# Neuro Flow Submission

Author: Sarvesh Shah

In [468]:

```
# import statemenrs
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re
import datetime as dt
```

In [469]:

```
# Read File
df = pd.read_csv('Documents/subj_measures.csv')
display(df.head())
df.dtypes
```

|  | date | user_id | type | value |
|---|---|---|---|---|
| 0 | 2019-06-08T23:19:34.418Z | 2348 | mood | 2.76 |
| 1 | 2019-06-13T16:33:34.399Z | 5232 | sleep | 2.44 |
| 2 | 2018-12-26T14:24:00.436Z | 4209 | sleep | 1.88 |
| 3 | 2019-07-17T20:11:23.792Z | 2802 | mood | 2.20 |
| 4 | 2019-05-09T17:27:50.900Z | 2025 | mood | 4.00 |

Out[469]:

```
date         object
user_id       int64
type         object
value       float64
dtype: object
```

In [470]:

```
df['user_id'] = df['user_id'].astype(str)
df['date'] = pd.to_datetime(df['date'])
df.describe()
```

Out[470]:

|  | value |
|---|---|
| count | 7460.000000 |
| mean | 2.406454 |
| std | 1.153750 |
| min | 0.000000 |
| 25% | 1.599209 |
| 50% | 2.400000 |
| 75% | 3.400000 |
| max | 4.000000 |

In [8]:

```
# Checking for null values
df.isna().sum()
```

Out[8]:

```
date       0
user_id    0
type       0
value      0
dtype: int64
```

In [472]:

```python
# Adding extra columns for better aggregation
df['month'] = df['date'].dt.month
df['year'] = df['date'].dt.year
df['day'] = df['date'].dt.day
df['hour'] = df['date'].dt.hour
```

## When do users report their scores?

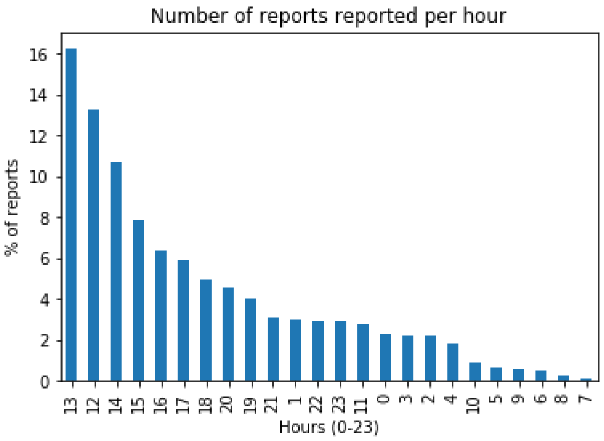
We were interested in looking at the reporting patterns for users.
In plot 1, we see that the users like to report at 1 pm.
About **16%** of our users report at **1 pm** and about **40%** of ours users report their scores from **12pm to 3pm**.
In plot 2, We also looked at the change in trend between two years 2018 and 2019, but nothing siginifant was observed. Thus, we can assume that the user behavior has largely remain same in this time frame.

In [494]:

```python
# Plot 1
# Looking at the time when users report their scores

(df['hour'].value_counts(normalize=True,sort=True)*100).plot(kind='bar')
plt.title('Number of reports reported per hour')
plt.xlabel('Hours (0-23)')
plt.ylabel('% of reports')
```

Out[494]:

```
Text(0, 0.5, '% of reports')
```

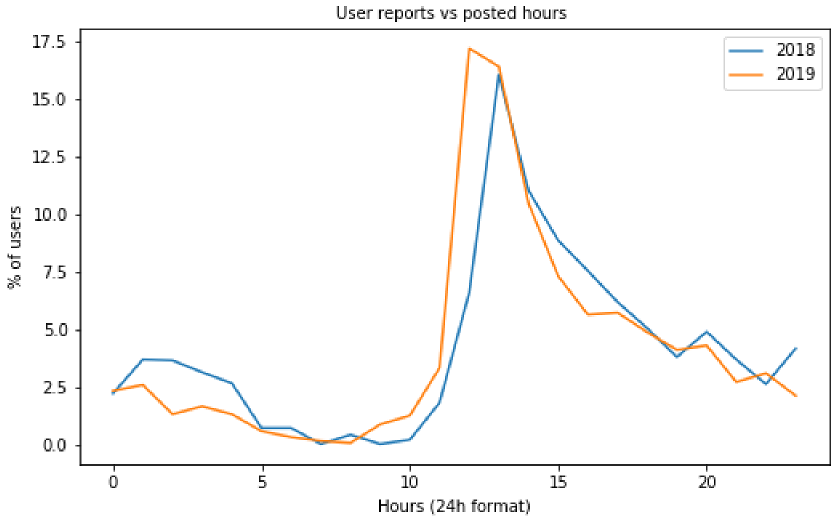


In [445]:

```python
# Plot 2
# Breaking down the users by year
df1 = df.groupby(['year','hour'])[['user_id']].count()

plt.figure(figsize=(5*1.68,5))
plt.plot(df1.loc[2018,:]['user_id']*100/sum(df1.loc[2018,:]['user_id']),label='2018')
plt.plot(df1.loc[2019,:]['user_id']*100/sum(df1.loc[2019,:]['user_id']),label='2019')
plt.title('User reports vs posted hours ',fontsize=10)
plt.xlabel('Hours (24h format)',fontsize=10)
plt.ylabel('% of users',fontsize=10)
plt.legend(fontsize=10)

# Found nothing significant, the trends are similar
```

```
Out[445]:
```

```
<matplotlib.legend.Legend at 0x1bd487bdfc8>
```



The chart shows, similar reporting patterns for both the years, indicating that nothing has changed in the user behavior and how they use our app.

### Differences in reporting type between the different types

We see from the table below that about **3400** reports have been registered for mood and sleep each, but both the stress reports are around **300**

```
In [14]:
```

```
# Looking at number of users reporting different types

df.groupby(['type']).agg({
    'user_id':'count',
    'value':'mean'
})
```

```
Out[14]:
```

| | user_id | value |
|---|---|---|
| **type** | | |
| **anticipatoryStress** | 332 | 2.495910 |
| **mood** | 3397 | 2.404216 |
| **ruminationStress** | 316 | 2.353558 |
| **sleep** | 3415 | 2.404878 |

### Hourly distribution by each behavior type

Next we broke down the behavior type by each reporting type to see, if any one particular type is driving the trend or not

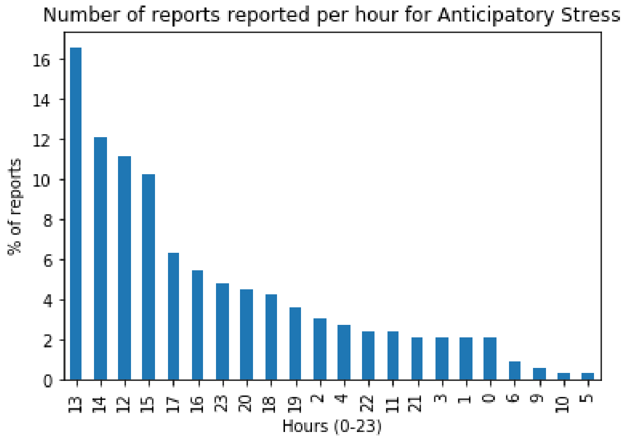
### Anticipatory Stress

```
In [498]:
```

```
(df[df.type=='anticipatoryStress']['hour'].value_counts(normalize=True)*100).plot(kind='bar')
plt.title('Number of reports reported per hour for Anticipatory Stress')
plt.xlabel('Hours (0-23)')
plt.ylabel('% of reports')
```

```
Out[498]:
```

```
Text(0, 0.5, '% of reports')
```



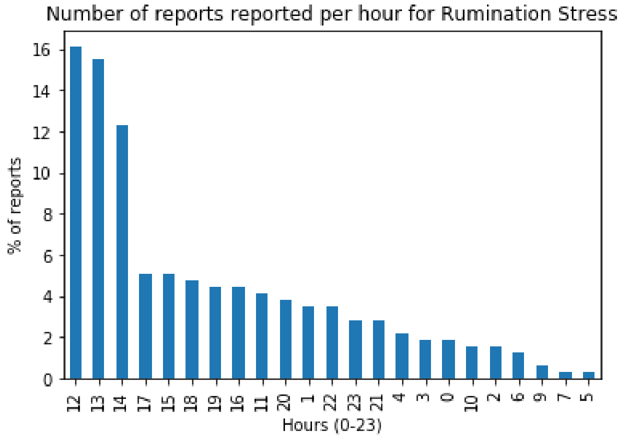
### Rumination Stress

In [497]:

```
(df[df.type=='ruminationStress']['hour'].value_counts(normalize=True)*100).plot(kind='bar')
plt.title('Number of reports reported per hour for Rumination Stress')
plt.xlabel('Hours (0-23)')
plt.ylabel('% of reports')
```

Out[497]:

```
Text(0, 0.5, '% of reports')
```



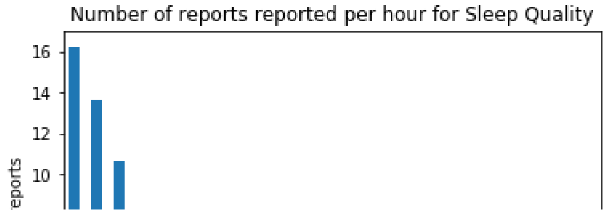
### Sleep

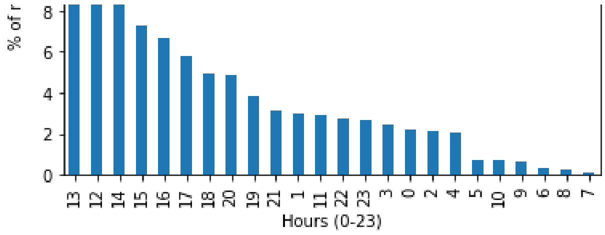
In [496]:

```
(df[df.type=='sleep']['hour'].value_counts(normalize=True)*100).plot(kind='bar')
plt.title('Number of reports reported per hour for Sleep Quality')
plt.xlabel('Hours (0-23)')
plt.ylabel('% of reports')
```

Out[496]:

```
Text(0, 0.5, '% of reports')
```





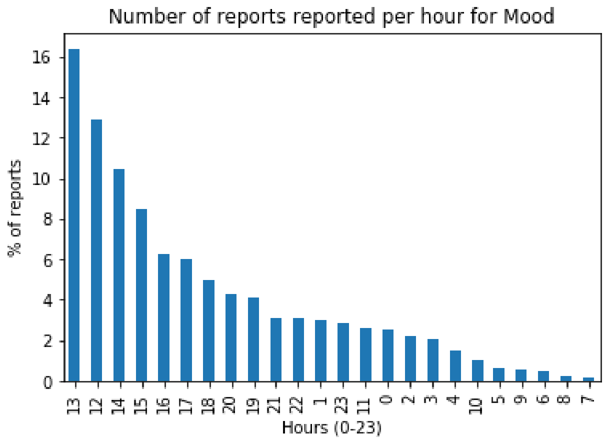
### Mood

In [495]:

```
(df[df.type=='mood']['hour'].value_counts(normalize=True)*100).plot(kind='bar')
plt.title('Number of reports reported per hour for Mood')
plt.xlabel('Hours (0-23)')
plt.ylabel('% of reports')
```

Out[495]:

```
Text(0, 0.5, '% of reports')
```



We see similar behavior for reporting for all types. Thus we further slice the data

### Number of users reporting by the month and the mean score, broken down by type

In [473]:

```
df1 = df.groupby(['year','month','user_id','type']).agg({
    'user_id':'count',
    'value':'mean'
})

df1.head()
```

Out[473]:

| | | | | user_id | value |
|---|---|---|---|---|---|
| **year** | **month** | **user_id** | **type** | | |
| **2018** | **8** | **1109** | **mood** | 1 | 0.040000 |
| | | **1173** | **mood** | 1 | 1.720000 |
| | | **1342** | **ruminationStress** | 1 | 3.360000 |
| | | **2025** | **mood** | 1 | 0.019238 |
| | | **2666** | **mood** | 2 | 3.600000 |

In [502]:

```
df1 = df.groupby(['user_id','type']).agg({
    'user_id':'count'
}).unstack().fillna(0)

df1.columns = df1.columns.droplevel(0)

df1.head()
```

Out[502]:

| type | anticipatoryStress | mood | ruminationStress | sleep |
|---|---|---|---|---|
| **user_id** | | | | |
| **1044** | 10.0 | 126.0 | 11.0 | 132.0 |
| **1074** | 6.0 | 90.0 | 9.0 | 77.0 |
| **1109** | 6.0 | 73.0 | 9.0 | 42.0 |
| **1140** | 4.0 | 22.0 | 4.0 | 17.0 |
| **1173** | 0.0 | 12.0 | 2.0 | 10.0 |

In [503]:

```
# Number of users who've never reported stress

print(df1[df1['anticipatoryStress']==0].shape[0])
print(df1[df1['ruminationStress']==0].shape[0])
print(df1[df1['mood']==0].shape[0])
print(df1[df1['sleep']==0].shape[0])
```

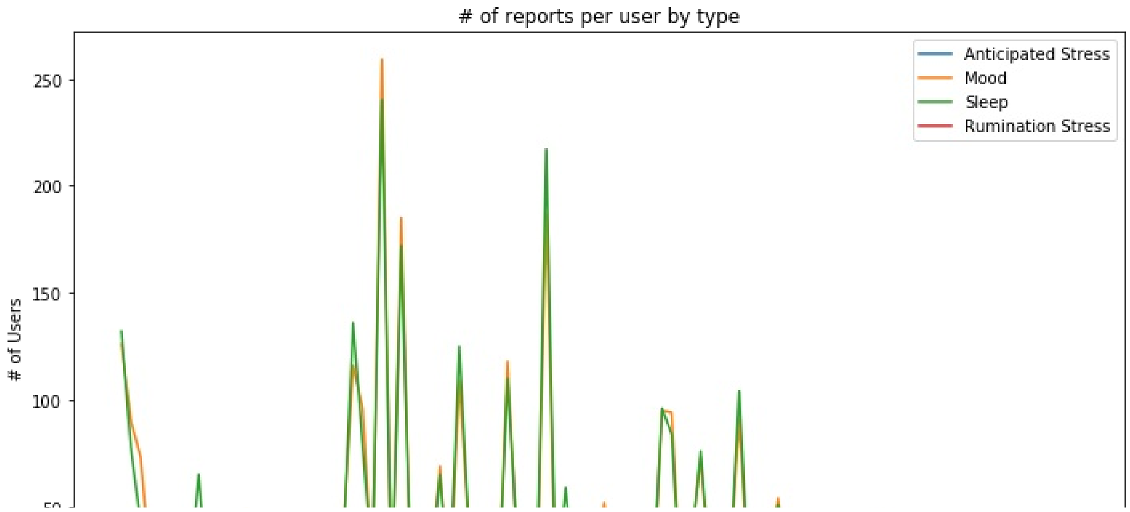
```
13
17
0
0
```

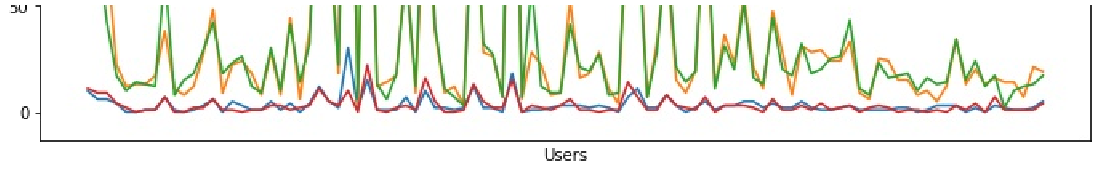
In [504]:

```
plt.figure(figsize=(7*1.68,7))
plt.plot(df1['anticipatoryStress'],label='Anticipated Stress')
plt.plot(df1['mood'],label='Mood')
plt.plot(df1['sleep'],label='Sleep')
plt.plot(df1['ruminationStress'],label='Rumination Stress')

plt.title('# of reports per user by type')
plt.xlabel('Users')
plt.ylabel('# of Users')
plt.tick_params(axis='x', which='both', bottom=False, top=False, labelbottom=False)

plt.legend()
```

Out[504]:

```
<matplotlib.legend.Legend at 0x1bd4c315088>
```





We can see that the number of reports reported by users vary drastically accross the board and most of the users do not report stress. We further hypothesize potential reasons and recommendations to improve the same.

## Progression Tracking

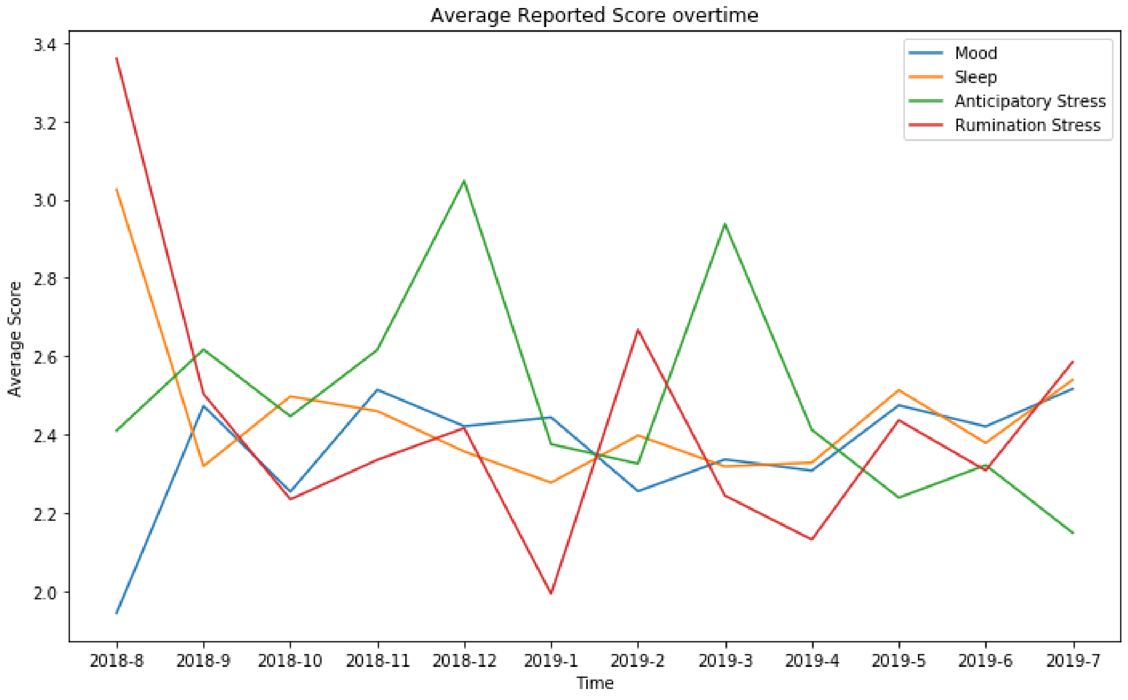
The prompt mentioned that visualizing this data is difficult. We tried visualizing the average trend for all users and the average reported score for the given range of dates.
The is a high variance in the average score from month to month and we suspect the day or weekly aggregation to yeild more noisy curves. This could be attributed to the fact that the number of users reporting daily is not as constant as desired and some users input extreme values and thus skweing the data for that given point.

In [499]:

```
df1 = df.groupby(['type','year','month'])[['value']].mean()

fig = plt.figure(figsize=(7*1.68,7))

ax = fig.add_subplot(111)
ind = np.arange(len(df1.index.levels[2]))

rects1 = ax.plot(ind,df1.loc['mood',:])
rects1 = ax.plot(ind,df1.loc['sleep',:])
rects1 = ax.plot(ind,df1.loc['anticipatoryStress',:])
rects1 = ax.plot(ind,df1.loc['ruminationStress',:])

ax.set_title('Average Reported Score overtime')
ax.set_ylabel('Average Score')
ax.set_xlabel('Time')
ax.set_xticks(ind)
ax.set_xticklabels(df1.loc['mood',:].reset_index()['year'].map(str) + "-" +df1.loc['mood',:].reset_
index()['month'].map(str))

ax.legend(['Mood', 'Sleep', 'Anticipatory Stress', 'Rumination Stress'])
```

Out[499]:

```
<matplotlib.legend.Legend at 0x1bd4c2902c8>
```



We look at two of the peaks specifically to see what's going on here, we see that there is a high variance in the reported scores and thus cannot draw any conclusions.

In [460]:

```
# Kindly remove head() to see the full data
df[(df.year==2018)&(df.month==12)&(df.type=='anticipatoryStress')].head()
```

Out[460]:

| | date | user_id | type | value | month | year | day | hour | rating |
|---|---|---|---|---|---|---|---|---|---|
| **527** | 2018-12-03 15:57:16.835000+00:00 | 3182 | anticipatoryStress | 0.870763 | 12 | 2018 | 3 | 15 | awful |
| **750** | 2018-12-13 01:54:37.973000+00:00 | 3609 | anticipatoryStress | 3.960000 | 12 | 2018 | 13 | 1 | good |
| **950** | 2018-12-14 14:57:43.663000+00:00 | 2802 | anticipatoryStress | 2.840000 | 12 | 2018 | 14 | 14 | okay |
| **1264** | 2018-12-10 14:00:32.732000+00:00 | 1269 | anticipatoryStress | 1.360000 | 12 | 2018 | 10 | 14 | bad |
| **1602** | 2018-12-13 15:08:08.461000+00:00 | 3435 | anticipatoryStress | 3.683807 | 12 | 2018 | 13 | 15 | good |

In [461]:

```
# Kindly remove head() to see the full data
df[(df.year==2019)&(df.month==1)&(df.type=='ruminationStress')].head()
```

Out[461]:

| | date | user_id | type | value | month | year | day | hour | rating |
|---|---|---|---|---|---|---|---|---|---|
| **431** | 2019-01-28 03:32:43.567000+00:00 | 2432 | ruminationStress | 1.400000 | 1 | 2019 | 28 | 3 | bad |
| **504** | 2019-01-16 18:58:52.222000+00:00 | 2025 | ruminationStress | 2.000000 | 1 | 2019 | 16 | 18 | bad |
| **514** | 2019-01-30 22:57:48.549000+00:00 | 1522 | ruminationStress | 0.120000 | 1 | 2019 | 30 | 22 | awful |
| **908** | 2019-01-30 13:04:38.623000+00:00 | 3435 | ruminationStress | 4.000000 | 1 | 2019 | 30 | 13 | good |
| **1299** | 2019-01-03 13:57:44.284000+00:00 | 6450 | ruminationStress | 0.946256 | 1 | 2019 | 3 | 13 | awful |

# Recommendations and Conclusion

Due to time constraint of 3 hours, we kept the scope of our analysis pretty narrow and thus did some exploratory analysis to get the sense of the data.

**Trend Analysis by hour of the day**
The trend analysis of 'reports per hour', defined below indicate that the maximum number of reports is reported at 1 pm, which accounts for 16% of our total reports. Followed by, 13.5% around 12pm and 10% around 2pm. On an average, 40% of all of our reports typically are recorded from 12pm to 3pm.
*Reports per hour = Total Number of reports recorded in a given hour (0-23)*
Further, we broke these reporting by specific types to see if there is a driving factor, but found nothing significant in numbers. We supsect that users might not prioritize reporting in early morning as they might have their early morning routine and jobs.

**Analysis by type of reporting**
We see that only 10% of our reports consist of stress reports. On a user level, 13% of our users never report Anticipated Stress, and 17% of our users have never report Ruminant Stress. And only 4% of user have never reported either the stress types. Combining this observation with the above finding, it can be infered that while most users do report stress, the number of times they have reported the stress is low.

**Recommendations**
It is unclear if the users get any notification reminder to post their stress reports. We cuurently assume that they do not get any stress report and thus would recommend that Neuroflow, might send reminder notification if the user does not report stress for a few days. This could be done by running an A/B test and testing impact of notification and their reporting habits over a period of few weeks. For the users that do report stress, we could do a user study with focus groups or interviews to understand the user experience and map their thought process to understand how they interact with app and use that data to improve our app. For instance, maybe the scale is confusing. It is tedious to report stress. One of the other factor could be that users simply forget to report stress when they're in stressful situation and or are unwilling to share that they're stressed.

**Additional Information to enhance the study**

Other information like, age, gender, occupation and lifestyle could help us understand more about why the users report the way they do. We also need information about how long the user has been using our platform as it helps us put the numbers and trend in context, for instance if the number of users in 2019 are more than in 2018, we can be more certain about the impact of the app on their health state in 2019. We could also use information about their work routine and sleep routine to better understand when is the best time for them to report and customize our notifications based on that. With additional information as described above, we can do some statistical modelling and parameterize these factors to see what influences the reported score.