SMARTWATCH ANALYSIS

Data Science With Python Lab Project Report

Bachelor

in

Computer Science

By

Data Dynamos

S190417

S190358

Rajiv Gandhi University Of Knowledge And Technologies

S.M. Puram , Srikakulam -532410

Andhra Pradesh, India

Abstract

Smartwatches have gained significant popularity as wearable devices that collect and monitor various types of data related to user activities, health, and well-being. Smartwatch analysis involves extracting insights from the collected data to provide valuable information and enhance user experiences.

There are many people in this present world who mostly care about their health.As you can't test your body everyday,there is a smart way to know how your body is actually working using technology.smart watches are preferred by people who like to take care of their fitness.Analysing the data gathered on fitness is one of the use cases of Data Science in healthcare.Wearing fitness trackers allows users to track their fitness levels through a variety of activities, such as daily steps or total daily distance travelled, hours of sleep and sleep stages, type of exercise and calories burned, heart rate monitoring, etc. This helps in predicting the health issues arising in a few days.

This project presents a comprehensive analysis of smartwatch data, aiming to predict calories burnt per day by each person.

Contents

Chapter 1

Introduction

1.1 Introduction to smart watch analysis:

In recent years, the advent of smartwatces has revolutionizied the way we track and monitor our fitness activities.Wearable technology is more common in today's digital world, with smart watches emerging as one of the most well-liked and versatile gadgets.

Smart watches have become a popular choice all over the world for improving health.As the smartwatches are attached to our hand ,these are called *"wearable devices"*.There are many other wearable devices but all are not equipped with sensors where as smartwatches are equipped with advanced sensors and algorithms, so as to record and keep track of a variety of aspects of our everyday life, including health and fitness metrics, sleep patterns, activity levels, etc.

This data helps data scientists to overview the present health of a person and can make a decision for future as a prevention(if necessary).The usage of smart watches for better health improvement is widespread.

People can predict their health using that information before a problem arises. As humans,we face a lot of health issues because of having no sufficient time to take care of our health and this leads to a major isssue in human body.Everytime we can't go to hospital to check our health,sometimes it is difficult to travel to far places.so the data collected from the watch can be used to track the user's health over time, giving them a better understanding of their health.

By understaing the data which is collected for a contigous days and by analyzing it,can play a significant role for health and well-being of a person.By observing the activity patterns in smartwatch fitness data,valuable data about each person's daily activities,working levels,etc can be revealed.This helps in informing the person to make changes in his/her daily activities if needed.

Smart watches not only show time but also offer the most current information about the body. Most people prefer using smart watches than the ordinary watches

because of this advantage.

Motto : The main motto of this project is "to explore and analyze smart watch fitness data using data science. "

1.2 APPLICATIONS:

The smartwatch analysis project using data science with Python has various applications across different domains. Some of the common applications include:

- *Health Monitoring* : Smartwatches collect data such as heart rate, sleep patterns, activity levels, and calories burned. By analyzing this data, we can monitor an individual's health, detect anomalies or irregularities, and provide personalized health recommendations.
- *Fitness Tracking* : Smartwatches track physical activities such as steps taken, distance covered, and active minutes. Data analysis can help in setting fitness goals, tracking progress, and providing insights for optimizing workouts and improving overall fitness.
- *Behavior Analysis* : By analyzing the data collected from smartwatches, we can gain

insights into users' behavior patterns. This includes understanding daily routines, activity preferences, and identifying factors that influence physical activity levels.

- *Sleep Analysis* : Smartwatches often include sleep tracking features. By analyzing sleep data, we can identify sleep patterns, measure sleep quality, and provide recommendations for improving sleep hygiene.

- *Performance Optimization* : For athletes and sports enthusiasts, smartwatch data analysis can be used to optimize performance. By analyzing metrics such as heart rate variability, training load, and recovery time, we can identify optimal training strategies, prevent overtraining, and enhance performance.

1.3 MOTIVATION TOWARDS PROJECT:

There are a number of reasons why a smartwatch analysis study utilising Python and data science should be undertaken.
Here are a few major drivers:

- Smartwatches are becoming more and more popular. These wearable electronics contain a variety of features and functionalities. Understanding user behaviour, health indicators, and performance through analysis of the data these devices capture can be quite beneficial.

- Machine learning and data science advancements: Python programming and data science approaches provide a potent toolkit for gaining insights from smartwatch data. With improvements in machine learning algorithms, we can find patterns in the data, anticipate the future, and comprehend it better.

- Personal Health and Fitness: Smartwatches give users access to real-time information on their fitness and health activities.

1.4 PROBLEM STATEMENT:

Problem Statement : The aim of this project is to analyze and extract valuable insights from smartwatch data using data science techniques with Python. The project will focus on exploring the collected data, understanding user behavior, and developing predictive models to improve health monitoring and fitness tracking.

The goal of this project is to thoroughly analyse smart watch data and derive valuable knowledge that can be applied to a variety of fields, including healthcare, wellness, and lifestyle optimisation. Our goal is to identify patterns, correlations, and trends in the data collected from smart watches that can help us better understand human physiology, enhance our personal health, and accomplish tasks more effectively every day.

By the project's conclusion, we hope to have a better knowledge of the potential uses for smart watch data analysis and how it might affect specific individuals.

Chapter 2

Approach To our Project

2.1 our project

The main motto of this project is "to explore and analyze smart watch fitness data using data science. "

Smartwatches have become increasingly popular wearable devices that provide a wide range of functionalities, including fitness tracking, health monitoring, and smartphone integration. These devices collect a wealth of data, such as heart rate, step count, sleep patterns, and more, which can be leveraged for insightful analysis.

Data science techniques in Python offer a powerful toolkit to extract valuable insights from smartwatch data. By applying data analysis and machine learning algorithms, we can uncover patterns, make predictions, and gain a deeper understanding of user behavior, health metrics, and performance.

By applying data science techniques and Python programming, we can unlock the potential of smartwatch data, uncover valuable insights, and develop predictive models for various applications, including health monitoring, fitness tracking, and user behavior analysis. Let's dive into the analysis and explore the fascinating world of smartwatch data using data science with Python.

The key steps involved in smartwatch analysis using data science with Python include:

- Data Preprocessing
- Exploratory Data Analysis (EDA)
- Feature Engineering
- Model Development
- Model Evaluation
- Predictive Analysis

2.2 Data Set

The dataset used in this project is publicly available in kaggle.There are 942 rows and 16 columns in the dataset. This dataset contain the following main atrributes:

- Id
- TotalSteps
- TotalDistance
- TrackerDistance
- VeryActiveDistance
- ModeratelyActiveDistance
- Calories

To see the whole dataset,we can use the following using python libraries:

Below is a code snippet from a Jupyter Notebook:(For IMPORTING and READING the file)

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```python
dataset = pd.read_csv('/home/rgukt/Desktop/dsp/dailyActivity_merged1.csv')
```

```python
dataset.head(5)
```

	snum	Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	VeryActiveDistance	ModeratelyActiveDistance	Light/
0	1	1503960366	04/12/16	13162	8.50	8.50	0.0	1.88	0.55	
1	2	1503960366	4/13/2016	10735	6.97	6.97	0.0	1.57	0.69	
2	3	1503960366	4/14/2016	10460	6.74	6.74	0.0	2.44	0.40	
3	4	1503960366	4/15/2016	9762	6.28	6.28	0.0	2.14	1.26	
4	5	1503960366	4/16/2016	12669	8.16	8.16	0.0	2.71	0.41	

Figure 2.1: Dataset

The dataframe contained null values in multiple columns.All those values are filled using dropna,fillna and interpolation.If there are null values there will be a chance of decrement in the accuracy at the end.So the dat should be cleaned well before processing the data.

2.3 Prediction technique

Due to a number of factors, RANDOM FOREST is a technique that is frequently used for smartwatch analytic projects to predict calories burned.

Accuracy: Random Forest models have a reputation for being highly accurate in making predictions. They can capture intricate correlations between the input features and the desired outcome, enabling precise forecasts of the number of calories burned depending on numerous variables including activity type, and other pertinent ones.

Random Forest is resistant to outliers and missing data, hence this statement is true. It reduces the possibility of biassed predictions by handling datasets with missing values and performing well even in the presence of outliers.

Multiple decision trees are combined in the ensemble learning method known as Random Forest. Multiple trees' predictions are combined, which lessens overfitting and strengthens the model's capacity to generalise, producing forecasts that can be trusted.

2.4 Visualization and Graphs

Commonly used graphing libraries in Python for data analysis include Matplotlib, Seaborn, Plotly, and ggplot. These libraries provide a wide range of graph types and customization options to create visually appealing and informative graphs.

Overall, graphs play a crucial role in data analysis by providing visual representations of data patterns and relationships, aiding in data exploration, model evaluation, and effective communication of findings.

```
label = ["Very Active Minutes", "Fairly Active Minutes",
         "Lightly Active Minutes", "Inactive Minutes"]
counts = data[["VeryActiveMinutes", "FairlyActiveMinutes",
               "LightlyActiveMinutes", "SedentaryMinutes"]].mean()
colors = ['gold','lightgreen', "pink", "blue"]

fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Total Active Minutes')
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=30,
                  marker=dict(colors=colors, line=dict(color='black', width=3)))
fig.show()
```

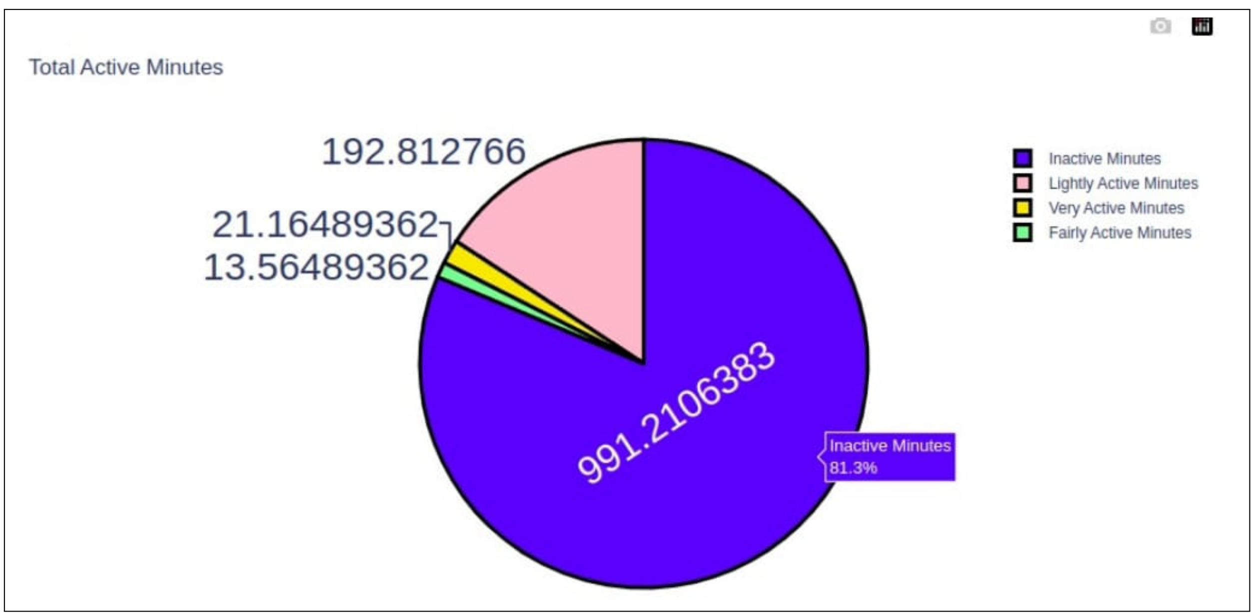


Figure 2.2: PIE CHART

Observations: 1) 81.3 % of Total inactive minutes in a day
2) 15.8 % of Lightly active minutes in a day

```
data.hist(figsize=(20,20))
```

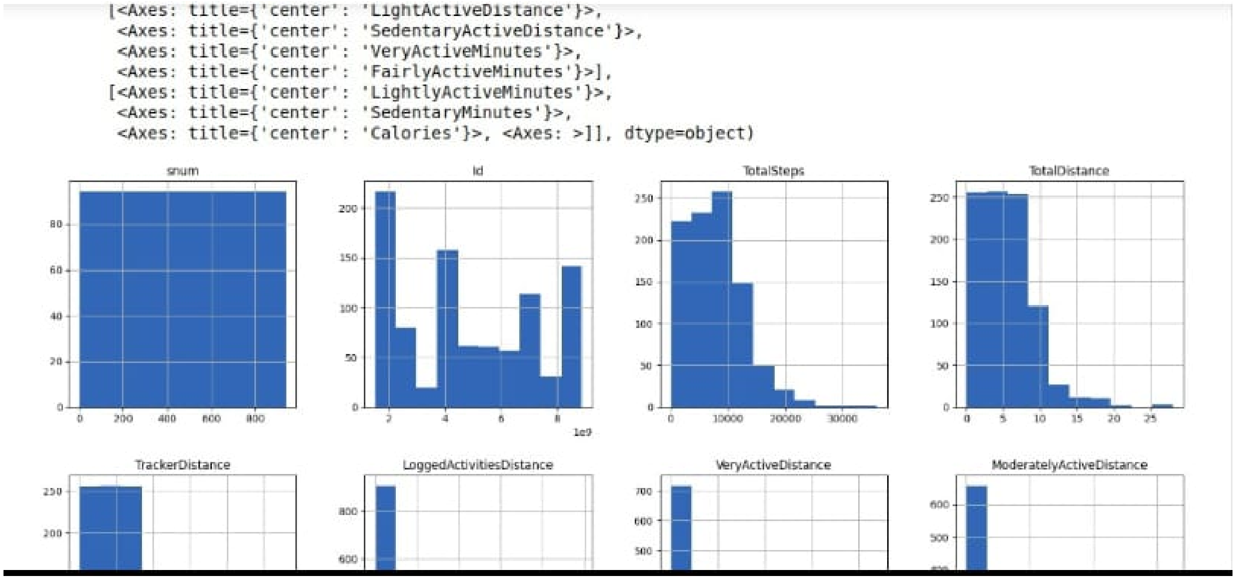


Figure 2.3: HISTOGRAM

```
1)

import seaborn as sns

sns.displot(data['Calories'],color='blue')
```

```
2)

a=data.loc[0:100,'Calories']

b=data.loc[0:100,'TotalSteps']

colors=data.loc[0:100,'Calories']

size=data.loc[0:100,'TotalSteps']

plt.scatter(a,b,c=colors,s=size,alpha=0.2,cmap='viridis')

plt.colorbar()
```

Below are the graphs for the respective codes of displot1 and scatterplot1

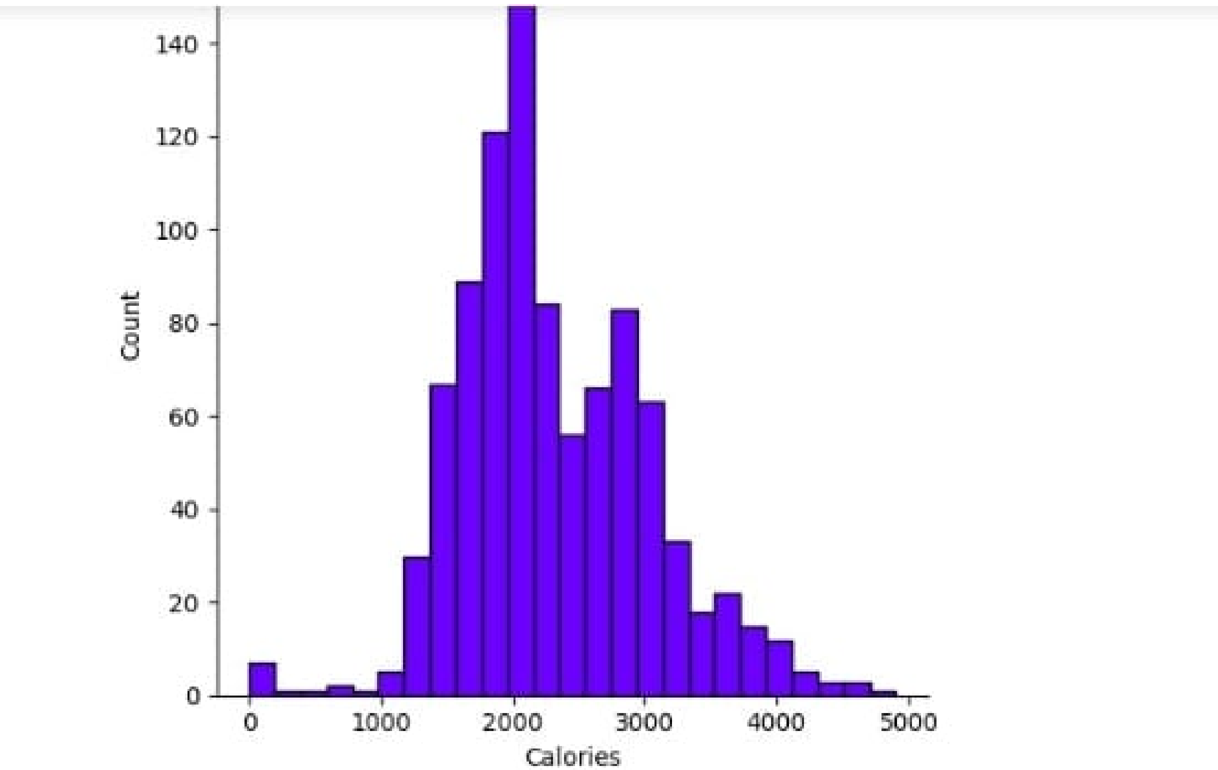


Figure 2.4: DISPLOT

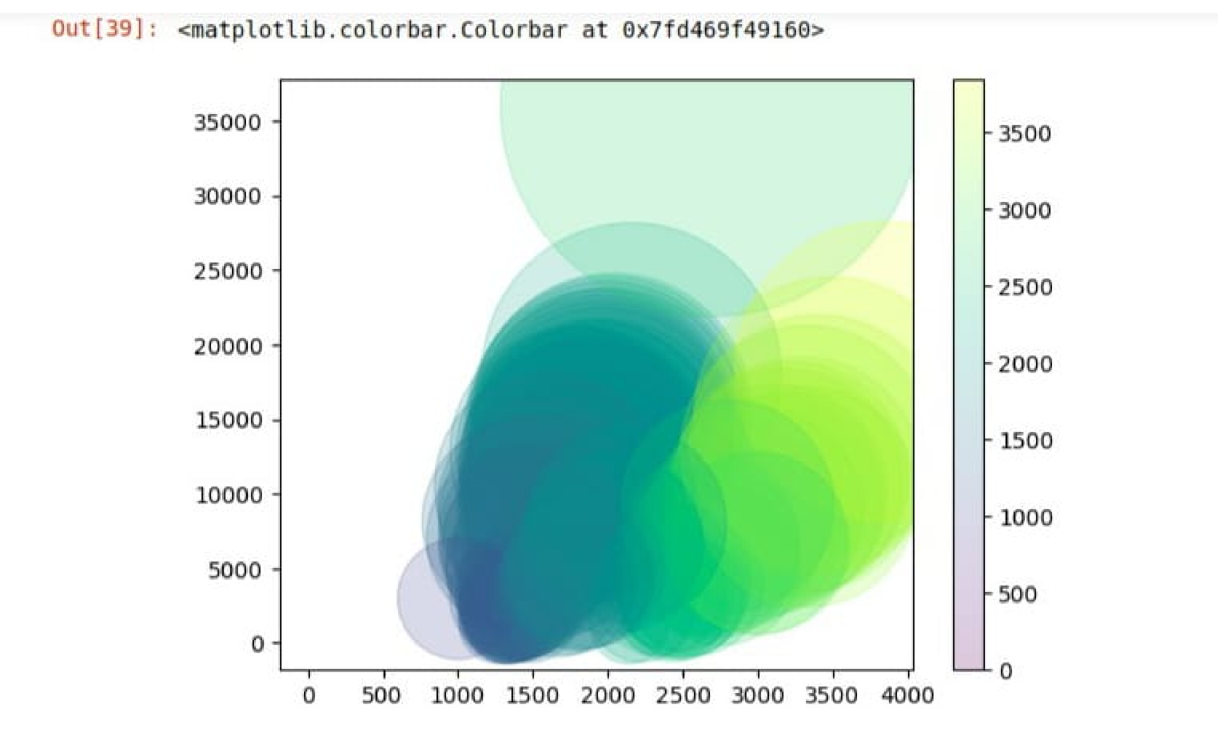


Figure 2.5: Scatterplot1

```
Out[40]: <matplotlib.colorbar.Colorbar at 0x7fd46a7a42b0>
```

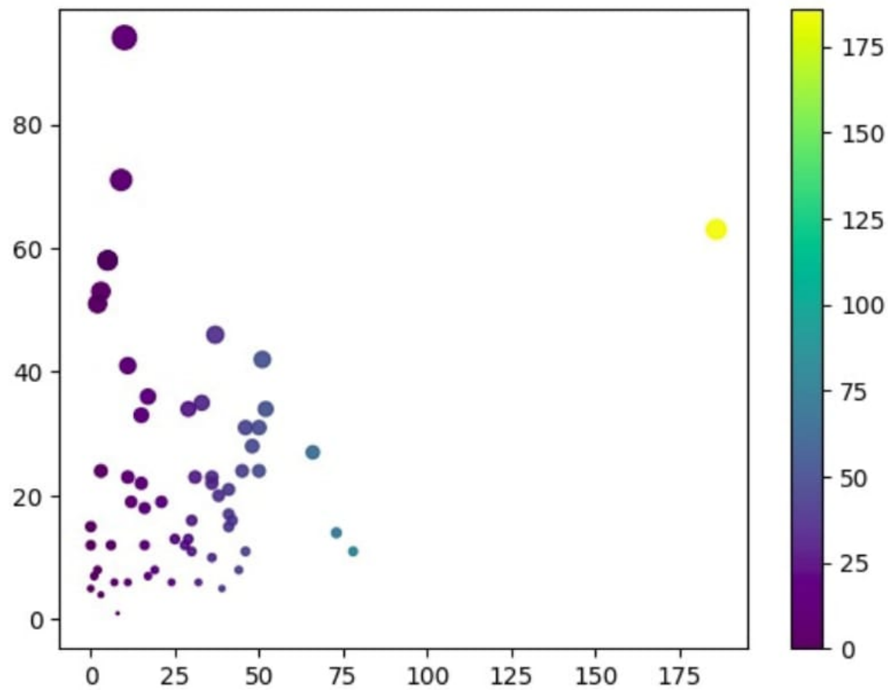


Figure 2.6: scatterplot2

3)

```
>> for this code above is the figure
a=data.loc[0:100,'VeryActiveMinutes']
b=data.loc[0:100,'FairlyActiveMinutes']
colors=data.loc[0:100,'VeryActiveMinutes']
size=data.loc[0:100,'FairlyActiveMinutes']
plt.scatter(a,b,c=colors,s=size,alpha=0.9,cmap='viridis')
plt.colorbar()
```

```
sns.scatterplot(x='Calories',y='TotalSteps',hue='VeryActiveMinutes',data=data,alpha=1)
```

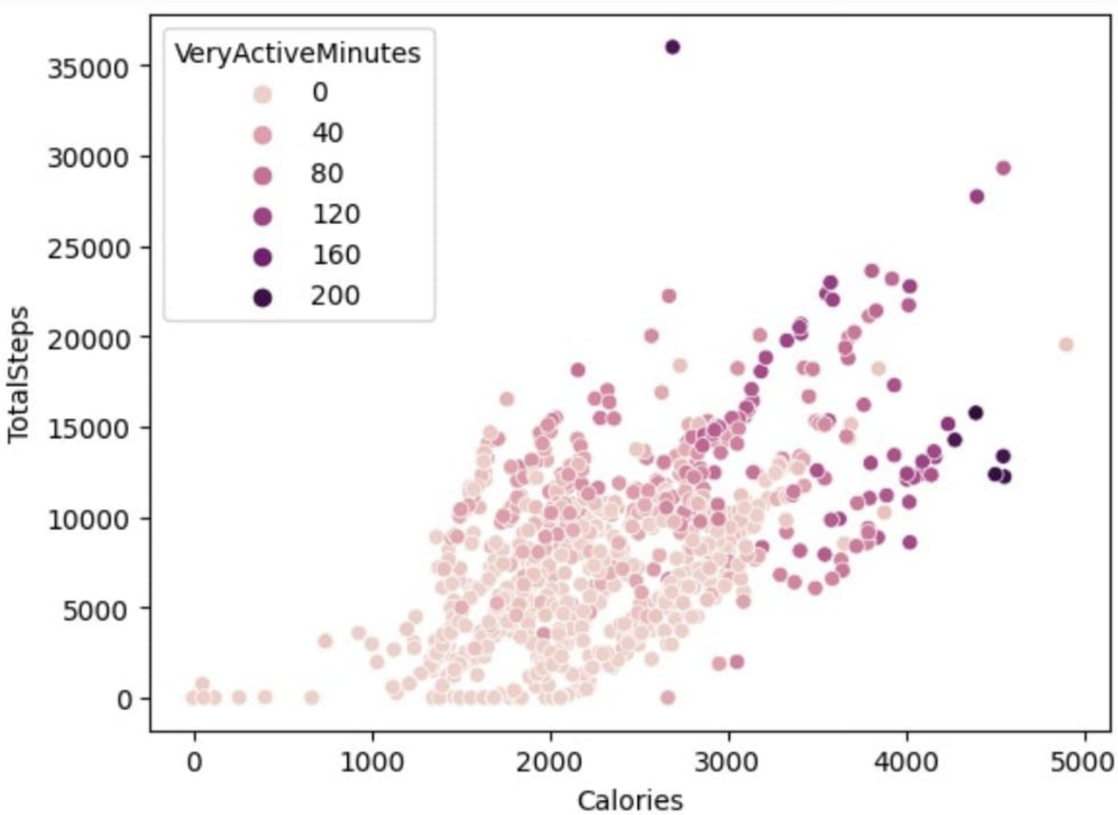


Figure 2.7: scatterplot3

```
figure = px.scatter(data_frame = data, x="Calories",
                    y="TotalSteps", size="VeryActiveMinutes",
                    trendline="ols",
                    title="Relationship between Calories & Total Steps")
figure.show()
```

The dataset has a "Calories" column; it contains the data about the number of calories burned in a day. Let's have a look at the relationship between calories burned and the total steps walked in a day

Figure 2.8: Relationship between calories and Totalsteps

Chapter 3

Code

3.1 Smartwatch analysis using python

Now the task is to import the necessary Python libraries and the dataset for smart watch analysis:

3.1.1 Import libraries of python

```
#import python libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
```

3.1.2 Load dataset

Now load the dataset called dailyActiviymerged.csv

```
data = pd.read_csv("dailyActivity_merged.csv")
print(data.head())
```

```
           Id ActivityDate  TotalSteps  TotalDistance  TrackerDistance  \
0  1503960366    4/12/2016       92341           8.50             8.50
1  1503960366    4/13/2016       10735           6.97             6.97
2  1503960366    4/14/2016       10460           6.74             6.74
3  1503960366    4/15/2016        9762           6.28             6.28
4  1503960366    4/16/2016       12669           8.16             8.16

   LoggedActivitiesDistance  VeryActiveDistance  ModeratelyActiveDistance  \
0                       0.0                1.88                      0.55
1                       0.0                1.57                      0.69
2                       0.0                2.44                      0.40
3                       0.0                2.14                      1.26
4                       0.0                2.71                      0.41

   LightActiveDistance  SedentaryActiveDistance  VeryActiveMinutes  \
0                 6.06                      0.0                 25
1                 4.71                      0.0                 21
2                 3.91                      0.0                 30
3                 2.83                      0.0                 29
4                 5.04                      0.0                 36

   FairlyActiveMinutes  LightlyActiveMinutes  SedentaryMinutes  Calories
0                   13                   328               728      1985
1                   19                   217               776      1797
2                   11                   181              1218      1776
3                   34                   209               726      1745
4                   10                   221               773      1863
```

3.1.3 Data Preprocessing

let's have a look at whether this dataset has any null values or not:

```
print(data.isnull().sum())
```

```
Out[8]: 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
```

So the dataset does not have any null values. Let's have a look at the information about columns in the dataset:

```
print(data.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
```

you will see information about very active, moderately active, lightly active, and sedentary minutes in the dataset.

Let's combine all these columns as total minutes:

```
data["TotalMinutes"] = data["VeryActiveMinutes"] + data["FairlyActiveMinutes"] + data[

print(data["TotalMinutes"].sample(5))
```

Out[10]:

Distance	LightActiveDistance	SedentaryActiveDistance	VeryActiveMinutes	FairlyActiveMinutes	LightlyActiveMinutes	SedentaryMinutes	Calories	TotalMinutes
0.55	6.06	0.0	25	13	328	728	1985	1094
0.69	4.71	0.0	21	19	217	776	1797	1033
0.40	3.91	0.0	30	11	181	1218	1776	1440
1.26	2.83	0.0	29	34	209	726	1745	998
0.41	5.04	0.0	36	10	221	773	1863	1040

```
data.describe()
```

Out[12]:

	Id	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	VeryActiveDistance	ModeratelyActiveDistance	LightActiveDistanc
count	9.400000e+02	940.000000	940.000000	940.000000	940.000000	940.000000	940.000000	940.0000
mean	4.855407e+09	7722.143617	5.489702	5.475351	0.108171	1.502681	0.567543	3.34081
std	2.424805e+09	5786.207547	3.924606	3.907276	0.619897	2.658941	0.883580	2.0406
min	1.503960e+09	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	2.320127e+09	3789.750000	2.620000	2.620000	0.000000	0.000000	0.000000	1.9450
50%	4.445115e+09	7405.500000	5.245000	5.245000	0.000000	0.210000	0.240000	3.3650
75%	6.962181e+09	10727.000000	7.712500	7.710000	0.000000	2.052500	0.800000	4.7825
max	8.877689e+09	92341.000000	28.030001	28.030001	4.942142	21.920000	6.480000	10.7100

```
data.duplicated().value counts()
```

output :

False 940

Name:count, dtype: int64

It shows that our dataset doesnt contain any duplicate values.

3.2 MODEL TRAINING:

Here,we use train-test-split using python libraries and split the data into train data and test data..As we imported the libraries above we directly go to step-2 ie.,splitting

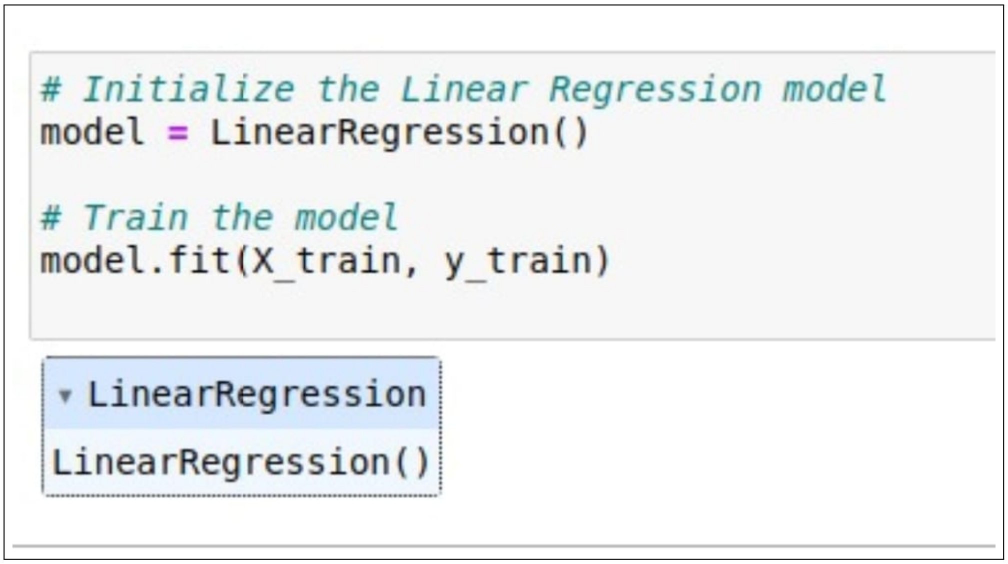
3.2.1 Splitting the Dataset

```
dataset = dataset.drop('ActivityDate', axis=1)
X=dataset.drop(['Calories'],axis=1).values
y=dataset['Calories'].values

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In the above code we splitted the attributes.
Here,
X=all the attributes based on which the calories are predicted
y=actual calorie values

3.2.2 Using Linear Regression:

```
# Initialize the Linear Regression model

model = LinearRegression()

# Train the model

model.fit(X_train, y_train)
```



```
# Initialize the Linear Regression model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)
```


3.2.3 Using Random Forest:

```
# Initialize the Random Forest regressor

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)


# Fit the model to the training data

rf_model.fit(X_train, y_train)
```

3.3 MODEL EVALUATION:

In REGRESSION we calculate MEAN SQUARED ERROR(MSE)

```
# Initialize the Random Forest regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Fit the model to the training data
rf_model.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

3.3.1 MSE using Linear Regression:

Mean squared error (MSE) is a measure of how close the predicted values of a regression model are to the actual values. It is calculated by taking the average of the squared residuals, which are the differences between the predicted and actual values. A lower MSE indicates that the model is more accurate.

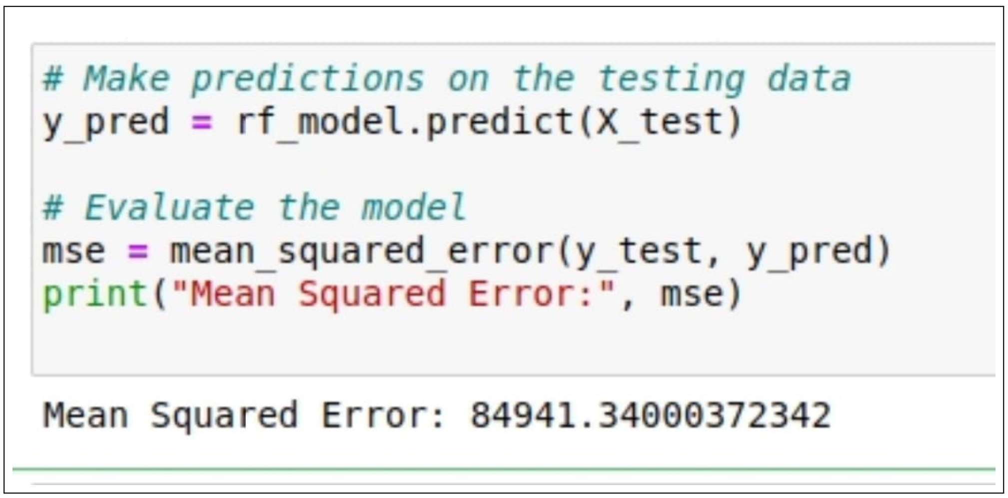
```
# Make predictions on the test set

y_pred = model.predict(X_test)


# Evaluate the model using mean squared error

mse = mean_squared_error(y_test, y_pred)

print('Mean Squared Error:', mse)
```

```
# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model using mean squared error
mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error:', mse)
```

```
Mean Squared Error: 135438.53811284926
```

3.3.2 MSE using Random Forest:

```
# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model using mean squared error
mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error:', mse)
```



```
# Make predictions on the testing data
y_pred = rf_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

Mean Squared Error: 84941.34000372342
```

3.3.3 Prediction:

Overall, the prediction of calories in smartwatch analysis using data science and Python empowers individuals with personalized insights, supports their fitness goals, and contributes to scientific understanding in the field of health and wellness. Model prediction is done as following:

```
y_pred=model.predict(X_test)
print(y_pred)
```

```
3]: y_pred=model.predict(X_test)
    print(y_pred)
```

```
[2782.80957264 2183.82350734 1588.80380078 2911.30800227 2988.91989122
 1960.8073841  2201.99646647 2177.05421915 1782.98109449 2736.13255027
 2210.15638679 1954.21122728 1692.62995479 1921.26865763 2322.88311837
 2352.89330284 1573.92128746 2041.07488855 2054.92199846 1996.68514698
 2082.08778102 2409.29862191 1842.12550619 2297.52129283 2209.9739993
 1987.30469492 2545.38802665 1602.43485101 1839.96747846 2157.47834415
 2062.86159193 2268.86415    2385.48218363 1701.32937051 1693.90754515
 2337.53460378 2925.71302301 1776.35939073 2500.09348243 1898.75180381
 2778.9457151  2557.50227284 2015.55115764 2948.39491608 3929.10391932
 1605.93138538 2172.9481476  1766.6449747  2545.57864478 2283.55921474
 1659.2937052  1776.88502799 2414.71003113 2620.31653271 2142.63632676
 3280.18697658 3195.33555551 1814.22834439 1964.01528216 2310.62979023
 2427.30037366 2891.83949288 1741.92400919 2320.42489208 2098.64273191
 1848.41490153 1987.94875649 2325.64087142 2247.74470155 2094.40768676
 2246.42606191 3952.5682866  2685.91539162 2304.53983917 3100.00336548
 2911.38564916 1860.75121299 2034.55556172 1587.31057388 1692.32625293
 3758.14302319 1963.13431124 3176.63627805 1917.702585   2421.75749602
 2927.01560111 1718.89024841 1931.86928356 2252.54489913 2516.53078508
 2645.81364849 1965.5821036  1961.5320885  2248.73359054 2108.08467984
 1804.18388248 1915.45453672 2117.64226338 2892.21367065 1738.73329029
 2225.4458808  2208.07600946 1758.98059435 2167.3219315  2658.99800755
 2556.5344016  1900.42527635 2186.94666463 3259.39552044 2753.93008412
 2136.79505468 2051.1580548  2170.5257842  2566.23978951 1539.1026766
 2456.85497564 2353.63314703 2893.17512506 2899.24171957 2976.82522658
```

3.4 ACCURACY:

As we are using the REGRESSION Technique,to calculate ACCURACY r2.score is used. r2.score is defined as:

R-squared is a measure of how much of the variation in the dependent variable is explained by the independent variables. A higher R-squared indicates that the model is more accurate.

3.4.1 Using Linear Regression:

```
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error


from sklearn.metrics import r2_score

r2_score(y_test,y_pred)
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

```
0.7134630867794772
```

3.4.2 Using Random Forest:

```
from sklearn.metrics import r2_score

r2_score(y_test,y_pred)
```

```
from sklearn.metrics import r2_score

# Calculate the R-squared score
r2 = r2_score(y_test, y_pred)
print("R-squared:", r2)
```

```
R-squared: 0.830698182753424
```

SO USING RANDOM FOREST IS BETTER THAN LINEAR REGRESSION TO OUR PROBLEM i.e.,Predicting the Claories Burnt using remaining activities.

Chapter 4

Conclusion and Future Work

4.1 conclusion

In this project, we utilized data science techniques in Python to analyze smartwatch data and predict calories burned using the Random Forest technique. The Random Forest model proved to be highly effective and provided valuable insights into energy expenditure estimation.

The versatility of Random Forest allows for its application beyond calorie burn prediction. It can be adapted for other smartwatch analysis tasks such as activity recognition, sleep quality assessment, and heart rate zone classification.

In conclusion, the implementation of the Random Forest technique in the smartwatch analysis project using data science with Python showcased its effectiveness in accurate calorie estimation. The obtained insights and predictions can empower individuals to make informed decisions about their health and fitness routines, leading to improved well-being and achieving their fitness goals.

Future enhancements to the project could involve exploring additional features, such as sleep patterns or nutrition data, to improve prediction accuracy and broaden the scope of analysis. Additionally, the model's performance could be further optimized by fine-tuning hyperparameters or exploring other advanced machine learning techniques.

Overall, this project highlights the potential of data science and the Random Forest technique in leveraging smartwatch data to provide valuable insights and improve health and fitness tracking.