Report

순천향대학교 SOON CHUN HYANG UNIVERSITY

<Basics of Data Analysis – Final Term Project>

BLEP Fitbit 생체정보를 이용한 날짜, 요일 별 사용자 걸음 수 패턴 분석

과목명 | 데이터 분석 기초

학과명 | 사물인터넷학과

서명 | 20201514 이현수

제출일 | 2022.12.11

Report

순천향대학교 SOON CHUN HYANG UNIVERSITY

-목차

1. 사용한 데이터셋

2. 수집 기간 동안의 평균 소모 칼로리, 평균 이동거리, 평균 Step수 출력.

    2.1. 작성한 코드

    2.2. 출력된 결과

3. Dates와 Calories를 각각 X축, Y축으로 가지는 꺾은선 그래프 출력

    3.1. 작성한 코드

    3.2. 출력된 결과

4. 이동한 거리(Distance)와 소모된 칼로리(Calories) 간의 관계를 알아보고자 함. 각자를 X축, Y축으로 가지는 산점도 출력(Scatter plotting).

    4.1. 작성한 코드

    4.2. 출력된 결과

5. 요일별로 소모된 칼로리, 이동한 거리, 평균 Steps 수를 분석하고 나름대로의 결론 제시

    5.1. 작성한 코드

    5.2. 출력된 결과

1. 사용한 데이터셋

   BLEP Fitbit 생체정보 활용 데이터셋(Fitbit_data.csv)

   SCH AI & 빅데이터 센터: http://aibig.sch.ac.kr/main.do

2. 수집 기간 동안의 평균 소모 칼로리, 평균 이동거리, 평균 Step수 출력.

   2.1. 작성한 코드

```
# 1. 수집 기간동안의 평균 소모 칼로리, 평균 이동거리, 평균 step수를 출력해 보세요.
cals_mean = df['calories'].mean()
dist_mean = df['distances'].mean()
step_mean = df['steps'].mean()

print("The average of 'Calories' is", cals_mean)
print("The average of 'Distances' is", dist_mean)
print("The average of 'Steps' is", step_mean)
```

   Mean 함수를 이용하여 각 칼럼들에 대한 평균값을 구하고, 그것을 xxx_mean이라는 변수에 할당하여 프린트하였다.

   2.2. 출력된 결과

```
The average of 'Calories' is 1828.5578455089824
The average of 'Distances' is 2.831137718562874
The average of 'Steps' is 3942.7844311377244
```

3. Date와 Calories를 각각 X축, Y축으로 가지는 꺾은선 그래프 출력.

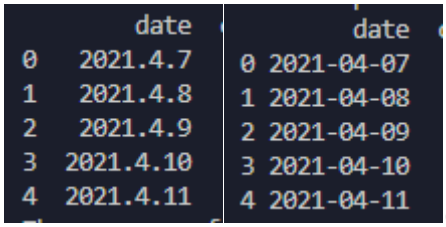
   3.1. 작성한 코드

```
# 2. x축은 date, y축은 calories로 꺾은선 그래프를 그려 보세요.
fig = plt.figure(figsize = (15,9)) #글자가 다 보일 정도의 창 크기
graph_first = fig.add_subplot() #도화지를 만들어준다.
df['date'] = pd.to_datetime(df['date']) #인식하기 쉽도록 index를 datetime으로 바꾸어준다.

graph_first.plot(df['date'], df['calories'], marker = 'o', markersize=3 ) #x,y축에 해당하는 데이터를 넣어준다.

graph_first.set_title('<Calories Per Date>') #그래프의 제목
graph_first.set_xlabel('<Date>') #x축 레이블
graph_first.set_ylabel('<Calories>') #y축 레이블
graph_first.grid() #그래프에 격자 설정
graph_first.xaxis.set_major_locator(dts.MonthLocator(interval=1)) #x축 눈금이 너무 많아 한달 간격으로 하나씩만 그려지도록 함수 작성
plt.show()
```

   Matplot library를 이용하여 그래프를 출력하였다. 그래프를 그리는 전반적인 과정에서 편의성을 가져가기 위해 date 칼럼의 datatype를 datetime으로 바꾸어 주었다.

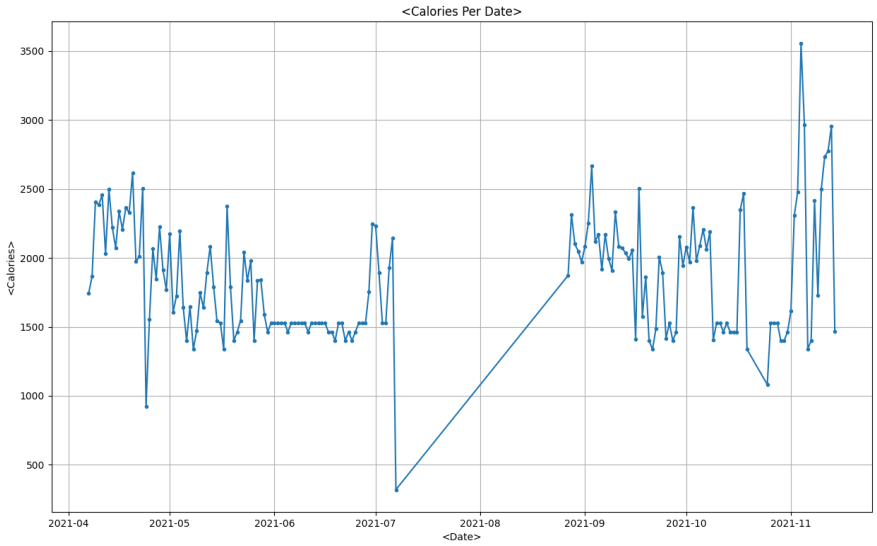


(datetime 변환 전, 후)

이후 plot 함수를 이용해 x-y축에 해당하는 데이터들을 넣어주었다.

마지막으로 제목, x-y축 레이블 설정, 격자 설정 등 그래프 외관을 그리고, x축의 눈금이 일별로 나와 보기 불편한 부분이 있어 set_major_locator 함수를 이용해 한달 간격으로 눈금이 그려지도록 수정하였다.

3.2. 출력된 결과



출력된 결과를 보면, 7월 초부터 8월 말까지 데이터가 비어 있는 것을 확인할 수 있다. 처음에는 잘못 설정하여 이상하게 그래프가 그려진 것인 것 알았는데, csv 파일을 확인해보니 실제로 7월 초 ~ 8월 말 사이에 데이터가 비어 있었다.

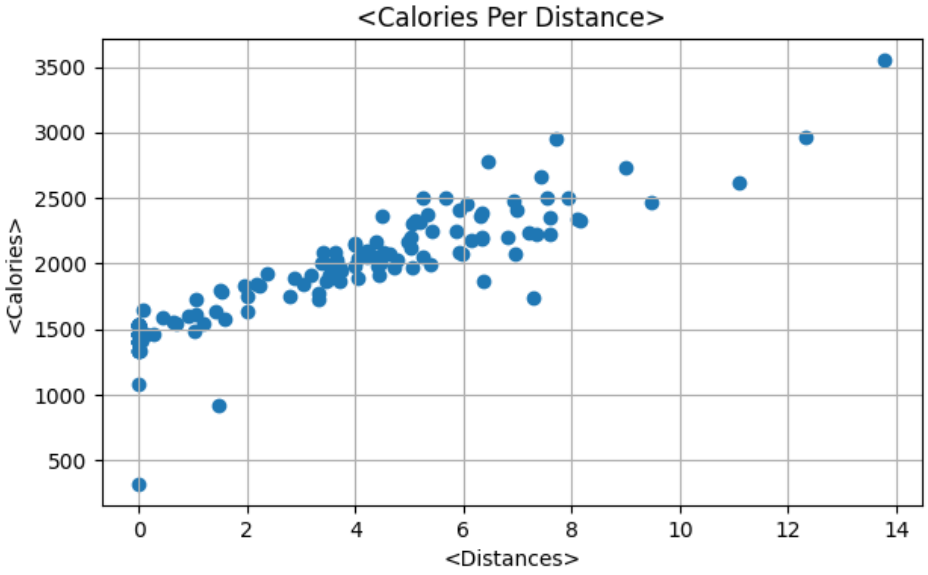
4. 이동한 거리(Distance)와 소모된 칼로리(Calories) 간의 관계를 알아보고자 함. 각자를 X축, Y축으로 가지는 산점도 출력(Scatter plotting).

4.1. 작성한 코드

```python
# 3. 이동한 거리(distance)와 소모된 칼로리(calories)와의 관계를 알아보고자 합니다.
# x축은 distance, y축은 calories로 산점도를 그려보세요. (scatter plotting)
fig = plt.figure(figsize = (7,4))
graph_second = fig.add_subplot()
graph_second.scatter(df['distances'], df['calories'])
graph_second.set_title('<Calories Per Distance>') #그래프의 제목
graph_second.set_xlabel('<Distances>') #x축 레이블
graph_second.set_ylabel('<Calories>') #y축 레이블
graph_second.grid() #그래프에 격자 설정
plt.show()
```

두번째 미션과 동일하게 figure 함수를 이용하여 도화지를 그리고, scatter 함수를 통해 데이터를 불러온 후, 제목이나 레이블, 격자 등을 설정하였다.

4.2. 출력된 결과



산점도를 통해 확인할 수 있는 것은 여러가지이다. 우선 사용자는 8(km - 단위는 확실치 않다.) 이상의 거리는 자주 달리지 않았으며, 대부분 0~8(km) 정도의 거리를 달렸다. 또한 당연한 이야기지만 거리가 늘어날수록 소모된 칼로리도 늘어나는 것을 확인할 수 있으며, 같은 거리라도 칼로리가 다른 날들이 존재한다는 것도 확인할 수 있다(같은 거리에 칼로리 수가 다르기 때문에 아마도 이런 경우에는 경사가 있는 곳을 다녔을 것이라 추측된다.)

5. 요일별로 소모된 칼로리, 이동한 거리, 평균 Steps 수를 분석하고 나름대로의 결론 제시

5.1. 작성한 코드

```python
# 요일 별로 소모 칼로리, 이동거리, 평균 steps 수를 분석해 보고 나름대로의 결론을 내려보세요.

#"요일별"이라는 단어 확인 -> weekday 함수 이용하여 요일 칼럼 추가
df['weekday'] = df['date'].dt.weekday
print(df.head())
df_dayname = df.groupby(by = 'weekday').mean()#요일을 인덱스로 가지는 df_dayname 데이터 프레임 만든다.
print(df_dayname)


fig = plt.figure(figsize = (13,5)) #그래프가 두개이므로 가로로 길게 생성한다.
fig.subplots_adjust(wspace=1)

graph_third_cal1 = fig.add_subplot(1,2,1) #첫번째 그래프를 위한 도화지 생성
graph_third_cal1.set_title('<Calories Per Steps>') #그래프 제목 설정
graph_third_cal1.grid() #그래프에 격자 설정
graph_third_cal1.plot(['Mon','Tue','Wed','Thu','Fri','Sat','Sun',], df_dayname['calories'], marker = 's', color = 'cornflowerblue', markersize=5, label = 'Calories')
graph_third_cal1.set_ylabel('<Calories>', fontsize = 15, color = 'cornflowerblue') #겹치지 않게 label 색, 폰트 지정

graph_third_steps = graph_third_cal1.twinx() # 한 그래프에 출력
graph_third_steps.plot(['Mon','Tue','Wed','Thu','Fri','Sat','Sun',], df_dayname['steps'], marker = 'o', color = 'lightpink', markersize=5, label = 'Steps')
graph_third_steps.set_ylabel('<Steps>', fontsize = 15, color = 'lightpink') #겹치지 않게 label 색, 폰트 지정

graph_third_cal1.legend(loc='upper left') #그래프별 이름 태그
graph_third_steps.legend(loc='upper right') #그래프별 이름 태그

graph_third_cal2 = fig.add_subplot(1,2,2) #두번째 그래프를 위한 도화지 생성
graph_third_cal2.set_title('<Calories Per distances>')#그래프 제목 설정
graph_third_cal2.grid() #그래프에 격자 설정
graph_third_cal2.plot(['Mon','Tue','Wed','Thu','Fri','Sat','Sun',], df_dayname['calories'], marker = 's', color = 'cornflowerblue', markersize=5, label = 'Calories')
graph_third_cal2.set_ylabel('<Calories>', fontsize = 15, color = 'cornflowerblue') #label 색, 폰트 지정

graph_third_distances = graph_third_cal2.twinx() # 한 그래프에 출력
graph_third_distances.plot(['Mon','Tue','Wed','Thu','Fri','Sat','Sun',], df_dayname['distances'], marker = 'o', color = 'orange', markersize=5, label = 'Distances')
graph_third_distances.set_ylabel('<Distances>', fontsize = 15, color = 'orange') #겹치지 않게 label 색, 폰트 지정

graph_third_cal2.legend(loc='upper left') #그래프별 이름 태그
graph_third_distances.legend(loc='upper right') #그래프별 이름 태그

plt.show()
```

weekday 함수를 이용해 date 칼럼에 맞는 요일 칼럼을 추가해준다. 이후 원본 데이터프레임에서 요일 칼럼을 인덱스로 가지는 새로운 데이터프레임 df_dayname을 만들어 프린트한다.

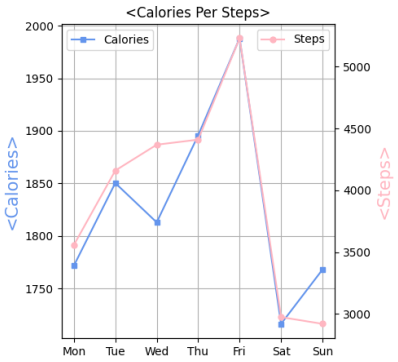
```
        date  calories  distances  steps  weekday
0 2021-04-07   1744.27   7.300000  10174        2
1 2021-04-08   1865.11   3.460000   4829        3
2 2021-04-09   2406.20   5.920000   8262        4
3 2021-04-10   2384.56   6.350000   8864        5
4 2021-04-11   2456.50   6.059999   8467        6
```

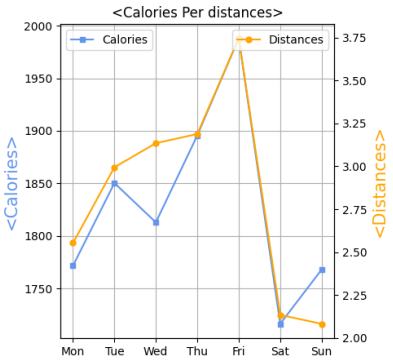
```
             calories  distances        steps
weekday
0         1771.546667   2.552917  3556.750000
1         1850.301250   2.994583  4159.416667
2         1812.871250   3.134583  4368.666667
3         1895.617826   3.188261  4409.652174
4         1988.057917   3.746250  5229.500000
5         1716.314592   2.134167  2974.625000
6         1767.989583   2.082083  2920.333333
```

(요일(weekday)가 추가된 기존 데이터 프레임/요일 칼럼을 인덱스로 하는 새로운 데이터 프레임)

이후는 다른 미션과 동일하게 진행된다. 한 번에 두가지의 그래프를 그려야 하기 때문에 add_subplot함수를 이용해 각각 자리를 만들어주고, 제목, 격자 등의 설정을 한 후, 그래프를 그린다. 이때 요일이 0~6의 숫자로 표시되기 때문에, 이를 각각 요일 약자(Mon, Tue 등)로 바꾸어 준다. 이후 legend 함수를 이용해 이름 태그를 달아준다.

5.2. 출력된 결과





Steps와 Distances는 동일한 모습을 보인다. 금요일에 운동을 가장 많이 한 것을 알 수 있으며, 수요일, 일요일에는 다른 요일에 비해 Steps/Distances와 Calories의 차이가 꽤 나는 것 역시 확인할 수 있다.

5.3. 결론 제시

사용자는 평일의 경우 월~금으로 가며 점점 운동량을 늘린다는 것을 알 수 있다. 또한 수요일의 경우 Steps/Distances에 비해 Calories가 높은 것을 보면 경사로를 걷는다던지, 조금 더 강도높은 운동을 한다는 것을 알 수 있다. 일요일은 반대의 경우인 것을 볼 수 있는데, 이 경우에는 평일을 의식해서일지 조금 강도가 낮은 운동을 하는 것을 확인할 수 있다. 모든 요일 중 금요일에 가장 많이 움직이며, 토요일, 일요일에 가장 움직이지 않는다는 것을 알 수 있다.