Final Report of Traineeship Program 2024
By Gayatri Bhinge

On

“Analysis of Fitness Data Project Proposal ”

MEDTOUREASY

28th Aug 2024

ACKNOWLDEGMENTS

The traineeship opportunity that I had with MedTourEasy was a great change for learning and understanding the intricacies of the subject of Data Visualizations in Data Analytics; and also, for personal as well as professional development. I am very obliged for having 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 & Developement Team of MedTourEasy who gave me an opportunity to carry out my traineeship at their esteemed organization. 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.

I would also like to thank the team of MedTourEasy and my colleagues who made the working environment productive and very conducive.

TABLE OF CONTENTS

ABSTRACT

With the surge in popularity of fitness trackers, runners worldwide are increasingly relying on devices such as smartphones and smartwatches to monitor and enhance their training routines. This project focuses on leveraging data collected from Runkeeper, a popular fitness tracking app, to gain insights into running activities. The data is provided in a CSV format, where each entry corresponds to a distinct training session.

Objectives:

•**Data Import and Cleaning:** Import the CSV file containing run data, clean the data to ensure accuracy and consistency, and prepare it for analysis.

•**Data Analysis:** Analyze the training data to address key questions including:

- How fast, long, and intense were the runs?
- Did the runner meet their training goals?
- What is the progress over time?
- What are the runner's best achievements?
- How does their performance compare to others?

The results will offer valuable insights into training performance and progress, with the potential to apply similar analytical strategies to other personal training data.

1. 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.

2. About the Project

In the contemporary world, the use of fitness trackers has become ubiquitous, with runners and athletes leveraging technology to monitor their performance and stay motivated. This project seeks to harness the data collected from Runkeeper, a leading fitness tracking application, to provide valuable insights into running activities. The dataset provided consists of a CSV file, where each row represents a unique training session recorded by the tracker.

The primary objectives of this project are to import, clean, and analyze the data to answer critical questions about the training activities. This involves several key steps:

1.Data Import and Cleaning: The first phase involves importing the CSV file into a suitable data analysis tool. Once imported, the data will be cleaned to rectify any inconsistencies, missing values, or errors. This ensures the dataset is accurate and reliable for subsequent analysis.

2.Exploratory Data Analysis (EDA): With the cleaned dataset, the next step is to perform exploratory data analysis. This includes examining various metrics such as run duration, distance covered, pace, and intensity. The analysis will help answer questions like how fast and how long the runs were, and whether the runner met their training goals.

3.Performance Evaluation: The project will also focus on evaluating progress over time and identifying significant achievements. By analyzing trends and patterns, the runner can gain insights into their overall performance and improvements.

4.Benchmarking: Another crucial aspect is comparing the runner's performance with others. This benchmarking will provide context on how the runner's achievements measure up against peers.

The findings from this project will offer actionable insights into training effectiveness and progress. Furthermore, the strategies and methods developed can be applied to personal training data for further analysis.

3. Objectives and Deliverables

Objectives:

1.Data Import and Preparation: Import the CSV file containing run data into a data analysis tool. Clean the dataset to address any issues such as missing values, inconsistencies, and erroneous entries. Ensure the data is in a format suitable for analysis.
2.Exploratory Data Analysis (EDA): Perform exploratory data analysis to understand the key characteristics of the dataset. This includes analyzing metrics such as distance, duration, pace, and intensity of each run. Identify patterns and trends to answer critical questions about the runner's performance.
3.Goal Achievement Assessment: Evaluate whether the runner has met their training goals by comparing recorded metrics against predefined targets. This involves assessing progress over time and identifying instances of goal attainment or shortfalls.
4.Performance Tracking: Analyze the runner's progress by examining changes in performance metrics over time. Identify improvements, plateaus, or declines in performance and provide insights into possible factors influencing these changes.
5.Achievement Highlights: Identify and highlight the runner's best achievements, including personal records and notable milestones. Provide a summary of significant accomplishments and improvements.
6.Comparative Analysis: Compare the runner's performance with aggregate data from other users to contextualize their achievements. This involves benchmarking the runner's metrics against those of peers to provide a comparative perspective.

Deliverables:

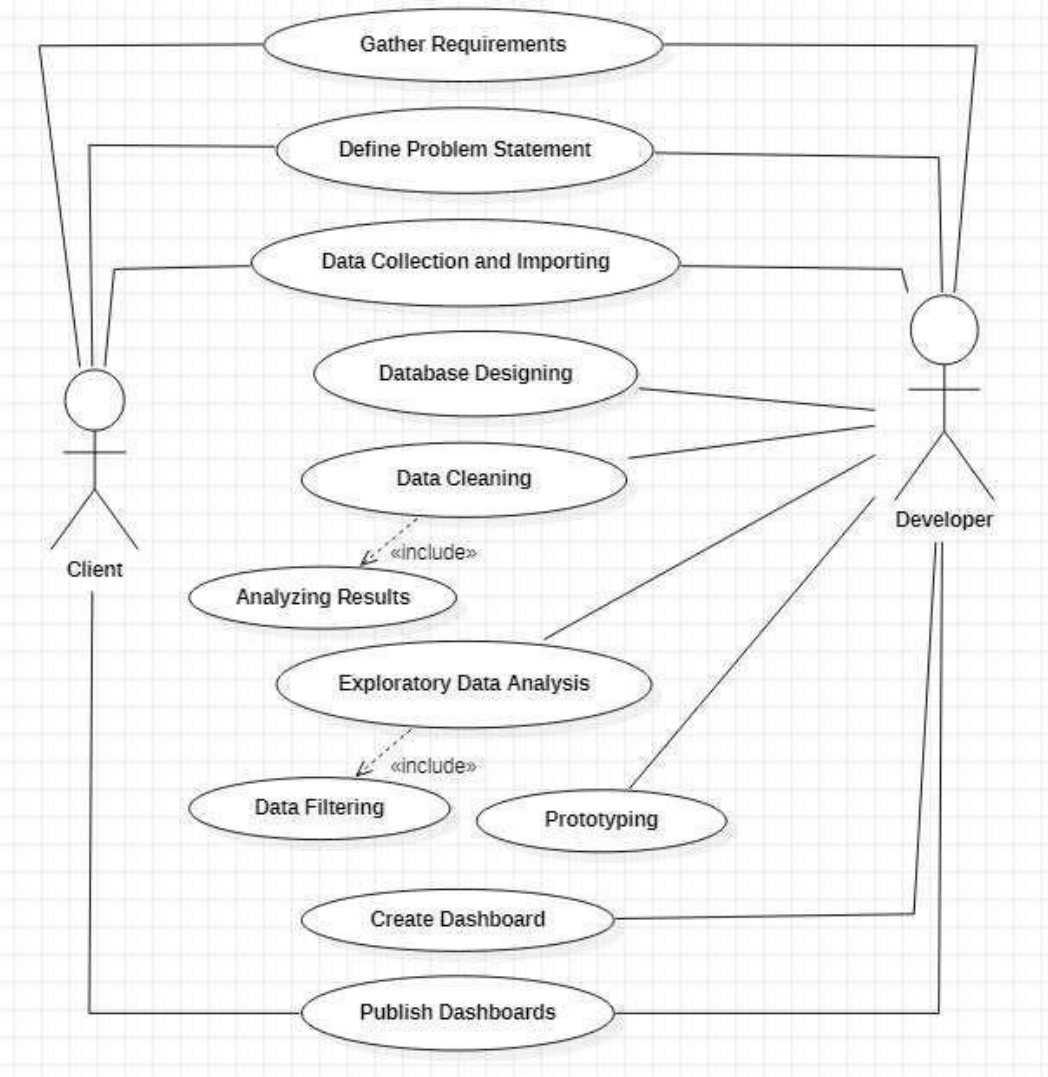
1.Cleaned Data Set: A well-structured and cleaned dataset ready for analysis, with any inconsistencies and errors resolved.
2.Exploratory Data Analysis Report: A comprehensive report detailing the key findings from the exploratory data analysis. This includes visualizations and summaries of metrics such as distance, duration, and pace.
3.Goal Achievement Summary: An assessment report indicating whether the runner has met their training goals, including a detailed analysis of goal attainment and any deviations.
4.Performance Trends Report: A report outlining trends in the runner's performance over time, highlighting areas of improvement or concern.
5.Achievement Highlights Document: A summary of the runner's notable achievements, including personal bests and significant milestones.
6.Comparative Analysis Report: A report providing insights into how the runner's performance compares with that of other users, offering context and benchmarks for their achievements.
These objectives and deliverables will provide a comprehensive understanding of the runner's performance and offer actionable insights to enhance their training regimen.

I. METHODOLOGY

2.1 Flow of the Project

The project followed the following steps to accomplish the desired objectives and deliverables. Each step has been explained in detail in the following section.

2.2 Use Case Diagram



Above figure shows the use case of the project. There are two main actors in the same: The Client and Developer. The developer will first gather requirements and define the problem statement then collecting the required data and importing it. Then the developer will design databases so as to identify various constraints and relations in the data. Next step is to clean the data to remove irregular values, blank values etc. Next, exploratory data analysis is conducted to filter the data according to the requirements of the project. Then a prototype of the dashboards is created using PowerBI to get a clear view of the visualizations to be developed. Finally, dashboard is developed and analyzed to publish the results to the client.

3. Language and Platform Used

Programming Language:

•Python: The primary programming language used for this project is Python. Python is widely recognized for its versatility and extensive support for data analysis and visualization. Its rich ecosystem of libraries and tools makes it an ideal choice for handling and analyzing large datasets.
Python Libraries:
1.pandas: This library is essential for data manipulation and analysis. It provides powerful data structures like DataFrames, which facilitate efficient data cleaning, transformation, and analysis. Pandas is used to import the CSV file, clean the data, and perform various data manipulation tasks.
2.matplotlib: Matplotlib is a plotting library used for creating static, interactive, and animated visualizations in Python. It is employed to generate graphs and charts that help in visualizing data trends, distributions, and performance metrics.
3.warnings: The warnings library is used to handle warning messages that may arise during data processing and analysis. It helps manage and suppress unnecessary warnings to ensure clean and clear output.
4.statsmodels: This library is used for statistical modeling and hypothesis testing. In this project, statsmodels is utilized for conducting statistical analyses and generating various statistical plots, such as Q-Q plots, to assess data distributions and model assumptions.
5.numpy: Numpy provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. It is used for numerical computations and to support other libraries like pandas and scikit-learn.
6.scikit-learn: Scikit-learn is a machine learning library that offers simple and efficient tools for data analysis and modeling. It is used for tasks such as data preprocessing, feature selection, and performance evaluation through various machine learning algorithms and metrics.

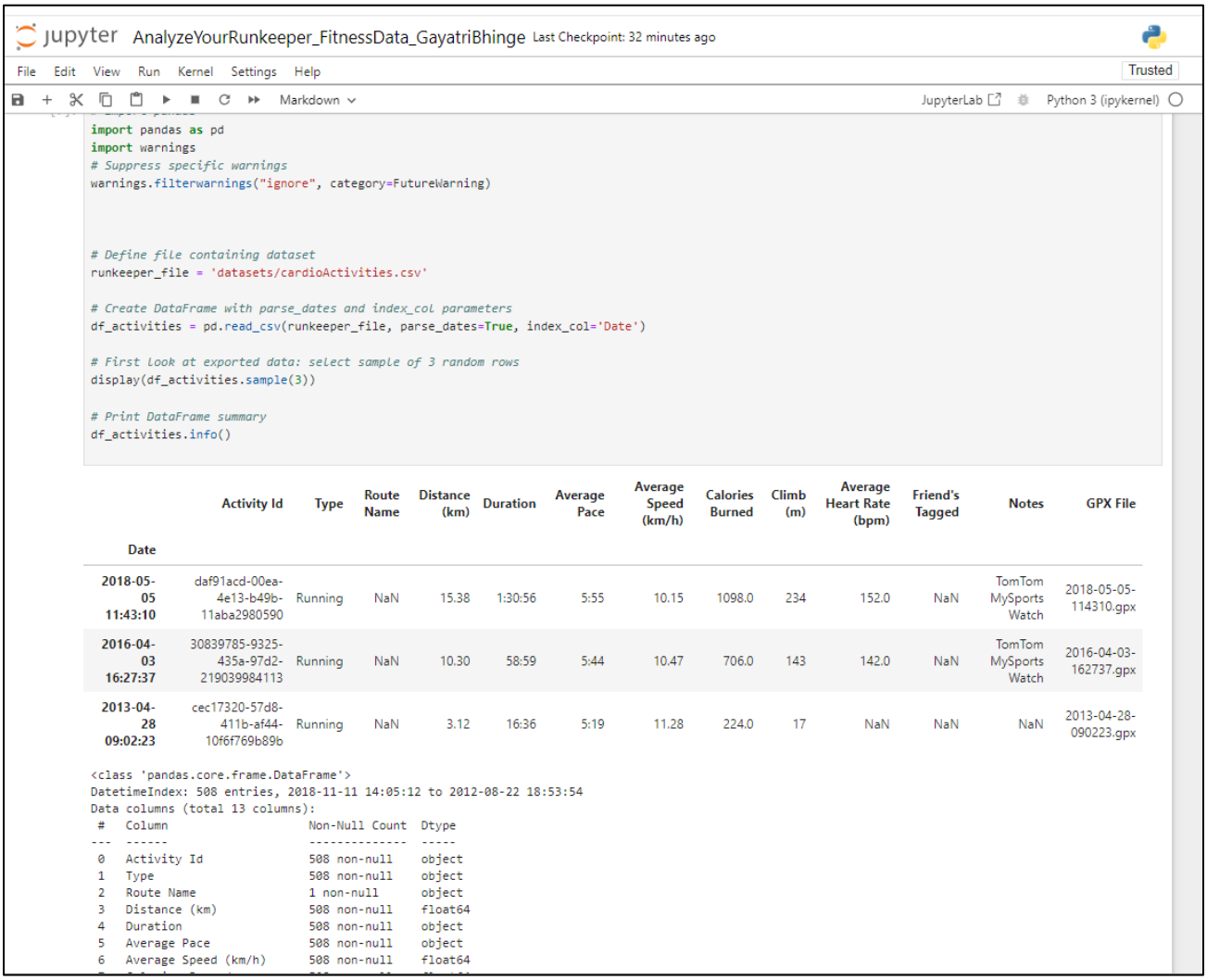
Platform:
•Jupyter Notebook/IDE: The project is developed using a Jupyter Notebook or an Integrated Development Environment (IDE) such as PyCharm or VS Code. These platforms provide an interactive environment for writing and executing Python code, visualizing data, and documenting the analysis process.

Dataset:
•CSV File: The dataset used in this project is a CSV (Comma-Separated Values) file. CSV files are a popular format for storing tabular data, where each row represents a single record, and each column represents a variable. The CSV format is easily readable and compatible with many data analysis tools and libraries.

One day, my old running friend and I were chatting about our running styles, training habits, and achievements, when I suddenly realized that I could take an in-depth analytical look at my training. I have been using a popular GPS fitness tracker called Runkeeper for years and decided it was time to analyze my running data to see how I was doing.

Since 2012, I've been using the Runkeeper app, and it's great. One key feature: its excellent data export. Anyone who has a smartphone can download the app and analyze their data like we will in this notebook.
After logging your run, the first step is to export the data from Runkeeper (which I've done already). Then import the data and start exploring to find potential problems. After that, create data cleaning strategies to fix the issues. Finally, analyze and visualize the clean time-series data.

I exported seven years worth of my training data, from 2012 through 2018. The data is a CSV file where each row is a single training activity. Let's load and inspect it.



Date	Activity Id	Type	Route Name	Distance (km)	Duration	Average Pace	Average Speed (km/h)	Calories Burned	Climb (m)	Average Heart Rate (bpm)	Friend's Tagged	Notes	GPX File
2018-05-05 11:43:10	daf91acd-00ea-4e13-b49b-11aba2980590	Running	NaN	15.38	1:30:56	5:55	10.15	1098.0	234	152.0	NaN	TomTom MySports Watch	2018-05-05-114310.gpx
2016-04-03 16:27:37	30839785-9325-435a-97d2-219039984113	Running	NaN	10.30	58:59	5:44	10.47	706.0	143	142.0	NaN	TomTom MySports Watch	2016-04-03-162737.gpx
2013-04-28 09:02:23	cec17320-57d8-411b-af44-10f6f769b89b	Running	NaN	3.12	16:36	5:19	11.28	224.0	17	NaN	NaN	NaN	2013-04-28-090223.gpx

```
<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
```

II. IMPLEMENTATION

1. Gathering Requirements and Defining Problem Statement

This is the first step wherein the requirements are collected from the clients to understand the deliverables and goals to be achieved after which a problem statement is defined which has to be adhered to while development of the project.

2. Data Collection and Importing

Data collection is a systematic approach for gathering and measuring information from a variety of sources in order to obtain a complete and accurate picture of an interest area. It helps an individual or organization to address specific questions, determine outcomes and forecast future probabilities and patterns.

With the explosion in fitness tracker popularity, runners all over the world are collecting data using gadgets such as smartphones, smartwatches, and other wearable devices to keep themselves motivated. They seek answers to important 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 has been exported from Runkeeper, a popular fitness tracking app. The data is stored in a CSV file, where each row represents a single training activity.

Functions Used:

```
import pandas as pd

# Load the dataset from a CSV

data = pd.read_csv(file_path)
```

This function we used to load a dataset stored in a CSV file into a pandas DataFrame using the read_csv() function. The file_path variable holds the path to the CSV file. After loading the data into the data DataFrame, the head() function is used to display the first few rows, verifying that the data has been successfully imported and is ready for analysis.

```
# Import pandas
import pandas as pd
import warnings
# Suppress specific warnings
warnings.filterwarnings("ignore", category=FutureWarning)



# Define file containing dataset
runkeeper_file = 'datasets/cardioActivities.csv'

# Create DataFrame with parse_dates and index_col parameters
df_activities = pd.read_csv(runkeeper_file, parse_dates=True, index_col='Date')

# First look at exported data: select sample of 3 random rows
display(df_activities.sample(3))

# Print DataFrame summary
df_activities.info()
```

Date	Activity Id	Type	Route Name	Distance (km)	Duration	Average Pace	Average Speed (km/h)	Calories Burned	Climb (m)	Average Heart Rate (bpm)	Friend's Tagged	Notes	GPX File
2018-05-05 11:43:10	daf91acd-00ea-4e13-b49b-11aba2980590	Running	NaN	15.38	1:30:56	5:55	10.15	1098.0	234	152.0	NaN	TomTom MySports Watch	2018-05-05-114310.gpx
2016-04-03 16:27:37	30839785-9325-435a-97d2-219039984113	Running	NaN	10.30	58:59	5:44	10.47	706.0	143	142.0	NaN	TomTom MySports Watch	2016-04-03-162737.gpx
2013-04-28 09:02:23	cec17320-57d8-411b-af44-10f6f769b89b	Running	NaN	3.12	16:36	5:19	11.28	224.0	17	NaN	NaN	NaN	2013-04-28-090223.gpx

```
<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
```

3.1 Steps used in this projects

"Quality data beats fancy algorithms"

Data is the most imperative aspect of Analytics and Machine Learning. Everywhere in computing or business, data is required. But many a times, the data may be incomplete, inconsistent or may contain missing values when it comes to the real world. If the data is corrupted then the process may be impeded or inaccurate results may be provided. Hence, Data cleaning is considered a foundational element of the basic data science.

Data Cleaning means the process by which the incorrect, incomplete, inaccurate, irrelevant or missing part of the data is identified and then modified, replaced or deleted as needed.

Here are some common functions used in the data cleaning process with Python and its libraries:
1.dropna(): Removes rows or columns with missing values (NaN) from a DataFrame.
2.fillna(): Replaces missing values in the DataFrame with a specified value or method (e.g., forward fill).
3.astype(): Converts the data type of a DataFrame column to a specified type (e.g., integer, float, string).
4.replace(): Substitutes specific values in the DataFrame with new ones, useful for correcting data entry errors.
5.duplicated(): Identifies duplicate rows in the DataFrame based on specified columns or the entire row.
6.drop_duplicates(): Removes duplicate rows from the DataFrame, keeping only the first occurrence by default.
7.str.strip(): Removes leading and trailing whitespace from string data in a DataFrame column.
8.apply(): Applies a custom function to each element or row in a DataFrame column for more complex cleaning tasks.
9.rename(): Renames DataFrame columns to improve clarity or correct mislabeling.
10.replace(): Replaces specific values in a DataFrame with others, often used for correcting categorical data or normalizing text.
These functions are commonly used in the data cleaning process to prepare datasets for analysis, ensuring that the data is consistent, complete, and properly formatted.

.3.5 Steps used in this projects

Step 1 Load pandas and the training activities data:

1. Import pandas under the alias pd.
    1. Use the read_csv() function to load the dataset (runkeeper_file) into a variable called df_activities.Parse the dates with the parse_dates parameter and set the index to the Date column using the index_col parameter.
2. Display data:
    1. Display 3 random rows from df_activities using the sample() method.
    2. Print a summary of df_activities using the info() method.

Step 2 Implement the following data preprocessing tasks:

1. Delete unnecessary columns:
    1. Use the drop() method on df_activities, setting the columns parameter to the cols_to_drop list.
2. Calculate activity type counts:
    1. Use the value_counts() method on the Type column to calculate the counts of each activity type.
3. Rename 'Other' to 'Unicycling':
    1. Use the str.replace() method on the Type column to rename all 'Other' values to 'Unicycling'.
4. Count missing values:
    1. Count the missing values in each column using the isnull().sum() method.

.3.5 Steps used in this projects

Step 3 Implement mean imputation for missing values:

1. Calculate the sample mean for 'Cycling' Average Heart Rate:
    1. Calculate the sample mean for the Average Heart Rate (bpm) for the 'Cycling' activity type.
    2. Assign the result to avg_hr_cycle.
2. Filter and copy the 'Cycling' activity type data:
    1. Filter df_activities for the 'Cycling' activity type.
    2. Create a copy of the filtered data using the copy() method and assign it to df_cycle.
3. Fill in missing values for Average Heart Rate (bpm):
    1. Fill in the missing values for Average Heart Rate (bpm) in df_cycle with int(avg_hr_cycle) using the fillna() method.
4. Count missing values:
    1. Count the missing values for all columns in df_run.

Step 4 Plot running data from 2013 through 2018:

1. Subset df_run for data from 2013 through 2018:
    1. Subset df_run to include only data from 2013 through 2018.
    2. Remember that the dataset is stored in chronological order with the most recent records first.
    3. Assign the result to runs_subset_2013_2018.
2. Enable subplots in the plotting code:
    1. Enable subplots by setting the subplots parameter to True in the plotting code.
    2. Follow PEP 8 guidelines by not using spaces around the = sign when indicating a keyword argument.
3. Show the plot:
    1. Use plt.show() to display the plot.

.3.5 Steps used in this projects

Step 5 Prepare data and create a plot:

1. Calculate annual and weekly means:
    1. Subset df_run for data from 2015 through 2018:
    1. Subset df_run to include only data from 2015 through 2018.
    2. Assign the result to runs_subset_2015_2018.
2. Count annual averages:
    1. Use resample('A') to aggregate by year and calculate the mean for Distance (km), Average Speed (km/h), Climb (m), and Average Heart Rate (bpm).
3. Count weekly averages:
    1. Use resample('W') to aggregate by week and calculate the mean.
    2. Apply mean() twice if necessary to get the average weekly statistics.
4. Count average number of trainings per week:
    1. Filter the Distance (km) column and use resample('W') to count the number of trainings per week.
    2. Calculate the average number of trainings per week using mean().
    3. Assign the result to weekly_counts_average.

.3.5 Steps used in this projects

Step 6 Prepare data and create a plot:

1. Select Information:
    1. Extract the distance data and assign it to runs_distance.
    2. Extract the heart rate data and assign it to runs_hr.
2. Create Subplots:
    1. Use the plt.subplots() method to create two subplots with a shared x-axis.
    2. Set the first positional parameter to 2 (for the number of subplots).
    3. Set sharex to True to share the x-axis.
    4. Set figsize to (12, 8).
    5. Assign the output to fig, (ax1, ax2).
3. Plot Data:
    1. On the first subplot (ax1), plot the distance data.
4. Add Horizontal Line:
    1. On the second subplot (ax2), add a horizontal line using axhline() for the average heart rate value (runs_hr.mean()).
    2. Set the line color to 'blue', the linewidth to 1, and the linestyle to '-.'.

Step 7 Prepare data and create a plot:

1. Subset Data:
    1. Filter df_run for data from 2013 through 2018.
    2. Select the Distance (km) column.
    3. Count annual totals using resample() and sum().
    4. Assign the result to df_run_dist_annual.
2. Create Plot:
    1. Use plt.figure() to create a figure, setting figsize to (8.0, 5.0).
3. Customize Plot:
    1. Use ax.axhspan() to add a horizontal span from 0 to 800 km.
    2. Set the color to 'red' and alpha to 0.2.
4. Show Plot:
    1. Display the plot with plt.show().

.3.5 Steps used in this projects

Step 8 Create a plot with observed distance of runs and decomposed trend

1. Import statsmodels.api under the alias sm.
2. Subset df_run from 2013 through 2018, select the Distance (km) column, resample the data to a weekly frequency, and fill any NaN values using the bfill() method. Assign this result to df_run_dist_wkly.
3. Create a plot using plt.figure(), setting the figure size to (12, 5).

Step 9 Create a customized histogram for heart rate distribution

1. Subset df_run from March 2015 through 2018, then select the Average Heart Rate (bpm) column. Assign the result to df_run_hr_all.
2. Create a plot using plt.subplots(), setting the figure size to (8, 5), and assign the result to fig, ax.
3. Customize the x-axis ticks with ax.set_xticklabels(). Set the labels to zone_names, rotation to -30, and horizontal alignment (ha) to 'left'.
4. Show the plot with plt.show().

Step 10 Create a summary report:

1. Concatenate df_run, df_walk, and df_cycle using the append() method and then sort the resulting DataFrame based on the index in descending order. Assign this result to df_run_walk_cycle.
2. Group df_run_walk_cycle by the activity type and select the columns specified in dist_climb_cols. Sum the grouped data and assign the result to df_totals.
3. Use the stack() method on df_summary to create a compact, reshaped form of the full summary report

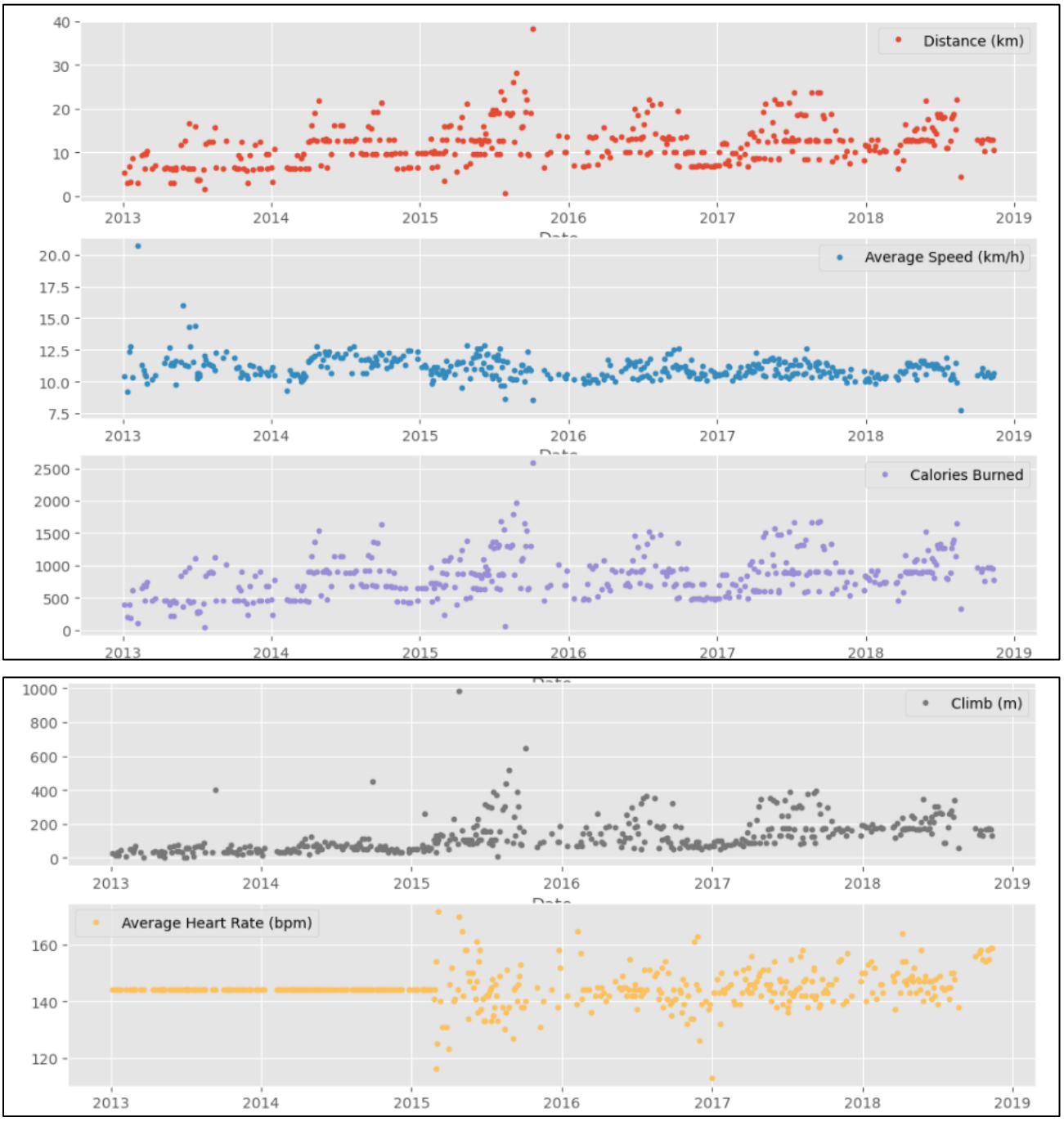
Step 10 Use FUN FACTS data to answer some fun questions:

1. To calculate the instructor's average shoes per lifetime, divide the total number of km run (from FUN FACTS) by the number of pairs of shoes gone through.
2. To estimate the number of shoes Forrest Gump would use for his route, divide the total number of km run (from FORREST RUN FACTS) by the result from the previous step (using floor division).

Plot running data

We can create our first plot! As we found earlier, most of the activities in my data were running (459 of them to be exact). There are only 29, 18, and two instances for cycling, walking, and unicycling, respectively. So for now, let's focus on plotting the different running metrics.

An excellent first visualization is a figure with four subplots, one for each running metric (each numerical column). Each subplot will have a different y-axis, which is explained in each legend. The x-axis, Date, is shared among all subplots.



Running statistics

No doubt, running helps people stay mentally and physically healthy and productive at any age. And it is great fun! When runners talk to each other about their hobby, we not only discuss our results, but we also discuss different training strategies.
You'll know you're with a group of runners if you commonly hear questions like:
What is your average distance?
How fast do you run?
Do you measure your heart rate?
How often do you train?
Let's find the answers to these questions in my data. If you look back at plots in Task 4, you can see the answer to, Do you measure your heart rate? Before 2015: no. To look at the averages, let's only use the data from 2015 through 2018.
In pandas, the resample() method is similar to the groupby() method - with resample() you group by a specific time span. We'll use resample() to group the time series data by a sampling period and apply several methods to each sampling period. In our case, we'll resample annually and weekly.

```
How my average run looks in last 4 years:
```

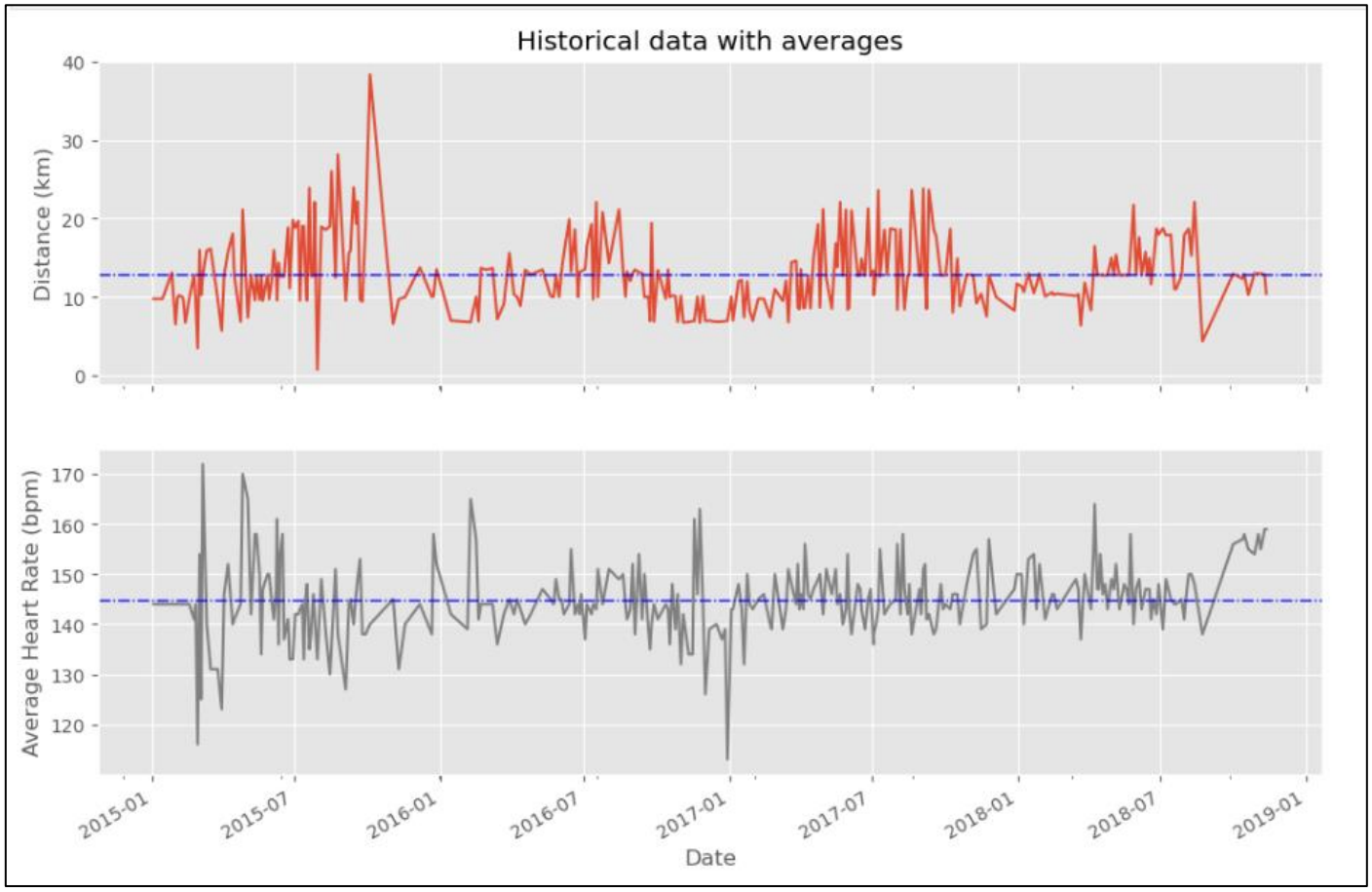
Date	Distance (km)	Average Speed (km/h)	Calories Burned	Climb (m)	Average Heart Rate (bpm)	Friend's Tagged
2015-12-31	13.602805	10.998902	932.906138	160.170732	143.353659	NaN
2016-12-31	11.411667	10.837778	796.152777	133.194444	143.388889	NaN
2017-12-31	12.935176	10.959059	914.164706	169.376471	145.247059	NaN
2018-12-31	13.339063	10.777969	952.359375	191.218750	148.125000	NaN

```
Weekly averages of last 4 years:
Distance (km)                12.518176
Average Speed (km/h)         10.835473
Calories Burned             877.861969
Climb (m)                   158.325444
Average Heart Rate (bpm)    144.801775
Friend's Tagged                    NaN
dtype: float64
How many trainings per week I had on average: 1.5
```

Visualization with averages

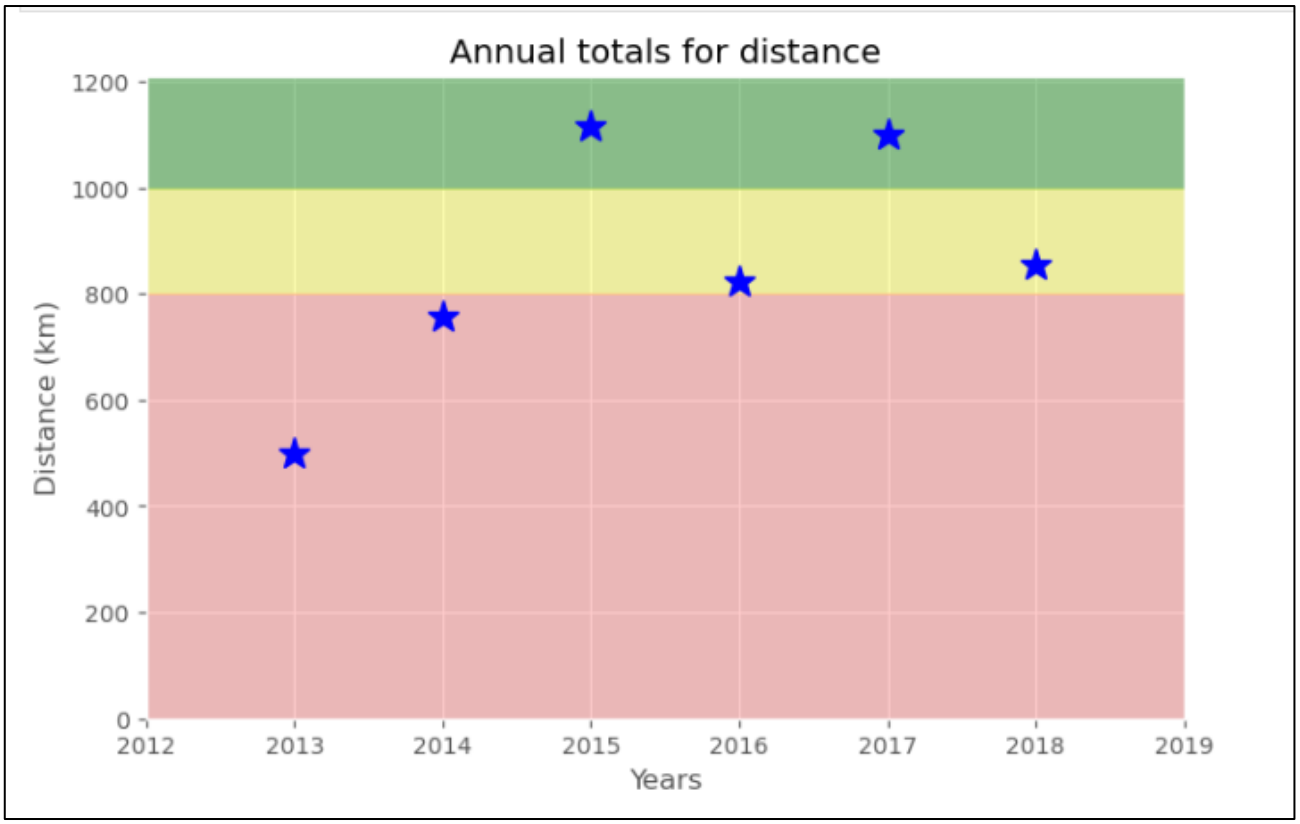
We plot the long term averages of my distance run and my heart rate with their raw data to visually compare the averages to each training session. Again, we'll use the data from 2015 through 2018.

In this task, we will use matplotlib functionality for plot creation and customization.



Did I reach my goals?

To motivate myself to run regularly, I set a target goal of running 1000 km per year. Let's visualize my annual running distance (km) from 2013 through 2018 to see if I reached my goal each year. Only stars in the green region indicate success.

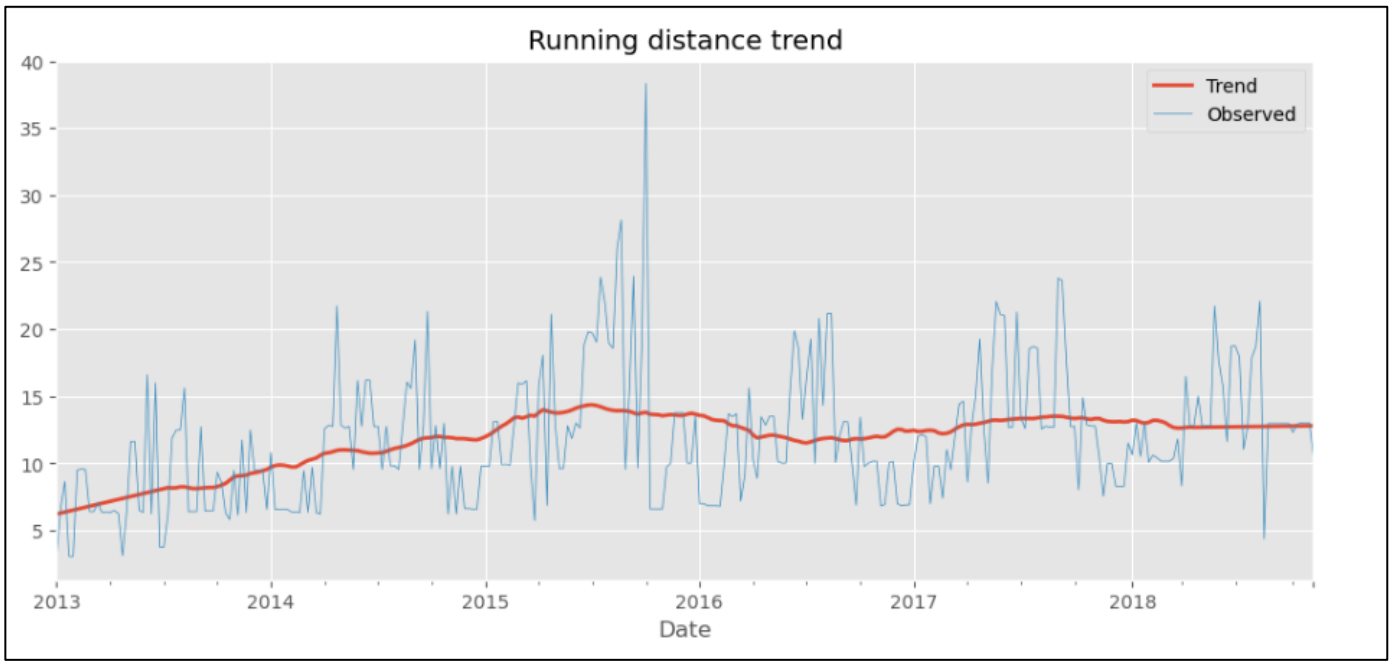


Am I progressing?

Let's dive a little deeper into the data to answer a tricky question: am I progressing in terms of my running skills?

To answer this question, we'll decompose my weekly distance run and visually compare it to the raw data. A red trend line will represent the weekly distance run.

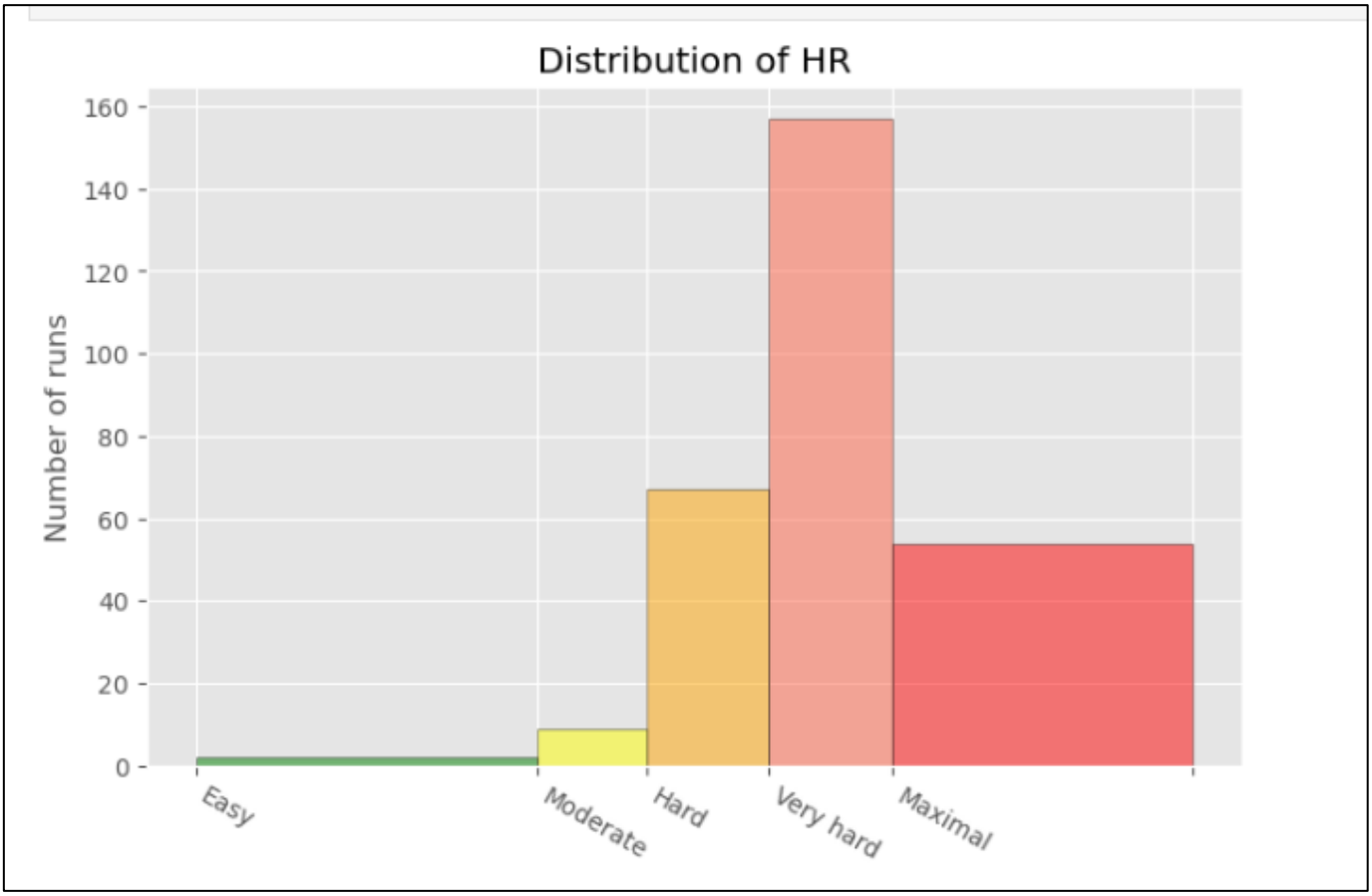
We are going to use statsmodels library to decompose the weekly trend.



Training intensity

Heart rate is a popular metric used to measure training intensity. Depending on age and fitness level, heart rates are grouped into different zones that people can target depending on training goals. A target heart rate during moderate-intensity activities is about 50-70% of maximum heart rate, while during vigorous physical activity it's about 70-85% of maximum.

We'll create a distribution plot of my heart rate data by training intensity. It will be a visual presentation for the number of activities from predefined training zones.



Detailed summary report

With all this data cleaning, analysis, and visualization, let's create detailed summary tables of my training.

To do this, we'll create two tables. The first table will be a summary of the distance (km) and climb (m) variables for each training activity. The second table will list the summary statistics for the average speed (km/hr), climb (m), and distance (km) variables for each training activity.

Type	Distance (km)	Climb (m)
Cycling	140	1400
Running	75	1500
Walking	6	60

Summary statistics for different training types:

57]:

Type		Distance (km)	Climb (m)	Average Speed (km/h)
Cycling	25%	27.500000	275.000000	15.750000
	50%	35.000000	350.000000	16.500000
	75%	42.500000	425.000000	17.250000
	count	4.000000	4.000000	4.000000
	max	50.000000	500.000000	18.000000
	mean	35.000000	350.000000	16.500000
	min	20.000000	200.000000	15.000000
	std	12.909944	129.099445	1.290994
	total	140.000000	1400.000000	NaN
Running	25%	10.000000	200.000000	12.000000
	50%	15.000000	300.000000	14.000000
	75%	20.000000	400.000000	16.000000
	count	5.000000	5.000000	5.000000
	max	25.000000	500.000000	18.000000
	mean	15.000000	300.000000	14.000000
	min	5.000000	100.000000	10.000000
	std	7.905694	158.113883	3.162278
	total	75.000000	1500.000000	NaN
Walking	25%	1.500000	15.000000	5.500000
	50%	2.000000	20.000000	6.000000
	75%	2.500000	25.000000	6.500000
	count	3.000000	3.000000	3.000000
	max	3.000000	30.000000	7.000000
	mean	2.000000	20.000000	6.000000
	min	1.000000	10.000000	5.000000
	std	1.000000	10.000000	1.000000
	total	6.000000	60.000000	NaN

Training intensity

To wrap up, let's pick some fun facts out of the summary tables and solve the last exercise.

These data (my running history) represent 6 years, 2 months, and 21 days. And I remember how many running shoes I went through–7.

FUN FACTS

- **Average distance:** 11.38 km
- **Longest distance:** 38.32 km
- **Highest climb:** 982 m
- **Total climb:** 57,278 m
- **Total number of km run:** 5,224 km
- **Total runs:** 459
- **Number of running shoes gone through:** 7 pairs

The story of Forrest Gump is well known–the man, who for no particular reason decided to go for a "little run." His epic run duration was 3 years, 2 months, and 14 days (1169 days). In the picture, you can see Forrest's route of 24,700 km.

FORREST RUN FACTS

- **Average distance:** 21.13 km
- **Total number of km run:** 24,700 km
- **Total runs:** 1169
- **Number of running shoes gone through:** ...

Assuming Forrest and I go through running shoes at the same rate, figure out how many pairs of shoes Forrest needed for his run.

CONCLUSION

This analysis of fitness data from the app has provided valuable insights into running habits and performance trends over the years, specifically from 2015 to 2018. By focusing on key metrics such as distance, speed, heart rate, and training frequency, we have been able to:

Visualize Progress:

The analysis demonstrated that running is not just a means of staying fit but also a methodical process where progress can be tracked and optimized over time. The annual running distances, particularly, helped identify whether yearly goals were met, indicating consistency and dedication.

Identify Trends:

The decomposition of weekly running distances revealed underlying trends, seasonality, and patterns in the raw data. This understanding can aid in making informed decisions about training plans and adjustments to meet long-term goals.

Assess Training Intensity:

By categorizing heart rate data into training zones, we could visualize the intensity of various running sessions. This information is crucial for optimizing training plans according to fitness goals, whether for endurance, speed, or recovery.

Overall, this analysis highlights the importance of regular monitoring and visualization in understanding and improving running performance. It also emphasizes how structured data analysis can provide actionable insights to enhance training effectiveness.

FUTURE SCOPE

While the current analysis has yielded valuable insights, there are several avenues for further exploration:

Incorporating More Data:

Expanding the dataset to include other activities such as cycling, walking, and unicycling could offer a more comprehensive view of overall fitness. Comparing and contrasting different types of training could reveal cross-training benefits and potential areas for improvement.

Exploring External Factors:

Analyzing the impact of external factors such as weather, terrain, and time of day on running performance could provide deeper insights into optimizing training conditions.

Machine Learning Models:

Implementing machine learning models to predict future performance based on past data could help set realistic goals and adjust training plans dynamically. For example, regression models could predict future distances or times, while clustering algorithms might identify distinct training patterns.

Integrating Other Health Metrics:

Including other health-related data such as sleep patterns, nutrition, and recovery times could help create a holistic view of the athlete's condition. This would enable more personalized training recommendations.

Real-Time Analytics:

Developing real-time analytics tools that provide immediate feedback on running sessions could enhance the training experience and allow for on-the-fly adjustments, leading to more effective workouts.

By addressing these areas, future analyses could not only deepen the understanding of individual performance but also contribute to broader applications in fitness and health management.

V. REFERENCES

Data Collection

The following provided by MedTourEasy:

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

Programming References

The following websites have been referred for Python coding:

a. https://www.coursera.org/professional-certificates/google-data-analytics