ML results_222_2

August 6, 2018

```
In [1]: patid = '222_2'
```

```
In [2]: import pandas as pd
        import logging
        import numpy as np
        import sys
        import matplotlib.pyplot as plt
        import time
        import operator

        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 sklearn.model_selection import train_test_split
        from sklearn.model_selection import StratifiedKFold
        from sklearn.model_selection import GridSearchCV
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import confusion_matrix
        from sklearn.externals import joblib
        import jj_basic_fn as JJ
        from sklearn import ensemble
        import seaborn as sns
        %matplotlib inline

        #PLOT CONFUSION MATRIX
        from sklearn.metrics import confusion_matrix
        import itertools

        #matrix inverse
        from numpy.linalg import inv

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

```
%load_ext autoreload
%autoreload 2

pd.set_option('display.max_rows', 10)
pd.set_option('display.max_columns', 10)
pd.set_option('display.max_colwidth', -1)

n_classifier = 7
```

```
/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]:
```
features_list = ['delta', 'beta', 'low_gamma']
plot_3d_var_list = ['low_gamma2', 'beta2','all1']
```

0.1 1. Data loading

0.1.1 What the data looks like

In [4]:
```
import pickle
data = pickle.load( open( "../data/ml_ready_data.p", "rb" ) )
# remove outliers
data = JJ.remove_outliers(data)
pd.set_option('display.max_rows', 10)
pd.set_option('display.max_columns', 10)

data
```

Out[4]:
```
         filename           region_start_time       delta1      delta2  \
86   1.309997e+17 2016-02-14 03:59:36.960000  61.166778   273.677298
87   1.310015e+17 2016-02-15 20:59:18.960000  40.548973   773.155101
88   1.310019e+17 2016-02-16 20:59:12.998400  41.771439   172.179808
89   1.310032e+17 2016-02-18 03:58:56.006400  42.171886   290.146546
90   1.310041e+17 2016-02-19 03:58:42.960000  45.669293   290.906731
..            ...                        ...         ...         ...
884  1.316288e+17 2018-02-11 15:51:35.971200  104.142656  43.925946
885  1.316296e+17 2018-02-11 21:51:24.998400  113.162000  50.395396
886  1.316296e+17 2018-02-12 03:51:23.011200  225.536331  153.708886
887  1.316296e+17 2018-02-12 09:51:21.974400  85.753303   34.006378
888  1.316296e+17 2018-02-12 15:51:21.024000  78.690558   35.500397

         delta3      delta4      theta1      theta2      theta3       theta4  \
86   33.567358   81.248635   67.960011   407.512272  63.612451   165.585550
87   25.976912   93.999416   88.948090   503.859680  73.651095   197.519216
88   32.841170   87.193192   80.706647   365.497321  85.883900   207.473196
89   36.623015   105.840151  63.743944   355.238470  74.584257   193.461227
90   25.191819   97.232429   86.755984   408.625743  64.573149   182.458502
```

```
..          ...        ...         ...         ...         ...         ...
884  121.402267  44.771501  100.667476  52.816564   128.193040  57.710658
885  91.166914   40.079455  89.966879   90.940386   96.887581   31.257395
886  189.820605  71.536294  225.670238  264.015213  207.829890  114.269363
887  103.498303  41.858699  113.615081  76.859677   149.850580  64.627288
888  83.216243   33.193961  84.812841   51.472621   88.715493   41.649183

         alpha1      alpha2      alpha3      alpha4       beta1       beta2  \
86   33.925524   162.907198  57.022700  124.072916  49.942659   210.731685
87   39.238212   112.808047  86.997672  128.031087  52.873261   138.517794
88   33.080845   94.292694   75.533254  123.768713  56.404587   143.718022
89   30.871455   144.966682  72.422374  116.980173  45.148116   206.176548
90   36.730032   156.588716  62.576354  139.567459  40.670068   208.031761
..         ...         ...        ...         ...         ...         ...
884  73.100934   80.999027   84.178656  55.898530   180.188003  153.371412
885  71.331918   74.857445   60.687237  23.724263   179.689975  199.345068
886  205.670618  163.124737  92.963880  126.173273  361.828344  249.855214
887  60.633478   73.064497   64.475270  22.612823   180.194488  178.442120
888  56.337578   71.637023   64.831803  26.952811   180.597947  173.893185

          beta3       beta4  low_gamma1  low_gamma2  low_gamma3  low_gamma4  \
86   98.466079   165.458683  18.695004   103.421166  105.274561  112.045059
87   152.987582  171.241961  14.265443   65.359496   72.104178   96.212837
88   118.482315  178.673804  14.339917   69.583152   72.095121   100.593240
89   108.270834  163.863623  16.005505   93.627380   76.516633   92.099606
90   100.449275  162.223919  14.537095   94.498731   69.536017   87.049100
..          ...         ...        ...         ...         ...         ...
884  134.589291  76.729897   73.132145   72.736170   66.563743   57.017947
885  133.518224  44.251346   69.958346   76.662801   64.934303   39.175853
886  162.831969  162.036871  103.103140  99.690895   70.906170   61.020523
887  128.550630  45.130712   70.494224   68.510785   63.167042   39.918934
888  119.663257  43.002175   77.854788   73.657659   70.356390   43.194016

     high_gamma1  high_gamma2  high_gamma3  high_gamma4         all1  \
86   16.015592    35.626594    48.474736    42.528788    246.179542
87   15.319884    27.377590    33.609060    41.340546    249.884114
88   15.511631    28.402951    44.420213    42.202363    241.541982
89   15.886343    34.121157    39.598289    37.980698    213.537969
90   15.697947    32.902126    34.724215    39.093305    239.781694
..         ...          ...          ...          ...          ...
884  35.054053    30.561880    34.055171    26.327198    565.235824
885  29.381314    29.061903    31.572590    24.162478    553.015062
886  43.167140    40.375479    33.251599    28.278869    1164.053593
887  31.170735    28.318351    34.481422    25.607426    539.689490
888  31.731855    32.480510    35.354434    28.281398    509.236989

            all2        all3        all4  i12  i34  epoch  label  patid  \
86   1191.769707  405.645495  690.393928  0.0  0.0  0      True   222_1
```

```
87   1618.783446  443.833246  725.660202  0.0  0.0  0      True   222_1
88   868.979000   428.166103  737.353398  0.0  0.0  0      True   222_1
89   1122.512570  406.710365  705.827207  0.0  0.0  0      True   222_1
90   1186.575819  356.510537  707.127559  0.0  0.0  0      True   222_1
..           ...         ...         ...  ...  ... ..       ...    ...
884  433.611692   565.892970  318.116432  0.0  1.0  11     False  231
885  520.205682   477.339595  202.079093  1.0  1.0  11     False  231
886  966.641831   753.566978  560.389747  5.0  8.0  11     False  231
887  458.521869   543.486784  239.335682  0.0  0.0  11     False  231
888  437.727781   460.643425  215.310917  0.0  0.0  11     False  231

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

[2153 rows x 36 columns]
```

0.2 2. Building Classifiers

0.2.1 Fitting 7 classfier to the training data and tune the hyperparameter using 10-fold cross-validation. Evaluate the performance of each classifier using test data

0.2.2 1:'Logistic Regression' (regulation type, regulation parameter)

0.2.3 2: 'SVM' (kernel type, degreee, regulation type, regulation parameter)

0.2.4 3: 'Gaussian Naive Bayes classifier'

0.2.5 4:'Linear Discriminant Analysis'

0.2.6 5:'Decision Tree' (criterion for splliting, max depth, min sample per leaf)

0.2.7 6:'Random Forest' (criterion for splliting, number of trees, number of features used in each tree, max depth, min sample per leaf)

0.2.8 7:'Gradient Boosting' (number of estimator, number of samples used in each estimator, max depth, min sample per leaf, learning rate)

0.3 3. Classifier Performance

0.3.1 Performace Overview of each Classifier

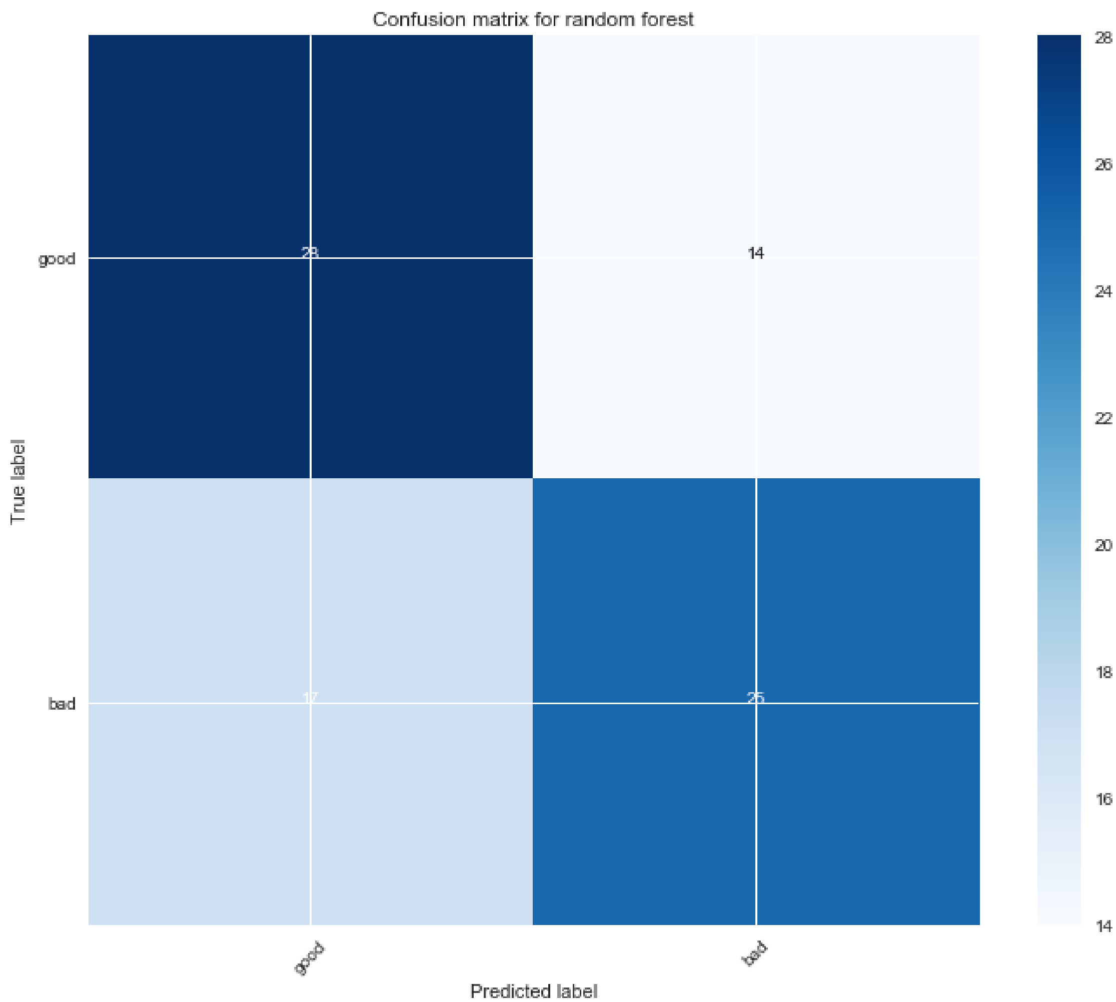
```
In [5]: X_train, X_test, y_train, y_test = JJ.get_ml_data(data, patid, if_scaler = 1, if_remove

        JJ.scores_estimators(X_test, y_test, patid = patid)
```

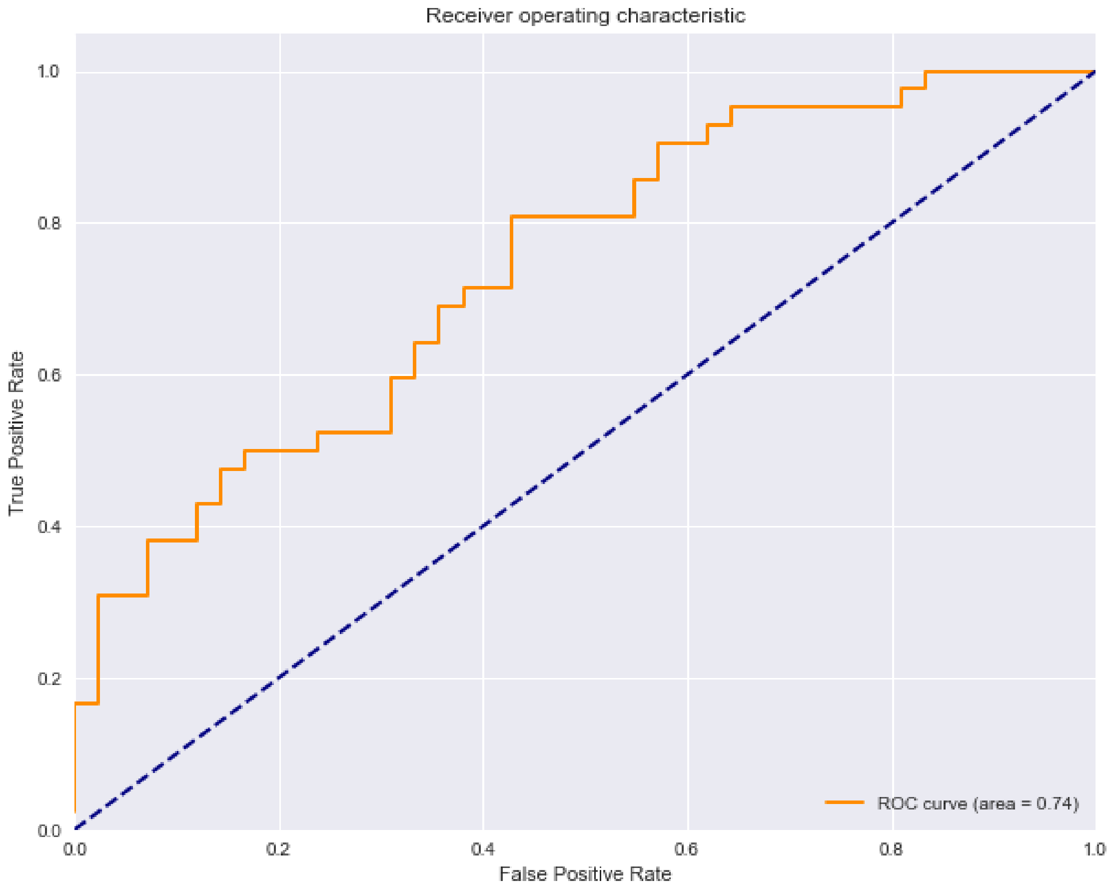
```
            Classifier       AUC
0  SVM                  0.785147
1  Logistic Regression  0.742630
2  random forest        0.742063
3  gradient boosting    0.735828
4  decision tree        0.705215


            Classifier  Accuracy
0  SVM                  0.714286
1  Logistic Regression  0.702381
2  decision tree        0.690476
3  gradient boosting    0.654762
4  random forest        0.630952
```

0.3.2 The confusion matrix and ROC of SVM(rbf kernel) (the best classifier in this case)

```
In [6]: X_train, X_test, y_train, y_test = JJ.get_ml_data(data, patid, if_scaler = 1, if_remove

        JJ.estimator_performance(6, X_test, y_test, patid = patid, if_plot_c = 1, if_plot_roc =
```

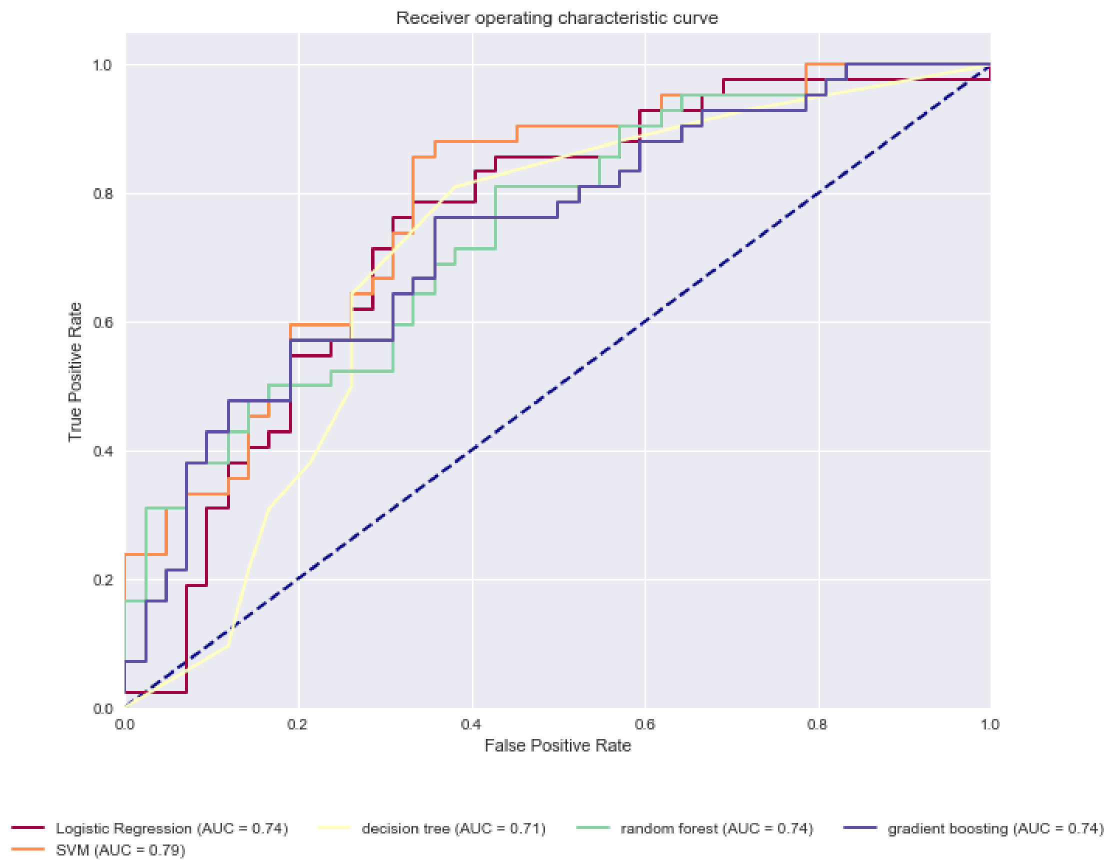
```
Confusion matrix, without normalization
```


Confusion matrix for random forest
True label
good
bad
28
14
17
25
Predicted label
good
bad
28
26
24
22
20
18
16
14



0.3.3 ROC curve for all classifiers

```
In [7]: JJ.plot_roc_all(X_test, y_test, patid = patid)
```



0.3.4 Ensemble SVM, Logistic Regression, Random Forest and Gradient Boosting using hard vote

```
In [8]: X_train, X_test, y_train, y_test = JJ.get_ml_data(data, patid, if_scaler = 1, if_remove
        #parameter_tuning(X_train, X_test, y_train, y_test, classifier = 1, C_range_num = 100,

        print("The accuracy for ensemble model is")
        JJ.ensemble_model(X_train, y_train, X_test, y_test, patid = patid,if_save = 0)
```

```
The accuracy for ensemble model is
0.654761904762
```

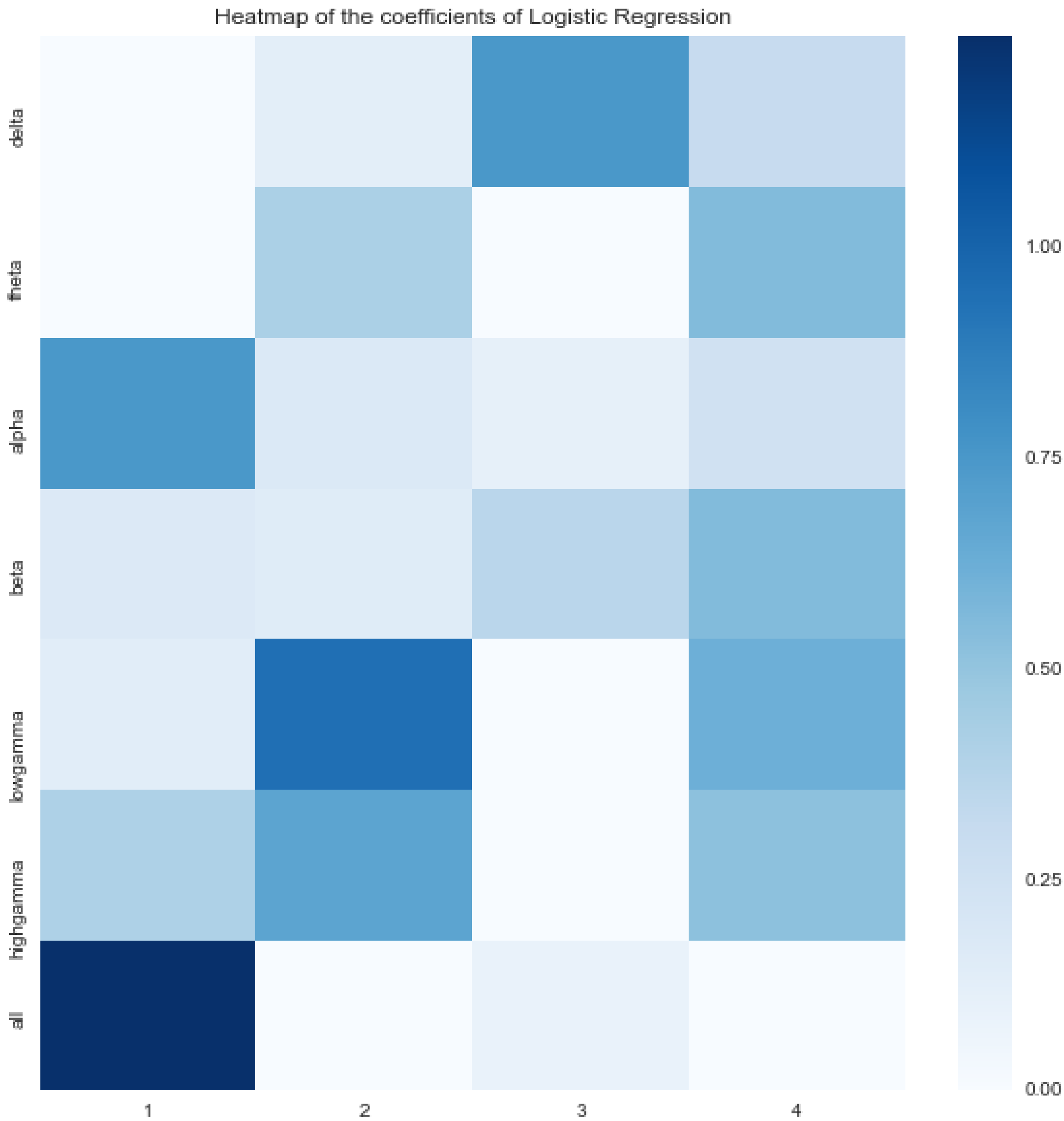
0.4 4. Feature Importance

0.4.1 Feature Importance for Logistic regression

```
In [9]: import matplotlib.pyplot as plt

        prepath = '../estimators/'+patid + '/'
        classifier_int = 1
```

```python
int2name = {1:'Logistic Regression', 2: 'SVM', 3: 'Gaussian Naive Bayes classifier', 4
clf_name = int2name[classifier_int]
clf = pickle.load(open(prepath + 'best_estimator_for_' + str(clf_name) + '.p', "rb" ))
coef = np.abs(clf.coef_.reshape(7,4))
powerband = ['delta', 'theta', 'alpha', 'beta', 'lowgamma', 'highgamma', 'all'][::-1]
channel = ['1', '2', '3', '4']
df = pd.DataFrame(coef, index = powerband, columns = channel)
import seaborn as sns
fig = plt.figure()
fig, ax = plt.subplots(1,1, figsize=(10,10))
r = sns.heatmap(coef, cmap = "Blues")
r.set_title("Heatmap of the coefficients of {}".format(clf_name))
ax.set_yticklabels(df.index)
ax.set_xticklabels(df.columns)
plt.show()
```

```
<matplotlib.figure.Figure at 0x10a7b6e10>
```

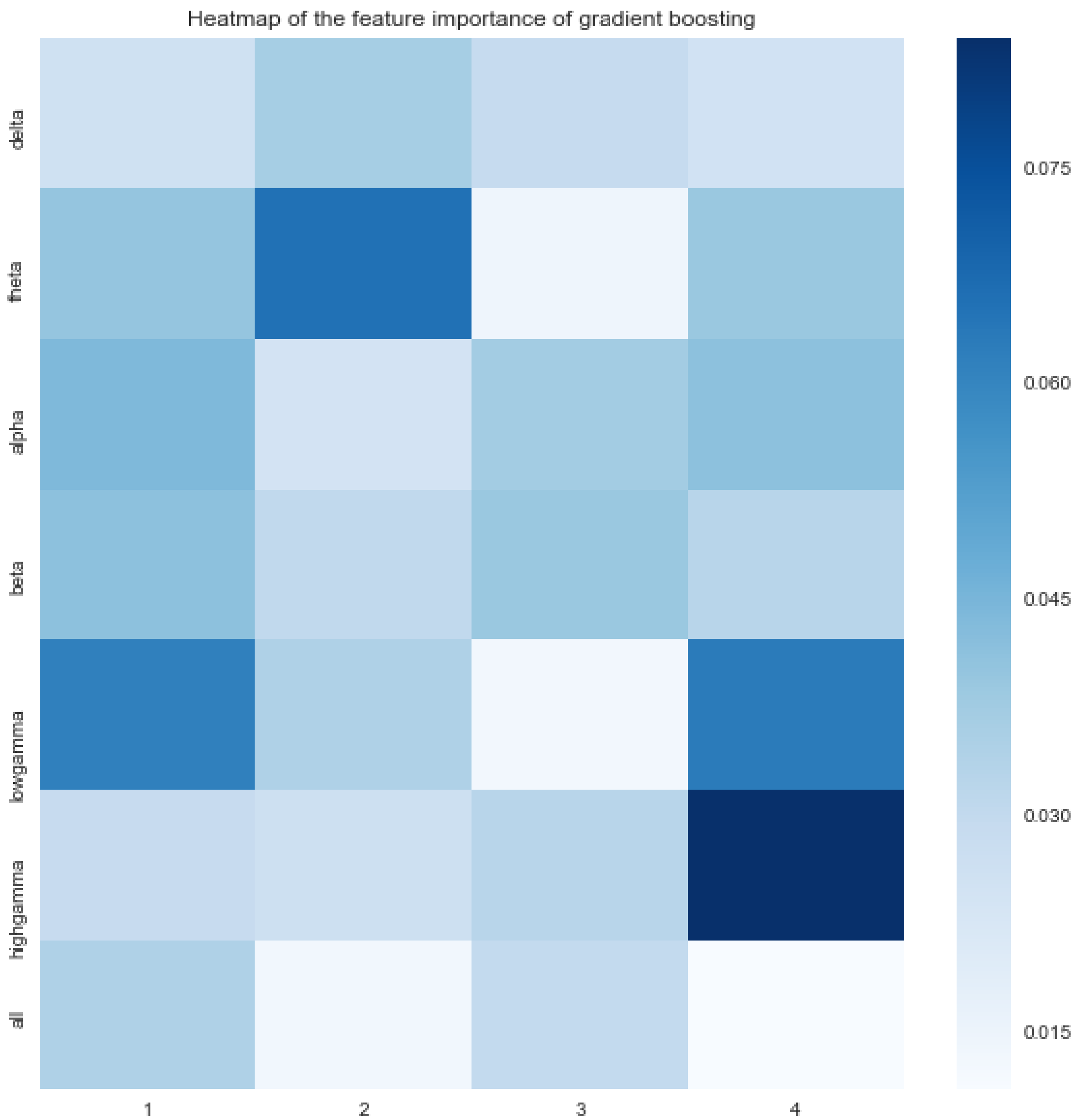


0.4.2 Feature Importance for Gradient Boosting

```
In [10]: import matplotlib.pyplot as plt
         %matplotlib inline
         prepath = '../estimators/'+patid + '/'
         classifier_int = 7
         int2name = {1:'Logistic Regression', 2: 'SVM', 3: 'Gaussian Naive Bayes classifier', 4
         clf_name = int2name[classifier_int]
         clf = pickle.load(open(prepath + 'best_estimator_for_' + str(clf_name) + '.p', "rb" ))
         coef = np.abs(clf.feature_importances_.reshape(7,4))
         powerband = ['delta', 'theta', 'alpha', 'beta', 'lowgamma', 'highgamma', 'all'][::-1]
```

```
channel = ['4', '3', '2', '1'][::-1]
df = pd.DataFrame(coef, index = powerband, columns = channel)
import seaborn as sns
fig = plt.figure()
fig, ax = plt.subplots(1,1, figsize=(10,10))
r = sns.heatmap(coef, cmap = "Blues")
r.set_title("Heatmap of the feature importance of {}".format(clf_name))
ax.set_yticklabels(df.index)
ax.set_xticklabels(df.columns)
sns.plt.show()
```

```
<matplotlib.figure.Figure at 0x10a7d04e0>
```

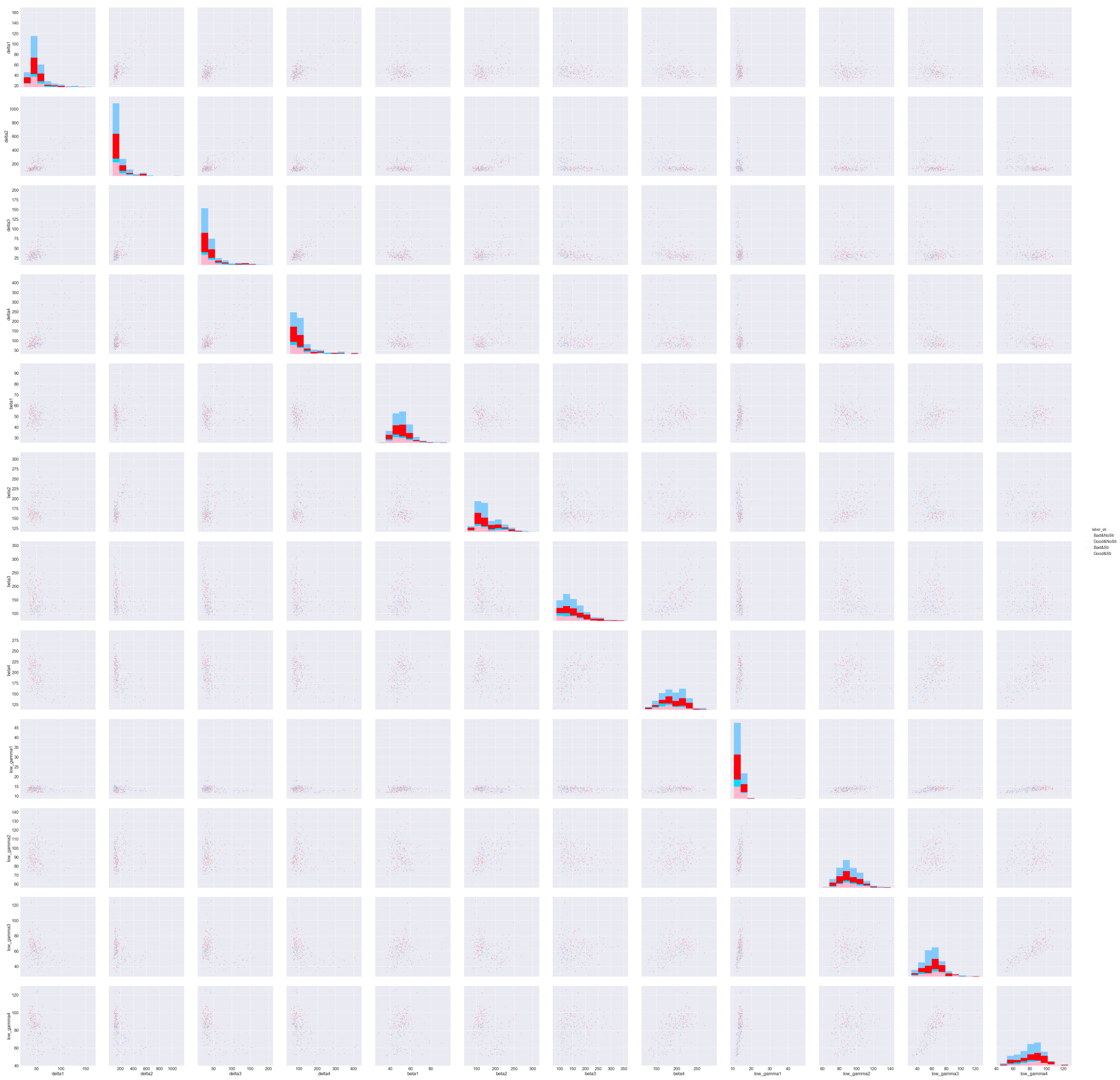


0.5 5. Data visualization

0.6 Pairwise features scatter plot

0.6.1 Each data point corresponds to a .dat file. Red points means it is in a good epoch, and blue points means it is in a bad epoch.

In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

data_ml = JJ.get_scatter_plot_data(data, patid)
sns.set(font_scale=2)
colors = ["baby pink", "neon blue", "bright red", "sky"]
g = sns.pairplot(data_ml, hue="label_sti", size = 6, vars=JJ.get_variable_name(feature
plt.show()
```



0.6.2 3D scatter plot

```
In [12]: %matplotlib notebook
         sns.set(font_scale=1)

         JJ.scatter_plot_3d(data,patid, var_list = plot_3d_var_list)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
In [ ]:
```