REPORT ON BELLABEAT

CASE STUDY: GOOGLE DATA ANALYTICS

By: KARINA VARMA

About Bellabeat:

Urška Sršen and Sando Mur founded Bellabeat, a high-tech company that manufactures health-focused smart products. Sršen used her background as an artist to develop beautifully designed technology that informs and inspires women around the world. Collecting data on activity, sleep, stress, and reproductive health has allowed Bellabeat to empower women with knowledge about their own health and habits. Since it was founded in 2013, Bellabeat has grown rapidly and quickly positioned itself as a tech-driven wellness company for women.

Introduction:

The purpose of this case study is to analyze non-bellabeat smart products to evaluate what features do users use the most and how we can bring Bellabeat forward by competing with other smart products and become favorable to the customers.

Problem:

As technology is growing everyday, many companies are coming forward with their smart products and Bellabeat is one of them. Now, to make customers purchase Bellabeat products is not an easy task, because other companies would be producing the same type of gadgets, providing similar technology and features, but maybe their cost would be lesser, so people are more attracted towards those products. Teens studying in middle and high school, would not be purchasing smart products that are way too costly. So keeping all the scenarios in mind, Bellabeat has to build something different but affordable in comparison to other market products and attract users.

Business Task:

As smart products are a big part of everyday life, Bellabeat needs trends of usage of these products. By analyzing trends and competitions through thorough analysis, Bellabeat can make data driver decisions in order to engage customers and create more opportunities for its own growth.

Data Sources:

The data used in this case-study is taken from Kaggle which contains Fitbit Fitness Tracker Data, and it was made available for public usage by Mobius. The dataset was generated by respondents to a distributed survey via Amazon Mechanical Turk between 03.12.2016-05.12.2016.

Data Cleaning and Manipulation

In [4]:

```
import pandas as pd
import numpy as np
import datetime as dt
```

pip install matplotlib

In [5]:

```
import matplotlib.pyplot as plt
```

In [27]:

```
pip install seaborn
```

```
Collecting seaborn
  Using cached seaborn-0.11.2-py3-none-any.whl (292 kB)
Requirement already satisfied: scipy>=1.0 in c:\users\homecomputer\appdata\loca
l\programs\python\python310\lib\site-packages (from seaborn) (1.8.0)
Requirement already satisfied: pandas>=0.23 in c:\users\homecomputer\appdata\lo
cal\programs\python\python310\lib\site-packages (from seaborn) (1.4.0)
Requirement already satisfied: matplotlib>=2.2 in c:\users\homecomputer\appdata
\local\programs\python\python310\lib\site-packages (from seaborn) (3.5.2)
Requirement already satisfied: numpy>=1.15 in c:\users\homecomputer\appdata\loc
al\programs\python\python310\lib\site-packages (from seaborn) (1.22.2)
Requirement already satisfied: packaging>=20.0 in c:\users\homecomputer\appdata
\local\programs\python\python310\lib\site-packages (from matplotlib>=2.2->seabo
rn) (21.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users\homecomputer\appdata\l
ocal\programs\python\python310\lib\site-packages (from matplotlib>=2.2->seabor
n) (9.2.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\homecomputer\appda
ta\local\programs\python\python310\lib\site-packages (from matplotlib>=2.2->sea
born) (1.4.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\homecomputer\ap
pdata\local\programs\python\python310\lib\site-packages (from matplotlib>=2.2->
seaborn) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\homecomputer\appda
ta\local\programs\python\python310\lib\site-packages (from matplotlib>=2.2->sea
born) (4.34.4)
Requirement already satisfied: cycler>=0.10 in c:\users\homecomputer\appdata\lo
cal\programs\python\python310\lib\site-packages (from matplotlib>=2.2->seaborn)
(0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\homecomputer\appdat
a\local\programs\python\python310\lib\site-packages (from matplotlib>=2.2->seab
orn) (3.0.9)
Requirement already satisfied: pytz>=2020.1 in c:\users\homecomputer\appdata\lo
cal\programs\python\python310\lib\site-packages (from pandas>=0.23->seaborn) (2
021.3)
Requirement already satisfied: six>=1.5 in c:\users\homecomputer\appdata\local
\programs\python\python310\lib\site-packages (from python-dateutil>=2.7->matplo
tlib>=2.2->seaborn) (1.16.0)
Installing collected packages: seaborn
Successfully installed seaborn-0.11.2
Note: you may need to restart the kernel to use updated packages.
```

In [28]:

```
import seaborn as sns
```

In [6]:

```
daily_Activity = pd.read_csv("dailyActivity_merged.csv")
```

In [19]:

```
daily_Activity.head(10)
```

Out[19]:

	Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	V
0	1503960366	4/12/2016	13162	8.50	8.50	0.0	
1	1503960366	4/13/2016	10735	6.97	6.97	0.0	
2	1503960366	4/14/2016	10460	6.74	6.74	0.0	
3	1503960366	4/15/2016	9762	6.28	6.28	0.0	
4	1503960366	4/16/2016	12669	8.16	8.16	0.0	
5	1503960366	4/17/2016	9705	6.48	6.48	0.0	
6	1503960366	4/18/2016	13019	8.59	8.59	0.0	
7	1503960366	4/19/2016	15506	9.88	9.88	0.0	
8	1503960366	4/20/2016	10544	6.68	6.68	0.0	
9	1503960366	4/21/2016	9819	6.34	6.34	0.0	

In [11]:

```
daily_Activity.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
```

In [12]:
```
daily_Activity.describe()
```

Out[12]:

	Id	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	VeryA
count	9.400000e+02	940.000000	940.000000	940.000000	940.000000	
mean	4.855407e+09	7637.910638	5.489702	5.475351	0.108171	
std	2.424805e+09	5087.150742	3.924606	3.907276	0.619897	
min	1.503960e+09	0.000000	0.000000	0.000000	0.000000	
25%	2.320127e+09	3789.750000	2.620000	2.620000	0.000000	
50%	4.445115e+09	7405.500000	5.245000	5.245000	0.000000	
75%	6.962181e+09	10727.000000	7.712500	7.710000	0.000000	
max	8.877689e+09	36019.000000	28.030001	28.030001	4.942142	

In [13]:
```
daily_Activity.isnull().sum()
```

Out[13]:
```
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
```

In [17]:
```
unique_id = len(pd.unique(daily_Activity["Id"]))
print(unique_id)
```

```
33
```

In [18]:
```
daily_Activity.shape
```

Out[18]:
```
(940, 15)
```

From performing above basic functions, we found out basic information like:

1. No. of rows and columns
2. Unique id count
3. sum of null values
4. the type of data

Now we will create a seperate column for having day of the week in our data table

In [8]:
```
daily_Activity["ActivityDate"] = pd.to_datetime(daily_Activity["ActivityDate"], f
```

In [9]:
```
daily_Activity["ActivityDate"].head()
```

Out[9]:
```
0   2016-04-12
1   2016-04-13
2   2016-04-14
3   2016-04-15
4   2016-04-16
Name: ActivityDate, dtype: datetime64[ns]
```

In [10]:
```
daily_Activity["day_of_the_week"] = daily_Activity["ActivityDate"].dt.day_name()
```

In [11]:
```
daily_Activity.head()
```

Out[11]:

	Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	V
0	1503960366	2016-04-12	13162	8.50	8.50	0.0	
1	1503960366	2016-04-13	10735	6.97	6.97	0.0	
2	1503960366	2016-04-14	10460	6.74	6.74	0.0	
3	1503960366	2016-04-15	9762	6.28	6.28	0.0	
4	1503960366	2016-04-16	12669	8.16	8.16	0.0	

In [12]:
```
daily_Activity.columns.values
```

Out[12]:
```
array(['Id', 'ActivityDate', 'TotalSteps', 'TotalDistance',
       'TrackerDistance', 'LoggedActivitiesDistance',
       'VeryActiveDistance', 'ModeratelyActiveDistance',
       'LightActiveDistance', 'SedentaryActiveDistance',
       'VeryActiveMinutes', 'FairlyActiveMinutes', 'LightlyActiveMinutes',
       'SedentaryMinutes', 'Calories', 'day_of_the_week'], dtype=object)
```

In [13]:
```
new_columns = ['Id', 'ActivityDate','day_of_the_week','TotalSteps', 'TotalDistanc
       'TrackerDistance', 'LoggedActivitiesDistance',
       'VeryActiveDistance', 'ModeratelyActiveDistance',
       'LightActiveDistance', 'SedentaryActiveDistance',
       'VeryActiveMinutes', 'FairlyActiveMinutes', 'LightlyActiveMinutes',
       'SedentaryMinutes', 'Calories']

daily_Activity = daily_Activity.reindex(columns = new_columns)
```

In [14]:
```
daily_Activity
```

Out[14]:

	Id	ActivityDate	day_of_the_week	TotalSteps	TotalDistance	TrackerDistance	Logge
0	1503960366	2016-04-12	Tuesday	13162	8.500000	8.500000	
1	1503960366	2016-04-13	Wednesday	10735	6.970000	6.970000	
2	1503960366	2016-04-14	Thursday	10460	6.740000	6.740000	
3	1503960366	2016-04-15	Friday	9762	6.280000	6.280000	
4	1503960366	2016-04-16	Saturday	12669	8.160000	8.160000	
...	...	...	...	...	...	...	
935	8877689391	2016-05-08	Sunday	10686	8.110000	8.110000	
936	8877689391	2016-05-09	Monday	20226	18.250000	18.250000	
937	8877689391	2016-05-10	Tuesday	10733	8.150000	8.150000	
938	8877689391	2016-05-11	Wednesday	21420	19.559999	19.559999	
939	8877689391	2016-05-12	Thursday	8064	6.120000	6.120000	

940 rows × 16 columns

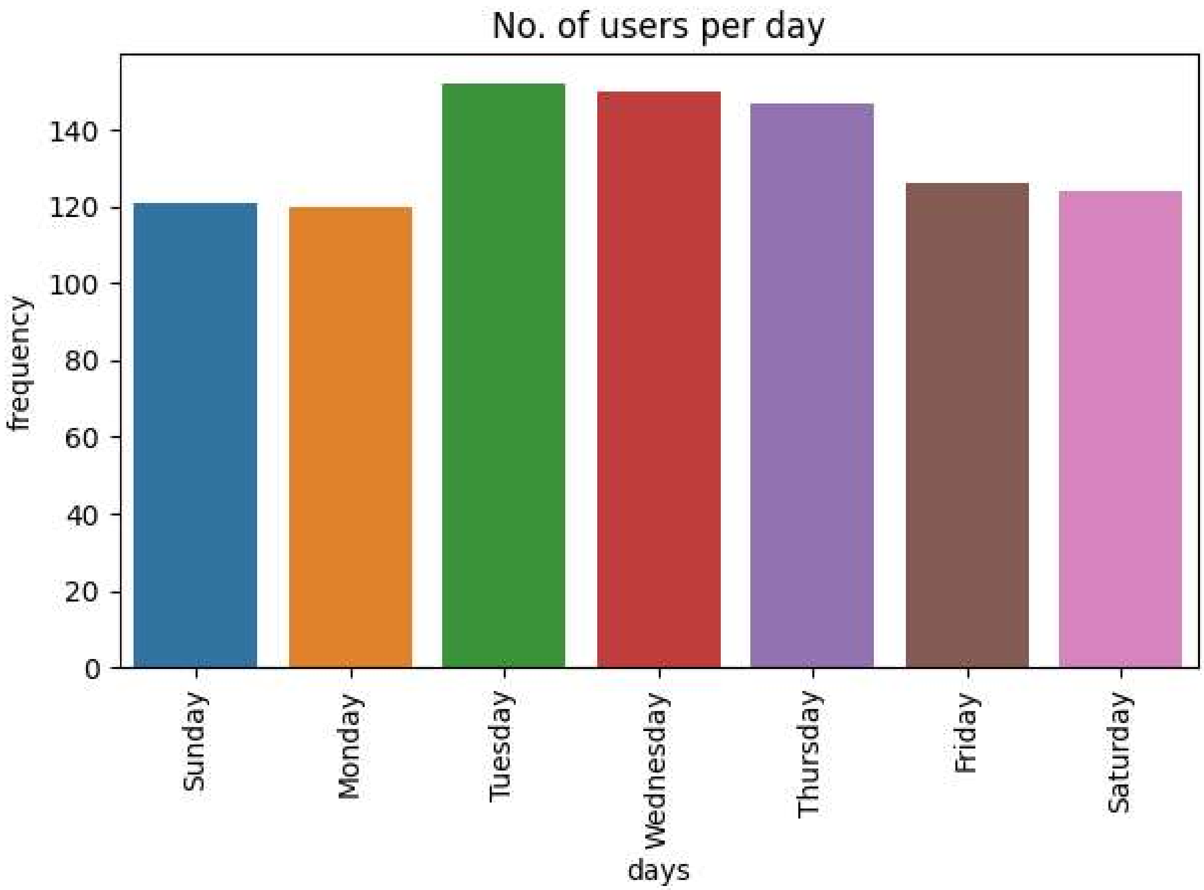
In [32]:
```
pd.unique(daily_Activity["day_of_the_week"])
```

Out[32]:
```
array(['Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday',
       'Monday'], dtype=object)
```

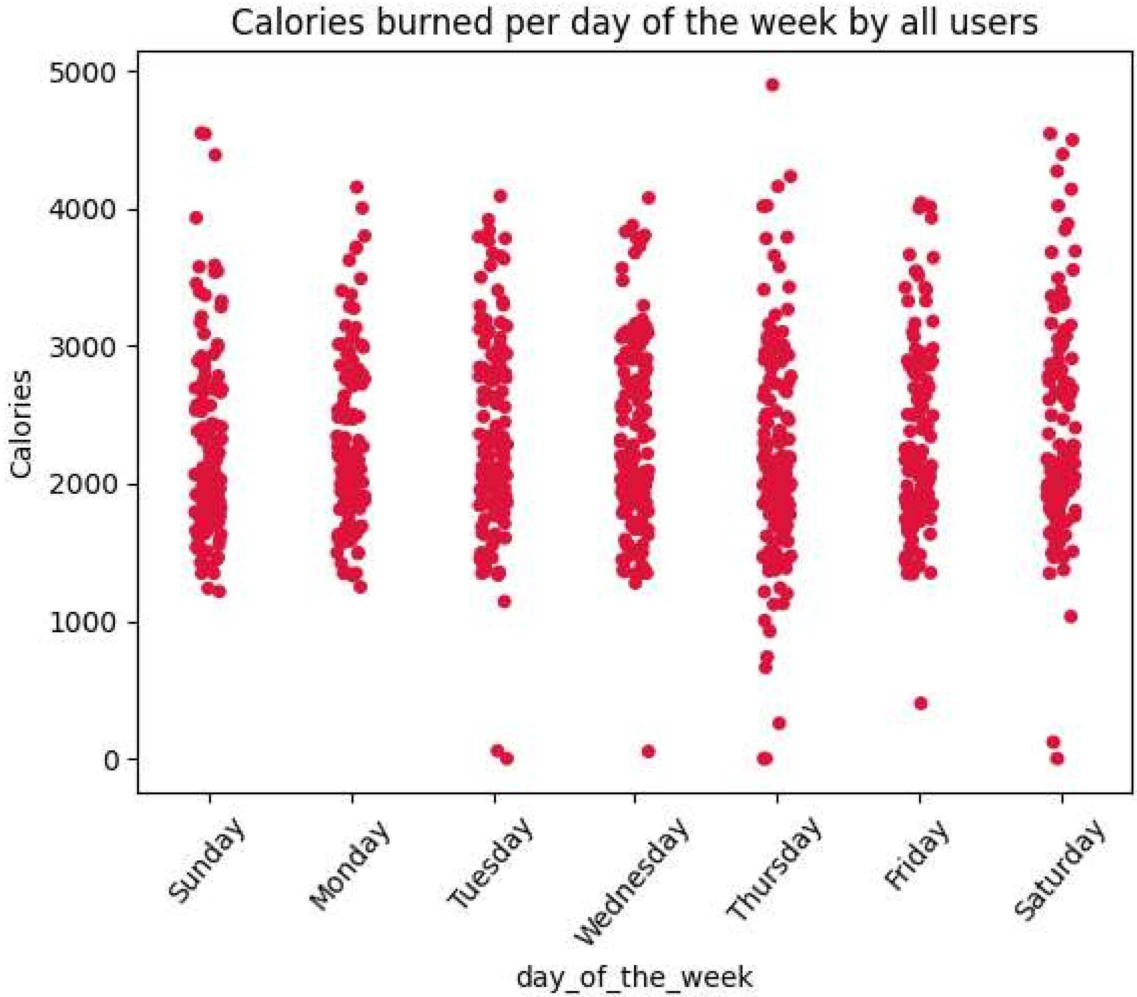
No. of users logged into app everyday

In [44]:

```
plt.style.use("default")
plt.figure(figsize=(7,4)) # specify size of the chart
sns.countplot(x="day_of_the_week", data = daily_Activity,
              order = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "F
plt.xlabel("days")
plt.xticks(rotation = 90)
plt.ylabel("frequency")
plt.title("No. of users per day")
plt.show()
```



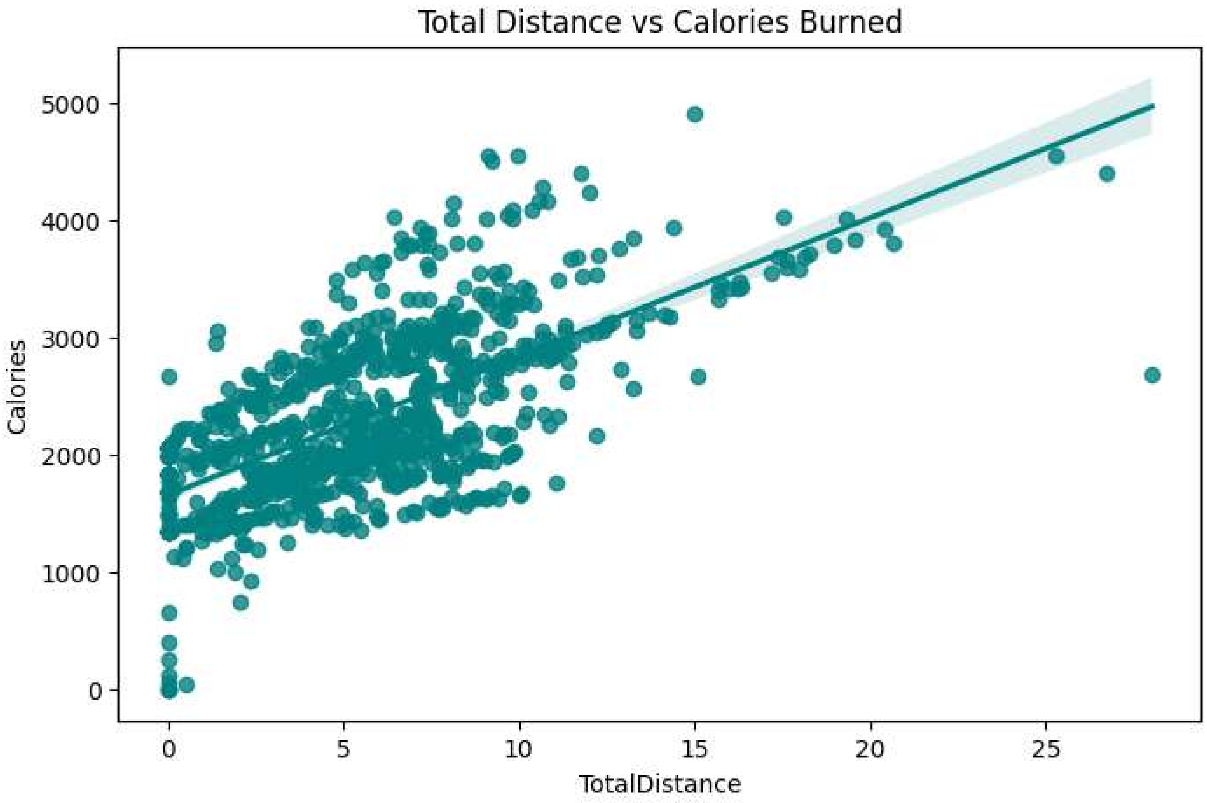
In [61]:
```
sns.stripplot(x="day_of_the_week", y = "Calories", data = daily_Activity, color =
              order = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "F
plt.xticks(rotation = 50)
plt.title("Calories burned per day of the week by all users")
plt.show()
```



In the above visualization, we have used stripplot instead of scatterplot, because in scatterplot we cannot arrange the order of the days.

In [74]:
```
plt.figure(figsize=(8,5))
sns.regplot (x="TotalDistance",y="Calories",data=daily_Activity, color = "teal")
plt.title("Total Distance vs Calories Burned")
```

Out[74]:
```
Text(0.5, 1.0, 'Total Distance vs Calories Burned')
```



From the above visualization we can say that as total distance increased, number of calories burned by people also increased.

Through data I understand where our healthy customers might miss lots of knowlegeable insights, which can translate into business opportunities for growth.This report tells a story about when and how people use their devices, and where there are opportunities to either market new products or put new features into existing services so that the customers get more positive advantages of the technologies they use.