data_ready_1

August 2, 2018

```
In [2]: import pandas as pd
        import logging
        import numpy as np
        import sys
        import matplotlib.pyplot as plt
        from sklearn.cross_validation import train_test_split
        from random import shuffle
        from sklearn.base import BaseEstimator, RegressorMixin
        from scipy.optimize import minimize
        from sklearn.model_selection import GridSearchCV, PredefinedSplit
        from sklearn.model_selection import ParameterGrid
        from sklearn.metrics import mean_squared_error, make_scorer

        from datetime import date
        from datetime import time
        from datetime import datetime
        from datetime import timedelta
        %matplotlib inline


        from IPython.core.interactiveshell import InteractiveShell
        InteractiveShell.ast_node_interactivity = "all"
        #PLOT CONFUSION MATRIX
        from sklearn.metrics import confusion_matrix
        import itertools

        #matrix inverse
        from numpy.linalg import inv

        #see the value of multiple statements at once
        from IPython.core.interactiveshell import InteractiveShell
        InteractiveShell.ast_node_interactivity = "all"

        #retina resolution figure
        %config InlineBackend.figure_format = 'retina'

        #default size of the graph
        plt.rcParams['figure.figsize'] = (10.0, 8.0)

        %load_ext autoreload
        %autoreload 2

/Users/hp/anaconda/lib/python3.5/site-packages/sklearn/cross_validation.py:41: DeprecationWarni
  "This module will be removed in 0.20.", DeprecationWarning)


In [3]: col_rs = 'region_start_time'
        col_es = 'episode_starts'
        col_le = 'long_episodes'

In [4]: np.mean
        ### shift + tab

Out[4]: <function numpy.core.fromnumeric.mean>

In [5]: def filtertime(dat, start_date, end_date):
            return np.array(start_date <= dat[col_rs]) & \
                 np.array(dat[col_rs] <= end_date)

In [6]: def addepoch(dat, start, end, num_per_epoch):

            c = end - start
            num_epoch  = int(np.floor(c.days/num_per_epoch))
            start_dates = [start + timedelta(days = i * num_per_epoch) for i in range(num_epoc
            end_dates = [start + timedelta(days = (i + 1) * num_per_epoch) for i in range(num_
            dates = list(zip(start_dates, end_dates))
            end = dates[-1][-1]
            for i in range(num_epoch):
                date_tup = dates[i]
                dstart, dend = date_tup[0],date_tup[1]
                dat.loc[filtertime(dat, dstart, dend),'epoch'] = int(i)
            data = dat[np.array(start <= dat.loc[:,col_rs]) & \
                 np.array(dat.loc[:,col_rs] <= end)]
            data.loc[:,'epoch'] = data.loc[:,'epoch'].astype(int)
            return data

In [7]: def firstnorm(dat, col):
            base = np.array(dat[col])[0]
            dat.loc[:,col] = dat[col] / base
            return dat, base

In [8]: def dat_agg(dat):
            N = dat.shape[0]
            dat_epi_agg = dat.loc[:, [col_es, 'epoch']].groupby('epoch').agg('mean')
            dat_epi_agg, base_epi = firstnorm(dat_epi_agg, col_es)
            dat_le_agg = dat.loc[:, [col_le, 'epoch']].groupby('epoch').agg('mean')
            dat_le_agg, base_le = firstnorm(dat_le_agg, col_le)
            dat_epi_agg_ste = dat.loc[:, [col_es, 'epoch']].groupby('epoch').std()/base_epi/np
            dat_le_agg_ste = dat.loc[:, [col_le, 'epoch']].groupby('epoch').std()/base_le/np.s

            return dat_epi_agg, dat_le_agg, dat_epi_agg_ste, dat_le_agg_ste
```

```
In [9]: def epoch_label(dat_tup, col_metric):
            if col_metric != col_es and col_metric != col_le:
                sys.exit('col_metric wrong')
            dat = dat_tup[0]
            dat_epi_agg, dat_le_agg, dat_epi_agg_ste, dat_le_agg_ste = dat_agg(dat)
            if col_metric == col_es:
                n = dat_epi_agg.shape[0]
                thres = np.median(dat_epi_agg)
                keys = list(np.array(dat_epi_agg.index, dtype = int))
                vals = list(np.array(dat_epi_agg.loc[:,col_metric] < thres))
            elif col_metric == col_le:
                n = dat_le_agg.shape[0]
                thres = np.median(dat_le_agg)
                keys = list(np.array(dat_le_agg.index, dtype = int))
                vals = list(np.array(dat_le_agg.loc[:,col_metric] < thres))
            epoch_label = dict(zip(keys, vals))
            for key in epoch_label:
                val = epoch_label[key]
                dat.loc[dat['epoch'] == key,'label'] = val
            data = (dat, dat_tup[1])
            return data, epoch_label
```

```
In [10]: def prep(dat, start, end, num_per_epoch, patid):
             dat.loc[:,col_rs] = pd.to_datetime(dat.loc[:,col_rs])
             data_1 = addepoch(dat, start, end, num_per_epoch)
             cache = [start, end, num_per_epoch, patid]
             data_2 = (data_1, cache)
             data_3_es, epoch_label_dict_es = epoch_label(data_2, col_es)
             data_3_le, epoch_label_dict_le = epoch_label(data_2, col_le)
             return data_3_es, epoch_label_dict_es, data_3_le, epoch_label_dict_le
```

```
In [11]: def plot_epoch_mean(dat_tup):
             dat = dat_tup[0]
             ptid = dat_tup[1][3]
             period_start = str(dat_tup[1][0])[:10]
             period_end = str(dat_tup[1][1])[:10]

             dat_epi_agg, dat_le_agg, dat_epi_agg_ste, dat_le_agg_ste = dat_agg(dat)
             plt.figure()
             fig, ax = plt.subplots(1,1)
             ax.set_xticks(range(dat_le_agg.shape[0]))
             ax.set_xticklabels(range(1,dat_le_agg.shape[0] + 1))
```

```
            plt.plot(dat_epi_agg, label = 'episodes start mean')
            plt.plot(dat_epi_agg + dat_epi_agg_ste,linestyle='dashed', label = 'episodes star
            plt.plot(dat_epi_agg - dat_epi_agg_ste,linestyle='dashed', label = 'episodes star
            plt.plot(dat_le_agg, label = 'long episode mean')
            plt.plot(dat_le_agg + dat_le_agg_ste,linestyle='dashed', label = 'long episode mea
            plt.plot(dat_le_agg - dat_le_agg_ste,linestyle='dashed', label = 'long episode mea
            plt.title('Patient {0}: period {1} - {2}'.format(ptid, period_start, period_end))
            plt.xlabel('epoch')
            plt.ylabel('count(nomalized)')
            plt.tight_layout()
            plt.legend()
            plt.show()
```

```
In [12]: start_222_1 = datetime.strptime('Feb 12 2016', '%b %d %Y')
         end_222_1 = datetime.strptime('Oct 24 2016', '%b %d %Y')
         num_per_epoch_222_1 = 31

         start_222_2 = datetime.strptime('Oct 26 2016', '%b %d %Y')
         end_222_2 = datetime.strptime('May 29 2017', '%b %d %Y')
         num_per_epoch_222_2 = 30

         start_222_3 = datetime.strptime('Sep 19 2017', '%b %d %Y')
         end_222_3 = datetime.strptime('Jan 30 2018', '%b %d %Y')
         num_per_epoch_222_3 = 31


         start_231 = datetime.strptime('Feb 7 2017', '%b %d %Y')
         end_231 = datetime.strptime('Feb 21 2018', '%b %d %Y')
         num_per_epoch_231 = 31
```

```
In [13]: raw_data_222 = pd.read_csv('../data/NY222_2015-08-11_to_2018-06-12_daily_201806131531
         raw_data_231 = pd.read_csv('../data/NY231_2016-07-05_to_2018-06-12_daily_201806131538
```

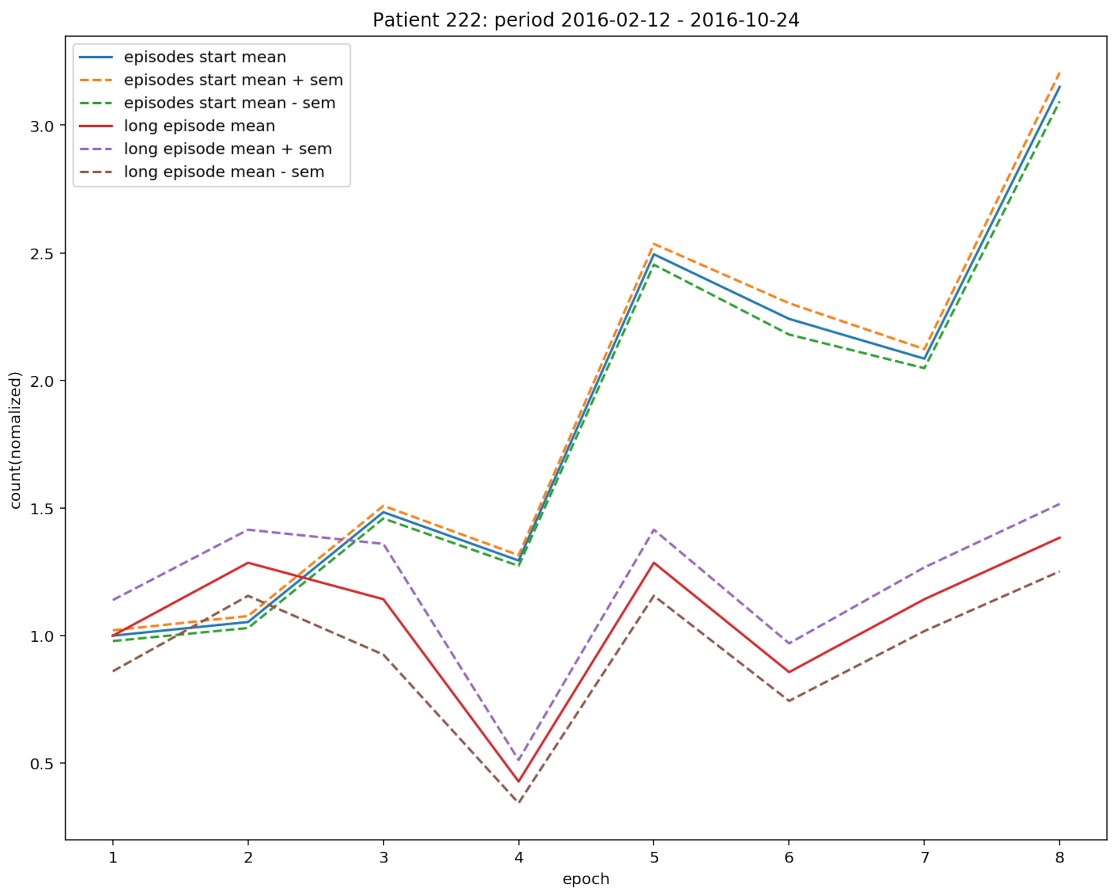
```
In [14]: data_222_1_es, epoch_label_222_1_es, data_222_1_le, epoch_label_222_1_le = prep(raw_da
         data_222_2_es, epoch_label_222_2_es, data_222_2_le, epoch_label_222_2_le = prep(raw_da
         data_222_3_es, epoch_label_222_3_es, data_222_3_le, epoch_label_222_3_le = prep(raw_da
         data_231_es, epoch_label_231_es, data_231_le, epoch_label_231_le = prep(raw_data_231,
```

```
/Users/hp/anaconda/lib/python3.5/site-packages/pandas/core/indexing.py:517: SettingWithCopyWar
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.htm
  self.obj[item] = s
/Users/hp/anaconda/lib/python3.5/site-packages/pandas/core/indexing.py:337: SettingWithCopyWar
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.htm
```

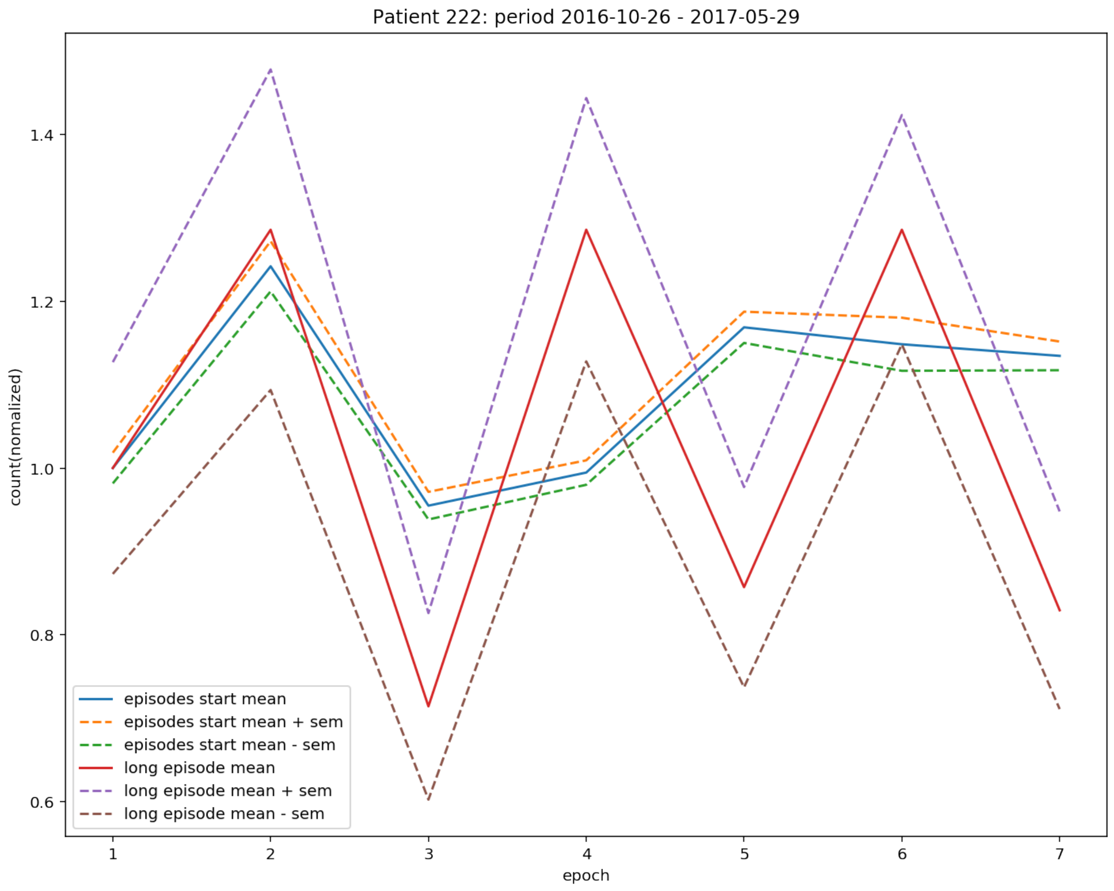
```
  self.obj[key] = _infer_fill_value(value)
```

```
In [15]: plot_epoch_mean(data_222_1_es)
         plot_epoch_mean(data_222_2_es)
         plot_epoch_mean(data_222_3_es)
         plot_epoch_mean(data_231_es)
```

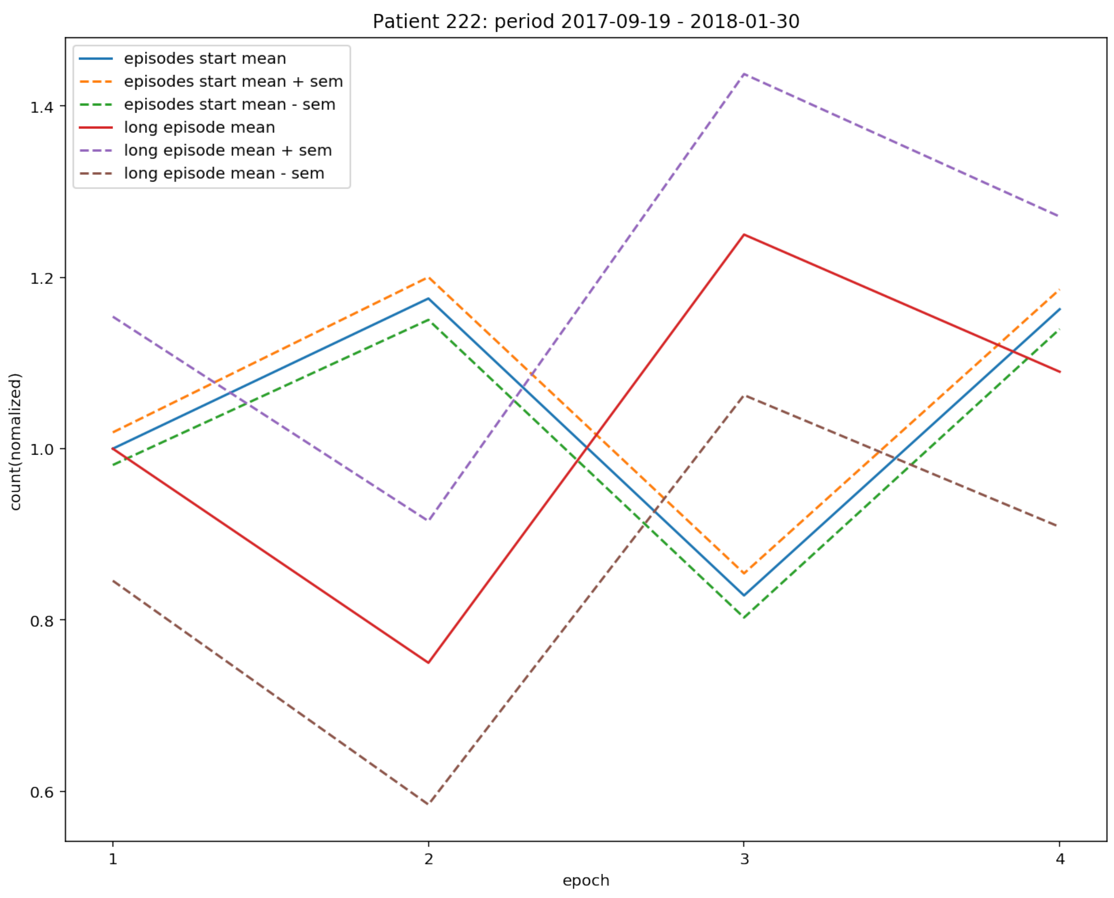
```
<matplotlib.figure.Figure at 0x1166d00f0>
```



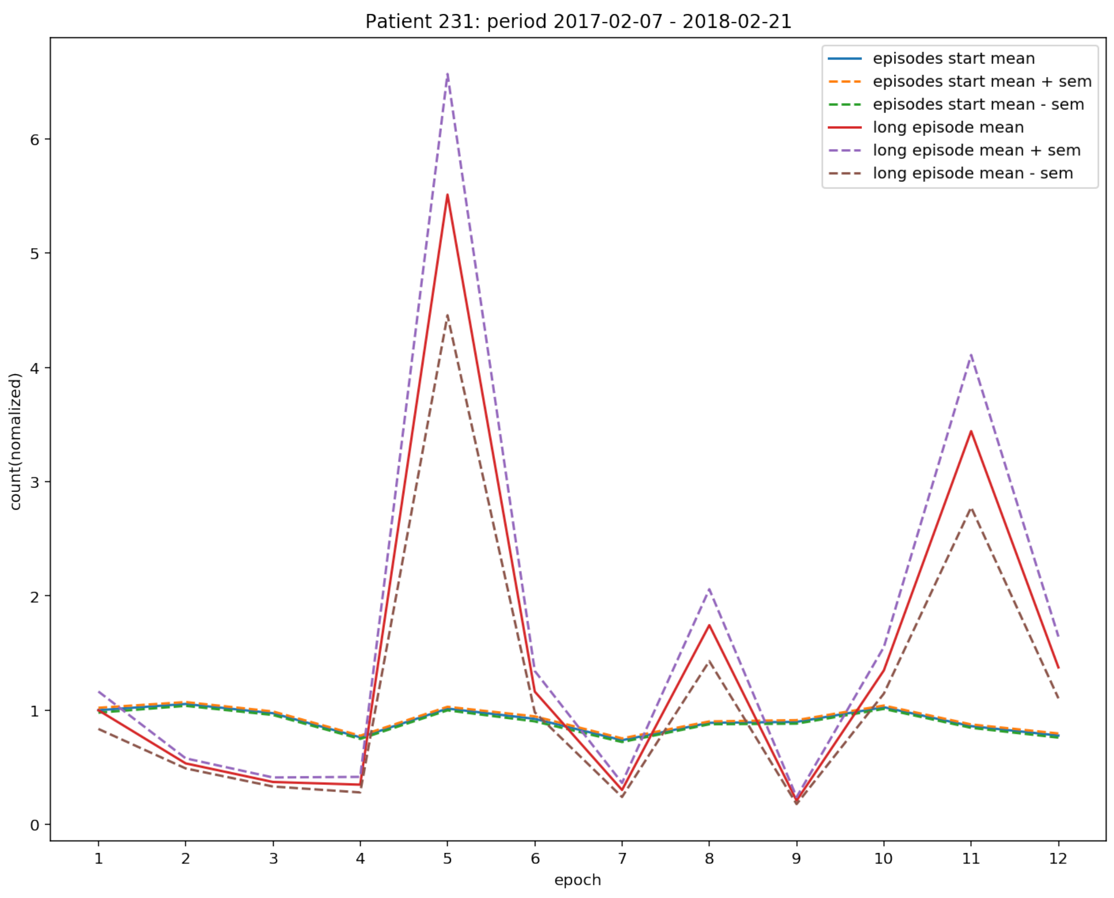
```
<matplotlib.figure.Figure at 0x116736198>
```


Patient 222: period 2016-10-26 - 2017-05-29
count(nomalized)
epoch
episodes start mean
episodes start mean + sem
episodes start mean - sem
long episode mean
long episode mean + sem
long episode mean - sem
1.4
1.2
1.0
0.8
0.6
1
2
3
4
5
6
7

```
<matplotlib.figure.Figure at 0x11e523240>
```


Patient 222: period 2017-09-19 - 2018-01-30
episodes start mean
episodes start mean + sem
episodes start mean - sem
long episode mean
long episode mean + sem
long episode mean - sem
count(nomalized)
1.4
1.2
1.0
0.8
0.6
1
2
3
4
epoch

```
<matplotlib.figure.Figure at 0x11e635940>
```



```
In [16]: epoch_label_231_le

Out[16]: {0: True,
          1: True,
          2: True,
          3: True,
          4: False,
          5: False,
          6: True,
          7: False,
          8: True,
          9: False,
          10: False,
          11: False}

In [17]: import h5py

         def read_features(f):
             powbands = ['delta', 'theta', 'alpha', 'beta', 'low_gamma', 'high_gamma', 'all']
             col_names = [col_rs]
```

```
            for powband in powbands:
                for i in range(1,5):
                    col_names.append(powband+str(i))
            col_names.append('i12')
            col_names.append('i34')
            features_222 = np.array(f['T_222_arr_scheduled']).T
            features_231 = np.array(f['T_231_arr_scheduled']).T
            features_df_222 = pd.DataFrame(features_222, columns = col_names)
            features_df_231 = pd.DataFrame(features_231, columns = col_names)
            features_df_222.loc[:,col_rs] = pd.to_datetime(features_df_222.loc[:,col_rs], uni
            features_df_231.loc[:,col_rs] = pd.to_datetime(features_df_231.loc[:,col_rs], uni
            return features_df_222, features_df_231

In [18]: def feature_label(features, start, end, num_per_epoch, epoch_label_dict):
            features_epoch = addepoch(features, start, end, num_per_epoch)
            for key in epoch_label_dict:
                val = epoch_label_dict[key]
                features_epoch.loc[features_epoch.loc[:,'epoch'] == key,'label'] = val
            return features_epoch

In [19]: def add_id(df, pat_id, if_stimulated):
            df.loc[:,'patid'] = pat_id
            df.loc[:,'if_stimulated'] = if_stimulated
            return df

In [20]: f = h5py.File('../data/features.mat', 'r')
         f_s = h5py.File('../data/features_sti.mat', 'r')

In [21]: fea_222, fea_231 = read_features(f)
         fea_222_s, fea_231_s = read_features(f_s)
         feature_label_222_1 = add_id(feature_label(fea_222, start_222_1, end_222_1, num_per_ep
         feature_label_222_2 = add_id(feature_label(fea_222, start_222_2, end_222_2, num_per_ep
         feature_label_222_3 = add_id(feature_label(fea_222, start_222_3, end_222_3, num_per_ep
         feature_label_231= add_id(feature_label(fea_231, start_231, end_231, num_per_epoch_23
         feature_label_222_1_s = add_id(feature_label(fea_222_s, start_222_1, end_222_1, num_pe
         feature_label_222_2_s = add_id(feature_label(fea_222_s, start_222_2, end_222_2, num_pe
         feature_label_222_3_s = add_id(feature_label(fea_222_s, start_222_3, end_222_3, num_pe
         feature_label_231_s = add_id(feature_label(fea_231_s, start_231, end_231, num_per_epo

/Users/hp/anaconda/lib/python3.5/site-packages/pandas/core/indexing.py:517: SettingWithCopyWar
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.htm
  self.obj[item] = s
/Users/hp/anaconda/lib/python3.5/site-packages/pandas/core/indexing.py:337: SettingWithCopyWar
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[key] = _infer_fill_value(value)
```

```
In [22]: pd.set_option('display.max_rows', 10)
         pd.set_option('display.max_columns', 5)
         data = pd.concat([feature_label_222_1, feature_label_222_2, feature_label_222_3, featu
         data
         data.to_pickle('../data/ml_ready_data.p')
```

```
Out[22]:              region_start_time      delta1      ...        patid  \
         86   2016-02-14 03:59:36.960000   61.166778      ...        222_1
         87   2016-02-15 20:59:18.960000   40.548973      ...        222_1
         88   2016-02-16 20:59:12.998400   41.771439      ...        222_1
         89   2016-02-18 03:58:56.006400   42.171886      ...        222_1
         90   2016-02-19 03:58:42.960000   45.669293      ...        222_1
         ..                          ...         ...      ...          ...
         884  2018-02-11 15:51:35.971200  104.142656      ...          231
         885  2018-02-11 21:51:24.998400  113.162000      ...          231
         886  2018-02-12 03:51:23.011200  225.536331      ...          231
         887  2018-02-12 09:51:21.974400   85.753303      ...          231
         888  2018-02-12 15:51:21.024000   78.690558      ...          231

              if_stimulated
         86           False
         87           False
         88           False
         89           False
         90           False
         ..             ...
         884           True
         885           True
         886           True
         887           True
         888           True

         [2221 rows x 35 columns]
```

```
In [23]: def report_count():
             df_num = pd.DataFrame([], columns = ['patient_id', 'if_stimiuated', 'label', 'numb
             pat_ids = ['222_1', '222_2', '222_3', '231']
             tf = [True, False]
             i = 0
             for pat_id in pat_ids:
                 for sti in tf:
                     for lab in tf:
                         df_num.loc[i,'patient_id'] = pat_id
                         df_num.loc[i,'if_stimiuated'] = sti
```

```
                        df_num.loc[i,'label'] = lab
                        df_num.loc[i,'number'] = data.loc[np.array(data.loc[:,'patid']
                                == pat_id) & np.array(data.loc[:,'if_stimulated'] == sti)
                                & np.array(data.loc[:,'label'] == lab)].shape[0]
                        i += 1
            print(df_num)
            print(pd.DataFrame(df_num.groupby('patient_id').agg('sum').loc[:,'number']))
            print(pd.DataFrame(df_num.groupby(['patient_id', 'label']).agg('sum').loc[:,'numbe
```

```
In [24]: report_count()
```

```
   patient_id if_stimiuated  label number
0       222_1          True   True    216
1       222_1          True  False    273
2       222_1         False   True     55
3       222_1         False  False     85
4       222_2          True   True    179
..        ...           ...    ...    ...
11      222_3         False  False     32
12        231          True   True    272
13        231          True  False    234
14        231         False   True    216
15        231         False  False    116

[16 rows x 4 columns]
            number
patient_id
222_1          629
222_2          429
222_3          325
231            838
                  number
patient_id label
222_1      False     358
           True      271
222_2      False     218
           True      211
222_3      False     178
           True      147
231        False     350
           True      488
```