In [1]:
```
import pandas as pd
import numpy as np
from scipy import stats
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from tqdm.notebook import tqdm
import re
pd.set_option('max_columns', None)
pd.options.display.max_colwidth = 100
```

In [2]:
```
df_groupby_user = pd.read_csv('data_groupby_user.csv', index_col=0)
df_groupby_user.shape
```

Out[2]: (21264, 3)

In [3]:
```
df = pd.read_csv('data.csv', index_col=0)
df.shape
```

Out[3]: (6643221, 4)

In this notebook we analyze agent churn from different perspectives.

1 Calculate user activity

In [ ]:
```
# keep only interested columns
df2 = df[['agent','bytes_returned']].copy()
# convert to datetime
df2['datetime'] = pd.to_datetime(df['timestamp'])
df2 = df2.set_index('datetime')
df2.head()
```

In [ ]:
```
# groupby agent and downsample into 1 hour bins
df2 = df2.groupby('agent').resample('60T').count()
df2 = df2.drop(columns=['agent'])
df2 = df2.reset_index()
df2 = df2.rename(columns={"datetime": "timestamp","bytes_returned": "request_count"})
# remove data from 01-03
df2 = df2[df2['timestamp'].dt.day==2]
# get hour from datetime
df2['hour'] = df2['timestamp'].dt.hour
df2 = df2.drop(['timestamp'], axis=1)
df2.shape
```

In [ ]:
```
df2.head()
```

In [ ]:
```
# pivot dataframe
df3 = df2.pivot(index='agent', columns='hour', values='request_count')
# fill NaN and change type to int
df3 = df3.fillna(0)
df3 = df3.astype(int)
df3.shape
```

In [ ]:
```
df3.head()
```

In [ ]:
```
df3.to_csv('data_user_activity.csv') # 4.5MB
```

2 Request heatmap

We draw a heatmap of the number of requests made by each agent in each hour. The x axis represents the agents, sorted in the order that they first appeared in the dataset. The y axis represents the hours of the day. The color of each cell in the heatmap indicates the number (in natural logarithmic scale) of requests made by a specific agent during a specific hour.

In [4]:
```
df4 = pd.read_csv('data_user_activity.csv', index_col=0)
df4.shape
```

Out[4]: (21262, 24)

In [5]:
```
# sort by the order users first appear in the data
df_agent = pd.DataFrame(df['agent'].unique())
df_agent.columns = ['agent']
df4 = df4.reindex(index=df_agent['agent'])
df4 = df4.reset_index()
# drop users with no request in the day
df4 = df4[~df4.isna().any(axis=1)]
# convert to int
df4 = df4.set_index('agent')
df4 = df4.astype(int)
df4 = df4.reset_index()
df4.shape
```

Out[5]: (21262, 25)

In [6]:
```
df4.set_index('agent',inplace=True)
# df4[df4 > 1] = 1
df4 = np.log(df4)
df4 = df4.replace(float('-inf'), 0)
df4 = df4.reset_index()

df4 = df4.reset_index()
df4['index'] = df4['index']/df4.shape[0]
df4 = df4.set_index('index')

df4.head()
```

Out[6]:

index	agent	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0.000000	axios/0.17.1	5.529429	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	8.442039	0.000000	0.000000	0.000000	0.000000	0.
0.000047	Mozilla/5.0 (Linux; U; Android 11; zh-cn; V2066A Build/RP1A.200720.012) AppleWebKit/537.36 (KHTM...	5.303305	0.000000	0.000000	4.615121	4.615121	0.000000	0.000000	3.258097	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
0.000094	Mozilla/5.0 (Linux; Android 11; V2046A; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 C...	6.056784	1.945910	0.000000	0.000000	5.225747	5.361292	1.791759	3.663562	2.995732	4.290459	2.302585	5.805135	0.000000	4.189655	4.007333	4.955827	4.
0.000141	Mozilla/5.0 (iPhone; CPU iPhone OS 14_7_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko...	7.024649	7.701652	5.541264	6.588926	6.300786	5.926926	6.551080	7.071573	7.768956	6.692084	6.208590	6.529419	5.062595	5.480639	6.436150	5.680173	5.
0.000188	GuzzleHttp/6.5.5 curl/7.68.0 PHP/7.4.3	7.853216	8.043984	7.549083	6.639876	6.576470	6.613384	6.216606	4.890349	0.000000	0.000000	0.000000	4.356709	5.123964	4.615121	4.634729	4.521789	4.

In [7]:
```
data = df4.drop(['agent'],axis=1)
data = data.T

fig = px.imshow(data, color_continuous_scale='turbo', origin='lower',
                labels=dict(color="number of request<br>an agent made per hour<br>(log scale)"),)

fig.update_xaxes(side='bottom')
fig.update_xaxes(title="agent (sorted by first occurrence)", tickformat = ',.0%')
fig.update_yaxes(title="hour")

fig.show()
```

3 Churn rate

Next, we look into the percentage of users who stay in the network after a specific time period.

In [8]:
```
# keep only interested columns
df3 = df[['agent']].copy()
# convert to datetime
df3['datetime'] = pd.to_datetime(df['timestamp'])
df3.head()
```

Out[8]:

	agent	datetime
0	axios/0.17.1	2022-01-02 00:00:38+00:00
1	Mozilla/5.0 (Linux; U; Android 11; zh-cn; V2066A Build/RP1A.200720.012) AppleWebKit/537.36 (KHTM...	2022-01-02 00:00:38+00:00
2	Mozilla/5.0 (Linux; Android 11; V2046A; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 C...	2022-01-02 00:00:38+00:00
3	Mozilla/5.0 (iPhone; CPU iPhone OS 14_7_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko...	2022-01-02 00:00:38+00:00
4	axios/0.17.1	2022-01-02 00:00:38+00:00

In [9]:
```
df4 = df3.groupby('agent').agg(['min', 'max', 'count'])
df4.columns = df4.columns.get_level_values(1)
df4[df4['count']==1].sum()
```

```
/var/folders/gh/hc3npzks3hq9y6jtyp23d8jm0000gn/T/ipykernel_3921/3749005683.py:3: FutureWarning:

Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.
Select only valid columns before calling the reduction.
```

Out[9]:
```
count    3610
dtype: int64
```

In [10]:
```
df4['last'] = df4['max'] - df4['min']
df4['minute'] = df4['last'].dt.seconds/60
df4['minute'] = df4['minute'].astype(int)
df4 = df4.drop(['min','max','last','count'],axis=1)
df4.reset_index()
df4.head()
```

Out[10]:

	minute
agent	
AVProMobileVideo/6.1.7.39280 (Linux;Android 10) ExoPlayerLib/2.15.0	0
AccompanyBot	643
ActionExtension/3 CFNetwork/1220.1 Darwin/20.3.0	0
AirPlay/2.0 (App/30.172.0) MFi_AirPlay_Device (MFiModelGroup/257872-0020)	55
AirPlay/2.0 (App/30.172.0) MFi_AirPlay_Device (MFiModelGroup/EIVU8BViYT0YUCNRKu1tWQNNxfpQUqz5a9U46rwjXGg)	234

In [11]:
```
df5 = pd.DataFrame(df4.value_counts(ascending=False))
df5 = df5.reset_index()
df5.columns = ['minute','count']
df5.head()
```

Out[11]:

	minute	count
0	0	7594
1	1	889
2	2	571
3	3	391
4	4	367

In [12]:
```
df4.shape
```

Out[12]:
```
(21262, 1)
```

In [13]:
```
df6 = pd.DataFrame(columns = ['stay_after_minute', 'count', 'name', 'hour'])

def addRow(df6, m, name):
    c = df5[df5['minute'] >= m].sum()['count']
    df6 = df6.append({'stay_after_minute' : m, 'count' : c, 'name':name, 'hour':m/60},
        ignore_index = True)
    return df6
```

In [14]:
```
df6 = addRow(df6, 1, "1m")
df6 = addRow(df6, 2, "2m")
df6 = addRow(df6, 4, "4m")
df6 = addRow(df6, 8, "8m")
df6 = addRow(df6, 16, "16m")
df6 = addRow(df6, 60, "1h")
df6 = addRow(df6, 120, "2h")
df6 = addRow(df6, 240, "4h")
df6 = addRow(df6, 480, "8h")
df6 = addRow(df6, 960, "16h")
df6 = addRow(df6, 1439, "24h")
```

In [15]:
```
df6['percentage'] = df6['count']/21985
# df6['stay_after_minute'] = df6['stay_after_minute'].astype(str)
```

In [16]:
```
df6.head()
```

Out[16]:

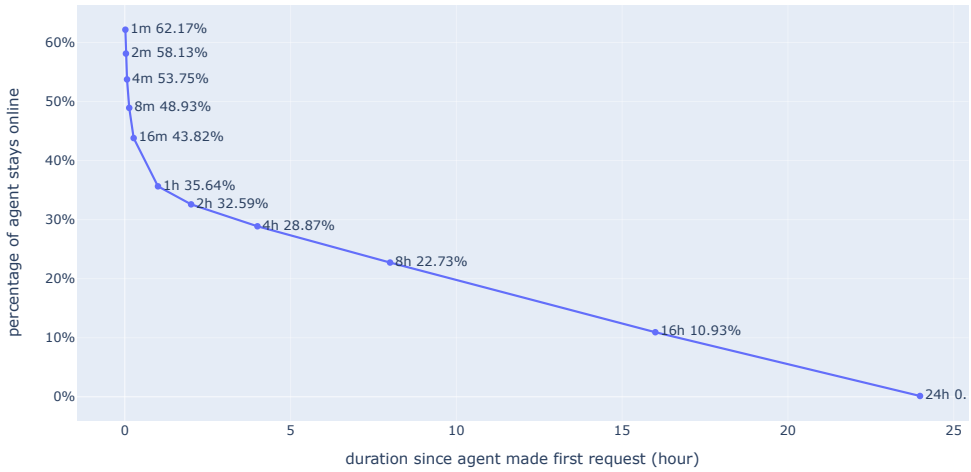
	stay_after_minute	count	name	hour	percentage
0	1	13668	1m	0.016667	0.621697
1	2	12779	2m	0.033333	0.58126
2	4	11817	4m	0.066667	0.537503
3	8	10757	8m	0.133333	0.489288
4	16	9633	16m	0.266667	0.438162

In [17]:
```
fig = px.line(df6, x='hour', y='percentage',
              text = [str(x[0])+' {0:1.2f}%'.format(x[1]*100) for x in zip(df6['name'],df6['percentage'])])

fig.update_traces(textposition="middle right")

fig.update_xaxes(title='duration since agent made first request (hour)')
fig.update_yaxes(title='percentage of agent stays online', tickformat=',.0%')

fig.show()
```



4 Users leave in 1 min

Next, we only keep the users who disconnect from the network within a minute and count the number of requests they made.

In [18]:
```
df4.head()
```

Out[18]:

agent	minute
AVProMobileVideo/6.1.7.39280 (Linux;Android 10) ExoPlayerLib/2.15.0	0
AccompanyBot	643
ActionExtension/3 CFNetwork/1220.1 Darwin/20.3.0	0
AirPlay/2.0 (App/30.172.0) MFi_AirPlay_Device (MFiModelGroup/257872-0020)	55
AirPlay/2.0 (App/30.172.0) MFi_AirPlay_Device (MFiModelGroup/EIVU8BViYT0YUCNRKu1tWQNNxfpQUqz5a9U46rwjXGg)	234

In [19]:
```
df9 = df4[df4['minute']<1]
df9.shape
```

Out[19]:
```
(7594, 1)
```

In [20]:
```
df9 = df9.reset_index()
agent_set = df9['agent'].unique()
```

In [21]:
```
df10 = df[df['agent'].isin(agent_set)]
df10.shape
```

Out[21]:
```
(34646, 4)
```

In [22]:
```
df.shape
```

Out[22]:
```
(6643221, 4)
```

In [23]:
```
df10.head()
```

Out[23]:

	timestamp	bytes_returned	agent	cid
95	2022-01-02T00:00:40+00:00	825520	Mozilla/5.0 (Linux; U; Android 9; zh-cn; COR-AL00 Build/HUAWEICOR-AL00) AppleWebKit/537.36 (KHTM...	QmZB8awpNvtuSP6JgVNam5KNEFfrx3d2YFvHTvddggUEBx
131	2022-01-02T00:00:40+00:00	1648960	Mozilla/5.0 (Linux; U; Android 9; zh-cn; COR-AL00 Build/HUAWEICOR-AL00) AppleWebKit/537.36 (KHTM...	QmZB8awpNvtuSP6JgVNam5KNEFfrx3d2YFvHTvddggUEBx
417	2022-01-02T00:00:44+00:00	769136	PlaySDK/10.3.18.0 (Linux;Android 5.1.1) ExoPlayerLib/2.8.2	bafybeigz4jdkoxq5yyv2p36iy6eyfa5bq7be5lnjiytdqywg5mqsihb3me
599	2022-01-02T00:00:46+00:00	943	Mozilla/5.0 (Linux; Android 9; VKY-AL00 Build/HUAWEIVKY-AL00; wv) AppleWebKit/537.36 (KHTML, lik...	bafybeiabasj5jhi2ghc3eu3eoj6ii7cgewoibjd6zat4royqa7ctmxwlf4
647	2022-01-02T00:00:47+00:00	131117	Mozilla/5.0 (Linux; Android 9; VKY-AL00 Build/HUAWEIVKY-AL00; wv) AppleWebKit/537.36 (KHTML, lik...	bafybeiabasj5jhi2ghc3eu3eoj6ii7cgewoibjd6zat4royqa7ctmxwlf4

In [24]:
```
df11_1 = df10[['agent','cid']].groupby('agent').agg(['count',pd.Series.nunique])
df11_1.columns = df11_1.columns.get_level_values(1)
df11_2 = df10[['agent','bytes_returned']].groupby('agent').agg('mean')
df11 = df11_1.join(df11_2, lsuffix='agent', rsuffix='agent')
df11['bytes_returned'] = df11['bytes_returned']/1024
df11 = df11.rename(columns={"count": "cid_count", "nunique": "cid_unique", "bytes_returned": "KB_returned_mean"})
df11.head()
```

Out[24]:

agent	cid_count	cid_unique	KB_returned_mean
AVProMobileVideo/6.1.7.39280 (Linux;Android 10) ExoPlayerLib/2.15.0	1	1	6474.051758
ActionExtension/3 CFNetwork/1220.1 Darwin/20.3.0	5	5	309.818945
AlphaWallet/417 CFNetwork/1240.0.4 Darwin/20.6.0	1	1	0.000000
Android.Thunder.Mozilla/5.0	1	1	16.053711
Android.Thunder.Mozilla/5.0 (Linux; Android 6.0.1; KIW-TL00H Build/HONORKIW-TL00H; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/55.0.2883.91 Mobile Safari/537.36	36	2	777.028076

In [25]:
```
df11.describe()
```

Out[25]:

	cid_count	cid_unique	KB_returned_mean
count	7594.000000	7594.000000	7594.000000
mean	4.562286	1.844878	429.125095
std	18.299832	5.400114	2166.155710
min	1.000000	1.000000	0.000000
25%	1.000000	1.000000	9.869141
50%	2.000000	1.000000	29.614258
75%	4.000000	2.000000	255.000977
max	1381.000000	300.000000	98218.436523

In [26]:
```
df11 = df11.sort_values(by=['cid_count','cid_unique'], ascending=False)
df11 = df11.reset_index()
df11 = df11.reset_index()
df11.head()
```

Out[26]:

	index	agent	cid_count	cid_unique	KB_returned_mean
0	0	Python/3.8 aiohttp/3.8.1	1381	1	0.330449
1	1	Mozilla/5.0 (iPhone; CPU iPhone OS 15_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) ...	300	300	141.134505
2	2	Mozilla/5.0 (Linux; Android 10; HarmonyOS; NOH-AN01; HMSCore 6.2.0.302) AppleWebKit/537.36 (KHTM...	163	6	1233.454377
3	3	Mozilla/5.0 (iPhone; CPU iPhone OS 15_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) ...	148	148	140.877125
4	4	Mozilla/5.0 (Linux; Android 8.0; Galaxy Nexus Build/IMM76B) AppleWebKit/535.19 (KHTML, like Geck...	142	2	1400.540548

In [27]:
```
df12 = pd.DataFrame(columns = ['request', 'count'])

def addRow(df12, l, r):
    df_temp = df11[(df11['cid_count'] >= l) & (df11['cid_count'] < r)]
    c = df_temp.count()[0]
    df12 = df12.append({'request':'['+str(l)+','+str(r)+')', 'count':c}, ignore_index = True)
    return df12

df12 = addRow(df12, 1, 2)
df12 = addRow(df12, 2, 10)
df12 = addRow(df12, 10, 100)
df12 = addRow(df12, 100, 1000)
df12 = addRow(df12, 1000, 10000)

df12 = df12.replace('[1,2)', '1')
df12 = df12.replace('[1000,10000)', '[1000,+∞)')

total = df12['count'].sum()
df12['percentage'] = df12['count']/total
df12
```

Out[27]:

	request	count	percentage
0	1	3610	0.475375
1	[2,10)	3186	0.419542
2	[10,100)	787	0.103634
3	[100,1000)	10	0.001317
4	[1000,+∞)	1	0.000132

In [28]:
```
fig = px.bar(df12, x='request', y='count', text=[str(x[0])+'\n{0:1.2f}%'.format(x[1]*100) for x in zip(df12['count'],df12['percentage'])])

fig.update_xaxes(title='Range of request count per agent (left within a minute)')
fig.update_yaxes(title='agent count')

fig.show()
```