BigData Mid-term Exam GirlsGeneration

M10609201 賈加平

Q1. Separate Datasets into 「Two Groups」

- According to exam's instruction, I split the dataset into two groups by feature "SEAT_ROW" whether is null.

- If "SEAT_ROW" is notnull
  - It means that there have seats and the floor is on 2F and 3F
  - It also means "non-Rock Zone" or "General admission"
  - I name this goupr as "A"
- If "SEAT_ROW" is null
  - It means that there has no seat and the floor is on B1
  - It also means "Rock Zone" or "Standing area"
  - I name this group as "B"

- So, Dataset split to 「A / B」 or 「RockZone / non-RockZone」 Two Groups
  - Then do some data preprocessing Before to ansIr Q2 and Q3

Q2 Define ticket selling speed for A and B under T1、T2

- Ticket Selling Speed Definition:
  - I think Selling Speed is amount of tickets sold in a interval Of Time
  - The unit of time is hour, because if I compute the average by minute or seconds, the average will too small to comapre A and B under T1、T2
  - The simple formula is described below
    - **Ticket Selling Speed ：△ amount of tickets selling / △timeDelta**
  - I follow this simple formula to show the result on the next page
    - The result is a form of Matrix of AB and T1 T2
    - To define selling speed

Q2 Define ticket selling speed for A and B Results (cont.)

The ticket selling speed for A/B under the T1 T2 Speed is described below:

Ticket selling speed	T1 (09-18 12:00-16:00)	T2 (09-18 16:00-20:00)
A (non-RockZone)	369.03 tickets/hr	723.00 tickets/hr
B (RockZone)	99.61 tickets/hr	268.07 tickets/hr

ph1

ph2

Data Visualization

According the Result of Matrix, I Found the two Iird phenomenon (ph1, ph2):

- The selling day is Saturday
- **Phenomenon 1 :**
  - **T2's selling speed > T1's selling speed** : Normally, GirlsGeneration is very famous, so there should has more tickets buyed under T1, **but the result show the reversed situation**. I think maybe it's because Saturday is holliday, everybody goes outside.
- **Phenomenon 2:**
  - A'speed under T1 > B's speed under T2 ：To be honest, if the concert is great, there should be many people booking for the RockZone . **But the result doesn't match our assumption**, I think the reason is that "There should many people to booking for RockZone, but confined the "System loading". So I should take more consideration about the issue and dataset I analyze." (More details is in the next page.)

Q3. Define new measurement to estimate “System Loading”

- use “Amount of Requests”

- Normally, The term “System Loading” is directly related to the “**amount that system accept the number of user’s requests**.” According to the description above, I found some special pattern when I observed the “**ORDER_ID**”:
  - **Each the same ORDER_ID always to map the unique Identity (person)**
    - It mean one unique ORDER_ID --represent--> one unique person --represent--> one unique request
  - **The ORDER_ID show the situation that the number of series of this columns is not consecutive**
    - Normally, The Order_ID means when person books the ticket, the system will give the ID to this ticket in the database, and when next person books the ticket, system will give the “consecutive” ID number to this guy. So if system runs lll, the order_id should consecutive.
    - But the dataset doesn’t show that, so I think possibly there are some people book fail, maybe it’s because system loading cannot afford many requests in a short interval of time
  - **So I have a assumption : Amount consecutive requests represent the loading that system can afford.**
    - **Simple Formula : sum (Amount consecutive requests) / count(block of Amount of Consecutive ORDER_ID)**
    - instructions: Block means the times that non-consecutive situations appear

Q3. Define “System Loading” By Amount of Requests‘ Result (cont.)

The result of system loading is shold below. **The two phenomenon (ph1, ph2) will be introduced on next page.**

Requests that System can suffer		T1 (09-18 12:00-16:00)	T2 (09-18 16:00-20:00)
A (non-RockZone)	Max Request	11	14
	Average Request	3.047	2.377
	Max Fail People	6	11
	Average Fail People	2.402	2.754
B (RockZone)	Max Request	4	5
	Average Request	1.274	1.272
	Book Fail People	29	27
	Average Fail People	7.315	5.844

ph1

ph2

Max Request（請求數）：最大能承受的請求數
Average Request（請求數）：平均能承受的請求數
Max Fail People（人數）：最大流失多少人（代表請求失敗的人數在非連續的區間）
Average Fail People（人數）：平均流失多少人（代表請求失敗的人數在非連續的區間）
說明：Reuquests越少、訂票人數失敗人數越多，代表System run more busyly

Q3. Define “System Loading” By Amount of Requests (cont.)

Requests that System can suffer		T1 (09-18 12:00-16:00)	T2 (09-18 16:00-20:00)
A (non-RockZone)	Max Request	11	14
	Average Request	3.047	2.377
	Max Fail People	6	11
	Average Fail People	2.402	2.754
B (RockZone)	Max Request	4	5
	Average Request	1.274	1.272
	Book Fail People	29	27
	Average Fail People	7.315	5.844

ph1

ph2

```
T1 : 人數觀點
編號 Range :  567
編號不重複數:  390
總共流失多少人: 177
A in T1:
最大能承受的請求數(人數):11
平均能承受的請求數(人數):3.047
最大流失多少人:6
平均流失多少人:2.402
-----------------
```

```
T2 : 人數觀點
編號 Range :  2249
編號不重複數:  1296
總共流失多少人: 953
A in T2:
最大能承受的請求數(人數):14
平均能承受的請求數(人數):2.377
最大流失多少人:11
平均流失多少人:2.754
-----------------
```

```
T1 : 人數觀點
編號 Range :  554
編號不重複數:  94
總共流失多少人: 460
B in T1:
最大能承受的請求數(人數):4
平均能承受的請求數(人數):1.274
最大流失多少人:29
平均流失多少人:7.315
-----------------
```

```
T2 : 人數觀點
編號 Range :  2043
編號不重複數:  426
總共流失多少人: 1617
B in T2:
最大能承受的請求數(人數):5
平均能承受的請求數(人數):1.272
最大流失多少人:27
平均流失多少人:5.844
-----------------
```

Two Phenomenons:

- System run badly when handle a lot of requests
- It also means System run badly when many people booking

- **Ph 1：T2’s system loading > T1’s system loading (T1 is busier)**
  - This result makes more common sense than that I analyzed by selling speed. Once the ticket system open, there would will be many people send request to the system’s server. At this time, the requests that system handles will be limited, because it has so many requests in the meantime. Although the result show little difference, it has big difference betIen A under T1 and T2.
- **Ph 2：T1 non-RockZone’s System Loading > T1 RockZone’s System Loading (B is busier)**
  - This result makes common sense, because **the system loading runs very poorly** when **people books for RockZone than books for non-RockZone.** It also means that there so many people to book for RockZone, and system is very busy at that time. So it doesn’t accept the requests completely. Besides, the “Book Fail People” is up to around 30 persons when booking for RockZone, it represent people really love GirlsGeneration very much.

Conclusion

- **From viewpoint of Ticket Selling Speed：**
  - I find out the some Iird situation according the analysis
  - For example :
    - ➢ It shows that T2 selling speed > T1 selling speed <= it should be incorrect (T1>T2 more common sense)
    - ➢ It shows that A selling speed > B selling speed <= it should be incorrect (B>A more common sense)
- **From viewpoint of System loading：**
  - The result match the reality
  - For Example：System Loading 越低代表該能接收的 requests 越少
    - ➢ It shows that T2 System Loading > T1 System Loading (T1 is more busy than T2, common sense)
    - ➢ It shows that A System Loading > B System Loading (Booking for B is busier than A, common sense)
- **總結來說**，開賣的時候，大家都會在很早的時候就會訂票，且都會訂搖滾區位置的票，這造成短時間大量的請求，因為礙於系統運行限制下，只能接收部分的請求，而資料集卻忽略了這些的請求，只有紀錄那些成功的請求（也就是所謂訂成功的票數），所以「以銷售票數的速度」來看，忽略了真實性反而得到假象，反觀從「系統負載」的觀點去分析，才會發現比較可能貼近的事實為何，**也就是很多人還是會很早訂票且會搶購搖滾區的票**

Backup

- Data Preprocessing
- Ticket selling speed

Data Preprocessing

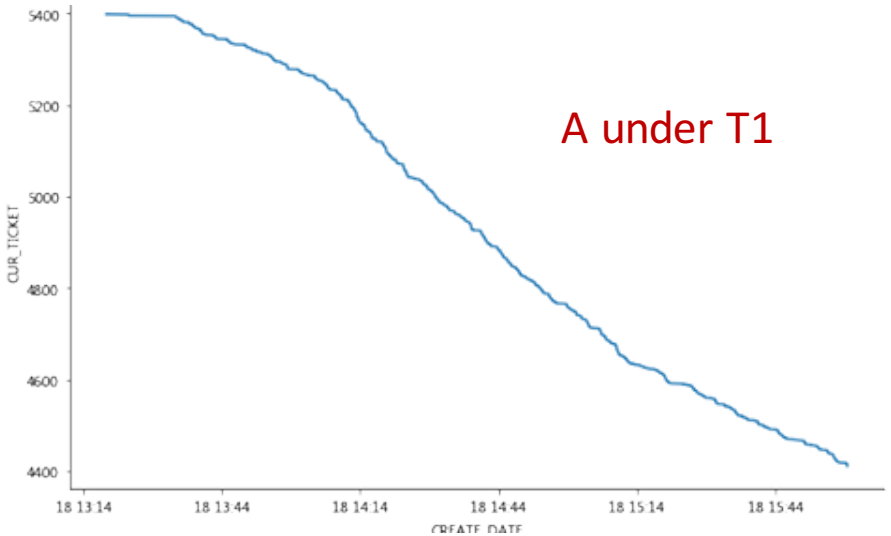
- **Data Selection:**
  - "CREATE_DATE" , "ORDER_ID","PRICE" , "T_STANDARD_TICKET_TYPE_NAME" , "SEAT_ROW","SEAT_REGION_NAME"

- **Feature construction and conversion:**
  - Construct the new feature "WITH_SEAT" ["A","B"] mapped from "SEAT_ROW"
  - Convert "CREATE_DATE" to comparable object like panda's DatetimeIndex

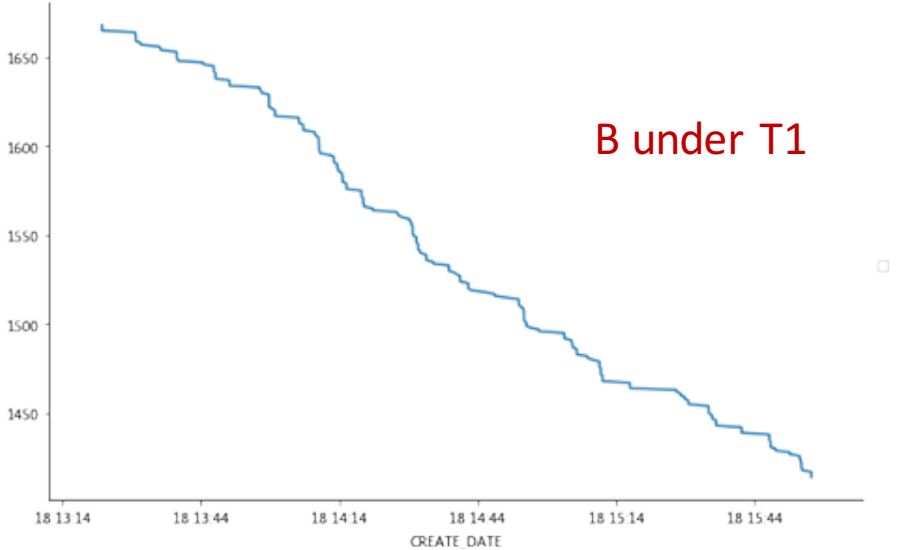
- **Feature Combination: Assume the total tickets is the same as sample size (n=7069)**
  - Compute the existing tickets that each kind of "WITH_SEAT"

Ticket selling speed

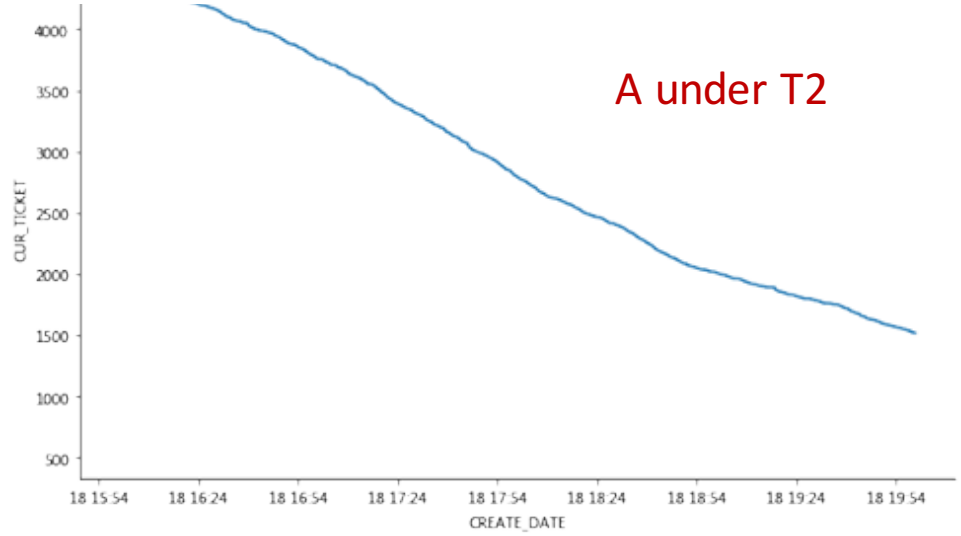
該時段起始時間至最後區間為：2010-09-18 13:18:55 -> 2010-09-18 15:59:37，販售票數變化狀況：5399 -> 4411
A in T1 (2010-09-18 12:00:00-16:00:00)
共銷售989張，時間2.68小時，
平均一小時販售369.03張：



該時段起始時間至最後區間為：2010-09-18 13:22:33 -> 2010-09-18 15:56:24，販售票數變化狀況：1668 -> 1414
B in T1 (2010-09-18 12:00:00-16:00:00)
共銷售255張，時間2.56小時，
平均一小時販售99.61張：



該時段起始時間至最後區間為：2010-09-18 16:00:07 -> 2010-09-18 19:59:54，販售票數變化狀況：4410 -> 1519
A in T2 (2010-09-18 16:00:00-20:00:00)
共銷售2892張，時間4.0小時，
平均一小時販售723.0張：



該時段起始時間至最後區間為：2010-09-18 16:00:05 -> 2010-09-18 19:19:20，販售票數變化狀況：1413 -> 524
B in T2 (2010-09-18 16:00:00-20:00:00)
共銷售890張，時間3.32小時，
平均一小時販售268.07張：