Part 3 -- Data analysis (50 points) Rappi's Operations team is interested in decreasing the number of orders that are not taken by any courier, due to the fact that they are not attractive enough for couriers. To solve this problem, we have provided a sample of orders created in September 2017.

Your objective is to use this dataset to help Rappi understand which factors determine whether an order is going to be taken by a courier, and offer recommendations to take actions based on your insights to improve Rappi's operation.

Below you can find a detailed description of the dataset. Please include any code/spreadsheet you wrote/build for the analysis and delete the dataset when you have finished the challenge. In addition, please detail any assumptions you may need or explain any issues you may encounter.

1. Perform any cleaning, exploratory analysis, and/or visualizations to use the provided data for this analysis (a few sentences/plots describing your approach will be enough).

- How many orders were not taken by any courier? 9689 orders

```
In [26]: df[df.taken == 0]
Out[26]:
        order_id   store_id  ...            created_at  taken
)       14364873   30000009  ...  2017-09-07T20:02:17Z      0
.       14370123   30000058  ...  2017-09-07T20:13:16Z      0
!       14368534  900003684  ...  2017-09-07T20:07:23Z      0
;       14367634  900006875  ...  2017-09-07T20:07:23Z      0

125530  15650740  900008202

[9689 rows x 7 columns]
```

● What weekday has the higher percentage of non-taken orders? (20 points)

Using spyder:

```
nt_orders = df[df.taken == 0]

#chage date format
nt_orders['created_at']= pd.to_datetime(nt_orders.created_at)
#create a copy to contrast data
nto_dayweek = nt_orders

#get day of week from created_at
nto_dayweek['created_at']= nto_dayweek.created_at.dt.dayofweek

#group by day of week
by_day_of_week = nto_dayweek.groupby(['created_at']).size()


#calculate de percentage
not_dayweek_percentage = []
indexx = []
result={'day':[],'percentage':[]}
total_nto = 9689
for i, v in by_day_of_week.iteritems():
    percentage = (v/total_nto)*100
    not_dayweek_percentage.append(percentage)
    indexx.append(i)
    result['day'].append(i)
    result['percentage'].append(percentage)


percentage_result = pd.DataFrame.from_dict(result)
```

Result:

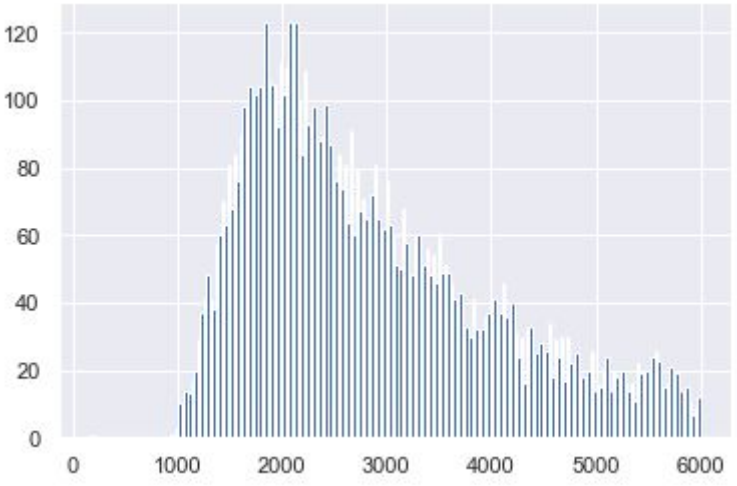
percentage_result - DataFrame

Index	day	percentage
0	Mon	11.4873
1	Tues	13.665
2	Weds	11.2499
3	Thurs	12.9219
4	Fri	16.8232
5	Sat	21.5709
6	Sun	12.282

The code used to get this results is located at 'data_challenge/part3/percentage_spyder.py'

2. Make an analysis to help Rappi determine whether or not an order is going to be taken by a courier. Discuss why you chose your approach, what alternatives you considered, and any concerns you may have (20 points).

- Alternative 1: Rate total_earning/to_user_distance, taken the mean of each case(taken and non-taken orders) shows that a higher rate encourage to take the order, the histograms shows an inflexion point around the 2500 earning units/km. Here some data visualization and results.



Histogram earning units/km non-taken orders.

Non-taken orders mean:4148.98 earning units/km
Taken orders mean: 6078.34 earning units/km

- Alternative 2: Other approach could be evaluate the distance between the store and the user but taking into account the elevation and get the rate earning/distance. The result doesn't give relevant information, been similar to the one obtain in the first alternative.

Non-taken orders mean:4427.24 earning units/km
Taken orders mean: 6064.34 earning units/km

The code used to get this results is located at 'data_challenge/part3/analysis_spyder.py'

3. Briefly discuss how Rappi can take advantage of the insights gained from your analysis to increase the number orders taken by our couriers (10 points).

Given the results, a measure that can be taken to reduce the number of non-taken orders is to offer some kind of benefit for the couriers who are willing to take orders at a low ratio of earning units/km.