

Envoy Interview - Mikiko Bazeley

IN [1]

1 import matplotlib.pyplot as plt

Py

IN [2]

1 datasets

Py

OUT [2]

index	qref	name	rows	columns
0	ba0fdd8a3c8d	Task 2: Percentile Check-Ins	8	2
1	18a4450ea107	Task 2: [TABLE] Frequency of Check-Ins	1217	3
2	9f8819cfce80	Task 3: [TABLE] Frequency - Time to Sign-Up	1217	6
3	aeffd78719c2	Task 3: Min/Mean/Max SignUp Time	1	4
4	d7296050d413	Task 3: Percentile Sign Ups	8	2
5	4d02f0aacb82	Task 4A: % Signups by Admins	2	2
6	a91b10684722	Task 4B: [TABLE] Composition of Admin/Non-Admin	3990	6
7	f9e019e2d2da	Task 4B: Composition of Admin/Non-Admin by Company Type	6	7
8	bcd8b252b5e18	Task 4B: Composition of Admin/Non-Admin - Total	1	6
9	7b3b38bd331f	Task 1: [DATA] Users	5259	5
10	dec6df2c647a	Task 1: [DATA] Events	68951	4

Task 1

As your first task, do an initial exploration of the data. What kind of statistics and/or visualizations could help you summarize this data? What specific things stick out to you? Determine characteristics of the dataset, such as the date of the first event, the distribution of events across users, etc. Feel free to flex your creativity here. There are no right or wrong answers.

Mikiko's Notes:

My initial steps would be to understand the structure of the tables and data sets, including the number of records, available data types, and basic summary statistics.

IN [3]

1 users_df = datasets['Task 1: [DATA] Users']
2
3 users_df.info()

Py

OUT [3]

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5259 entries, 0 to 5258  
Data columns (total 5 columns):  
user_id      5259 non-null float64  
created_at   5259 non-null object  
company_id   5259 non-null float64  
is_admin     5259 non-null bool  
company_type 5259 non-null object  
dtypes: bool(1), float64(2), object(2)  
memory usage: 169.6+ KB
```

IN [4]

1 users_df.head()

Py

OUT [4]

● Ready

IN [8]

OUT [8]

1

events_df.head()

Py

index	event_id	created_at	user_id	event_type
0	194938.0	2018-07-01 00:00:42	11847.0	check_in
1	277953.0	2018-07-01 00:03:42	15224.0	check_in
2	219641.0	2018-07-01 00:04:56	15895.0	check_in
3	267292.0	2018-07-01 00:12:37	13606.0	check_in
4	107640.0	2018-07-01 00:14:50	4844.0	check_in

IN [9]

OUT [9]

1

events_df.tail()

Py

index	event_id	created_at	user_id	event_type
68946	130028.0	2018-10-21 23:52:46	6657.0	check_in
68947	126680.0	2018-10-21 23:53:43	6318.0	check_in
68948	45085.0	2018-10-21 23:54:18	18195.0	check_in
68949	231272.0	2018-10-21 23:55:53	8615.0	sign_up
68950	172033.0	2018-10-21 23:57:04	10055.0	check_in

IN [10]

OUT [10]

1

events_df.groupby('event_type').count()

Py

index	event_id	created_at	user_id
event_type			
check_in	67583	67583	67583
sign_up	1368	1368	1368

IN [11]

OUT [11]

1

events_df.groupby(['event_type', 'user_id']).count()

Py

index	event_id	created_at
event_type	user_id	
check_in	8.0	19
	11.0	23
	19.0	18
	28.0	11
	32.0	20
	44.0	20
	45.0	20

Ready

18891.0	1	1
18892.0	1	1
18896.0	1	1
18899.0	1	1
18902.0	1	1
18903.0	1	1

Task 2

The Product Managers on your team have come to you with a question;

“How many check-in’s does a user tend to have before signing up their own company for the product?”

Help them answer this question and deliver the answer with enough context and nuance as you feel is appropriate. Make sure to also show the code you used to obtain this answer.

Mikiko's Notes:

- Users will check in an average of ~ 8 times before signing up their own company for a product.
- The minimum amount of check-ins is 1 and the maximim number of check ins is 25.
- 50% of users will have checked in 7 times before sign up & 75% of users will have checked in 13 times.

To summarize, a majority of users will have checkedin 13 times before sign up. We have a long tail of users that check in up to 25 times before sign up.

An interesting follow up would be to understand:

If there are systemic differences between the short and long-cycle signups due to:

- Company Type
- Sign-up Cohort
- Admin vs. Non-Admin

```
IN [12]: 1 table_checkIn_freq = datasets['Task 2: [TABLE] Frequency of Check-Ins']
2
3 print(table_checkIn_freq.head())
4 print(table_checkIn_freq.tail())
```








```
OUT [12]:  user_id  total_prior_check_in  percentile
0  18310.0                1            1
1  18521.0                1            1
2  14817.0                1            1
3  14402.0                1            1
4   9543.0                1            1
 user_id  total_prior_check_in  percentile
1212  15728.0                24           100
1213   9246.0                24           100
1214  16685.0                24           100
1215  16708.0                24           100
1216  17037.0                25           100
```

```
IN [13]: 1 plt.hist(table_checkIn_freq.total_prior_check_in)
2 plt.ylabel('Count of Users')
3 plt.xlabel('# of Check-Ins before Company Sign-up')
4 plt.suptitle('Distribution of Check-In #s before Sign-up')
5 plt.title('Red = Avg # of Check-Ins | Green = Mid of Check-Ins')
6 plt.vlines(x=table_checkIn_freq.total_prior_check_in.mean(),ymin=0,ymax=350,color='r',label='avg')
7 plt.vlines(x=table_checkIn_freq.total_prior_check_in.median(),ymin=0,ymax=350,color='g',label='MED')
```

Ready

Mikiko's Notes:

● Ready

https://app.mode.co/... Markdow... interview/reports/ea8b4cb9f583/notebooks/dca8116dc177/export.py?trk_element_id=1

IN [18]

1

table_time_to_signup.hours_signup.describe()

Py

OUT [18]

count1217.000000

mean1114.725555

std755.649220

min1.000000

25%441.000000

50%991.000000

75%1745.000000

max2634.000000

Name: hours_signup, dtype: float64

IN [19]

1

plt.hist(table_time_to_signup.hours_signup)

2

plt.ylabel('Count of Users')

3

plt.xlabel('Hrs(#) from first use to Company Sign-up')

4

plt.suptitle('Distribution of Hrs(#) before Sign-up')

5

plt.title('Red = Avg # of Check-Ins | Green = Mid of Check-Ins')

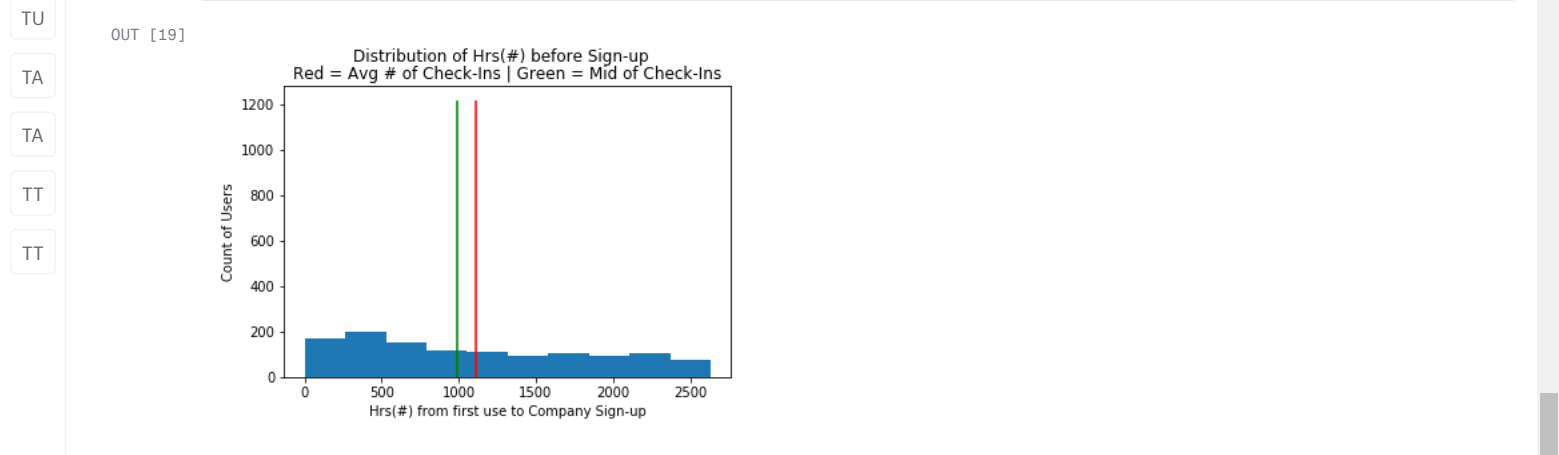
6

plt.vlines(x=table_time_to_signup.hours_signup.mean(),ymin=table_time_to_signup.hours_signup.min(),ymax=table_time_to_signup.hours_signup.max(),color='red')

7

plt.vlines(x=table_time_to_signup.hours_signup.median(),ymin=table_time_to_signup.hours_signup.min(),ymax=table_time_to_signup.hours_signup.max(),color='green')

Py



Task 4A

The Sales Managers come to you with a set of questions;

“How often is the user who signs up a company for the product also an admin of that company?

Help them answer these questions and deliver the answers with enough context and nuance as you feel is appropriate. Make sure to also show the code you used to obtain these answers.

Mikiko's Notes:

A majority of sign-ups are by non-admins of the company, potentially indicating that the product is purchased as an o-ff-vendor/business side purchase as opposed to producrement or CTO side of the company.

IN [20]

1

"How often is the user who signs up a company for the product also an admin of that company?"

2

3

table_signUps_by_admins = datasets['Task 4A: % Signups by Admins']

4

5




table_signUps_by_admins

Py

OUT [20]

	index	is_admin	count
0		False	1187
1		True	181

Ready

- 
- 
- 
- TE
- TU
- TC
- TC
- TS
- TT
- TU
- TA
- TA
- TT
- TT

Avg. % of users as admins by company type:

- Government = 14%
- B2B Software = 16%
- E-Commerce = 12%
- Gaming = 16%
- Healthcare = 18%
- B2C Software = 13%

Typically software will have more non-admin users than admin users so differents in % listed above potentially reflect greater numbers of total users.

IN [21]

```
1 ### Also, what is the typical breakdown of Admins/Non-Admins at companies?"
2
3 table_admins_vs_nonadmins = datasets['Task 4B: [TABLE] Composition of Admin/Non-Admin']
4
5 print(table_admins_vs_nonadmins.head())
6 print(table_admins_vs_nonadmins.tail())
```

Py

OUT [21]

	company_type	company_id	admin_num	nonadmin_num	total_users	perc_admins
0	Gaming	11260.0	1	0	1	100.0
1	E-Commerce	12030.0	0	1	1	0.0
2	B2B Software	1212.0	0	1	1	0.0
3	B2B Software	7218.0	0	1	1	0.0
4	B2B Software	154.0	0	1	1	0.0
	company_type	company_id	admin_num	nonadmin_num	total_users	\
3985	B2C Software	8646.0	1	0	1	
3986	B2B Software	42.0	0	4	4	
3987	B2B Software	10281.0	0	1	1	
3988	B2C Software	5912.0	0	1	1	
3989	Gaming	9375.0	0	1	1	
	perc_admins					
3985	100.0					
3986	0.0					
3987	0.0					
3988	0.0					
3989	0.0					

IN [22]

```
1 table_admins_vs_nonadmins.total_users.describe()
```

Py

OUT [22]

```
count    3990.000000
mean      1.318045
std       4.674821
min       1.000000
25%       1.000000
50%       1.000000
75%       1.000000
max       241.000000
Name: total_users, dtype: float64
```

IN [23]

```
1 admins_vs_nonadmins_comp = datasets['Task 4B: Composition of Admin/Non-Admin by Company Type']
2
3 admins_vs_nonadmins_comp
```

Py

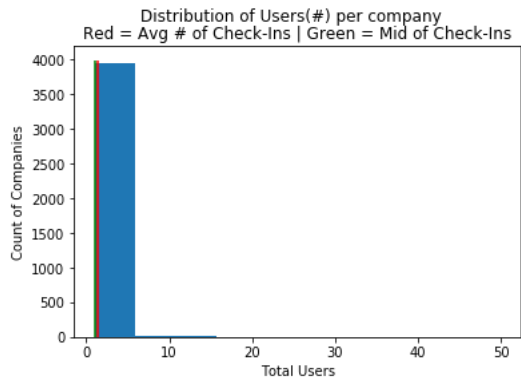
OUT [23]

index	company_type	mean_admin_num	mean_nonadmin_num	mean_total_users	min_perc_admins	mean_perc_admins	max_perc_admins
0	Government	0	1	1	0.0	14.0	100.0
1	B2B Software	0	1	2	0.0	16.0	100.0
2	E-Commerce	0	1	1	0.0	12.0	100.0
3	Gaming	0	1	1	0.0	16.0	100.0
4	Healthcare	0	1	1	0.0	18.0	100.0

IN [25]

```
1 plt.hist(table_admins_vs_nonadmins.total_users[table_admins_vs_nonadmins.total_users<60])
2 plt.ylabel('Count of Companies')
3 plt.xlabel('Total Users')
4 plt.suptitle('Distribution of Users(#) per company')
5 plt.title('Red = Avg # of Check-Ins | Green = Mid of Check-Ins')
6 plt.vlines(x=table_admins_vs_nonadmins.total_users.mean(),ymin=table_admins_vs_nonadmins.total_users.min(),ymax=table_admins_vs_nonadmins.total_users.max(),color='red')
7 plt.vlines(x=table_admins_vs_nonadmins.total_users.median(),ymin=table_admins_vs_nonadmins.total_users.min(),ymax=table_admins_vs_nonadmins.total_users.max(),color='green')
```

OUT [25]



Queries for data sets listed below:

IN []

```
1 #####
2 ##### Task 1 #####
3 #####
4
5 # Task 1: [DATA] Events
6
7 select *
8 from arvindr12.dummy_events
9
10 # Task 1: [DATA] Users
11
12 select *
13 from arvindr12.dummy_users
```

IN [_]

```

1 #####
2 ##### Task 2 #####
3 #####
4
5
6 # Task 2: [TABLE] Frequency of Check-Ins
7
8
9 SELECT d.user_id,
10        sum(d.prior_check_in) AS total_prior_check_in,
11        NTILE(100) over (
12            ORDER BY sum(d.prior_check_in)) AS percentile
13 FROM
14     (SELECT c.user_id,
15            c.created_at,
16            c.event_type,
17            earliest_sign_up_date,
18            (CASE
19                WHEN c.created_at < earliest_sign_up_date then 1
20                ELSE null
21            END) AS prior_check_in --3. If an event was before the earliest sign up date, flag

```

● Ready



IN [_]

```
9 SELECT d.user_id,
10        sum(d.prior_check_in) AS total_prior_check_in,
11        NTILE(100) over (
12            ORDER BY sum(d.prior_check_in)) AS percentile
13 FROM
14     (SELECT c.user_id,
15            c.created_at,
16            c.event_type,
17            earliest_sign_up_date,
18            (CASE
19                WHEN c.created_at < earliest_sign_up_date then 1
20                ELSE null
21            END) AS prior_check_in --3. If an event was before the earliest sign up date, flag
22 FROM arvindr12.dummy_events AS c --2. Join back to events table
23 left join -- 1. Pull list of earliest sign up date by user
24
25     (SELECT a.user_id,
26            min(a.created_at) AS earliest_sign_up_date
27 FROM arvindr12.dummy_events AS a
28 WHERE a.event_type = 'sign_up'
29 group by 1) AS b
30 ON (b.user_id = c.user_id)
31 WHERE earliest_sign_up_date is not null ) AS d
32 GROUP BY 1
33 HAVING sum(d.prior_check_in) > 0
34 ORDER BY 2 ASC
35
36
37
38
```

Py

```
1 #####
2 ##### Task 3 #####
3 #####
4
5 # Task 3: [TABLE] Frequency - Time to Sign-Up
6
7 SELECT t1.user_id,
8        earliest_check_in_date,
9        earliest_sign_up_date,
10       age(earliest_sign_up_date,earliest_check_in_date) AS time_to_signup,
11       round(EXTRACT(epoch
12           FROM age(earliest_sign_up_date,earliest_check_in_date)/3600)) AS hours_signup,
13       NTILE(100) over (
14           ORDER BY (age(earliest_sign_up_date,earliest_check_in_date))) AS percentile
15 FROM -- Get earliest check in date
16
17     (SELECT a.user_id,
18            min(a.created_at) AS earliest_check_in_date
19 FROM arvindr12.dummy_events AS a
20 WHERE a.event_type = 'check_in'
21 GROUP BY 1) t1
22 inner join
23     (SELECT a.user_id,
24            min(a.created_at) AS earliest_sign_up_date
25 FROM arvindr12.dummy_events AS a
26 WHERE a.event_type = 'sign_up'
27 GROUP BY 1) t2
28 ON (t1.user_id = t2.user_id)
29 WHERE t1.earliest_check_in_date < t2. earliest_sign_up_date
30 ORDER BY 4 ASC
```

Py

IN [_]

```
1 #####
2 ##### Task 4 #####
3 #####
4
5 # Task 4A: % Signups by Admins
6 SELECT t2.is_admin,
```

● Ready

