Team: Eccentric
Members: Kshitiz Kumawat,Sreeya

Price Setter and Follower Detection (Problem Description)

Objective

Your challenge is to find out which services is/are pricing independently (the price leaders) and which services are the followers (and following whom?), given the history of prices set by the operators.

Data

There are two types of seats, but within one type also there can be multiple prices for different categories (front/back/upper/lower etc.) of seats. This categorization is decided by the operator based on the bus, hence there may be different numbers of prices for different services.

Data Fields

1. Seat Fare Type 1 – Within Seat Type 1, the prices of all categories of available seats as defined by the operator.
2. Seat Fare Type 2 - Within Seat Type 2, the prices of all categories of available seats as defined by the operator.
3. Bus – A particular bus service, for example, Hyderabad to Pune Go Tours 9:15 PM bus.
4. Service Date – The date of journey for which the prices are recorded.
5. Recorded At - The time when prices were recorded.

Steps we followed
(Approach)

1. Data Exploration
2. Data Preprocessing
3. Data Cleaning
4. ‘follows’ function (**Statistical Model for prediction**)
5. ‘Follows’ column formation
6. ‘Is followed by’ column formation
7. csv generation

Terminology used:
Seat Fare Type => SFT
non0_SFT1_buses (list) => Buses with non zero SFT 1
zero_SFT1_buses (list) => Buses with zero SFT 1
non0_SFT_buses (list) => Buses with both non 0 SFT1 & 2

Data Preprocessing Steps

1. Converted ‘RecordedAt’ & ‘Service Date’ columns from string data type to datetime format.
2. Filled the ‘NaN’ (Not a number) values with ‘0’ (string zero).
3. Removed the rows where both SFT 1 & SFT 2 are ‘0’ (string zero).
4. Converted SFT 1 & 2 Values from string to float and then taken mean of them (for ease of calculation).
5. For a particular bus & service date, appended the succeeding cells with preceding non 0 cell values.
6. Dropped duplicates.

Data Preprocessing 5th Step

5. For a particular bus & service date, appended the succeeding cells with preceding non 0 cell values

Seat Fare Type 1	Seat Fare Type 2	Bus	Service Date	RecordedAt
0.000000	1119.0	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-14 23:52:00
1330.000000	0.0	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-14 23:52:00
0.000000	1140.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:21:00
1352.333333	0.0	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:21:00
0.000000	1135.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:57:00
1349.333333	0.0	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:57:00

Seat Fare Type 1	Seat Fare Type 2	Bus	Service Date	RecordedAt
0.000000	1119.0	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-14 23:52:00
1330.000000	1119.0	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-14 23:52:00
1330.000000	1140.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:21:00
1352.333333	1140.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:21:00
1352.333333	1135.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:57:00
1349.333333	1135.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:57:00

Data Cleaning Step 1

1. Removed the first rows of non0_SFT1_buses, which contains 0 SFT 1.

Seat Fare Type 1	Seat Fare Type 2	Bus	Service Date	RecordedAt
0.000000	1119.0	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-14 23:52:00
1330.000000	1119.0	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-14 23:52:00
1330.000000	1140.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:21:00
1352.333333	1140.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:21:00
1352.333333	1135.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:57:00
1349.333333	1135.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:57:00

Seat Fare Type 1	Seat Fare Type 2	Bus	Service Date	RecordedAt
1330.000000	1119.0	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-14 23:52:00
1330.000000	1140.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:21:00
1352.333333	1140.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:21:00
1352.333333	1135.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:57:00
1349.333333	1135.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:57:00

Data Cleaning Step 2

2. Removed the initial rows (redundant rows) with same RecordedAt.

Seat Fare Type 1	Seat Fare Type 2	Bus	Service Date	RecordedAt
1330.000000	1119.0	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-14 23:52:00
1330.000000	1140.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:21:00
1352.333333	1140.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:21:00
1352.333333	1135.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:57:00
1349.333333	1135.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:57:00

Seat Fare Type 1	Seat Fare Type 2	Bus	Service Date	RecordedAt
1330.000000	1119.0	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-14 23:52:00
1352.333333	1140.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:21:00
1349.333333	1135.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 01:57:00
1371.000000	1154.5	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 02:44:00
1311.333333	1105.0	5580f995d6f4d3bcceca7e2db6c77bf7	2020-07-15	2020-07-15 03:04:00

follows(bus1list,bus2list,sf_types,followed_by)

- **Steps to determine which bus follows which bus:**

1. For a bus1 in ‘bus1list’, for a bus2 in ‘bus2list’, for a SFT in sf_type list (Let’s say for SFT 2) & for a particular service date, we have created 2 dataframes df1 for bus 1 & df2 for bus2, inside follows function.

2. We appended a ‘fare_change_df1_Seat Fare Type 2’ column to df1 & ‘fare_change_df2_Seat Fare Type 2’ column to df2. If fare increases we append 1, if decreases then -1 else 0. For example:

Seat Fare Type 2	Bus	RecordedAt	fare_change_df1_Seat Fare Type 2
12	A	2020-06-15 18:41:00	0
14	A	2020-07-09 07:32:00	1
13	A	2020-07-09 07:33:00	-1
12	A	2020-07-09 07:59:00	-1
12	A	2020-07-14 09:33:00	0
12	A	2020-07-14 13:12:00	0
15	A	2020-07-14 13:13:00	1
15	A	2020-07-14 13:37:00	0
15	A	2020-07-15 08:15:00	0
14	A	2020-07-15 10:33:00	-1

Bus 1

Seat Fare Type 2	Bus	RecordedAt	fare_change_df2_Seat Fare Type 2
30	B	2020-06-11 19:20:00	0
25	B	2020-07-12 10:11:00	-1
25	B	2020-07-14 06:54:00	0
25	B	2020-07-14 11:08:00	0
26	B	2020-07-14 12:26:00	1
25	B	2020-07-14 15:09:00	-1
25	B	2020-07-14 17:15:00	0
25	B	2020-07-15 02:16:00	0

Bus 2

follows(bus1list,bus2list,sf_types,followed_by)

- **Steps to determine which bus follows which bus:**

3. Then we made two lists **change_in_df1_due_to_df2** & **time_difference**. If -1/1 has appeared in df2 fare change column then we see the df1 fare change column (immediately after the Recorded At time of df2 fare change) if -1/1 appears there then we append 1(reward) to change_in_df1_due_to_df2, & if 1/-1 appears in df1 fare change column then we append -1(penalise).
In time_difference we appended the time after which change has happened in the df1 fare change column.

Seat Fare Type 2	Bus	RecordedAt	fare_change_df1_Seat Fare Type 2
12	A	2020-06-15 18:41:00	0
14	A	2020-07-09 07:32:00	1
13	A	2020-07-09 07:33:00	-1
12	A	2020-07-09 07:59:00	-1
12	A	2020-07-12 12:33:00	1
12	A	2020-07-14 13:12:00	0
15	A	2020-07-14 13:13:00	1
15	A	2020-07-14 13:37:00	0
15	A	2020-07-15 08:15:00	0
14	A	2020-07-15 10:33:00	-1

Bus 1

Seat Fare Type 2	Bus	RecordedAt	fare_change_df2_Seat Fare Type 2
30	B	2020-06-11 19:20:00	0
25	B	2020-07-12 10:11:00	-1
25	B	2020-07-14 06:54:00	0
25	B	2020-07-14 11:08:00	0
26	B	2020-07-14 12:26:00	1
25	B	2020-07-14 15:09:00	-1
25	B	2020-07-14 17:15:00	0
25	B	2020-07-15 02:16:00	0

Bus 2

Here change_in_df1_due_to_df2 = [-1,1,1]
time_difference = [Timedelta('0 days 02:22:00'), Timedelta('0 days 00:47:00'), Timedelta('0 days 19:24:00')]

Intuition behind approach

- **Steps to determine which bus follows which bus:**

4. Then we appended the average(change_in_df1_due_to_df2) & average(time_difference) in **s_dates_corrs** list.

s_dates_corrs = [[0.3333, Timedelta('0 days 07:31:00')]]

5. Then we looped this procedure for all service dates & appended non empty lists.

s_dates_corrs = [[0.3333, Timedelta('0 days 07:31:00')], [0.75, Timedelta('2 days 06:28:30)],[1.0, Timedelta('1 days 15:54:00')]]

6. Then we created a dictionary **bus2_corrs**.

In which corresponding to the key 'B' (name of bus2) we appended [avg(avg(change_in_df1_due_to_df2)), avg(avg(change_in_df1_due_to_df2))*len(s_dates_corrs) , avg(avg(time_difference))]

[Here first average is for a single service date then next average is for all service dates.]

bus2_corrs = {'B' : [0.6944,2.083,Timedelta('1 days 15:22:17)]}

avg(avg(change_in_df1_due_to_df2)) := Conditional Probability of Change in Bus 1 fare when change in Bus 2 fare has already occurred or Confidence Score

len(s_dates_corrs) := Number of common service days

Metric for ranking 'Conditional Probability * Number of common service days' = avg(avg(change_in_df1_due_to_df2))*len(s_dates_corrs)

7. If this metric is same for 2 buses then we take the bus with smaller avg(avg(time_difference)).

Experimental results & details

- **Steps to determine which bus follows which bus:**

8. Then we looped this procedure for all bus 2 there in bus2list. Updated bus2_corrs dictionary. For example:
bus1list = ['09d3a01cf347bce0b92631414af3fea8'] (a zero_SFT1_bus), bus2list = zero_SFT1_buses+non0_SFT_buses, sf_type = ['Seat Fare Type 2'], followed_by = False
bus2_corrs is

```
{'b38712db6ce31a420650d8e701799aee': [1.0,  4.0,  Timedelta('0 days 00:21:21.250000')],
       '60468e0a545d3f8f7a6a19eb6cfc3ca8':    [1.0, 1.0, Timedelta('0 days 10:34:30')],
    'ab479dab4a9e6bc3eaefe77a09f027ed':   [0.8333,  2.5,  Timedelta('0 days 23:16:55')],
      'd1854bef9a416d20150912c61a1fb9e1':   [-1.0,  -1.0,  Timedelta('0 days 00:42:30')],
       'a68a88422070999db27c80a179ad4664': [1.0, 2.0, Timedelta('0 days 01:45:00')],
    '74b23405506957f6b6ddb29c0574260b':   [0.3333,  0.3333,  Timedelta('0 days 13:19:00')],
       '74cc09e5cb64134800650f0c21f94942':    [1.0, 1.0, Timedelta('0 days 01:25:00')],
       '3100ec449829e65ed2da9f7af62ef03d':    [1.0, 2.0, Timedelta('1 days 12:10:30')],
  '225554f4707e5122d97d5fb642b10af3':    [-0.3333,  -0.3333,  Timedelta('0 days 11:54:40')],
       'c0e0a47587fbf6247f4f9a22ba22cc80':      [1.0, 1.0, Timedelta('0 days 10:22:00')],
       '6d364920e6c9f9f71b1d881107e639f0':      [1.0, 1.0, Timedelta('0 days 02:01:00')]}
```

=> bus1 follows 'b38712db6ce31a420650d8e701799aee' with confidence score of 1.

9. Then we looped this procedure for all buses in bus1list. And appended the results in **bus1_corrs** dataframe.
follows(zero_SFT2_buses,zero_SFT2_buses+non0_SFT_buses,['Seat Fare Type 1'],False)
follows(zero_SFT1_buses,zero_SFT1_buses+non0_SFT_buses,['Seat Fare Type 2'],False)
follows(non0_SFT_buses,Buses,['Seat Fare Type 1','Seat Fare Type 2'],False)

Note: While considering SFT1 & SFT2 both we just simply appended the change_in_df1_due_to_df2 & time_difference while considering SFT1 & SFT2 one by one.

Intuition behind approach

- **Steps to determine the 'Is followed by' column (Just the reverse way):**

1. We picked up buses one by one from the dataframe, used them as bus2 & seen among suitable bus1s which bus has the maximum value of used metric.
2. Inserted the obtained bus & confidence score in dataframe.
3. If a clash happens then we have taken the bus with a smaller avg(avg(time_difference)).

For Independent buses (with empty change_in_df1_due_to_df2), we skipped them or filled a blank space or 0.

2. We appended the independent buses in submission file.

Generalisation of the approach & final notes

- Our approach works well with various test scenarios which we faced with given dataset.
- Our code can be modifiable for other test scenarios as well.
- Our code needs a little modification to tackle the case when there are all negative values of our metric in bus2_corrs.
- Our code runs from starting to end in around 5 mins. It can be optimised to execute on large datasets. **(Vectorization)**
- *Our model is transparent instead of being a kind of black box ML model.*

Thank You !!

‘follows’ function (Our Statistical Model for prediction)

Note: Here we are considering change in fare of bus 1 due to change in fare of bus 2.

follows(bus1list,bus2list,sf_types,followed_by)

- **bus1list :** It is the list from which we take bus 1
- **bus2list :** It is the list from which we take bus 2
- **sf_types**: It is the list which contains seat fare types which we consider
- When **followed_by** = True: function will create the follow column
- When **followed_by** = False: function gives us the dataframe of followed_by column

[If time permits]

- **(Observation)** A modified better metric which can be used is:
  for **max(sum(avg(change_in_df1_due_to_df2)))>2** is **sum(avg(change_in_df1_due_to_df2))/avg(avg(time_difference))**
  avg(avg(time_difference) is in seconds
- **sum(avg(change_in_df1_due_to_df2)) =** avg(avg(change_in_df1_due_to_df2))*len(s_dates_corrs)

Extra slide for Information