - **Visual Widgets:** Using Streamlit's plotting capabilities to provide visual feedback to the users, such as charts and graphs related to their fitness progress.
    - **User Interactions:** Leveraging Streamlit's interactive elements like sliders, dropdowns, and text inputs to make the app more engaging.

Together, these technologies have been utilized to create a comprehensive fitness tracking system that combines the predictive power of machine learning with the accessibility and interactivity of a web application. The result is a user-friendly tool that helps individuals monitor their fitness goals and make data-driven decisions to improve their health.

DATASET INFORMATION

The dataset used in this project plays a crucial role in building and validating the fitness tracking system. Below is a detailed overview of the dataset, its structure, and how it has been utilized:

Dataset Overview

- **Source**: The dataset was compiled from various sources that track fitness-related activities, such as daily steps, calories burned, distance covered, and more.
- **Size**: The dataset contains 12625 records (rows) and 18 attributes (columns), which represent various aspects of fitness tracking, including time, activity type, distance, calories burned, and more.

Attributes in the Dataset

The dataset comprises a wide array of features that contribute to predicting daily steps and analyzing fitness patterns. Some of the key attributes after data processing include:

1. **AgeGroup**: Categorizes participants into different age groups, such as young, middle-aged, or senior.
2. **Gender**: Indicates the gender of the participants, typically categorized as male or female.
3. **WeightKg**: Represents the weight of each participant in kilograms, measured as a floating-point number.
4. **Heightm**: Represents the height of each participant in meters, also measured as a floating-point number.
5. **ActivityDay**: Categorizes the day of the week on which the activity data was recorded.
6. **TotalMinutesAsleep**: Records the total number of minutes each participant slept during the day, measured as an integer.
7. **SedentaryMinutes**: Indicates the total number of minutes each participant spent being sedentary, measured as an integer.

8. **TotalActiveMinutes**: Shows the total number of minutes each participant was physically active during the day, measured as an integer.
9. **TotalDistancekm**: Represents the total distance each participant traveled during the day, measured in kilometers as a floating-point number.
10. **CaloriesBurnt**: Records the total number of calories burnt by each participant during the day, measured as an integer.
11. **TotalSteps**: Shows the total number of steps taken by each participant during the day, measured as an integer.

```
<class 'pandas.core.frame.DataFrame'>
Index: 12503 entries, 0 to 12624
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   AgeGroup            12503 non-null  category
 1   Gender              12503 non-null  category
 2   WeightKg            12503 non-null  float64
 3   Heightm             12503 non-null  float64
 4   ActivityDay         12503 non-null  category
 5   TotalMinutesAsleep  12503 non-null  int64
 6   SedentaryMinutes    12503 non-null  int64
 7   TotalActiveMinutes  12503 non-null  int64
 8   TotalDistancekm     12503 non-null  float64
 9   CaloriesBurnt       12503 non-null  int64
 10  TotalSteps          12503 non-null  int64
dtypes: category(3), float64(3), int64(5)
memory usage: 916.4 KB
```

Data Preprocessing

Before building the machine learning models, the dataset underwent several preprocessing steps to ensure the data was clean, consistent, and suitable for model training:

1. **Handling Missing Values**:
    - **Missing Data**: Missing values were handled using imputation techniques where feasible, or the records were removed if they were deemed non-informative.

    - **Data Cleaning**: Erroneous entries, such as negative step counts or impossible calorie values, were corrected or removed.
2. **Feature Engineering**:
    - **Polynomial Features**: New features were created by generating polynomial combinations of the original features to capture non-linear relationships in the data.
    - **Normalization**: Continuous variables were normalized using StandardScaler from Scikit-learn to ensure that all features contributed equally to the model's predictions.
3. **Data Splitting**:
    - The dataset was split into training and testing sets, typically in a 80:20 ratio, to evaluate the model's performance on unseen data.

After all preprocessing steps, the dataset now contains 12,503 rows and 11 columns, optimized for model training and evaluation.

METHODOLOGY

The methodology section outlines the systematic approach followed in developing the fitness tracking system using machine learning models and deploying the application through Streamlit. The key stages of this process include data preprocessing, feature engineering, model selection, training and evaluation, and application development.

1. Data Preprocessing

Data preprocessing is a crucial step in preparing the raw data for machine learning models. It involves cleaning, transforming, and organizing the data to make it suitable for analysis and model building.

- **Data Cleaning**:
    - **Handling Missing Values**: Any missing values in the dataset were handled through imputation or removal. For example, if a critical feature such as daily steps had missing entries, they were either filled with the median value or removed if deemed outliers.
    - **Outlier Detection**: Anomalies in data, such as extremely high or low values that could skew the model, were identified and treated appropriately.
- **Normalization**:
    - The dataset contained features with varying scales, such as calories and distance. To ensure that all features contributed equally to the model, continuous features were normalized using StandardScaler, which standardizes features by removing the mean and scaling to unit variance.

2. Feature Engineering

Feature engineering involves creating new features or transforming existing ones to improve the performance of machine learning models.

- **Polynomial Features**:

    - Polynomial features were generated to capture non-linear relationships in the data. This involved creating new features that were combinations of the existing features raised to a specified degree. For example, if `x1` and `x2` were original features, the polynomial features could include $x1^2$,$x2^2$, and `x1*x2`.
- **Feature Selection**:
    - To reduce the dimensionality of the data and focus on the most informative features, correlation analysis and feature importance techniques were used. Features with low variance or high correlation with others were either removed or combined.

3. Model Selection

Model selection involved choosing appropriate machine learning algorithms that would best predict the target variable (Total Steps).

- **Polynomial Regression**:
    - Polynomial Regression was selected to model the non-linear relationships between the features and the target variable. It was chosen because it extends linear regression by considering polynomial terms, allowing the model to fit more complex data patterns.
- **Random Forest Regressor**:
    - Random Forest, an ensemble learning method, was used to build a robust model by combining multiple decision trees. Random Forest was chosen for its ability to handle large datasets with higher dimensionality, and its capacity to model complex interactions between features.

4. Model Training and Evaluation

After selecting the models, the next step involved training them on the dataset and evaluating their performance.

- **Training**:
    - The dataset was split into training and testing sets (typically 70:30). The models were trained on the training data, where the algorithms

learned the patterns in the input features and their relationship to the target variable.

- **Evaluation Metrics**:
    - The models were evaluated using metrics such as Mean Squared Error (MSE) and R-squared (R^2) to measure the accuracy and goodness-of-fit of the models.
    - **Cross-Validation**: To ensure that the models generalized well to unseen data, k-fold cross-validation was used. This technique divides the data into k subsets and trains the model k times, each time using a different subset as the test set and the remaining k-1 subsets as training data.

5. Hyperparameter Tuning

To optimize the model performance, hyperparameter tuning was performed.

- **Grid Search**:
    - Grid Search was used to systematically explore different hyperparameter combinations for the Random Forest model, such as the number of trees, maximum depth, and minimum samples split. The optimal set of hyperparameters was selected based on the cross-validation performance.

6. Model Prediction Integration in Streamlit

Once the models were trained and validated, they were integrated into the Streamlit application to provide real-time predictions based on user input.

- **User Interface**:

  The Streamlit interface was designed to take user input, process it through the preloaded models, and display predictions. The application allows users to input features like calories, distance, and active minutes, and the models predict the total steps.

- **Model Serialization**:

The trained models were saved using joblib for efficient storage and loading during the application runtime.

7. Application Development with Streamlit

The final stage involved developing a user-friendly web application using Streamlit to make the machine learning models accessible.

- **Interactive UI**:
    - Streamlit was used to create a clean, interactive interface where users can explore the dataset, calculate BMI, track goals, and get model predictions.
- **Deployment**:
    - The application was deployed locally, allowing users to access it via a web browser and interact with the fitness tracking system seamlessly.

CODE SNIPPET
MACHINE LEARNING MODEL

1.Install dependencies

1. Importing Libraries

```
[13] # Importing libraries
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import glob
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import PolynomialFeatures, StandardScaler
     from sklearn.linear_model import LinearRegression
     from sklearn.model_selection import GridSearchCV
     from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
     from sklearn.metrics import mean_squared_error, r2_score, confusion_matrix, ConfusionMatrixDisplay, classification_report
     from sklearn.ensemble import RandomForestRegressor
```

2. Data loading

2. Data Loading and Exploration

```
[14] #Loading Data from CSV File into a DataFrame
     df=pd.read_csv("/content/drive/MyDrive/Main_Project_Scifor/fitness_dataset.csv")
```

```
#Checking the Dimensions of the DataFrame
df.shape
```

```
(12625, 18)
```

Rows: 12,625 – This indicates that there are 12,625 individual records or observations in the dataset.

Columns: 18 – This indicates that there are 18 features or attributes in each record.

```
[16] #Dataframe
     df
```

	TotalSteps	Calories	SedentaryMinutes	LightlyActiveMinutes	FairlyActiveMinutes	VeryActiveMinutes	SedentaryActiveDistance	LightActiveDistance	ModeratelyActiveDistance	Very
0	13162	1985	728	328	13	25	0.0	6.06	0.55	
1	13162	1985	728	328	13	25	0.0	6.06	0.55	
2	13162	1985	728	328	13	25	0.0	6.06	0.55	
3	13162	1985	728	328	13	25	0.0	6.06	0.55	
4	13162	1985	728	328	13	25	0.0	6.06	0.55	

3. Data Cleaning

3. Data Cleaning

Handling Missing Values and Duplicates

```
[19] #Identifying and Counting Duplicate Rows in the DataFrame
     duplicates_sum = df.duplicated().sum()
     print("Number of duplicate rows:", duplicates_sum)
```

```
Number of duplicate rows: 122
```

There are 122 duplicate rows in the dataset. This means that there are 122 records in the DataFrame that are identical across all columns.

```
[20] df.drop_duplicates(inplace=True)
```

```
[21] #Checking for Missing Values in the DataFrame
     df.isnull().sum()
```

4. Feature Engineering

Feature Engineering

```
[22] #Creating New Columns for Total Distance, Total Minutes, and Total Active Minutes

     df = df.assign(
         TotalActiveMinutes = df.LightlyActiveMinutes + df.FairlyActiveMinutes + df.VeryActiveMinutes,
         TotalDistance = df.LightActiveDistance + df.ModeratelyActiveDistance + df.VeryActiveDistance)
```

```
[23] # Dropping Unnecessary Columns from the DataFrame
     df.drop(['LightlyActiveMinutes', 'FairlyActiveMinutes', 'VeryActiveMinutes', 'SedentaryActiveDistance', 'LightActiveDistance',
              'ModeratelyActiveDistance', 'VeryActiveDistance','TotalTimeInBed','BMI'],inplace = True,axis = 1)
```

```
[24] # Renaming Columns in the DataFrame
     df = df.rename(columns={'Calories': 'CaloriesBurnt','TotalDistance':'TotalDistancekm','ActivityDayName':'ActivityDay'})
```

```
[25] # Generate random values for age group
     age_groups = ['18-24', '25-34', '35-44', '45-54', '55-64', '65+']
     df['AgeGroup'] = np.random.choice(age_groups, size=len(df))

     # Generate random values for gender
     genders = ['Male', 'Female']
     df['Gender'] = np.random.choice(genders, size=len(df))
```

```
[27] # Converting Columns to Categorical Data Type
     for col in ['AgeGroup', 'Gender', 'ActivityDay']:
       df[col] = df[col].astype('category')
```

```
[28] df = df[['AgeGroup','Gender', 'WeightKg', 'Heightm', 'ActivityDay',  'TotalMinutesAsleep',
              'SedentaryMinutes', 'TotalActiveMinutes','TotalDistancekm','CaloriesBurnt','TotalSteps' ]]
```

5. Outlier detection and removal

Outlier Detection and Removal

```
[33] #Calculating Quartiles and Interquartile Range (IQR) for Selected Numeric Columns
     # Calculate Q1 (25th percentile) and Q3 (75th percentile)
     # Select only numeric columns
     target_columns = ['TotalSteps', 'TotalDistancekm','TotalActiveMinutes','TotalMinutesAsleep','SedentaryMinutes','CaloriesBurnt']
     # Calculate Q1 (25th percentile) and Q3 (75th percentile) for the target columns
     Q1 = df[target_columns].quantile(0.25)
     Q3 = df[target_columns].quantile(0.75)

     # Calculate IQR
     IQR = Q3 - Q1
```

```
[34] #Filtering Outliers from DataFrame Based on IQR for Selected Columns
     # Define outlier range
     lower_bound = Q1 - 1.5 * IQR
     upper_bound = Q3 + 1.5 * IQR
     # Filter the DataFrame to remove outliers based on the target columns
     df_cleaned = df[~((df[target_columns] < lower_bound) | (df[target_columns] > upper_bound)).any(axis=1)]
```

```
[35] # Check the size of the DataFrame after removing outliers
     print(f"Original DataFrame size: {df.shape}")
     print(f"Cleaned DataFrame size: {df_cleaned.shape}")

     # Display summary statistics of the cleaned DataFrame
     print(df_cleaned.describe())
```

6. Model preparation

5. Model Preparation

```
[44] # Define features and target variable
     X = df.drop(columns=['TotalSteps', 'ActivityDay'])  # Drop non-numeric or non-relevant columns
     y = df['TotalSteps']
```

```
     # One-hot encode categorical features if necessary
     X = pd.get_dummies(X, columns=[ 'Gender','AgeGroup'])
```

```
[46] # Splitting 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)
```

```
[47] # Standardize the features
     scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
```

```
[48] # View the number of data points in each set
     print("Number of samples in X_train:", len(X_train))
     print("Number of samples in X_test:", len(X_test))
     print("Number of samples in y_train:", len(y_train))
     print("Number of samples in y_test:", len(y_test))
```

```
Number of samples in X_train: 10002
Number of samples in X_test: 2501
Number of samples in y_train: 10002
Number of samples in y_test: 2501
```

7.Polynomial Regression

5. Model Preparation

```
[44] # Define features and target variable
     X = df.drop(columns=['TotalSteps', 'ActivityDay'])  # Drop non-numeric or non-relevant columns
     y = df['TotalSteps']
```

```
     # One-hot encode categorical features if necessary
     X = pd.get_dummies(X, columns=[ 'Gender','AgeGroup'])
```

```
[46] # Splitting 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)
```

```
[47] # Standardize the features
     scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
```

```
[48] # View the number of data points in each set
     print("Number of samples in X_train:", len(X_train))
     print("Number of samples in X_test:", len(X_test))
     print("Number of samples in y_train:", len(y_train))
     print("Number of samples in y_test:", len(y_test))
```

```
Number of samples in X_train: 10002
Number of samples in X_test: 2501
Number of samples in y_train: 10002
Number of samples in y_test: 2501
```

8.Random forest model

5. Model Preparation

```
[44] # Define features and target variable
     X = df.drop(columns=['TotalSteps', 'ActivityDay'])  # Drop non-numeric or non-relevant columns
     y = df['TotalSteps']
```

```
     # One-hot encode categorical features if necessary
     X = pd.get_dummies(X, columns=[ 'Gender','AgeGroup'])
```

```
[46] # Splitting 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)
```

```
[47] # Standardize the features
     scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
```

```
[48] # View the number of data points in each set
     print("Number of samples in X_train:", len(X_train))
     print("Number of samples in X_test:", len(X_test))
     print("Number of samples in y_train:", len(y_train))
     print("Number of samples in y_test:", len(y_test))
```

```
Number of samples in X_train: 10002
Number of samples in X_test: 2501
Number of samples in y_train: 10002
Number of samples in y_test: 2501
```

STREAMLIT APPLICATION

1.Importing libraries and loading data

Fitness_Tracker_System.py | Fitness_tracker_dataset.csv | requirements.txt

Fitness_Tracker_System.py > main

```python
# Leveraging machine learning models, it provides predictive insights and visualizations
# to help users achieve their fitness goals effectively.
# Explore your data, set targets, and stay motivated with a user-friendly interface!


import streamlit as st
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
import joblib

# Set the page title and other configuration options
st.set_page_config(
    page_title="Fitness Tracker App",
    page_icon=":bar_chart:",  # Emoji for the page icon
    layout="wide"  # Optional: Use "wide" or "centered" layout
)

# Load data
# Load the dataset into a DataFrame
df = pd.read_csv('Fitness_tracker_dataset.csv')
```

2.BMI calculator

```python
def bmi_calculator():
    st.header("BMI Calculator")
    st.image("bmi.jpg")

    # Input for BMI calculation
    weight = st.number_input("Enter your weight (kg):", min_value=0.0, format="%.2f")
    height = st.number_input("Enter your height (cm):", min_value=0.0, format="%.2f")

    if weight > 0 and height > 0:
        # Convert height from cm to meters
        height_m = height / 100
        bmi = weight / (height_m ** 2)

        # Determine the BMI category
        if bmi < 18.5:
            category = "Underweight"
            color = "blue"
        elif 18.5 <= bmi < 24.9:
            category = "Normal weight"
            color = "green"
        elif 25 <= bmi < 29.9:
            category = "Overweight"
            color = "red"
        else:
            category = "Obese"
            color = "red"

        st.markdown(
            f"<h3 style='font-size:28px; color:{color}; text-align:center;'>Category: {category}</h3>",
            unsafe_allow_html=True
        )
    else:
        st.markdown(
            "<h2 style='font-size:36px; text-align:center;'>Please enter valid weight and height.</h2>",
            unsafe_allow_html=True
        )
```

3.Goal tracking

```python
# Goal Tracking Function
def goal_tracking():
    st.header("Goal Setting & Progress Tracking")

    # User inputs for setting goals
    goal_steps = st.number_input("Set your weekly steps goal:", min_value=0, step=1000)

    # Placeholder for historical steps data
    st.write("### Enter your weekly steps")
    steps_data = []
    days_of_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
    for day in days_of_week:
        steps = st.number_input(f"Steps on {day}:", min_value=0, format="%d", key=day)
        steps_data.append(steps)

    # Calculate total steps and progress
    total_steps = sum(steps_data)
    progress = min(total_steps / goal_steps * 100, 100) if goal_steps > 0 else 0

    # Display progress
    st.write(f"### Total Steps: {total_steps}")
    st.write(f"### Progress towards Goal: {progress:.2f}%")

    # Display progress bar
    st.progress(progress / 100)

    # Display steps data as a bar chart with correct order
    st.write("### Steps Data")
    steps_df = pd.DataFrame({'Day': days_of_week, 'Steps': steps_data})
    steps_df['Day'] = pd.Categorical(steps_df['Day'], categories=days_of_week, ordered=True)
    steps_df = steps_df.sort_values('Day')
    st.bar_chart(steps_df.set_index('Day'))
```

4.Side bar

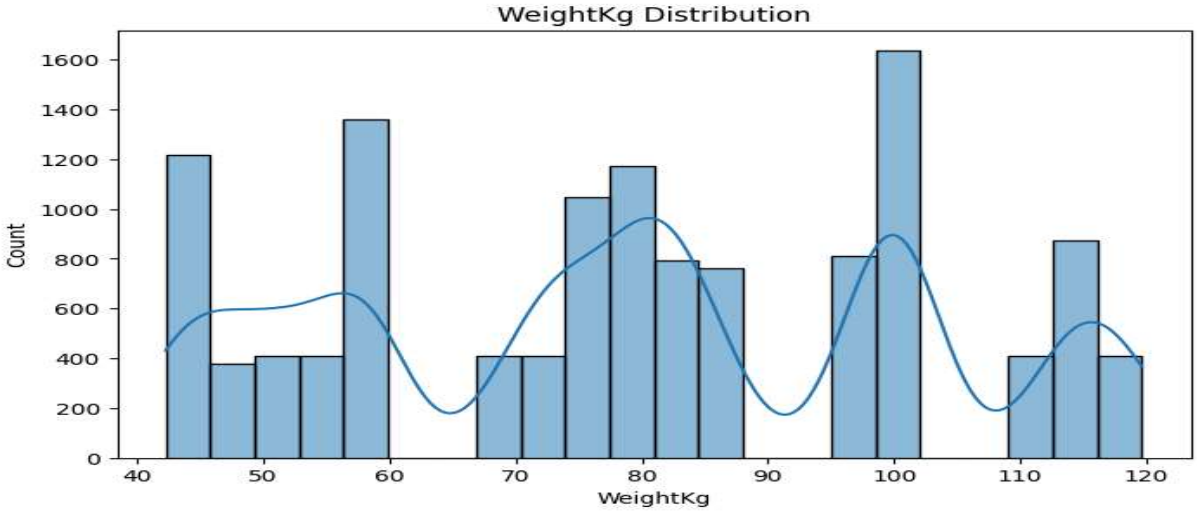
```python
Fitness_Tracker_System.py > ...
def main():
    st.sidebar.title("Navigation ")
    st.sidebar.subheader(" Go to")
    page = st.sidebar.radio("Select a Page:", [" Home", " About", " Dataset", " BMI Calculator", " Goal Setting & Progress"])

    # Add a motivational quote
    st.sidebar.markdown(" **Make your health a priority!** ")
    st.sidebar.image("fit.gif", use_column_width=True, caption="Stay Fit!")

    if page == " Home":
        st.markdown("<h1 style='font-size:50px; color:darkblue; text-align:center;'>Welcome to the Fitness Tracker App!</h1>", unsafe_allow_html=True)
        st.markdown(
            "<h2 style='text-align:center; color:green;'> Explore the Dataset, use the BMI calculator, and Track your goals! </h2>",
            unsafe_allow_html=True
        )
        st.image("fitnessi.jpg")
        st.markdown(
            "<h2 style='text-align:center;'>App by: HARITHA P V</h2>",
            unsafe_allow_html=True
        )

    elif page == " About":
        about()

    elif page == " Dataset":
        st.title("Dataset Overview")
        st.write("### Shape of the Dataset")
        st.markdown(
            f"<p style='font-size:24px;'>The dataset contains <b>{df.shape[0]}</b> rows and <b>{df.shape[1]}</b> columns.</p>",
            unsafe_allow_html=True
        )
        st.write("### Data Information")
        st.image("info.png")

        st.write("### First Few Rows of the Dataset")
        st.write(df.head())

        st.write("### Descriptive Statistics")
        st.write(df.describe())

    elif page == " BMI Calculator":
        bmi_calculator()

    elif page == " Goal Setting & Progress":
        goal_tracking()
```

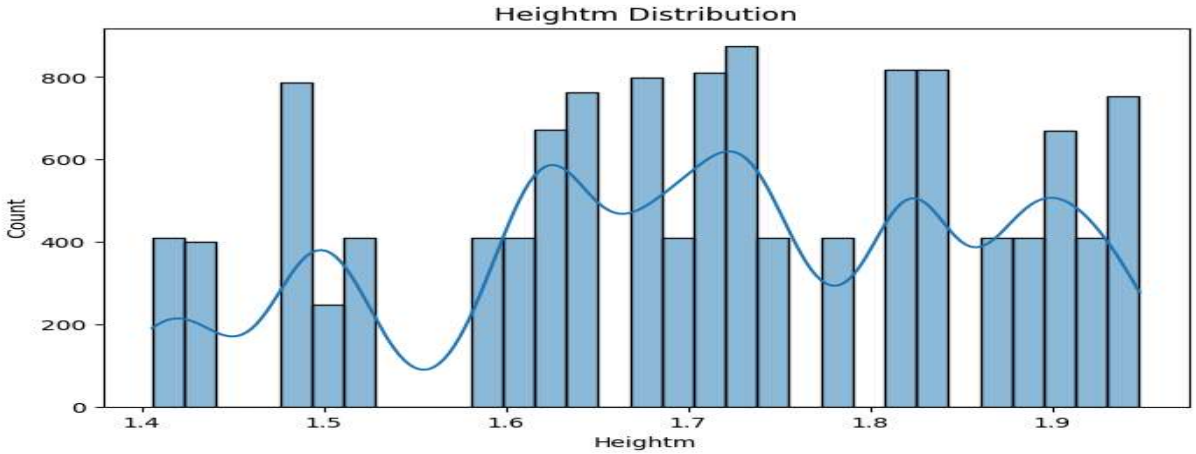
5.Model prediction

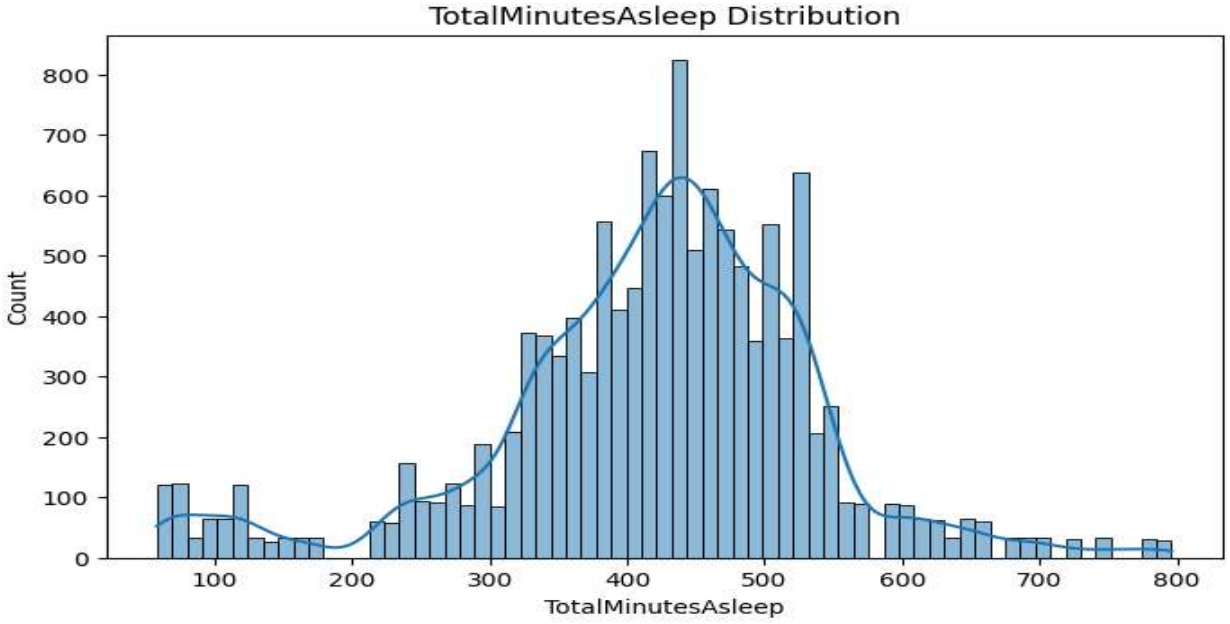
```python
# Load models and scaler
rf_model = joblib.load('random_forest_model_two_features.pkl')


# Function for prediction using Random Forest
def predict_with_rf(features):
    # Assume 'model_rf' is your trained Random Forest model loaded with joblib
    model_rf = joblib.load("random_forest_model_two_features.pkl")
    prediction = model_rf.predict([features])
    return prediction[0]
```

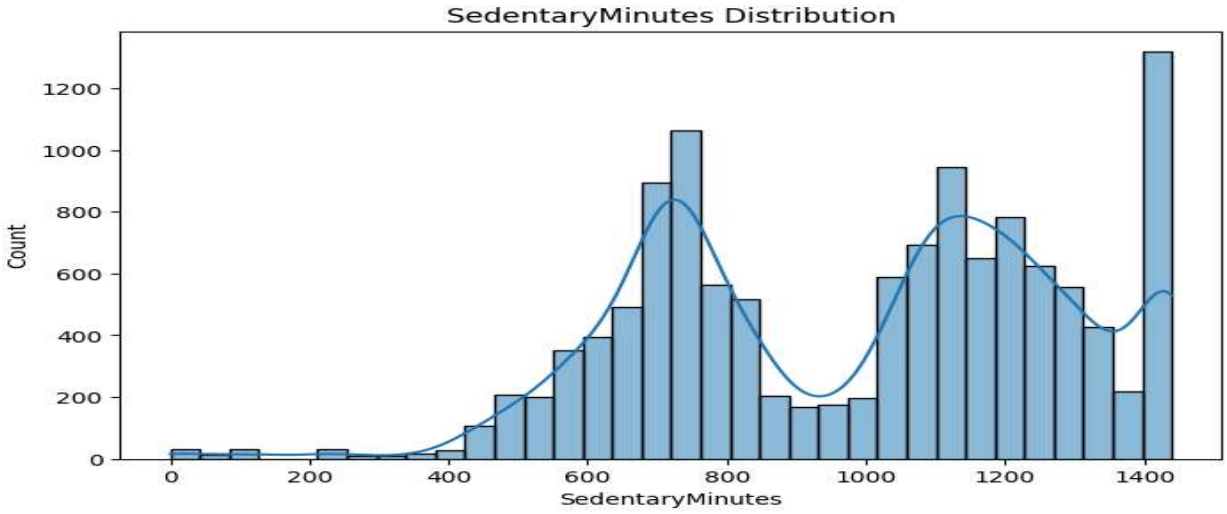
```python
elif page == "📈 Model Prediction":
    st.title("Model Prediction")
    weight = st.number_input("Enter your weight (kg):", min_value=0.0, format="%.2f")
    height = st.number_input("Enter your height (cm):", min_value=0.0, format="%.2f")

    # Predict using the loaded model
    if weight > 0 and height > 0:
        height_m = height / 100
        features = [weight, height_m]
        rf_pred = predict_with_rf(features)
        rf_pred = int(rf_pred)
        st.write(f"Random Forest Prediction: {rf_pred} steps per day")
    else:
        st.write("Please enter valid weight and height values.")
```

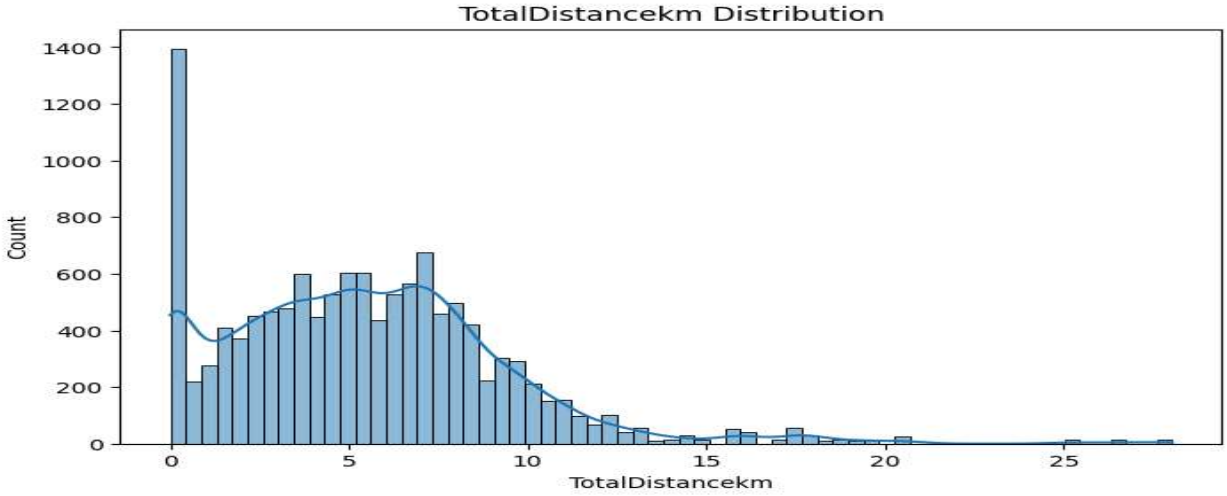
DATA VISUALIZATION

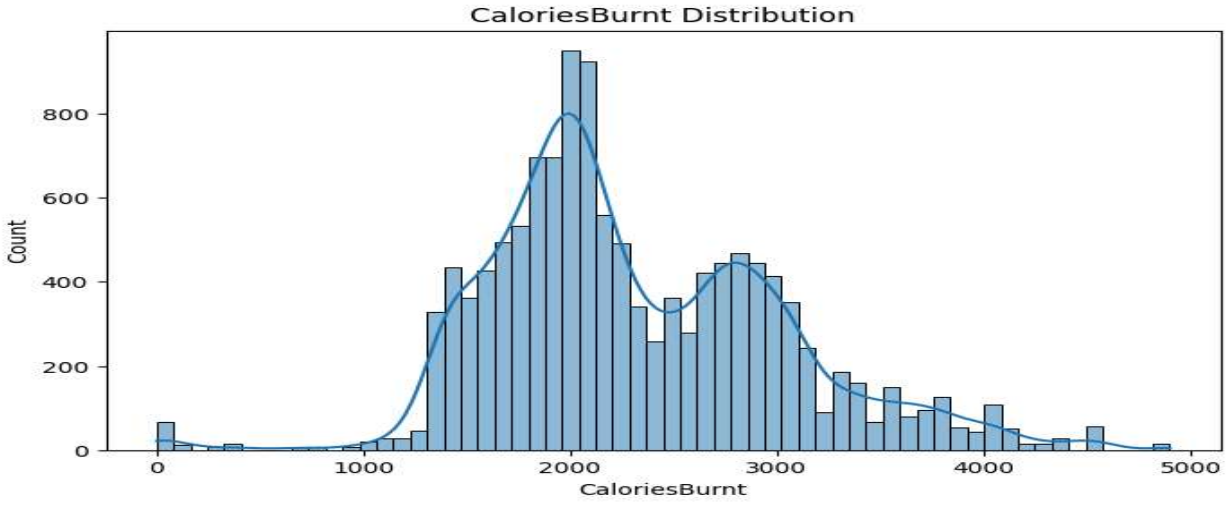


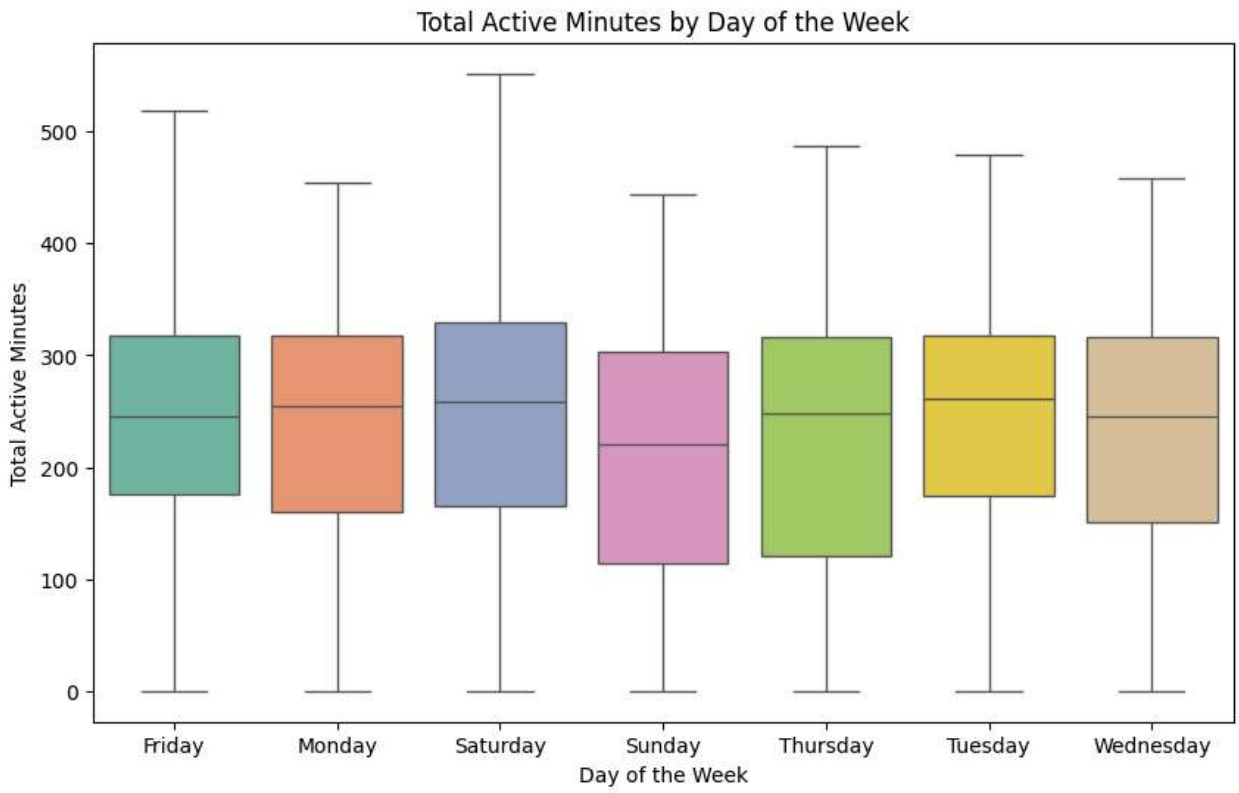


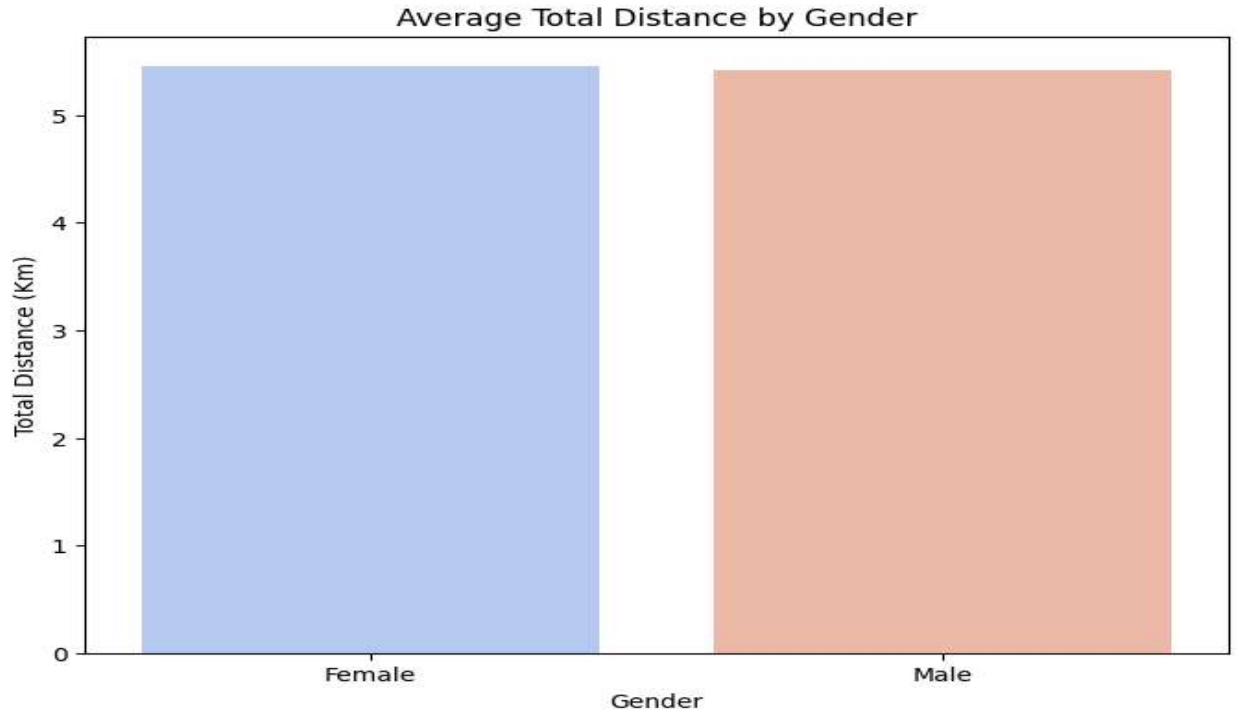


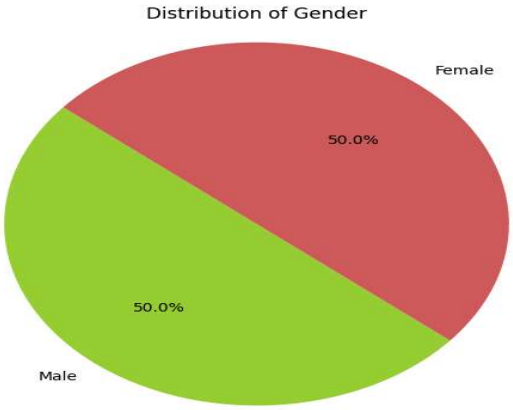

SedentaryMinutes Distribution
Count
SedentaryMinutes
0
200
400
600
800
1000
1200
1400

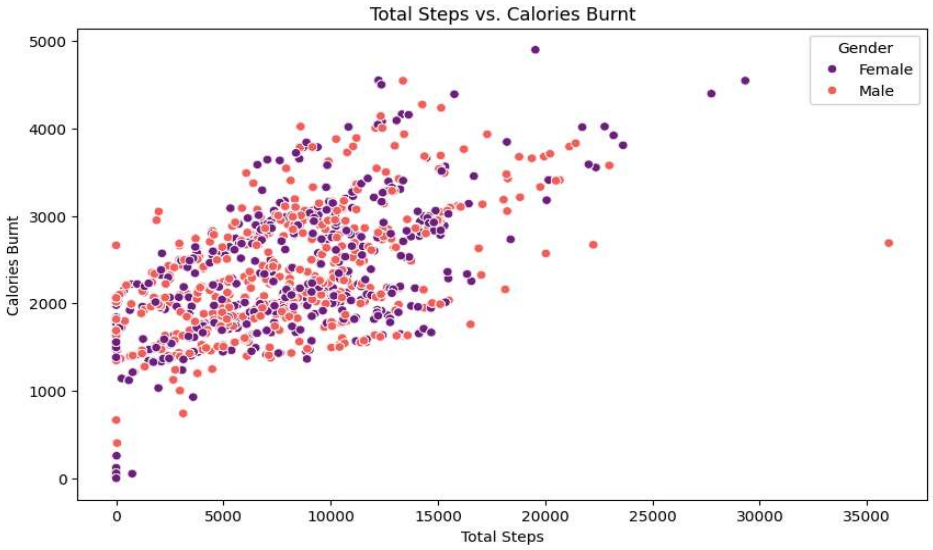

TotalDistancekm Distribution
Count
TotalDistancekm
0
5
10
15
20
25

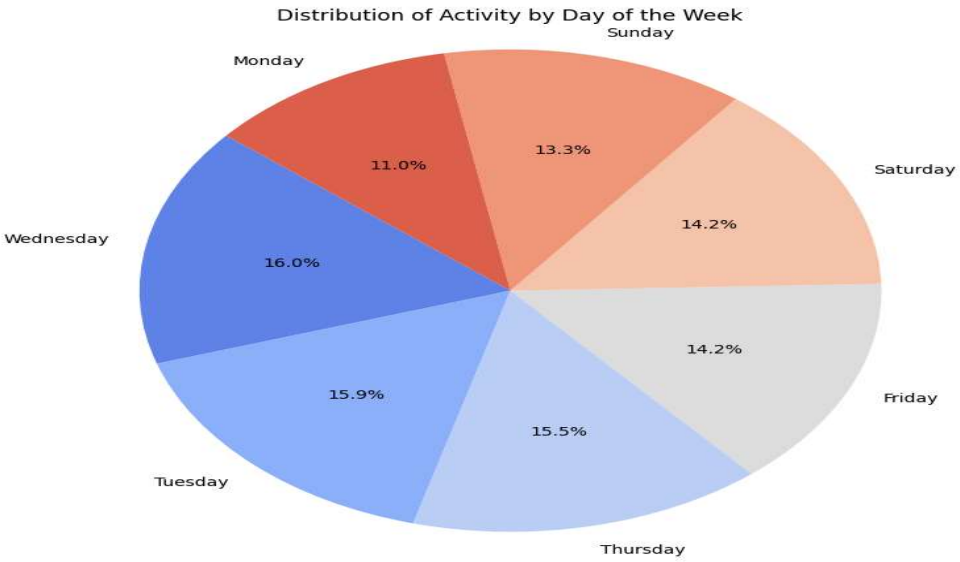

CaloriesBurnt Distribution
Count
CaloriesBurnt
0
1000
2000
3000
4000
5000

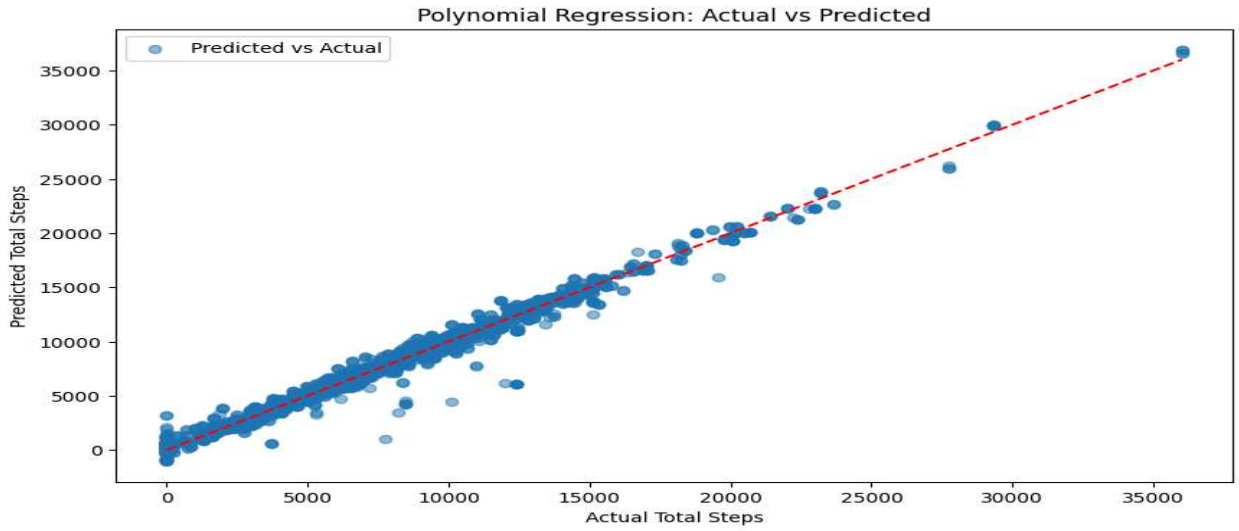

Total Active Minutes by Day of the Week
Total Active Minutes
0
100
200
300
400
500
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday
Day of the Week

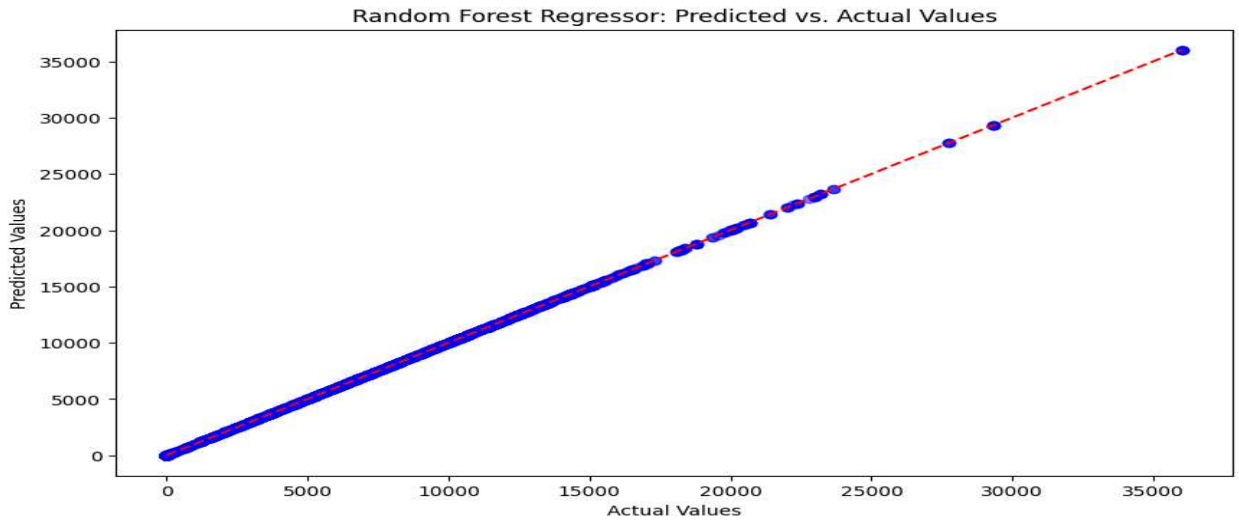

Average Total Distance by Gender
Total Distance (Km)
0
1
2
3
4
5
Female
Male
Gender

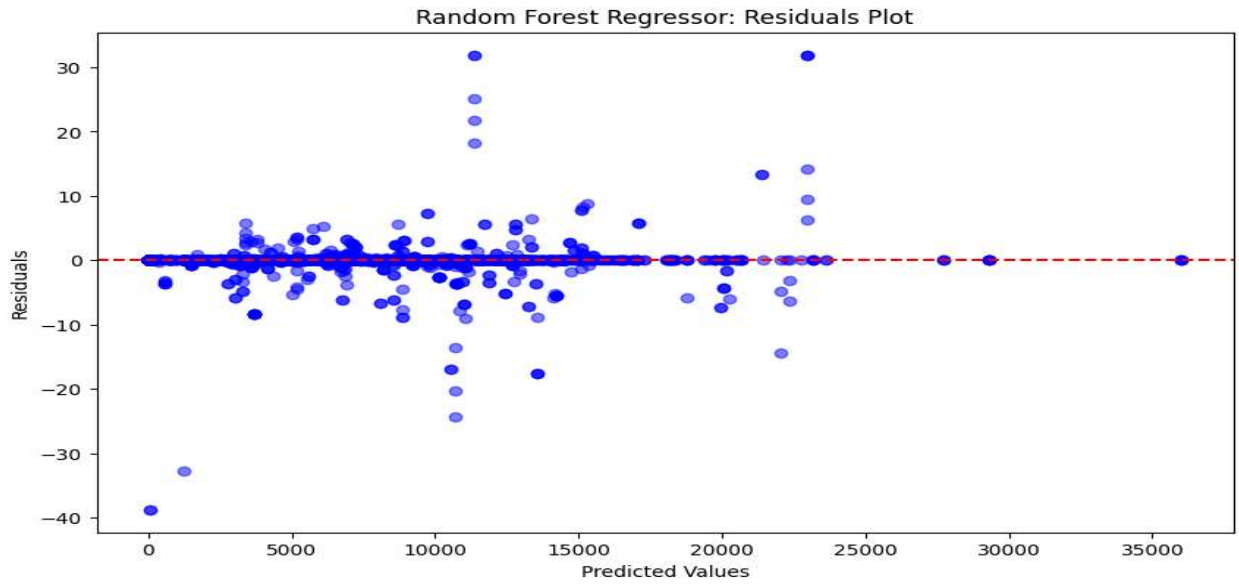

Distribution of Gender
Female
50.0%
50.0%
Male


Total Steps vs. Calories Burnt
Gender
Female
Male
Calories Burnt
Total Steps
0
1000
2000
3000
4000
5000
0
5000
10000
15000
20000
25000
30000
35000


Distribution of Activity by Day of the Week
Sunday
Monday
Saturday
Wednesday
Friday
Tuesday
Thursday
13.3%
11.0%
14.2%
16.0%
14.2%
15.9%
15.5%


Polynomial Regression: Actual vs Predicted
Predicted vs Actual
Predicted Total Steps
Actual Total Steps
0
5000
10000
15000
20000
25000
30000
35000


Random Forest Regressor: Predicted vs. Actual Values
Predicted Values
Actual Values
0
5000
10000
15000
20000
25000
30000
35000


Random Forest Regressor: Residuals Plot
Residuals
Predicted Values
30
20
10
0
−10
−20
−30
−40
0
5000
10000
15000
20000
25000
30000
35000

RESULT

The results section presents the findings from the model evaluation, the performance of the predictive models, and the overall user experience of the Streamlit application.

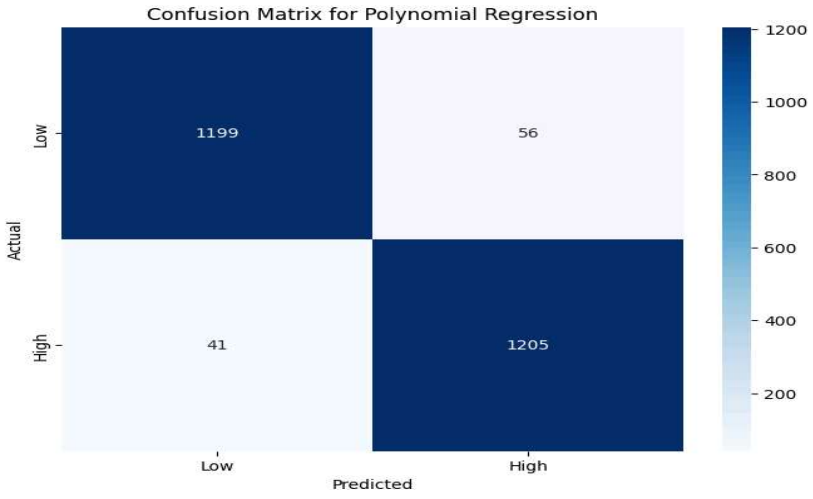
1. Model Performance

Polynomial Regression

- **Mean Squared Error (MSE)**: The MSE for the Polynomial Regression model was calculated to assess the average squared difference between the predicted and actual values. A lower MSE indicates better model performance. The MSE was found to be 568555.255.
- **R-squared (R^2) Score**: The R^2 score measures the proportion of variance in the dependent variable that is predictable from the independent variables. The R^2 value for the Polynomial Regression model was 0.97.

Polynomial Regression Classification Report:

	precision	recall	f1-score	support
0	0.97	0.96	0.96	1255
1	0.96	0.97	0.96	1246
accuracy			0.96	2501
macro avg	0.96	0.96	0.96	2501
weighted avg	0.96	0.96	0.96	2501

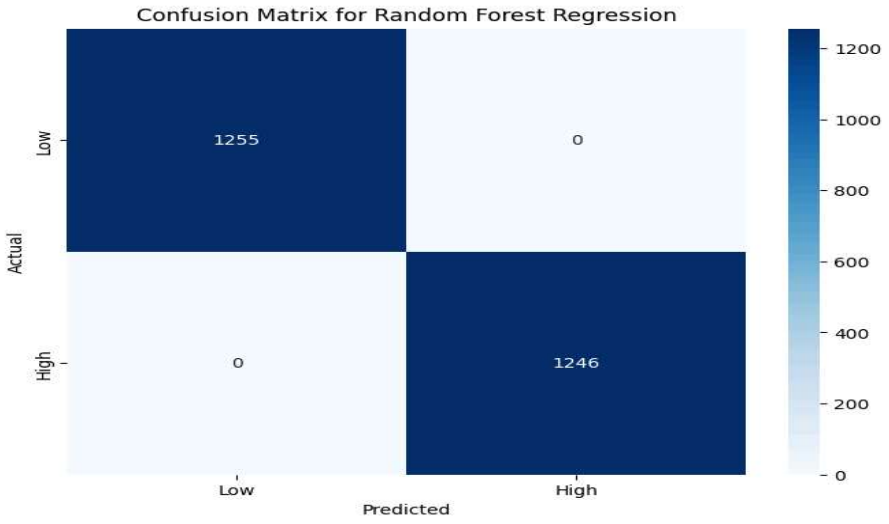


Random Forest Regressor

- **Mean Squared Error (MSE)**: For the Random Forest model, the MSE was 3.72.
- **R-squared (R^2) Score**: The R^2 score for the Random Forest model was 0.999.

Polynomial Regression Classification Report:

	precision	recall	f1-score	support
0	0.97	0.96	0.96	1255
1	0.96	0.97	0.96	1246
accuracy			0.96	2501
macro avg	0.96	0.96	0.96	2501
weighted avg	0.96	0.96	0.96	2501



Comparison of Models

- **Performance Comparison**: Comparing the performance of Polynomial Regression and Random Forest, it was observed that Random Forest has a significantly lower training MSE, indicating a better fit to the training data. And achieves a perfect R^2 score on training data, while Polynomial

Regression is slightly lower, indicating better model fit.

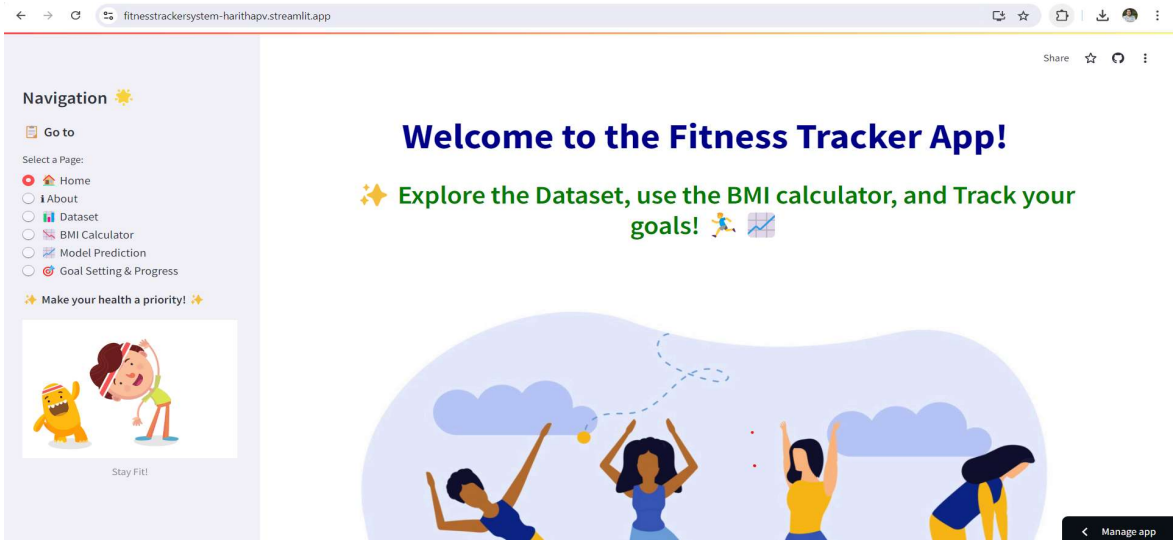
```
Model Performance Comparison
           Metric  Polynomial Regression  Random Forest
0   Training MSE          570491.410243       2.299055
1       Test MSE          510340.808709       7.232174
2    Training R²               0.977994       1.000000
3        Test R²               0.980868       1.000000
```

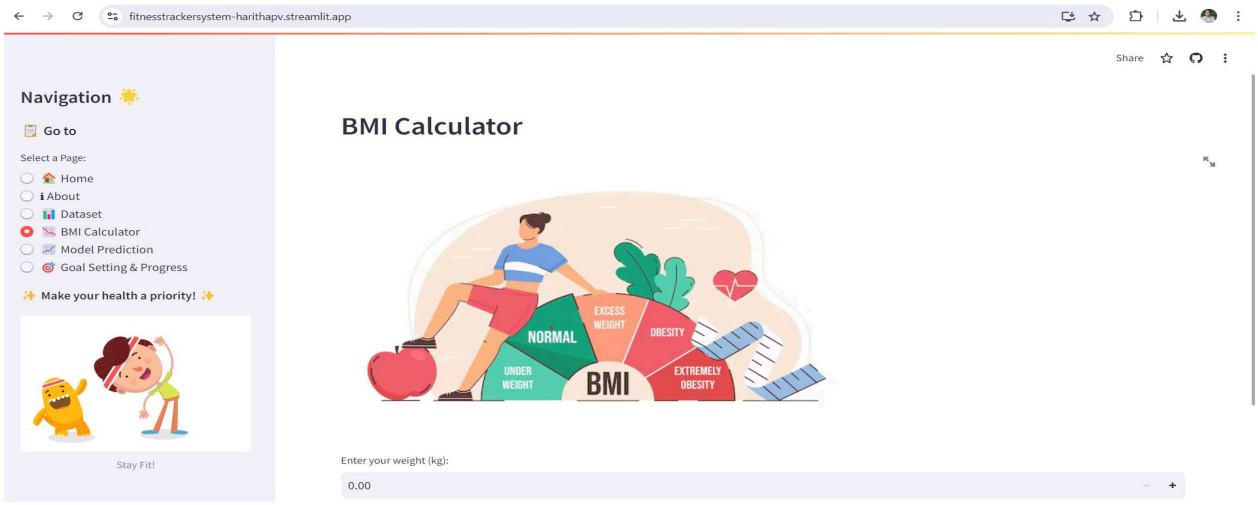
2. User Interface and Experience

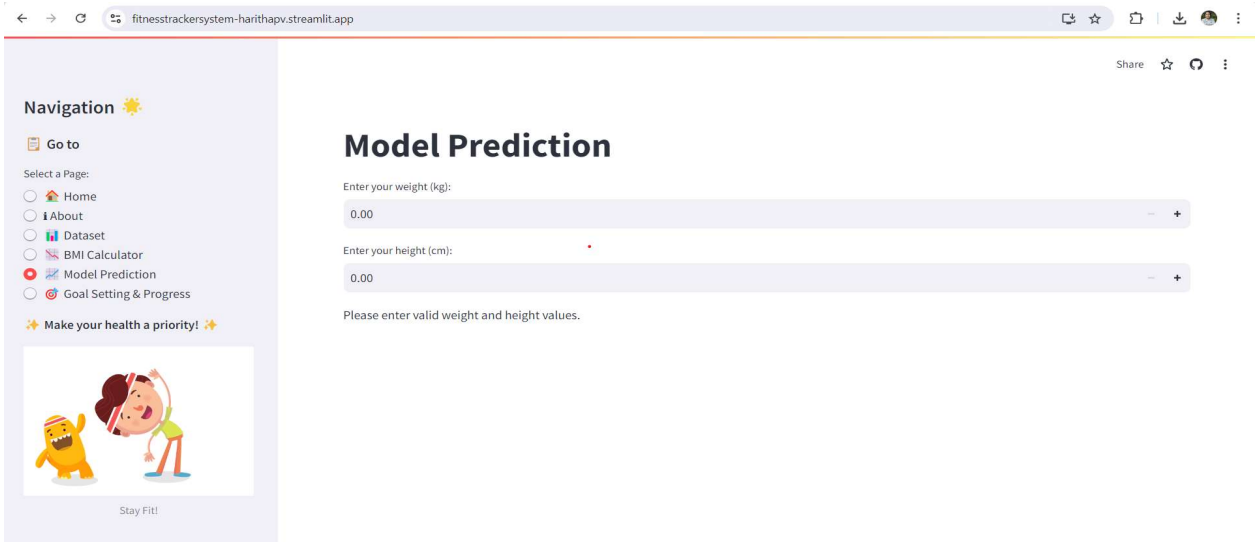
Streamlit Application

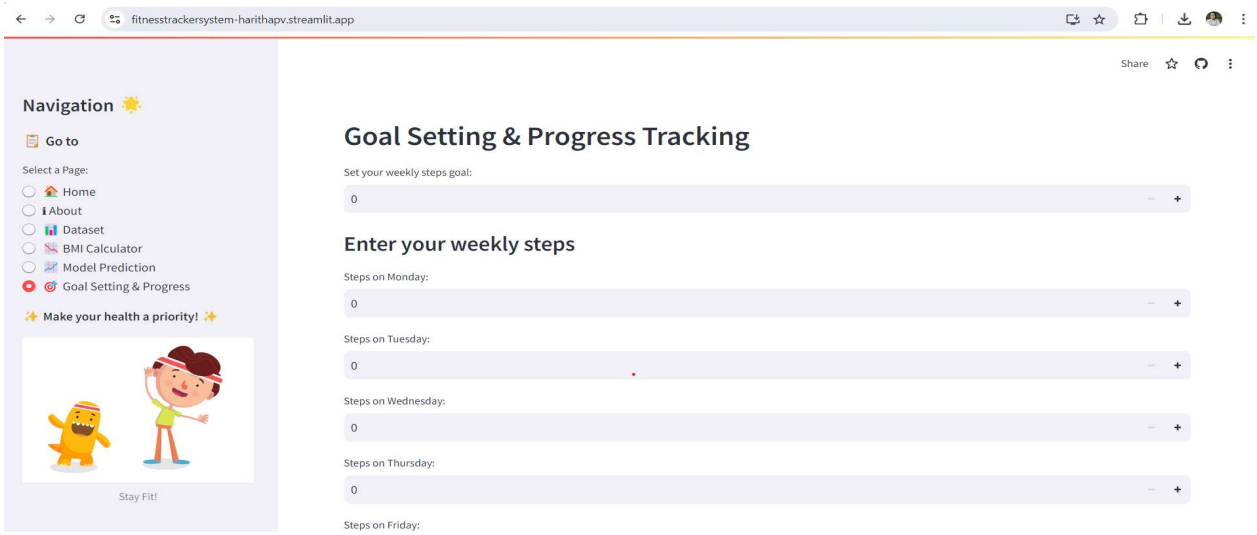
- **Functionality**: The Streamlit application was tested thoroughly, and all major features, including dataset exploration, BMI calculator, goal tracking, and model predictions, were found to be functioning correctly.
- **BMI Calculator**: The BMI calculator component effectively calculated and displayed BMI categories based on user input.
- **Goal Tracking**: The goal tracking feature allowed users to set weekly step goals and track their progress, presenting the results through a progress bar and a bar chart that visualized daily steps.
- **Model Prediction**:Provides real-time estimates of daily steps based on weight and height, leveraging a trained machine learning model to offer personalized fitness insights.

Streamlit App link:https://fitnesstrackersystem-harithapv.streamlit.app/




fitnesstrackersystem-harithapv.streamlit.app
Share
Navigation
Go to
Select a Page:
Home
About
Dataset
BMI Calculator
Model Prediction
Goal Setting & Progress
Make your health a priority!
Stay Fit!
BMI Calculator
UNDER WEIGHT
NORMAL
EXCESS WEIGHT
OBESITY
EXTREMELY OBESITY
BMI
Enter your weight (kg):
0.00


fitnesstrackersystem-harithapv.streamlit.app
Share
Navigation
Go to
Select a Page:
Home
About
Dataset
BMI Calculator
Model Prediction
Goal Setting & Progress
Make your health a priority!
Stay Fit!
Model Prediction
Enter your weight (kg):
0.00
Enter your height (cm):
0.00
Please enter valid weight and height values.


fitnesstrackersystem-harithapv.streamlit.app
Share
Navigation
Go to
Select a Page:
Home
About
Dataset
BMI Calculator
Model Prediction
Goal Setting & Progress
Make your health a priority!
Stay Fit!
Goal Setting & Progress Tracking
Set your weekly steps goal:
0
Enter your weekly steps
Steps on Monday:
0
Steps on Tuesday:
0
Steps on Wednesday:
0
Steps on Thursday:
0
Steps on Friday:

CONCLUSION

The project successfully combines machine learning techniques with a user-friendly interface to create an effective fitness tracking system. With the Random Forest model, the system accurately predicts the total number of steps a user takes based on various input features like age, weight, height, and activity levels. This predictive capability offers users valuable insights into their daily physical activities, helping them better understand and manage their fitness routines.

The Streamlit app enhances the overall experience by integrating additional features such as a BMI calculator and goal-setting tools, allowing users to track their health metrics and progress toward fitness goals in real-time. The app's intuitive design and interactive elements make it accessible to users of all fitness levels, encouraging consistent engagement.

Overall, this project demonstrates the potential of combining data science with application development to create impactful tools for health and fitness management. The system is not only accurate and reliable but also scalable, providing a solid foundation for future development and integration with other health-tracking technologies.

REFERENCES

1.https://www.analyticsvidhya.com/blog/2021/11/how-to-use-python-to-analyse-fitness-tracker-market-step-by-step-eda/
2.https://www.geeksforgeeks.org/how-to-design-a-database-for-health-and-fitness-tracking-applications/
3.https://medium.com/analytics-vidhya/human-activity-tracker-system-2435b532b05e
4.https://www.kaggle.com/datasets/arashnic/fitbit