SIES (NERUL) COLLEGE OF ARTS, SCIENCE AND COMMERCE

NAAC ACCREDITED ‘A’ GRADE COLLEGE

(ISO 9001:2008 CERTIFIED INSTITUTION)

NERUL, NAVI MUMBAI – 400706

PROJECT REPORT ON

LIFESTYLE IMPROVEMENT THROUGH WEARABLE DATA ANALYSIS

SUBMITTED BY

KISHORE KUMAR MADASAMY NADAR

UNDER THE GUIDANCE OF

ASST. PROF. MANASVI SHARMA

SUBMITTED IN THE PARTIAL FULLFILMENT FOR THE DEGREE OF

MSc. COMPUTER SCIENCE

SEMESTER – IV, 2021 – 2022

SIES (NERUL) COLLEGE OF ARTS, SCIENCE AND COMMERCE

NAAC ACCREDITED 'A' GRADE COLLEGE

(ISO 9001:2015 CERTIFIED INSTITUTION)

NERUL, NAVI MUMBAI - 400706

Certificate

THIS IS TO CERTIFY THAT THE PROJECT TITLED

LIFESTYLE IMPROVEMENT THROUGH WEARABLE DATA ANALYSIS

IS UNDERTAKEN BY

KISHORE KUMAR MADASAMY NADAR

Seat No: 09

In partial fulfillment of the MSc - IT / CS Degree (Semester IV) Examinationin the academic year 2021-2022 and has not been submitted for any other examination and does not form part of any other course undergone by the candidate. It is further certified that he/she has completed all the required phases of the Project.

Project Guide External Examiner

Head of Department Principal

ACKNOWLEDGMENT

I extend my heartfelt gratitude and thanks to Asst. Professor Manasvi Sharma for providing me excellent guidance to work on this project and for their understanding and assistance by providing all the necessary information needed for my project topic. I would also like to acknowledge all the staffs for providing a helping hand to us in times of queries & problems. The project is a result of the efforts of all the peoples who are associated with the project directly or indirectly, who helped me to work to complete the project within the specified time frame. They motivated me in the project and gave a feedback on it to improve my adroitness.

Thanks to all my teachers, who were a part of the project in numerous ways and for the help and inspiration they extended to me and for providing the needed motivation.

With all Respects & Gratitude, I would like to thanks to all the people, who have helped for the development of the Project.

KISHORE KUMAR MADASAMY NADAR
MSc. Computer Science (Part-II) SIES (Nerul)
College of Arts, Science, and Commerce.

PROJECT ON

LIFESTYLE IMPROVEMENT THROUGH WEARABLE DATA ANALYSIS

ABSTRACT

Wearable technology comes with the promise of improving one’s lifestyles thru data mining of their physiological condition. The potential to generate a change in daily or routine habits thru these devices leaves little doubt. Whilst the hardware capabilities of wearables have evolved rapidly, software apps that interpret and present the physiological data and make recommendations in a simple, clear and meaningful way have not followed a similar pattern of evolution. Existing fitness apps provide routinely some information to the wearer by mining personal data but the subsequent analysis is limited to supporting ad hoc personal goals. The information and recommendations presented are often either not entirely relevant or incomplete and often not easy to interpret by the wearer. The primary motivation behind this project is to address this wearable technology software by data analytics and machine learning to assist with interpretation of wearer data and with making of personal lifestyle improvement recommendations on the go which may then be used to feedback to the wearer’s daily goals and activities. The secondary motivation is to correlate and compare with trends in the wearer’s peer community.

INTRODUCTION

Google's Verily Life Sciences have recently unveiled an ordinary looking health tracking "study watch" to unobtrusively and continuously collect physiological data from the wearer, such as their heart rate, electrocardiograms (ECG), movement data, their skin's electrical conductance as well as ambient light and sound. The Google watch does not currently allow the wearer to see their health data, only the date, time, and some instructions. The long-term vision of Verily is to unravel biomarkers through tracking thousands of healthy people especially when they fall ill. Although the watch features a processor that manages and encrypts its wearer's data and its software is updatable over-the-air, verily is not currently marketing the watch as a medical device, for which they will need FDA approval, but as a clinical and observational tool for scalable collection of rich and complex data sets. The watch's battery lasts for up to a week without a charge and it stores "raw" data produced over the same amount of time, so it does not have to be synched as frequently as other watches. The inclusion of an ECG which Verily regards as their biggest technical novelty can reveal heart abnormalities. Such measurements are normally taken in hospitals with several stick-on electrodes, but the Google watch picks up a lower-resolution signal from just two electrical contacts when the wearer grasps the metal bezel with their other hand. The watch is being deployed in a study in Europe whose aim is to track the progression of Parkinson's disease among patients diagnosed with the disease. Uninterrupted long-term sleep and heart readings are invaluable in monitoring the progression of Parkinson's and cannot be monitored in hospitals, especially the patients' sleep patterns. Verily argues that the watch's scope can be extended through the inclusion of additional sensors that will help generate additional biofeedback thus shaping its place in the Internet of Things (IoT) landscape

Having continuous and uninterrupted access to such personal data in real time may help with the diagnosis of underlying conditions on the go, as maybe the case with Google's watch, and making personal recommendations on lifestyle improvements, otherwise, by identifying personal preferences and behavior including food habits, sleep patterns, and daily activity schedules. It is widely acknowledged that all these devices can motivate individuals to change habits towards a better health or lifestyle. For instance, they are being considered in the workplace to monitor workers' activities and control schedules and as they hold consumer’s health data, these devices are also a point of interest for physicians and health insurers

Samsung, Fitbit, Apple and Sony have long joined the race in wearable technology hardware but according to Verily, the next but necessary stage in the evolution is the development of software algorithms that can “interpret” the wearer’s raw data, thus making accurate diagnoses and personalized recommendations, where possible, often by comparing the wearer’s data against the data of other wearers in the same peer group. As these devices cover a broad range of wearers of different ages, physical conditions and lifestyles, it is important that the presented information is as useful to all wearers as the wearable hardware with which it has been recorded. This would in turn inform the development of personalized immersive and alternative reality environments.

The focus of this project is to address the wearable technology software challenge by developing such a personal immersive environment that collects, analyses and visualizes personal wearer data mined with a wearable device and which integrates a machine learning approach to making personal recommendations for lifestyle improvements on the go in close consideration of trends in the wearer’s peer community. These lifestyle improvement recommendations may then be used to feedback to the wearer’s daily goals and activities.

IMPLEMENTATION DETAILS

Data Collection

Existing Data:
I have downloaded the data from the kaggle. The dataset is available in csv format. Data set link: https://www.kaggle.com/arashnic/fitbit

The dataset is of personal tracker data, including minute-level output for physical activity, heart rate, and sleep monitoring. Individual reports can be parsed by export session ID or timestamp. Variation between output represents use of different types of Fitbit trackers and individual tracking behaviors / preferences.

System Requirement:

Google Colab:
I used Google Colab for this project to write and execute arbitrary python code through the browser.Colab is hosted notebook service that requires no setup to use , while providing free access to computing resources including GPUs.
You can search Colab notebooks using Google Drive.Clicking on the Colab logo at the top left of the notebook view will show all notebooks in Drive. You can also search for notebooks that you have opened recently using File > Open notebook.

Algorithm:

I have used the classification algorithms in this project.

Matlab's Genetic Algorithm with Pareto Optimality:

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. You can apply the genetic algorithm to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear. The genetic algorithm can address problems of *mixed integer programming*, where some components are restricted to be integer-valued

To analyse and compare the wearer's data to peer community data with the former and to use the result to generate recommendations that may support the wearer with monitoring their daily goals and activities with the latter.

Decision Tree:
Decision tree is a supervised learning algorithm that is perfect for classification problems. As my dataset contains a high number of categorical values, I used decision tree to divide the data into leaf and nodes to predict the outcome.

Bootstrap Forest:
The Bootstrap Forest uses many decision tree type classification models, based on data and variable subsets to determine an optimal model. Through this bootstrapping methodology, a superior model can typically be generated relative to typical decision tree partitioning methods.

Naive Bayes algorithm:
It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

EXPERIMENTAL SET UP AND RESULTS

Existing Data:

The Dataset has 28 attributes that contains information of fitness data of a community, Id,ActivityDate,TotalSteps,TotalDistance,TrackerDistance,LoggedActivitiesDistance,VeryActiveDistance,ModeratelyActiveDistance,LightActiveDistance,SedentaryActiveDistance,VeryActiveMinutes,FairlyActiveMinutes,LightlyActiveMinutes,SedentaryMinutes,Calories,ActivityDay, SedentaryMinutes,LightlyActiveMinutes,FairlyActiveMinutes,VeryActiveMinutes,SedentaryActiveDistance,LightActiveDistance,ModeratelyActiveDistance,VeryActiveDistance,TotalSteps.

```
df_total.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 940 entries, 0 to 939
Data columns (total 15 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Id                        940 non-null    int64
 1   ActivityDate              940 non-null    object
 2   TotalSteps                940 non-null    int64
 3   TotalDistance             940 non-null    float64
 4   TrackerDistance           940 non-null    float64
 5   LoggedActivitiesDistance  940 non-null    float64
 6   VeryActiveDistance        940 non-null    float64
 7   ModeratelyActiveDistance  940 non-null    float64
 8   LightActiveDistance       940 non-null    float64
 9   SedentaryActiveDistance   940 non-null    float64
 10  VeryActiveMinutes         940 non-null    int64
 11  FairlyActiveMinutes       940 non-null    int64
 12  LightlyActiveMinutes      940 non-null    int64
 13  SedentaryMinutes          940 non-null    int64
 14  Calories                  940 non-null    int64
dtypes: float64(7), int64(7), object(1)
memory usage: 110.3+ KB
```

	Id	Time	Value
0	2022484408	4/12/2022 7:21:00 AM	97
1	2022484408	4/12/2022 7:21:05 AM	102
2	2022484408	4/12/2022 7:21:10 AM	105
3	2022484408	4/12/2022 7:21:20 AM	103
4	2022484408	4/12/2022 7:21:25 AM	101
...	...	...	...
2483653	8877689391	5/12/2022 2:43:53 PM	57
2483654	8877689391	5/12/2022 2:43:58 PM	56
2483655	8877689391	5/12/2022 2:44:03 PM	55
2483656	8877689391	5/12/2022 2:44:18 PM	55
2483657	8877689391	5/12/2022 2:44:28 PM	56

2483658 rows × 3 columns

The Dataset Consist of 2483658 rows and 3 columns.

Reading The Files:

```
import pandas as pd
df_heart = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/msc project dataset 2022/heartrate_seconds.csv')
df_heart.head(100000000)
```

	Id	Time	Value
0	2022484408	4/12/2022 7:21:00 AM	97
1	2022484408	4/12/2022 7:21:05 AM	102
2	2022484408	4/12/2022 7:21:10 AM	105
3	2022484408	4/12/2022 7:21:20 AM	103
4	2022484408	4/12/2022 7:21:25 AM	101
...	...	...	...
2483653	8877689391	5/12/2022 2:43:53 PM	57
2483654	8877689391	5/12/2022 2:43:58 PM	56
2483655	8877689391	5/12/2022 2:44:03 PM	55
2483656	8877689391	5/12/2022 2:44:18 PM	55
2483657	8877689391	5/12/2022 2:44:28 PM	56

2483658 rows × 3 columns


```
import pandas as pd
df_heart = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/msc project dataset 2022/heartrate_seconds.csv')
df_heart.head(100000000)
```

	Id	Time	Value
0	2022484408	4/12/2022 7:21:00 AM	97
1	2022484408	4/12/2022 7:21:05 AM	102
2	2022484408	4/12/2022 7:21:10 AM	105
3	2022484408	4/12/2022 7:21:20 AM	103
4	2022484408	4/12/2022 7:21:25 AM	101
...	...	...	...
2483653	8877689391	5/12/2022 2:43:53 PM	57
2483654	8877689391	5/12/2022 2:43:58 PM	56
2483655	8877689391	5/12/2022 2:44:03 PM	55
2483656	8877689391	5/12/2022 2:44:18 PM	55
2483657	8877689391	5/12/2022 2:44:28 PM	56

2483658 rows × 3 columns

Data Cleaning:

#Checking null values, But looking at the dataset, we can see that there is no missing values

Checking missing values

```
total=df_total.isnull().sum().sort_values(ascending=False)
print(total)
```

```
Id                          0
ActivityDate                0
TotalSteps                  0
TotalDistance               0
TrackerDistance             0
LoggedActivitiesDistance    0
VeryActiveDistance          0
ModeratelyActiveDistance    0
LightActiveDistance         0
SedentaryActiveDistance     0
VeryActiveMinutes           0
FairlyActiveMinutes         0
LightlyActiveMinutes        0
SedentaryMinutes            0
Calories                    0
dtype: int64
```

- hourlycalories.csv
- hourlyintensities.csv
- hourlysteps.csv
- minutecalories.csv
- minutecalorieswid...
- minuteintensitiesna...
- minuteintensitieswi...

```
[5] total2=df_heart.isnull().sum().sort_values(ascending=False)
    print(total2)
```

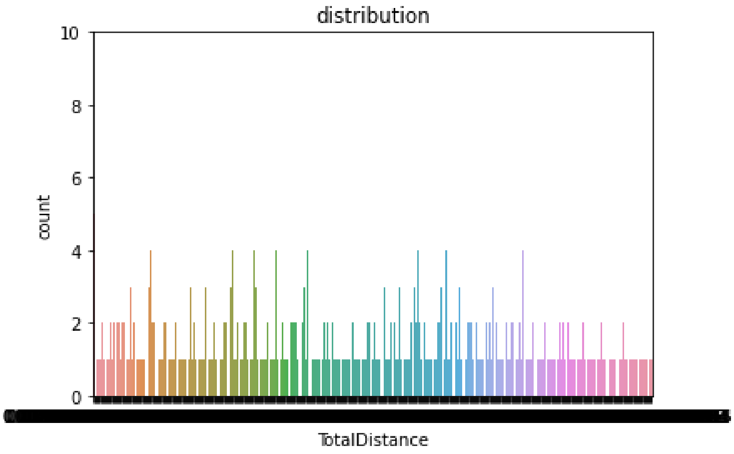
```
Id       0
Time     0
Value    0
dtype: int64
```

Data Visualization:

#countplot of total distance and calories burnt

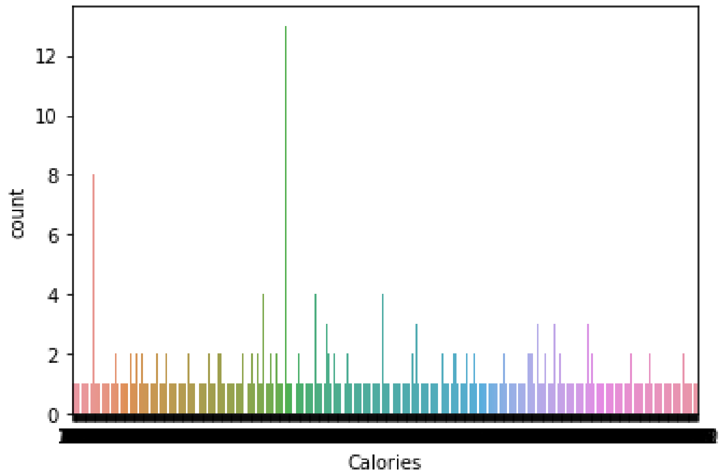
```
sns.countplot(df_total['TotalDistance']).set_title('distribution')
plt.ylim(0,10)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
  FutureWarning
```




```
sns.countplot(df_total['Calories'])
plt.show()
```

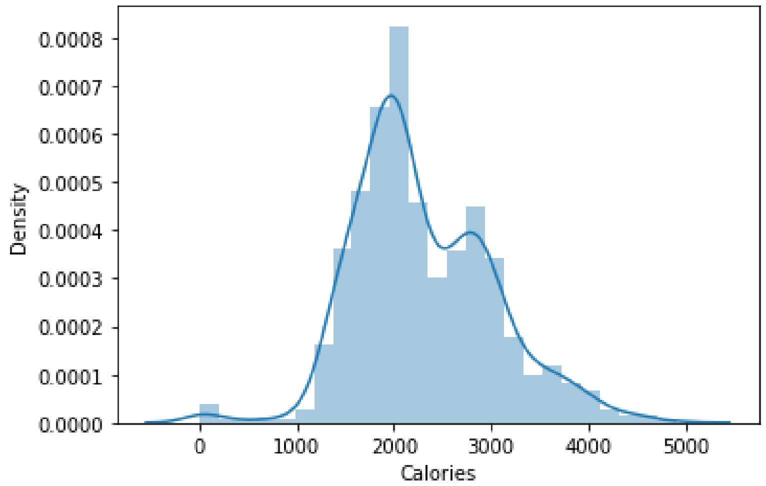
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
  FutureWarning
```



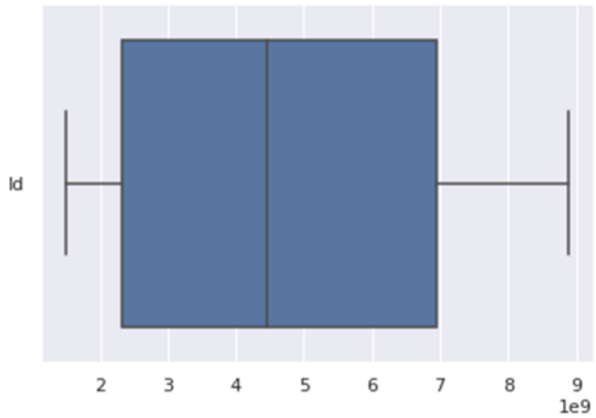
DISTPLOT OF **CALORIES**

```
fig1=sns.distplot(df_total['Calories'])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py
  warnings.warn(msg, FutureWarning)
```

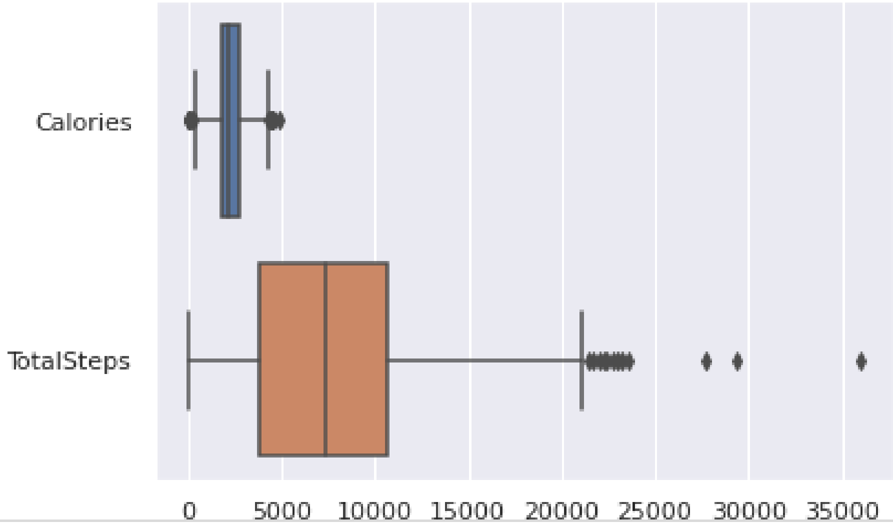



```
[80] data1=['Id','ActivityDate']
     ax=sns.boxplot(data=df_total[data1],orient='h')
```



#Box plot of the whole data

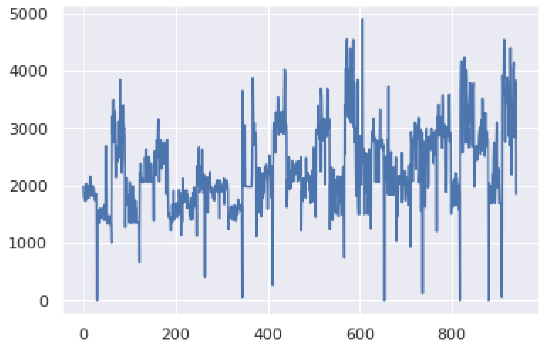
```
data2=['Calories','TotalSteps']
ax=sns.boxplot(data=df_total[data2],orient='h')
```



#Visualization of Total steps per month and calories burnt

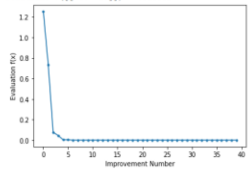
```
[94] df_calories=df_total['Calories']
     df_calories.shape

     (940,)
```

```
plt.plot(df_calories)

[<matplotlib.lines.Line2D at 0x7fc6a909be10>]
```



#Genetic algorithm

```
Iteration: 49 f([[ 0. -0.]]) = 0.00000
Iteration: 50 f([[0. 0.]]) = 0.00000
Iteration: 51 f([[ 0. -0.]]) = 0.00000
Iteration: 53 f([[-0. -0.]]) = 0.00000
Iteration: 55 f([[-0.  0.]]) = 0.00000
Iteration: 56 f([[-0. -0.]]) = 0.00000
Iteration: 57 f([[-0. -0.]]) = 0.00000
Iteration: 58 f([[-0. -0.]]) = 0.00000
Iteration: 60 f([[-0. -0.]]) = 0.00000
Iteration: 61 f([[ 0. -0.]]) = 0.00000
Iteration: 62 f([[ 0. -0.]]) = 0.00000
Iteration: 63 f([[-0. -0.]]) = 0.00000
Iteration: 65 f([[ 0. -0.]]) = 0.00000
Iteration: 66 f([[ 0. -0.]]) = 0.00000
Iteration: 70 f([[-0.  0.]]) = 0.00000
Iteration: 72 f([[-0.  0.]]) = 0.00000
Iteration: 75 f([[0. 0.]]) = 0.00000
Iteration: 76 f([[-0. -0.]]) = 0.00000
Iteration: 77 f([[ 0. -0.]]) = 0.00000
Iteration: 81 f([[0. 0.]]) = 0.00000
Iteration: 86 f([[-0.  0.]]) = 0.00000
Iteration: 91 f([[-0. -0.]]) = 0.00000
Iteration: 94 f([[-0.  0.]]) = 0.00000
Done!

Solution: f([[-0.  0.]]) = 0.00000
```




```
from numpy.random import randint
from numpy.random import rand
```

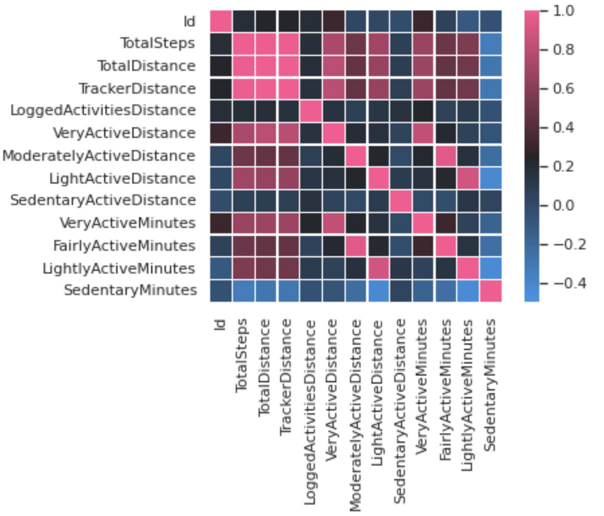
```
>0, iteration f([-1.51214599609375, -0.866851806640625]) = 3.038018
>0, iteration f([-0.364227294921875, 1.629638671875]) = 2.788384
>0, iteration f([0.101318359375, -1.02249145507 8125]) = 1.055754
>0, iteration f([-0.55160522460 9375, 0.3997802734375]) = 0.464093
>1, iteration f([-0.548858642578125, 0.059661865234375]) = 0.304805
>1, iteration f([-0.168609619140625, -0.496826171875]) = 0.275265
>1, iteration f([-0.239105224609375, 0.3997802734375]) = 0.216996
>2, iteration f([0.111083984375, -0.12725830078125]) = 0.028534
>3, iteration f([0.003662109375, 0.10894775390625]) = 0.011883
>6, iteration f([0.01708984375, 0.0125122070 3125]) = 0.000449
>8, iteration f([0.0164794921875, 0.00015258789 0625]) = 0.000272
>10, iteration f([0.0067138671875, 0.01373291015625]) = 0.000234
>11, iteration f([0.001983642578125, 0.013275146484375]) = 0.000180
>12, iteration f([0.0042724609375, 0.01129150390625]) = 0.000146
>13, iteration f([0.0042724609375, 0.01007080078125]) = 0.000120
>14, iteration f([0.0103759765625, 0.0]) = 0.000108
>14, iteration f([0.0094604492187 5, 0.00030517578125]) = 0.000090
>14, iteration f([0.00457763671875, 0.00320434570 3125]) = 0.000031
>14, iteration f([0.0042724609375, 0.0033569335 9375]) = 0.000030
>15, iteration f([0.0030517578125, 0.00030517578125]) = 0.000009
>17, iteration f([0.00213623046875, 0.00030517578125]) = 0.000005
>19, iteration f([0.0018310546875, 0.000762939453125]) = 0.000004
>20, iteration f([0.00030517578125, 0.0006103515625]) = 0.000000
>22, iteration f([0.000152587890625, 0.0006103515625]) = 0.000000
>23, iteration f([0.000152587890625, 0.00030517578125]) = 0.000000
>24, iteration f([0.000152587890625, 0.0]) = 0.000000
>36, iteration f([0.0, 0.0]) = 0.000000
Done!
f([0.0, 0.0]) = 0.000000
```

All changes saved

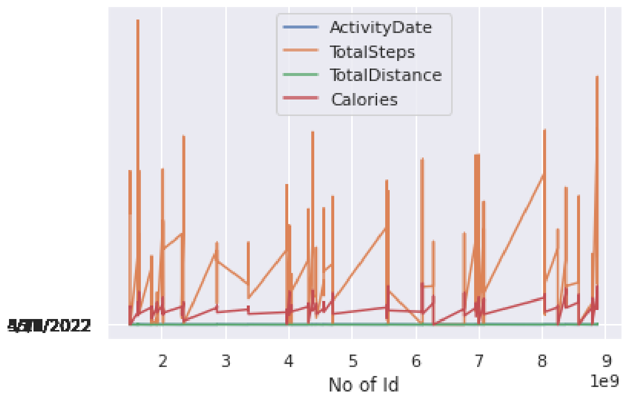
\+ Code + Text

```
corr = df_total.iloc[:,:-1].corr(method="pearson")
cmap = sns.diverging_palette(250,354,80,60,center='dark',as_cmap=True)
sns.heatmap(corr, vmax=1, vmin=-.5, cmap=cmap, square=True, linewidths=.2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc63e95fe10>
```

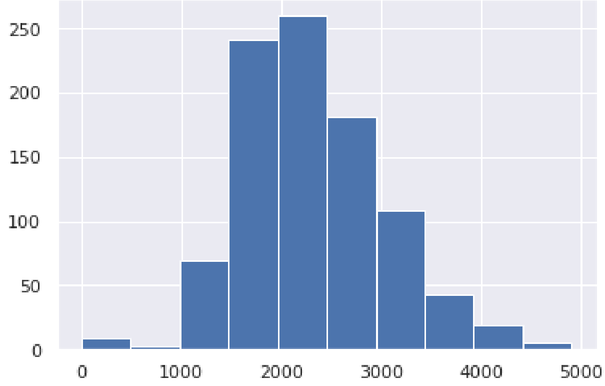


#File (df_total) cmap and heatmap of whole user data



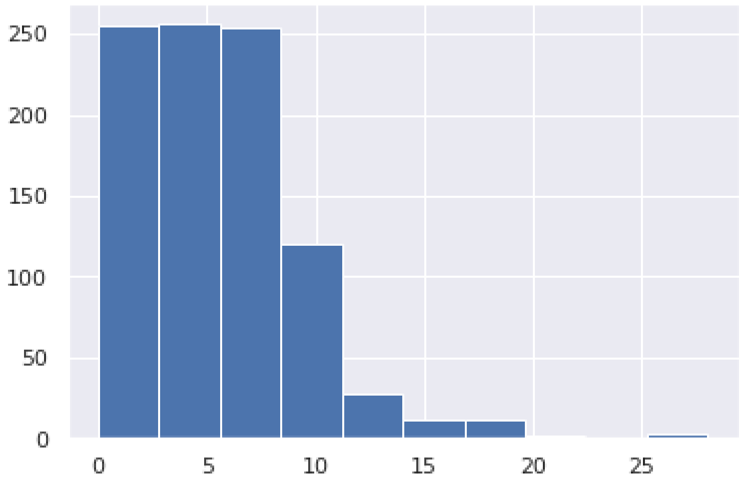
```
df_total["Calories"].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc637e21110>
```



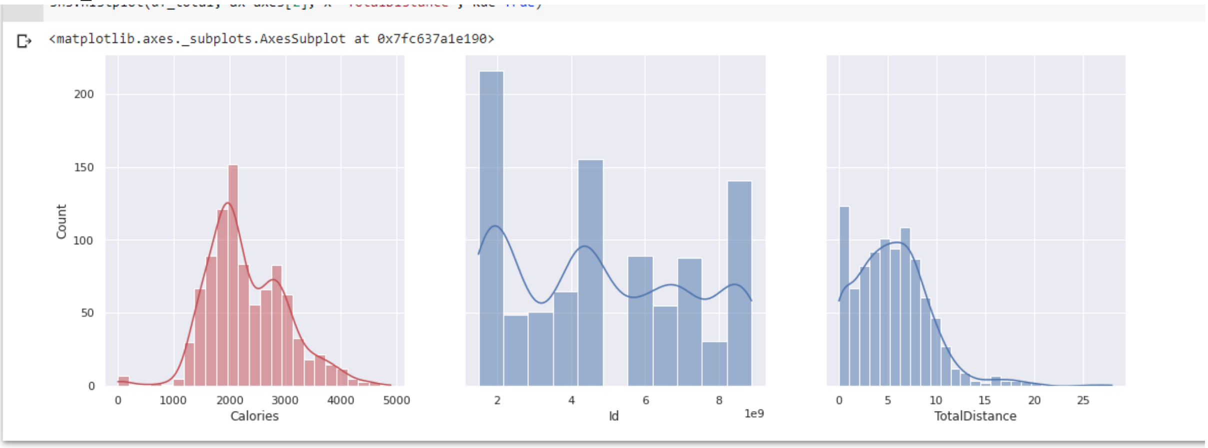

```
df_total["TotalDistance"].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc637db3290>
```

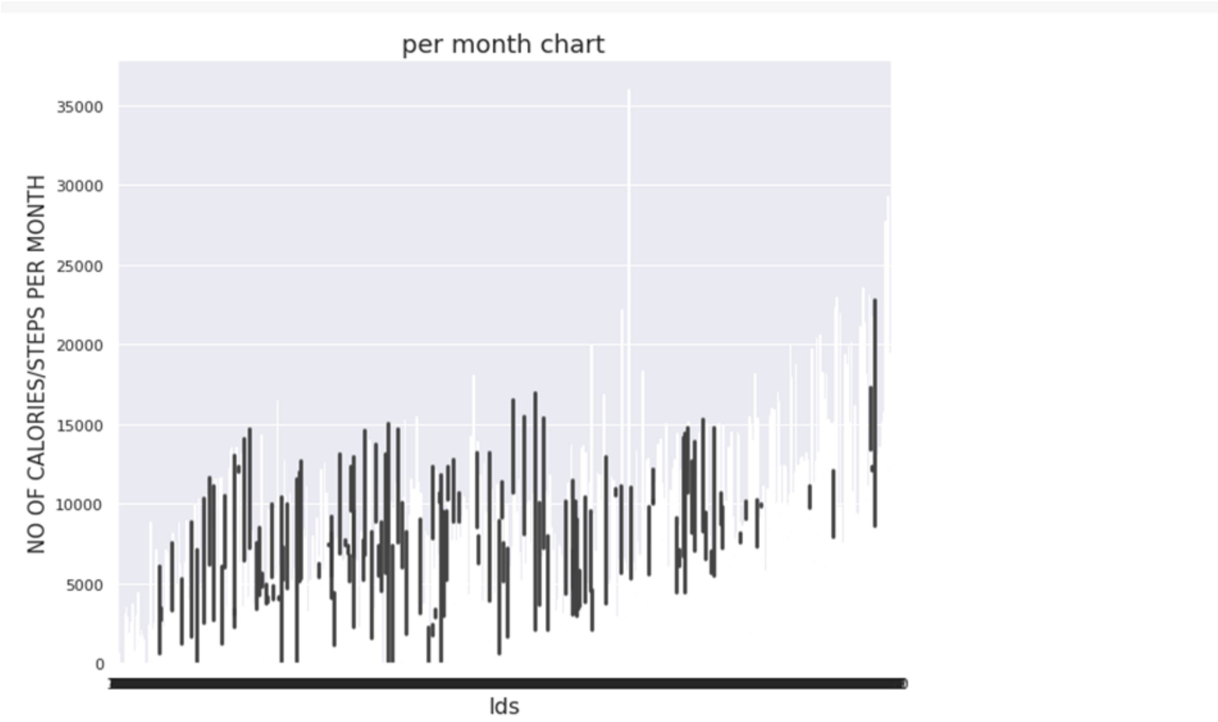


#Bar graph of distance traveled per day

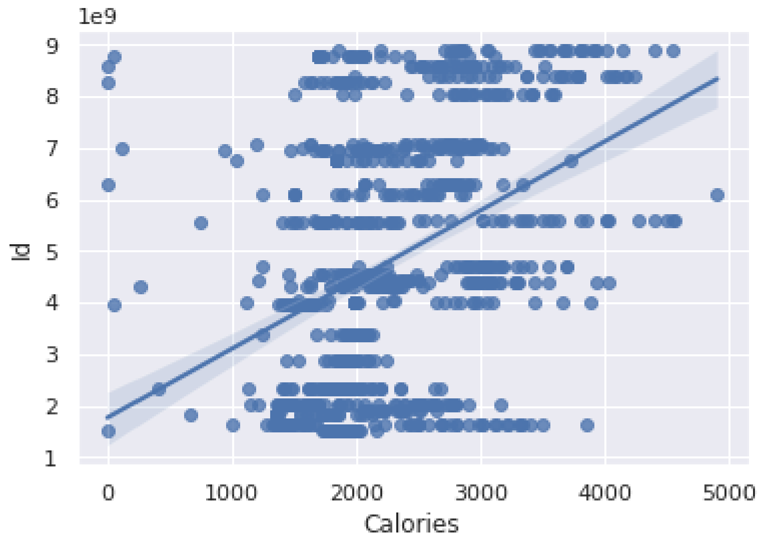
#Subplots of calories ,id and totaldistance



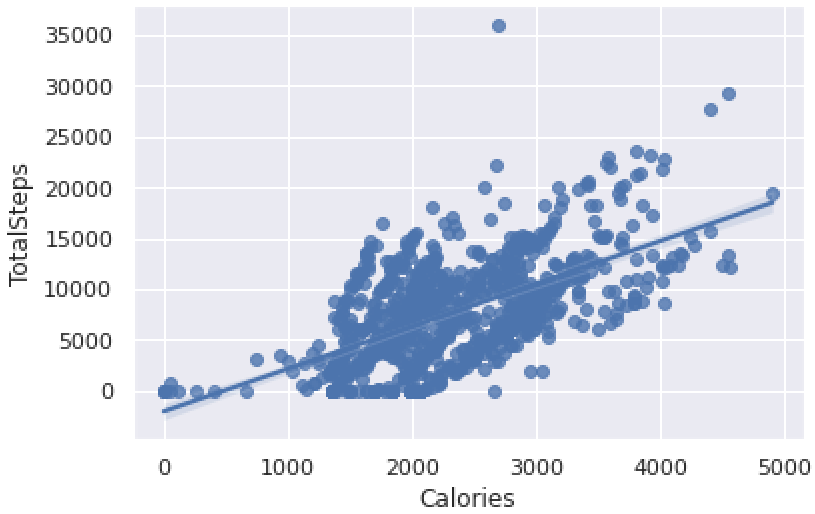
#No of calories and steps per month



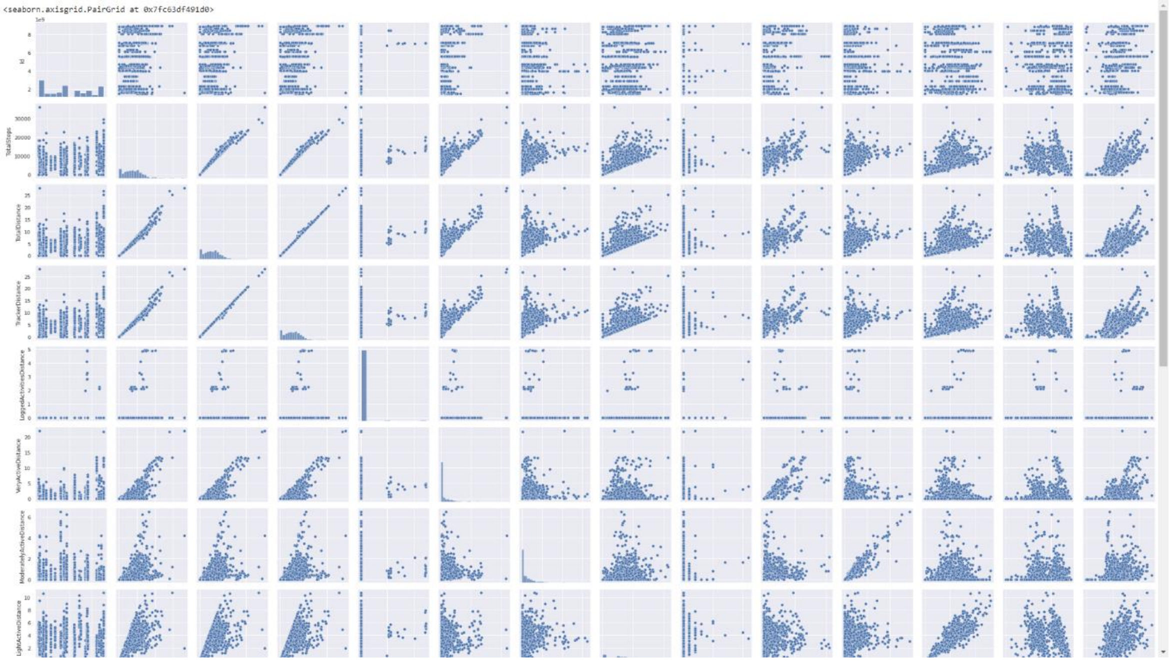
```
[159] sns.regplot(x="Calories", y="Id", data=df_total);
```



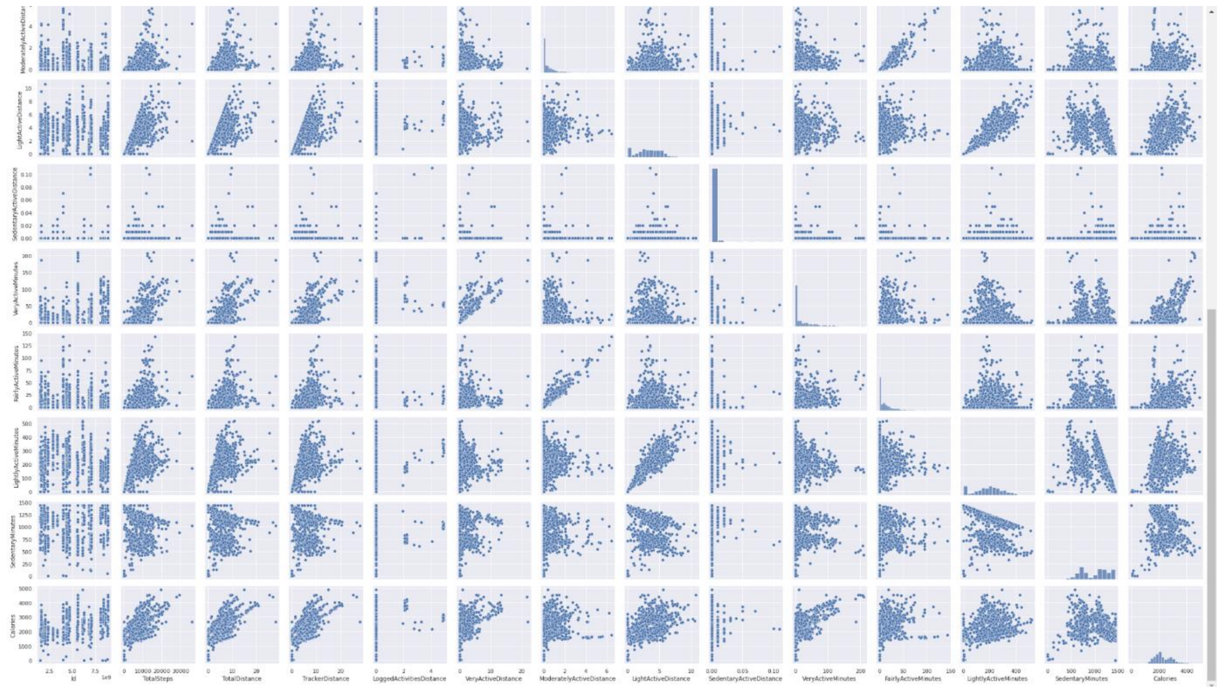

```
sns.regplot(x="Calories", y="TotalSteps", data=df_total);
```



#regplot of calories with id and calories with total steps



#pairgrid of total calories,totalsteps,VeryActiveMinutes,SedentaryActiveDistance,LightActiveDistance,ModeratelyActiveDistance,VeryActiveDistance,TotalSteps.



#PairgridofId,ActivityDate,TotalSteps,TotalDistance,TrackerDistance,LoggedActivitiesDistance,VeryActiveDistance,ModeratelyActiveDistance,LightActiveDistance,SedentaryActiveDistance,VeryActiveMinutes,FairlyActiveMinutes,LightlyActiveMinutes,SedentaryMinutes,Calories,ActivityDay,SedentaryMinutes,LightlyActiveMinutes,FairlyActiveMinutes

Analysis of the results

Interpretation:

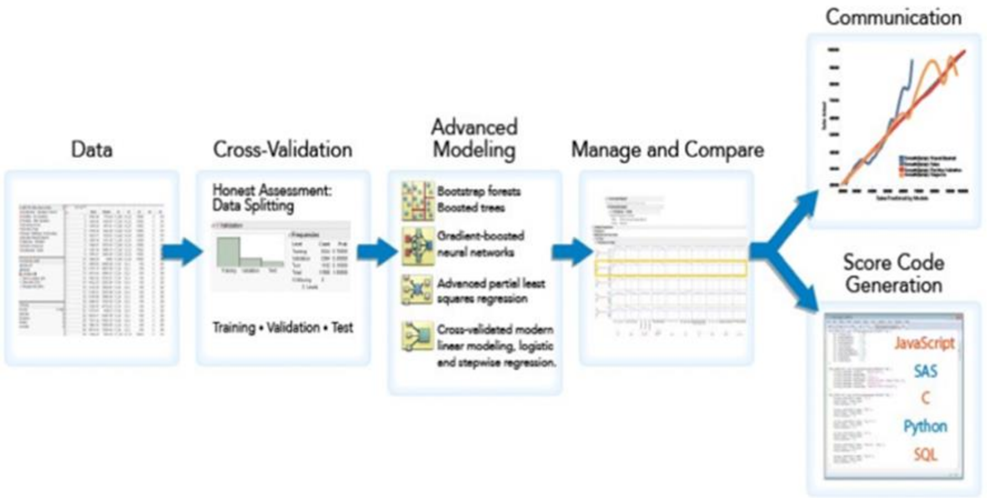
Existing Dataset:-

Matlab’s Genetic Algorithm with Pareto Optimality:

We have integrated JMP PRO with MatLab’s Genetic Algorithms (GA) and Pareto Optimality (PO) to analyse and compare the wearer’s data to peer community data with the former and to use the result to generate recommendations that may support the wearer with monitoring their daily goals and activities.

Figure shows a flowchart of the use of a wearable device, the analysis and visualisation of the wearer activity data it records and the recommendations generated using a machine learning technique to support the wearer with their monitoring of their daily goals and activities. The wearer configures their wearable and sets initial goals. Generating an optimal recommendation D_i for each wearer becomes a multi-optimisation problem (MOP) which seeks to optimise objectives ⋃kd=1Od(D)≤0⋃d=1kOd(D)≤0 subject to limit constraints ⋃md=1Ld(D)≤0⋃d=1mLd(D)≤0: F(x→)=(f1(x→)……fk(x→)) F(x→)=(f1(x→)……fk(x→)) where x→x→ is an n-dimensional decision variable vector (x→=x1,…,xn)(x→=x1,…,xn). The objectives and limit constraints are sourced from the best goodness of fit wearer model produced by colab in comparison to peer community data, e.g. steps per day, hours sleeping per night, hours exercising per day

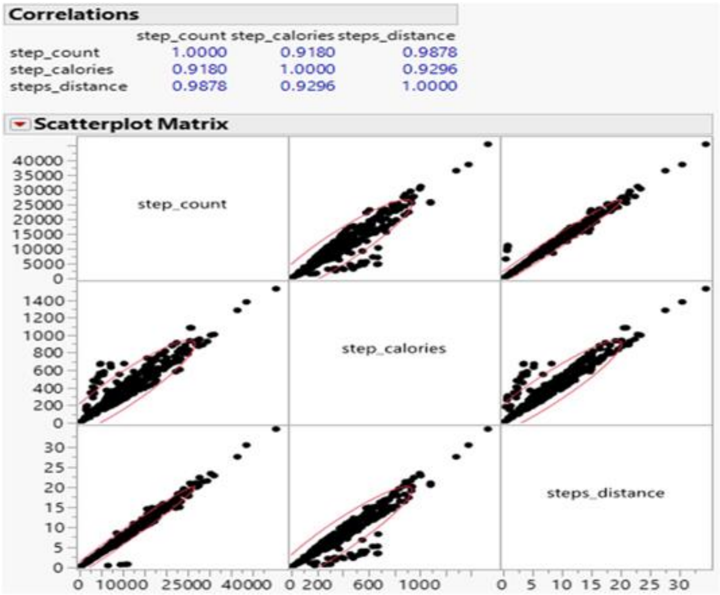
Decision Tree,Bootstrap Forest,Naive Bayes algorithm:



Comparative analysis and recommendation generation

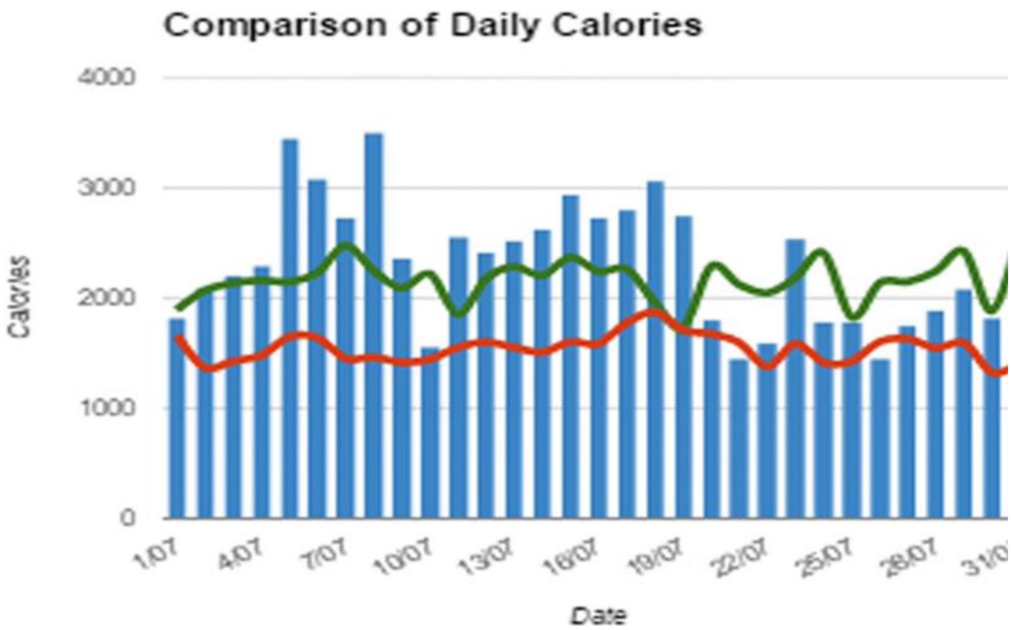
The above Figure shows a flowchart of the process of comparative analysis with GoogleColab. GoogleColab has been selected for raw wearable data analysis and comparison to peer data for many reasons. First, it enables predictive modelling through the set of techniques it deploys, e.g. **decision trees**, **bootstrap forest**, **Naive Bayes** and **neural networks**. Stats emerge with these techniques, e.g. the average influence of additional hours of sleeping on calorie burning, residual variation of number of steps mid-week and at weekends, all of which help with model prediction accuracy even with missing or incomplete data. Second, it enables cross-validation through the set of techniques it uses, e.g. data partitioning or holdback. Cross-validation is not only invaluable when comparing wearer data to peer data but also building a predictive model that is not based on a single wearer sample which in turn avoids over-fitting. Third, it allows model comparison through common quality measures, e.g. R2, ROC, AUC, etc. This helps with selecting a model with the best goodness of fit, parsimony and cross-validation as input to the GA. Fourth, it uses generalized regression through a set of regularization techniques such as Ridge, Lasso, adaptive Lasso, Elastic Net, adaptive Elastic Net that help overcome the biases that arise with strongly correlated predictors, e.g. number of hours sleeping and calorie burning which may result in over-fitting. This approach helps with building a diverse predictive model that may include data with many outliers, or skewed data.

Fitbit tracks activity and calories burned through exercise intensity using Steps, Floors and Heart Rate along with Metabolic Equivalent (MET), BMR, weight and height. BMR values usually account for at least half of the calories burnt in a day and this is estimated based on gender, age, height, and weight. Table shows the Fitbit parameters tracked

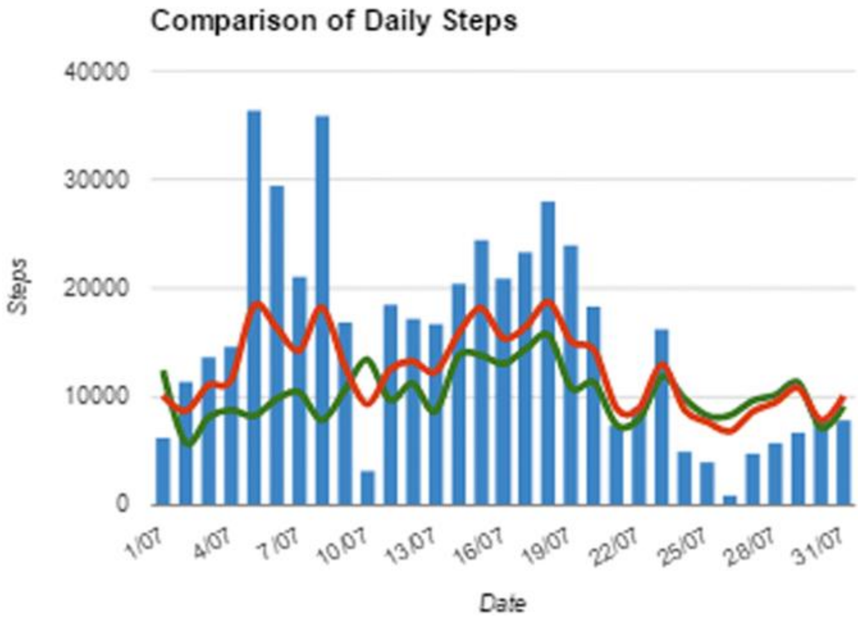

Correlations

	step_count	step_calories	steps_distance
step_count	1.0000	0.9180	0.9878
step_calories	0.9180	1.0000	0.9296
steps_distance	0.9878	0.9296	1.0000

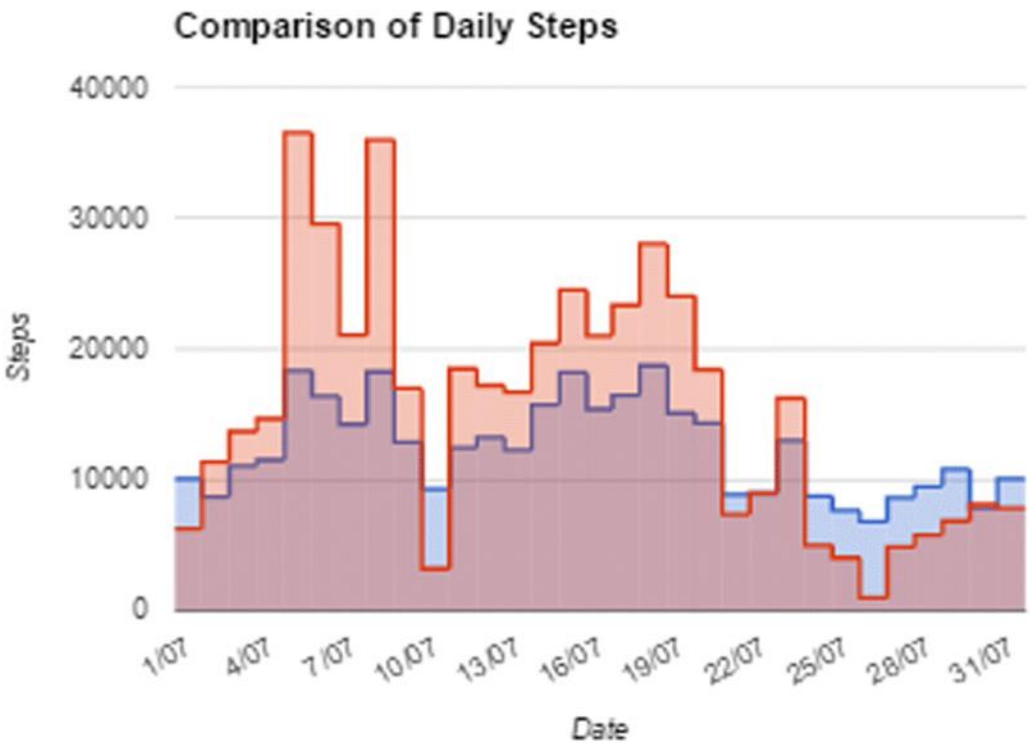
#Camparison of user data and peer community data



#Camparison of daily steps of user data,daily goal and peer community data



#Camparison of daily steps of user data and peer community data



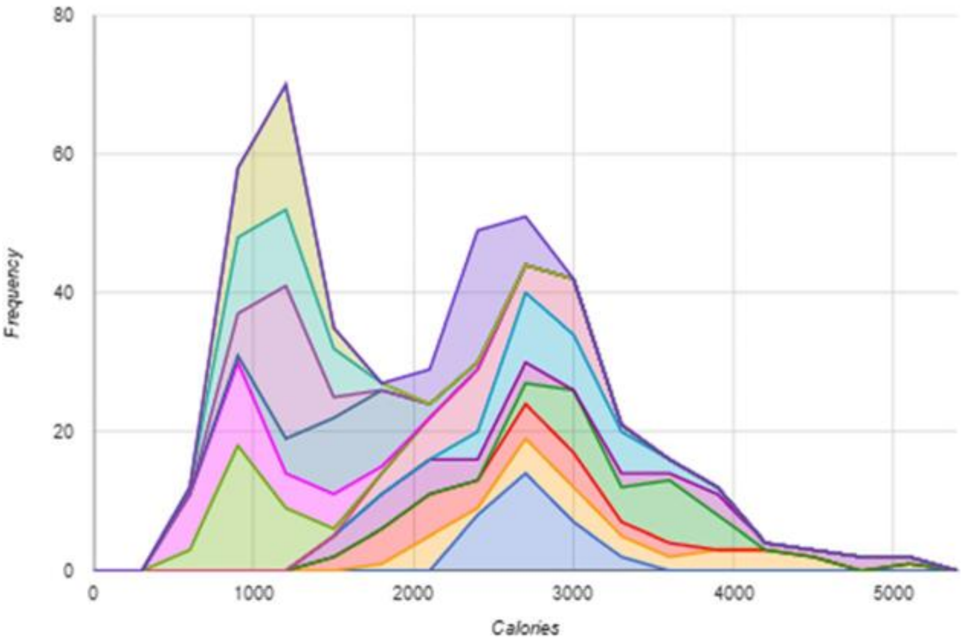
```
df_total.describe().transpose()
```

1 to 14 of 14 entries Filter

index	count	mean	std	min	25%	50%	75%	max
Id	940.0	4855407369.332978	2424805475.657955	1503960366.0	2320127002.0	4445114986.0	6962181067.0	8877689391.0
TotalSteps	940.0	7637.9106382978725	5087.150741753409	0.0	3789.75	7405.5	10727.0	36019.0
TotalDistance	940.0	5.489702121915416	3.924605908624871	0.0	2.61999988555908	5.24499988555908	7.71249997615814	28.0300006866455
TrackerDistance	940.0	5.475351057821845	3.9072759432009443	0.0	2.61999988555908	5.24499988555908	7.71000003814697	28.0300006866455
LoggedActivitiesDistance	940.0	0.10817093988682361	0.6198965182108744	0.0	0.0	0.0	0.0	4.94214200973511
VeryActiveDistance	940.0	1.502680850999945	2.6589411648346166	0.0	0.0	0.209999993443489	2.0524999499321	21.9200000762939
ModeratelyActiveDistance	940.0	0.5675425513706943	0.883580319140428	0.0	0.0	0.239999994635582	0.800000011920929	6.48000001907349
LightActiveDistance	940.0	3.3408191485885292	2.04065538820603	0.0	1.9450000226497675	3.364999890327455	4.78250014781952	10.710000038147
SedentaryActiveDistance	940.0	0.0016063829566887054	0.007346176286859482	0.0	0.0	0.0	0.0	0.109999999403954
VeryActiveMinutes	940.0	21.164893617021278	32.84480305692363	0.0	0.0	4.0	32.0	210.0
FairlyActiveMinutes	940.0	13.564893617021276	19.987403953867602	0.0	0.0	6.0	19.0	143.0
LightlyActiveMinutes	940.0	192.8127659574468	109.17469975147056	0.0	127.0	199.0	264.0	518.0
SedentaryMinutes	940.0	991.2106382978724	301.2674367904795	0.0	729.75	1057.5	1229.5	1440.0
Calories	940.0	2303.609574468085	718.1668621342561	0.0	1828.5	2134.0	2793.25	4900.0

Show 100 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.

#Average of steps,calories burnt,sleepcycle compared with main subject



#Participates Data of Calories burnt

Conclusion

The project demonstrates how wearable technology may be used beyond the hype to improve personal lifestyles by suggesting areas where generating a change in daily or routine habits may lead to a healthier lifestyle. Currently, such cycles appear to be driven by irrelevant or incomplete information which may not be easy to interpret but under peer pressure and the fear of exclusion, wearers adopt *ad hoc* change regardless. My project demonstrates that a planned approach to data collection, analysis and visualization coupled with a machine learning recommender approach may transpose wearer decisions to be informed and that the use of peer group data is for generating correlated recommendations not for applying social pressure.

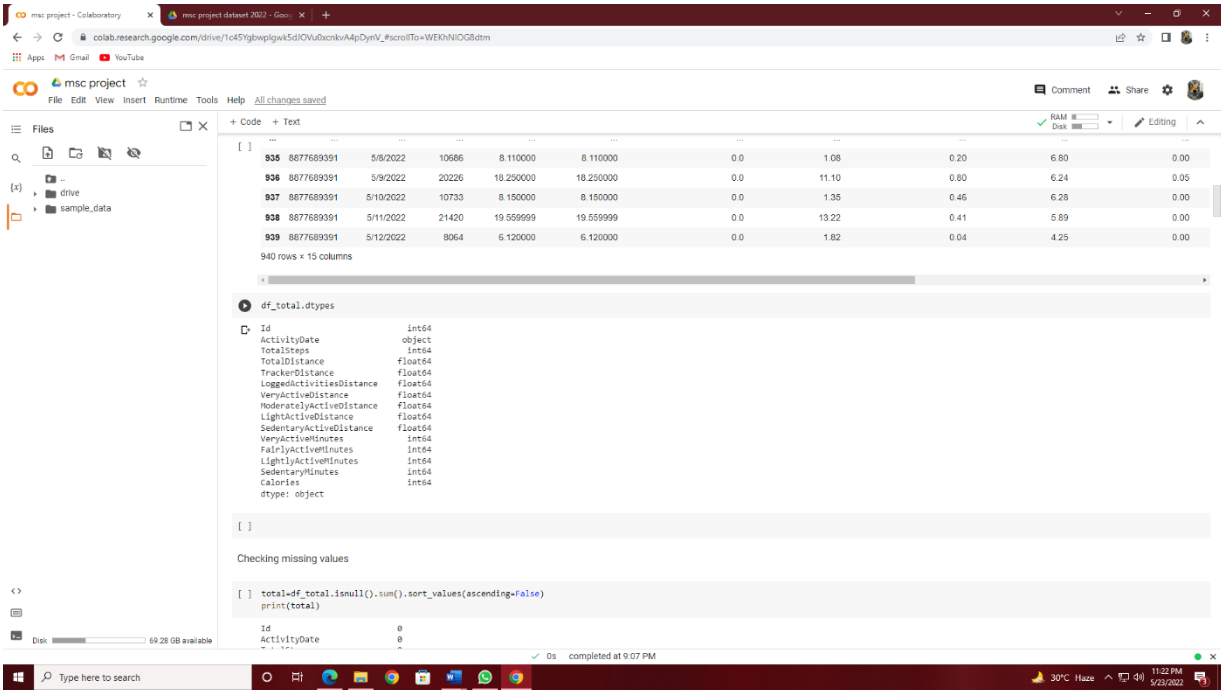
The results also reveal that males are more likely to be overweight and fall into the sedentary range than females are. Unlike figures do not correlate sleep and daily activity. Sundays the least active with walking being the most popular activity. Daily tracking of sleep efficiency has picked up that this rises during weekdays but declines at weekends despite a higher sleep activity during weekends.

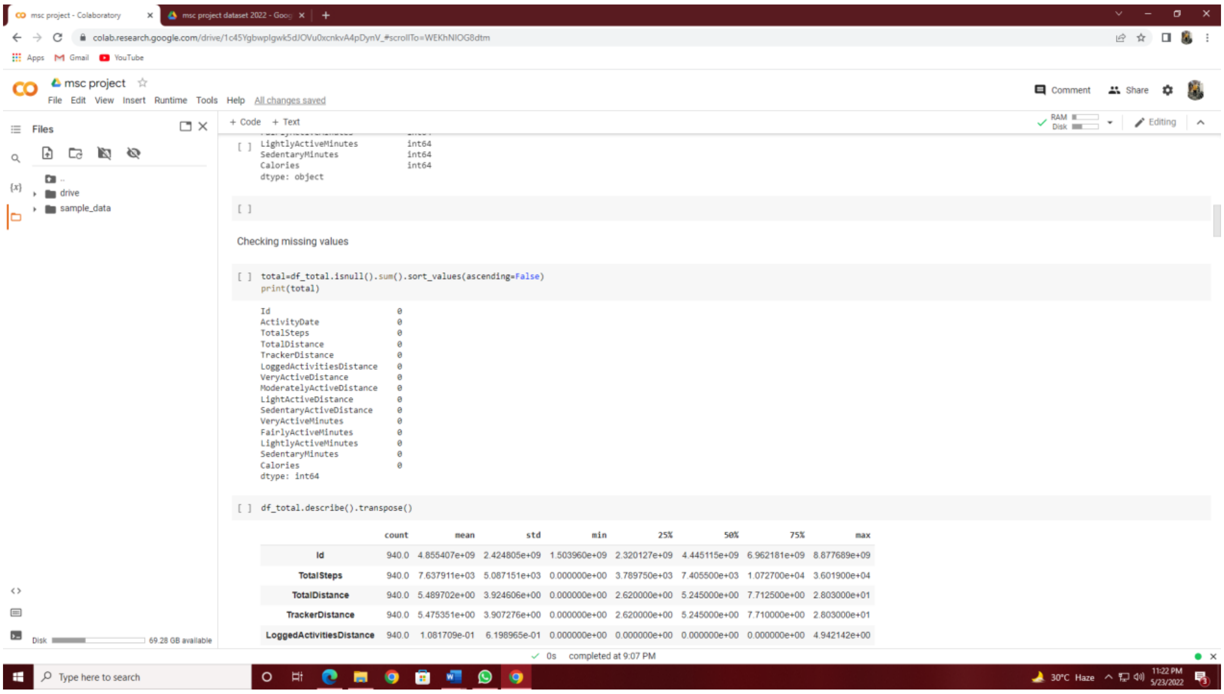
Future Enhancement

- There is clearly room for improvement in relation to physical activity and sleeping at weekends and the generated recommendation visualised .

- The wearer may also wish to consider as much evidence behind this recommendation as possible and may also wish to consider their performance in relation to that of their peers.

Program Code





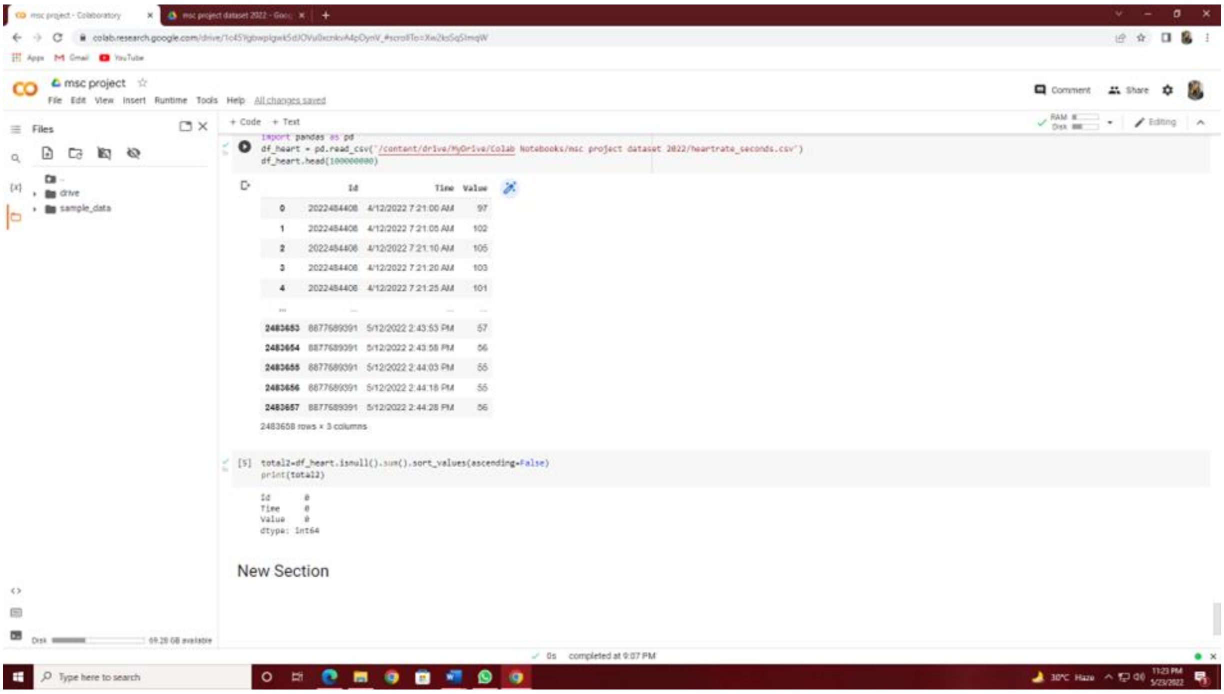
+ Code + Text

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp
import math
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
import tensorflow as tf
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras.layers import LSTM
```

```
[ ] from google.colab import drive
```

```
[ ] drive.mount("/content/gdrive")

Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
[ ] pd.read_csv('/content/gdrive/My Drive/Colab Notebooks/msc project dataset 2022/dailyactivitytotal.csv')
```



ALGORITHM

```
from numpy.random import rand
from numpy.random import choice
from numpy import asarray
from numpy import clip
from numpy import argmin
from numpy import min
from numpy import around
from matplotlib import pyplot
import numpy as np



def obj(x):
    return x[0]**2.0 + x[1]**2.0

    #sum = -20. * np.exp(-0.2 * np.sqrt(0.5 * (x[0] ** 2 + x[1] ** 2))) - np.exp(0.5 * (np.cos(2. * np.pi * x[0]) + np.cos(2. * np.pi * x[1]))) + 20. + np.e
    #return sum



def mutation(x, F):
    return x[0] + F * (x[1] - x[2])

def check_bounds(mutated, bounds):
    mutated_bound = [clip(mutated[i], bounds[i, 0], bounds[i, 1]) for i in range(len(bounds))]
    return mutated_bound



def crossover(mutated, target, dims, cr):

    p = rand(dims)
    trial = [mutated[i] if p[i] < cr else target[i] for i in range(dims)]
    return trial

def differential_evolution(pop_size, bounds, iter, F, cr):
    pop = bounds[:, 0] + (rand(pop_size, len(bounds)) * (bounds[:, 1] - bounds[:, 0]))
    obj_all = [obj(ind) for ind in pop]
```

\+ Code + Text

```
    best_obj = min(obj_all)
    prev_obj = best_obj
    obj_iter = list()
    for i in range(iter):
            for j in range(pop_size):
                candidates = [candidate for candidate in range(pop_size) if candidate != j]
                a, b, c = pop[choice(candidates, 3, replace=False)]
                mutated = mutation([a, b, c], F)
                mutated = check_bounds(mutated, bounds)
                trial = crossover(mutated, pop[j], len(bounds), cr)
                obj_target = obj(pop[j])
                obj_trial = obj(trial)
                if obj_trial < obj_target:
                    pop[j] = trial
                    obj_all[j] = obj_trial
            best_obj = min(obj_all)
            if best_obj < prev_obj:
             best_vector = pop[argmin(obj_all)]
             prev_obj = best_obj
             obj_iter.append(best_obj)
             print('Iteration: %d f([%s]) = %.5f' % (i, around(best_vector, decimals=5), best_obj))
    return [best_vector, best_obj, obj_iter]


pop_size = 10
bounds = asarray([(-5.0, 5.0), (-5.0, 5.0)])
iter = 100
F = 0.5
cr = 0.7

solution = differential_evolution(pop_size, bounds, iter, F, cr)
print('Done!')
print('\nSolution: f([%s]) = %.5f' % (around(solution[0], decimals=5), solution[1]))

pyplot.plot(solution[2], '.-')
pyplot.xlabel('Improvement Number')
pyplot.ylabel('Evaluation f(x)')
pyplot.show()
```

```
Iteration: 3 f([[0.23005 0.82428]]) = 0.73237
Iteration: 4 f([[0.26438 0.07944]]) = 0.07621
Iteration: 8 f([[ 0.20208 -0.06604]]) = 0.04520
Iteration: 10 f([[0.04366 0.03999]]) = 0.00351
Iteration: 11 f([[-0.05228  0.02766]]) = 0.00350
```

```
  return [c1, c2]

def mutation(bitstring, r_mut):
  for i in range(len(bitstring)):
        if rand() < r_mut:
          bitstring[i] = 1 - bitstring[i]

def genetic_algorithm(objective, bounds, n_bits, n_iter, n_pop, r_cross, r_mut):
  pop = [randint(0, 2, n_bits*len(bounds)).tolist() for _ in range(n_pop)]
  best, best_eval = 0, objective(decode(bounds, n_bits, pop[0]))
  for gen in range(n_iter):
    decoded = [decode(bounds, n_bits, p) for p in pop]
    scores = [objective(d) for d in decoded]
    for i in range(n_pop):
      if scores[i] < best_eval:
        best, best_eval = pop[i], scores[i]
        print(">%d, iteration f(%s) = %f" % (gen,  decoded[i], scores[i]))
    selected = [selection(pop, scores) for _ in range(n_pop)]
    children = list()
    for i in range(0, n_pop, 2):
      p1, p2 = selected[i], selected[i+1]
      for c in crossover(p1, p2, r_cross):
        mutation(c, r_mut)
        children.append(c)
      pop = children
  return [best, best_eval]

bounds = [[-5.0, 5.0], [-5.0, 5.0]]
n_iter = 100
n_bits = 16
n_pop = 100
r_cross = 0.9
r_mut = 1.0 / (float(n_bits) * len(bounds))
best, score = genetic_algorithm(objective, bounds, n_bits, n_iter, n_pop, r_cross, r_mut)
print('Done!')
decoded = decode(bounds, n_bits, best)
print('f(%s) = %f' % (decoded, score))
```

```
>0, iteration f([-1.5121459960937 5, -0.866851806640625]) = 3.038018
>0, iteration f([-0.364227294921875, 1.629638671875]) = 2.788384
>0, iteration f([0.101318359375, -1.022491455078125]) = 1.055754
>0, iteration f([-0.551605224609375, 0.3997802734375]) = 0.464093
>1, iteration f([-0.548858642578125, 0.059661865234375]) = 0.304805
```

```
df_total.dtypes
```

```
Id                            int64
ActivityDate                 object
TotalSteps                    int64
TotalDistance               float64
TrackerDistance             float64
LoggedActivitiesDistance    float64
VeryActiveDistance          float64
ModeratelyActiveDistance    float64
LightActiveDistance         float64
SedentaryActiveDistance     float64
VeryActiveMinutes             int64
FairlyActiveMinutes           int64
LightlyActiveMinutes          int64
SedentaryMinutes              int64
Calories                      int64
dtype: object
```

```
[ ]
```

Checking missing values

```
[ ] total=df_total.isnull().sum().sort_values(ascending=False)
    print(total)
```

```
[5] total2=df_heart.isnull().sum().sort_values(ascending=False)
    print(total2)
```

```
Id       0
Time     0
Value    0
dtype: int64
```

```
sns.countplot(df_total['TotalDistance']).set_title('distribution')
plt.ylim(0,10)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional ar
  FutureWarning
```


```
df_total.info()
```

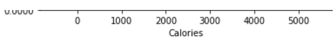
```
sns.countplot(df_total['Calories'])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
```

DISTPLOT OF **CALORIES**

```
fig1=sns.distplot(df_total['Calories'])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your
  warnings.warn(msg, FutureWarning)
```



ks
dataset 2022
vitytotal.csv

```
[80] data1=['Id','ActivityDate']
     ax=sns.boxplot(data=df_total[data1],orient='h')
```

```
corr = df_total.iloc[:,:-1].corr(method="pearson")
cmap = sns.diverging_palette(250,354,80,60,center='dark',as_cmap=True)
sns.heatmap(corr, vmax=1, vmin=-.5, cmap=cmap, square=True, linewidths=.2)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc63e95fe10>


```
Id = df_total['Id']
Ad = df_total['ActivityDate']
Ts = df_total['TotalSteps']
Td = df_total['TotalDistance']
Ca = df_total['Calories']

plt.plot(Id, Ad, label="ActivityDate")
plt.plot(Id, Ts, label="TotalSteps")
plt.plot(Id, Td, label="TotalDistance")
plt.plot(Id, Ca, label="Calories")


plt.legend()
plt.title('')
plt.ylabel('')
plt.xlabel('No of Id')

plt.savefig("plot.png")

plt.show()
```

0 5 10 15 20 25

```
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharey=True)
sns.histplot(df_total, ax=axes[0], x="Calories", kde=True, color='r')
sns.histplot(df_total, ax=axes[1], x="Id", kde=True, color='b')
sns.histplot(df_total, ax=axes[2], x="TotalDistance", kde=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc637a1e190>

1s completed at 5:51 PM

```
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharey=True)
sns.histplot(df_total, ax=axes[0], x="Calories", kde=True, color='r')
sns.histplot(df_total, ax=axes[1], x="TrackerDistance", kde=True, color='b')
sns.histplot(df_total, ax=axes[2], x="VeryActiveDistance", kde=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc63761df50>

+ Code + Text

```
#Show Bar Chart
plt.figure(figsize=(10,8))
sns.barplot(data=df_total, x='Calories', y='TotalSteps')
plt.title('per month chart', fontsize=18)
plt.xlabel ('Ids', fontsize=16)
plt.ylabel ('NO OF CALORIES/STEPS PER MONTH', fontsize=15)
plt.show()
```