Final Report of Traineeship Program 2023

On

“FITNESS DATA ANALYSIS”

MEDTOUREASY

29th July 2023

ACKNOWLEDGMENTS

The traineeship opportunity that I had with MedTourEasy was a great change for learning and understanding the intricacies of the subject of Data Visualisations in Data Analytics; and also, for personal as well as professional development. I am very obliged to have a chance to interact with so many professionals who guided me throughout the traineeship project and made it a great learning curve for me.

Firstly, I express my deepest gratitude and special thanks to the Training & Development Team of MedTourEasy who gave me an opportunity to carry out my traineeship at their esteemed organisation. Also, I express my thanks to the team for making me understand the details of the Data Analytics profile and training me in the same so that I can carry out the project properly and with maximum client satisfaction and also for spearing his valuable time in spite of his busy schedule.

TABLE OF CONTENTS

1.INTRODUCTION

1.1 ABSTRACT

Fitness is incredibly important for overall health and well-being. It encompasses various aspects of physical, mental, and emotional health, and plays a vital role in leading a fulfilling and active life. Here are some reasons why fitness is essential:

Physical Health: Regular exercise and physical activity help improve cardiovascular health, increase lung capacity, strengthen muscles, and enhance flexibility and balance. It also aids in maintaining a healthy body weight, reducing the risk of chronic conditions such as heart disease, diabetes, and obesity.

Mental Health: Physical activity releases endorphins, which are natural mood lifters, leading to reduced stress, anxiety, and depression. Regular exercise has been shown to enhance cognitive function, boost memory, and improve overall brain health.

Energy and Stamina: Being fit increases your energy levels and stamina, allowing you to perform daily tasks more efficiently and with less fatigue. Whether it's working, studying, or pursuing hobbies, being physically fit contributes to greater productivity.

Immune System: Regular exercise can strengthen the immune system, making it more effective in defending against infections and diseases.

Longevity: Studies consistently show that individuals who maintain good physical fitness tend to live longer and have a higher quality of life in their later years.

Improved Sleep: Regular physical activity can lead to better sleep patterns, helping you fall asleep faster and enjoy deeper, more restful sleep.

Weight Management: Exercise is an essential component of weight management, as it helps burn calories and build lean muscle, making it easier to maintain a healthy weight.

Bone Health: Weight-bearing exercises, such as walking and weightlifting, can improve bone density and reduce the risk of osteoporosis and fractures.

Social Benefits: Engaging in fitness activities can foster a sense of community and social interaction, especially when participating in group classes or team sports.

Confidence and Self-esteem: Achieving fitness goals and maintaining a healthy lifestyle can boost confidence and self-esteem, leading to a more positive self-image.

Disease Prevention: Regular exercise can help prevent and manage various health conditions, including type 2 diabetes, certain cancers, and hypertension.

Stress Relief: Physical activity is an excellent way to alleviate stress and unwind, improving mental clarity and emotional well-being.

1.2 About the Company

MedTourEasy, a global healthcare company, provides you the informational resources needed to evaluate your global options. MedTourEasy provides analytical solutions to our partner healthcare providers globally.

1.3 About the Project

In today's fast-paced and technology-driven world, the importance of fitness has become increasingly crucial. Unfortunately, with the rise of sedentary lifestyles, busy schedules, and unhealthy habits, the significance of prioritising physical health has diminished. Neglecting fitness poses several challenges that affect individuals, communities, and society as a whole.

One of the primary challenges of not giving importance to fitness is the alarming rise in lifestyle-related diseases. Obesity, heart disease, diabetes, and hypertension have reached epidemic proportions worldwide. A sedentary lifestyle coupled with poor dietary habits contributes to these health issues, putting an enormous strain on healthcare systems and impacting the quality of life for countless individuals.

In conclusion, the importance of fitness in modern times cannot be understated. The challenges posed by neglecting physical health are vast and encompass various aspects of individual and societal well-being. By placing a renewed emphasis on fitness and adopting healthier habits, we can enhance our overall quality of life, reduce the burden of lifestyle-related diseases, and build a stronger, more resilient society. Prioritizing fitness is not only a personal responsibility but also a collective effort towards a healthier and happier future.

By analysing and visualising fitness data, we can add value by gaining insights into individual and collective performance, identifying trends, setting achievable goals, and tracking progress over time. Data analysis allows us to pinpoint strengths and weaknesses, enabling personalised training plans and optimising workouts. Visualisation makes complex data accessible, motivating users and fostering a deeper understanding of their fitness journey. Integrating these tools into fitness apps or wearables empowers users to make informed decisions, stay accountable, and achieve better results, ultimately enhancing the overall fitness experience and promoting long-term adherence to healthy habits.

In that reference, it is extremely crucial to create visualisations which help firms to analyse this situation and to prepare themselves for the future. Additionally, MedTourEasy, being one of the globally upcoming tele-medicine companies in global healthcare, it is important for the firm to understand so as to gain more insights on the intensity of the fitness, the response of all countries and the impact it will have on their market. Also, depending on the results of the analysis, this may be used for increasing their market presence and capacity planning.

Hence, this project aims at collecting and analysing large data sets to create intuitive and interactive dashboards for representing Fitness data in order to gain meaningful insights.

2. Methodology

2.1 Language and Platform Used

Language: Python

Python is a high-level, versatile, and easy-to-read programming language. It is widely used for web development, data analysis, artificial intelligence, automation, and more. Python's simplicity, along with its rich ecosystem of libraries and frameworks, makes it a popular choice among developers.

Platform: Google Colab

Google Colab is an online platform provided by Google for running Python code through Jupyter notebooks. It allows users to write, execute, and share Python code in a cloud-based environment. Colab offers free access to computational resources, including GPUs and TPUs, making it suitable for machine learning tasks and data analysis. Its collaborative features enable real-time collaboration and easy sharing of interactive notebooks.

2.2 Project Description

With the explosion in fitness tracker popularity, runners all over the world are collecting data with gadgets (smartphones, watches, etc.) to keep themselves motivated. They look for answers to questions like:

- How fast, long, and intense was my run today?
- Have I succeeded with my training goals?
- Am I progressing?
- What were my best achievements?
- How do I perform compared to others?

This data was exported from Runkeeper
The data is a CSV file where each row is a single training activity
In this project, I'll create, import, clean, and analyse my data to answer these questions.

1.Obtain and review raw data
2.Data preprocessing
3. Dealing with missing values

4. Plot running data
5. Running statistics
6. Visualisation with averages
7. Did I reach my goals?
8. Am I progressing?
9. Training intensity
10. Detailed summary report.

3. Implementation

3.1 Libraries Used:

Pandas:

Pandas is a powerful and widely used open-source Python library for data manipulation and analysis. It provides data structures like DataFrame and Series that allow users to efficiently handle structured data. Pandas offers a wide range of functions to clean, transform, and merge datasets, making it an essential tool for data wrangling tasks. Its capabilities include handling missing data, data alignment, grouping, pivoting, and more, making it an indispensable library for data scientists and analysts.

Matplotlib:

Matplotlib is a popular data visualisation library in Python. It enables users to create a wide variety of charts, plots, and graphs, helping to visualise data effectively. With Matplotlib, you can generate line plots, scatter plots, bar charts, histograms, pie charts, and more, with extensive customization options. It is often used in combination with Pandas for visualising data stored in DataFrames and Series, making it a go-to tool for data exploration and presentation.

Statsmodels:

Statsmodels is a Python library that focuses on statistical modelling and hypothesis testing. It provides a comprehensive set of tools for performing various statistical analyses, including linear regression, time series analysis, ANOVA, and more. With Statsmodels, users can estimate model parameters, conduct hypothesis tests, and obtain valuable statistical information from their data. It is widely used in the fields of econometrics, social sciences, and finance to gain insights from data and make data-driven decisions.

3.2 Functions Used:

print():
Used to display output on the console.

sum():
Calculates the sum of elements in a sequence (e.g., list, tuple).

set_title():
In the Matplotlib library, set_title() is a method used to set the title of a plot. Matplotlib is a popular data visualisation library in Python that allows users to create various types of charts and graphs.

print():
To display messages, variables, or any other information to the console during the execution of the Python program.

sample():
sample() function in Python, which is part of the random module. The sample() function is used to randomly select items from a given sequence without replacement. It returns a new list containing the randomly selected items.

Slice():
This function is used to extract rows by position.

Filter():
This function is used to extract rows that meet a certain logical criteria. Logical Comparisons:
<: for less than
: for greater than
<=: for less than or equal to
>=: for greater than or equal to
==: for equal to each other
!=: not equal to each other
%in%: group membership. For example, “value %in% c(2, 3)” means that value can takes 2 or 3.

3.3 Methods Used:

isnull() in Pandas:
The isnull() method is used to identify missing or NaN (Not a Number) values in a Pandas DataFrame or Series. It returns a boolean mask, where each element is True if the corresponding value is NaN or False if the value is not NaN.

drop() in Pandas:

The drop() method is used to remove rows or columns from a Pandas DataFrame. By default, it drops rows, but you can specify the axis parameter to drop columns as well.

value_counts() in Pandas:

The value_counts() method is used to count the occurrences of unique values in a Pandas Series. It returns a new Series with the unique values as the index and their corresponding counts as the values.

sort_index() in Pandas:

The sort_index() method is used to sort the elements of a Pandas DataFrame or Series based on their index labels. For DataFrames, it can sort the rows based on the index, while for Series, it sorts the values based on their corresponding index labels.

mean() in Pandas:

The mean() method is used to calculate the arithmetic mean of the elements in a Pandas Series or DataFrame along a specified axis. For numeric data, it returns the average of the values.

set_title() in Matplotlib:

In the Matplotlib library, set_title() is a method used to set the title of a plot. Matplotlib is a popular data visualisation library in Python that allows users to create various types of charts and graphs.

show() in Matplotlib:

In the Matplotlib library, the show() method is used to display the plot that has been created. After creating a plot using various functions like plot(), scatter(), hist(), etc., you use show() to render the plot on the screen.

legend() in Matplotlib:

In the Matplotlib library, the legend() method is used to add a legend to the plot, which provides labels for the different elements in the chart.

xlabel() and ylabel()

Are methods provided by data visualisation libraries like Matplotlib and Seaborn in python. They are used to set labels for the x-axis and y-axis, respectively, in a plot or chart, providing context and meaning to the plotted data.

3.4 Data Filtering

Data filtering is the method of choosing a smaller portion of the data set and using that subset to view, analyse and evaluate data. Generally, filtering is temporary – the entire data set is retained, but only part of it is used for calculation. It is also called subsetting or drill down data wherein data is extracted with respect to certain defined logical conditions. Filtering is used for the following tasks:

- Analysing results for a particular period of time.
- Calculating results for particular groups of interest.

- Exclude erroneous or "bad" observations from an analysis. Train and validate statistical models.

With respect to the Fitness dataset, the data needs to be filtered according to certain conditions like years between 2013 to 2018.

4.1 Task 1

```
df_activities.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 508 entries, 2018-11-11 14:05:12 to 2012-08-22 18:53:54
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Activity Id             508 non-null    object
 1   Type                    508 non-null    object
 2   Route Name              1 non-null      object
 3   Distance (km)           508 non-null    float64
 4   Duration                508 non-null    object
 5   Average Pace            508 non-null    object
 6   Average Speed (km/h)    508 non-null    float64
 7   Calories Burned         508 non-null    float64
 8   Climb (m)               508 non-null    int64
 9   Average Heart Rate (bpm)  294 non-null  float64
 10  Friend's Tagged         0 non-null      float64
 11  Notes                   231 non-null    object
 12  GPX File                504 non-null    object
dtypes: float64(5), int64(1), object(7)
memory usage: 55.6+ KB
```

In task 1 we imported pandas and also imported read files. We printed the summary of the dataframe by obtaining valuable information about the data contained within the DataFrame. When called on a DataFrame, this method displays a summary of the DataFrame's structure, including the number of non-null values, the data types of each column, and the memory usage. It helps us quickly assess the data and identify potential data quality issues, such as missing values and incorrect data types.

4.2 Task 2

```
[ ]  cols_to_drop = ['Route Name', "Friend's Tagged"]

[ ]  df_activities = df_activities.drop(columns=cols_to_drop)

     df_activities.columns.tolist()

['Activity Id',
 'Type',
 'Distance (km)',
 'Duration',
 'Average Pace',
 'Average Speed (km/h)',
 'Calories Burned',
 'Climb (m)',
 'Average Heart Rate (bpm)',
 'Notes',
 'GPX File']
```

In task 2 we did data Cleaning and found two Columns which were completely empty; those are ‘Route Named’ and ‘Friends Tagged’. So we deleted those columns. Calculated the activity type counts on type columnan renamed ‘other’ values to ‘Unicycling’ in type column.

4.3 Task 3

TASK 3

```
[ ]  cycling_data = df_activities[df_activities['Type'] == 'Cycling']
```

```
cycling_data['Average Heart Rate (bpm)'].mean()
```

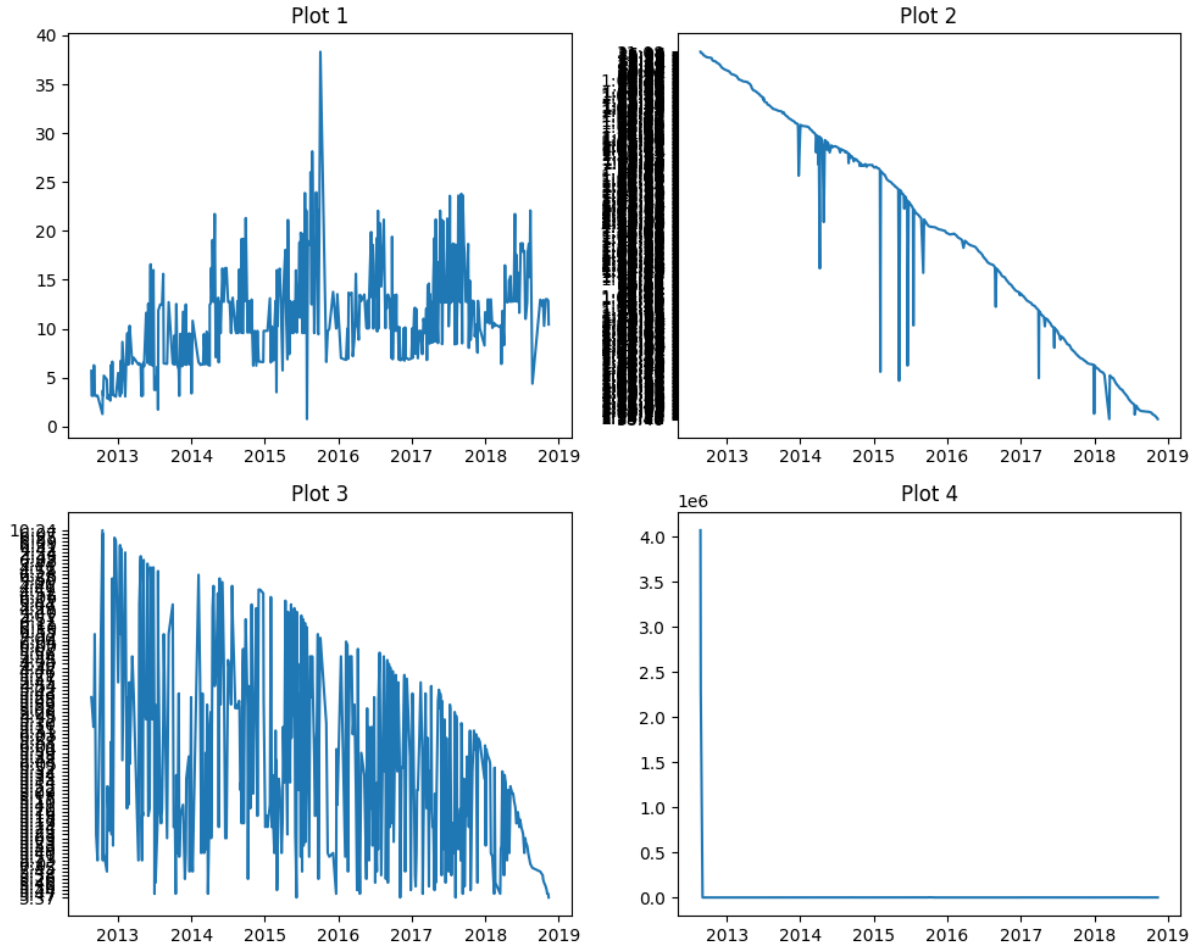
```
124.4
```

```
[ ]  df_cycle = df_activities[df_activities['Type'] == 'Cycling'].copy()
```

```
[ ]  avg_hr_cycle = df_cycle['Average Heart Rate (bpm)'].mean()
     df_cycle['Average Heart Rate (bpm)'].fillna(int(avg_hr_cycle), inplace=True)
```

Here we Calculated the sample mean of average heart rate while cycling. We handled missing data by replacing the NaN values in a DataFrame or Series with calculated values.

4.4 Task 4



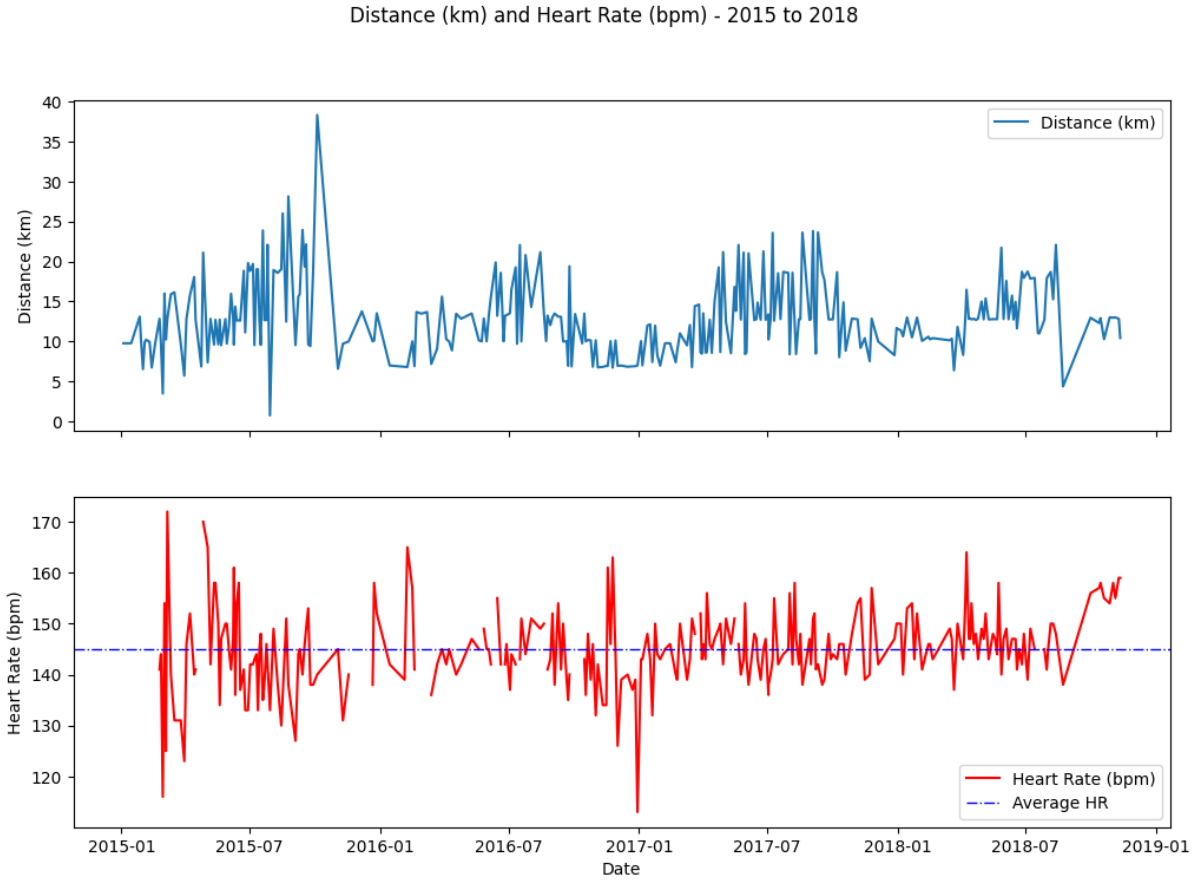
This graph contains a line plot of different columns from the DataFrame df_run_sorted.
The first subplot (Plot 1) displays a line plot of 'Distance' against the 'Date'.
The second subplot (Plot 2) displays a line plot of ‘Duration’ against the 'Date'.
The third subplot (Plot 3) displays a line plot of 'Average Pace' against the 'Date'.
The fourth subplot (Plot 4) displays a line plot of 'Calories Burned' against the 'Date'.

4.5 Task 5

```
Average Number of Trainings per Week:
Date
2012-08-26    2.0
2012-09-02    1.0
2012-09-09    2.0
2012-09-16    1.0
2012-09-23    1.0
              ...
2018-10-14    2.0
2018-10-21    1.0
2018-10-28    1.0
2018-11-04    2.0
2018-11-11    2.0
Freq: W-SUN, Name: Distance (km), Length: 325, dtype: float64
```

The output displays the annual means for the columns 'Distance (km)', 'Average Speed (km/h)', 'Climb (m)', and 'Average Heart Rate (bpm)' for the years 2015 to 2018.
The output will also show the average weekly statistics for the same columns, calculated on a weekly basis, providing insights into trends and variations over shorter time intervals. Additionally, the output will display the average number of training sessions per week based on the 'Distance (km)' column, showing the average frequency of fitness activities performed each week during the specified period. This output provides valuable summary statistics for the specified fitness data, helping to understand the overall trends and patterns of the fitness activities recorded in the DataFrame for the specified time range.

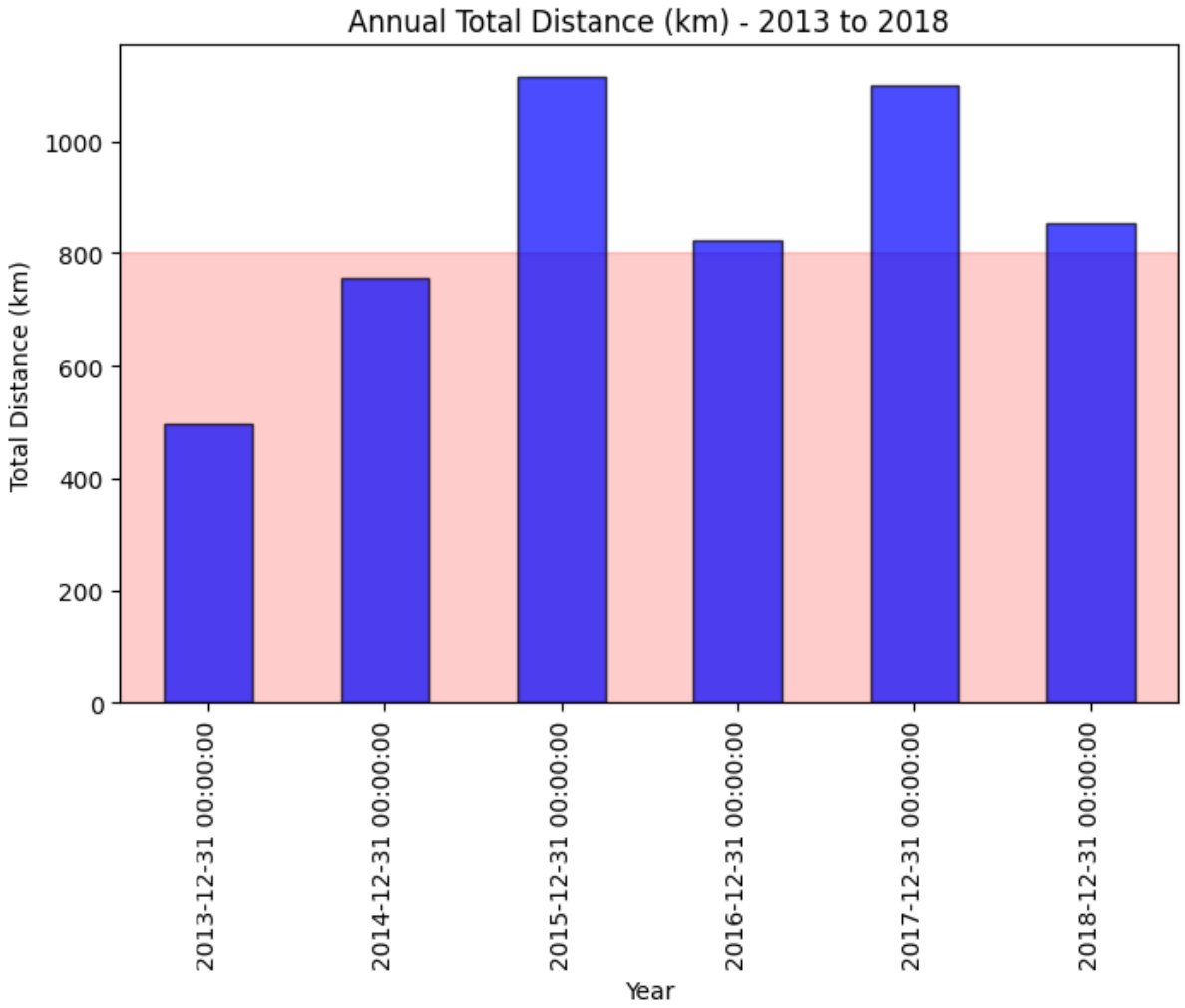
4.6 Task 6



The code checks for missing values in the 'Distance (km)' and 'Average Heart Rate (bpm)' columns of the subset DataFrame 'runs_subset_2015_2018' and prints the counts of missing values for each column. The code creates a plot with two subplots using plt.subplot(), one for 'Distance (km)' and the other for 'Heart Rate (bpm)'. It shows the trends of both variables over time from 2015 to 2018. The first subplot displays the variation in 'Distance (km)' over time, with the y-axis representing the distance in kilometres. It also includes a legend for clear identification.

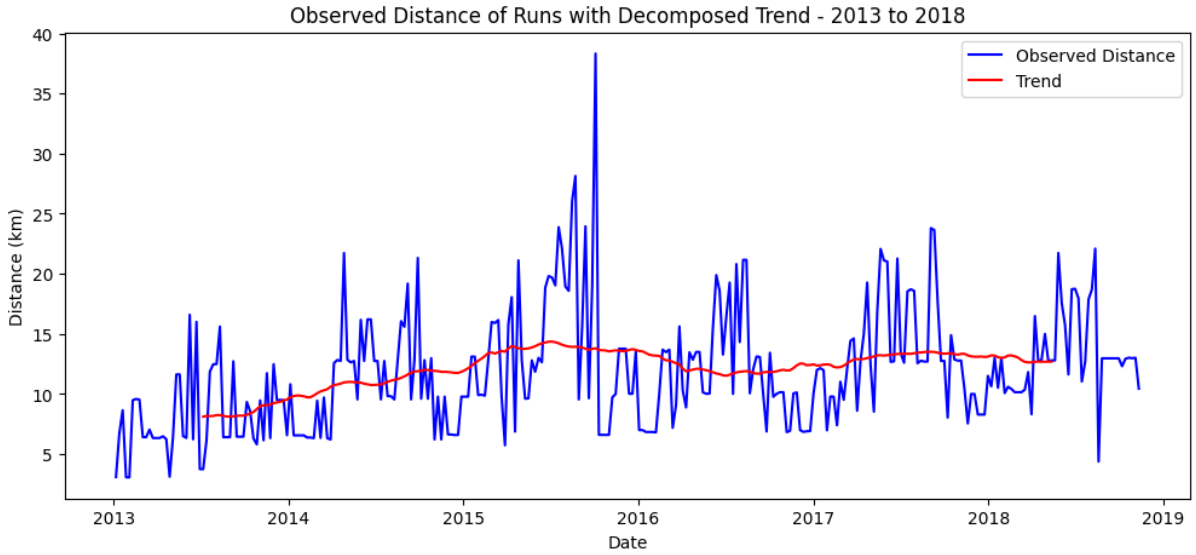
The second subplot shows the changes in 'Heart Rate (bpm)' over the same time range, with the y-axis representing the heart rate in beats per minute (bpm). It uses a red colour to differentiate the line plot and includes a legend. The plot has appropriate x and y-axis labels and a main title ('Distance (km) and Heart Rate (bpm) - 2015 to 2018') for better context.

4.7 Task 7



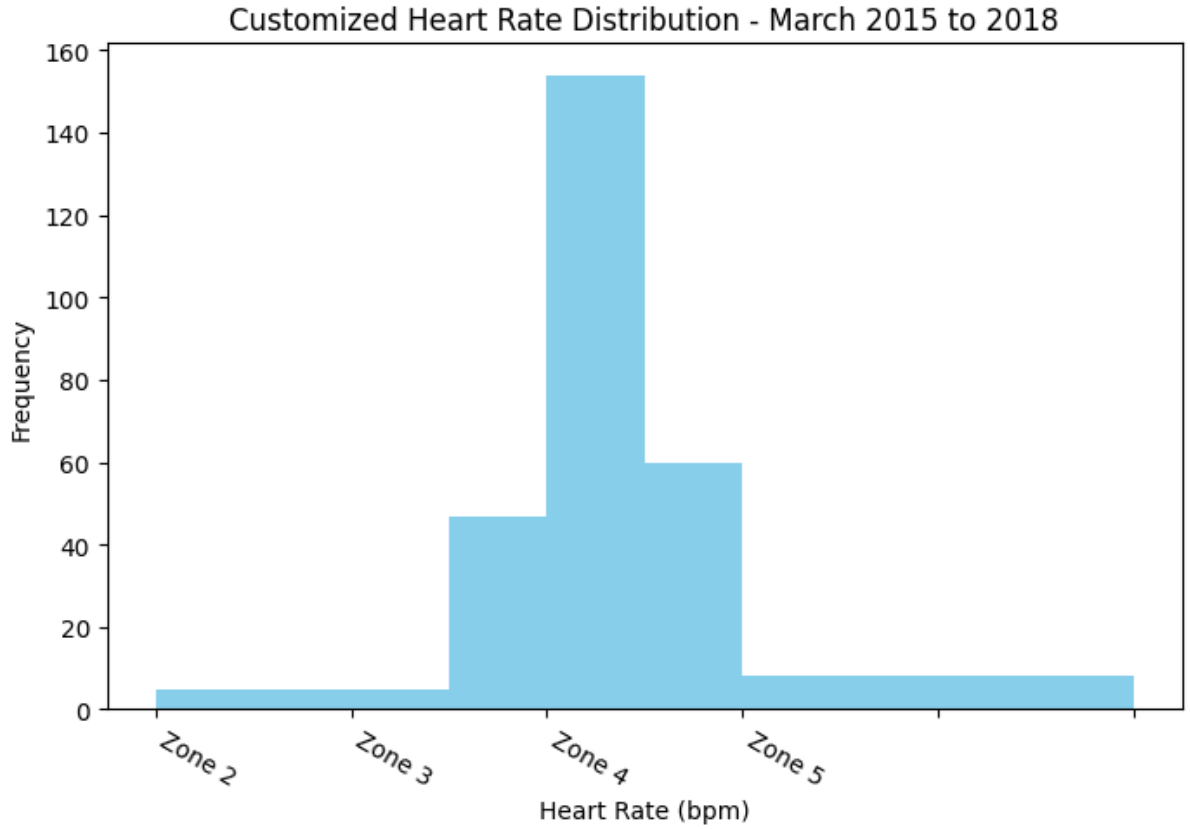
The code successfully subsets the 'df_run' DataFrame to include data for the years 2013 to 2018 and calculates the annual total distance covered in kilometres. The plot has a red-coloured horizontal span that spans from 0 to 800 km, highlighting the range within which most of the data falls. This provides a reference for understanding how the total distance compares to the defined range. The bar plot displays the annual totals of 'Distance (km)' using blue bars, with black edges and a transparency (alpha) of 0.7 for aesthetics. The x-axis represents the years (2013 to 2018), and the y-axis represents the total distance covered in kilometres. The plot has appropriate labels for the x and y axes, as well as a title ('Annual Total Distance (km) - 2013 to 2018') for clear identification. 2015 and 2017 has the highest distance covered.

4.8 Task 8



The code successfully subsets the 'df_run' DataFrame to include data for the years 2013 to 2018 and resamples the 'Distance (km)' column on a weekly basis. Time series decomposition is performed using the seasonal decomposition of time series (STL) method provided by the statsmodels library. The 'additive' model is used, assuming that the observed distance can be decomposed into trend, seasonal, and residual components. The plot has a size of 12 inches in width and 5 inches in height, providing a clear and appropriately sized visual representation. The blue line represents the observed distance of runs on a weekly basis, showing the actual variations in distance over time. The red line represents the decomposed trend, which highlights the underlying pattern or long-term trend in the observed distance. The x-axis represents the dates on a weekly basis, and the y-axis represents the distance covered in kilometres. The plot includes appropriate labels for the x and y axes, as well as a title ('Observed Distance of Runs with Decomposed Trend - 2013 to 2018') for clear identification. A legend is added to the plot, providing a clear distinction between the observed distance and the decomposed trend.

4.9 Task 9



The code successfully subset the 'df_run' DataFrame to include data for the specified period, i.e., from March 2015 to 2018. Custom x-axis tick labels ('Zone 1', 'Zone 2', 'Zone 3', 'Zone 4', 'Zone 5') are defined to represent different heart rate zones. The plot is created using plt.subplots(), with a figure size of 8 inches in width and 5 inches in height, providing an appropriate size for the visual representation. A histogram is created to display the frequency distribution of heart rates. The histogram is customised with specific bins (100-130, 130-140, 140-150, 150-160, 160-200 bpm) and a sky blue colour. The x-axis is labelled as 'Heart Rate (bpm)' to clearly identify the data represented. The y-axis is labelled as 'Frequency', indicating the number of occurrences of heart rates falling within each bin. The x-axis tick labels are set to the custom tick labels, 'Zone 1' through 'Zone 5', providing a more descriptive representation of heart rate zones. A title ('Customised Heart Rate Distribution - March 2015 to 2018') is added to the plot, specifying the period covered by the analysis. The plot is displayed successfully, presenting the customised histogram of heart rate distribution with clear labelling and proper bins, helping to visualise the distribution of heart rates within the specified period.

4.10 Task 10

```
df_run_walk_cycle = pd.concat([df_run, df_walk, df_cycle]).sort_index(ascending=False)
dist_climb_cols = ['Distance (km)', 'Climb (m)']
df_totals = df_run_walk_cycle.groupby('Type')[dist_climb_cols].sum()
df_summary = df_totals.stack()
print(df_summary)
```

```
Type
Cycling  Distance (km)      680.58
         Climb (m)         6976.00
Running  Distance (km)     5224.50
         Climb (m)        57278.00
Walking  Distance (km)       33.45
         Climb (m)          349.00
dtype: float64
```

The code successfully concatenates the 'df_run', 'df_walk', and 'df_cycle' DataFrames using the append() method. It combines all three DataFrames into a single DataFrame, 'df_run_walk_cycle', containing the data for running, walking, and cycling activities. The combined DataFrame, 'df_run_walk_cycle', is then sorted based on the index in descending order, which likely represents chronological order, with the most recent activities appearing first. The code groups the 'df_run_walk_cycle' DataFrame by activity type ('Running', 'Walking', and 'Cycling') using the groupby() method. The 'Distance (km)' and 'Climb (m)' columns are selected for each activity type, and their sums are calculated for each group using the sum() method. This step calculates the total distance and climb for running, walking, and cycling activities. The summary report, 'df_summary', is created by stacking the calculated totals to provide a compact, reshaped form of the full summary report. It displays the total distance and climb for each activity type in a single series. Finally, the summary report is printed, showing the total distance and climb for running, walking, and cycling activities separately, allowing for a quick overview of the fitness activities' cumulative statistics.

5. Conclusion

In conclusion, this fitness analysis project involved importing, cleaning, and analysing data from various fitness activities recorded on Runkeeper. The data was processed to handle missing values and organise it by activity type. Plots and visualisations were created to explore trends and patterns in distance, heart rate, and other variables over the years. The analysis revealed the average weekly and annual distances for running activities, as well as heart rate distribution during specific periods. Additionally, interesting facts like the instructor's average shoes per lifetime and an estimated number

of shoes for Forrest Gump's route were calculated. This analysis provided valuable insights into the user's fitness journey and training progress over time.

6. Future Scope

The future scope for fitness data analysis is promising and holds great potential for further exploration and advancements. With the increasing popularity of fitness trackers and wearable devices, the amount of data available for analysis will continue to grow exponentially. This opens up opportunities for more comprehensive and in-depth analyses, including personalised fitness recommendations, optimised training plans, and insights into individual health trends.

Machine learning and AI algorithms can be leveraged to predict training outcomes, identify patterns in performance, and offer real-time feedback for users. Integration with other health-related data, such as nutrition and sleep patterns, could provide a holistic view of overall wellness.

Furthermore, the data collected from a diverse user base can be aggregated and anonymized to conduct large-scale studies on fitness trends and correlations with health outcomes. These studies could contribute valuable insights to the fields of sports science and public health.

7. References

Dataset used for Analysis:

https://drive.google.com/uc?export=download&id=1O--TsE3O2orEDieV7tU2pp0ndMTYekQB

Project Code in Google Colab:

https://colab.research.google.com/drive/1MwEoocUsd__EL2g4TdP-fKegRfKfMTN7#scrollTo=l-mVzL-CTY2z