

In [1]:

```
1 import daal4py as d4p
2 import os,sys,time
3 import numpy as np
4 from sklearn.metrics import accuracy_score
5 from sklearn.metrics import f1_score
6 from sklearn.metrics import recall_score
7 from sklearn.metrics import precision_score
8 from sklearn.metrics import confusion_matrix
9 from sklearn.metrics import f1_score
10 from sklearn.metrics import mean_absolute_error
11 from sklearn.metrics import mean_squared_error
12
13 from sklearn import datasets
14 from sklearn.model_selection import train_test_split
```

In [2]:

```

1  #KNN
2
3  # let's try to use pandas' fast csv reader
4  try:
5      import pandas
6      read_csv = lambda f, c, t=np.float64: pandas.read_csv(f, usecols=c, delimiter=',')
7  except:
8      # fall back to numpy loadtxt
9      read_csv = lambda f, c, t=np.float64: np.loadtxt(f, usecols=c, delimiter=',')
10
11
12 def main(readcsv=read_csv, method='defaultDense'):
13     start = time.clock() # 开始计时
14
15     # Input data set parameters
16     train_file = os.path.join('data', 'k_nearest_neighbors_train.csv')
17     predict_file = os.path.join('data', 'k_nearest_neighbors_test.csv')
18
19     # Read data. Let's use 5 features per observation
20     nFeatures = 5
21     nClasses = 5
22     train_data = readcsv(train_file, range(nFeatures))
23     train_labels = readcsv(train_file, range(nFeatures, nFeatures+1))
24
25     # Create an algorithm object and call compute
26     train_algo = d4p.kdtree_knn_classification_training(nClasses=nClasses)
27     # 'weights' is optional argument, let's use equal weights
28     # in this case results must be the same as without weights
29     weights = np.ones((train_data.shape[0], 1))
30     train_result = train_algo.compute(train_data, train_labels, weights)
31
32     # Now let's do some prediction
33     predict_data = readcsv(predict_file, range(nFeatures))
34     predict_labels = readcsv(predict_file, range(nFeatures, nFeatures+1))
35
36     # Create an algorithm object and call compute
37     predict_algo = d4p.kdtree_knn_classification_prediction()
38     predict_result = predict_algo.compute(predict_data, train_result.model)
39
40     # We expect less than 170 mispredicted values
41     assert np.count_nonzero(predict_labels != predict_result.prediction) < 170
42
43     end=time.clock() #结束计时
44     print("time", end-start)
45
46     return (train_result, predict_result, predict_labels)
47
48
49 if __name__ == "__main__":
50     (train_result, predict_result, predict_labels) = main()
51     # print(predict_result.prediction.shape)
52     # print(predict_labels.shape)
53     print("accuracy_score", accuracy_score(predict_labels, predict_result.prediction))
54     print("precision_score", precision_score(predict_labels, predict_result.prediction))
55     print("recall_score", recall_score(predict_labels, predict_result.prediction))
56     print("f1_score", f1_score(predict_labels, predict_result.prediction, average='micro'))
57     print("confusion_matrix\n", confusion_matrix(predict_labels, predict_result.prediction))

```

/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher

```
er.py:13: DeprecationWarning: time.clock has been deprecated in Python 3.3 and will be removed from Python 3.8: use time.perf_counter or time.process_time instead
```

```
del sys.path[0]
```

```
time 0.25040500000000002
```

```
accuracy_score 0.95775
```

```
precision_score 0.9571618356058407
```

```
recall_score 0.9574422555601799
```

```
f1_score 0.9572627831746068
```

```
confusion_matrix
```

```
[[695  0  0  0  57]
```

```
[  0 823  14  8  0]
```

```
[  0  8 810  2  0]
```

```
[  0  7  1 760  0]
```

```
[ 72  0  0  0 743]]
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher.py:43: DeprecationWarning: time.clock has been deprecated in Python 3.3 and will be removed from Python 3.8: use time.perf_counter or time.process_time instead
```

In [3]:

```

1  #SVM
2
3  # let's try to use pandas' fast csv reader
4  try:
5      import pandas
6      read_csv = lambda f, c, t=np.float64: pandas.read_csv(f, usecols=c, delimiter=',')
7  except:
8      # fall back to numpy loadtxt
9      read_csv = lambda f, c, t=np.float64: np.loadtxt(f, usecols=c, delimiter=',')
10
11
12 def main(readcsv=read_csv, method='defaultDense'):
13     start = time.clock()
14     # input data file
15     infile = "./data/svm_two_class_train_dense.csv"
16     testfile = "./data/svm_two_class_test_dense.csv"
17
18     # Configure a SVM object to use rbf kernel (and adjusting cachesize)
19     kern = d4p.kernel_function_linear(method=method) # need an object that lives
20     train_algo = d4p.svm_training(doShrinking=True, kernel=kern, cacheSize=60000)
21
22     # Read data. Let's use features per observation
23     data = readcsv(infile, range(20))
24     labels = readcsv(infile, range(20,21))
25     train_result = train_algo.compute(data, labels)
26
27     # Now let's do some prediction
28     predict_algo = d4p.svm_prediction(kernel=kern)
29     # read test data (with same #features)
30     pdata = readcsv(testfile, range(20))
31     plabels = readcsv(testfile, range(20,21))
32     # now predict using the model from the training above
33     predict_result = predict_algo.compute(pdata, train_result.model)
34
35     # Prediction result provides prediction
36     assert(predict_result.prediction.shape == (pdata.shape[0], 1))
37
38     end=time.clock() #结束计时
39     print("time", end-start)
40
41     return (predict_result, plabels)
42
43
44 if __name__ == "__main__":
45     (predict_result, predict_labels) = main()
46     #predict_labels = np.squeeze(predict_labels)
47     # print(predict_labels.shape)
48     # print(predict_result.prediction.shape)
49     # print(predict_result.prediction)
50     # 官方的包写错了, 没有进行二值化, 这里进行改正
51     for i in range(len(predict_result.prediction)):
52         if predict_result.prediction[i][0]>=0:
53             predict_result.prediction[i][0]=1
54         else:
55             predict_result.prediction[i][0]=-1
56     # print(predict_result.prediction)
57     print("accuracy_score", accuracy_score(predict_labels, predict_result.prediction))
58     print("precision_score", precision_score(predict_labels, predict_result.prediction))
59     print("recall_score", recall_score(predict_labels, predict_result.prediction))

```

```
60     print("f1_score", f1_score(predict_labels, predict_result.prediction, average="weighted"))
61     print("confusion_matrix\n", confusion_matrix(predict_labels, predict_result.prediction))
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher.py:13: DeprecationWarning: time.clock has been deprecated in Python 3.3 and will be removed from Python 3.8: use time.perf_counter or time.process_time instead
  del sys.path[0]
```

```
time 0.5032990000000002
accuracy_score 0.971
precision_score 0.971037103710371
recall_score 0.9709918839675359
f1_score 0.9709989559624146
confusion_matrix
[[977  25]
 [ 33 965]]
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher.py:38: DeprecationWarning: time.clock has been deprecated in Python 3.3 and will be removed from Python 3.8: use time.perf_counter or time.process_time instead
```

In [4]:

```

1  #朴素贝叶斯
2  # let's try to use pandas' fast csv reader
3  try:
4      import pandas
5      read_csv = lambda f, c, t=np.float64: pandas.read_csv(f, usecols=c, delimiter=','
6  except:
7      # fall back to numpy loadtxt
8      read_csv = lambda f, c, t=np.float64: np.loadtxt(f, usecols=c, delimiter=',',
9
10
11 def main(readcsv=read_csv, method='defaultDense'):
12     start = time.clock()
13
14     # input data file
15     infile = "./data/naivebayes_train_dense.csv"
16     testfile = "./data/naivebayes_test_dense.csv"
17
18     # Configure a training object (20 classes)
19     talgo = d4p.multinomial_naive_bayes_training(20, method=method)
20
21     # Read data. Let's use 20 features per observation
22     data = readcsv(infile, range(20))
23     labels = readcsv(infile, range(20,21))
24     tresult = talgo.compute(data, labels)
25
26     # Now let's do some prediction
27     palgo = d4p.multinomial_naive_bayes_prediction(20, method=method)
28     # read test data (with same #features)
29     pdata = readcsv(testfile, range(20))
30     plabels = readcsv(testfile, range(20,21))
31     # now predict using the model from the training above
32     presult = palgo.compute(pdata, tresult.model)
33
34     # Prediction result provides prediction
35     assert(presult.prediction.shape == (pdata.shape[0], 1))
36
37     end=time.clock() #结束计时
38     print("time", end-start)
39
40     return (presult, plabels)
41
42
43 if __name__ == "__main__":
44     (predict_result, predict_labels) = main()
45     print("accuracy_score", accuracy_score(predict_labels, predict_result.predic
46     print("precision_score", precision_score(predict_labels, predict_result.prec
47     print("recall_score", recall_score(predict_labels, predict_result.prediction
48     print("f1_score", f1_score(predict_labels, predict_result.prediction, averag
49     print("confusion_matrix\n", confusion_matrix(predict_labels, predict_result.

```

```

/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher
r.py:12: DeprecationWarning: time.clock has been deprecated in Python
3.3 and will be removed from Python 3.8: use time.perf_counter or tim
e.process_time instead
    if sys.path[0] == '':

```

```

time 0.195393000000000015
accuracy_score 1.0
precision_score 1.0

```

recall_score 1.0

f1_score 1.0

confusion_matrix

```

[[ 83  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
  0  0]
[  0 90  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0
  0  0]
[  0  0 104  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0
  0  0]
[  0  0  0 91  0  0  0  0  0  0  0  0  0  0  0  0  0
0
  0  0]
[  0  0  0  0 95  0  0  0  0  0  0  0  0  0  0  0  0
0
  0  0]
[  0  0  0  0  0 93  0  0  0  0  0  0  0  0  0  0  0
0
  0  0]
[  0  0  0  0  0  0 97  0  0  0  0  0  0  0  0  0  0
0
  0  0]
[  0  0  0  0  0  0  0 106  0  0  0  0  0  0  0  0  0
0
  0  0]
[  0  0  0  0  0  0  0  0 92  0  0  0  0  0  0  0  0
0
  0  0]
[  0  0  0  0  0  0  0  0  0 105  0  0  0  0  0  0  0
0
  0  0]
[  0  0  0  0  0  0  0  0  0  0 107  0  0  0  0  0  0
0
  0  0]
[  0  0  0  0  0  0  0  0  0  0  0 98  0  0  0  0  0
0
  0  0]
[  0  0  0  0  0  0  0  0  0  0  0  0 115  0  0  0  0
0
  0  0]
[  0  0  0  0  0  0  0  0  0  0  0  0  0 103  0  0  0
0
  0  0]
[  0  0  0  0  0  0  0  0  0  0  0  0  0  0 109  0  0
0
  0  0]
[  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 109  0
0
  0  0]
[  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 106
0
  0  0]
[  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
97
  0  0]
[  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0
  0  0]
[  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
101
  0]
[  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

```

0

^ 0011

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher.py:37: DeprecationWarning: time.clock has been deprecated in Python 3.3 and will be removed from Python 3.8: use time.perf_counter or time.process_time instead
```


In [5]:

```

1  # adaboost
2
3  # let's try to use pandas' fast csv reader
4  try:
5      import pandas
6      read_csv = lambda f, c, t=np.float64: pandas.read_csv(f, usecols=c, delimiter=',')
7  except:
8      # fall back to numpy loadtxt
9      read_csv = lambda f, c, t=np.float64: np.loadtxt(f, usecols=c, delimiter=',')
10
11
12 def main(readcsv=read_csv, method='defaultDense'):
13     start = time.clock()
14
15     infile = "./data/adaboost_train.csv"
16     testfile = "./data/adaboost_test.csv"
17     nClasses = 2
18
19     # Configure a adaboost training object
20     train_algo = d4p.adaboost_training(nClasses=nClasses)
21
22     # Read data. Let's have 20 independent, and 1 dependent variable (for each class)
23     indep_data = readcsv(infile, range(20))
24     dep_data = readcsv(infile, range(20,21))
25     # Now train/compute, the result provides the model for prediction
26     train_result = train_algo.compute(indep_data, dep_data)
27
28     # Now let's do some prediction
29     predict_algo = d4p.adaboost_prediction(nClasses=nClasses)
30     # read test data (with same #features)
31     pdata = readcsv(testfile, range(20))
32     # now predict using the model from the training above
33     predict_result = predict_algo.compute(pdata, train_result.model)
34
35     # The prediction result provides prediction
36     assert predict_result.prediction.shape == (pdata.shape[0], dep_data.shape[1])
37     predict_labels = np.loadtxt(testfile, usecols=range(20,21), delimiter=',', dtype=int)
38     assert np.allclose(predict_result.prediction, predict_labels)
39
40     end=time.clock() #结束计时
41     print("time", end-start)
42
43     return (train_result, predict_result, predict_labels)
44
45
46 if __name__ == "__main__":
47     (train_result, predict_result, predict_labels) = main()
48     print("accuracy_score", accuracy_score(predict_labels, predict_result.prediction))
49     print("precision_score", precision_score(predict_labels, predict_result.prediction))
50     print("recall_score", recall_score(predict_labels, predict_result.prediction))
51     print("f1_score", f1_score(predict_labels, predict_result.prediction, average='micro'))
52     print("confusion_matrix\n", confusion_matrix(predict_labels, predict_result.prediction))

```

```

/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher
r.py:13: DeprecationWarning: time.clock has been deprecated in Python
3.3 and will be removed from Python 3.8: use time.perf_counter or tim
e.process_time instead
del sys.path[0]

```

```
time 0.23287499999999994
accuracy_score 1.0
precision_score 1.0
recall_score 1.0
f1_score 1.0
confusion_matrix
[[1599    0]
 [    0  401]]
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher.py:40: DeprecationWarning: time.clock has been deprecated in Python 3.3 and will be removed from Python 3.8: use time.perf_counter or time.process_time instead
```

In [6]:

```

1  # 随机森林RandomFroest
2
3  # let's try to use pandas' fast csv reader
4  try:
5      import pandas
6      read_csv = lambda f, c, t=np.float64: pandas.read_csv(f, usecols=c, delimiter=',')
7  except:
8      # fall back to numpy loadtxt
9      read_csv = lambda f, c, t=np.float64: np.loadtxt(f, usecols=c, delimiter=',')
10
11 # Get Intel(R) Data Analytics Acceleration Library (Intel(R) DAAL) version
12 from daal4py import __daal_link_version__ as dv
13 daal_version = tuple(map(int, (dv[0:4], dv[4:8])))
14
15 def main(readcsv=read_csv, method='defaultDense'):
16     start = time.clock()
17     # input data file
18     infile = "./data/df_classification_train.csv"
19     testfile = "./data/df_classification_test.csv"
20
21     # Configure a training object (5 classes)
22     train_algo = d4p.decision_forest_classification_training(5,
23                                                             nTrees=10,
24                                                             minObservationsInLe
25                                                             featuresPerNode=3,
26                                                             engine = d4p.engine
27                                                             varImportance='MDI'
28                                                             bootstrap=True,
29                                                             resultsToCompute='c
30
31     # Read data. Let's use 3 features per observation
32     data = readcsv(infile, range(3), t=np.float32)
33     labels = readcsv(infile, range(3,4), t=np.float32)
34     train_result = train_algo.compute(data, labels)
35     # Traing result provides (depending on parameters) model, outOfBagError, ou
36
37     # Now let's do some prediction
38     predict_algo = d4p.decision_forest_classification_prediction(nClasses=5)
39     # if daal_version < (2020,1):
40     #     predict_algo = d4p.decision_forest_classification_prediction(nClasses=
41     # else:
42     #     predict_algo = d4p.decision_forest_classification_prediction(nClasses=
43     #     resultsToEvaluate="computeClassLabels/computeClassProbabilities",
44     # read test data (with same #features)
45     pdata = readcsv(testfile, range(3), t=np.float32)
46     plabels = readcsv(testfile, range(3,4), t=np.float32)
47     # now predict using the model from the training above
48     predict_result = predict_algo.compute(pdata, train_result.model)
49
50     # Prediction result provides prediction
51     assert(predict_result.prediction.shape == (pdata.shape[0], 1))
52
53     end=time.clock() #结束计时
54     print("time", end-start)
55     return (train_result, predict_result, plabels)
56
57
58 if __name__ == "__main__":
59     (train_result, predict_result, predict_labels) = main()

```

```
60     print("accuracy_score", accuracy_score(predict_labels, predict_result.predic
61     print("precision_score", precision_score(predict_labels, predict_result.prec
62     print("recall_score", recall_score(predict_labels, predict_result.prediction
63     print("f1_score", f1_score(predict_labels, predict_result.prediction, averag
64     print("confusion_matrix\n", confusion_matrix(predict_labels, predict_result.
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher
r.py:16: DeprecationWarning: time.clock has been deprecated in Python
3.3 and will be removed from Python 3.8: use time.perf_counter or tim
e.process_time instead
    app.launch_new_instance()
```

```
time 2.4232190000000005
accuracy_score 0.972
precision_score 0.9724331546028709
recall_score 0.9724483374081128
f1_score 0.9723101652373585
confusion_matrix
[[161   2   0   0   0]
 [  2 155   3   0   0]
 [  0   8 310   2   0]
 [  0   0   5 193   0]
 [  0   0   0   6 153]]
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher
r.py:53: DeprecationWarning: time.clock has been deprecated in Python
3.3 and will be removed from Python 3.8: use time.perf_counter or tim
e.process_time instead
```

In [7]:

```

1  # EM算法
2
3  # let's try to use pandas' fast csv reader
4  try:
5      import pandas
6      read_csv = lambda f, c=None, t=np.float64: pandas.read_csv(f, usecols=c, del
7  except:
8      # fall back to numpy loadtxt
9      read_csv = lambda f, c=None, t=np.float64: np.loadtxt(f, usecols=c, delimit
10
11
12  def main(readcsv=read_csv, method='defaultDense'):
13      start = time.clock()
14      nComponents = 2
15      infile = "./data/em_gmm.csv"
16      # We load the data
17      data = readcsv(infile)
18
19      # configure a em_gmm init object
20      algo1 = d4p.em_gmm_init(nComponents)
21      # and compute initial model
22      result1 = algo1.compute(data)
23
24      # configure a em_gmm object
25      algo2 = d4p.em_gmm(nComponents)
26
27      # and compute em_gmm using initial weights and means
28      result2 = algo2.compute(data, result1.weights, result1.means, result1.covari
29
30      end=time.clock() #结束计时
31      print("time", end-start)
32
33      # implicit als prediction result objects provide covariances, goalFunction,
34      return result2
35
36
37  if __name__ == "__main__":
38      res = main()
39      # daal库的api不支持预测
40      print("Weights:\n", res.weights)
41      print("Means:\n", res.means)
42      for c in res.covariances:
43          print("Covariance:\n", c)
44      print('All looks good!')
```

time 0.005870999999999996

Weights:

[[0.50004707 0.49995293]]

Means:

```
[[10.16638183  0.04546081 -7.21141    ]
 [ 0.19393028  0.01859487  0.37642873]]
```

Covariance:

```
[[ 7.56786995  1.30583815 -0.06336321]
 [ 1.30583815  0.97932635  0.63631872]
 [-0.06336321  0.63631872  2.27659338]]
```

Covariance:

```
[[ 0.85968907 -0.22851204  0.08127243]
 [-0.22851204  2.2197825  -0.10165961]]
```

```
[ 0.08127243 -0.10165961  2.57295252]]  
All looks good!
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher.py:13: DeprecationWarning: time.clock has been deprecated in Python 3.3 and will be removed from Python 3.8: use time.perf_counter or time.process_time instead  
  del sys.path[0]  
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher.py:30: DeprecationWarning: time.clock has been deprecated in Python 3.3 and will be removed from Python 3.8: use time.perf_counter or time.process_time instead
```

In [8]:

```

1  # 随机森林回归
2
3  # let's try to use pandas' fast csv reader
4  try:
5      import pandas
6      read_csv = lambda f, c, t=np.float64: pandas.read_csv(f, usecols=c, delimiter=',')
7  except:
8      # fall back to numpy loadtxt
9      read_csv = lambda f, c, t=np.float64: np.loadtxt(f, usecols=c, delimiter=',')
10
11
12 def main(readcsv=read_csv, method='defaultDense'):
13     start = time.clock()
14     infile = "./data/df_regression_train.csv"
15     testfile = "./data/df_regression_test.csv"
16
17     # Configure a Linear regression training object
18     train_algo = d4p.decision_forest_regression_training(nTrees=100,
19                                                         varImportance='MDA_Raw',
20                                                         bootstrap=True,
21                                                         engine = d4p.engines_mt,
22                                                         resultsToCompute='compute')
23
24     # Read data. Let's have 13 independent, and 1 dependent variables (for each
25     indep_data = readcsv(infile, range(13), t=np.float32)
26     dep_data = readcsv(infile, range(13,14), t=np.float32)
27     # Now train/compute, the result provides the model for prediction
28     train_result = train_algo.compute(indep_data, dep_data)
29     # Training result provides (depending on parameters) model, outOfBagError, outOfSampleError, etc.
30
31     # Now let's do some prediction
32     predict_algo = d4p.decision_forest_regression_prediction()
33     # read test data (with same #features)
34     pdata = readcsv(testfile, range(13), t=np.float32)
35     ptdata = readcsv(testfile, range(13,14), t=np.float32)
36     # now predict using the model from the training above
37     predict_result = predict_algo.compute(pdata, train_result.model)
38
39     # The prediction result provides prediction
40     assert predict_result.prediction.shape == (pdata.shape[0], dep_data.shape[1])
41
42     end=time.clock() #结束计时
43     print("time", end-start)
44
45     return (train_result, predict_result, ptdata)
46
47
48 if __name__ == "__main__":
49     from daal4py import __daal_link_version__ as dv
50     daal_version = tuple(map(int, (dv[0:4], dv[4:8])))
51     if daal_version < (2019, 1):
52         print("Need Intel(R) DAAL 2019.1 or later")
53     else:
54         (train_result, predict_result, ptdata) = main()
55         print("MAE", mean_absolute_error(ptdata, predict_result.prediction))
56         print("MSE", mean_squared_error(ptdata, predict_result.prediction))
57
58

```

```
time 0.16352499999999992
MAE 2.708411087293018
MSE 17.652745413976746
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher.py:13: DeprecationWarning: time.clock has been deprecated in Python 3.3 and will be removed from Python 3.8: use time.perf_counter or time.process_time instead
```

```
del sys.path[0]
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launcher.py:42: DeprecationWarning: time.clock has been deprecated in Python 3.3 and will be removed from Python 3.8: use time.perf_counter or time.process_time instead
```

In []:

```
1
```

In []:

```
1
```