Final Report of Traineeship Program 2024

By MedTourEasy

On

Fitness Analysis

28th June 2024

Name: Siva Arun Kumar P

Mail: siva.arun.k08@gmail.com

ACKNOWLDEGMENTS

I want to extend my thanks to the team of MedTourEasy and my colleagues who made the working environment very productive.

It is an opportunity just great at MedTourEasy for learning and understanding the minutiae of the subject on Data Visualizations in the domain of data analytics, and for personal and professional development. Really obliged, I am to get a chance to interact with so many professionals who guided me throughout the traineeship project and gave me a great learning curve.

First of all, I am highly indebted and extend special thanks to the Training & Development Team of MedTourEasy for giving me an opportunity to carry out my traineeship at their esteemed organization. I am also thankful to the team, who made me understand the minute details of the Data Analytics profile, trained me in the same so that I can carry out the project properly and with maximum client satisfaction, and for spearing his valuable time in spite of his busy scheduled routine.

I will not forget to extend my appreciation to the team at MedTourEasy and colleagues who made the working environment very productive and conducive.

TABLE OF CONTENTS

ABSTRACT

This project involves the analysis of a fitness dataset collected from RunKeeper—a popular fitness tracking application. This is data about several types of training activities: running, cycling, or even unicycling. The goal of the project is to import user data and cleanse it for analysis regarding questions to their training progress, achievements, and performance compared with others.

In data preprocessing, visualization, and statistical analysis, this project includes 11 tasks: mean imputation, plotting running data, computation of annual and weekly means, and plot creation to understand relationships between variables. This project is powered by Python, along with its popular libraries Pandas and Matplotlib, to import, clean, and analyze data.

Such results would provide insight into the fitness journey of users, show where improvements should really be made, and compare with others. Results like this on a different fitness dataset are also of importance not only to the user but also to the professional level.

INTRODUCTION

Interest in fitness tracking technologies has coincided with a hike in data collection amongst runners. However, it is hard to gain benefit from such data without proper analysis. In this project, the author has tried to do so using Python and its avid libraries, Pandas and Matplotlib, to import, clean, and analyze data. In this approach for the project, the analysis ranges from the 11 tasks targeting the narrow aspects down of the data. This will allow an in-depth analysis of the data to be derived and a minute detailing of the fitness journey of the user.

Communities of runners have grown, hence data collected by such devices is at a large amount. This data is used to monitor progress, keeping the individual motivated and learning many other training habits. Analyzing the data, however, may prove to be cumbersome. The current project tries to do this with the help of Python and its popular libraries dedicated to data import, cleaning, and analysis.

The project therefore provides a full overview of the obtained data by breaking down the analysis into 11 tasks, which give a picture regarding a user's fitness journey. It includes visualization and statistical processing, among other tasks involved in preprocessing the data, all contributing towards clearly understanding the progress and achievements of a user in training.

IMPLEMENTATION

Task 1: Importing and Understanding Data

In this exercise, we will read in data from Training Activities into a Pandas DataFrame. Further, we have viewed 3 random rows from the DataFrame to understand what data it holds. Then we print a summary of the dataframe (1).

```
# Import pandas
# ... YOUR CODE FOR TASK 1 ...
import pandas as pd
# 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=['Date'],
                            index_col='Date')

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

# Print DataFrame summary
df_activities.info()
# ... YOUR CODE FOR TASK 1 ...
```

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
2018-11-11 14:05:12	c9627fed-14ac-47a2-bed3-2a2630c63c15	Running	NaN	10.44	58:40	5:37	10.68	774.0	130	159.0	NaN	NaN	2018-11-11-140512.gpx
2018-11-09 15:02:35	be65818d-a801-4847-a43b-2acdf4dc70e7	Running	NaN	12.84	1:14:12	5:47	10.39	954.0	168	159.0	NaN	NaN	2018-11-09-150235.gpx
2018-11-04 16:05:00	c09b2f92-f855-497c-b624-c196b3ef036c	Running	NaN	13.01	1:15:16	5:47	10.37	967.0	171	155.0	NaN	NaN	2018-11-04-160500.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
```

1. Summary of the dataframe

Task 2: Data Preprocessing

The dimensionality of the data was reduced by dropping columns considered irrelevant in the DataFrame. The count across activity type variable was calculated to determine the most common types of activities. We renamed 'Other' values as 'Unicycling' for cohesiveness of the data. Finally, the count of missing values in each column (2) was done to indicate areas where data imputation may be needed. First, why this works, it understands what data it holds. Then we will print the summary of the dataframe.

```
# Define list of columns to be deleted
cols_to_drop = ['Friend\'s Tagged','Route Name','GPX File','Activity Id','Calories Burned', 'Notes']

# Delete unnecessary columns
# ... YOUR CODE FOR TASK 2 ...
df_activities.drop(columns=cols_to_drop,inplace=True)
# Count types of training activities
display(df_activities['Type'].value_counts())

# Rename 'Other' type to 'Unicycling'
df_activities['Type'] = df_activities['Type'].replace('Other', 'Unicycling')

# Count missing values for each column
print(df_activities.isnull().sum())
# ... YOUR CODE FOR TASK 2 ...
```

```
Type
Running    459
Cycling     29
Walking     18
Other        2
Name: count, dtype: int64

Type                          0
Distance (km)                 0
Duration                      0
Average Pace                  0
Average Speed (km/h)          0
Climb (m)                     0
Average Heart Rate (bpm)    214
dtype: int64
```

2. The count of missing values in each column

Task 3: Mean Imputation

In this exercise, we implemented mean imputation for missing values. We computed the sample mean of the feature 'Average Heart Rate (bpm)' for the 'Cycling' type of activity, after which we filled in the missing values with this mean. using this, we eliminated missing values and replaced with a possible estimator, reducing their missing data (3) impact on our fitness analysis

```
# Calculate sample means for heart rate for each training activity type
avg_hr_run = df_activities[df_activities['Type'] == 'Running']['Average Heart Rate (bpm)'].mean()
avg_hr_cycle = df_activities[df_activities['Type'] == 'Cycling']['Average Heart Rate (bpm)'].mean()
avg_hr_walk=df_activities[df_activities['Type'] == 'Walking']['Average Heart Rate (bpm)'].mean()
avg_hr_unicycle=df_activities[df_activities['Type'] == 'Unicycling']['Average Heart Rate (bpm)'].mean()
# Split whole DataFrame into several, specific for different activities
df_run = df_activities[df_activities['Type'] == 'Running'].copy()
df_walk = df_activities[df_activities['Type'] == 'Walking'].copy()
df_cycle = df_activities[df_activities['Type'] == 'Cycling'].copy()
df_unicycle=df_activities[df_activities['Type'] == 'Unicycling'].copy()

# Filling missing values with counted means
df_walk['Average Heart Rate (bpm)'].fillna(110, inplace=True)
df_run['Average Heart Rate (bpm)'].fillna(int(avg_hr_run), inplace=True)
df_cycle['Average Heart Rate (bpm)'].fillna(int(avg_hr_cycle), inplace=True)
df_unicycle['Average Heart Rate (bpm)'].fillna(int(avg_hr_unicycle), inplace=True)
# ... YOUR CODE FOR TASK 3 ...

# Count missing values for each column in running data
print(df_walk.isnull().sum())
print(df_run.isnull().sum())
print(df_cycle.isnull().sum())
print(df_unicycle.isnull().sum())
# ... YOUR CODE FOR TASK 3 ...
```

```
Type                    0
Distance (km)           0
Duration                0
Average Pace            0
Average Speed (km/h)    0
Climb (m)               0
Average Heart Rate (bpm)    0
dtype: int64
Type                    0
Distance (km)           0
Duration                0
Average Pace            0
Average Speed (km/h)    0
Climb (m)               0
Average Heart Rate (bpm)    0
dtype: int64
Type                    0
Distance (km)           0
Duration                0
Average Pace            0
Average Speed (km/h)    0
Climb (m)               0
Average Heart Rate (bpm)    0
dtype: int64
Type                    0
Distance (km)           0
Duration                0
Average Pace            0
Average Speed (km/h)    0
Climb (m)               0
```

3. Reducing their missing data

Task 4: Running Plotting Data

In this exercise, running data has been plotted from 2013 to 2018. This plot expresses how far the user has come in running over the years (4). We subset the data for a running activity within this period and generated a plot of distance covered against date. This plot gives a sense of progress of running by the user and allows identification of trends and patterns.

4. How far the user has come in running over the year(2013-2018)

Task 5: Compute means, annual and weekly.

We have computed annual and weekly means for the variables Distance, Average Speed, Climb, and Average Heart Rate in this exercise. We used the resample() function to compute the means at different frequencies (5) to understand the user's training progress in a minute way.

How my average run looks in last 4 years:

Date	Distance (km)	Average Speed (km/h)	Climb (m)	Average Heart Rate (bpm)
2015-12-31	14.522826	11.781413	163.467391	142.704225
2016-12-31	12.526582	11.697342	148.518987	142.289855
2017-12-31	13.458977	11.308636	171.954545	145.070588
2018-12-31	14.434648	11.365211	203.112676	143.724638

Weekly averages of last 4 years:

Date	Distance (km)	Average Speed (km/h)	Climb (m)	Average Heart Rate (bpm)
2015-01-04	9.780000	11.120000	51.0	NaN
2015-01-11	NaN	NaN	NaN	NaN
2015-01-18	9.780000	11.230000	51.0	NaN
2015-01-25	NaN	NaN	NaN	NaN
2015-02-01	9.893333	10.423333	58.0	NaN
...	...	...	...	...
2018-10-14	12.620000	10.840000	146.5	157.5
2018-10-21	10.290000	10.410000	133.0	155.0
2018-10-28	13.020000	10.730000	170.0	154.0
2018-11-04	12.995000	10.420000	170.0	156.5
2018-11-11	11.640000	10.535000	149.0	159.0

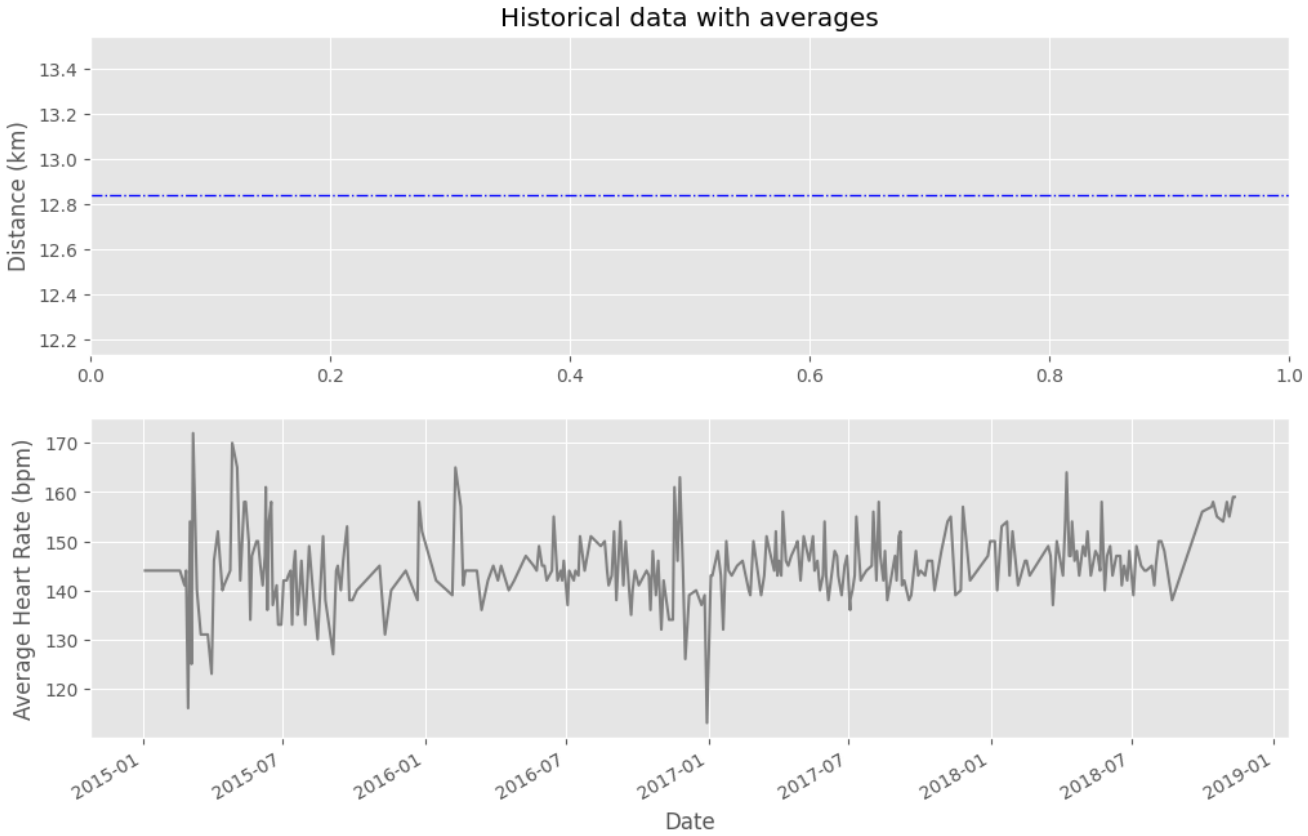
202 rows × 4 columns

```
How many trainings per week I had on average: Distance (km)            1.633663
Average Speed (km/h)        1.633663
Climb (m)                   1.633663
Average Heart Rate (bpm)    1.455446
dtype: float64
```

5. Means at different frequencies

Task 6: Data Preparation and Plotting

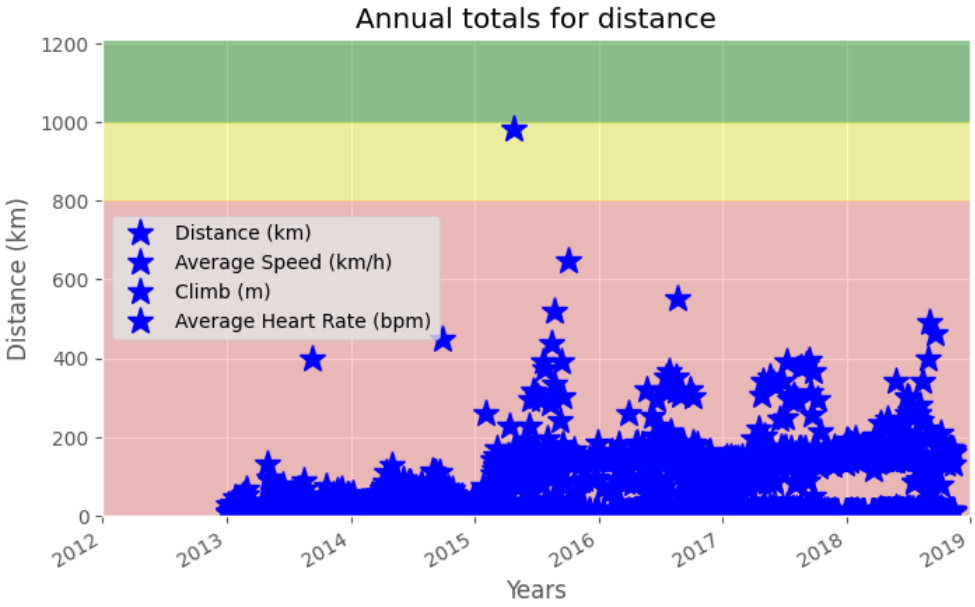
In this exercise, we prepared the data by taking out only two columns relevant to make another DataFrame. Then, we will have created a plot to understand the trend in the parameters distance and heart rate (6); thus, it proved helpful in knowledge relating to cardiovascular fitness for the user.



6. Trend in the parameters distance and heart rate

Task 7: Data preparation and plotting

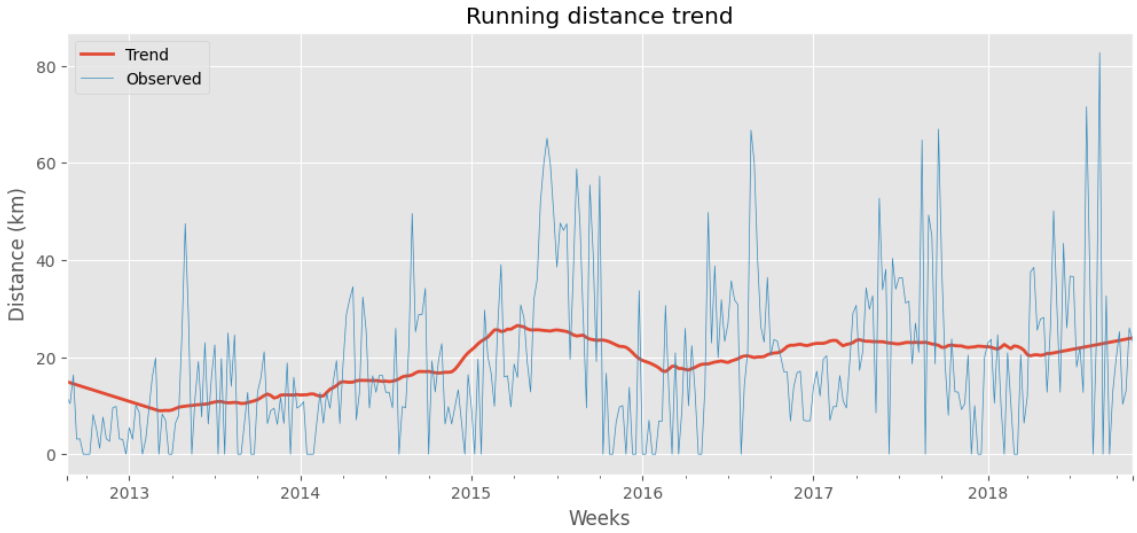
In this task, we prepared data by extracting columns of interest and creating a new DataFrame. After that, we created a plot of the annual totals of distances (7), which would let the user view his net progress in training.



7. Annual totals of distances

Task 8: Observed Distance and Decomposed Trend Plot

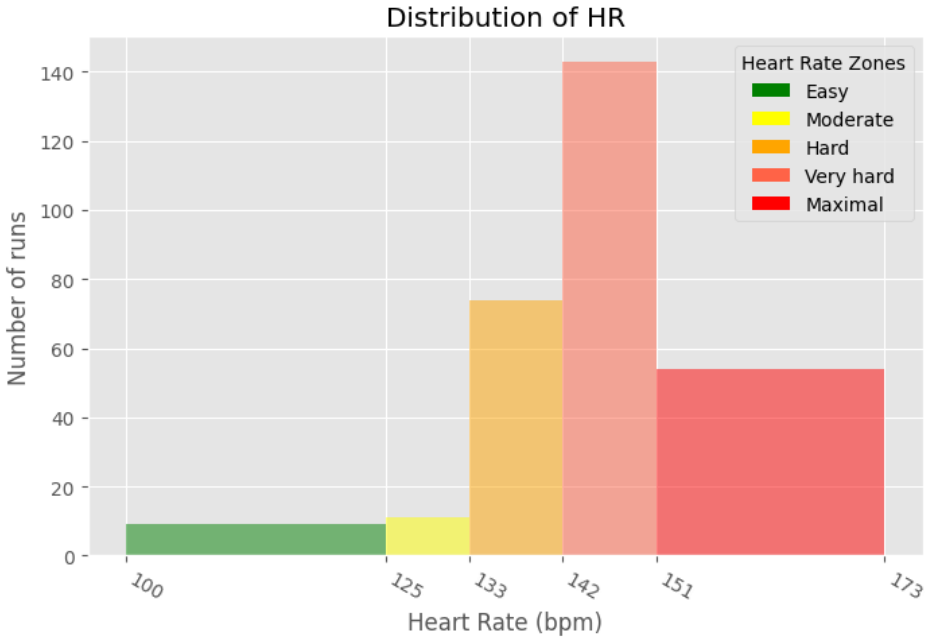
For this assignment, we generated a plot of observed distance of runs that includes decomposed trend using the statsmodels library. This graph conveys a lot of information about the user's progress with running—Trends and Seasonality (8).



8. User's progress with running—Trends and Seasonality

Task 9: Design a Customized Histogram for Heart Rate Distribution

In this exercise, we have generated a custom histogram of heart rate distribution that will let the user see their cardiovascular fitness (9). We applied plt.subplots() to create a customized plot with specific labels and titles.



9. Heart Rate Distribution

Task 10: Summary Report

This exercise has created the summary report by concatenating DataFrames using append() and grouping type of activity using groupby(). Totals are computed thereafter for each type of activity to summarize the progress of the user (10).

Totals for different training types:

	Type	Distance (km)	Climb (m)
0	Cycling	680.58	6976
1	Running	5224.50	57278
2	Walking	33.45	349

Summary statistics for different training types:

Type		Distance (km)	Climb (m)	Average Speed (km/h)
Cycling	25%	15.530000	139.000000	16.980000
	50%	20.300000	199.000000	19.500000
	75%	29.400000	318.000000	21.490000
	count	29.000000	29.000000	29.000000
	max	49.180000	553.000000	24.330000
	mean	23.468276	240.551724	19.125172
	min	11.410000	58.000000	11.380000
	std	9.451040	128.960289	3.257100
Running	25%	7.415000	54.000000	10.495000
	50%	10.810000	91.000000	10.980000
	75%	13.190000	171.000000	11.520000
	count	459.000000	459.000000	459.000000
	max	38.320000	982.000000	20.720000
	mean	11.382353	124.788671	11.056296
	min	0.760000	0.000000	5.770000
	std	4.937853	103.382177	0.953273
Walking	25%	1.385000	7.000000	5.555000
	50%	1.485000	10.000000	5.970000
	75%	1.787500	15.500000	6.512500
	count	18.000000	18.000000	18.000000
	max	4.290000	112.000000	6.910000
	mean	1.858333	19.388889	5.549444
	min	1.220000	5.000000	1.040000
	std	0.880055	27.110100	1.459309

10. Progress summary of the user

Task 11: Fun Questions

In this exercise we use FUN FACTS data to answer questions that were, well, just for fun with regard to the user's training progress. Amongst other items, we calculated an instructor's average shoes per lifetime and estimated how many shoes had gone through on the route of Forrest Gump, all to provide some light-hearted and entertaining project closure (11).

```
# Count average shoes per lifetime (as km per pair) using our fun facts
average_shoes_lifetime = 5224 / 7

# Count number of shoes for Forrest's run distance
shoes_for_forrest_run = 24700 / average_shoes_lifetime
shoes_for_forrest_run=int(shoes_for_forrest_run)
print('Forrest Gump would need {} pairs of shoes!'.format(shoes_for_forrest_run))
```

✓ 0.0s

```
Forrest Gump would need 33 pairs of shoes!
```

11. How many shoes had gone through on the route of Forrest Gump.

CONCLUSION

Furthermore, this would be an example of how data analysis could be very powerful in the light of fitness tracking. Information will be imported, cleaned, and analyzed to show our progress, achievements, and performance relative to other individuals. What sets this approach off from others is that the analysis is broken down into 11 tasks, each considering a specific aspect of the data.

These results can be extrapolated to all data related to fitness, thus making this tool useful for all people handling fitness data. It epitomizes the power of analyzing data in fitness tracking, offering comprehensive knowledge about one's fitness journey. We import, clean, then analyze the data, using Python and its popular libraries, Pandas and Matplotlib, so we are very informed of the training progress and the achievements realized by the user.

Worth pointing out here, however, is that the dataset used in the whole project corresponds to a single individual's fitness data. The beauty of this project will, however, be flexible and scalable—in other words, allowing us to make very minimal changes so as to be able to use other persons' datasets, thereby easily computing and analyzing their training progress and achievements. This makes the approach to the project very versatile and applicable in most datasets in fitness.