# DATA MINING - PROJECT 1

# PROJECT OUTLINE

**Data Context:**

Lifelogging Data of Elderly Citizens

**Provided Data:**

User Information, Device Uplink Data

**Project Goal:**

Explore and Preprocess Data

## Processing Steps

- Data Exploration
- Missing Data Management
- Error Handling
- Dimension Reduction
- Visualization

# 1 Data Exploration

# Sample Given Data

```
In [3]: users_df.sample(5)
```

Out[3]:

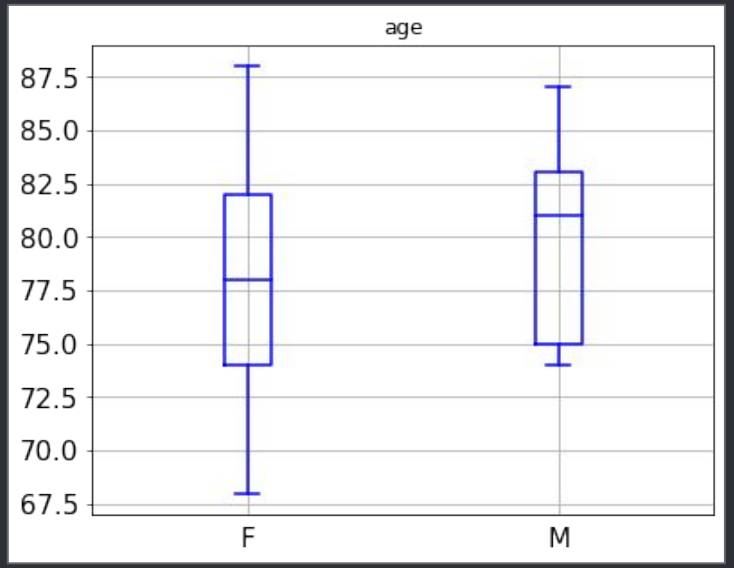
| | id | birth year | age | sex | etc |
|---|---|---|---|---|---|
| **25** | 583 | 1948 | 74 | F | 무자녀 |
| **8** | 503 | 1940 | 82 | F | 난청이 있으며 신경통약 복용 |
| **22** | 569 | 1947 | 75 | F | NaN |
| **48** | 574 | 1947 | 75 | M | NaN |
| **4** | 486 | 1937 | 85 | F | 혈압, 관절염, 허리다침(우울증) |

```
In [4]: uplink_df.sample(5)
```

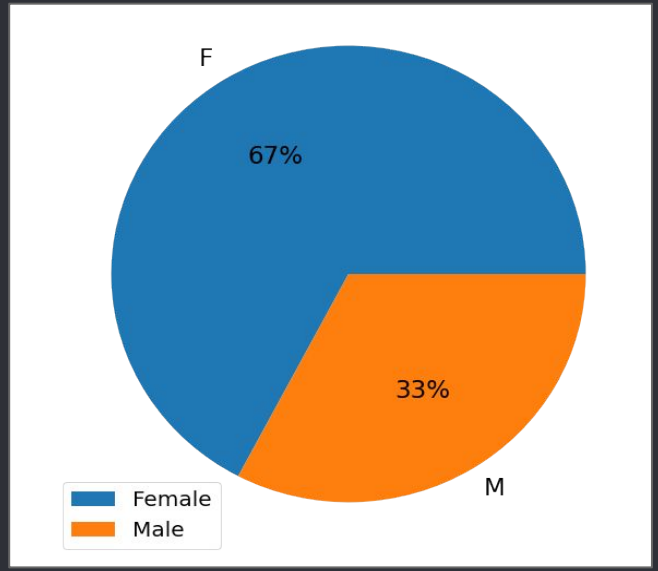
Out[4]:

| | uplink_id | owner_id | client_time | tag_id | step | battery_low | is_charge | tag_battery_low |
|---|---|---|---|---|---|---|---|---|
| **230476** | 2462046 | 585 | 2020-12-11 22:40:11 | 0.0 | 2598 | 59 | 0 | 0 |
| **292655** | 3287467 | 650 | 2021-02-11 14:28:29 | 10.0 | 1991 | 12 | 0 | 0 |
| **320990** | 3736567 | 486 | 2021-03-09 08:58:02 | 0.0 | 1933 | 81 | 0 | 0 |
| **29637** | 720161 | 486 | 2020-08-19 17:10:00 | 0.0 | 5579 | 96 | 1 | 0 |
| **19831** | 601087 | 574 | 2020-08-12 07:27:53 | 9.0 | 273 | 54 | 1 | 0 |

# User Composition




Age Distribution by Sex




User Composition

# Inspect and Convert Data Types

```
In [5]: users_df.dtypes

Out[5]: id             int64
        birth year     int64
        age            int64
        sex           object
        etc           object
        dtype: object
```

```
In [5]: uplink_df.dtypes

Out[5]: uplink_id            int64
        owner_id             int64
        client_time         object
        tag_id             float64
        step                 int64
        battery_low          int64
        is_charge            int64
        tag_battery_low      int64
        dtype: object
```

```
In [6]: # Convert client_time values to datetime
        uplink_df['client_time'] = pd.to_datetime(uplink_df['client_time'])
```

# 2 Missing Data Management

# Inspect for NAN-values

```
In [7]: uplink_df.count()

Out[7]: uplink_id          324823
        owner_id           324823
        client_time        324823
        tag_id             323617
        step               324823
        battery_low        324823
        is_charge          324823
        tag_battery_low    324823
        dtype: int64

In [8]: users_df.count()

Out[8]: id            52
        birth year    52
        age           52
        sex           52
        etc           33
        dtype: int64
```

Use Pandas in-built count() function to inspect for NAN-values

# Replace NAN-values with Appropriate Value

```
In [9]: # #Replace Nan in etc with empty string
        users_df['etc'] = users_df['etc'].fillna(value='')
```

1. Replace missing ‘etc’ values with empty string

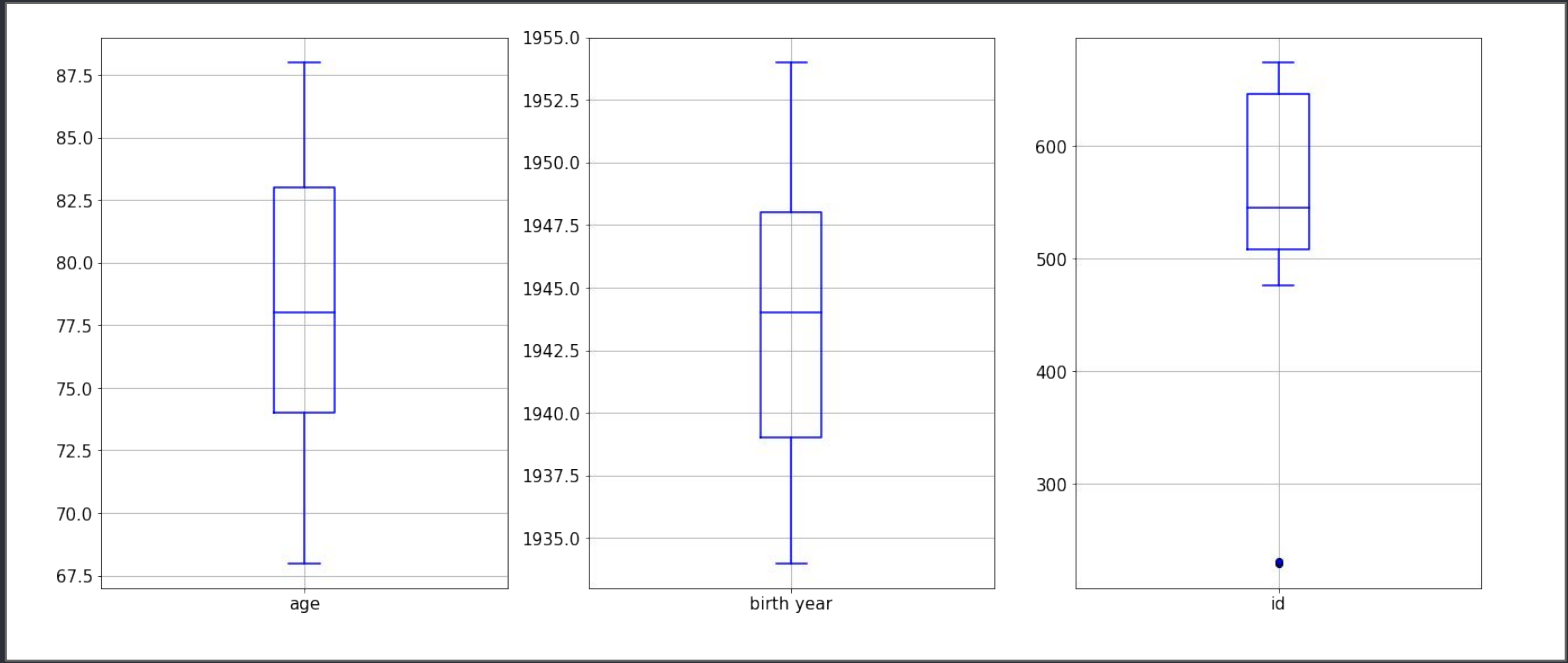
```
In [10]: #Replace NAN-values with zero values according to specific data set convention
         uplink_df['tag_id'] = uplink_df['tag_id'].fillna(value=0.0)
         uplink_df['tag_id'] = [int(n) for n in uplink_df['tag_id']]
```

1. Replace missing tag id values with zero value
2. Preserve additional information of log entry while marking tag id as faulty

3

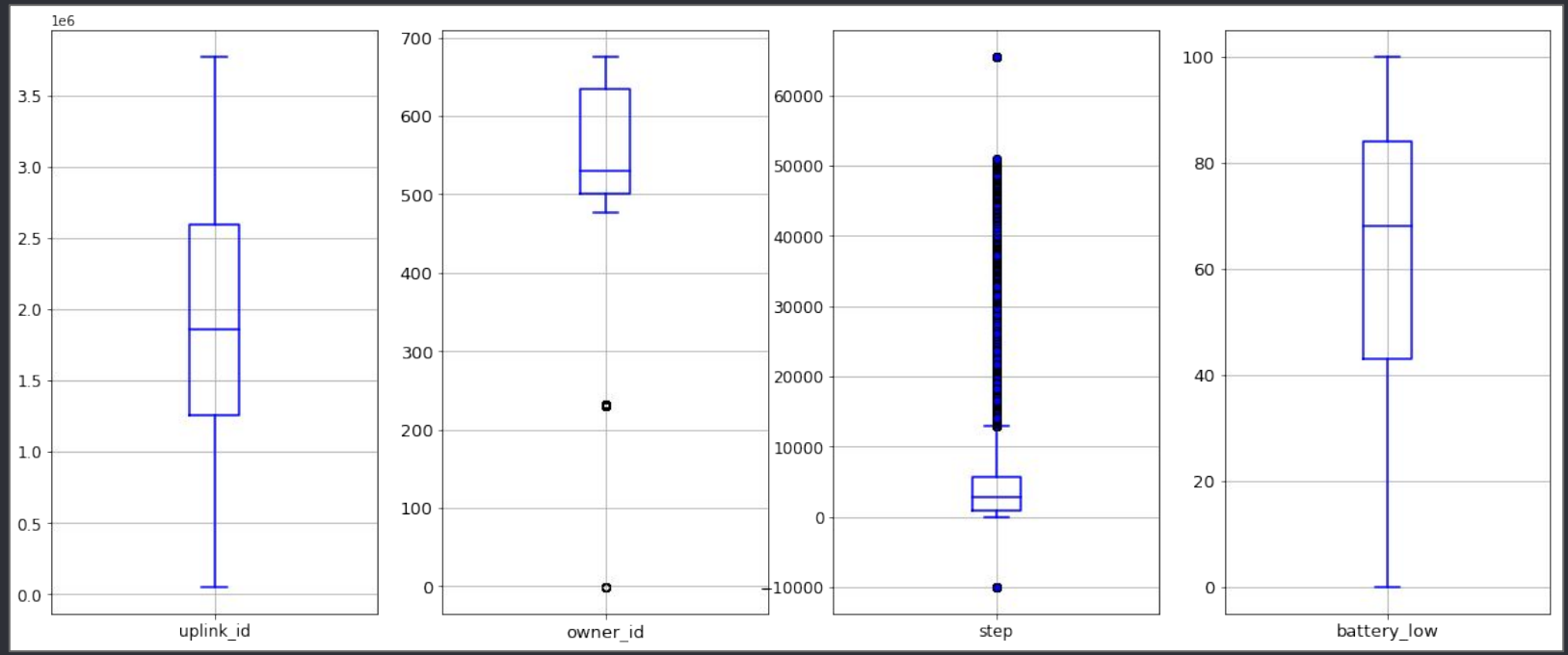
# Error Handling

# Inspect Distribution of User Attributes



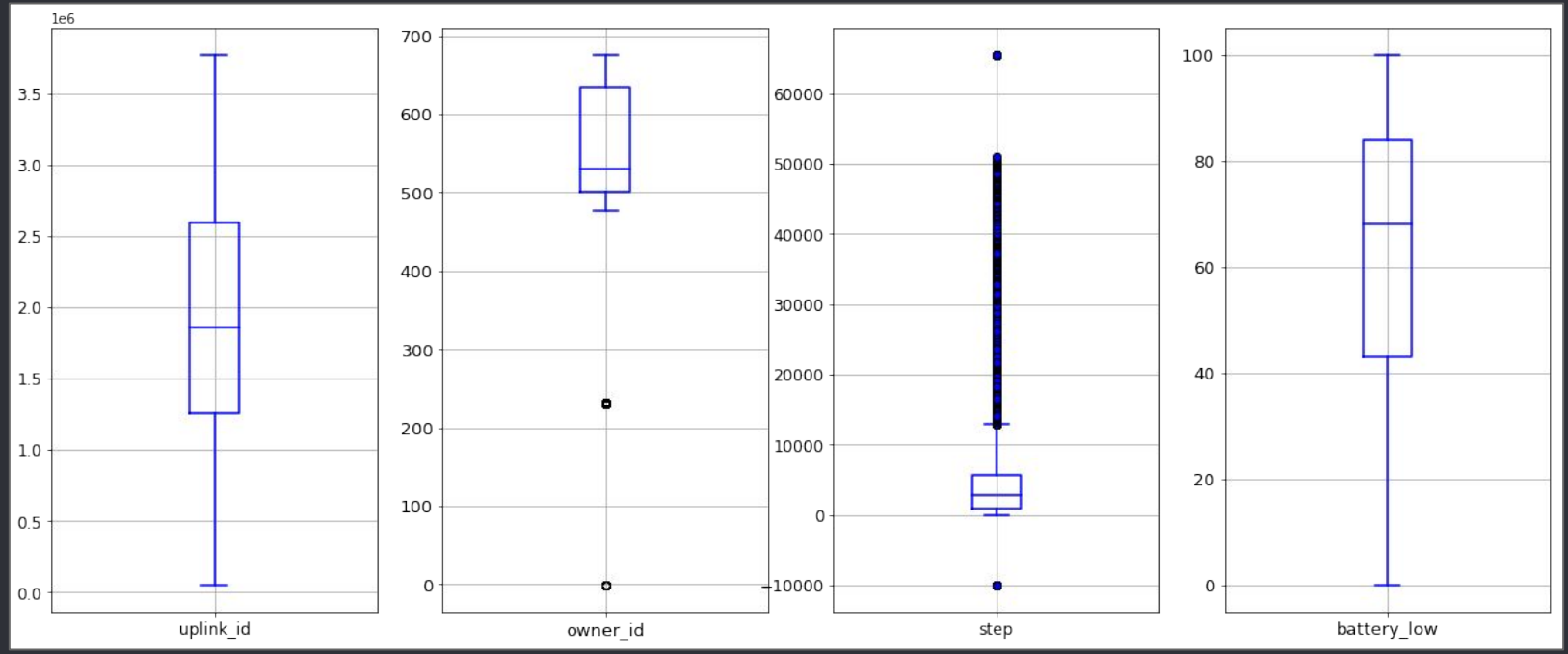

Distribution of numerical user attributes

# Inspect Distribution of Uplink Attributes




Distribution of numerical uplink attributes

# Inspect Distribution of Uplink Attributes




1. **Owner Id:** Negative Value (-1)
2. **Step:** Negative Value (-9999)

# Owner Id

Matching lifelogging entry to specific user is necessary for pattern analysis

-> Delete entries with faulty owner id

```
In [13]: #Remove all rows with non-valid user id
         uplink_df = uplink_df.loc[uplink_df['owner_id'] >=0]
```

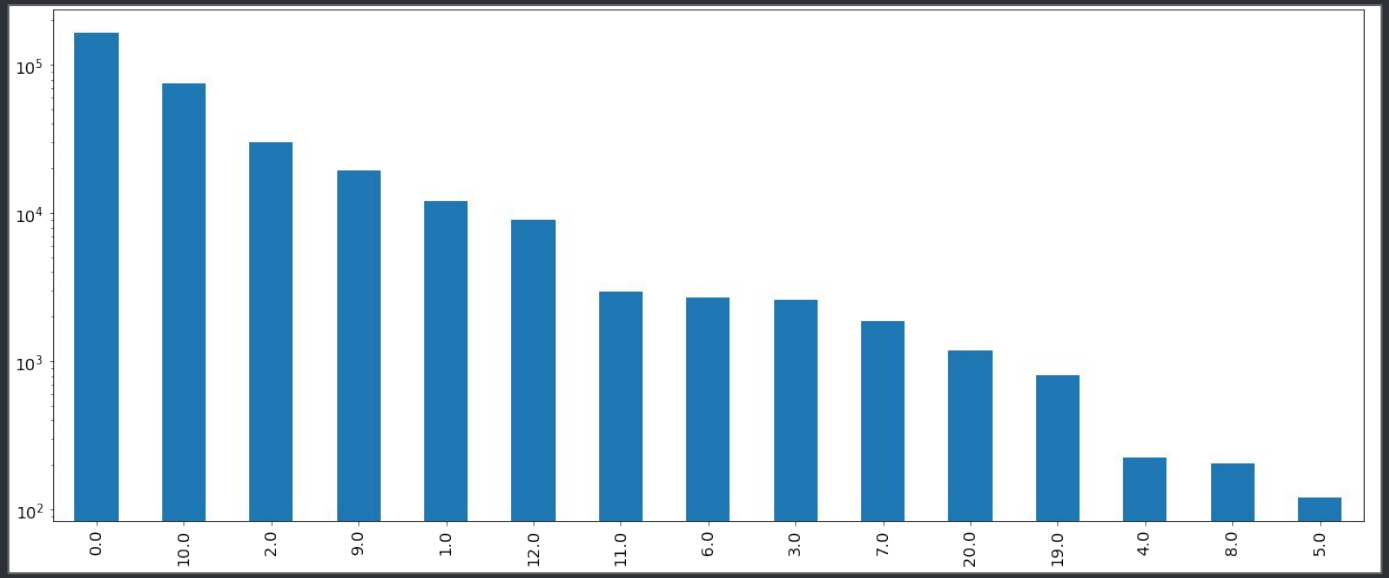
# Steps

Interpolate missing step values from surrounding values

| | uplink_id | owner_id | client_time | tag_id | step | battery_low | is_charge | tag_battery_low |
|---|---|---|---|---|---|---|---|---|
| **132328** | 1638050 | 662 | 2020-10-19 10:10:13 | 0 | 4495 | 96 | 0 | 0 |
| **132351** | 1638203 | 662 | 2020-10-19 10:20:09 | 0 | 4624 | 95 | 0 | 0 |
| **132353** | 1638203 | 662 | 2020-10-19 10:20:09 | 20 | -9999 | 95 | 0 | 0 |
| **132393** | 1638346 | 662 | 2020-10-19 10:30:09 | 0 | 4717 | 95 | 0 | 0 |
| **132432** | 1638484 | 662 | 2020-10-19 10:40:10 | 0 | 4808 | 93 | 0 | 0 |

# Tag Id

Remove log entries with invalid tag Id




Distribution of Tag Id values

```
In [17]: #Remove invalid tag ids
         uplink_df = uplink_df.loc[uplink_df['tag_id'] < 18]
```

# Client Time

Without valid timestamp no pattern analysis possible

```
In [18]: uplink_df['client_time'].min(), uplink_df['client_time'].max()
Out[18]: (Timestamp('1970-01-01 00:00:00'), Timestamp('2021-03-11 16:05:40'))
```

Remove log entries with timestamp outside of project duration

```
In [19]: #Remove all rows with timestamps outside of 2019.01.01. to 2021.03.11 timespan
start = pd.to_datetime('2019-01-01 00:00:00')
end = pd.to_datetime('2021-03-11 23:59:59')
uplink_df = uplink_df.loc[(uplink_df['client_time']>= start) & (uplink_df['client_time']<= end)]
```

# 4 Dimension Reduction

# User Data

```
In [21]: users_df.corr()
```

Out[21]:

| | id | birth year | age |
|---|---|---|---|
| **id** | 1.00000 | 0.15413 | -0.15413 |
| **birth year** | 0.15413 | 1.00000 | -1.00000 |
| **age** | -0.15413 | -1.00000 | 1.00000 |

1. Birth year column is redundant as seen in the correlation table

```
In [22]: #Drop birth year column
         users_df= users_df.drop(columns=['birth year'])
```

2. Convert Sex Attribute to dummy variable

```
In [23]: #Convert sex attribute by dummy attribute
         users_df['sex'] = users_df['sex'].astype('category')
         users_df['sex']= pd.get_dummies(users_df['sex'], prefix_sep='_', drop_first=True)
```

# Medical Information

```
In [15]: users_df['etc'].value_counts()

Out[15]:                                                                                                  19
         고지혈증                                                                                          4
         복약통에 설치 희망하심                                                                            2
         혈압,당뇨,신장약 복용                                                                             1
         차상위, 혈압, 당뇨, 관절                                                                          1
         혈압약, 전립선                                                                                    1
         협심증으로 약 복용 중                                                                             1
         항암치료중                                                                                        1
         협심증약, 고지혈약, 혈압약, 아스피린복용. 15년넘게 심장약 복용중이고 매일 허리와 고관절통증으로 물리치료다니고 계심    1
         신장약, 혈압                                                                                      1
         심장약, 당뇨, 혈압                                                                                1
         기저질환으로 고지혈증, 고혈압으로 약 복용 중 / 보청기 착용 / 심장질환                            1
```

Replace information string by two dummy variables

```
In [26]: #Use additional user information in order to create new simpler variables
         medicine_use = [1 if '약' in s else 0 for s in users_df['etc']]
         users_df['medicine_use'] = medicine_use
         has_diabetes = [1 if '당뇨' in s else 0 for s in users_df['etc']]
         users_df['has_diabetes'] = has_diabetes
         users_df = users_df.drop(columns=['etc'])
```

# Uplink Data

```
In [25]: uplink_df.corr()
```

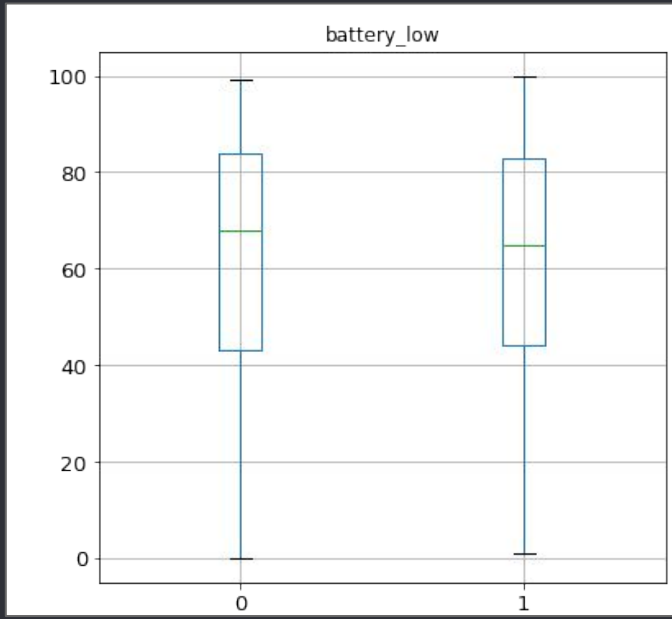
Out[25]:

| | uplink_id | owner_id | tag_id | step | battery_low | is_charge | tag_battery_low |
|---|---|---|---|---|---|---|---|
| **uplink_id** | 1.000000 | 0.228201 | 0.042081 | -0.012391 | 0.118560 | 0.002724 | 0.067532 |
| **owner_id** | 0.228201 | 1.000000 | 0.021234 | -0.103737 | 0.096774 | 0.000159 | -0.039102 |
| **tag_id** | 0.042081 | 0.021234 | 1.000000 | -0.011944 | 0.003575 | -0.042524 | 0.147781 |
| **step** | -0.012391 | -0.103737 | -0.011944 | 1.000000 | -0.097741 | 0.003649 | -0.016925 |
| **battery_low** | 0.118560 | 0.096774 | 0.003575 | -0.097741 | 1.000000 | 0.003073 | -0.023290 |
| **is_charge** | 0.002724 | 0.000159 | -0.042524 | 0.003649 | 0.003073 | 1.000000 | -0.008826 |
| **tag_battery_low** | 0.067532 | -0.039102 | 0.147781 | -0.016925 | -0.023290 | -0.008826 | 1.000000 |

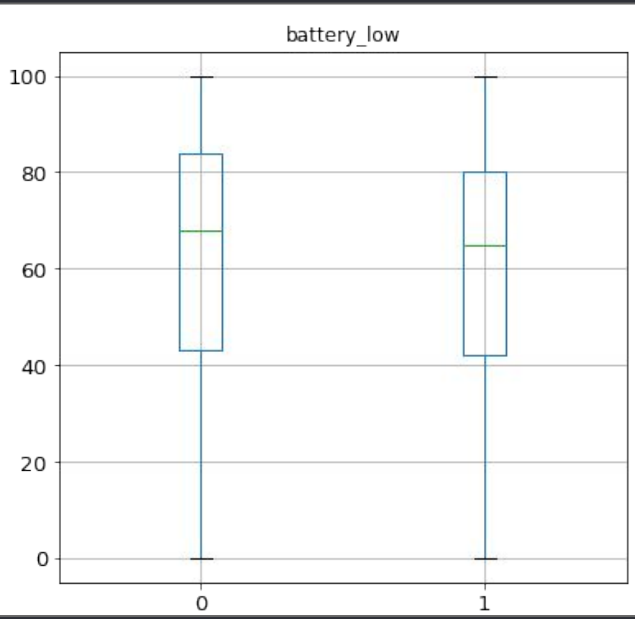
1. Uplink Id only relevant in combination with other dataset

```
In [26]: #Drop uplink_id column
uplink_df = uplink_df.drop(columns=['uplink_id'])
```

# Battery Attributes




Battery Level by Charge Category




Battery Level by Tag Battery Category

# Is Charge

In [28]:
```
z = uplink_df.loc[(uplink_df['owner_id']==501) & (uplink_df['tag_battery_low']==1)]
z2 = z.reset_index()
ind =z2.index[z2['is_charge']==1].tolist()
emp = pd.DataFrame()

for i in ind[1:]:
    emp = pd.concat([emp,z2.iloc[[i-1,i,i+1]]], ignore_index=True)
emp.head()
```

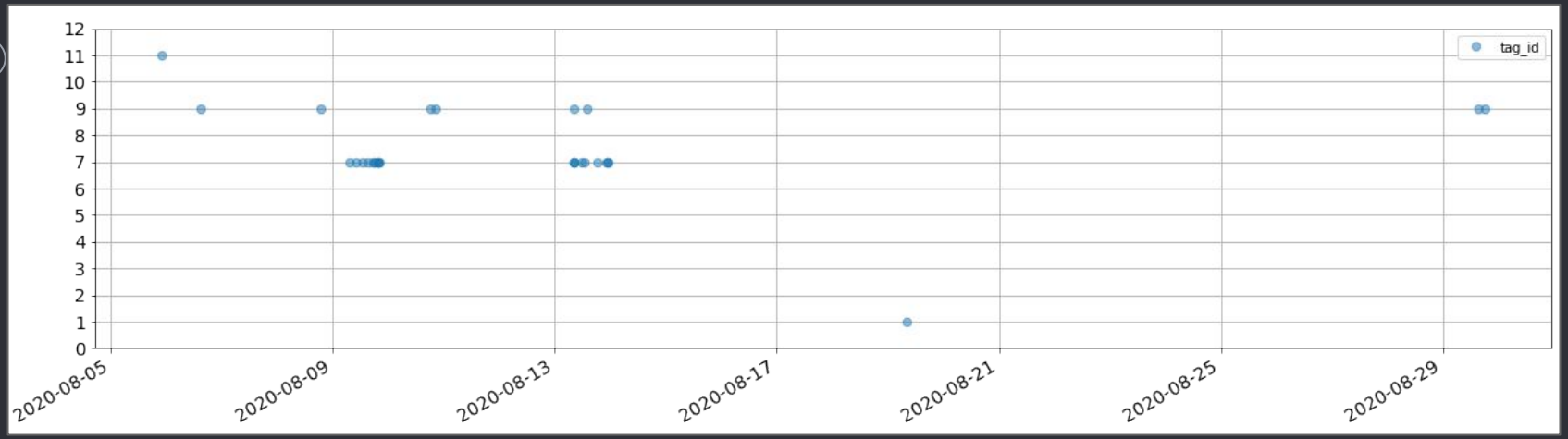
Out[28]:

| | index | owner_id | client_time | tag_id | step | battery_low | is_charge | tag_battery_low |
|---|---|---|---|---|---|---|---|---|
| 0 | 153141 | 501 | 2020-10-28 18:36:52 | 6 | 8586 | 67 | 0 | 1 |
| 1 | 153428 | 501 | 2020-10-29 03:49:10 | 6 | 38 | 70 | 1 | 1 |
| 2 | 154055 | 501 | 2020-10-29 11:04:34 | 1 | 3772 | 82 | 0 | 1 |

Behaviour of charge attribute can be completely inferred from battery low attribute

In [28]:
```
#Drop is_charge column
uplink_df = uplink_df.drop(columns=['is_charge'])
```

# Tag Battery




Tag Battery Low Occurrences for User 230 plotted over different Tag Ids

Tag battery attribute seems to showcase localized occurrences -> Indicator for relation to sensor tag

# Result

In [31]:
```
users_df.sample(5)
```

Out[31]:

| | id | age | sex | medicine_use | has_diabetes |
|---|---|---|---|---|---|
| **24** | 508 | 74 | 0 | 0 | 0 |
| **34** | 653 | 68 | 0 | 0 | 0 |
| **13** | 499 | 79 | 0 | 0 | 0 |
| **30** | 658 | 73 | 0 | 1 | 0 |
| **40** | 650 | 83 | 1 | 1 | 0 |

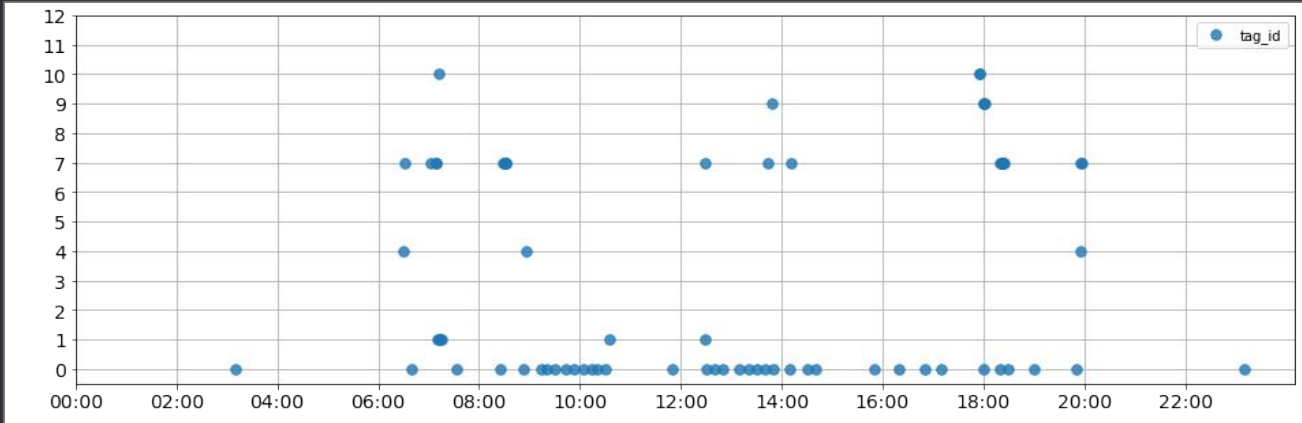
In [32]:
```
uplink_df.sample(5)
```

Out[32]:

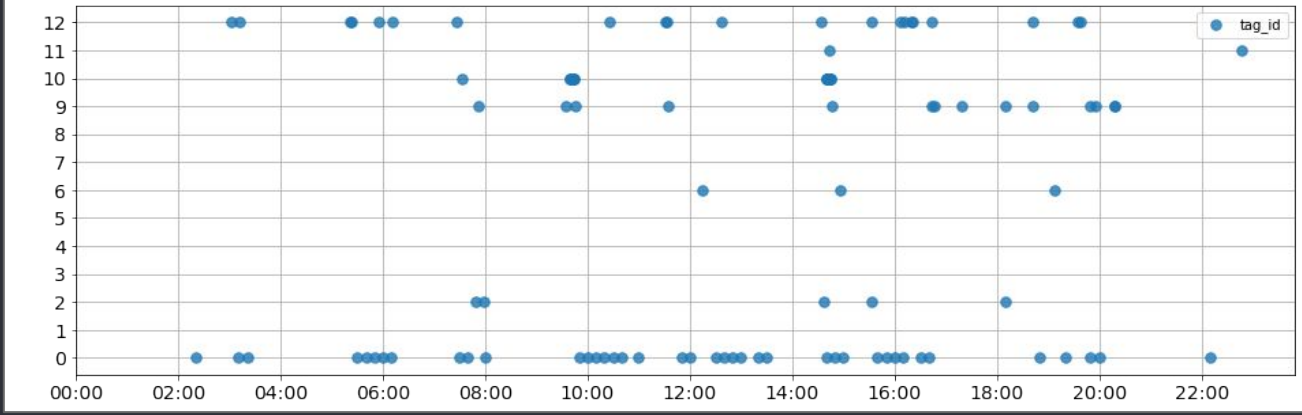
| | owner_id | client_time | tag_id | step | battery_low | tag_battery_low |
|---|---|---|---|---|---|---|
| **66860** | 530 | 2020-09-19 08:20:12 | 0 | 11 | 30 | 0 |
| **83749** | 670 | 2020-09-28 16:30:54 | 10 | 5223 | 30 | 0 |
| **71202** | 507 | 2020-09-22 11:28:21 | 12 | 9164 | 40 | 0 |
| **91745** | 505 | 2020-10-01 15:26:33 | 2 | 6473 | 84 | 0 |
| **142055** | 492 | 2020-10-23 11:50:12 | 0 | 7387 | 75 | 0 |

# 5 Visualization

# Tagging Logs

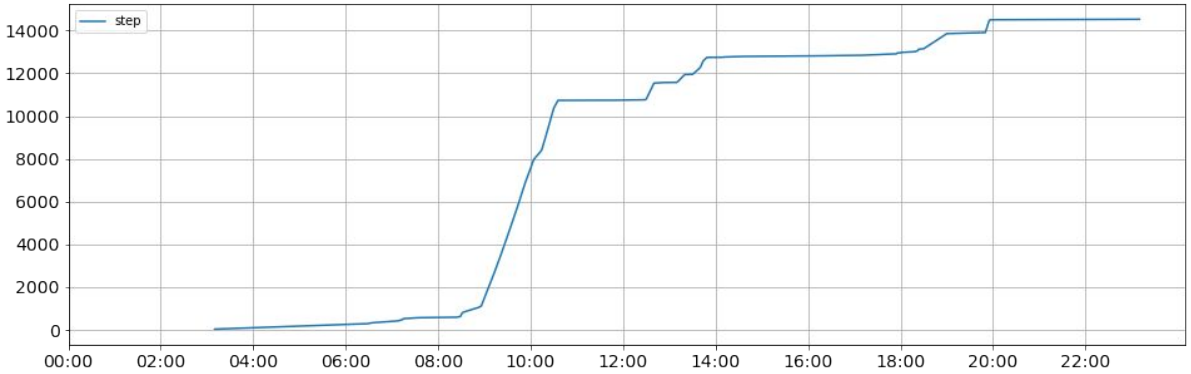



Example Tagging Log - User 230

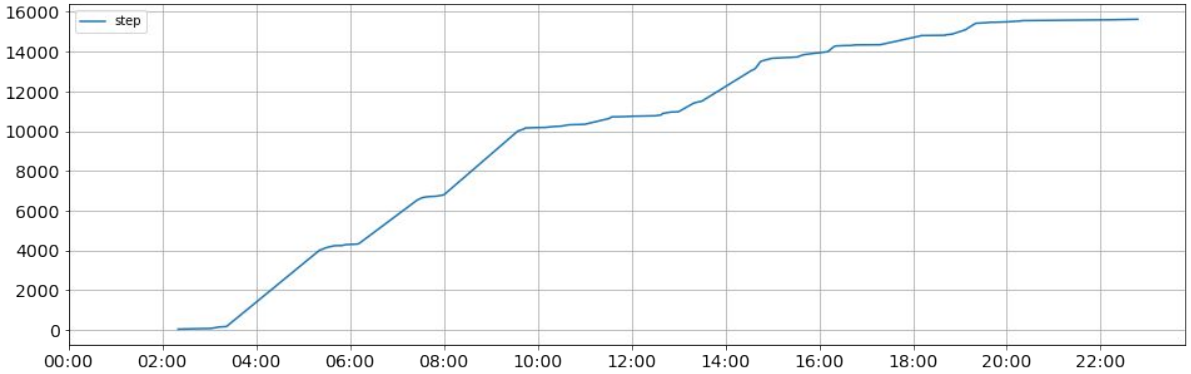



Example Tagging Log - User 504

# Activity Logs

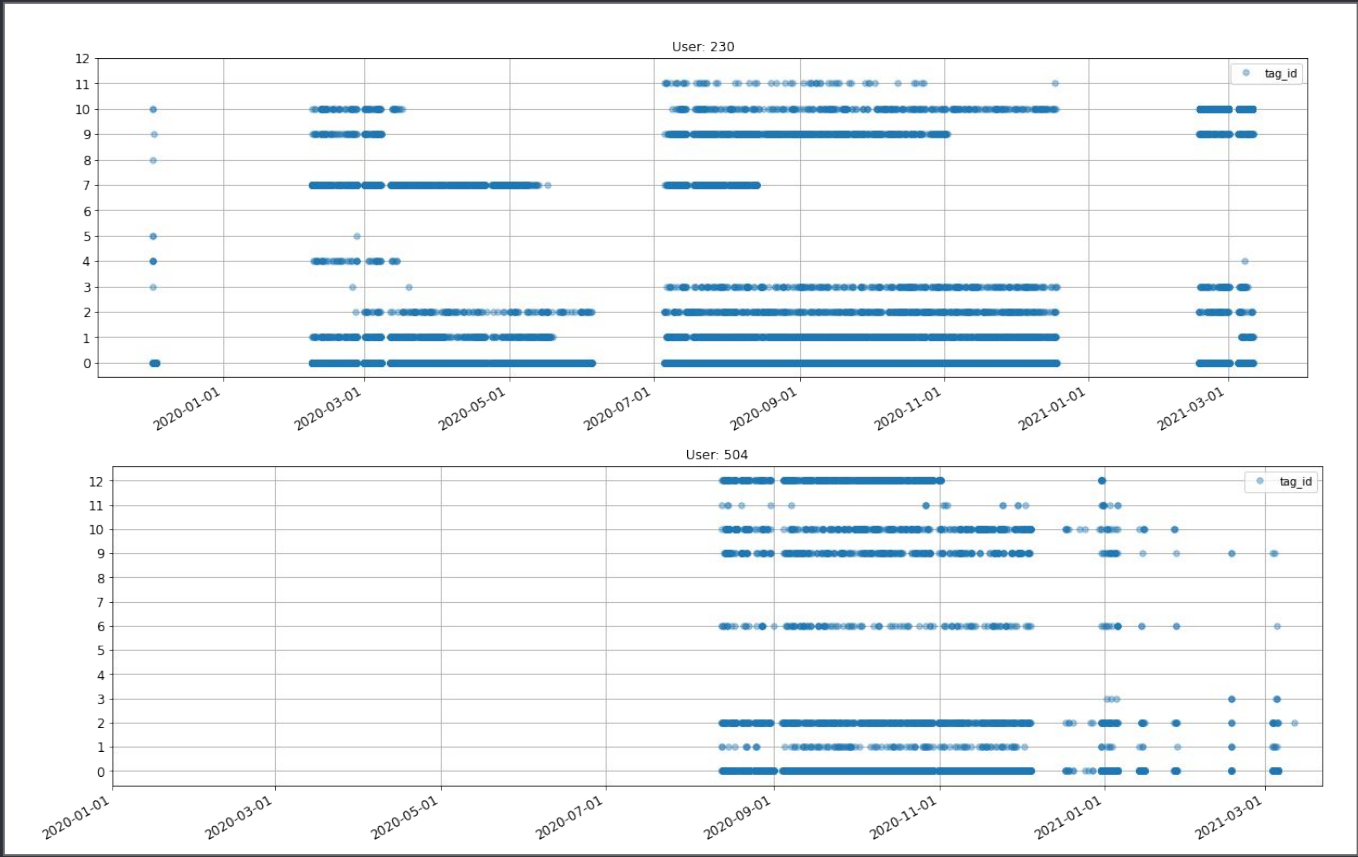



Example Activity Log - User 230




Example Activity Log - User 504

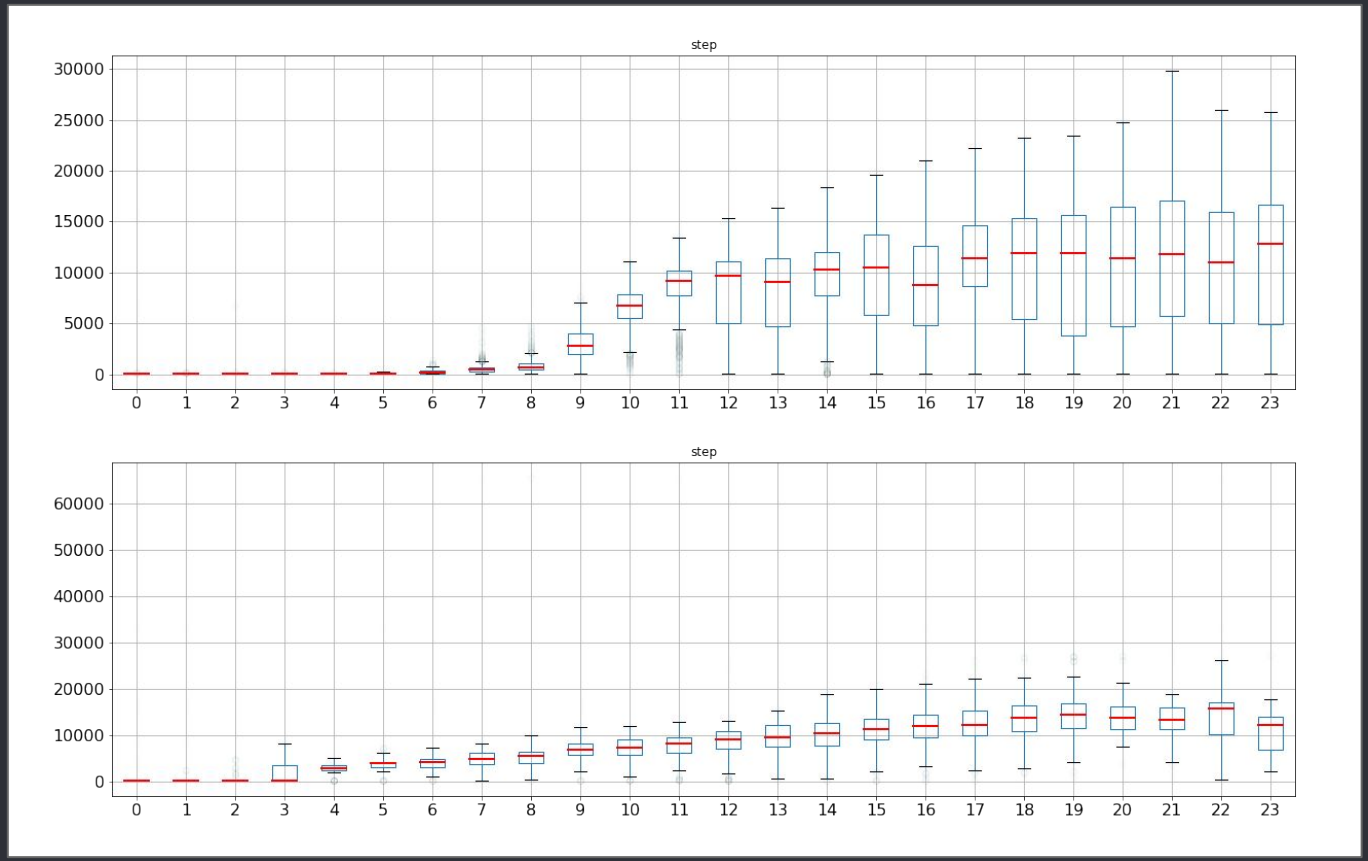
# Tagging History




Tagging History - User 230

Tagging History - User 504

# Daily Activity Pattern




Activity Distribution - User 230

Activity Distribution - User 504

# 6 Proposed Services

# PROPOSED SERVICES

**Activity Monitoring:**

Monitor users activity pattern and compare it to predicted pattern.

Alarm officials or family if strong deviations (medical complications) occur.

**Task Reminder:**

Remind elderly citizens of important tasks in their daily lives.

Example: Monitor medicine intake and remind users when to take medicine

**Battery Service:**

Implement Service that ensures working tagging devices at all times.

- Remind users of charging device when battery runs out
- Develop improved version of tagging device (senior smartphone/smartwatch)

# Thank you for your Attention

Questions at: annaribic01@gmail.com