In [1]:

```python
import daal4py as d4p
import os,sys,time
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error

from sklearn import datasets
from sklearn.model_selection import train_test_split
```

In [2]:

```python
#KNN

# let's try to use pandas' fast csv reader
try:
    import pandas
    read_csv = lambda f, c, t=np.float64: pandas.read_csv(f, usecols=c, delimite
except:
    # fall back to numpy loadtxt
    read_csv = lambda f, c, t=np.float64: np.loadtxt(f, usecols=c, delimiter=','


def main(readcsv=read_csv, method='defaultDense'):
    start = time.clock() # 开始计时

    # Input data set parameters
    train_file = os.path.join('data', 'k_nearest_neighbors_train.csv')
    predict_file  = os.path.join('data', 'k_nearest_neighbors_test.csv')

    # Read data. Let's use 5 features per observation
    nFeatures = 5
    nClasses = 5
    train_data   = readcsv(train_file, range(nFeatures))
    train_labels = readcsv(train_file, range(nFeatures, nFeatures+1))

    # Create an algorithm object and call compute
    train_algo = d4p.kdtree_knn_classification_training(nClasses=nClasses)
    # 'weights' is optional argument, let's use equal weights
    # in this case results must be the same as without weights
    weights = np.ones((train_data.shape[0], 1))
    train_result = train_algo.compute(train_data, train_labels, weights)

    # Now let's do some prediction
    predict_data = readcsv(predict_file, range(nFeatures))
    predict_labels = readcsv(predict_file, range(nFeatures, nFeatures+1))

    # Create an algorithm object and call compute
    predict_algo = d4p.kdtree_knn_classification_prediction()
    predict_result = predict_algo.compute(predict_data, train_result.model)

    # We expect less than 170 mispredicted values
    assert np.count_nonzero(predict_labels != predict_result.prediction) < 170

    end=time.clock() #结束计时
    print("time", end-start)

    return (train_result, predict_result, predict_labels)


if __name__ == "__main__":
    (train_result, predict_result, predict_labels) = main()
#     print(predict_result.prediction.shape)
#     print(predict_labels.shape)
    print("accuracy_score", accuracy_score(predict_labels, predict_result.predic
    print("precision_score", precision_score(predict_labels, predict_result.prec
    print("recall_score", recall_score(predict_labels, predict_result.predictior
    print("f1_score", f1_score(predict_labels, predict_result.prediction, averag
    print("confusion_matrix\n", confusion_matrix(predict_labels, predict_result.
```

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

```
er.py:13: DeprecationWarning: time.clock has been deprecated in Pytho
n 3.3 and will be removed from Python 3.8: use time.perf_counter or t
ime.process_time instead
  del sys.path[0]

time 0.2504050000000002
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_launche
r.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 tim
e.process_time instead
```

In [3]:

```
#SVM

# let's try to use pandas' fast csv reader
try:
    import pandas
    read_csv = lambda f, c, t=np.float64: pandas.read_csv(f, usecols=c, delimite
except:
    # fall back to numpy loadtxt
    read_csv = lambda f, c, t=np.float64: np.loadtxt(f, usecols=c, delimiter=','


def main(readcsv=read_csv, method='defaultDense'):
    start = time.clock()
    # input data file
    infile = "./data/svm_two_class_train_dense.csv"
    testfile = "./data/svm_two_class_test_dense.csv"

    # Configure a SVM object to use rbf kernel (and adjusting cachesize)
    kern = d4p.kernel_function_linear(method=method)  # need an object that live
    train_algo = d4p.svm_training(doShrinking=True, kernel=kern, cacheSize=60000

    # Read data. Let's use features per observation
    data   = readcsv(infile, range(20))
    labels = readcsv(infile, range(20,21))
    train_result = train_algo.compute(data, labels)

    # Now let's do some prediction
    predict_algo = d4p.svm_prediction(kernel=kern)
    # read test data (with same #features)
    pdata = readcsv(testfile, range(20))
    plabels = readcsv(testfile, range(20,21))
    # now predict using the model from the training above
    predict_result = predict_algo.compute(pdata, train_result.model)

    # Prediction result provides prediction
    assert(predict_result.prediction.shape == (pdata.shape[0], 1))

    end=time.clock() #结束计时
    print("time", end-start)

    return (predict_result, plabels)


if __name__ == "__main__":
    (predict_result, predict_labels) = main()
    #predict_labels = np.squeeze(predict_labels)
#     print(predict_labels.shape)
#     print(predict_result.prediction.shape)
#     print(predict_result.prediction)
#     官方的包写错了，没有进行二值化，这里进行改正
    for i in range(len(predict_result.prediction)):
        if predict_result.prediction[i][0]>=0:
            predict_result.prediction[i][0]=1
        else:
            predict_result.prediction[i][0]=-1
#     print(predict_result.prediction)
    print("accuracy_score", accuracy_score(predict_labels, predict_result.predic
    print("precision_score", precision_score(predict_labels, predict_result.prec
    print("recall_score", recall_score(predict_labels, predict_result.predictior
    print("f1_score", f1_score(predict_labels, predict_result.prediction, averag
    print("confusion_matrix\n", confusion_matrix(predict_labels, predict_result.
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launche
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 0.5032990000000002
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_launche
r.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 tim
e.process_time instead
```

In [4]:

```
#朴素贝叶斯
# let's try to use pandas' fast csv reader
try:
    import pandas
    read_csv = lambda f, c, t=np.float64: pandas.read_csv(f, usecols=c, delimite
except:
    # fall back to numpy loadtxt
    read_csv = lambda f, c, t=np.float64: np.loadtxt(f, usecols=c, delimiter=','


def main(readcsv=read_csv, method='defaultDense'):
    start = time.clock()

    # input data file
    infile = "./data/naivebayes_train_dense.csv"
    testfile = "./data/naivebayes_test_dense.csv"

    # Configure a training object (20 classes)
    talgo = d4p.multinomial_naive_bayes_training(20, method=method)

    # Read data. Let's use 20 features per observation
    data   = readcsv(infile, range(20))
    labels = readcsv(infile, range(20,21))
    tresult = talgo.compute(data, labels)

    # Now let's do some prediction
    palgo = d4p.multinomial_naive_bayes_prediction(20, method=method)
    # read test data (with same #features)
    pdata = readcsv(testfile, range(20))
    plabels = readcsv(testfile, range(20,21))
    # now predict using the model from the training above
    presult = palgo.compute(pdata, tresult.model)

    # Prediction result provides prediction
    assert(presult.prediction.shape == (pdata.shape[0], 1))

    end=time.clock() #结束计时
    print("time", end-start)

    return (presult, plabels)


if __name__ == "__main__":
    (predict_result, predict_labels) = main()
    print("accuracy_score", accuracy_score(predict_labels, predict_result.predic
    print("precision_score", precision_score(predict_labels, predict_result.prec
    print("recall_score", recall_score(predict_labels, predict_result.predictior
    print("f1_score", f1_score(predict_labels, predict_result.prediction, averag
    print("confusion_matrix\n", confusion_matrix(predict_labels, predict_result.
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launche
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 0.19539300000000015
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  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
  101   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
0
    0.9811
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launche
r.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 tim
e.process_time instead
```

In [5]:

```python
# adaboost

# let's try to use pandas' fast csv reader
try:
    import pandas
    read_csv = lambda f, c, t=np.float64: pandas.read_csv(f, usecols=c, delimite
except:
    # fall back to numpy loadtxt
    read_csv = lambda f, c, t=np.float64: np.loadtxt(f, usecols=c, delimiter=','


def main(readcsv=read_csv, method='defaultDense'):
    start  = time.clock()

    infile = "./data/adaboost_train.csv"
    testfile = "./data/adaboost_test.csv"
    nClasses = 2

    # Configure a adaboost training object
    train_algo = d4p.adaboost_training(nClasses=nClasses)

    # Read data. Let's have 20 independent, and 1 dependent variable (for each o
    indep_data = readcsv(infile, range(20))
    dep_data   = readcsv(infile, range(20,21))
    # Now train/compute, the result provides the model for prediction
    train_result = train_algo.compute(indep_data, dep_data)

    # Now let's do some prediction
    predict_algo = d4p.adaboost_prediction(nClasses=nClasses)
    # read test data (with same #features)
    pdata = readcsv(testfile, range(20))
    # now predict using the model from the training above
    predict_result = predict_algo.compute(pdata, train_result.model)

    # The prediction result provides prediction
    assert predict_result.prediction.shape == (pdata.shape[0], dep_data.shape[1]
    predict_labels = np.loadtxt(testfile, usecols=range(20,21), delimiter=',', r
    assert np.allclose(predict_result.prediction, predict_labels)

    end=time.clock() #结束计时
    print("time", end-start)

    return (train_result, predict_result, predict_labels)


if __name__ == "__main__":
    (train_result, predict_result, predict_labels) = main()
    print("accuracy_score", accuracy_score(predict_labels, predict_result.predic
    print("precision_score", precision_score(predict_labels, predict_result.prec
    print("recall_score", recall_score(predict_labels, predict_result.predictior
    print("f1_score", f1_score(predict_labels, predict_result.prediction, averag
    print("confusion_matrix\n", confusion_matrix(predict_labels, predict_result.
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launche
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_launche
r.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 tim
e.process_time instead
```

In [6]:

```
# 随机森林RandomFroest

# let's try to use pandas' fast csv reader
try:
    import pandas
    read_csv = lambda f, c, t=np.float64: pandas.read_csv(f, usecols=c, delimite
except:
    # fall back to numpy loadtxt
    read_csv = lambda f, c, t=np.float64: np.loadtxt(f, usecols=c, delimiter=','

# Get Intel(R) Data Analytics Acceleration Library (Intel(R) DAAL) version
from daal4py import __daal_link_version__ as dv
daal_version = tuple(map(int, (dv[0:4], dv[4:8])))

def main(readcsv=read_csv, method='defaultDense'):
    start = time.clock()
    # input data file
    infile = "./data/df_classification_train.csv"
    testfile = "./data/df_classification_test.csv"

    # Configure a training object (5 classes)
    train_algo = d4p.decision_forest_classification_training(5,
                                                              nTrees=10,
                                                              minObservationsInLe
                                                              featuresPerNode=3,
                                                              engine = d4p.engine
                                                              varImportance='MDI'
                                                              bootstrap=True,
                                                              resultsToCompute='c

    # Read data. Let's use 3 features per observation
    data   = readcsv(infile, range(3), t=np.float32)
    labels = readcsv(infile, range(3,4), t=np.float32)
    train_result = train_algo.compute(data, labels)
    # Traiing result provides (depending on parameters) model, outOfBagError, ou

    # Now let's do some prediction
    predict_algo = d4p.decision_forest_classification_prediction(nClasses=5)
#     if daal_version < (2020,1):
#         predict_algo = d4p.decision_forest_classification_prediction(nClasses=
#     else:
#         predict_algo = d4p.decision_forest_classification_prediction(nClasses=
#             resultsToEvaluate="computeClassLabels|computeClassProbabilities",
    # read test data (with same #features)
    pdata = readcsv(testfile, range(3), t=np.float32)
    plabels = readcsv(testfile, range(3,4), t=np.float32)
    # now predict using the model from the training above
    predict_result = predict_algo.compute(pdata, train_result.model)

    # Prediction result provides prediction
    assert(predict_result.prediction.shape == (pdata.shape[0], 1))

    end=time.clock() #结束计时
    print("time", end-start)
    return (train_result, predict_result, plabels)


if __name__ == "__main__":
    (train_result, predict_result, predict_labels) = main()
    print("accuracy_score", accuracy_score(predict_labels, predict_result.predic
    print("precision_score", precision_score(predict_labels, predict_result.prec
    print("recall_score", recall_score(predict_labels, predict_result.predictior
    print("f1_score", f1_score(predict_labels, predict_result.prediction, averag
    print("confusion_matrix\n", confusion_matrix(predict_labels, predict_result.
```

```
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launche
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_launche
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]:

```python
# EM算法

# let's try to use pandas' fast csv reader
try:
    import pandas
    read_csv = lambda f, c=None, t=np.float64: pandas.read_csv(f, usecols=c, del
except:
    # fall back to numpy loadtxt
    read_csv = lambda f, c=None, t=np.float64: np.loadtxt(f, usecols=c, delimite


def main(readcsv=read_csv, method='defaultDense'):
    start = time.clock()
    nComponents = 2
    infile = "./data/em_gmm.csv"
    # We load the data
    data = readcsv(infile)

    # configure a em_gmm init object
    algo1 = d4p.em_gmm_init(nComponents)
    # and compute initial model
    result1 = algo1.compute(data)

    # configure a em_gmm object
    algo2 = d4p.em_gmm(nComponents)

    # and compute em_gmm using initial weights and means
    result2 = algo2.compute(data, result1.weights, result1.means, result1.covari

    end=time.clock() #结束计时
    print("time", end-start)

    # implicit als prediction result objects provide covariances, goalFunction,
    return result2


if __name__ == "__main__":
    res = main()
    # daal库的api不支持预测
    print("Weights:\n", res.weights)
    print("Means:\n", res.means)
    for c in res.covariances:
        print("Covariance:\n", c)
    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_launche
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]
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launche
r.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 tim
e.process_time instead
```

In [8]:

```
# 随机森林回归

# let's try to use pandas' fast csv reader
try:
    import pandas
    read_csv = lambda f, c, t=np.float64: pandas.read_csv(f, usecols=c, delimite
except:
    # fall back to numpy loadtxt
    read_csv = lambda f, c, t=np.float64: np.loadtxt(f, usecols=c, delimiter=',',


def main(readcsv=read_csv, method='defaultDense'):
    start = time.clock()
    infile = "./data/df_regression_train.csv"
    testfile = "./data/df_regression_test.csv"

    # Configure a Linear regression training object
    train_algo = d4p.decision_forest_regression_training(nTrees=100,
                                                         varImportance='MDA_Raw'
                                                         bootstrap=True,
                                                         engine = d4p.engines_mt
                                                         resultsToCompute='compu

    # Read data. Let's have 13 independent, and 1 dependent variables (for each
    indep_data = readcsv(infile, range(13), t=np.float32)
    dep_data   = readcsv(infile, range(13,14), t=np.float32)
    # Now train/compute, the result provides the model for prediction
    train_result = train_algo.compute(indep_data, dep_data)
    # Traiing result provides (depending on parameters) model, outOfBagError, ou

    # Now let's do some prediction
    predict_algo = d4p.decision_forest_regression_prediction()
    # read test data (with same #features)
    pdata = readcsv(testfile, range(13), t=np.float32)
    ptdata = readcsv(testfile, range(13,14), t=np.float32)
    # now predict using the model from the training above
    predict_result = predict_algo.compute(pdata, train_result.model)

    # The prediction result provides prediction
    assert predict_result.prediction.shape == (pdata.shape[0], dep_data.shape[1]

    end=time.clock() #结束计时
    print("time", end-start)

    return (train_result, predict_result, ptdata)


if __name__ == "__main__":
    from daal4py import __daal_link_version__ as dv
    daal_version = tuple(map(int, (dv[0:4], dv[4:8])))
    if daal_version < (2019, 1):
        print("Need Intel(R) DAAL 2019.1 or later")
    else:
        (train_result, predict_result, ptdata) = main()
        print("MAE", mean_absolute_error(ptdata, predict_result.prediction))
        print("MSE", mean_squared_error(ptdata, predict_result.prediction))


```

```
time 0.16352499999999992
MAE 2.708411087293018
MSE 17.652745413976746

/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launche
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]
/opt/anaconda3/envs/daal/lib/python3.7/site-packages/ipykernel_launche
r.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 tim
e.process_time instead
```

In [ ]:

In [ ]: