classifier

March 13, 2019

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.path as mpath
plt.rcParams['figure.figsize'] = [17, 5]

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import precision_recall_curve, average_precision_score
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This
  "This module will be removed in 0.20.", DeprecationWarning)
/anaconda3/lib/python3.6/site-packages/sklearn/grid_search.py:42: DeprecationWarning: This modul
  DeprecationWarning)
```

In [2]:
```
# load data
df_sub1 = pd.read_csv('./labeled_data/sub1label.csv')
df_sub2 = pd.read_csv('./labeled_data/sub1label.csv')

# drop unneccessary columns
df_sub1.drop('Unnamed: 0', axis=1, inplace=True)
df_sub2.drop('Unnamed: 0', axis=1, inplace=True)

df_sub1.head(10)
```

Out[2]:

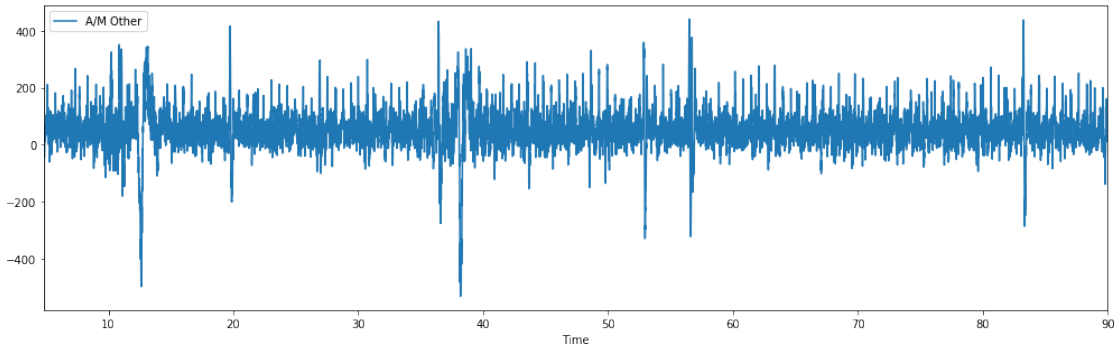
	Time	A/M Other	A/M	delta	theta	low_alpha	high_alpha	\
0	4.880859	-44.0	51.0	74.0	567109.0	74006.0	38310.0	
1	4.882812	-38.0	51.0	74.0	567109.0	74006.0	38310.0	
2	4.884766	-27.0	51.0	74.0	567109.0	74006.0	38310.0	
3	4.886719	-25.0	51.0	74.0	567109.0	74006.0	38310.0	
4	4.888672	-19.0	51.0	74.0	567109.0	74006.0	38310.0	
5	4.890625	-22.0	51.0	74.0	567109.0	74006.0	38310.0	
6	4.892578	-21.0	51.0	74.0	567109.0	74006.0	38310.0	
7	4.894531	-8.0	51.0	74.0	567109.0	74006.0	38310.0	
8	4.896484	4.0	51.0	74.0	567109.0	74006.0	38310.0	

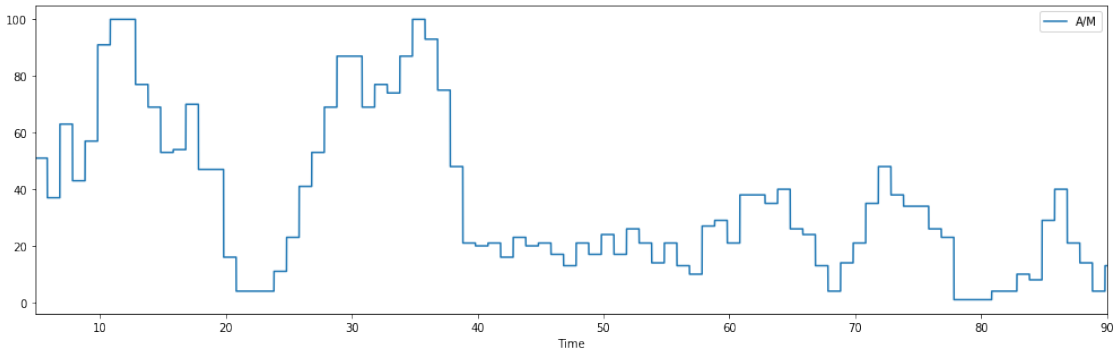
```
9  4.898438        7.0  51.0   74.0  567109.0    74006.0     38310.0

   low_beta  high_beta  low_gamma  mid_gamma  blink_stimulation  \
0    6806.0     9837.0    10228.0     1916.0              849.0
1    6806.0     9837.0    10228.0     1916.0              849.0
2    6806.0     9837.0    10228.0     1916.0              849.0
3    6806.0     9837.0    10228.0     1916.0              849.0
4    6806.0     9837.0    10228.0     1916.0              849.0
5    6806.0     9837.0    10228.0     1916.0              849.0
6    6806.0     9837.0    10228.0     1916.0              849.0
7    6806.0     9837.0    10228.0     1916.0              849.0
8    6806.0     9837.0    10228.0     1916.0              849.0
9    6806.0     9837.0    10228.0     1916.0              849.0

   blink_strength  label
0    3.783506e-44      0
1    4.764415e-43      0
2    3.363116e-44      0
3    4.203895e-44      0
4    0.000000e+00      0
5    1.386716e-38      0
6    4.049753e-43      0
7    1.191104e-43      0
8    3.783506e-44      0
9    4.764415e-43      0
```

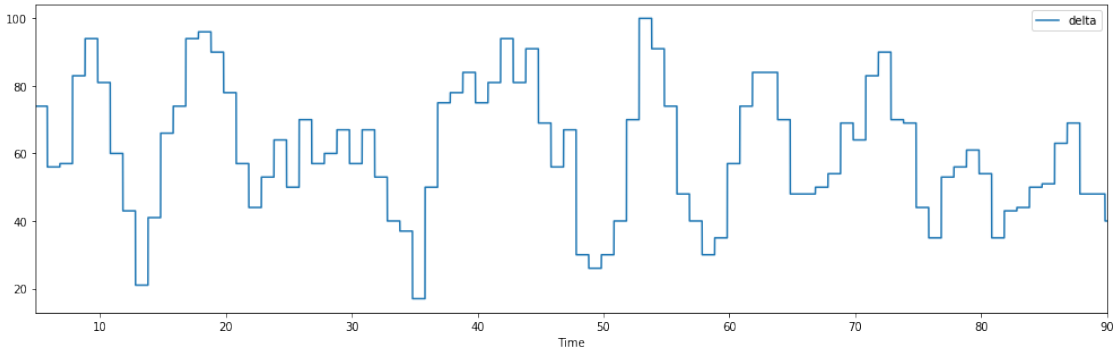
In [3]:
```
print(df_sub1.shape)
```

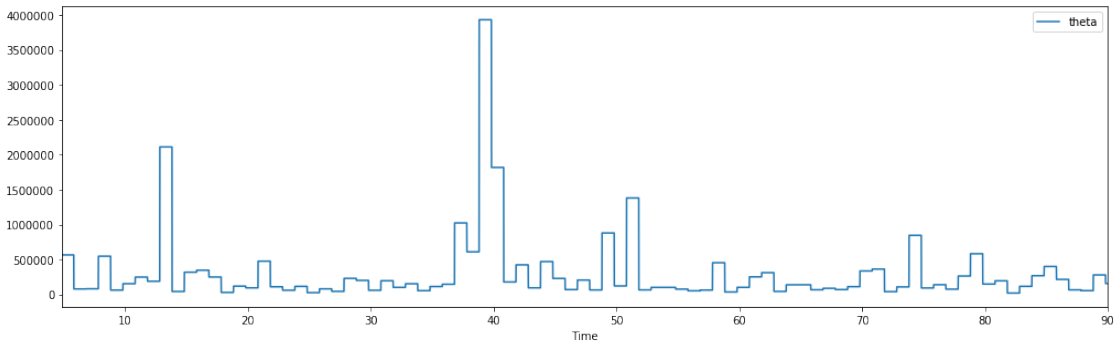
```
(43574, 14)
```

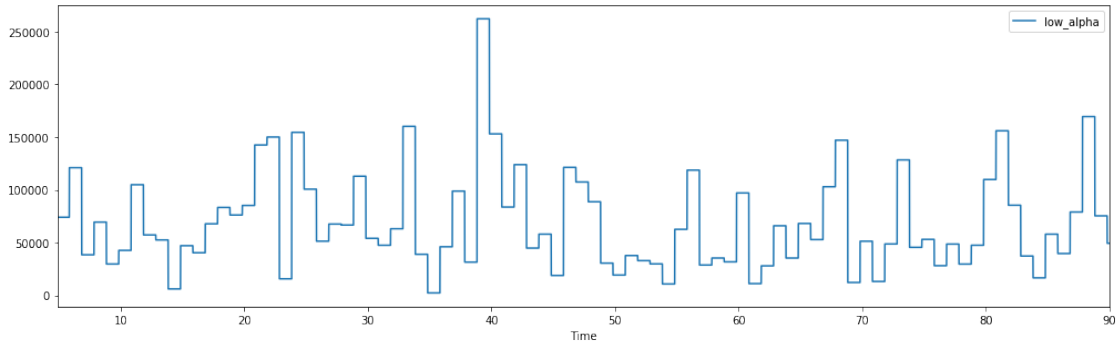
In [4]:
```
# plot all features vs time
cols = list(df_sub1)
for c in cols:
    if (c != "Time"):
        df_sub1.plot.line(x='Time', y=c)
plt.show()
```

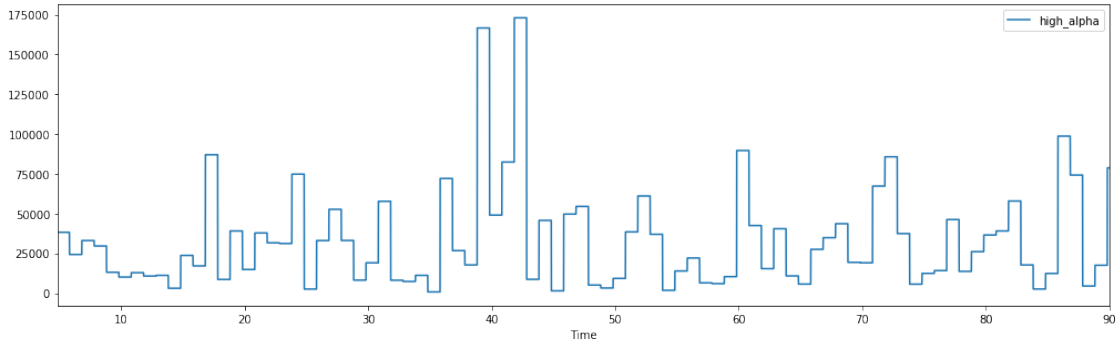


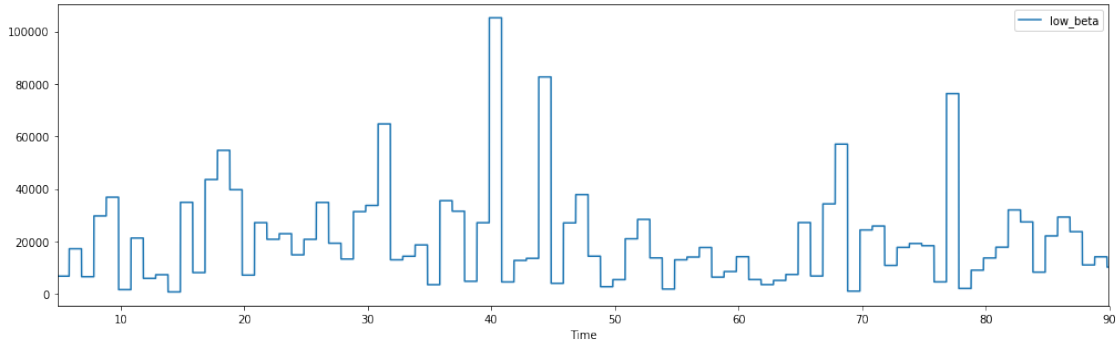

A/M
100
80
60
40
20
0
10
20
30
40
50
60
70
80
90
Time

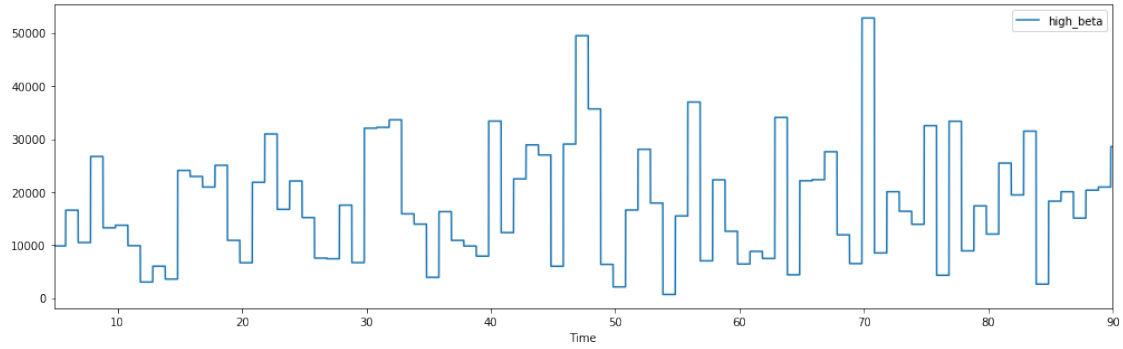

delta
100
80
60
40
20
10
20
30
40
50
60
70
80
90
Time

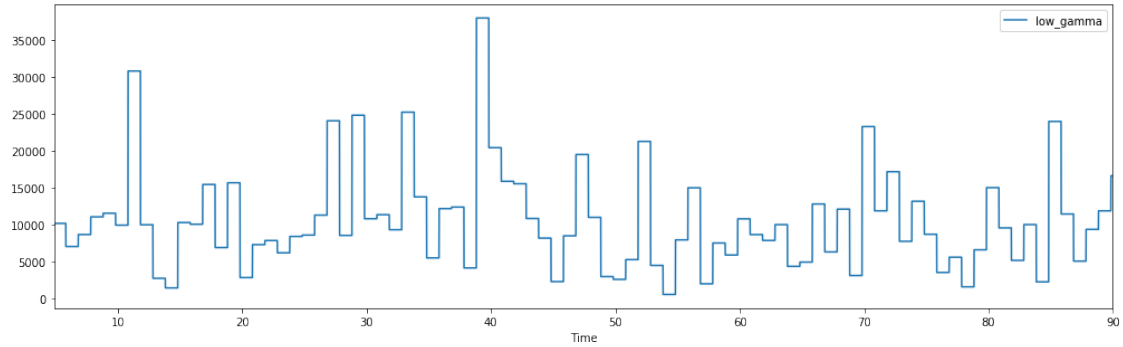

theta
4000000
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
10
20
30
40
50
60
70
80
90
Time

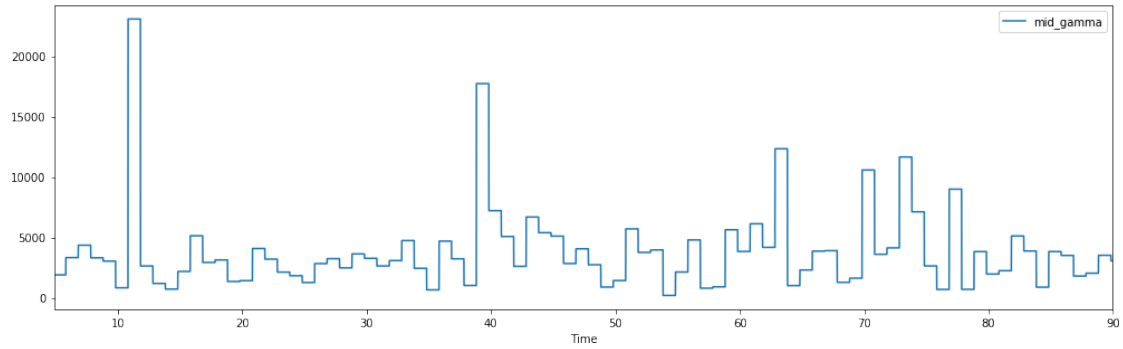

low_alpha
250000
200000
150000
100000
50000
0
10
20
30
40
50
60
70
80
90
Time

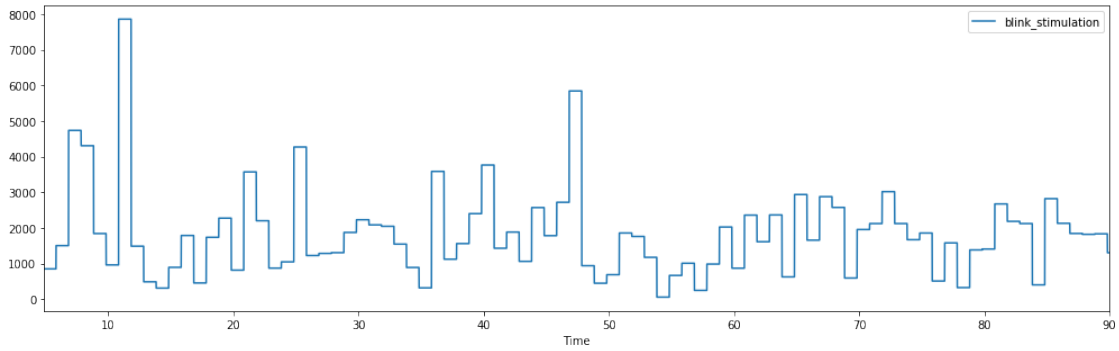

high_alpha
175000
150000
125000
100000
75000
50000
25000
0
10
20
30
40
50
60
70
80
90
Time

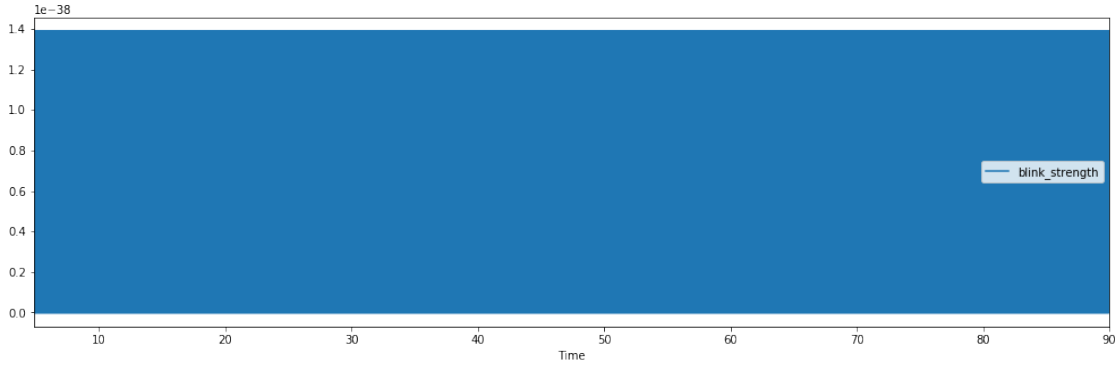

low_beta
100000
80000
60000
40000
20000
0
10
20
30
40
50
60
70
80
90
Time

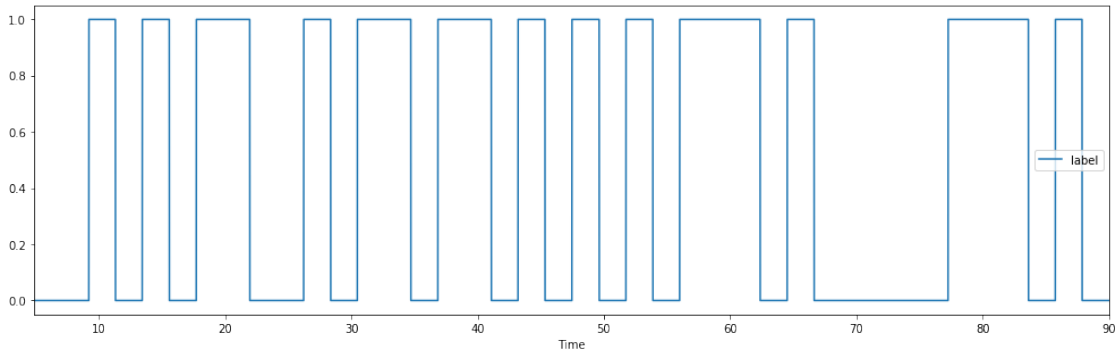

high_beta
50000
40000
30000
20000
10000
0
10
20
30
40
50
60
70
80
90
Time


low_gamma
35000
30000
25000
20000
15000
10000
5000
0
10
20
30
40
50
60
70
80
90
Time


mid_gamma
20000
15000
10000
5000
0
10
20
30
40
50
60
70
80
90
Time








```
In [5]: # correlation between features
        corr = df_sub1.corr()
        corr.style.background_gradient(cmap='coolwarm')
```

```
Out[5]: <pandas.io.formats.style.Styler at 0x1061254e0>
```

0.1 Random Forest

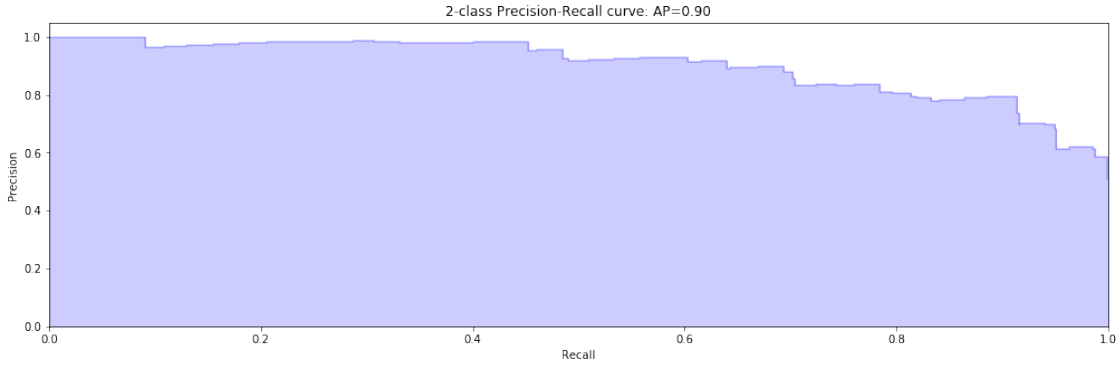
In [6]:
```
# random forest
train, test = train_test_split(df_sub1, test_size=0.2)
X = train.values[:,0:13]
Y = train.values[:,13]

X_test = test.values[:,0:13]
Y_test = test.values[:,13]

clf = RandomForestClassifier(n_estimators=200, max_depth=2, random_state=0)
clf.fit(X, Y)
for importance, feature in zip(clf.feature_importances_, cols):
    print(feature, importance)
```

```
Time 0.24131204490366298
A/M Other 0.0
A/M 0.11437825780082496
delta 0.026945840961480717
theta 0.043822915986626715
low_alpha 0.09265582860715403
high_alpha 0.07042036958345474
low_beta 0.04508016173146128
high_beta 0.1747724139410346
low_gamma 0.02750318195672576
mid_gamma 0.09739434757224781
blink_stimulation 0.0657146369553264
blink_strength 0.0
```

In [7]:
```
# accuracy
print("mean accuracy: ", clf.score(X_test, Y_test))
Y_scores = []
Y_scores = clf.predict_proba(X_test)[:,-1]
#print(Y_scores)

# cross validation
scores = cross_val_score(clf, X, Y, cv=5)
print("CV score:", scores)

# precision recall
precision, recall, thresholds = precision_recall_curve(Y_test, Y_scores)
average_precision = average_precision_score(Y_test, Y_scores)

plt.step(recall, precision, color='b', alpha=0.2,
         where='post')
plt.fill_between(recall, precision, step='post', alpha=0.2,
                 color='b')
```

```
        plt.xlabel('Recall')
        plt.ylabel('Precision')
        plt.ylim([0.0, 1.05])
        plt.xlim([0.0, 1.0])
        plt.title('2-class Precision-Recall curve: AP={0:0.2f}'.format(average_precision))
        plt.show()
```

```
mean accuracy:  0.7602983362019506
CV score: [0.7562025  0.75502008 0.72819851 0.74996414 0.75111175]
```



In [8]:
```
# perform grid search for best hyperparams (DO NOT UNCOMMENT UNLESS GRID SEARCH IS NEEDE
#param_grid = {
#    'n_estimators': [200, 500, 800],
#    'max_features': ['auto', 'sqrt', 'log2'],
#    'max_depth': [1, 2, 3]
#}
#CV_rfc = GridSearchCV(estimator=clf, param_grid=param_grid, cv=5)
#CV_rfc.fit(X, Y)
#print(CV_rfc.best_params_)
```

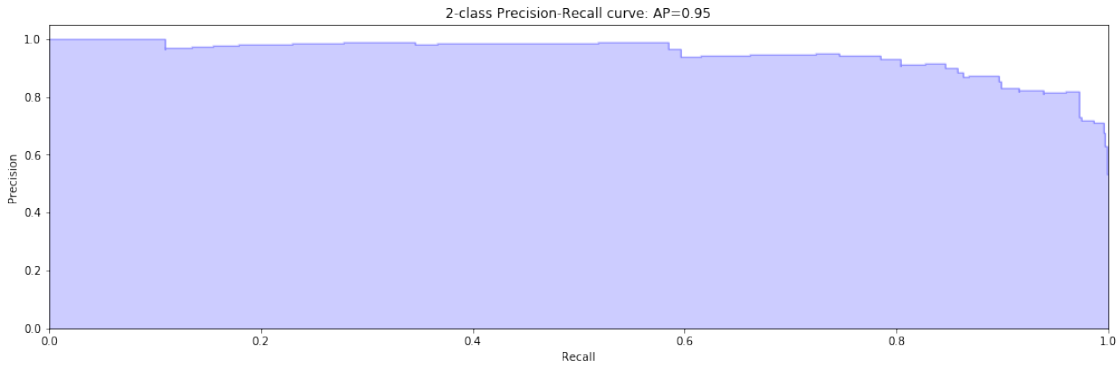
In [9]:
```
# random forest with grid searched hyperparams
clf = RandomForestClassifier(n_estimators=500, max_depth=3, max_features="auto", random_
clf.fit(X, Y)
print("mean accuracy: ", clf.score(X_test, Y_test))
```

```
mean accuracy:  0.8764199655765921
```

In [10]:
```
# accuracy
print("mean accuracy: ", clf.score(X_test, Y_test))
Y_scores = []
Y_scores = clf.predict_proba(X_test)[:,-1]
#print(Y_scores)
```

```
        # cross validation
        scores = cross_val_score(clf, X, Y, cv=5)
        print("CV score:", scores)

        # precision recall
        precision, recall, thresholds = precision_recall_curve(Y_test, Y_scores)
        average_precision = average_precision_score(Y_test, Y_scores)

        plt.step(recall, precision, color='b', alpha=0.2,
                 where='post')
        plt.fill_between(recall, precision, step='post', alpha=0.2,
                         color='b')

        plt.xlabel('Recall')
        plt.ylabel('Precision')
        plt.ylim([0.0, 1.05])
        plt.xlim([0.0, 1.0])
        plt.title('2-class Precision-Recall curve: AP={0:0.2f}'.format(average_precision))
        plt.show()
```

```
mean accuracy:  0.8764199655765921
CV score: [0.86820594 0.8759323  0.87191624 0.8806484  0.86673361]
```



In [11]:
```
        # testing on subject 2
        train2, test2 = train_test_split(df_sub2, test_size=0.2)
        X2 = train2.values[:,0:13]
        Y2 = train2.values[:,13]

        X_test2 = test2.values[:,0:13]
        Y_test2 = test2.values[:,13]

        clf.fit(X2, Y2)
```

```
        # accuracy
        print("mean accuracy: ", clf.score(X_test2, Y_test2))
        Y_scores2 = []
        Y_scores2 = clf.predict_proba(X_test2)[:,-1]
        #print(Y_scores)


        # cross validation
        scores = cross_val_score(clf, X, Y, cv=5)
        print("CV score:", scores)

        precision, recall, thresholds = precision_recall_curve(Y_test2, Y_scores2)
        average_precision = average_precision_score(Y_test2, Y_scores2)

        plt.step(recall, precision, color='b', alpha=0.2,
                 where='post')
        plt.fill_between(recall, precision, step='post', alpha=0.2,
                         color='b')

        plt.xlabel('Recall')
        plt.ylabel('Precision')
        plt.ylim([0.0, 1.05])
        plt.xlim([0.0, 1.0])
        plt.title('2-class Precision-Recall curve: AP={0:0.2f}'.format(average_precision))
        plt.show()
```

```
mean accuracy:  0.8526678141135973
CV score: [0.86820594 0.8759323  0.87191624 0.8806484  0.86673361]
```