IntelOptimisedDAAL_otherAlgorithms

August 5, 2018

In [60]:
```
import numpy as np
import os
import pandas as pd
import dill
from datetime import timedelta
from dateutil.parser import parser
import sys
import os
f = 'csv_pkl_sql.py'
for path, dirs, files in os.walk(os.path.expanduser('~/')):
    if f in files:
        dir_ = path
        break
os.chdir(dir_)
try:
    from csv_pkl_sql import save_it, csv_it, pkl_it
except:
    print("Change to the project directory")
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

In [9]:
```
with open('./pkl/11_features_engineered.pkl', 'rb') as fh:
    features = dill.load(fh) #Load features
features.head(3).T
```

Out[9]:
```
                                  0                    1  \
date                2015-11-28 00:00:00  2015-12-05 00:00:00
max_temp                             94                   93
max_temp1                            94                   94
max_temp2                            94                   94
location                Mexico-Guerrero      Mexico-Guerrero
mean_temp                            84                   82
mean_temp1                           84                   84
mean_temp2                           84                   84
min_temp                             73                   72
min_temp1                            75                   73
```

```
min_temp2                         74                  75
dew_point                         75                  74
dew_point1                        78                  75
dew_point2                        77                  78
precipitation                   1.22                   0
precipitation1                     0                1.22
precipitation2                     0                   0
wind                               4                   4
wind1                              4                   4
wind2                              4                   4
density_per_km               29.2857             29.2857
airport_dist_any            0.509027            0.509027
airport_dist_large          0.509027            0.509027
mosquito_dist              0.0166282           0.0166282
gdp                           1169.6              1169.6
gdp_ppp                       2270.7              2270.7

                                   2
date                2015-12-09 00:00:00
max_temp                          92
max_temp1                         93
max_temp2                         35
location             Mexico-Guerrero
mean_temp                         82
mean_temp1                        82
mean_temp2                        29
min_temp                          72
min_temp1                         72
min_temp2                         23
dew_point                         74
dew_point1                        74
dew_point2                        24
precipitation                      0
precipitation1                     0
precipitation2                 30.99
wind                               4
wind1                              4
wind2                              6
density_per_km               29.2857
airport_dist_any            0.509027
airport_dist_large          0.509027
mosquito_dist              0.0166282
gdp                           1169.6
gdp_ppp                       2270.7
```

In [20]:
```
from sklearn.preprocessing import Normalizer
from sklearn.model_selection import train_test_split, StratifiedShuffleSplit, KFold
from sklearn.linear_model import LogisticRegressionCV
```

```
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, f1_score, roc_auc_score, roc_curve
from sklearn.metrics import auc, accuracy_score, precision_recall_curve
from sklearn.metrics import precision_score, recall_score
from sklearn.datasets import make_blobs
from imblearn.over_sampling import SMOTE, ADASYN
from imblearn.over_sampling import RandomOverSampler
from imblearn.combine import SMOTEENN
from imblearn.combine import SMOTETomek
from imblearn.pipeline import Pipeline
import numpy as np
from scipy import interp
import matplotlib.pyplot as plt
from itertools import cycle
from sklearn.preprocessing import RobustScaler
from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import RobustScaler
scaler = Normalizer()
#scaler = StandardScaler()
smote_etomek=SMOTETomek(ratio='auto')
smote_enn = SMOTEENN(ratio='auto',random_state=0)
from tempfile import mkdtemp
from shutil import rmtree
from sklearn.decomposition import PCA
cachedir = mkdtemp()
cv = StratifiedKFold(n_splits=5,shuffle=True)
```

In [21]:
```
for col in features.columns:
    if col not in ['date', 'location']:
        features[col] = features[col].astype(np.float)
```

In [22]:
```
feat_cols = [x for x in features.columns if x not in ['date','location']]
features
```

Out[22]:

	date	max_temp	max_temp1	max_temp2	location	\
0	2015-11-28	94.0	94.0	94.0	Mexico-Guerrero	
1	2015-12-05	93.0	94.0	94.0	Mexico-Guerrero	
2	2015-12-09	92.0	93.0	35.0	Mexico-Guerrero	
3	2015-12-12	92.0	93.0	94.0	Mexico-Guerrero	
4	2015-12-16	32.0	92.0	93.0	Mexico-Guerrero	
5	2015-12-19	90.0	92.0	93.0	Mexico-Guerrero	

6	2015-12-23	33.0	32.0	92.0	Mexico-Guerrero
7	2015-12-26	90.0	90.0	92.0	Mexico-Guerrero
8	2015-12-27	94.0	90.0	90.0	Mexico-Guerrero
9	2015-12-29	35.0	33.0	32.0	Mexico-Guerrero
10	2016-01-02	94.0	90.0	90.0	Mexico-Guerrero
11	2016-01-05	34.0	35.0	33.0	Mexico-Guerrero
12	2016-01-09	93.0	94.0	90.0	Mexico-Guerrero
13	2016-01-12	33.0	34.0	35.0	Mexico-Guerrero
14	2016-01-14	33.0	34.0	35.0	Mexico-Guerrero
15	2016-01-16	91.0	93.0	94.0	Mexico-Guerrero
16	2016-01-19	33.0	33.0	34.0	Mexico-Guerrero
17	2016-01-23	93.0	91.0	93.0	Mexico-Guerrero
18	2016-01-26	33.0	33.0	33.0	Mexico-Guerrero
19	2016-01-30	91.0	93.0	91.0	Mexico-Guerrero
20	2016-02-02	33.0	33.0	33.0	Mexico-Guerrero
21	2016-02-03	33.0	33.0	33.0	Mexico-Guerrero
22	2016-02-06	33.0	91.0	93.0	Mexico-Guerrero
23	2016-02-09	35.0	33.0	33.0	Mexico-Guerrero
24	2016-02-10	35.0	33.0	33.0	Mexico-Guerrero
25	2016-02-11	35.0	33.0	33.0	Mexico-Guerrero
26	2016-02-12	35.0	33.0	33.0	Mexico-Guerrero
27	2016-02-13	35.0	33.0	91.0	Mexico-Guerrero
28	2016-02-16	34.0	35.0	33.0	Mexico-Guerrero
29	2016-02-17	34.0	35.0	33.0	Mexico-Guerrero
...	...	...	...	...	...
127959	2016-06-06	27.0	29.0	31.0	Mexico-Mexico
127960	2016-06-06	27.0	29.0	31.0	Mexico-Zacatecas
127961	2016-06-07	27.0	29.0	31.0	Mexico-Mexico
127962	2016-06-07	27.0	29.0	31.0	Mexico-Zacatecas
127963	2016-06-08	80.0	83.0	87.0	Mexico-Mexico
127964	2016-06-08	80.0	83.0	87.0	Mexico-Zacatecas
127965	2016-06-11	27.0	29.0	87.0	Mexico-Mexico
127966	2016-06-11	27.0	29.0	87.0	Mexico-Zacatecas
127967	2016-06-13	28.0	27.0	29.0	Mexico-Mexico
127968	2016-06-13	28.0	27.0	29.0	Mexico-Zacatecas
127969	2016-06-14	28.0	27.0	29.0	Mexico-Mexico
127970	2016-06-14	28.0	27.0	29.0	Mexico-Zacatecas
127971	2016-06-15	82.0	80.0	83.0	Mexico-Mexico
127972	2016-06-15	82.0	80.0	83.0	Mexico-Zacatecas
127973	2016-06-18	28.0	27.0	29.0	Mexico-Mexico
127974	2016-06-18	28.0	27.0	29.0	Mexico-Zacatecas
127975	2016-06-21	25.0	28.0	27.0	Mexico-Mexico
127976	2016-06-21	25.0	28.0	27.0	Mexico-Zacatecas
127977	2016-06-22	77.0	82.0	80.0	Mexico-Mexico
127978	2016-06-22	77.0	82.0	80.0	Mexico-Zacatecas
127979	2016-06-25	77.0	28.0	27.0	Mexico-Mexico
127980	2016-06-25	77.0	28.0	27.0	Mexico-Zacatecas
127981	2016-06-26	27.0	25.0	28.0	Mexico-Mexico

```
127982 2016-06-26      27.0       25.0       28.0  Mexico-Zacatecas
127983 2016-06-28      27.0       25.0       28.0     Mexico-Mexico
127984 2016-06-28      27.0       25.0       28.0  Mexico-Zacatecas
127985 2016-06-29      80.0       77.0       82.0     Mexico-Mexico
127986 2016-06-29      80.0       77.0       82.0  Mexico-Zacatecas
127987 2016-07-02      80.0       77.0       28.0     Mexico-Mexico
127988 2016-07-02      80.0       77.0       28.0  Mexico-Zacatecas

        mean_temp  mean_temp1  mean_temp2  min_temp  min_temp1   ...      \
0            84.0        84.0        84.0      73.0       75.0   ...
1            82.0        84.0        84.0      72.0       73.0   ...
2            82.0        82.0        29.0      72.0       72.0   ...
3            82.0        82.0        84.0      72.0       72.0   ...
4            28.0        82.0        82.0      23.0       72.0   ...
5            82.0        82.0        82.0      74.0       72.0   ...
6            28.0        28.0        82.0      23.0       23.0   ...
7            82.0        82.0        82.0      74.0       74.0   ...
8            83.0        82.0        82.0      71.0       74.0   ...
9            28.0        28.0        28.0      22.0       23.0   ...
10           83.0        82.0        82.0      71.0       74.0   ...
11           28.0        28.0        28.0      21.0       22.0   ...
12           82.0        83.0        82.0      71.0       71.0   ...
13           27.0        28.0        28.0      21.0       21.0   ...
14           27.0        28.0        28.0      21.0       21.0   ...
15           80.0        82.0        83.0      70.0       71.0   ...
16           27.0        27.0        28.0      21.0       21.0   ...
17           81.0        80.0        82.0      69.0       70.0   ...
18           27.0        27.0        27.0      20.0       21.0   ...
19           79.0        81.0        80.0      68.0       69.0   ...
20           27.0        27.0        27.0      21.0       20.0   ...
21           27.0        27.0        27.0      21.0       20.0   ...
22           27.0        79.0        81.0      21.0       68.0   ...
23           28.0        27.0        27.0      20.0       21.0   ...
24           28.0        27.0        27.0      20.0       21.0   ...
25           28.0        27.0        27.0      20.0       21.0   ...
26           28.0        27.0        27.0      20.0       21.0   ...
27           28.0        27.0        79.0      20.0       21.0   ...
28           27.0        28.0        27.0      20.0       20.0   ...
29           27.0        28.0        27.0      20.0       20.0   ...
...           ...         ...         ...       ...        ...   ...
127959       18.0        20.0        21.0      10.0       12.0   ...
127960       18.0        20.0        21.0      10.0       12.0   ...
127961       18.0        20.0        21.0      10.0       12.0   ...
127962       18.0        20.0        21.0      10.0       12.0   ...
127963       65.0        68.0        70.0      49.0       54.0   ...
127964       65.0        68.0        70.0      49.0       54.0   ...
127965       18.0        20.0        70.0      10.0       12.0   ...
127966       18.0        20.0        70.0      10.0       12.0   ...
```

127967	21.0	18.0	20.0	14.0	10.0	...
127968	21.0	18.0	20.0	14.0	10.0	...
127969	21.0	18.0	20.0	14.0	10.0	...
127970	21.0	18.0	20.0	14.0	10.0	...
127971	69.0	65.0	68.0	56.0	49.0	...
127972	69.0	65.0	68.0	56.0	49.0	...
127973	21.0	18.0	20.0	14.0	10.0	...
127974	21.0	18.0	20.0	14.0	10.0	...
127975	18.0	21.0	18.0	12.0	14.0	...
127976	18.0	21.0	18.0	12.0	14.0	...
127977	64.0	69.0	65.0	53.0	56.0	...
127978	64.0	69.0	65.0	53.0	56.0	...
127979	64.0	21.0	18.0	53.0	14.0	...
127980	64.0	21.0	18.0	53.0	14.0	...
127981	20.0	18.0	21.0	13.0	12.0	...
127982	20.0	18.0	21.0	13.0	12.0	...
127983	20.0	18.0	21.0	13.0	12.0	...
127984	20.0	18.0	21.0	13.0	12.0	...
127985	67.0	64.0	69.0	55.0	53.0	...
127986	67.0	64.0	69.0	55.0	53.0	...
127987	67.0	64.0	21.0	55.0	53.0	...
127988	67.0	64.0	21.0	55.0	53.0	...

	precipitation2	wind	wind1	wind2	density_per_km	airport_dist_any	\
0	0.00	4.0	4.0	4.0	29.285658	0.509027	
1	0.00	4.0	4.0	4.0	29.285658	0.509027	
2	30.99	4.0	4.0	6.0	29.285658	0.509027	
3	1.22	4.0	4.0	4.0	29.285658	0.509027	
4	0.00	7.0	4.0	4.0	29.285658	0.509027	
5	0.00	4.0	4.0	4.0	29.285658	0.509027	
6	0.00	8.0	7.0	4.0	29.285658	0.509027	
7	0.00	5.0	4.0	4.0	29.285658	0.509027	
8	0.25	4.0	5.0	4.0	29.285658	0.509027	
9	6.35	6.0	8.0	7.0	29.285658	0.509027	
10	0.25	4.0	5.0	4.0	29.285658	0.509027	
11	0.00	8.0	6.0	8.0	29.285658	0.509027	
12	0.00	5.0	4.0	5.0	29.285658	0.509027	
13	0.25	7.0	8.0	6.0	29.285658	0.509027	
14	0.25	7.0	8.0	6.0	29.285658	0.509027	
15	0.01	5.0	5.0	4.0	29.285658	0.509027	
16	0.00	5.0	7.0	8.0	29.285658	0.509027	
17	0.00	3.0	5.0	5.0	29.285658	0.509027	
18	10.92	6.0	5.0	7.0	29.285658	0.509027	
19	0.43	4.0	3.0	5.0	29.285658	0.509027	
20	0.00	6.0	6.0	5.0	29.285658	0.509027	
21	0.00	6.0	6.0	5.0	29.285658	0.509027	
22	0.00	6.0	4.0	3.0	29.285658	0.509027	
23	0.00	6.0	6.0	6.0	29.285658	0.509027	

```
24               0.00   6.0    6.0    6.0       29.285658           0.509027
25               0.00   6.0    6.0    6.0       29.285658           0.509027
26               0.00   6.0    6.0    6.0       29.285658           0.509027
27               0.00   6.0    6.0    4.0       29.285658           0.509027
28               0.00   8.0    6.0    6.0       29.285658           0.509027
29               0.00   8.0    6.0    6.0       29.285658           0.509027
...               ...   ...    ...    ...             ...                ...
127959           0.00  14.0   12.0   11.0        7.910104           0.561922
127960           0.00  14.0   12.0   11.0     8414.691406           0.026678
127961           0.00  14.0   12.0   11.0        7.910104           0.561922
127962           0.00  14.0   12.0   11.0     8414.691406           0.026678
127963           0.00   9.0    8.0    7.0        7.910104           0.561922
127964           0.00   9.0    8.0    7.0     8414.691406           0.026678
127965           0.00  14.0   12.0    7.0        7.910104           0.561922
127966           0.00  14.0   12.0    7.0     8414.691406           0.026678
127967           0.00  13.0   14.0   12.0        7.910104           0.561922
127968           0.00  13.0   14.0   12.0     8414.691406           0.026678
127969           0.00  13.0   14.0   12.0        7.910104           0.561922
127970           0.00  13.0   14.0   12.0     8414.691406           0.026678
127971           0.00   8.0    9.0    8.0        7.910104           0.561922
127972           0.00   8.0    9.0    8.0     8414.691406           0.026678
127973           0.00  13.0   14.0   12.0        7.910104           0.561922
127974           0.00  13.0   14.0   12.0     8414.691406           0.026678
127975           0.00  14.0   13.0   14.0        7.910104           0.561922
127976           0.00  14.0   13.0   14.0     8414.691406           0.026678
127977           0.00   9.0    8.0    9.0        7.910104           0.561922
127978           0.00   9.0    8.0    9.0     8414.691406           0.026678
127979           0.00   9.0   13.0   14.0        7.910104           0.561922
127980           0.00   9.0   13.0   14.0     8414.691406           0.026678
127981           0.00  10.0   14.0   13.0        7.910104           0.561922
127982           0.00  10.0   14.0   13.0     8414.691406           0.026678
127983           0.00  10.0   14.0   13.0        7.910104           0.561922
127984           0.00  10.0   14.0   13.0     8414.691406           0.026678
127985           0.00   6.0    9.0    8.0        7.910104           0.561922
127986           0.00   6.0    9.0    8.0     8414.691406           0.026678
127987           0.00   6.0    9.0   13.0        7.910104           0.561922
127988           0.00   6.0    9.0   13.0     8414.691406           0.026678

        airport_dist_large  mosquito_dist     gdp  gdp_ppp
0                 0.509027       0.016628  1169.6   2270.7
1                 0.509027       0.016628  1169.6   2270.7
2                 0.509027       0.016628  1169.6   2270.7
3                 0.509027       0.016628  1169.6   2270.7
4                 0.509027       0.016628  1169.6   2270.7
5                 0.509027       0.016628  1169.6   2270.7
6                 0.509027       0.016628  1169.6   2270.7
7                 0.509027       0.016628  1169.6   2270.7
8                 0.509027       0.016628  1169.6   2270.7
```

9	0.509027	0.016628	1169.6	2270.7
10	0.509027	0.016628	1169.6	2270.7
11	0.509027	0.016628	1169.6	2270.7
12	0.509027	0.016628	1169.6	2270.7
13	0.509027	0.016628	1169.6	2270.7
14	0.509027	0.016628	1169.6	2270.7
15	0.509027	0.016628	1169.6	2270.7
16	0.509027	0.016628	1169.6	2270.7
17	0.509027	0.016628	1169.6	2270.7
18	0.509027	0.016628	1169.6	2270.7
19	0.509027	0.016628	1169.6	2270.7
20	0.509027	0.016628	1169.6	2270.7
21	0.509027	0.016628	1169.6	2270.7
22	0.509027	0.016628	1169.6	2270.7
23	0.509027	0.016628	1169.6	2270.7
24	0.509027	0.016628	1169.6	2270.7
25	0.509027	0.016628	1169.6	2270.7
26	0.509027	0.016628	1169.6	2270.7
27	0.509027	0.016628	1169.6	2270.7
28	0.509027	0.016628	1169.6	2270.7
29	0.509027	0.016628	1169.6	2270.7
...	...	...	...	...
127959	10.262554	8.432757	1169.6	2270.7
127960	5.586982	4.591341	1169.6	2270.7
127961	10.262554	8.432757	1169.6	2270.7
127962	5.586982	4.591341	1169.6	2270.7
127963	10.262554	8.432757	1169.6	2270.7
127964	5.586982	4.591341	1169.6	2270.7
127965	10.262554	8.432757	1169.6	2270.7
127966	5.586982	4.591341	1169.6	2270.7
127967	10.262554	8.432757	1169.6	2270.7
127968	5.586982	4.591341	1169.6	2270.7
127969	10.262554	8.432757	1169.6	2270.7
127970	5.586982	4.591341	1169.6	2270.7
127971	10.262554	8.432757	1169.6	2270.7
127972	5.586982	4.591341	1169.6	2270.7
127973	10.262554	8.432757	1169.6	2270.7
127974	5.586982	4.591341	1169.6	2270.7
127975	10.262554	8.432757	1169.6	2270.7
127976	5.586982	4.591341	1169.6	2270.7
127977	10.262554	8.432757	1169.6	2270.7
127978	5.586982	4.591341	1169.6	2270.7
127979	10.262554	8.432757	1169.6	2270.7
127980	5.586982	4.591341	1169.6	2270.7
127981	10.262554	8.432757	1169.6	2270.7
127982	5.586982	4.591341	1169.6	2270.7
127983	10.262554	8.432757	1169.6	2270.7
127984	5.586982	4.591341	1169.6	2270.7

```
127985           10.262554       8.432757  1169.6   2270.7
127986            5.586982       4.591341  1169.6   2270.7
127987           10.262554       8.432757  1169.6   2270.7
127988            5.586982       4.591341  1169.6   2270.7

[127989 rows x 26 columns]
```

```
In [23]: framework_a_first = pd.read_pickle('./pkl/10_class_balancing_framework_a_first.pkl')
         framework_a_max   = pd.read_pickle('./pkl/10_class_balancing_framework_a_max.pkl')
```

```
In [24]: print (framework_a_first.shape, framework_a_first.isnull().sum().max())

         fwd_a_first = pd.merge(framework_a_first,
                                features,
                                on=['date','location'], how='left').dropna()

         print (fwd_a_first.shape, fwd_a_first.isnull().sum().max())

         print (fwd_a_first.zika_bool.value_counts())
```

```
(1605, 3) 0
(1213, 27) 0
1    1004
0     209
Name: zika_bool, dtype: int64
```

```
In [25]: print (framework_a_max.shape, framework_a_max.isnull().sum().max())

         fwd_a_max = pd.merge(framework_a_max,
                                features,
                                on=['date','location'], how='left').dropna()

         print (fwd_a_max.shape, fwd_a_max.isnull().sum().max())

         print (fwd_a_max.zika_bool.value_counts())
```

```
(1605, 3) 0
(1213, 27) 0
1    1004
0     209
Name: zika_bool, dtype: int64
```

```
In [26]: fwd_a_max.head()
```

```
Out[26]:                  location       date  zika_bool  max_temp  max_temp1  \
         0  Argentina-Buenos_Aires 2016-05-22          1      15.0       13.0
         1          Argentina-CABA 2016-05-22          1      15.0       13.0
```

```
2     Argentina-Catamarca 2016-05-07          1      23.0       18.0
3         Argentina-Chaco 2016-05-07          1      22.0       21.0
4        Argentina-Chubut 2016-03-19          1      26.0       26.0

   max_temp2  mean_temp  mean_temp1  mean_temp2  min_temp   ...      \
0       15.0       12.0        11.0        13.0      10.0   ...
1       15.0       12.0        11.0        13.0      10.0   ...
2       24.0       17.0        13.0        18.0      12.0   ...
3       28.0       16.0        15.0        24.0      10.0   ...
4       25.0       18.0        19.0        18.0      10.0   ...

   precipitation2  wind  wind1  wind2  density_per_km  airport_dist_any  \
0            0.00  13.0   12.0   17.0    12625.800781          0.003183
1            0.00  13.0   12.0   17.0    12625.800781          0.003183
2           12.95  14.0   13.0   14.0      460.153595          0.016047
3           73.92   5.0   10.0   11.0      121.331650          0.001590
4            3.30  15.0   17.0   15.0       37.095642          0.031922

   airport_dist_large  mosquito_dist    gdp  gdp_ppp
0            0.071514       0.008009  642.5    883.9
1            0.071514       0.008009  642.5    883.9
2           92.822158       1.188750  642.5    883.9
3           54.943508       3.194624  642.5    883.9
4          115.001996      27.500226  642.5    883.9

[5 rows x 27 columns]
```

In [49]:
```
# X = fwd_a_first[feat_cols].values
# y = fwd_a_first['zika_bool'].values
# X = fwd_a_max[feat_cols].values
# y = fwd_a_max['zika_bool'].values
fwd_a_max.head() #curated features (fwd_a_max class balance See Project description)
```

Out[49]:
```
                  location        date  zika_bool  max_temp  max_temp1  \
0  Argentina-Buenos_Aires 2016-05-22          1      15.0       13.0
1          Argentina-CABA 2016-05-22          1      15.0       13.0
2     Argentina-Catamarca 2016-05-07          1      23.0       18.0
3         Argentina-Chaco 2016-05-07          1      22.0       21.0
4        Argentina-Chubut 2016-03-19          1      26.0       26.0

   max_temp2  mean_temp  mean_temp1  mean_temp2  min_temp   ...      \
0       15.0       12.0        11.0        13.0      10.0   ...
1       15.0       12.0        11.0        13.0      10.0   ...
2       24.0       17.0        13.0        18.0      12.0   ...
3       28.0       16.0        15.0        24.0      10.0   ...
4       25.0       18.0        19.0        18.0      10.0   ...

   precipitation2  wind  wind1  wind2  density_per_km  airport_dist_any  \
```

```
0            0.00   13.0    12.0    17.0    12625.800781            0.003183
1            0.00   13.0    12.0    17.0    12625.800781            0.003183
2           12.95   14.0    13.0    14.0      460.153595            0.016047
3           73.92    5.0    10.0    11.0      121.331650            0.001590
4            3.30   15.0    17.0    15.0       37.095642            0.031922

   airport_dist_large  mosquito_dist    gdp  gdp_ppp
0            0.071514       0.008009  642.5    883.9
1            0.071514       0.008009  642.5    883.9
2           92.822158       1.188750  642.5    883.9
3           54.943508       3.194624  642.5    883.9
4          115.001996      27.500226  642.5    883.9

[5 rows x 27 columns]
```

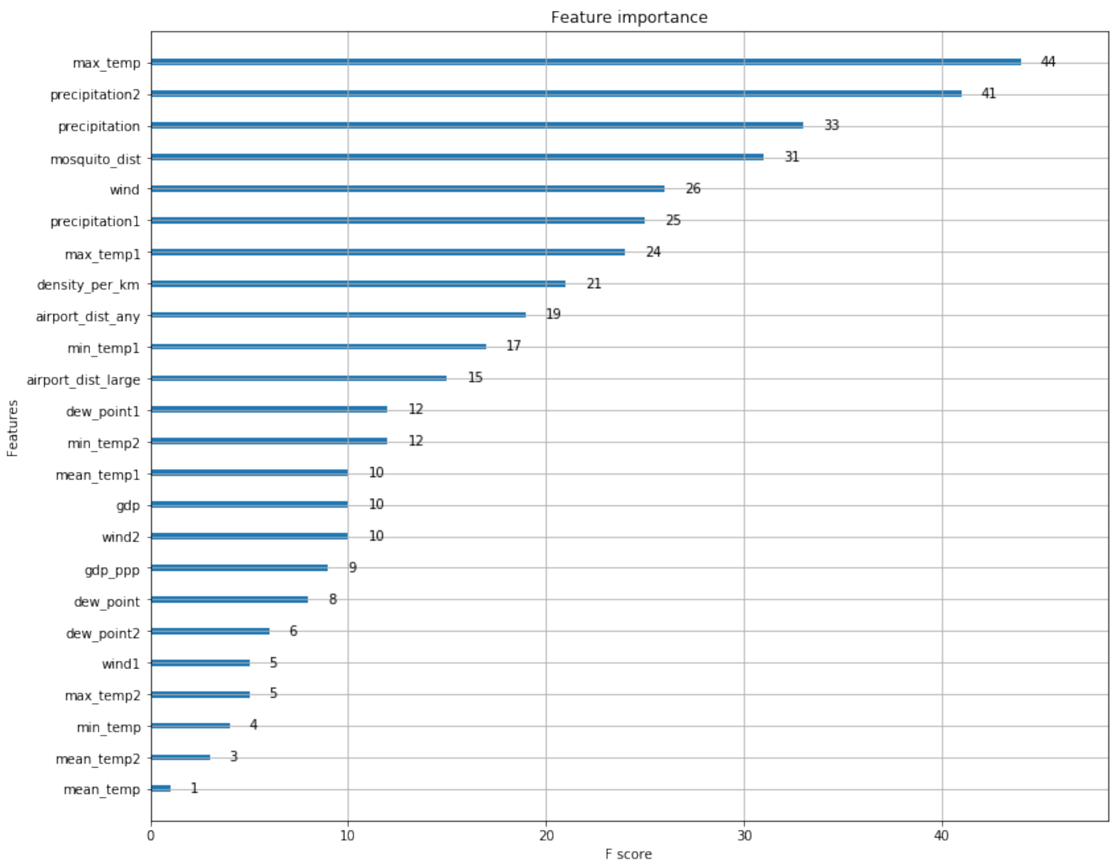
0.1 Feature Importance

```
In [36]: from xgboost import plot_importance
         import xgboost as xgb
         from xgboost import XGBClassifier
         from matplotlib import pyplot
         # load data
         # split data into X and y
         # with open('/home/abhijit/Documents/Abhijit_epidemicModel/pkl/X.pkl', 'rb') as fh:
         #         X = dill.load(fh)
         # with open('/home/abhijit/Documents/Abhijit_epidemicModel/pkl/y.pkl', 'rb') as fh:
         #         y = dill.load(fh)
         # fit model no training data
         X = fwd_a_max[feat_cols]
         y = fwd_a_max['zika_bool']
         sss = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=0)
         sss.get_n_splits(X, y)


         for train_index, test_index in sss.split(X.values, y.values):
                 #print("TRAIN:", train_index, "TEST:", test_index)
             Xtrain, Xval = X.values[train_index], X.values[test_index]
             ytrain, yval = y.values[train_index], y.values[test_index]
         model = XGBClassifier()
         scaler = Normalizer()
         smote_etomek=SMOTETomek(ratio='auto')
         pipeline = Pipeline([('scaler',scaler),('smt', smote_etomek)])
         Xtrain,ytrain = pipeline.fit_sample(Xtrain,ytrain)
         X_train = pd.DataFrame(data=Xtrain, columns=feat_cols)
         Xval = pd.DataFrame(data=Xval, columns=feat_cols)
         dtrain = xgb.DMatrix(X_train, label=ytrain)
         dtrain.feature_names
         model = xgb.train({'gamma': 0},dtrain)
```

```
# plot feature importance
def my_plot_importance(booster, figsize, **kwargs):
    from matplotlib import pyplot as plt
    from xgboost import plot_importance
    fig, ax = plt.subplots(1,1,figsize=figsize)
    return plot_importance(booster=booster, ax=ax, **kwargs)

my_plot_importance(model, figsize=(13,11))
pyplot.savefig(r"./feature_importance.png", format = 'png')
```




```
In [48]: dtrain.feature_names
```

```
Out[48]: ['max_temp',
          'max_temp1',
          'max_temp2',
          'mean_temp',
          'mean_temp1',
          'mean_temp2',
          'min_temp',
```

```
 'min_temp1',
 'min_temp2',
 'dew_point',
 'dew_point1',
 'dew_point2',
 'precipitation',
 'precipitation1',
 'precipitation2',
 'wind',
 'wind1',
 'wind2',
 'density_per_km',
 'airport_dist_any',
 'airport_dist_large',
 'mosquito_dist',
 'gdp',
 'gdp_ppp']
```

In [23]:
```
X = fwd_a_max[feat_cols].values
y = fwd_a_max['zika_bool'].values
```

In [57]:
```
X.dtype, y.dtype
y=y.astype('int32')
y = np.where(y > 0, 1, -1)
-1 in y
```

Out[57]:
```
True
```

In [58]:
```
import collections
collections.Counter(y)
```

Out[58]:
```
Counter({1: 1004, -1: 209})
```

In [61]:
```
from daal.data_management import HomogenNumericTable
import sys
#X = HomogenNumericTable(X, ntype = np.float64)
#y = y[:,np.newaxis]
#y = HomogenNumericTable(y, ntype= np.intc)
os.path.dirname(sys.executable)
```

Out[61]:
```
'/home/abhijit/anaconda3/envs/idp/bin'
```

In [62]:
```
# import dask, time
# import dask
X
```

Out[62]:
```
array([[1.50000000e+01, 1.30000000e+01, 1.50000000e+01, ...,
        8.00905560e-03, 6.42500000e+02, 8.83900000e+02],
       [1.50000000e+01, 1.30000000e+01, 1.50000000e+01, ...,
```

```
        8.00905560e-03, 6.42500000e+02, 8.83900000e+02],
       [2.30000000e+01, 1.80000000e+01, 2.40000000e+01, ...,
        1.18875049e+00, 6.42500000e+02, 8.83900000e+02],
       ...,
       [3.10000000e+01, 3.10000000e+01, 2.90000000e+01, ...,
        1.26194400e+00, 1.81207000e+04, 1.81207000e+04],
       [3.10000000e+01, 3.10000000e+01, 2.90000000e+01, ...,
        1.55761620e+00, 1.81207000e+04, 1.81207000e+04],
       [3.10000000e+01, 3.10000000e+01, 2.90000000e+01, ...,
        1.11506338e+00, 1.81207000e+04, 1.81207000e+04]])
```

In [25]:
```
import os
import sys
from daal.algorithms.adaboost import prediction, training, quality_metric_set
from daal.algorithms import classifier
from daal.data_management import (FileDataSource, DataSourceIface, HomogenNumericTable
                                  MergedNumericTable, NumericTableIface)
from utils import printNumericTables, printNumericTable
```

In [69]:
```
from sklearn.model_selection import StratifiedShuffleSplit

fam_train, fam_test = train_test_split(fwd_a_max, test_size=0.30, stratify = fwd_a_ma
```

In [70]:
```
fam_test[fam_test['zika_bool']<=0].describe()
fam_train[fam_train['zika_bool']==0].describe()

#fam_train.describe()
os.getcwd()
save_it(fam_test,"fam_test")
save_it(fam_train,"fam_train")
```

In [74]:
```
fam_test
```

Out[74]:

	location	date	zika_bool	max_temp	\
5	Argentina-Cordoba	2016-05-22	1	17.00	
609	Colombia-Cordoba-San_Jose_De_Ure	2016-06-04	1	87.68	
1129	Colombia-Sucre-Toluviejo	2016-06-18	1	33.00	
20	Argentina-Santa_Fe	2016-05-07	1	19.00	
1429	Ecuador-Los_Rios-Jaramijo	2016-05-25	1	89.00	
610	Colombia-Cordoba-San_Pelayo	2016-03-19	1	35.00	
1637	Panama-Metro-Calidonia	2016-04-11	1	34.00	
46	Brazil-Rio_de_Janeiro	2016-05-28	1	77.00	
280	Colombia-Bolivar-San_Pablo	2016-04-09	1	34.00	
1594	Nicaragua-Masaya-Nindiri	2016-02-12	1	33.00	
659	Colombia-Cundinamarca-Guayabetal	2016-05-21	1	86.00	
2	Argentina-Catamarca	2016-05-07	1	23.00	
1644	Panama-Metro-Parque_Lefevre	2016-04-11	1	34.00	
855	Colombia-Meta-San_Juan_De_Arama	2016-06-25	1	82.00	
1050	Colombia-Santander-Guaca	2016-05-14	1	26.00	

943	Colombia-Norte_Santander-Hacari	2016-06-25	1	42.05
1264	Dominican_Republic-Barahona	2016-05-07	1	30.00
655	Colombia-Cundinamarca-Guasca	2016-01-09	0	70.00
1577	Nicaragua-Carazo	2016-05-05	1	35.00
258	Colombia-Bolivar-El_Carmen_De_Bolivar	2016-04-23	1	35.00
1567	Mexico-San_Luis_Potosi	2015-11-28	0	72.00
26	Brazil-Acre	2016-05-21	1	88.00
959	Colombia-Norte_Santander-Santiago	2016-03-26	1	27.00
175	Colombia-Antioquia-Tarso	2016-01-09	0	84.00
1004	Colombia-Risaralda-Pereira	2016-06-25	1	81.00
997	Colombia-Risaralda-Belen_De_Umbria	2016-06-18	1	28.00
157	Colombia-Antioquia-San_Jeronimo	2016-04-23	1	31.00
131	Colombia-Antioquia-Liborina	2016-01-23	1	86.00
1134	Colombia-Tolima-Anzoategui	2016-06-18	1	31.00
1177	Colombia-Tolima-Venadillo	2016-02-20	1	92.00
...	...	...	...	...
288	Colombia-Bolivar-Turbaco	2016-04-02	1	32.00
1002	Colombia-Risaralda-Marsella	2016-06-11	1	28.00
1171	Colombia-Tolima-San_Antonio	2016-06-25	1	83.00
1624	Panama-Kuna_Yala-Ogobsucum	2016-04-11	1	37.00
1364	Dominican_Republic-Santiago-Caballeros	2016-02-06	1	31.00
940	Colombia-Norte_Santander-El_Tarra	2016-06-18	1	43.28
1353	Dominican_Republic-Pedernales	2016-05-07	1	30.00
1355	Dominican_Republic-Puerto_Plata	2016-06-04	1	32.00
88	Colombia-Antioquia-Caicedo	2016-06-18	1	29.00
94	Colombia-Antioquia-Carepa	2016-06-25	1	89.00
636	Colombia-Cundinamarca-Cota	2016-04-23	1	21.00
1420	Ecuador-Guayas	2016-05-04	1	90.00
670	Colombia-Cundinamarca-Madrid	2016-02-27	1	71.00
155	Colombia-Antioquia-San_Carlos	2016-06-25	1	73.00
869	Colombia-Narino-Chachagui	2016-01-09	0	75.00
142	Colombia-Antioquia-Peque	2016-02-27	1	83.00
1756	United_States_Virgin_Islands	2016-04-19	1	31.00
1372	Dominican_Republic-Santo_Domingo	2016-06-04	1	31.00
838	Colombia-Meta-El_Calvario	2016-01-09	0	89.00
1103	Colombia-Sucre-Caimito	2016-01-09	0	84.93
222	Colombia-Atlantico-Usiacuri	2016-03-05	1	33.00
1634	Panama-Metro-Ancon	2016-04-11	1	34.00
491	Colombia-Cauca-Corinto	2016-06-11	1	31.00
1650	Panama-P_Oeste	2016-04-11	1	34.00
375	Colombia-Boyaca-San_Mateo	2016-01-09	0	91.70
1669	Panama-San_Miguelito-Villa_Lucre	2016-04-11	1	35.00
60	Colombia-Amazonas-Puerto_Alegria	2016-01-09	0	88.78
996	Colombia-Risaralda-Balboa	2016-05-21	1	80.00
1659	Panama-Panama_Oeste-Vista_Alegre	2016-04-11	1	34.00
717	Colombia-Cundinamarca-Ubaque	2016-01-09	0	70.00

```
   max_temp1  max_temp2  mean_temp  mean_temp1  mean_temp2  min_temp  \
```

5	14.00	17.00	11.00	10.00	14.00	5.00
609	88.52	79.72	79.64	82.00	72.68	71.76
1129	33.00	32.00	28.00	28.00	28.00	23.00
20	17.00	22.00	15.00	13.00	18.00	11.00
1429	87.00	87.00	82.00	81.00	80.00	74.00
610	36.00	35.00	30.00	30.00	29.00	25.00
1637	34.00	34.00	29.00	29.00	29.00	25.00
46	80.00	27.00	73.00	75.00	24.00	68.00
280	32.00	35.00	29.00	28.00	30.00	24.00
1594	34.00	33.00	27.00	28.00	28.00	22.00
659	30.00	30.00	79.00	26.00	25.00	71.00
2	18.00	24.00	17.00	13.00	18.00	12.00
1644	34.00	34.00	29.00	29.00	29.00	25.00
855	27.00	30.00	76.00	23.00	25.00	69.00
1050	25.00	26.00	22.00	21.00	22.00	18.00
943	50.60	50.80	35.10	44.60	40.15	28.15
1264	31.00	32.00	27.00	27.00	28.00	23.00
655	69.00	71.00	59.00	58.00	58.00	48.00
1577	35.00	36.00	30.00	31.00	31.00	26.00
258	35.00	36.00	30.00	29.00	29.00	25.00
1567	77.00	76.00	60.00	64.00	63.00	48.00
26	32.00	32.00	81.00	28.00	26.00	74.00
959	24.00	24.00	21.00	20.00	20.00	16.00
175	85.00	86.00	75.00	76.00	75.00	65.00
1004	28.00	28.00	73.00	22.00	23.00	64.00
997	28.00	27.00	22.00	23.00	22.00	18.00
157	27.00	28.00	25.00	22.00	23.00	19.00
131	85.00	84.00	75.00	75.00	75.00	63.00
1134	29.00	28.00	25.00	25.00	23.00	20.00
1177	31.00	32.00	82.00	26.00	26.00	71.00
...	...	...	...	...	...	...
288	33.00	31.00	29.00	29.00	28.00	26.00
1002	27.00	80.00	23.00	22.00	73.00	18.00
1171	29.00	28.00	74.00	23.00	22.00	64.00
1624	36.00	37.00	30.00	30.00	29.00	24.00
1364	83.00	84.00	24.00	74.00	73.00	18.00
940	51.56	50.08	36.36	45.56	39.64	29.44
1353	31.00	32.00	27.00	27.00	28.00	23.00
1355	90.00	90.00	28.00	82.00	82.00	23.00
88	28.00	28.00	23.00	22.00	23.00	17.00
94	33.00	33.00	82.00	28.00	28.00	75.00
636	20.00	19.00	15.00	16.00	16.00	10.00
1420	86.00	91.00	83.00	81.00	84.00	76.00
670	72.00	21.00	61.00	61.00	15.00	51.00
155	23.00	23.00	64.00	18.00	18.00	56.00
869	79.00	80.00	67.00	71.00	70.00	59.00
142	87.00	30.00	74.00	77.00	24.00	65.00
1756	31.00	29.00	29.00	28.00	27.00	25.00

1372	88.00	88.00	27.00	83.00	82.00	25.00
838	87.00	90.00	81.00	79.00	79.00	72.00
1103	85.19	34.86	77.54	77.80	30.34	70.02
222	91.00	89.00	29.00	83.00	83.00	25.00
1634	34.00	34.00	29.00	29.00	29.00	25.00
491	30.00	86.00	25.00	24.00	77.00	19.00
1650	34.00	34.00	29.00	29.00	29.00	25.00
375	92.22	40.28	81.83	82.83	34.24	71.83
1669	35.00	35.00	29.00	29.00	28.00	23.00
60	89.91	39.54	78.65	78.91	34.15	68.91
996	23.00	24.00	70.00	19.00	19.00	60.00
1659	34.00	34.00	29.00	29.00	29.00	25.00
717	69.00	71.00	59.00	58.00	58.00	48.00

	...	precipitation2	wind	wind1	wind2	density_per_km	\
5	...	7.1100	8.00	8.00	5.00	2404.108887	
609	...	0.0756	8.00	8.84	9.16	32.199776	
1129	...	29.2100	1.00	1.00	2.00	67.596573	
20	...	44.9500	8.00	15.00	12.00	208.092285	
1429	...	0.0000	6.00	6.00	5.00	221.937378	
610	...	0.0000	5.00	4.00	6.00	108.661255	
1637	...	0.0000	10.00	12.00	11.00	16169.605469	
46	...	0.0000	5.00	5.00	8.00	1505.105469	
280	...	0.2500	5.00	4.00	4.00	26.240145	
1594	...	0.0000	18.00	15.00	14.00	387.784698	
659	...	10.9300	4.00	3.00	2.00	21.039572	
2	...	12.9500	14.00	13.00	14.00	460.153595	
1644	...	0.0000	10.00	12.00	11.00	6783.277832	
855	...	4.0600	3.00	2.00	3.00	8.038857	
1050	...	3.0500	7.00	7.00	7.00	22.843395	
943	...	1.1900	8.70	10.60	9.85	25.791088	
1264	...	0.0000	12.00	17.00	23.00	2050.260742	
655	...	0.0000	6.00	7.00	8.00	50.041904	
1577	...	0.0000	10.00	10.00	16.00	114.353119	
258	...	0.0000	4.00	5.00	6.00	80.454025	
1567	...	0.0200	3.00	3.00	4.00	7812.388672	
26	...	13.9700	3.00	4.00	4.00	0.466645	
959	...	6.1000	6.00	5.00	5.00	50.235493	
175	...	0.0000	4.00	7.00	5.00	82.616989	
1004	...	16.0000	5.00	8.00	9.00	1749.240479	
997	...	8.6400	8.00	9.00	7.00	152.890320	
157	...	0.0000	8.00	5.00	6.00	98.976410	
131	...	0.0000	4.00	5.00	4.00	61.181160	
1134	...	11.1800	11.00	9.00	7.00	69.729340	
1177	...	0.0000	6.00	10.00	8.00	70.368637	
...	...	...	...	...	...	...	
288	...	0.0000	12.00	13.00	14.00	600.611694	
1002	...	0.3500	9.00	7.00	5.00	171.221008	

1171	...	9.1400	2.00	3.00	3.00	35.874210
1624	...	0.0000	8.00	14.00	7.00	19.993397
1364	...	0.0000	6.00	7.00	5.00	4150.221680
940	...	1.7540	8.52	10.36	9.76	13.913511
1353	...	0.0000	12.00	17.00	23.00	85.202751
1355	...	0.0000	6.00	5.00	5.00	651.690430
88	...	0.0000	4.00	8.00	7.00	55.119736
94	...	2.0300	5.00	6.00	7.00	154.775177
636	...	5.8400	12.00	9.00	7.00	691.877075
1420	...	0.3700	7.00	6.00	6.00	75.215546
670	...	0.0000	5.00	6.00	11.00	912.494446
155	...	15.7500	3.00	6.00	5.00	30.709114
869	...	0.0000	3.00	11.00	10.00	109.786285
142	...	0.0000	5.00	8.00	9.00	30.664455
1756	...	4.8300	15.00	10.00	15.00	835.046448
1372	...	0.0000	5.00	5.00	6.00	12376.085938
838	...	0.0000	1.00	1.00	2.00	6.505045
1103	...	0.1040	6.74	5.87	7.74	38.152618
222	...	0.0000	18.00	12.00	13.00	102.548935
1634	...	0.0000	10.00	12.00	11.00	178.645111
491	...	0.0000	7.00	7.00	5.00	138.306305
1650	...	0.0000	10.00	12.00	11.00	157.148300
375	...	0.0000	5.74	3.26	5.61	22.391060
1669	...	0.0000	7.00	9.00	6.00	11670.176758
60	...	8.6620	3.00	3.00	5.48	0.370995
996	...	0.5100	7.00	7.00	5.00	81.277481
1659	...	0.0000	10.00	12.00	11.00	3720.747070
717	...	0.0000	6.00	7.00	8.00	61.370335

	airport_dist_any	airport_dist_large	mosquito_dist	gdp	gdp_ppp
5	0.009602	43.526915	0.000480	642.5	883.9
609	0.146736	11.448838	0.968325	291.5	666.9
1129	0.037814	15.699881	0.707361	291.5	666.9
20	0.023428	14.983910	0.742703	642.5	883.9
1429	0.001741	4.092887	6.513707	99.3	184.8
610	0.018464	12.587172	1.536243	291.5	666.9
1637	0.000441	0.033023	8.627815	54.3	91.1
46	0.000108	0.014630	0.012919	1799.7	3224.4
280	0.219728	7.755229	0.185934	291.5	666.9
1594	0.021393	10.679887	0.068208	12.7	32.1
659	0.042580	0.346871	0.034858	291.5	666.9
2	0.016047	92.822158	1.188750	642.5	883.9
1644	0.006769	0.013585	8.421686	54.3	91.1
855	0.638787	1.903868	0.418133	291.5	666.9
1050	0.170984	6.393379	0.124230	291.5	666.9
943	0.045049	14.107790	0.659988	291.5	666.9
1264	0.001824	2.102146	1.167112	68.2	149.9
655	0.099986	0.099986	0.402809	291.5	666.9

```
1577        0.173294     11.002629    0.002988     12.7     32.1
258         0.179585     18.632151    0.487120    291.5    666.9
1567        0.012487      8.079252    1.714500   1169.6   2270.7
26          0.755648    126.784873    0.015330   1799.7   3224.4
959         0.145214      3.466192    0.245201    291.5    666.9
175         0.179626      4.159370    0.189786    291.5    666.9
1004        0.002459      2.394093    0.077772    291.5    666.9
997         0.167177      3.215087    0.475475    291.5    666.9
157         0.068325      5.529449    0.038786    291.5    666.9
131         0.257988      6.673775    0.177248    291.5    666.9
1134        0.045742      0.901387    0.190968    291.5    666.9
1177        0.126480      0.615073    0.186751    291.5    666.9
...              ...           ...         ...      ...      ...
288         0.021592     17.360285    0.008805    291.5    666.9
1002        0.015441      2.585188    0.166600    291.5    666.9
1171        0.372290      2.397306    0.427795    291.5    666.9
1624        0.230814      2.241666    8.587769     54.3     91.1
1364        0.013112      2.149505    2.400945     68.2    149.9
940         0.137364     16.106428    0.855302    291.5    666.9
1353        0.021729      4.466269    0.465713     68.2    149.9
1355        0.014233      2.861122    2.900222     68.2    149.9
88          0.189048      6.277119    0.153849    291.5    666.9
94          0.006877      9.170558    0.124341    291.5    666.9
636         0.013432      0.013432    0.163253    291.5    666.9
1420        0.041051      2.802030    2.434152     99.3    184.8
670         0.014410      0.014410    0.051744    291.5    666.9
155         0.185233      2.924234    0.218915    291.5    666.9
869         0.001372      3.372698    0.646357    291.5    666.9
142         0.743015      8.481740    0.553103    291.5    666.9
1756        0.005879      1.233153    1.109178  18120.7  18120.7
1372        0.010561      0.071897    4.687244     68.2    149.9
838         0.043773      0.310653    0.046262    291.5    666.9
1103        0.317765     17.644941    2.415440    291.5    666.9
222         0.060384     22.206440    0.006657    291.5    666.9
1634        0.010034      0.054863    9.239864     54.3     91.1
491         0.151221      6.794006    0.144313    291.5    666.9
1650        0.147839      0.339829   10.325893     54.3     91.1
375         0.633843      5.425674    0.820884    291.5    666.9
1669        0.010415      0.010415    8.495956     54.3     91.1
60          4.555932     19.636582    8.060951    291.5    666.9
996         0.421209      6.010500    1.999031    291.5    666.9
1659        0.022327      0.119794    9.504694     54.3     91.1
717         0.093110      0.093110    0.197620    291.5    666.9

[364 rows x 27 columns]
```

In [72]:
```
from sklearn.model_selection import StratifiedShuffleSplit
sss = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=1)
```

```
sss.get_n_splits(X, y)
for train_index, test_index in sss.split(X, y):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
y_train
```

```
TRAIN: [ 695  764  393  773   14  226    3 1052   11  490 1200  276   60  507
   77  241 1195  977  845  310   35  416 1109 1149  193 1032  146   80
 1014 1131 1059  597  619  219  843  495  236  383   82  403 1020  758
 1127  622   34 1177 1049  405 1077  728  627  963  373  945  930  541
  389  249  922  967  616 1203  161  362  729 1114  306  117  726  330
  255   32 1015  474  234 1098 1166  687  442  672  670  575  287  142
  388  873  903  266 1152  273  467  626  940  171  571   76  567  305
  738  381  355 1188  178  182  353  479 1187  982 1159  410   17  113
  105  303 1208 1016  282   99 1087  529  256  497 1057  688    9 1205
  662 1108   88 1013  808  525 1042 1137 1006   68  949  734  955  787
  819 1046  217  902 1036  326  730   53  196  828  128   39  714   28
   92  398 1043 1176  503  800  325  418 1180   97  172  278  221  723
  934   23 1027  997  700   87 1073  970  549  686  492  958 1193 1158
   67   15  981  647   66  935  476  208  140  584  554 1167 1061  660
  707  801  483  655  216  120  109  650 1178 1075  522  605  519  884
  301   30  491  471  298  559  414  926  951 1174  705 1101  106 1034
   85  865  850  347 1148 1111  644  374  455  272  636   65  332 1130
  719 1126  377  708  223  784  195  379  847  744  735  214  540  646
   71  177   70  264   50  262  671  869   93  548  811  635  478   21
  402 1185  667  477  740  280 1097  387  533  858  239  164  980  275
  864  460  948  755 1047  892  511 1071   86  213 1078  628   90  776
  799 1117  612 1135  933 1085  420  653    6  116  570  689 1069  690
 1093  133  803  268  369   78  180  311  617  136  822  502  546  883
  198  361  290  590   47   22  295  327  231  953  969 1199  237  163
  844  816 1103  908 1067   42  813 1030  565  956  380  823  364  308
  939  946    2  938  141  742  370 1118  739  323 1088   79  609  602
  691  659  436  220  360  578    0  319   57 1110 1172  679  318   13
   24  451  322  356  192  753 1068  254  721  152  344  898 1089  545
 1196 1076  961 1094  494  212  151  601  432  185  448  257  331  350
  759  462  496  805  699  456  568  304  271   84  228  465  103 1173
  179   16  238 1168  454  657 1021 1079  921  645  720  731  781 1054
   56  385  999  852   54  925   41 1160 1151   64  914 1146  337   20
 1051  621  827 1161  469  270   62   49  515  855  752  992  285  493
 1186  309  832  830  971 1125  587  240  596  866  115  392  352  429
  760  677  481  895  512   75  988  818  713 1169 1017 1037  423  807
  875  851  445  243  168  202  604  788   89  449   69  974  210  878
 1201  174  727 1003  124  642 1045 1026  404  767 1104  156  307  340
  918  964  367  777  802  882  225 1134 1031  757  250  135  313  283
 1184  504 1138  694  160  698  985 1179  880  750  461 1106  631  300
  518  643  498  463  551  804  440  607  102  121  170    5  936  666
  737 1164  984 1129   25  443  562    7 1070  183  446  154  184   73
```

```
 1198 1018  431 1133  524 1084  872 1207  382  712  412  893  947 1175
  998  487  538  188  871  975  959  235 1007  768  638  932  944 1056
  874 1119 1128  484 1183  685 1096 1041 1081 1035  917  143  718  118
  281  215   10  544  615  157  909  348  696  341 1095  125  833   40
  610  663  745  466  749  245  279  641  437  315  990 1202 1116  499
 1044  269 1150  475   31  430 1209  675  582  563 1099  401 1140  915
  447    8 1012  566 1008  623  230  919   46  836  555  762  415  585
  941  153  725  763  863  252  390  613  668  962 1048  438 1155  664
  260  480  328  594  359  523  204  942  820  771 1058   94  831  835
  510  881  704  530  175  513 1050  986  104 1100 1040   19  560  629
  333   33  441  989  987  856  810  747   12  556   45  531  108  996
 1009  247  501 1171 1023  532   36  201 1072  860  357  569 1181  242
   55  779  792 1082  702  342  676 1083   27  611  710 1121  630  561
  288  904  200  419  407 1028  127  321  138  203 1147  137  937  329
 1192  785  899  408  797 1091  131  633  812  716 1074  434  150  748
  365  751  795  358  526 1157  123   72 1191 1105  346  542  867  122
  339  849  246  547  375  424   26  426  209 1212  536  439  232   95
  450  791   83  472  244  130  966  372  620  406  846   98  743 1011
  261  632 1004 1124  149 1145  927  229  991  588  681  766  765  132
  825  697  821  599  770  886  425  396  857] TEST: [ 366 1190  887  754  905  848  457 1000
  774  923  297 1066  486 1142  534  199  674 1053   51  780  741  224
  654  222  603  837  912 1062  284  277  834  581  656 1204  572  207
  550  890  957  929  840  543  453  189  593  409  706  134  509  814
  343  233  794  190  901  724  786  624  586  263  684 1170  896  859
  793  778  854 1063  320   38  422  815   74  139  251  648  553  592
  897  783 1120  796  391  885  978  580  206  316  879  678  806  824
  505  769  711  649  514  411  173  376   29  312  126 1153  692  606
  458 1055  715  395  197  335  817  841  682  906  334  468 1086  826
  972  110  181  464  583  661  907  782 1206 1002  147  669  166  296
   58   44  709  651 1162  165 1024   81  433  218  640  186  345  427
  489 1029  292 1182  839  634  973  500  162  435  703  673  145   63
  516   59  517   91  258  194  775 1005 1165  259  317  400 1123  267
 1001  717  701 1033  772 1132 1144  527  314  979  299 1080   37 1102
  148  187  413  853  993  368 1107  520  399  100  371   18  324  248
  868  652  535  293  994  155  354  552 1038  485  452  683 1139  205
  954 1154 1090  114  746 1141  891  614  302 1189 1019  101  809  169
  924 1060  394  960  338  911  129  176 1025 1194  608    4  894  693
   96  386 1163  363  976  736  916 1122  421  889  888  428  950   52
  107  286  111  600  862 1143  470  557  191  167 1112  943  658  968
  861  798  265  637  761  732  680  112  625 1065  983  227  589 1039
  444  579  144 1211   43  482  349 1064  913  473  351    1  119  537
  965  291  573  952  294  931  211  528  722  397  378  829  384  995
 1115  838  900  756  159 1092  870  733  488  459  577  521  576  877
  274  574  665  558 1010  564 1197 1136  598  253  789  842  591  508
   48 1210 1113  639  910  876  595 1156  417  336  790  506  928  618]
```

```
Out[72]: array([-1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
```

```
 1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1, -1,
 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,
 1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
-1,  1,  1,  1,  1,  1,  1,  1, -1,  1, -1,  1,  1, -1,  1,  1,  1,
 1,  1, -1,  1,  1, -1,  1,  1,  1, -1,  1, -1,  1, -1,  1,  1,  1,
 1,  1, -1,  1,  1,  1,  1, -1,  1,  1,  1,  1, -1,  1,  1,  1,  1,
 1, -1,  1, -1,  1,  1, -1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
 1, -1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1, -1,  1,  1,  1,
 1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
 1,  1, -1,  1, -1, -1, -1, -1, -1,  1,  1,  1,  1, -1,  1,  1, -1,
 1, -1,  1,  1, -1,  1,  1,  1,  1,  1,  1, -1, -1,  1,  1,  1, -1,
 1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
 1,  1,  1, -1, -1,  1,  1,  1,  1,  1, -1, -1,  1,  1,  1,  1,  1,
 1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,
 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
-1, -1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,
 1, -1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
 1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
-1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
 1,  1,  1,  1, -1,  1,  1,  1,  1,  1, -1,  1, -1,  1,  1, -1,  1,
 1,  1,  1,  1,  1, -1,  1, -1,  1,  1, -1,  1,  1,  1, -1,  1,  1,
 1,  1,  1,  1,  1,  1, -1, -1, -1,  1,  1, -1,  1,  1,  1,  1, -1,
 1,  1,  1,  1,  1,  1, -1,  1,  1, -1,  1,  1,  1,  1,  1, -1,  1,
-1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1, -1,  1,  1, -1,
 1, -1,  1,  1, -1,  1,  1,  1,  1, -1,  1, -1,  1, -1,  1,  1,  1,
 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1, -1,
 1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1, -1,  1,  1,  1,  1,
 1,  1,  1, -1,  1,  1, -1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,
 1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1, -1,  1,  1,
 1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1, -1,  1,
 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,
-1,  1,  1,  1, -1, -1,  1, -1,  1, -1,  1,  1,  1,  1,  1,  1,  1,
 1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
 1,  1,  1,  1,  1,  1, -1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
 1, -1,  1,  1, -1, -1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,
 1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,
 1, -1,  1,  1,  1, -1, -1, -1,  1, -1,  1,  1,  1,  1,  1, -1,  1,
 1,  1,  1,  1,  1, -1,  1, -1,  1,  1,  1,  1,  1,  1,  1, -1,  1,
 1,  1, -1, -1,  1,  1, -1,  1,  1, -1,  1,  1, -1,  1,  1,  1,  1,
-1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1, -1,  1,  1, -1,  1,
 1,  1, -1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,
-1,  1,  1,  1,  1, -1,  1,  1, -1,  1,  1, -1,  1,  1,  1, -1,  1,
 1,  1,  1, -1, -1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
 1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1, -1, -1,
 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,
 1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
 1, -1,  1,  1,  1, -1,  1, -1,  1,  1, -1,  1,  1,  1, -1,  1,  1,
```

```
            1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1])
```

```
In [73]: df1 = pd.DataFrame(np.column_stack((X_train,y_train)))
         df2 = pd.DataFrame(np.column_stack((X_test,y_test)))
         df1.to_csv('./csv/fam_train_unlabelled.csv',header=None, index=None)
         df2.to_csv('./csv/fam_test_unlabeleed.csv',header=None, index=None)
         #X_train.shape
         df1.head()
```

```
Out[73]:       0     1     2     3     4     5     6     7     8     9  ...     15  \
         0  82.0  84.0  85.0  74.0  75.0  75.0  66.0  66.0  66.0  68.0  ...    4.0
         1  31.0  29.0  28.0  25.0  25.0  23.0  20.0  20.0  19.0  18.0  ...   11.0
         2  23.0  21.0  22.0  15.0  16.0  16.0   8.0  11.0   9.0   8.0  ...   10.0
         3  92.0  32.0  33.0  83.0  28.0  28.0  74.0  23.0  23.0  65.0  ...    6.0
         4  19.0  20.0  26.0  14.0  14.0  19.0   8.0   8.0  12.0   9.0  ...    8.0

              16   17          18        19          20        21     22     23   24
         0   5.0  5.0  112.532761  0.446300    5.957206  0.372145  291.5  666.9 -1.0
         1   9.0  7.0   69.729340  0.045742    0.901387  0.190968  291.5  666.9  1.0
         2   9.0  9.0   61.961617  0.077390    0.077390  0.215525  291.5  666.9  1.0
         3  10.0  7.0   11.319407  0.486850    3.723189  0.711255  291.5  666.9  1.0
         4   7.0  4.0   64.008926  0.009380  107.745369  0.000087  642.5  883.9  1.0

         [5 rows x 25 columns]
```

```
In [75]: import os
         import sys
         # Input data set parameters
         trainDatasetFileName = "./csv/fam_train_unlabelled.csv.csv"
         testDatasetFileName = "./csv/fam_test_unlabeleed.csv.csv"


         from daal.data_management import (
             HomogenNumericTable, MergedNumericTable, BlockDescriptor, readWrite, readOnly
         )
         from daal.algorithms.adaboost import prediction, training
         #from daal.algorithms.svm import prediction, training
         #from daal.algorithms.brownboost import prediction, training
         from daal.algorithms import classifier
         from daal.data_management import (
             FileDataSource, DataSourceIface, HomogenNumericTable, MergedNumericTable, Numeric
         )
         import os
         import sys
         from daal.algorithms.classifier.quality_metric import binary_confusion_matrix
         from daal.algorithms import svm
         from daal.algorithms import classifier
         from daal.data_management import (
```

```
    DataSourceIface, FileDataSource, readOnly, BlockDescriptor,
    HomogenNumericTable, NumericTableIface, MergedNumericTable
)
from utils import printNumericTables, printNumericTable
nFeatures = 24
trainingResult = None
predictionResult = None
groundTruthLabels = None
trainingResult = None
predictionResult = None
AUC = []

def crossValidation(X_train, y_train):
    scaler = Normalizer()
    smote_etomek=SMOTETomek(ratio='auto')
    #smote_enn = SMOTEENN(ratio='auto')
    pipeline = Pipeline([('scaler',scaler),('smt', smote_etomek)])
    pipeline_ = Pipeline([('scaler',scaler)])
    for train, test in cv.split(X_train, y_train):
        Xtrain,ytrain = pipeline.fit_sample(X_train[train],y_train[train])
        Xtest = pipeline_.fit_transform(X_train[test])
        ytest = y_train[test]
        #print(X_train, y_train)


        def trainModel():
            global trainingResult

            # Initialize FileDataSource<CSVFeatureManager> to retrieve the input data
            trainDataSource = HomogenNumericTable(np.column_stack((Xtrain,ytrain)))

            # Create Numeric Tables for training data and labels
            trainData = HomogenNumericTable(Xtrain)
            trainGroundTruth = HomogenNumericTable(ytrain[:,np.newaxis])
            mergedData = MergedNumericTable(trainData, trainGroundTruth)

            # Retrieve the data from the input file
            block = BlockDescriptor()

            # Read one row from merged numeric table
            mergedData.getBlockOfRows(0, Xtrain.shape[0], readWrite, block)
            # Create an algorithm object to train the AdaBoost model
            algorithm = training.Batch()

            # Pass the training data set and dependent values to the algorithm
            algorithm.input.set(classifier.training.data, trainData)
            algorithm.input.set(classifier.training.labels, trainGroundTruth)
```

```
            # Train the AdaBoost model and retrieve the results of the training algor
            trainingResult = algorithm.compute()
#          print(Xtest,ytest)


        def testModel():
            global predictionResult, groundTruthLabels

            # Initialize FileDataSource<CSVFeatureManager> to retrieve the test data f
            testDataSource = HomogenNumericTable(np.column_stack((Xtest,ytest)))

            # Create Numeric Tables for testing data and labels
            testData = HomogenNumericTable(Xtest)
            groundTruthLabels = HomogenNumericTable(ytest[:,np.newaxis])
            mergedData = MergedNumericTable(testData, groundTruthLabels)

            block = BlockDescriptor()

            # Read one row from merged numeric table
            mergedData.getBlockOfRows(0, Xtest.shape[0], readWrite, block)

            # Retrieve the data from input file


            # Create algorithm objects for AdaBoost prediction with the default metho
            algorithm = prediction.Batch()
            # Pass the testing data set and trained model to the algorithm
            algorithm.input.setTable(classifier.prediction.data,  testData)
            algorithm.input.setModel(classifier.prediction.model, trainingResult.get(c

            # Compute prediction results and retrieve algorithm results
            # (Result class from classifier.prediction)
            predictionResult = algorithm.compute()
        def testModelQuality():
            global predictedLabels, qualityMetricSetResult, groundTruthLabels

            # Retrieve predicted labels
            predictedLabels = predictionResult.get(classifier.prediction.prediction)

            # Create a quality metric set object to compute quality metrics of the SV
            qualityMetricSet = svm.quality_metric_set.Batch()

            input = qualityMetricSet.getInputDataCollection().getInput(svm.quality_met

            input.set(binary_confusion_matrix.predictedLabels,   predictedLabels)
            input.set(binary_confusion_matrix.groundTruthLabels, groundTruthLabels)

            # Compute quality metrics and get the quality metrics
```

```
            # returns ResultCollection class from svm.quality_metric_set
            qualityMetricSetResult = qualityMetricSet.compute()


        def printResults():

            # Print the classification results
            printNumericTables(
                groundTruthLabels, predictedLabels,
                "Ground truth", "Classification results",
                "ADABOOST (first 20 observations):", 20, interval=15, flt64=False
            )

            # Print the quality metrics
            qualityMetricResult = qualityMetricSetResult.getResult(svm.quality_metric_
            printNumericTable(qualityMetricResult.get(binary_confusion_matrix.confusi

            block = BlockDescriptor()
            qualityMetricsTable = qualityMetricResult.get(binary_confusion_matrix.bina
            qualityMetricsTable.getBlockOfRows(0, 1, readOnly, block)
            qualityMetricsData = block.getArray().flatten()
            print("Accuracy:      {0:.3f}".format(qualityMetricsData[binary_confusion_
            print("Precision:     {0:.3f}".format(qualityMetricsData[binary_confusion_
            print("Recall:        {0:.3f}".format(qualityMetricsData[binary_confusion_
            print("F-score:       {0:.3f}".format(qualityMetricsData[binary_confusion_
            print("Specificity:   {0:.3f}".format(qualityMetricsData[binary_confusion_
            print("AUC:           {0:.3f}".format(qualityMetricsData[binary_confusion_
            AUC.append(qualityMetricsData[binary_confusion_matrix.AUC])
            qualityMetricsTable.releaseBlockOfRows(block)

        trainModel()
        #print(trainingResult)
        testModel()
        testModelQuality()
        printResults()
crossValidation(X_train,y_train)
```

```
ADABOOST (first 20 observations):
Ground truth   Classification results
1              1
1              1
-1             -1
1              1
1              1
-1             1
1              1
1              1
1              -1
```

```
1               -1
1               1
1               1
1               1
1               1
-1              1
-1              -1
1               1
-1              -1
-1              -1
1               1

Confusion matrix:
132.000   9.000
11.000    19.000

Accuracy:       0.883
Precision:      0.923
Recall:         0.936
F-score:        0.930
Specificity:    0.633
AUC:            0.785
ADABOOST (first 20 observations):
Ground truth    Classification results
1               1
1               1
1               1
1               1
1               -1
1               -1
1               1
1               -1
-1              -1
1               1
1               1
1               1
1               1
1               -1
1               1
1               -1
1               1
1               1
1               1
1               1

Confusion matrix:
113.000   28.000
4.000     25.000
```

```
Accuracy:       0.812
Precision:      0.966
Recall:         0.801
F-score:        0.876
Specificity:    0.862
AUC:            0.832
ADABOOST (first 20 observations):
Ground truth   Classification results
-1             -1
1              1
1              1
1              -1
1              1
1              1
1              1
-1             1
1              -1
1              1
1              1
1              1
1              1
1              1
1              1
-1             -1
1              1
-1             -1
1              1
1              1

Confusion matrix:
124.000   17.000
5.000     24.000

Accuracy:       0.871
Precision:      0.961
Recall:         0.879
F-score:        0.919
Specificity:    0.828
AUC:            0.854
ADABOOST (first 20 observations):
Ground truth   Classification results
1              1
-1             -1
1              1
1              1
1              -1
1              1
```

```
1              1
1              1
1              1
1              1
-1             -1
1              1
1              -1
1              1
-1             -1
1              1
1              1
1              1
-1             1
1              1

Confusion matrix:
130.000   10.000
7.000     22.000

Accuracy:      0.899
Precision:     0.949
Recall:        0.929
F-score:       0.939
Specificity:   0.759
AUC:           0.844
ADABOOST (first 20 observations):
Ground truth   Classification results
1              1
1              1
1              1
1              1
1              1
1              1
1              1
1              1
1              1
1              -1
1              1
1              -1
1              1
1              1
1              1
1              1
1              1
1              1
1              1
1              1
```

```
Confusion matrix:
130.000   10.000
8.000     21.000

Accuracy:      0.893
Precision:     0.942
Recall:        0.929
F-score:       0.935
Specificity:   0.724
AUC:           0.826
```

```
In [77]: import sys
         import os
         # sys.path.append(r'..')
         # sys.path.append(os.path.join(os.path.dirname(sys.executable),'share','pydaal_exampl
         import numpy as np
         from SVM import BinarySVM
         from daal.data_management import HomogenNumericTable
         from utils import printNumericTables, printNumericTable
         from sklearn.datasets import load_breast_cancer
         from sklearn.model_selection import train_test_split

         # Create train and test datasets
         # data = load_breast_cancer()
         # x = data.data
         # y = data.target
         # y[y==0]=-1 # DAAL's SVM binary classifier labels must be -1 and 1
         from sklearn.model_selection import StratifiedShuffleSplit
         sss = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=0)
         sss.get_n_splits(X, y)
         for train_index, test_index in sss.split(X, y):
             #print("TRAIN:", train_index, "TEST:", test_index)
             X_train, X_test = X[train_index], X[test_index]
             y_train, y_test = y[train_index], y[test_index]
         #x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.40, random_sta
         trainData = HomogenNumericTable(X_train)
         testData=HomogenNumericTable(X_test)
         nD_y_train= y_train[:,np.newaxis]
         trainDependentVariables= HomogenNumericTable(nD_y_train)
         nD_y_test = y_test[:,np.newaxis]
         testGroundTruth = HomogenNumericTable(nD_y_test)


         daal_svm = BinarySVM(cacheSize=6000000)
         #Train
         trainingResult = daal_svm.training(trainData,trainDependentVariables)
         #Predict
```

```python
predictResults = daal_svm.predict(trainingResult,testData)
#Evaluate you model
qualityMet = daal_svm.qualityMetrics(predictResults,testGroundTruth)
#print accuracy
print(qualityMet.get('accuracy'))
#print confusion matrix
printNumericTable(qualityMet.get('confusionMatrix'))
#print all metrics
daal_svm.printAllQualityMetrics(qualityMet)
#Serialize
daal_svm.serialize(trainingResult, fileName='svm', useCompression=True)
#Deserialize
dese_trainingRes = daal_svm.deserialize(fileName='svm.npy', useCompression=True)

#Print predicted responses and actual responses
printNumericTables (
        testGroundTruth, predictResults,
        "Ground truth", "Classification results",
        "SVM classification results (first 20 observations):", 20,  flt64=False
)
```

```
0.760989010989011

239.000   62.000
25.000    38.000

Confusion matrix:
239.000   62.000
25.000    38.000

Accuracy:       0.761
Precision:      0.905
Recall:         0.794
F1-score:       0.846
Specificity:    0.603
AUC:            0.699
SVM classification results (first 20 observations):
Ground truth   Classification results
1              5
1              2
1              3
1              5
1              -2
1              2
1              -1
1              1
1              5
1              1
```

```
1               2
1               -0
-1              -1
1               1
-1              -4
-1              -1
1               3
1               1
1               -2
1               -0
```

```
In [79]: #BrownBoost

         import os
         import sys

         from daal.algorithms.brownboost import prediction, training
         from daal.algorithms import classifier
         from daal.data_management import (
             FileDataSource, DataSourceIface, NumericTableIface, HomogenNumericTable, MergedNu
         )

         # utils_folder = os.path.realpath(os.path.abspath(os.path.dirname(os.path.dirname(__f
         # if utils_folder not in sys.path:
         #     sys.path.insert(0, utils_folder)
         from utils import printNumericTables

         DAAL_PREFIX = os.path.join('..', 'data')

         # Input data set parameters
         # trainDatasetFileName = os.path.join(DAAL_PREFIX, 'batch', 'brownboost_train.csv')
         # testDatasetFileName = os.path.join(DAAL_PREFIX, 'batch', 'brownboost_test.csv')
         trainDatasetFileName = "./csv/fam_train_unlabelled.csv"
         testDatasetFileName = "./csv/fam_test_unlabeleed.csv"
         nFeatures = 24

         trainingResult = None
         predictionResult = None
         groundTruthLabels = None


         def trainModel():
             global trainingResult

             # Initialize FileDataSource<CSVFeatureManager> to retrieve the input data from a
             trainDataSource = FileDataSource(
```

```
        trainDatasetFileName,
        DataSourceIface.notAllocateNumericTable,
        DataSourceIface.doDictionaryFromContext
    )
    # Create Numeric Tables for training data and labels
    trainData = HomogenNumericTable(nFeatures, 0, NumericTableIface.doNotAllocate)
    trainGroundTruth = HomogenNumericTable(1, 0, NumericTableIface.doNotAllocate)
    mergedData = MergedNumericTable(trainData, trainGroundTruth)

    # Retrieve the data from the input file
    trainDataSource.loadDataBlock(mergedData)

    # Create an algorithm object to train the BrownBoost model
    algorithm = training.Batch()

    # Pass the training data set and dependent values to the algorithm
    algorithm.input.set(classifier.training.data, trainData)
    algorithm.input.set(classifier.training.labels, trainGroundTruth)

    # Train the BrownBoost model and retrieve the results of the training algorithm
    trainingResult = algorithm.compute()


def testModel():
    global groundTruthLabels, predictionResult

    # Initialize FileDataSource<CSVFeatureManager> to retrieve the test data from a .
    testDataSource = FileDataSource(
        testDatasetFileName,
        DataSourceIface.notAllocateNumericTable,
        DataSourceIface.doDictionaryFromContext
    )

    # Create Numeric Tables for testing data and labels
    testData = HomogenNumericTable(nFeatures, 0, NumericTableIface.doNotAllocate)
    groundTruthLabels = HomogenNumericTable(1, 0, NumericTableIface.doNotAllocate)
    mergedData = MergedNumericTable(testData, groundTruthLabels)

    # Retrieve the data from input file
    testDataSource.loadDataBlock(mergedData)

    # Create algorithm objects for BrownBoost prediction with the default method
    algorithm = prediction.Batch()

    # Pass the testing data set and trained model to the algorithm
    algorithm.input.setTable(classifier.prediction.data,  testData)
    algorithm.input.setModel(classifier.prediction.model, trainingResult.get(classifie
```

```
    # Compute prediction results and retrieve algorithm results
    # (Result class from classifier.prediction)
    predictionResult = algorithm.compute()

def testModelQuality():
    global predictedLabels, qualityMetricSetResult, groundTruthLabels

    # Retrieve predicted labels
    predictedLabels = predictionResult.get(classifier.prediction.prediction)

    # Create a quality metric set object to compute quality metrics of the SVM algori
    qualityMetricSet = svm.quality_metric_set.Batch()

    input = qualityMetricSet.getInputDataCollection().getInput(svm.quality_metric_set

    input.set(binary_confusion_matrix.predictedLabels,   predictedLabels)
    input.set(binary_confusion_matrix.groundTruthLabels, groundTruthLabels)

    # Compute quality metrics and get the quality metrics
    # returns ResultCollection class from svm.quality_metric_set
    qualityMetricSetResult = qualityMetricSet.compute()


def printResults():

    # Print the classification results
    printNumericTables(
        groundTruthLabels, predictedLabels,
        "Ground truth", "Classification results",
        "ADABOOST (first 50 observations):", 50, interval=15, flt64=False
    )

    # Print the quality metrics
    qualityMetricResult = qualityMetricSetResult.getResult(svm.quality_metric_set.con
    printNumericTable(qualityMetricResult.get(binary_confusion_matrix.confusionMatrix

    block = BlockDescriptor()
    qualityMetricsTable = qualityMetricResult.get(binary_confusion_matrix.binaryMetri
    qualityMetricsTable.getBlockOfRows(0, 1, readOnly, block)
    qualityMetricsData = block.getArray().flatten()
    print("Accuracy:      {0:.3f}".format(qualityMetricsData[binary_confusion_matrix.a
    print("Precision:     {0:.3f}".format(qualityMetricsData[binary_confusion_matrix.p
    print("Recall:        {0:.3f}".format(qualityMetricsData[binary_confusion_matrix.r
    print("F-score:       {0:.3f}".format(qualityMetricsData[binary_confusion_matrix.f
    print("Specificity:   {0:.3f}".format(qualityMetricsData[binary_confusion_matrix.s
    print("AUC:           {0:.3f}".format(qualityMetricsData[binary_confusion_matrix.A
    qualityMetricsTable.releaseBlockOfRows(block)
```

```python
if __name__ == "__main__":
    trainModel()
    testModel()
    testModelQuality()
    printResults()
```

```
ADABOOST (first 50 observations):
Ground truth   Classification results
1              0
1              0
1              1
1              1
1              1
1              1
1              1
1              0
-1             -1
1              1
1              1
1              1
1              0
1              1
-1             -1
1              1
1              1
1              1
1              1
1              1
-1             -1
1              1
1              1
-1             0
-1             0
1              1
1              1
-1             -1
1              1
-1             -1
-1             -1
1              1
1              1
1              1
-1             -1
-1             -1
1              1
-1             -1
1              1
-1             -1
```

```
1           1
-1          -1
1           1
1           1
1           1
1           1
1           1
1           1
1           1
1           1

Confusion matrix:
296.000   5.000
9.000     54.000

Accuracy:      0.962
Precision:     0.970
Recall:        0.983
F-score:       0.977
Specificity:   0.857
AUC:           0.920
```

```
In [81]: # EXAMPLES_DIR='/home/abhijit/anaconda3/envs/idp/share/pydaal_examples/examples'
         import os
         import sys
         sys.path.append('..')
         from GridSearch import GridSearch
         from  daal.algorithms.svm import training, prediction
         import   daal.algorithms.svm as svm
         from daal.data_management import (
           FileDataSource, DataSourceIface, HomogenNumericTable, MergedNumericTable, NumericTal
         )


         #trainDatasetFileName = os.path.join(DATA_PREFIX,'svm_two_class_train_dense.csv')
         trainDatasetFileName = "./csv/fam_train_unlabelled.csv"
         nFeatures = 24

         # Initialize FileDataSource<CSVFeatureManager> to retrieve the input data from a .csv

         trainDataSource = FileDataSource(
                 trainDatasetFileName,
                 DataSourceIface.notAllocateNumericTable,
                 DataSourceIface.doDictionaryFromContext
             )
         # Create Numeric Tables for training data and labels
         trainData = HomogenNumericTable(nFeatures, 0, NumericTableIface.doNotAllocate)
```

```python
trainGroundTruth = HomogenNumericTable(1, 0, NumericTableIface.doNotAllocate)
mergedData = MergedNumericTable(trainData, trainGroundTruth)

# Retrieve the data from the input file
trainDataSource.loadDataBlock(mergedData)

#default keyword arguments
'''
GridSearch(<args>, tuned_parameters = None, score=None,
                        best_score_criteria='high',
                        create_best_training_model = False,
                        save_model=False,nClasses=None )
'''
#create a dictionary of hyperparameter values in a list
svm_params = [{'C':[0.5,1],
                                'accuracyThreshold':[0.01,0.001],
                                'cacheSize':[600000000],
                                'tau':[1.0e-6,1.0e-5],
                                'maxIterations':[100,10],
                                'doShrinking':[True, False]}]
#Create GridSearch object
clf = GridSearch(svm,training,prediction,
                                tuned_parameters = svm_params,score=None,
                                best_score_criteria='high',
                                create_best_training_model=True,
                                save_model=True,nClasses=None)
#Train on all combinations of hyperparameters
result = clf.train(trainData,trainGroundTruth)
#view all the parameters and scores in best to worst order
result.viewAllResults()
#view the best parameters with score
print(result.bestResult())
```

```
Data successfully serialized and saved as trainRes-2018-08-05_20-09-54 and trainRes-2018-08-05
{'C': 0.5, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations':
{'C': 0.5, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations':
{'C': 0.5, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations':
{'C': 0.5, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations':
{'C': 0.5, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 10
{'C': 0.5, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 10
{'C': 0.5, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 10
{'C': 0.5, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 10
{'C': 1, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 10
{'C': 1, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 10
{'C': 1, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 10
{'C': 1, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 10
{'C': 1, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 10,
{'C': 1, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 10,
```

```
{'C': 1, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 10,
{'C': 1, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 10,
{'C': 0.5, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 1
{'C': 0.5, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 1
{'C': 0.5, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 1
{'C': 0.5, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 1
{'C': 0.5, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 10
{'C': 0.5, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 10
{'C': 0.5, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 10
{'C': 0.5, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 10
{'C': 1, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 100
{'C': 1, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 100
{'C': 1, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 100
{'C': 1, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 100
{'C': 1, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 100
{'C': 1, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-06, 'maxIterations': 100
{'C': 1, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 100
{'C': 1, 'accuracyThreshold': 0.01, 'cacheSize': 600000000, 'tau': 1e-05, 'maxIterations': 100
{'Best Parmeters': ["{'C': 0.5, 'accuracyThreshold': 0.001, 'cacheSize': 600000000, 'tau': 1e-0
```

In [82]:
```
import sys
sys.path.append('..')
import os
import daal.algorithms.adaboost as adaB
from daal.algorithms.adaboost import prediction, training
from GridSearch import GridSearch
from daal.data_management import (
        FileDataSource, DataSourceIface, HomogenNumericTable, MergedNumericTable, Nume
)


DATA_PREFIX = os.path.join(EXAMPLES_DIR,'data','batch')
#trainDatasetFileName = os.path.join(DATA_PREFIX, 'adaboost_train.csv')
trainDatasetFileName = "./csv/fam_train_unlabelled.csv"
nFeatures = 20

# Initialize FileDataSource<CSVFeatureManager> to retrieve the input data from a .csv
trainDataSource = FileDataSource(
        trainDatasetFileName, DataSourceIface.notAllocateNumericTable,
        DataSourceIface.doDictionaryFromContext
)

# Create Numeric Tables for training data and labels
trainData = HomogenNumericTable(nFeatures, 0, NumericTableIface.doNotAllocate)
trainGroundTruth = HomogenNumericTable(1, 0, NumericTableIface.doNotAllocate)
mergedData = MergedNumericTable(trainData, trainGroundTruth)
```

```
    # Retrieve the data from the input file
    trainDataSource.loadDataBlock(mergedData)

    #default keyword arguments
    '''
    GridSearch(<args>, tuned_parameters = None, score=None,
                        best_score_criteria='high',
                        create_best_training_model = False,
                        save_model=False,nClasses=None )
    '''
    #create a dictionary of hyperparameter values in a list
    adaB_params = [{'accuracyThreshold': [0.99,0.1],
                            'maxIterations' :[1,5]}]
    #Create GridSearch object
    clf = GridSearch(adaB,training,prediction,
                            tuned_parameters = adaB_params,score=None,
                            best_score_criteria='high',
                            create_best_training_model=True,
                            save_model=True,nClasses=5)
    #Train on all combinations of hyperparameters
    result = clf.train(trainData,trainGroundTruth)
    #view all the parameters and scores in best to worst order
    result.viewAllResults()
    #view the best parameters with score
    print(result.bestResult())
```

```
Data successfully serialized and saved as trainRes-2018-08-05_20-10-58 and trainRes-2018-08-05
{'accuracyThreshold': 0.1, 'maxIterations': 1}: 1.0
{'accuracyThreshold': 0.1, 'maxIterations': 5}: 1.0
{'accuracyThreshold': 0.99, 'maxIterations': 1}: 1.0
{'accuracyThreshold': 0.99, 'maxIterations': 5}: 1.0
{'Best Parmeters': ["{'accuracyThreshold': 0.1, 'maxIterations': 1}", "{'accuracyThreshold': 0
```

In [83]:
```
result.bestParams
```

Out[83]:
```
["{'accuracyThreshold': 0.1, 'maxIterations': 1}",
 "{'accuracyThreshold': 0.1, 'maxIterations': 5}",
 "{'accuracyThreshold': 0.99, 'maxIterations': 1}",
 "{'accuracyThreshold': 0.99, 'maxIterations': 5}"]
```