ML results_222_1

August 6, 2018

```
In [19]: patid = '222_1'
```

```
In [20]: 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
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

```
In [21]: features_list = ['delta', 'beta','low_gamma']
         plot_3d_var_list = ['beta2', 'beta4','low_gamma3']
```

0.1 1. Data loading

0.1.1 What the data looks like

```
In [33]: 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[33]:          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       ...         i34  epoch  label  patid  if_stimulated
         86   33.567358        ...         0.0  0      True   222_1  False
         87   25.976912        ...         0.0  0      True   222_1  False
         88   32.841170        ...         0.0  0      True   222_1  False
         89   36.623015        ...         0.0  0      True   222_1  False
         90   25.191819        ...         0.0  0      True   222_1  False
```

```
..          ...         ...         ... ..       ...      ...     ...
884  121.402267      ...         1.0  11      False  231     True
885  91.166914       ...         1.0  11      False  231     True
886  189.820605      ...         8.0  11      False  231     True
887  103.498303      ...         0.0  11      False  231     True
888  83.216243       ...         0.0  11      False  231     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

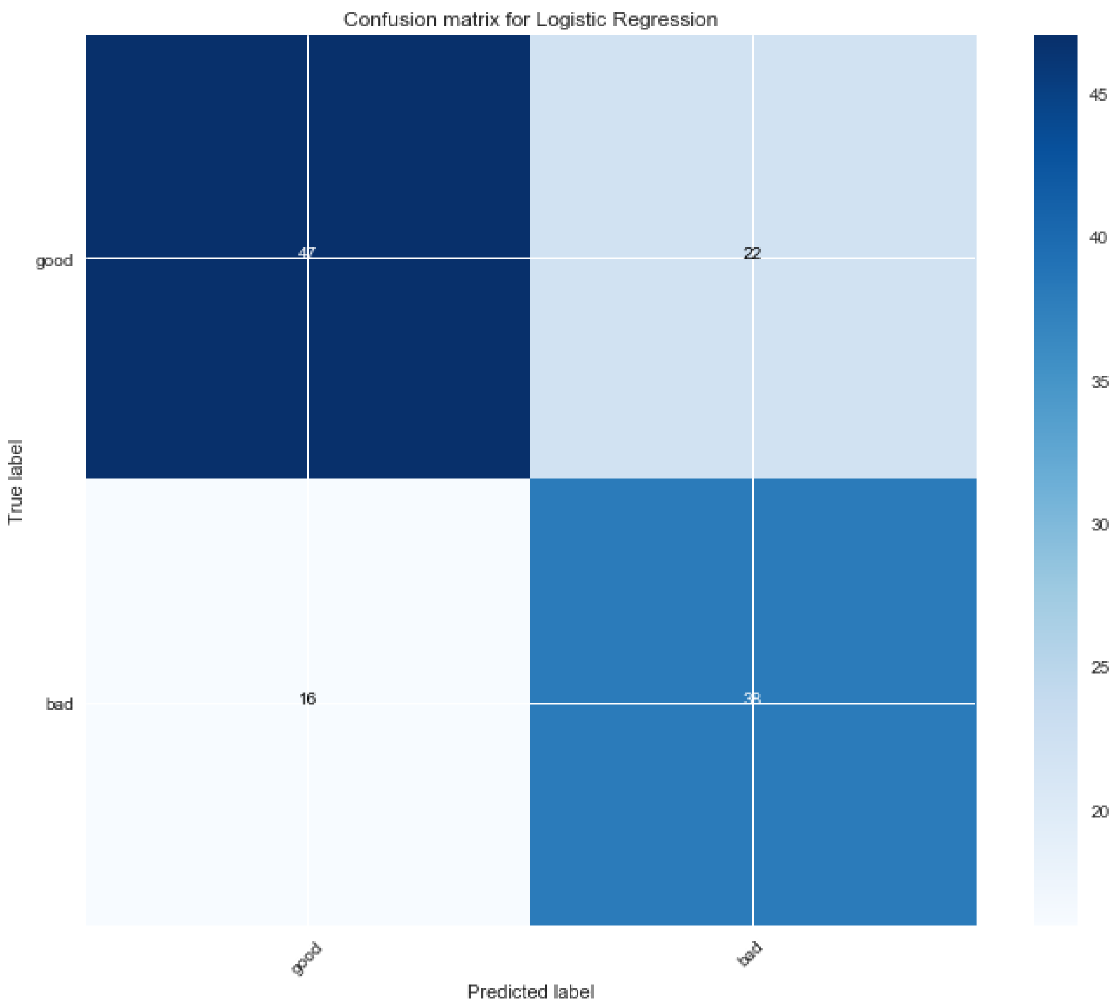
In [23]:
```
X_train, X_test, y_train, y_test = JJ.get_ml_data(data, patid, if_scaler = 1, if_remov

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

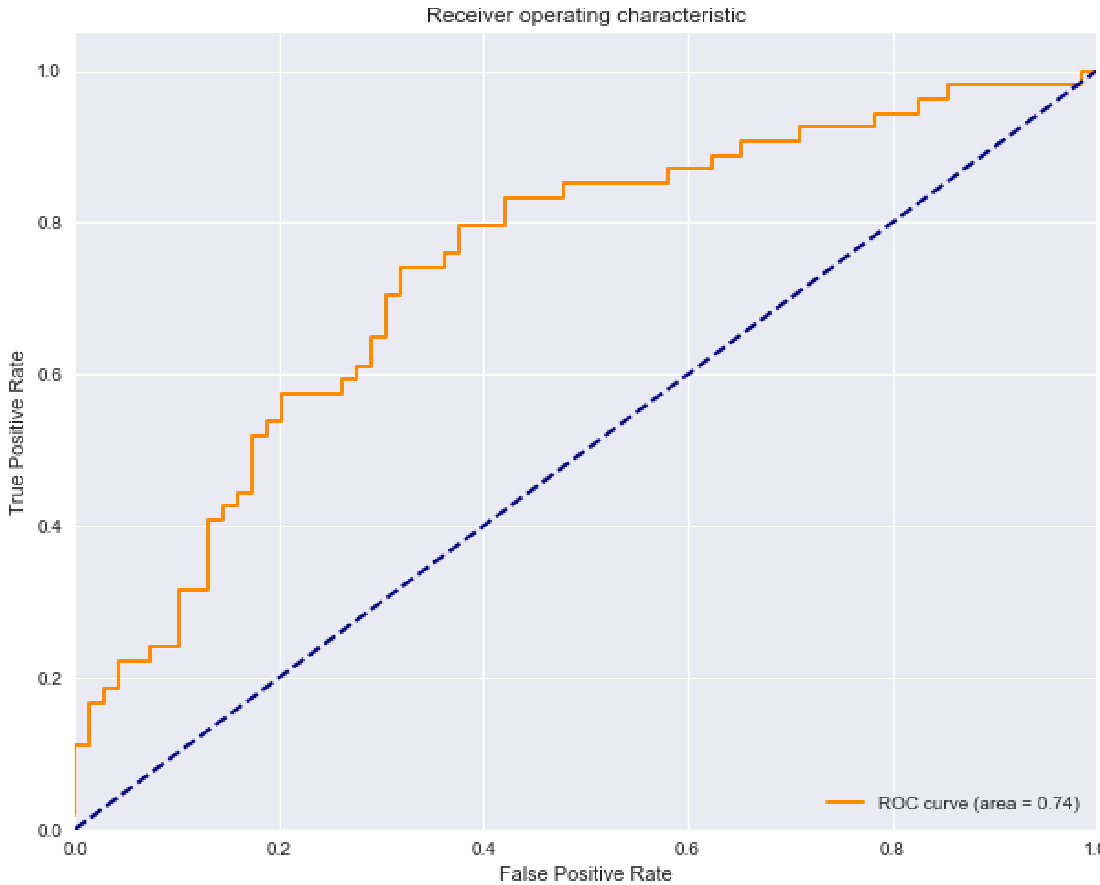
```
           Classifier       AUC
0  Logistic Regression  0.740741
1  random forest        0.731616
2  SVM                  0.731616
3  gradient boosting    0.715781
4  decision tree        0.625067


           Classifier  Accuracy
0  random forest        0.715447
1  Logistic Regression  0.691057
2  gradient boosting    0.682927
3  SVM                  0.666667
4  decision tree        0.617886
```

0.3.2 The confusion matrix and ROC of Logistic Regression (the best classifier in this case)

```
In [24]: X_train, X_test, y_train, y_test = JJ.get_ml_data(data, patid, if_scaler = 1, if_remov

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

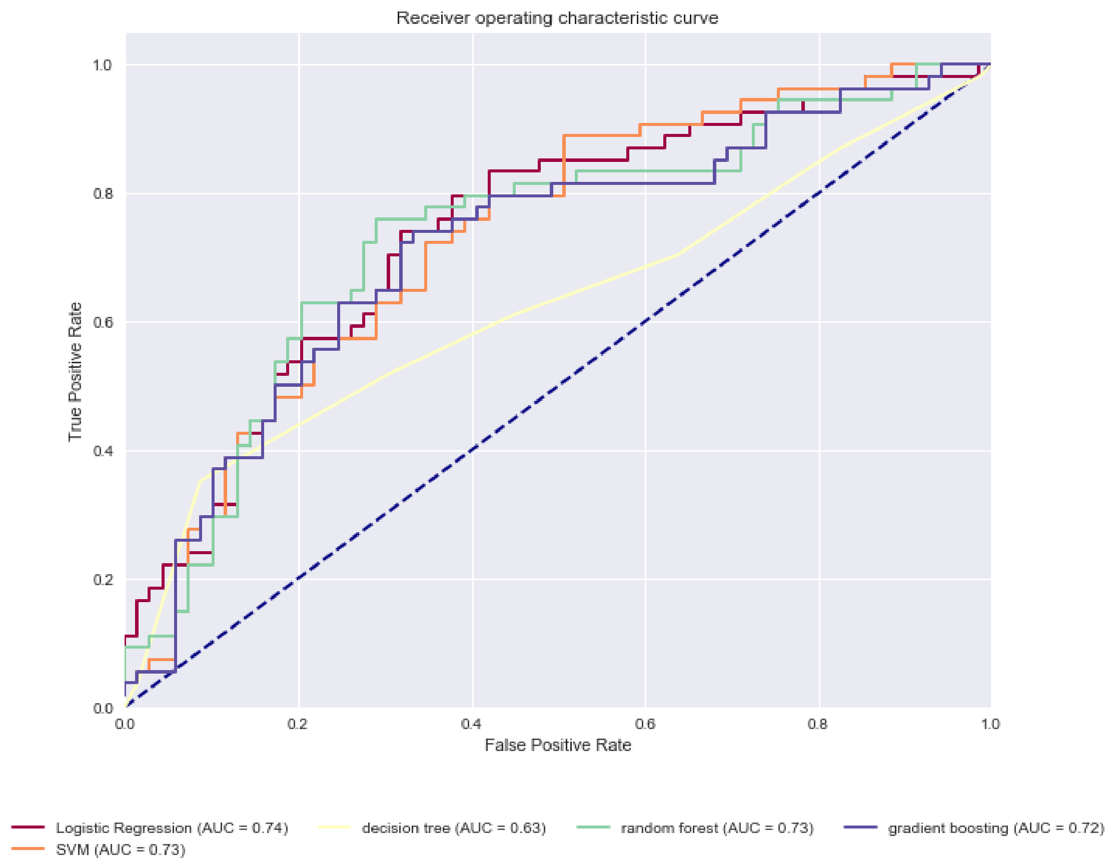
```
Confusion matrix, without normalization
```





0.3.3 ROC curve for all classifiers

```
In [25]: 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 [26]:
```
X_train, X_test, y_train, y_test = JJ.get_ml_data(data, patid, if_scaler = 1, if_remov
#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
0.682926829268
```

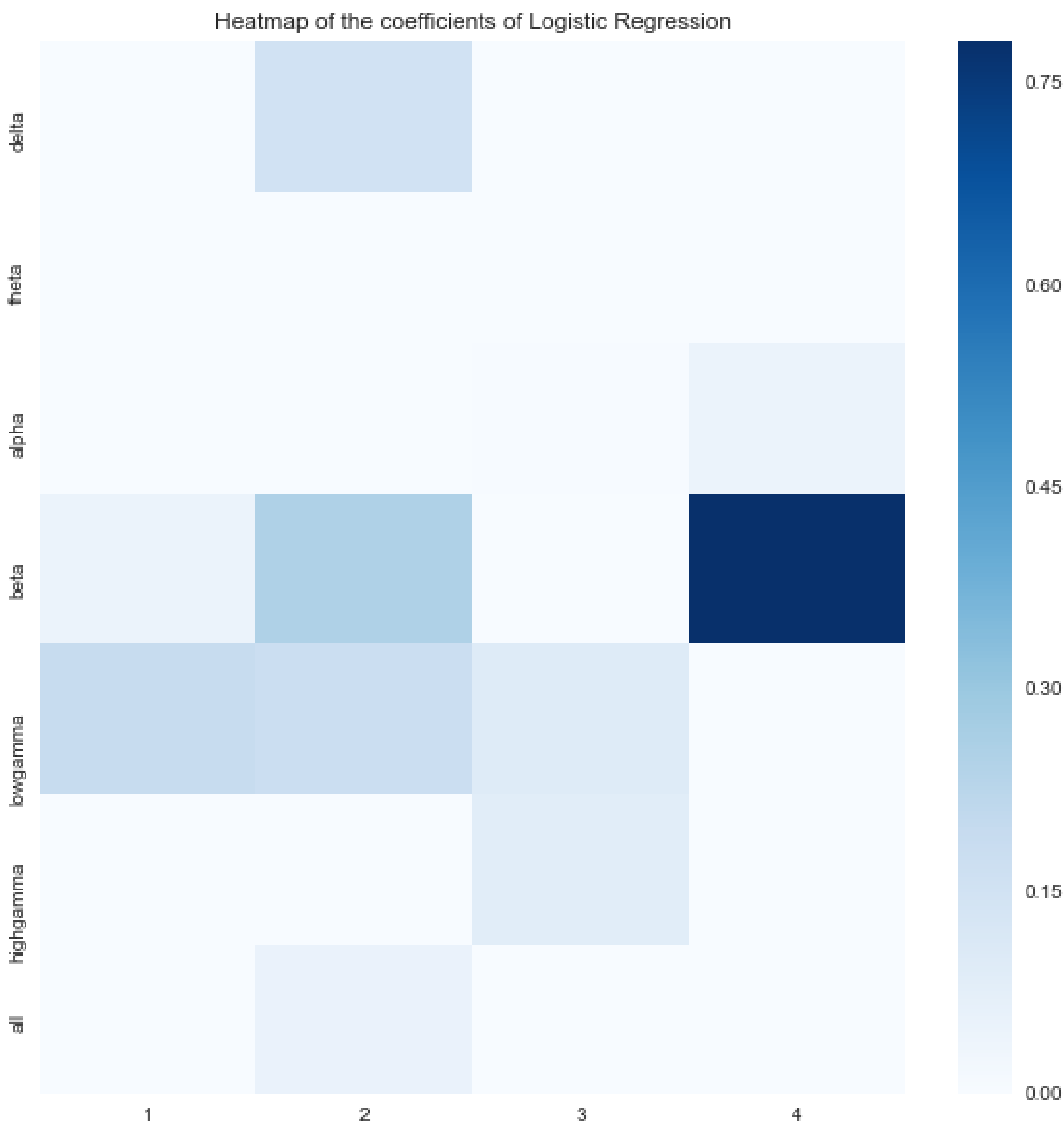
0.4 4. Feature Importance

0.4.1 Feature Importance for Logistic regression

In [27]:
```
import matplotlib.pyplot as plt

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

```
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 0x1090fc320>
```

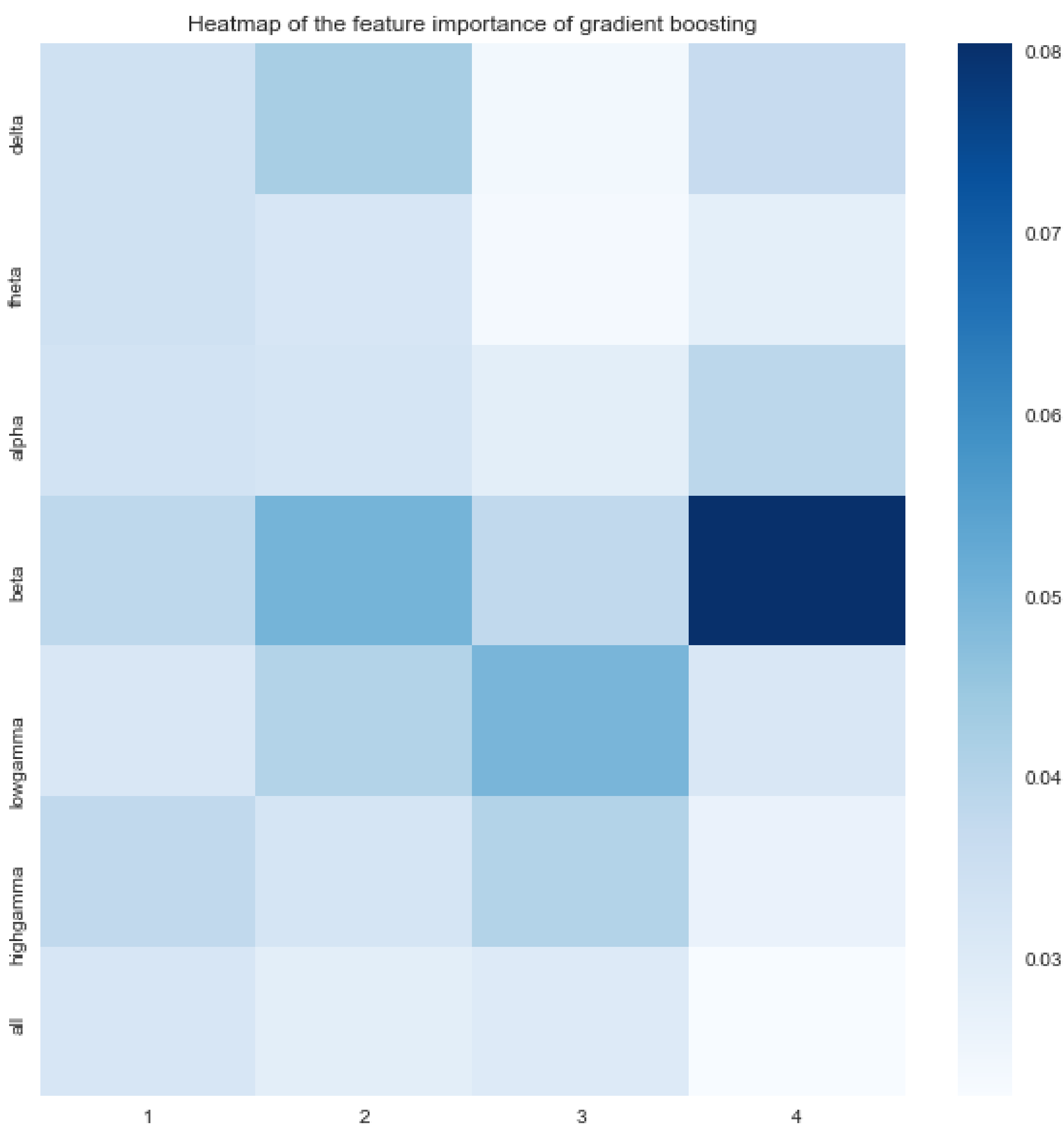


0.4.2 Feature Importance for Gradient Boosting

```
In [28]: 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 0x1058bce10>
```

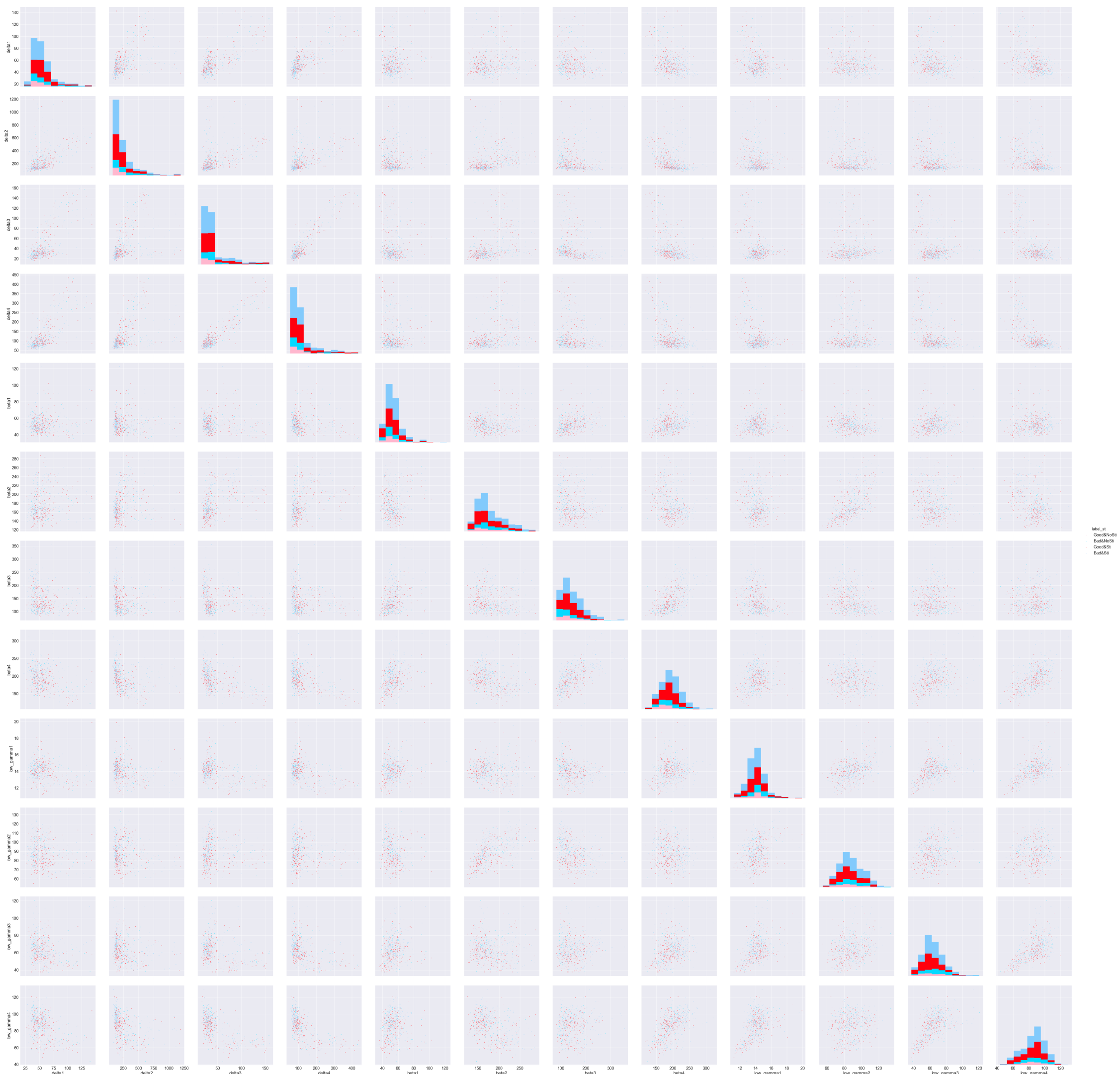


0.5 5. Data visualization

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 [31]:
```
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(featur
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 [ ]:
```