Summary of Findings

I imported the users.csv first and then did some data cleaning for some exploratory analysis.

Dataset info

Number of variables	10
Number of observations	12000
Missing cells	8760 (7.3%)
Duplicate rows	0 (0.0%)
Total size in memory	937.6 KiB
Average record size in memory	80.0 B

Variables types

Numeric	4
Categorical	4
Boolean	2
Date	0
URL	0
Text (Unique)	0
Rejected	0
Unsupported	0

```
#creation time to dt
users['creation_time'] = pd.to_datetime(users['creation_time'])
```

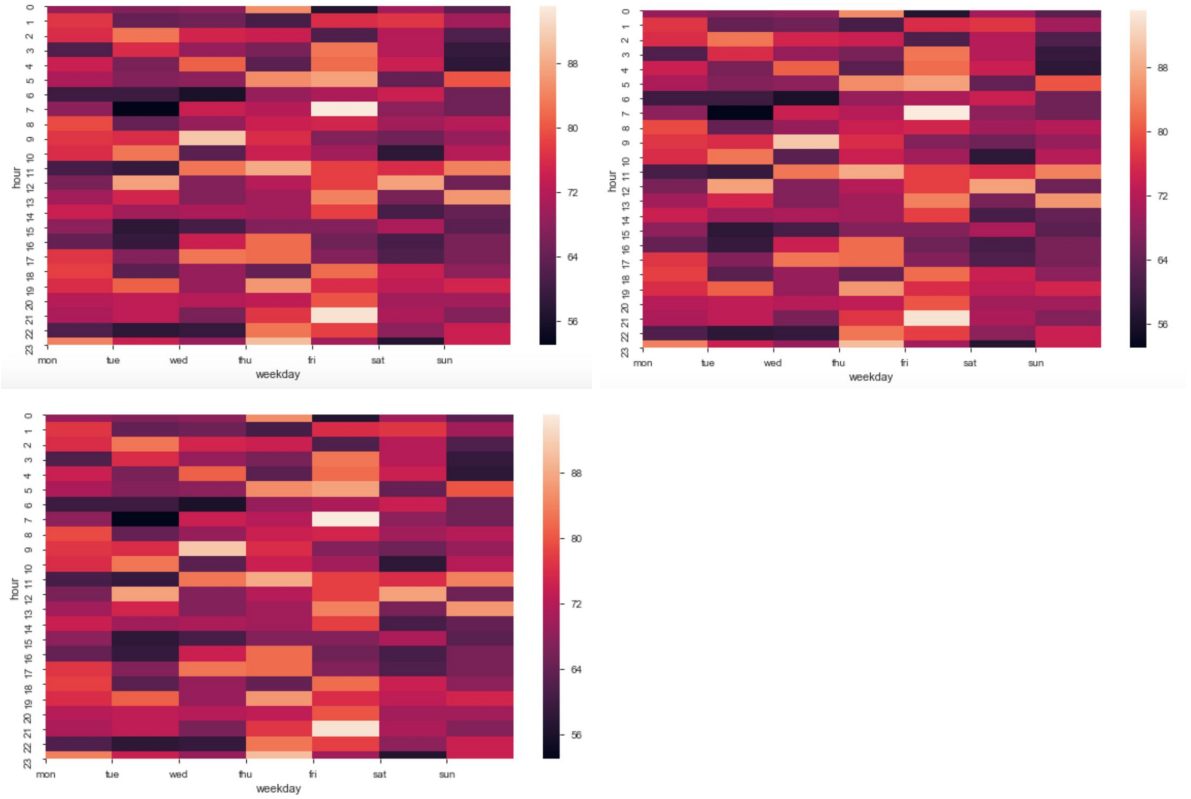
```
#creation source one hot encoding
oh = OneHotEncoder(use_cat_names=True, cols='creation_source')
users = oh.fit_transform(users)
```

```
#drop unnecessary cols
users = users.drop(['name', 'email'], axis=1)
```

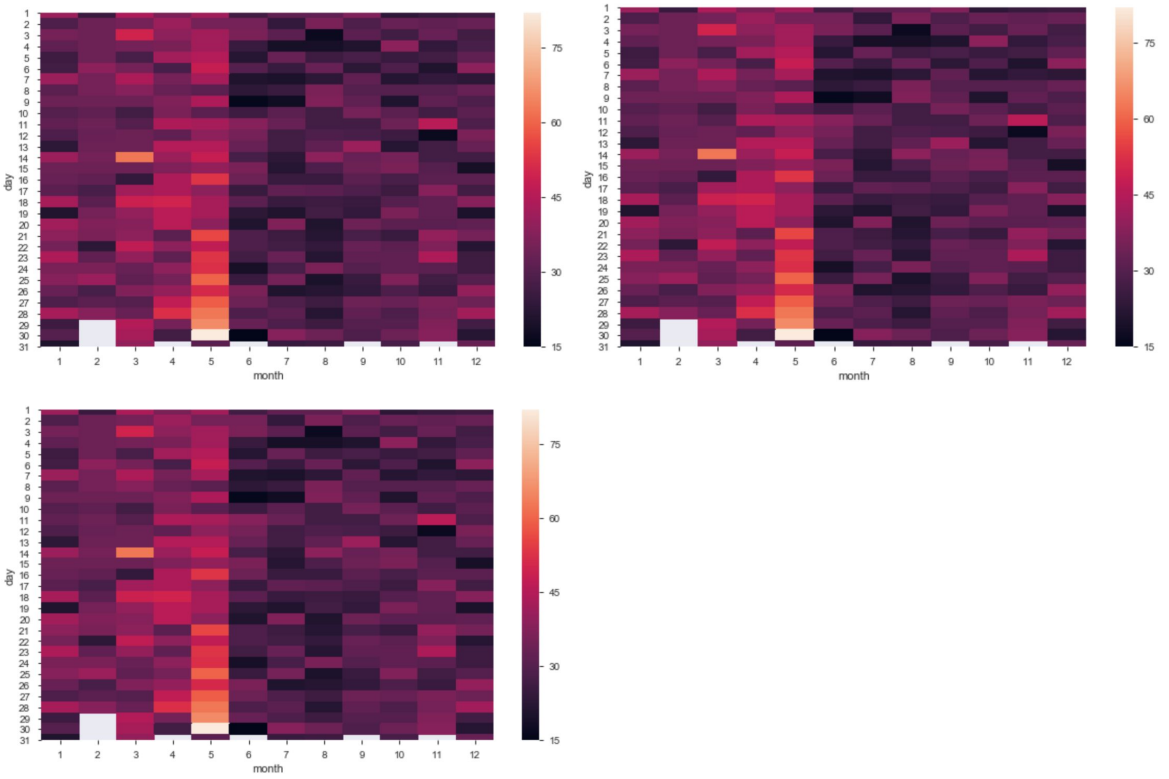
```
fill missing values
sers['last_session_creation_time'] = users['last_session_creation_time'].fillna(users['last_session_creation_time'].
```

```
users['invited_by_user_id'] = users['invited_by_user_id'].fillna(0)
```

Then for exploratory analysis, I listed hour, day, month and weekday in creation_time and tried some heatmap plots to see if there's any correlation between creation_source and creation_time.



There doesn't seem to be any patterns based on hour and weekday.



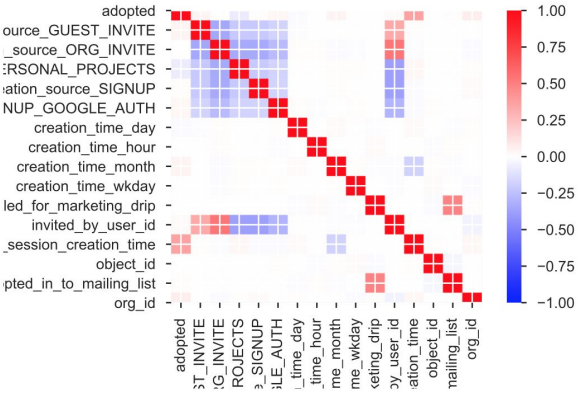
Same for day and month on creation_source, I didn't see any clear difference from the plots.

From user_engagement.csv I added adopted column to users to indicate if the user is an adopted user by the given definition. And the adopted rate is 13.7%.

o_mailing_list	enabled_for_marketing_drip	org_id	invited_by_user_id	creation_time_hour	creation_time_day	creation_time_month	creation_time_wkday	adopted
1	0	11	10803.0	3	22	4	1	0
0	0	1	316.0	3	15	11	4	1
0	0	94	1525.0	23	19	3	1	0
0	0	1	5151.0	8	21	5	1	0
0	0	193	5240.0	10	17	1	3	0

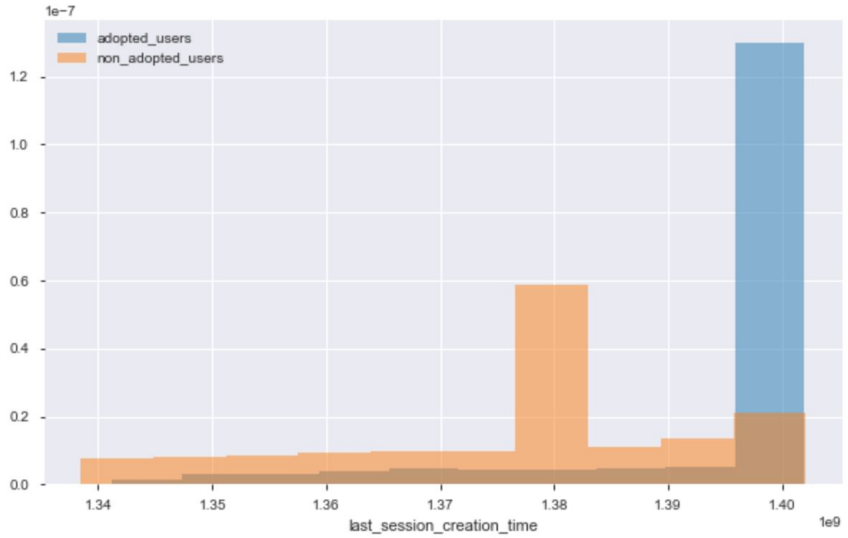
```
#adopted rate
users['adopted'].value_counts()[1]/users['adopted'].count()
```

```
0.137
```



From Pearson's correlation last_session_creation_time and adopted seems to be related.

Adopted users seem to have a different pattern of last session creation time than others based on the normed histogram.



Then I used ordinal encoder on creation_time to make the data more suitable for sklearn and applied select k best features with chi2 on the data.

```
k = SelectKBest(chi2, k=10)
X_k = k.fit_transform(X, Y)
X_k
```

```
array([[1.0000e+00, 1.0000e+00, 1.0000e+00, ..., 1.0803e+04, 2.2000e+01,
        4.0000e+00],
       [2.0000e+00, 2.0000e+00, 0.0000e+00, ..., 3.1600e+02, 1.5000e+01,
        1.1000e+01],
       [3.0000e+00, 3.0000e+00, 0.0000e+00, ..., 1.5250e+03, 1.9000e+01,
        3.0000e+00],
       ...,
       [1.1998e+04, 1.1994e+04, 1.0000e+00, ..., 8.0740e+03, 2.7000e+01,
        4.0000e+00],
       [1.1999e+04, 1.1995e+04, 0.0000e+00, ..., 0.0000e+00, 3.1000e+01,
        5.0000e+00],
       [1.2000e+04, 1.1996e+04, 0.0000e+00, ..., 0.0000e+00, 2.6000e+01,
        1.0000e+00]])
```

```
k.scores_
```

```
array([5.30000781e+02, 4.64214048e+02, 1.84388386e+01, 1.90873669e-01,
       6.93470641e-01, 5.69356025e+01, 1.36368605e+01, 3.11962569e+08,
       7.07195684e-01, 3.40690198e-01, 5.89623290e+03, 2.73823071e+04,
       9.31115875e-05, 1.90857081e+01, 3.74924601e+01, 6.60712871e-01])
```