1 Introduction

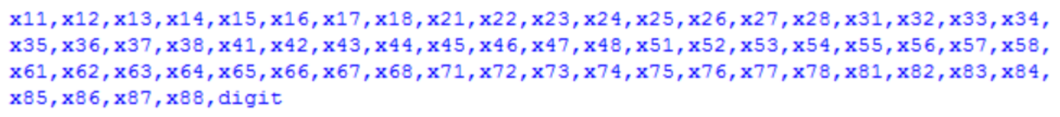
The experiment aims at realizing handwritten digit (from 0 to 9) recognition using specific method.

For each handwritten digit, a 32*32 normalized bitmap is generated to represent the digit. Then, the bitmap is divided into blocks. Each block has a size of 4*4 bit, so there are 8*8 blocks in a bitmap for a digit. The number of pixels (ranging from 0 to 16) are counted in each block. Therefore, the input contains 64 numeric attributes.

2 Visualization

To better visualize the generated images without rotation or mirror, I modified the line 75 of the Python script, switching “row” and “col”. That is “digitImage.setPixel(col,row,px)”.

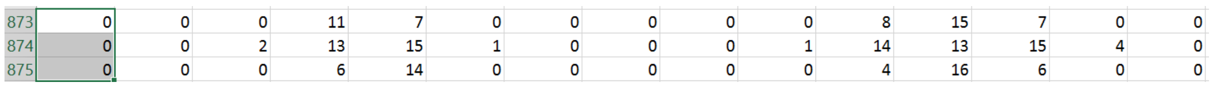
The Python program opens the test file, “optdigits.tes.csv”, as input file. When starting running the program, it firstly prints out the header line of the csv file, as is shown below.



```
x11,x12,x13,x14,x15,x16,x17,x18,x21,x22,x23,x24,x25,x26,x27,x28,x31,x32,x33,x34,
x35,x36,x37,x38,x41,x42,x43,x44,x45,x46,x47,x48,x51,x52,x53,x54,x55,x56,x57,x58,
x61,x62,x63,x64,x65,x66,x67,x68,x71,x72,x73,x74,x75,x76,x77,x78,x81,x82,x83,x84,
x85,x86,x87,x88,digit
```

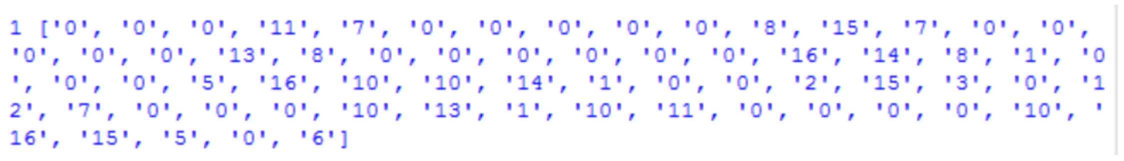
For an image with 8*8 blocks, “x23” represents the block located at the **second row from the top** and the **third column from the left**. The last one of the header is the real digit which that image shows, regarded as the class label.

Since the Python program sets “lines = lines[871:874]”, it extracts three instances from the csv file and produce corresponding images. The positions of the three instances in “optdigits.tes.csv” are shown in the following screen shot.



873	0	0	0	11	7	0	0	0	0	0	8	15	7	0	0
874	0	0	2	13	15	1	0	0	0	1	14	13	15	4	0
875	0	0	0	6	14	0	0	0	0	0	4	16	6	0	0

For the first image, the Python program prints number “1” followed by an array with 64 values of the 64 blocks, as is shown below. Each value ranges from 0 to 16, representing the number of pixels of that block. Among them, the first appeared “11” which is the fourth element in the array, means that the color intensity of block located at the first row from the top and the fourth column from the left is 11.



```
1 ['0', '0', '0', '11', '7', '0', '0', '0', '0', '0', '8', '15', '7', '0', '0',
'0', '0', '0', '13', '8', '0', '0', '0', '0', '0', '0', '16', '14', '8', '1', '0
', '0', '0', '5', '16', '10', '10', '14', '1', '0', '0', '2', '15', '3', '0', '1
2', '7', '0', '0', '0', '10', '13', '1', '10', '11', '0', '0', '0', '0', '10', '
16', '15', '5', '0', '6']
```

Then, the program prints out five columns for the 64 blocks. The first column is the row number of a block, ranging from 0 to 7. The second column is the column number of a block, also ranging from 0 to 7. The third column is the index of a block, ranging from 0 to 63. The fourth column is the value of the integer of a block, ranging from 0 to 16. The fifth column is the RGB value of a block, ranging from 0 to 255.

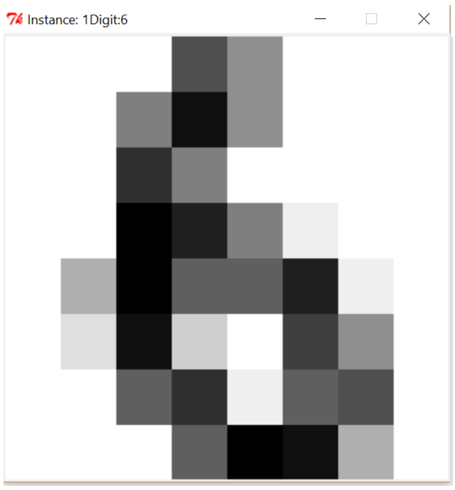
```
0 0 0 0 255
0 1 1 0 255
0 2 2 0 255
0 3 3 11 79
0 4 4 7 143
0 5 5 0 255
0 6 6 0 255
0 7 7 0 255
1 0 8 0 255
1 1 9 0 255
1 2 10 8 127
1 3 11 15 15
1 4 12 7 143
1 5 13 0 255
1 6 14 0 255
1 7 15 0 255
2 0 16 0 255
2 1 17 0 255
```

In order to draw a whole image with pixel RGB values, it is necessary to change the scale from 0-16 to 255-0 for each block. The pixel RGB of white color is (255,255,255), and the pixel RGB of black color is (0,0,0). Since it is a bitmap (black-and-white), for each pixel, the RGB three values should be the same ranging from 255 to 0. Then the program adopts the following equation to do the transformation.

```
#change scale from 0-16 to 255-0
intensity = int(((16-val)/16.0)*255)
```

Then, to better visualize in a larger figure, the Python program blows up the image by a factor of 50.

Finally, to display the image, a window titled with the instance number of the image followed by the correct digit number it represents, is generated, as is shown below.



```
1 ['0', '0', '0', '11', '7', '0', '0', '0'
, '0', '0', '8', '15', '7', '0', '0', '0',
'0', '0', '13', '8', '0', '0', '0', '0', '
0', '0', '16', '14', '8', '1', '0', '0', '
0', '5', '16', '10', '10', '14', '1', '0',
'0', '2', '15', '3', '0', '12', '7', '0',
'0', '0', '10', '13', '1', '10', '11', '0'
, '0', '0', '0', '10', '16', '15', '5', '0
', '6']
0 0 0 0 255
0 1 1 0 255
0 2 2 0 255
0 3 3 11 79
0 4 4 7 143
0 5 5 0 255
0 6 6 0 255
0 7 7 0 255
```

Compared with the two screen shots, it can be found that the first three blocks on the first row are white colors. The 4th block on the first row has original value “11”, transformed to RGB (79,79,79). The 5th block on the first row has original value “7”, transformed to RGB (143,143,143), a bit lighter than the 4th block on the same row. As a whole, the image indeed looks like a handwritten digit “6”.

After clicking on the first generated image, the image disappeared, and then press enter to view the second image (digit “8”). Then likewise the third image (digit “4”) generates.

3 Replicating the Results

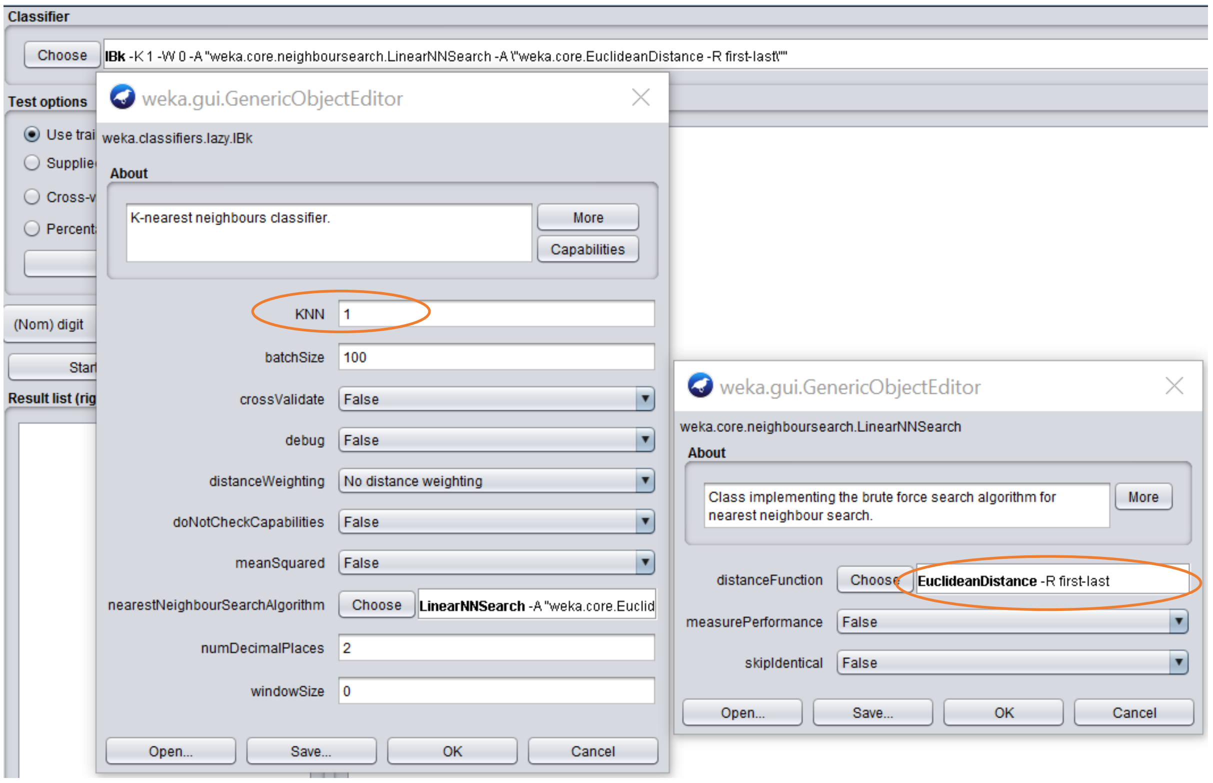
(Weka log is not quite easy to read, so I decide not to append it to my documentation.)

3.1 Methods

Step1: Weka Explorer -> Open file “optdigits.tra.arff”.

Step2: Classify -> Choose lazy -> IBk classifier.

Step3: Set kNN=1 -> Choose “EuclideanDistancee”.



Classifier

Choose **IBk** -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.gui.GenericObjectEditor

weka.classifiers.lazy.IBk

About

K-nearest neighbours classifier.

KNN	1
batchSize	100
crossValidate	False
debug	False
distanceWeighting	No distance weighting
doNotCheckCapabilities	False
meanSquared	False
nearestNeighbourSearchAlgorithm	Choose **LinearNNSearch** -A "weka.core.Euclid
numDecimalPlaces	2
windowSize	0

weka.gui.GenericObjectEditor

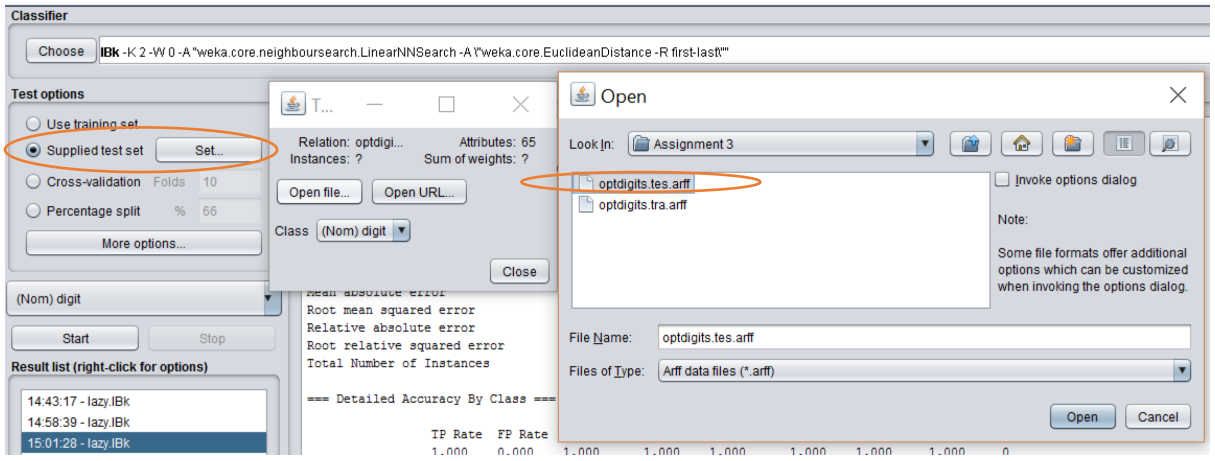
weka.core.neighboursearch.LinearNNSearch

About

Class implementing the brute force search algorithm for nearest neighbour search.

distanceFunction	Choose **EuclideanDistance** -R first-last
measurePerformance	False
skipIdentical	False

Step4: Select supplied test set as “optdigits.tes.arff”.



Classifier

Choose **IBk** -K 2 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Test options

Use training set

Supplied test set Set...

Cross-validation Folds 10

Percentage split % 66

Relation: optdigi... Attributes: 65
Instances: ? Sum of weights: ?

Class (Nom) digit

Open

Look In: Assignment 3

optdigits.tes.arff
optdigits.tra.arff

Invoke options dialog

Note:

Some file formats offer additional options which can be customized when invoking the options dialog.

File Name: optdigits.tes.arff

Files of Type: Arff data files (*.arff)

```
Mean absolute error
Root mean squared error
Relative absolute error
Root relative squared error
Total Number of Instances

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate
          1.000    0.000    1.000    1.000    1.000    1.000    1.000    1.000    0
```

Result list (right-click for options)

14:43:17 - lazy.IBk
14:58:39 - lazy.IBk
15:01:28 - lazy.IBk

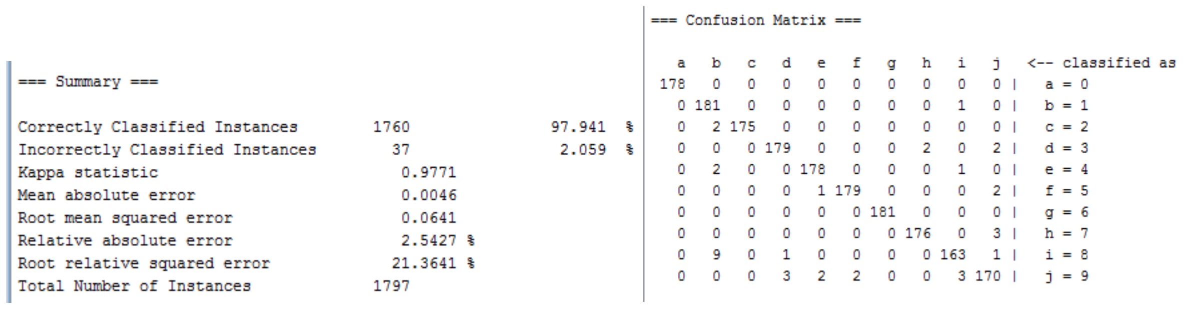
Step5: Start classification.

Step6: Change the number of K of IBk classifier.

3.2 Results

I use 3823 training instances to build the model and evaluate it with1797 test instances. I select IBk (k-nearest neighbor) classifier as the instance-based classifier.

1. k=1, using 1 nearest neighbor(s) for classification

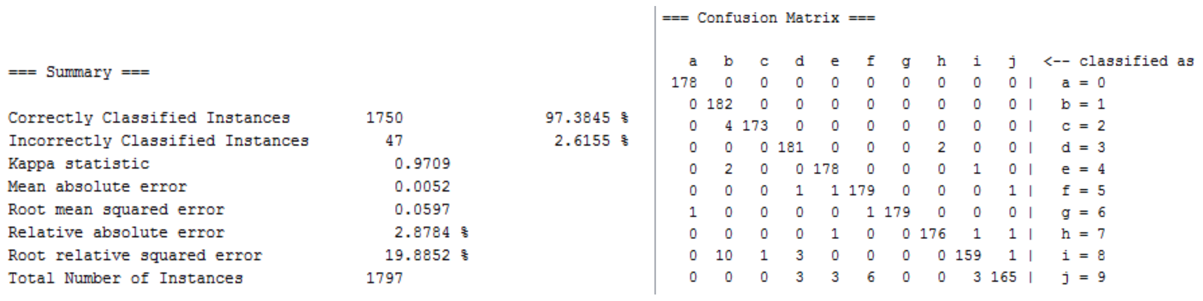


```
=== Summary ===

Correctly Classified Instances        1760               97.941  %
Incorrectly Classified Instances        37                2.059  %
Kappa statistic                          0.9771
Mean absolute error                      0.0046
Root mean squared error                  0.0641
Relative absolute error                  2.5427 %
Root relative squared error             21.3641 %
Total Number of Instances             1797
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 181   0   0   0   0   0   0   1   0 |   b = 1
   0   2 175   0   0   0   0   0   0   0 |   c = 2
   0   0   0 179   0   0   0   2   0   2 |   d = 3
   0   2   0   0 178   0   0   0   1   0 |   e = 4
   0   0   0   0   1 179   0   0   0   2 |   f = 5
   0   0   0   0   0   0 181   0   0   0 |   g = 6
   0   0   0   0   0   0   0 176   0   3 |   h = 7
   0   9   0   1   0   0   0   0 163   1 |   i = 8
   0   0   0   3   2   2   0   0   3 170 |   j = 9
```

The actual accuracy rate is 97.941%, and the given accuracy rate is 98.00%. They are much similar. Based on values of **F-Measure**, it can be found that digit "8" and "9" are more difficult to distinguish, and digit "0" and "6" are easier to distinguish than others.

2. k=2, using 2 nearest neighbor(s) for classification

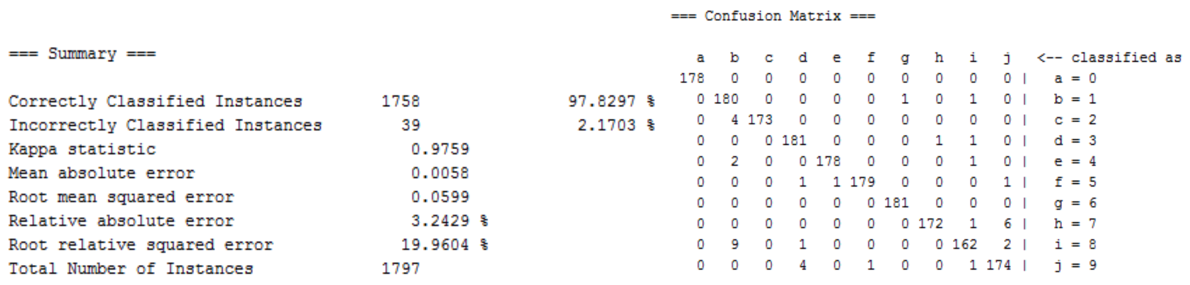


```
=== Summary ===

Correctly Classified Instances        1750               97.3845 %
Incorrectly Classified Instances        47                2.6155 %
Kappa statistic                          0.9709
Mean absolute error                      0.0052
Root mean squared error                  0.0597
Relative absolute error                  2.8784 %
Root relative squared error             19.8852 %
Total Number of Instances             1797
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 182   0   0   0   0   0   0   0   0 |   b = 1
   0   4 173   0   0   0   0   0   0   0 |   c = 2
   0   0   0 181   0   0   0   2   0   0 |   d = 3
   0   2   0   0 178   0   0   0   1   0 |   e = 4
   0   0   0   1   1 179   0   0   0   1 |   f = 5
   1   0   0   0   0   1 179   0   0   0 |   g = 6
   0   0   0   0   1   0   0 176   1   1 |   h = 7
   0  10   1   3   0   0   0   0 159   1 |   i = 8
   0   0   0   3   3   6   0   0   3 165 |   j = 9
```

The actual accuracy rate is 97.3845%, and the given accuracy rate is 97.38%. They are much similar. Based on values of **F-Measure**, it can be found that digit "8" and "9" are more difficult to distinguish, and digit "0" and "6" are easier to distinguish than others.

3. k=3, using 3 nearest neighbor(s) for classification

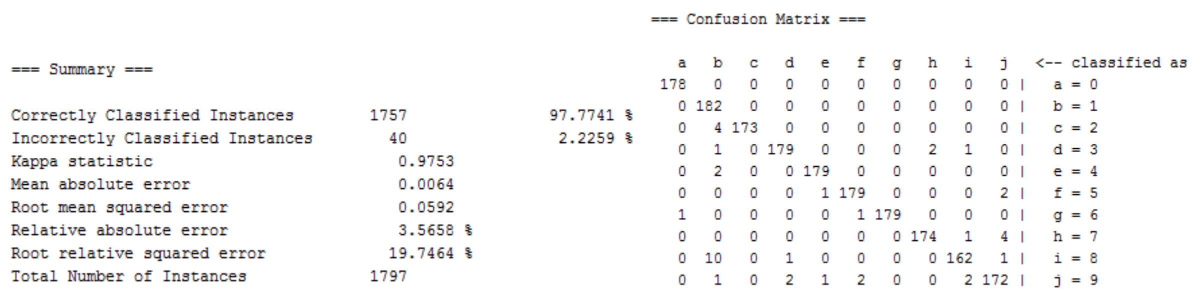


```
=== Summary ===

Correctly Classified Instances        1758               97.8297 %
Incorrectly Classified Instances        39                2.1703 %
Kappa statistic                          0.9759
Mean absolute error                      0.0058
Root mean squared error                  0.0599
Relative absolute error                  3.2429 %
Root relative squared error             19.9604 %
Total Number of Instances             1797
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 180   0   0   0   0   1   0   1   0 |   b = 1
   0   4 173   0   0   0   0   0   0   0 |   c = 2
   0   0   0 181   0   0   0   1   1   0 |   d = 3
   0   2   0   0 178   0   0   0   1   0 |   e = 4
   0   0   0   1   1 179   0   0   0   1 |   f = 5
   0   0   0   0   0   0 181   0   0   0 |   g = 6
   0   0   0   0   0   0   0 172   1   6 |   h = 7
   0   9   0   1   0   0   0   0 162   2 |   i = 8
   0   0   0   4   0   1   0   0   1 174 |   j = 9
```

The actual accuracy rate is 97.8297%, and the given accuracy rate is 97.83%. They are much similar. Based on values of **F-Measure**, it can be found that digit "8" and "1" are more difficult to distinguish, and digit "0" and "6" are easier to distinguish than others.

4. k=4, using 4 nearest neighbor(s) for classification

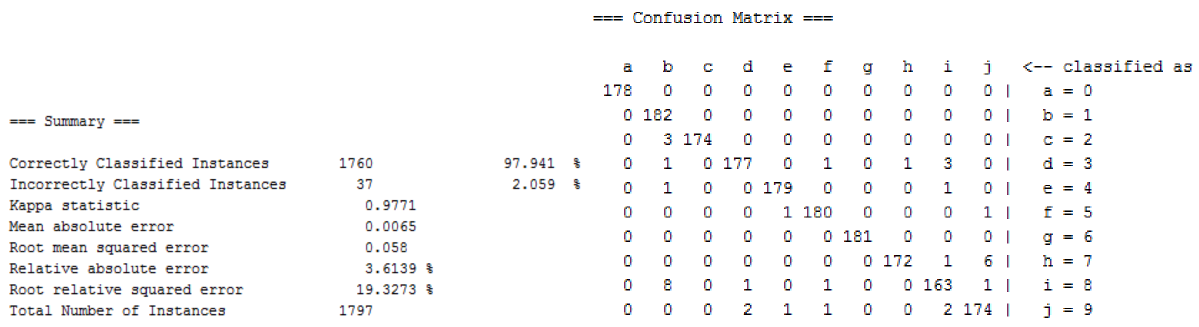


```
=== Summary ===

Correctly Classified Instances        1757               97.7741 %
Incorrectly Classified Instances        40                2.2259 %
Kappa statistic                          0.9753
Mean absolute error                      0.0064
Root mean squared error                  0.0592
Relative absolute error                  3.5658 %
Root relative squared error             19.7464 %
Total Number of Instances             1797
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 182   0   0   0   0   0   0   0   0 |   b = 1
   0   4 173   0   0   0   0   0   0   0 |   c = 2
   0   1   0 179   0   0   0   2   1   0 |   d = 3
   0   2   0   0 179   0   0   0   0   0 |   e = 4
   0   0   0   0   1 179   0   0   0   2 |   f = 5
   1   0   0   0   0   1 179   0   0   0 |   g = 6
   0   0   0   0   0   0   0 174   1   4 |   h = 7
   0  10   0   1   0   0   0   0 162   1 |   i = 8
   0   1   0   2   1   2   0   0   2 172 |   j = 9
```

The actual accuracy rate is 97.7741%, and the given accuracy rate is 97.61%. They are much similar. Based on values of **F-Measure**, it can be found that digit "8" and "1" are more difficult to distinguish, and digit "0" and "6" are easier to distinguish than others.

5. k=5, using 5 nearest neighbor(s) for classification

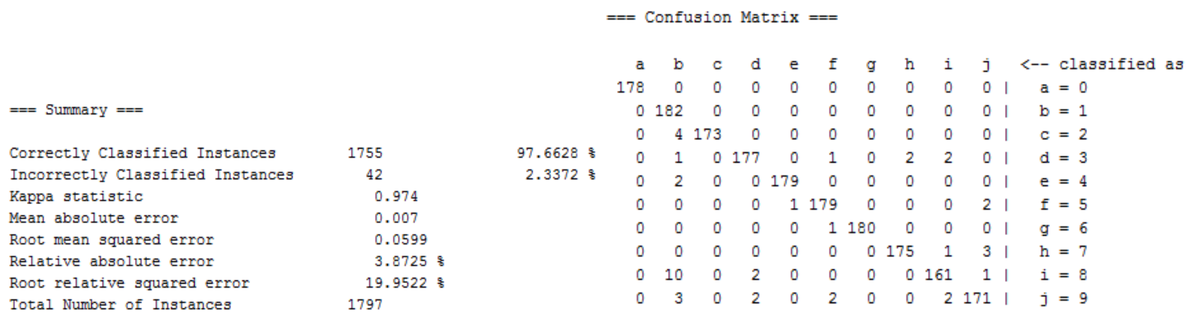


```
=== Summary ===

Correctly Classified Instances        1760               97.941  %
Incorrectly Classified Instances        37                2.059  %
Kappa statistic                          0.9771
Mean absolute error                      0.0065
Root mean squared error                  0.058
Relative absolute error                  3.6139 %
Root relative squared error             19.3273 %
Total Number of Instances             1797
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 182   0   0   0   0   0   0   0   0 |   b = 1
   0   3 174   0   0   0   0   0   0   0 |   c = 2
   0   1   0 177   0   1   0   1   3   0 |   d = 3
   0   1   0   0 179   0   0   0   1   0 |   e = 4
   0   0   0   0   1 180   0   0   0   1 |   f = 5
   0   0   0   0   0   0 181   0   0   0 |   g = 6
   0   0   0   0   0   0   0 172   1   6 |   h = 7
   0   8   0   1   0   1   0   0 163   1 |   i = 8
   0   0   0   2   1   1   0   0   2 174 |   j = 9
```

The actual accuracy rate is 97.941%, and the given accuracy rate is 97.89%. They are much similar. Based on values of **F-Measure**, it can be found that digit "8" and "9" are more difficult to distinguish, and digit "0" and "6" are easier to distinguish than others.

6. k=6, using 6 nearest neighbor(s) for classification

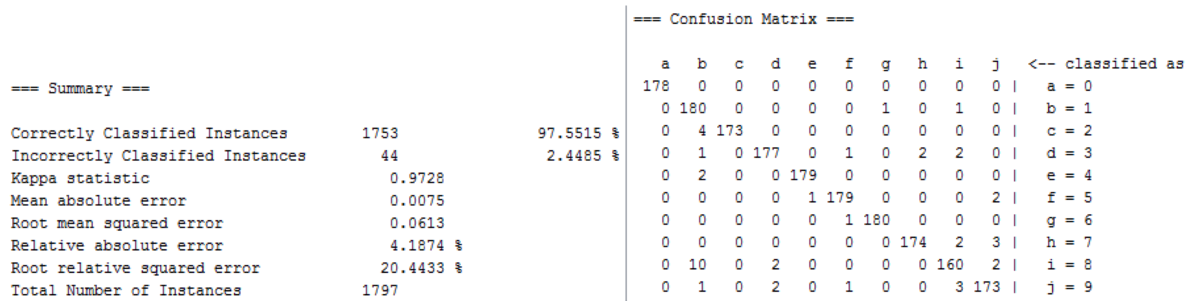


```
=== Summary ===

Correctly Classified Instances        1755               97.6628 %
Incorrectly Classified Instances        42                2.3372 %
Kappa statistic                          0.974
Mean absolute error                      0.007
Root mean squared error                  0.0599
Relative absolute error                  3.8725 %
Root relative squared error             19.9522 %
Total Number of Instances             1797
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 182   0   0   0   0   0   0   0   0 |   b = 1
   0   4 173   0   0   0   0   0   0   0 |   c = 2
   0   1   0 177   0   1   0   2   2   0 |   d = 3
   0   2   0   0 179   0   0   0   0   0 |   e = 4
   0   0   0   0   1 179   0   0   0   2 |   f = 5
   0   0   0   0   0   1 180   0   0   0 |   g = 6
   0   0   0   0   0   0   0 175   1   3 |   h = 7
   0  10   0   2   0   0   0   0 161   1 |   i = 8
   0   3   0   2   0   2   0   0   2 171 |   j = 9
```

The actual accuracy rate is 97.6628%, and the given accuracy rate is 97.77%. They are much similar. Based on values of **F-Measure**, it can be found that digit "8" and "1" are more difficult to distinguish, and digit "0" and "6" are easier to distinguish than others.

7. k=7, using 7 nearest neighbor(s) for classification

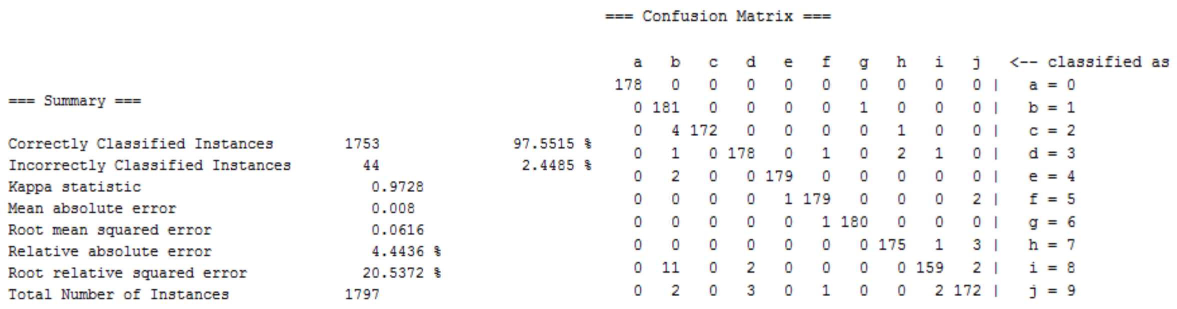


```
=== Summary ===

Correctly Classified Instances        1753               97.5515 %
Incorrectly Classified Instances        44                2.4485 %
Kappa statistic                          0.9728
Mean absolute error                      0.0075
Root mean squared error                  0.0613
Relative absolute error                  4.1874 %
Root relative squared error             20.4433 %
Total Number of Instances             1797
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 180   0   0   0   0   1   0   1   0 |   b = 1
   0   4 173   0   0   0   0   0   0   0 |   c = 2
   0   1   0 177   0   1   0   2   2   0 |   d = 3
   0   2   0   0 179   0   0   0   0   0 |   e = 4
   0   0   0   0   1 179   0   0   0   2 |   f = 5
   0   0   0   0   0   1 180   0   0   0 |   g = 6
   0   0   0   0   0   0   0 174   2   3 |   h = 7
   0  10   0   2   0   0   0   0 160   2 |   i = 8
   0   1   0   2   0   1   0   0   3 173 |   j = 9
```

The actual accuracy rate is 97.5515%, and the given accuracy rate is 97.66%. They are much similar. Based on values of **F-Measure**, it can be found that digit “8” and “1” are more difficult to distinguish, and digit “0” and “6” are easier to distinguish than others.

8. k=8, using 8 nearest neighbor(s) for classification

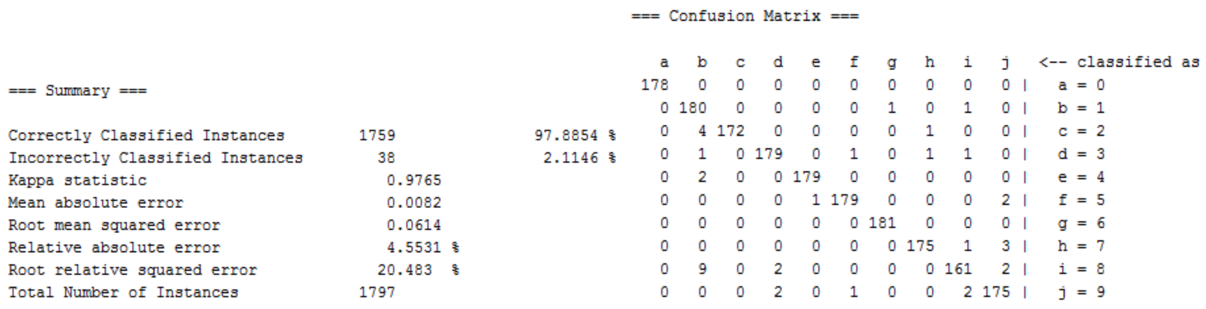


```
=== Summary ===

Correctly Classified Instances        1753               97.5515 %
Incorrectly Classified Instances        44                2.4485 %
Kappa statistic                          0.9728
Mean absolute error                      0.008
Root mean squared error                  0.0616
Relative absolute error                  4.4436 %
Root relative squared error             20.5372 %
Total Number of Instances             1797
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 181   0   0   0   0   1   0   0   0 |   b = 1
   0   4 172   0   0   0   0   1   0   0 |   c = 2
   0   1   0 178   0   1   0   2   1   0 |   d = 3
   0   2   0   0 179   0   0   0   0   0 |   e = 4
   0   0   0   0   1 179   0   0   0   2 |   f = 5
   0   0   0   0   0   1 180   0   0   0 |   g = 6
   0   0   0   0   0   0   0 175   1   3 |   h = 7
   0  11   0   2   0   0   0   0 159   2 |   i = 8
   0   2   0   3   0   1   0   0   2 172 |   j = 9
```

The actual accuracy rate is 97.5515%, and the given accuracy rate is 97.66%. They are much similar. Based on values of **F-Measure**, it can be found that digit “8” and “1” are more difficult to distinguish, and digit “0” and “6” are easier to distinguish than others.

9. k=9, using 9 nearest neighbor(s) for classification

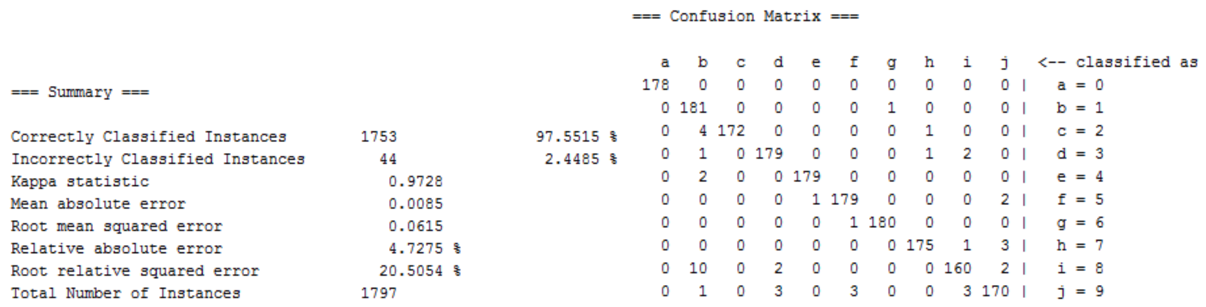


```
=== Summary ===

Correctly Classified Instances        1759               97.8854 %
Incorrectly Classified Instances        38                2.1146 %
Kappa statistic                          0.9765
Mean absolute error                      0.0082
Root mean squared error                  0.0614
Relative absolute error                  4.5531 %
Root relative squared error             20.483  %
Total Number of Instances             1797
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 180   0   0   0   0   1   0   1   0 |   b = 1
   0   4 172   0   0   0   0   1   0   0 |   c = 2
   0   1   0 179   0   1   0   1   1   0 |   d = 3
   0   2   0   0 179   0   0   0   0   0 |   e = 4
   0   0   0   0   1 179   0   0   0   2 |   f = 5
   0   0   0   0   0   0 181   0   0   0 |   g = 6
   0   0   0   0   0   0   0 175   1   3 |   h = 7
   0   9   0   2   0   0   0   0 161   2 |   i = 8
   0   0   0   2   0   1   0   0   2 175 |   j = 9
```

The actual accuracy rate is 97.8854%, and the given accuracy rate is 97.72%. They are much similar. Based on values of **F-Measure**, it can be found that digit “8” and “1” are more difficult to distinguish, and digit “0” and “6” are easier to distinguish than others.

10. k=10, using 10 nearest neighbor(s) for classification

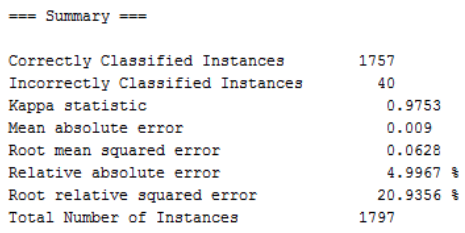


```
=== Summary ===

Correctly Classified Instances        1753               97.5515 %
Incorrectly Classified Instances        44                2.4485 %
Kappa statistic                          0.9728
Mean absolute error                      0.0085
Root mean squared error                  0.0615
Relative absolute error                  4.7275 %
Root relative squared error             20.5054 %
Total Number of Instances             1797
```

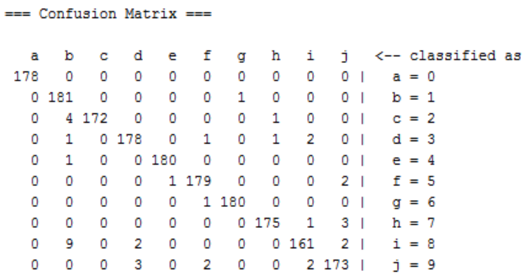
```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 181   0   0   0   0   1   0   0   0 |   b = 1
   0   4 172   0   0   0   0   1   0   0 |   c = 2
   0   1   0 179   0   0   0   1   2   0 |   d = 3
   0   2   0   0 179   0   0   0   0   0 |   e = 4
   0   0   0   0   1 179   0   0   0   2 |   f = 5
   0   0   0   0   0   1 180   0   0   0 |   g = 6
   0   0   0   0   0   0   0 175   1   3 |   h = 7
   0  10   0   2   0   0   0   0 160   2 |   i = 8
   0   1   0   3   0   3   0   0   3 170 |   j = 9
```

The actual accuracy rate is 97.5515%, and the given accuracy rate is 97.55%. They are much similar. Based on values of **F-Measure**, it can be found that digit “8” and “1” are more difficult to distinguish, and digit “0” and “6” are easier to distinguish than others.

11. k=11, using 11 nearest neighbor(s) for classification



```
=== Summary ===

Correctly Classified Instances        1757               97.7741 %
Incorrectly Classified Instances        40                2.2259 %
Kappa statistic                          0.9753
Mean absolute error                      0.009
Root mean squared error                  0.0628
Relative absolute error                  4.9967 %
Root relative squared error             20.9356 %
Total Number of Instances             1797
```



```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 181   0   0   0   0   1   0   0   0 |   b = 1
   0   4 172   0   0   0   0   1   0   0 |   c = 2
   0   1   0 178   0   1   0   1   2   0 |   d = 3
   0   1   0   0 180   0   0   0   0   0 |   e = 4
   0   0   0   0   1 179   0   0   0   2 |   f = 5
   0   0   0   0   0   1 180   0   0   0 |   g = 6
   0   0   0   0   0   0   0 175   1   3 |   h = 7
   0   9   0   2   0   0   0   0 161   2 |   i = 8
   0   0   0   3   0   2   0   0   2 173 |   j = 9
```

The actual accuracy rate is 97.7741%, and the given accuracy rate is 97.89%. They are much similar. Based on values of **F-Measure**, it can be found that digit "8" and "1" are more difficult to distinguish, and digit "0" and "6" are easier to distinguish than others.

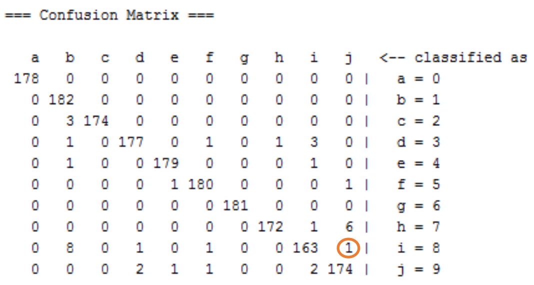
In general, the following table is the comparison of actual and given accuracy rates. They are similar.

	Actual accuracy rate	Given accuracy rate in the original file
k=1	97.941%	98.00%
k=2	97.3845%	97.38%
k=3	97.8297%	97.83%
k=4	97.7741%	97.61%
k=5	97.941%	97.89%
k=6	97.6628%	97.77%
k=7	97.5515%	97.66%
k=8	97.5515%	97.66%
k=9	97.8854%	97.72%
k=10	97.5515%	97.55%
k=11	97.7741%	97.89%

For KNN method, based on values of F-Measure, it can be found that digit "8", "1", and "9" are more difficult to distinguish, and digit "0" and "6" are easier to distinguish than others.

3.3 Identify a Misclassified Instance

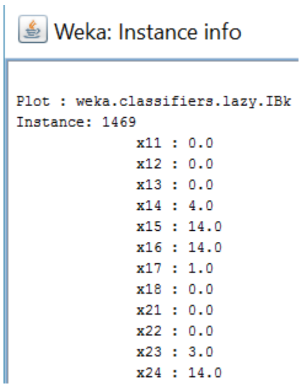
For IBk, k=5, the confusion matrix is shown below. I pick one instance that represents digit "8" but is misclassified as digit "9", circled below.



```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 182   0   0   0   0   0   0   0   0 |   b = 1
   0   3 174   0   0   0   0   0   0   0 |   c = 2
   0   1   0 177   0   1   0   1   3   0 |   d = 3
   0   1   0   0 179   0   0   0   1   0 |   e = 4
   0   0   0   0   1 180   0   0   0   1 |   f = 5
   0   0   0   0   0   0 181   0   0   0 |   g = 6
   0   0   0   0   0   0   0 172   1   6 |   h = 7
   0   8   0   1   0   1   0   0 163   1 |   i = 8
   0   0   0   2   1   1   0   0   2 174 |   j = 9
```

Right click on the model, and choose "Visualize classifier errors".

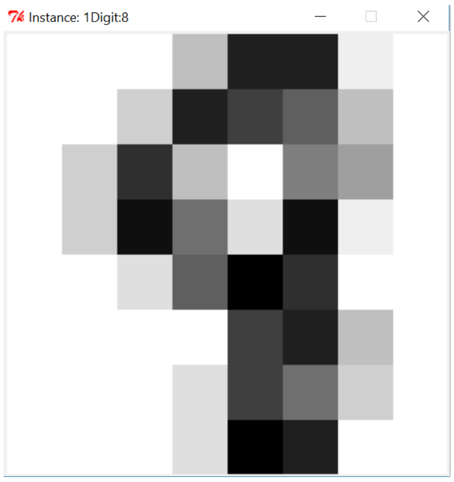
Find the green square box which shows "predicted digit : 9; digit : 8". The detailed information is shown below. It is instance No. 1469.



```
Plot : weka.classifiers.lazy.IBk
Instance: 1469
              x11 : 0.0
              x12 : 0.0
              x13 : 0.0
              x14 : 4.0
              x15 : 14.0
              x16 : 14.0
              x17 : 1.0
              x18 : 0.0
              x21 : 0.0
              x22 : 0.0
              x23 : 3.0
              x24 : 14.0
```

Modify the Python program to display the instance No. 1469.

```
#check out instance number 1469
# it was predicted 9 but actually an 8 (based on IBk, k=5)
lines = lines[1468:1469]
```



It can be seen that the digit “8” of instance No. 1469 indeed looks like digit “9”.

4 Other Classifiers Comparison

1. Function classifier: Logistic Regression Model (Non-instance based)

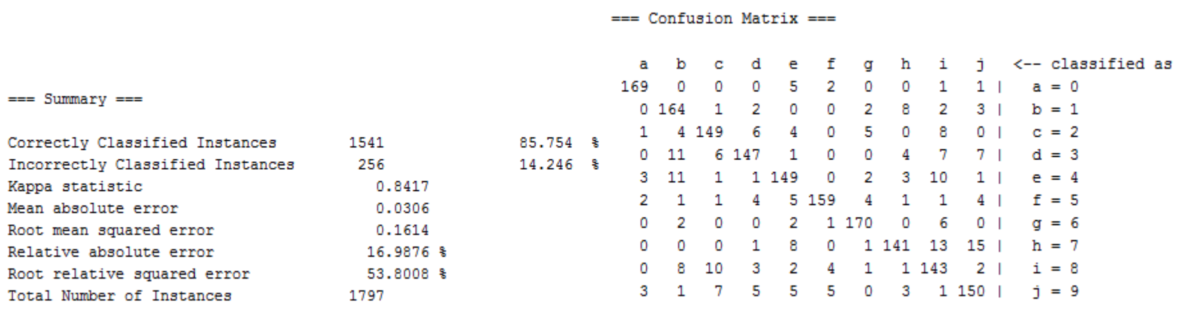
Accuracy rate: 92.1536%.

```
=== Summary ===

Correctly Classified Instances        1656               92.1536 %
Incorrectly Classified Instances       141                7.8464 %
Kappa statistic                          0.9128
Mean absolute error                      0.0157
Root mean squared error                  0.1236
Relative absolute error                  8.6954 %
Root relative squared error             41.2095 %
Total Number of Instances             1797
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 170   0   1   0   1   3   1   1   0   1 |   a = 0
   3 169   0   0   4   0   1   1   4   0 |   b = 1
   0   2 169   1   0   1   1   0   3   0 |   c = 2
   0   0   3 167   0   5   0   1   5   2 |   d = 3
   1   3   0   0 168   0   0   3   1   5 |   e = 4
   0   0   2   5   0 171   0   2   1   1 |   f = 5
   0   0   1   0   8   0 171   0   1   0 |   g = 6
   3   0   0   0   1   5   0 165   2   3 |   h = 7
   0   6   3   7   1   2   3   2 146   4 |   i = 8
   6   0   0   6   0   1   1   3   3 160 |   j = 9
```

2. Trees classifier: J48 (Non-instance based)

Accuracy rate: 85.754%.

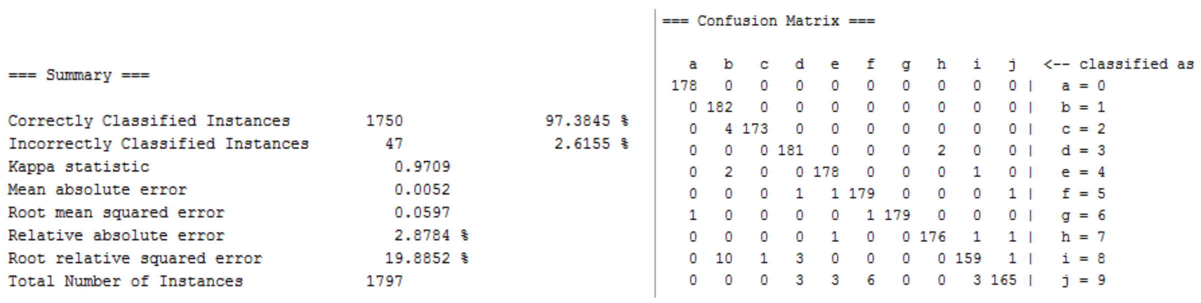


```
=== Summary ===

Correctly Classified Instances        1541               85.754  %
Incorrectly Classified Instances       256               14.246  %
Kappa statistic                          0.8417
Mean absolute error                      0.0306
Root mean squared error                  0.1614
Relative absolute error                 16.9876 %
Root relative squared error             53.8008 %
Total Number of Instances             1797
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 169   0   0   0   5   2   0   0   1   1 |   a = 0
   0 164   1   2   0   0   2   8   2   3 |   b = 1
   1   4 149   6   4   0   5   0   8   0 |   c = 2
   0  11   6 147   1   0   0   4   7   7 |   d = 3
   3  11   1   1 149   0   2   3  10   1 |   e = 4
   2   1   1   4   5 159   4   1   1   4 |   f = 5
   0   2   0   0   2   1 170   0   6   0 |   g = 6
   0   0   0   1   8   0   1 141  13  15 |   h = 7
   0   8  10   3   2   4   1   1 143   2 |   i = 8
   3   1   7   5   5   5   0   3   1 150 |   j = 9
```

3. Instance-based classifier: IBK (kNN, k=2)

Accuracy rate: 97.3845%



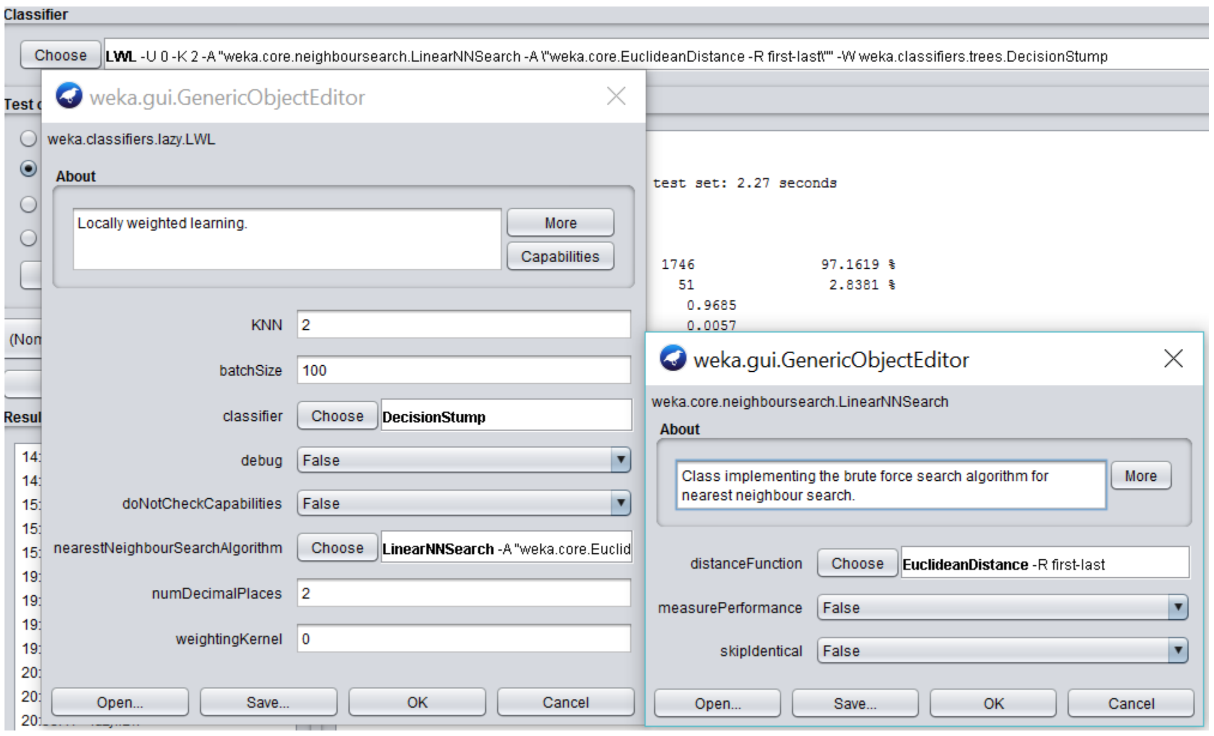
```
=== Summary ===

Correctly Classified Instances        1750               97.3845 %
Incorrectly Classified Instances        47                2.6155 %
Kappa statistic                          0.9709
Mean absolute error                      0.0052
Root mean squared error                  0.0597
Relative absolute error                  2.8784 %
Root relative squared error             19.8852 %
Total Number of Instances             1797
```

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 182   0   0   0   0   0   0   0   0 |   b = 1
   0   4 173   0   0   0   0   0   0   0 |   c = 2
   0   0   0 181   0   0   0   2   0   0 |   d = 3
   0   2   0   0 178   0   0   0   1   0 |   e = 4
   0   0   0   1   1 179   0   0   0   1 |   f = 5
   1   0   0   0   0   1 179   0   0   0 |   g = 6
   0   0   0   0   1   0   0 176   1   1 |   h = 7
   0  10   1   3   0   0   0   0 159   1 |   i = 8
   0   0   0   3   3   6   0   0   3 165 |   j = 9
```

4. Instance-based classifier: LWL (k=2)

Locally weighted learning. Uses an instance-based algorithm to assign instance weights which are then used by a specified WeightedInstancesHandler.

Accuracy rate: 97.1619%



Classifier

Choose LWL -U 0 -K 2 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" -W weka.classifiers.trees.DecisionStump

weka.gui.GenericObjectEditor

weka.classifiers.lazy.LWL

About

Locally weighted learning.

KNN	2
batchSize	100
classifier	DecisionStump
debug	False
doNotCheckCapabilities	False
nearestNeighbourSearchAlgorithm	LinearNNSearch -A "weka.core.Euclid
numDecimalPlaces	2
weightingKernel	0

```
test set: 2.27 seconds

 1746               97.1619 %
   51                2.8381 %
    0.9685
    0.0057
```

weka.gui.GenericObjectEditor

weka.core.neighboursearch.LinearNNSearch

About

Class implementing the brute force search algorithm for nearest neighbour search.

distanceFunction	EuclideanDistance -R first-last
measurePerformance	False
skipIdentical	False

```
=== Summary ===

Correctly Classified Instances        1746               97.1619 %
Incorrectly Classified Instances        51                2.8381 %
Kappa statistic                          0.9685
Mean absolute error                      0.0057
Root mean squared error                  0.0753
Relative absolute error                  3.1534 %
Root relative squared error             25.1133 %
Total Number of Instances             1797
```

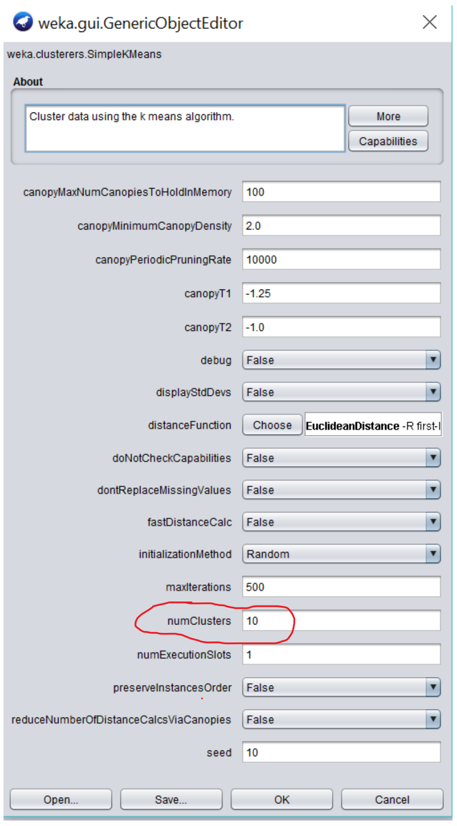
```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 178   0   0   0   0   0   0   0   0   0 |   a = 0
   0 180   0   0   0   0   2   0   0   0 |   b = 1
   0   3 174   0   0   0   0   0   0   0 |   c = 2
   0   0   0 177   0   1   0   1   2   2 |   d = 3
   0   2   0   0 178   0   0   0   1   0 |   e = 4
   0   0   0   1   1 179   0   0   0   1 |   f = 5
   1   0   0   0   0   0 180   0   0   0 |   g = 6
   0   0   0   0   1   0   0 172   1   5 |   h = 7
   0   9   1   0   0   0   0   0 160   4 |   i = 8
   0   0   0   2   3   5   0   0   2 168 |   j = 9
```

In general, instance-based classifiers perform better than non-instance-based classifiers in this problem domain. For instance-based classifiers, they do not conduct rule induction, instead, they compare instances in the test set with instances in the training test. They construct assumptions based on training set, and apply the assumptions to the test set. If there are many overlapping instances or patterns between the training and test sets, the accuracy rate will be high. In this digit recognition scenario, the thousands of instances come from 43 people (30 training set, 13 test set), so there exists some similarity between instances from the 13 people and instances from the previous 30 people. Therefore, the assumptions based on the training set could be well applied to the test set, and the corresponding accuracy rate could also be high.

For other non-instance based classifiers such as logistic regression and J48 (decision tree), they fit (parametric or non-parametric) models based on the training set and predict instances in the test set based on the models. Some statistical issues in these models may cause lower accuracy rates than those of instance-based classifiers.

5 Comparison of Classifiers and Clusterers

1. SimpleKMeans Cluster -> 10 clusters


weka.gui.GenericObjectEditor

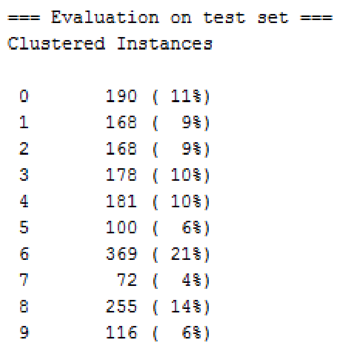
weka.clusterers.SimpleKMeans

About

Cluster data using the k means algorithm.

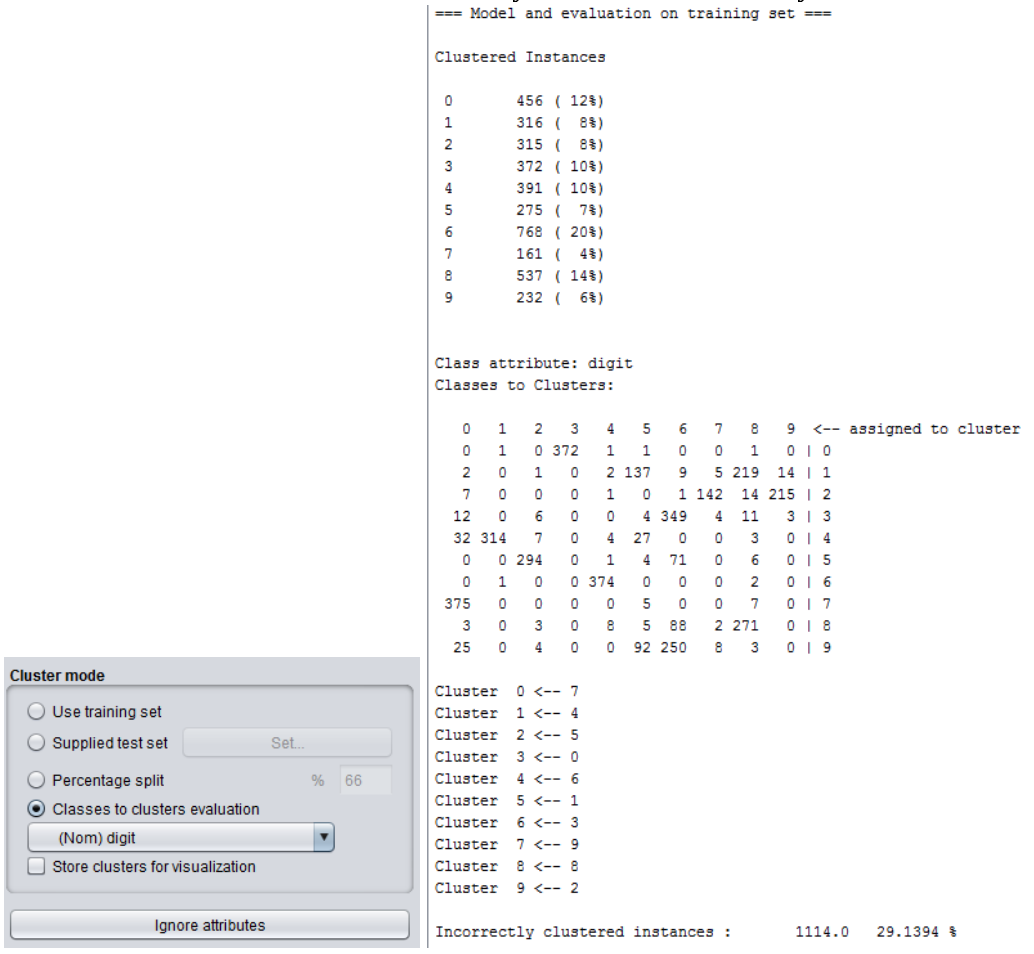
canopyMaxNumCanopiesToHoldInMemory	100
canopyMinimumCanopyDensity	2.0
canopyPeriodicPruningRate	10000
canopyT1	-1.25
canopyT2	-1.0
debug	False
displayStdDevs	False
distanceFunction	EuclideanDistance -R first-l
doNotCheckCapabilities	False
dontReplaceMissingValues	False
fastDistanceCalc	False
initializationMethod	Random
maxIterations	500
numClusters	10
numExecutionSlots	1
preserveInstancesOrder	False
reduceNumberOfDistanceCalcsViaCanopies	False
seed	10

- Result of Supplied test set
  All the nine digits should be even distributed in the test set. However, the clustered result of the test set is quite not uniform distributed.



```
=== Evaluation on test set ===
Clustered Instances

 0       190 ( 11%)
 1       168 (  9%)
 2       168 (  9%)
 3       178 ( 10%)
 4       181 ( 10%)
 5       100 (  6%)
 6       369 ( 21%)
 7        72 (  4%)
 8       255 ( 14%)
 9       116 (  6%)
```

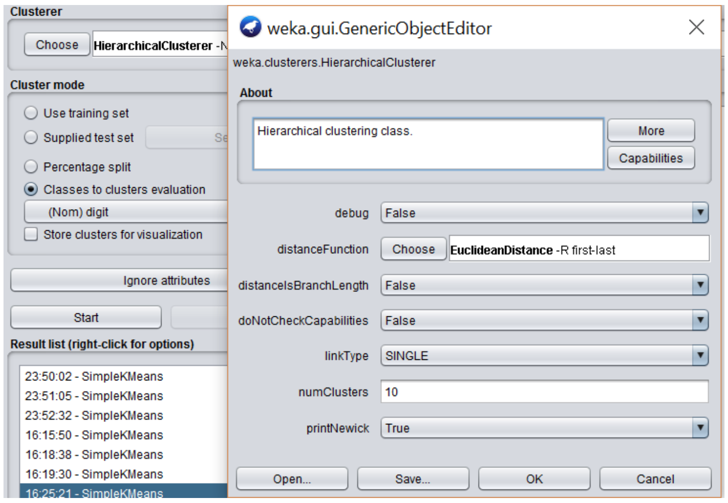
- Result of training set cluster evaluation
  Almost one third of instances are incorrectly clustered. The accuracy rate is about 70%.



```
Cluster mode
( ) Use training set
( ) Supplied test set   Set...
( ) Percentage split   % 66
(•) Classes to clusters evaluation
    (Nom) digit
[ ] Store clusters for visualization
Ignore attributes
```

```
=== Model and evaluation on training set ===

Clustered Instances

 0       456 ( 12%)
 1       316 (  8%)
 2       315 (  8%)
 3       372 ( 10%)
 4       391 ( 10%)
 5       275 (  7%)
 6       768 ( 20%)
 7       161 (  4%)
 8       537 ( 14%)
 9       232 (  6%)


Class attribute: digit
Classes to Clusters:

   0   1   2   3   4   5   6   7   8   9  <-- assigned to cluster
   0   1   0 372   1   1   0   0   1   0 | 0
   2   0   1   0   2 137   9   5 219  14 | 1
   7   0   0   0   1   0   1 142  14 215 | 2
  12   0   6   0   0   4 349   4  11   3 | 3
  32 314   7   0   4  27   0   0   3   0 | 4
   0   0 294   0   1   4  71   0   6   0 | 5
   0   1   0   0 374   0   0   0   2   0 | 6
 375   0   0   0   0   5   0   0   7   0 | 7
   3   0   3   0   8   5  88   2 271   0 | 8
  25   0   4   0   0  92 250   8   3   0 | 9

Cluster  0 <-- 7
Cluster  1 <-- 4
Cluster  2 <-- 5
Cluster  3 <-- 0
Cluster  4 <-- 6
Cluster  5 <-- 1
Cluster  6 <-- 3
Cluster  7 <-- 9
Cluster  8 <-- 8
Cluster  9 <-- 2

Incorrectly clustered instances :	1114.0	 29.1394 %
```

2. Hierarchical Cluster -> 10 clusters


weka.gui.GenericObjectEditor

weka.clusterers.HierarchicalClusterer

About: Hierarchical clustering class.

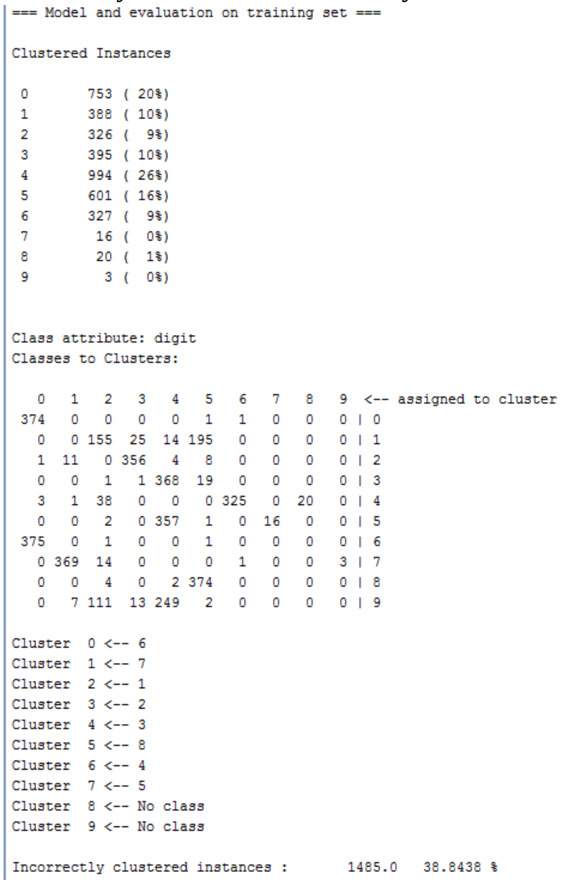
debug	False
distanceFunction	EuclideanDistance -R first-last
distanceIsBranchLength	False
doNotCheckCapabilities	False
linkType	SINGLE
numClusters	10
printNewick	True

Clusterer: HierarchicalClusterer -N

Cluster mode: Classes to clusters evaluation, (Nom) digit

Result list (right-click for options):
23:50:02 - SimpleKMeans
23:51:05 - SimpleKMeans
23:52:32 - SimpleKMeans
16:15:50 - SimpleKMeans
16:18:38 - SimpleKMeans
16:19:30 - SimpleKMeans
16:25:21 - SimpleKMeans

- Result of training set cluster evaluation (change linkType to "Average")
  In the training set, only 8 digit numbers have corresponding classes. 38.8438% instances are incorrectly clustered. The accuracy rate is about 61%.


```
=== Model and evaluation on training set ===

Clustered Instances

 0       753 ( 20%)
 1       388 ( 10%)
 2       326 (  9%)
 3       395 ( 10%)
 4       994 ( 26%)
 5       601 ( 16%)
 6       327 (  9%)
 7        16 (  0%)
 8        20 (  1%)
 9         3 (  0%)


Class attribute: digit
Classes to Clusters:

   0   1   2   3   4   5   6   7   8   9  <-- assigned to cluster
 374   0   0   0   0   1   1   0   0   0 | 0
   0   0 155  25  14 195   0   0   0   0 | 1
   1  11   0 356   4   8   0   0   0   0 | 2
   0   0   1   1 368  19   0   0   0   0 | 3
   3   1  38   0   0   0 325   0  20   0 | 4
   0   0   2   0 357   1   0  16   0   0 | 5
 375   0   1   0   0   1   0   0   0   0 | 6
   0 369  14   0   0   0   1   0   0   3 | 7
   0   0   4   0   2 374   0   0   0   0 | 8
   0   7 111  13 249   2   0   0   0   0 | 9

Cluster  0 <-- 6
Cluster  1 <-- 7
Cluster  2 <-- 1
Cluster  3 <-- 2
Cluster  4 <-- 3
Cluster  5 <-- 8
Cluster  6 <-- 4
Cluster  7 <-- 5
Cluster  8 <-- No class
Cluster  9 <-- No class

Incorrectly clustered instances :      1485.0    38.8438 %
```

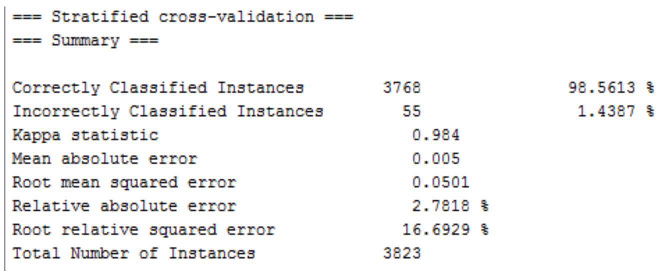
The performance of the clustering is not quite surprising.

For classification, there exists gold standard as the class labels for supervised learning.

For clustering, it is unsupervised learning without any gold standard. Similar instances are clustered as one group, and the class label is ignored. It should be less accurate than classification since similar instances probably may come from the same person but not always the same digit number. The personal handwritten style may affect the cluster results. Clustering may be based on the characteristics of both digit numbers and different people handwritten styles in this domain.

6 10-fold Cross-Validation of Training set

I pick the best classifier "IBK, K=5" so far for evaluating using 10-fold cross-validation of the training set. The accuracy rate is 98.5613%, higher than the previous provided test set (97.941%).



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3768               98.5613 %
Incorrectly Classified Instances        55                1.4387 %
Kappa statistic                          0.984
Mean absolute error                      0.005
Root mean squared error                  0.0501
Relative absolute error                  2.7818 %
Root relative squared error             16.6929 %
Total Number of Instances             3823
```

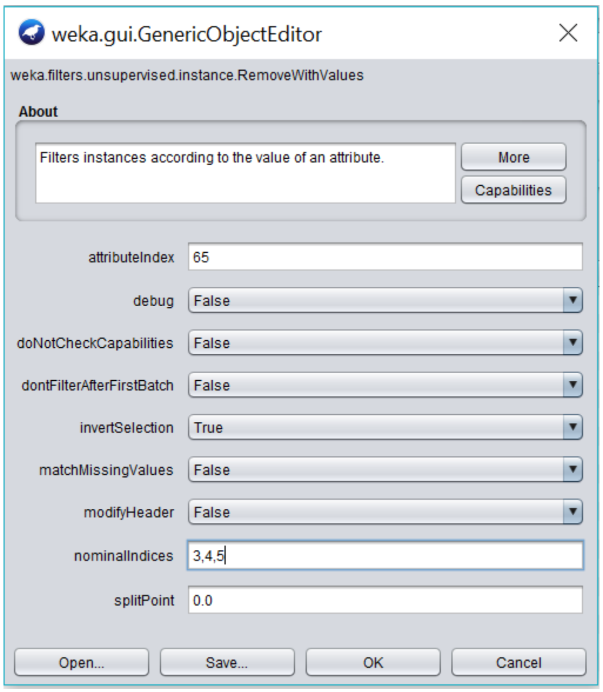
For the 10-fold cross-validation, each subset in the 10 subsets of the training set has been used as test set, and then 10 models have been evaluated on the 10 subsets. The classified accuracy rate is an average of accuracy rates of the 10 models. Compared to evaluating model on another test set, 10-fold cross-validation of the training set can decrease the bias between actual training/test set between the whole dataset, since each subset has been used as training or test set in the 10 models, like sampling method in statistics. Therefore, 10-fold cross-validation of the training set performs higher accuracy rate. It can also reduce overfitting to some extent.

7 Accuracy VS. Number of Classes

I suppose the question is to find a relationship between number of class labels and the accuracy rate of classification. Expanding the dataset means adding more instances with more class labels to it.

7.1 Methods

Step1: "optdigits.tra.arff" -> Preprocess -> RemoveWithValues -> select instances of certain digit numbers



weka.gui.GenericObjectEditor

weka.filters.unsupervised.instance.RemoveWithValues

About

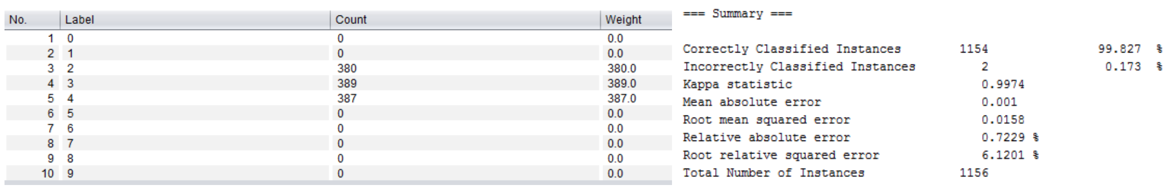
Filters instances according to the value of an attribute.

attributeIndex	65
debug	False
doNotCheckCapabilities	False
dontFilterAfterFirstBatch	False
invertSelection	True
matchMissingValues	False
modifyHeader	False
nominalIndices	3,4,5
splitPoint	0.0

Step2: Classify -> IBK (k=5) -> 10-fold cross-validation

7.2 Results

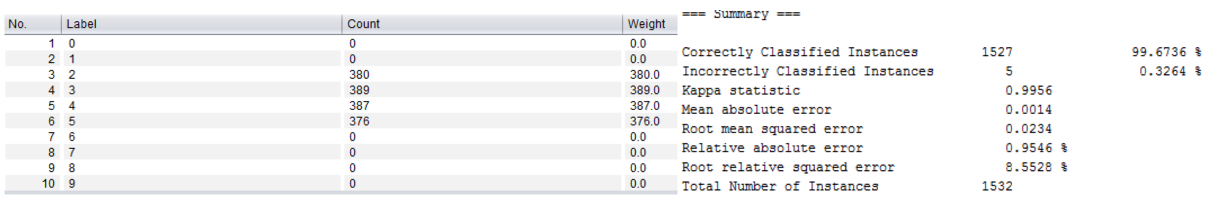
1. 3 classes (digits “2”, “3”, “4”); Accuracy rate: 99.827%



No.	Label	Count	Weight
1	0	0	0.0
2	1	0	0.0
3	2	380	380.0
4	3	389	389.0
5	4	387	387.0
6	5	0	0.0
7	6	0	0.0
8	7	0	0.0
9	8	0	0.0
10	9	0	0.0

```
=== Summary ===

Correctly Classified Instances        1154               99.827  %
Incorrectly Classified Instances         2                0.173  %
Kappa statistic                          0.9974
Mean absolute error                      0.001
Root mean squared error                  0.0158
Relative absolute error                  0.7229 %
Root relative squared error              6.1201 %
Total Number of Instances             1156
```

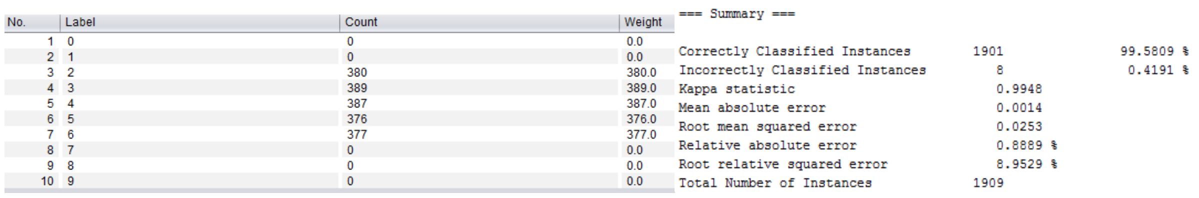
2. 4 classes (digits “2”, “3”, “4”, “5”); Accuracy rate: 99.6736%



No.	Label	Count	Weight
1	0	0	0.0
2	1	0	0.0
3	2	380	380.0
4	3	389	389.0
5	4	387	387.0
6	5	376	376.0
7	6	0	0.0
8	7	0	0.0
9	8	0	0.0
10	9	0	0.0

```
=== Summary ===

Correctly Classified Instances        1527               99.6736 %
Incorrectly Classified Instances         5                0.3264 %
Kappa statistic                          0.9956
Mean absolute error                      0.0014
Root mean squared error                  0.0234
Relative absolute error                  0.9546 %
Root relative squared error              8.5528 %
Total Number of Instances             1532
```

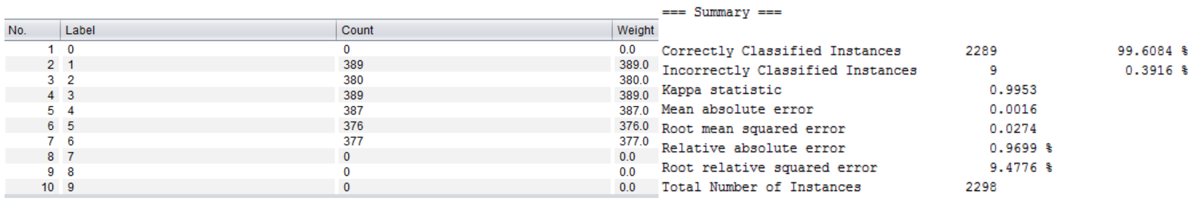
3. 5 classes (digits “2”, “3”, “4”, “5”, “6”); Accuracy rate: 99.5809%



No.	Label	Count	Weight
1	0	0	0.0
2	1	0	0.0
3	2	380	380.0
4	3	389	389.0
5	4	387	387.0
6	5	376	376.0
7	6	377	377.0
8	7	0	0.0
9	8	0	0.0
10	9	0	0.0

```
=== Summary ===

Correctly Classified Instances        1901               99.5809 %
Incorrectly Classified Instances         8                0.4191 %
Kappa statistic                          0.9948
Mean absolute error                      0.0014
Root mean squared error                  0.0253
Relative absolute error                  0.8889 %
Root relative squared error              8.9529 %
Total Number of Instances             1909
```

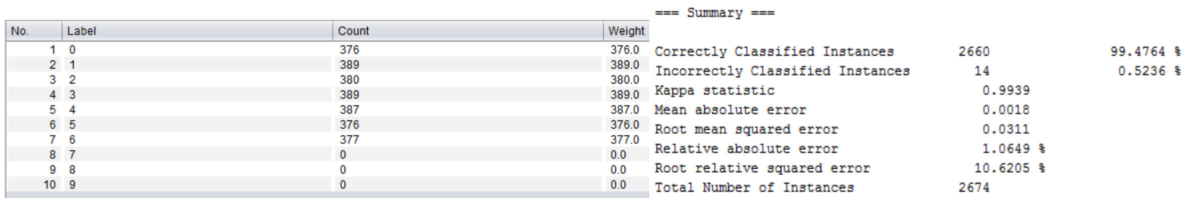
4. 6 classes (digits “1”, “2”, “3”, “4”, “5”, “6”); Accuracy rate: 99.6084%



No.	Label	Count	Weight
1	0	0	0.0
2	1	389	389.0
3	2	380	380.0
4	3	389	389.0
5	4	387	387.0
6	5	376	376.0
7	6	377	377.0
8	7	0	0.0
9	8	0	0.0
10	9	0	0.0

```
=== Summary ===

Correctly Classified Instances        2289               99.6084 %
Incorrectly Classified Instances         9                0.3916 %
Kappa statistic                          0.9953
Mean absolute error                      0.0016
Root mean squared error                  0.0274
Relative absolute error                  0.9699 %
Root relative squared error              9.4776 %
Total Number of Instances             2298
```

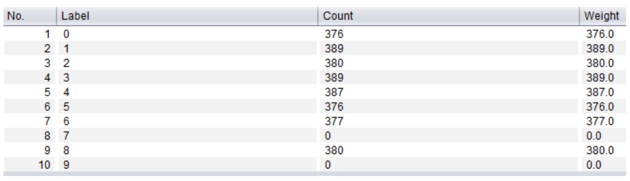
5. 7 classes (digits “0”, “1”, “2”, “3”, “4”, “5”, “6”); Accuracy rate: 99.4764%



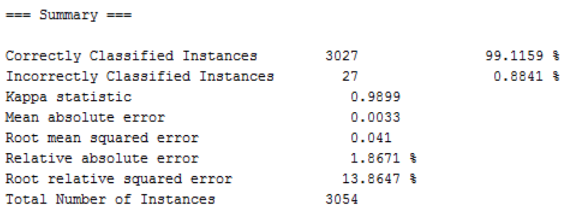
No.	Label	Count	Weight
1	0	376	376.0
2	1	389	389.0
3	2	380	380.0
4	3	389	389.0
5	4	387	387.0
6	5	376	376.0
7	6	377	377.0
8	7	0	0.0
9	8	0	0.0
10	9	0	0.0

```
=== Summary ===

Correctly Classified Instances        2660               99.4764 %
Incorrectly Classified Instances        14                0.5236 %
Kappa statistic                          0.9939
Mean absolute error                      0.0018
Root mean squared error                  0.0311
Relative absolute error                  1.0649 %
Root relative squared error             10.6205 %
Total Number of Instances             2674
```

6. 8 classes (digits “0”, “1”, “2”, “3”, “4”, “5”, “6”, “8”); Accuracy rate: 99.1159%

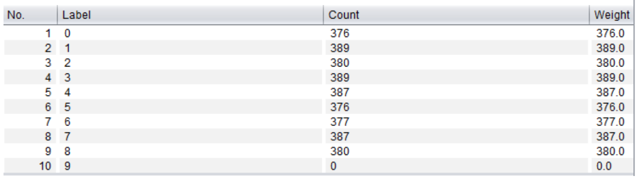


No.	Label	Count	Weight
1	0	376	376.0
2	1	389	389.0
3	2	380	380.0
4	3	389	389.0
5	4	387	387.0
6	5	376	376.0
7	6	377	377.0
8	7	0	0.0
9	8	380	380.0
10	9	0	0.0

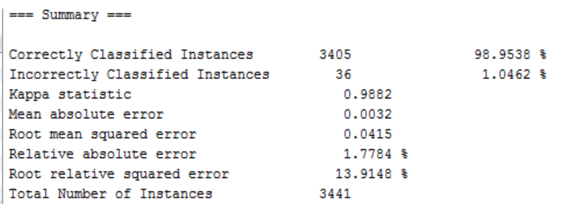


```
=== Summary ===

Correctly Classified Instances        3027               99.1159 %
Incorrectly Classified Instances        27                0.8841 %
Kappa statistic                          0.9899
Mean absolute error                      0.0033
Root mean squared error                  0.041
Relative absolute error                  1.8671 %
Root relative squared error             13.8647 %
Total Number of Instances             3054
```

7. 9 classes (digits “0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”); Accuracy rate: 98.9538%

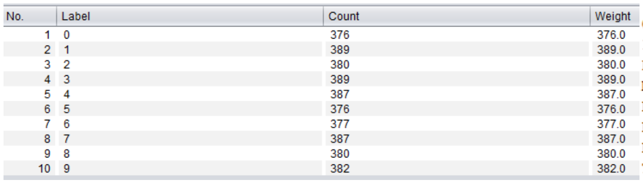


No.	Label	Count	Weight
1	0	376	376.0
2	1	389	389.0
3	2	380	380.0
4	3	389	389.0
5	4	387	387.0
6	5	376	376.0
7	6	377	377.0
8	7	387	387.0
9	8	380	380.0
10	9	0	0.0

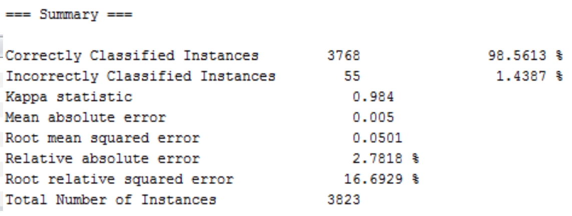


```
=== Summary ===

Correctly Classified Instances        3405               98.9538 %
Incorrectly Classified Instances        36                1.0462 %
Kappa statistic                          0.9882
Mean absolute error                      0.0032
Root mean squared error                  0.0415
Relative absolute error                  1.7784 %
Root relative squared error             13.9148 %
Total Number of Instances             3441
```

8. 10 classes (digits “0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”); Accuracy rate: 98.5613%



No.	Label	Count	Weight
1	0	376	376.0
2	1	389	389.0
3	2	380	380.0
4	3	389	389.0
5	4	387	387.0
6	5	376	376.0
7	6	377	377.0
8	7	387	387.0
9	8	380	380.0
10	9	382	382.0

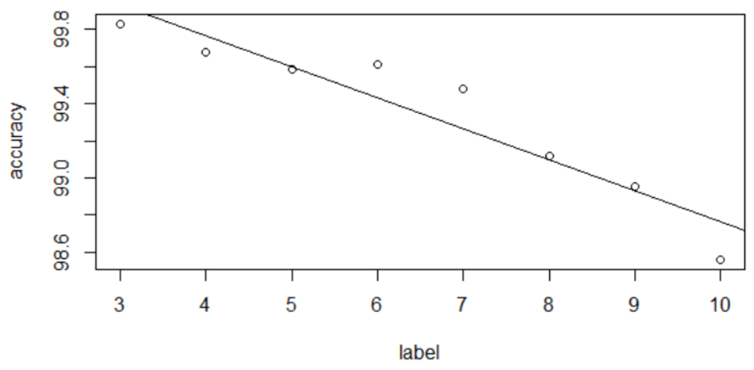


```
=== Summary ===

Correctly Classified Instances        3768               98.5613 %
Incorrectly Classified Instances        55                1.4387 %
Kappa statistic                          0.984
Mean absolute error                      0.005
Root mean squared error                  0.0501
Relative absolute error                  2.7818 %
Root relative squared error             16.6929 %
Total Number of Instances             3823
```

In general, a linear regression model is established based on the 8 pairs of values. (Much uncertainty)

Accuracy = 100.4319% - 0.1665* # of classes (not accurate)

```
lm(formula = accuracy ~ label, data = rate)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.205617 -0.095584 -0.001263  0.059160  0.209987 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 100.43190    0.16117  623.13 1.15e-15 ***
label        -0.16650    0.02339   -7.12 0.000386 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1516 on 6 degrees of freedom
Multiple R-squared:  0.8942,	Adjusted R-squared:  0.8765 
F-statistic: 50.69 on 1 and 6 DF,  p-value: 0.0003862
```



If lower-case English alphabetical characters are included in the dataset, there will be 36 classes. Then the accuracy rate may be around 94.4% for IBK (k=5) 10-fold cross-validation on the training set, reduced by nearly 4% compared to the original 10 classes.

If all ASCII characters are included in the dataset, there will be 128 classes. Then the accuracy rate may be around 79% for IBK (k=5) 10-fold cross-validation on the training set, reduced by nearly 20% compared to the original 10 classes.

If all UTF-8 characters are included in the dataset, there will be 256 classes. Then the accuracy rate may be around 58% for IBK (k=5) 10-fold cross-validation on the training set, reduced by nearly 41% compared to the original 10 classes.

I just assume the accuracy rate is linear correlated to the number of classes. So, there could be high errors based on my prediction model. Generally, as the number of classes increases, the accuracy rate declines.

8 Pre-Processing Steps

I suppose the handwritten note only has digits on it. After scanning it into a pixel-based image, it may be possible to firstly recognize the image region for each digit number, and to divide the whole image into several sub-images which include only one digit for each image (removing some extra white space if necessary). Since there could be rotations for some sub-images, multiple tries could be done to capture better results. Besides, I suppose initially it is not compulsory to restrict the size of all sub-images to be the same. Maybe other methods can be utilized to modify these images in equal size in the following step. Furthermore, it may be possible to decrease noise through some programs.

Then for each sub-image with single digit, preprocessing programs available by NIST could be used to extract normalized bitmaps (black-and-white), as the paper stated. Then 32x32 bitmaps are divided into nonoverlapping blocks of 4x4 and the number of on pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range 0 through 16. This reduces dimensionality and gives invariance to small distortions. The 64 values for each digit image should be recorded in order as one row (instance) in the dataset.

If supervised learning is needed, the class label (real digit number) of each instance (image) should also be recorded as the class attribute column in the dataset. Then a trained classifier can be applied to the dataset.