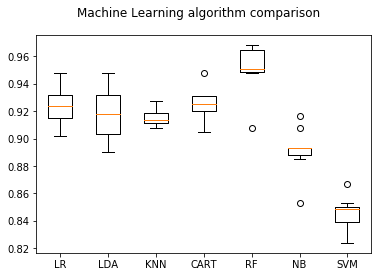
Multitypes machine learning test (corn diseases test):

1. Data set

There are 4 types’ samples: healthy (1162 samples), leaf spot disease (513 samples), and northern leaf blight (985 samples) and rust disease (1192 samples). The data set (4 folds) are copy from \\leaf-disease-data\leaf-disease-plant-village-master\plantvillage\_deeplearning\_paper\_dataset\color (4 sub folds start with Corn\_marize\_: Corn\_maize\_healthy, Corn\_maize\_Cercospora\_leaf\_spot Gray\_leaf\_spot, Corn\_maize\_Northern\_leaf\_Blight, and Corn\_maize\_Common\_rust\_), and saved in \EPQ codes\dataset\Corn. The 4 sample sub folds are renamed as healthy, leaf\_spot, northern\_leaf\_blight and rust, respectively. There is no other treatment to the data files.

1. Method and Codes
   1. featuregen.py: this program pre-defines the feature generation method and data save method. This program then will be import into training and testing programs.
   2. Sampling6.py: this program firstly read all sample file names from input the data set fold (here is: dataset/Corn) to a list and labelled each sample numerically and save the labels to another list. Then uses random sampling method by train\_test\_split function to generate train and test sample file name list and label list. This program then save those data lists in two data file (“output/Corn\_train\_samples6.h5”, and “output/Corn\_test\_samples6.h5). The test sample set size is 0.1 of total samples.
   3. image\_feature\_generation6.py: this program reads train samples based on the sampling result file (“output/Corn\_train\_samples6.h5”) and generates train globe features. The features are saved in an h5f file (“output/Corn\_train\_features6.h5”).
   4. method\_compare6.py: this programs uses 10 folds cross validation method (kfold and cross\_val\_score function) to compare several machine learning methods in the disease detection on the train feature file generated in section 2.3. The features are normalized to 0-1 before validation.
   5. Train\_RF61.py: this program uses train feature data set generated in section 2.3 to train scaler and RF classifier models. The classifier is trained by normalized train globe feature data which read from the h5f file generation in section2.3 (“output/Corn\_train\_features6.h5”). The trained scaler and classifier are saved in two files: “output/Corn\_train\_scaler.h5” and “output/Corn\_train\_model.h5”, respectively.
   6. Test\_RF61.py: this program tests one machine method (RF, trained in section 2.5, “output/Corn\_train\_model.h5”) in disease detection in detail on the test feature data file generated in section 2.3. The test globe feature data is generated from the samples specified in the list in test sample file (“output/Corn\_test\_samples6.h5) using predefined function “feature\_generate” in section 1. Generated test globe features are normalized by the scaler which is trained by train globe feature data in section 2.5(“output/Corn\_train\_scaler.h5”). The results are used to calculate real positive, real negative, false positive and false negative.
   7. Demo\_test\_RF61.PY: this program demonstrates one machine method (RF, using clf function) in disease detection. The scaler and classifier were trained in section 2.5 (“output/Corn\_train\_scaler.h5” and “output/Corn\_train\_model.h5”). The test step is same as section 2.6, but the results for each test samples are print out with the sample pictures. The test samples are selected from previous test sample manually. No statics means for this demonstration.
2. Results
   1. several machine learning method comparisons by 10 fold cross validations:

|  |  |  |
| --- | --- | --- |
| Method | Score | Std |
| LR: , (random\_state=9) | 0.923549 | 0.013147 |
| LDA: | 0.918071 | 0.018438 |
| KNN: | 0.915754 | 0.006143 |
| CART: (random\_state=9) | 0.925265 | 0.010816 |
| RF: (n\_estimators=100, random\_state=9) | 0.952102 | 0.016818 |
| NB: | 0.891224 | 0.015704 |
| SVM: (random\_state=9) | 0.845641 | 0.011083 |



* 1. RandomForestClassifier(n\_estimators=100, random\_state=9)

Classifier scores: 0.9637

Total test sample: 386

Correct predictions are: 372

lassNames ['healthy' 'leaf\_spot' 'Northern\_Leaf\_Blight' 'rust']

Total sample: [ 124 41 94 127]

Detected results:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | healthy (detected) | leaf\_spot (detected) | Northern\_Leaf\_Blight (detected) | Rust (detected) | Total real |
| healthy (real) | 124 | 0 | 0 | 0 | 124 |
| leaf\_spot (real) | 0 | 30 | 11 | 0 | 41 |
| Northern\_Leaf\_Blight (real) | 0 | 3 | 91 | 0 | 94 |
| Rust (real) | 0 | 0 | 0 | 127 | 127 |
| Total detected samples | 124 | 33 | 102 | 127 | 386 |

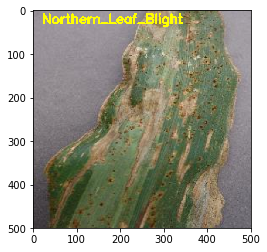
RP/RN/FP/FN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | healthy | leaf\_spot | Northern\_Leaf\_Blight | Rust |
| Real Positive | 1.0 | 0.9091 | 0.8922 | 1.0 |
| Real Negative | 1.0 | 0.9688 | 0.9894 | 1.0 |
| False Positive | 0 | 0.0909 | 0.1078 | 0 |
| False Negative | 0 | 0.0312 | 0.0106 | 0 |

The results shows, the Northern leaf blight and leaf spot is hard to detect than others.

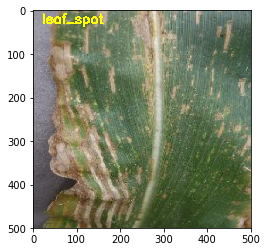
* 1. Demo

From test results, select two error classify sample: GLSp4334.jpg(leaf sport, but was marked as Northern leaf blight), NLB3833.jpg(Northern leaf blight, but was marked as leaf spot) and randomly select one other sample from left spot and northern leaf blight, respectively. Also randomly select 2 samples from healthy and rust data set, respectively.



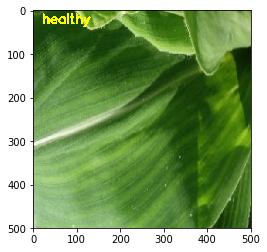
file name: GLSp 4334.JPG

prediction lable: Northern\_Leaf\_Blight



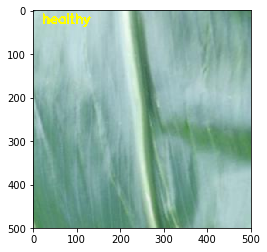
file name: GLSp 4587.JPG

prediction lable: leaf\_spot



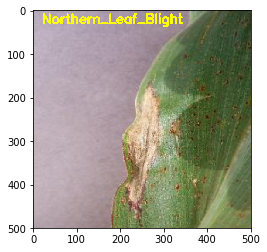
file name: HL 5520.jpg

prediction lable: healthy



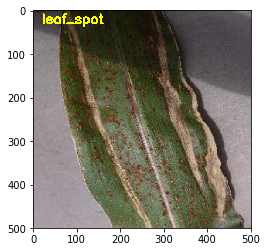
file name: HL 8180.jpg

prediction lable: healthy



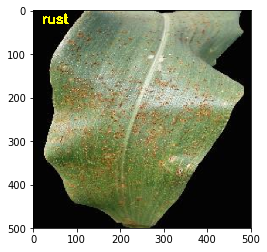
file name: NLB 3617.JPG

prediction lable: Northern\_Leaf\_Blight



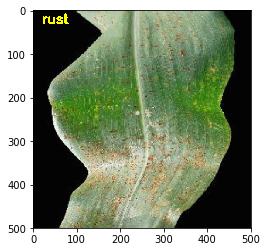
file name: NLB 3833.JPG

prediction lable: leaf\_spot



file name: Rust 1566.JPG

prediction lable: rust



file name: Rust 1567.JPG

prediction lable: rust

1. Raised question
   1. Classifiers: we compared 7 methods, you might be asked why you selected those method to compare and what those method are.
   2. How do you train the classifier? (using clf.fit) and using the classifier? (using clf.prediction)
   3. The pre-processing of the data. Do you know any other data pre-processing method? How to normalize the test data? (using scaler.fit to train the scaler and using scaler. Transform to normalize test data)
   4. Kfold method
   5. Cross validation method
   6. Train-test-splite
   7. H5 file related
   8. Coding relative, how to find resource, debug…
   9. Program design and structure
2. Program lists:
   1. Featuregen.py
   2. Sampling6.py
      1. Output data files: output/Corn\_train\_samples6.h5, output/Corn\_test\_samples6.h5
      2. Data saved in output files:
         1. Samples: sample image file name and path
         2. Labels: sample numerical labels
         3. ClassNames: sample class name
         4. Sample\_Path: sample file original path
      3. Relative program:
         1. image\_feature\_generation6.py
         2. Test\_RF6.py
   3. image\_feature\_generation6.py
      1. Input data files: output/Corn\_train\_samples6.h5
      2. Output data files: output/Corn\_train\_features6.h5
      3. Data saved in output files:
         1. Samples: sample features
         2. Labels: sample numerical labels
         3. ClassNames: sample class name
         4. Sample\_Path: sample file original path
      4. Relative program:
         1. method\_compare6.py
         2. Train\_RF61.py
         3. Demo\_test\_RF61.PY
   4. method\_compare6.py
      1. Input data files: output/Corn\_train\_features6.h5
   5. Train\_RF61.py
      1. Input data files: output/Corn\_train\_features6.h5
      2. Output data files: output/Corn\_train\_scaler.h5 and output/Corn\_train\_model.h5
      3. Relative program:
         1. Test\_RF61.py
         2. Demo\_test\_RF61.PY
   6. Test\_RF61.py
      1. Input data files: output/Corn\_test\_samples6.h5, output/Corn\_train\_scaler.h5 and output/Corn\_train\_model.h5
   7. Demo\_test\_RF61.PY
      1. Input data files: output/Corn\_train\_features6.h5, output/Corn\_train\_scaler.h5 and output/Corn\_train\_model.h5
3. Others

The results are same as version 6

1. more