Ministry of Education and Science of Ukraine
National Technical University of Ukraine
« Igor Sikorsky Kyiv Polytechnic Institute»

№1.4

Work with WEKA. CLUSTERING, PCA AND EVALUATION

(LINK: HTTPS://DOCS.GOOGLE.COM/DOCUMENT/D/1XIXOAY6TUR3HD6BO22FWLFFT6INJ0XGR/EDIT)

The work was done by
Zvychaynaya Anastasia

The work was checked by
Alexander Oriekhov

Kyiv 2022

In this final lab we consider unsupervised learning in the form of *clustering* methods and *principal component analysis* (PCA), as well as more thorough performance evaluation of classifiers.

Execution of work:

1. CLUSTERING

We first consider clustering of the Landsat data. You will need the following dataset for this task:

- landsat.arff: landsat data

To refresh your memory about the Landsat data you can read this description. Since there are 6 classes in the data, it would be interesting to try clustering with *k=6* centres. Load the landsat data into Weka and go to the *Cluster* tab:

1. Select the *SimpleKMeans* clusterer, bring up its options window and set *numClusters* to 6.
2. In the *Cluster mode* panel, select ***Classes to clusters evaluation*** **and** hit *Start*
3. Ideally, we would hope to see all instances from a single class assigned to a single cluster, and no instances from different classes assigned to the same cluster. Look at the *Classes to Clusters* confusion matrix. Clearly, we don't have a perfect correspondence between classes and clusters, but:
    - How successful has the clustering been in this regard?

Clusterer output

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

Clustered Instances

0       798 ( 18%)
1       997 ( 22%)
2       662 ( 15%)
3       585 ( 13%)
4       383 (  9%)
5      1010 ( 23%)


Class attribute: class
Classes to Clusters:

   0   1   2   3   4   5  <-- assigned to cluster
  17  22 627 401   0   5 | 1
  12   0   1  80 383   3 | 2
  71 881   9   0   0   0 | 3
 308  80   1   7   0  19 | 4
  38   0  24  95   0 313 | 5
 352  14   0   2   0 670 | 7

Cluster 0 <-- 4
Cluster 1 <-- 3
Cluster 2 <-- 1
Cluster 3 <-- 5
Cluster 4 <-- 2
Cluster 5 <-- 7

Incorrectly clustered instances :       1471.0   33.168  %
```

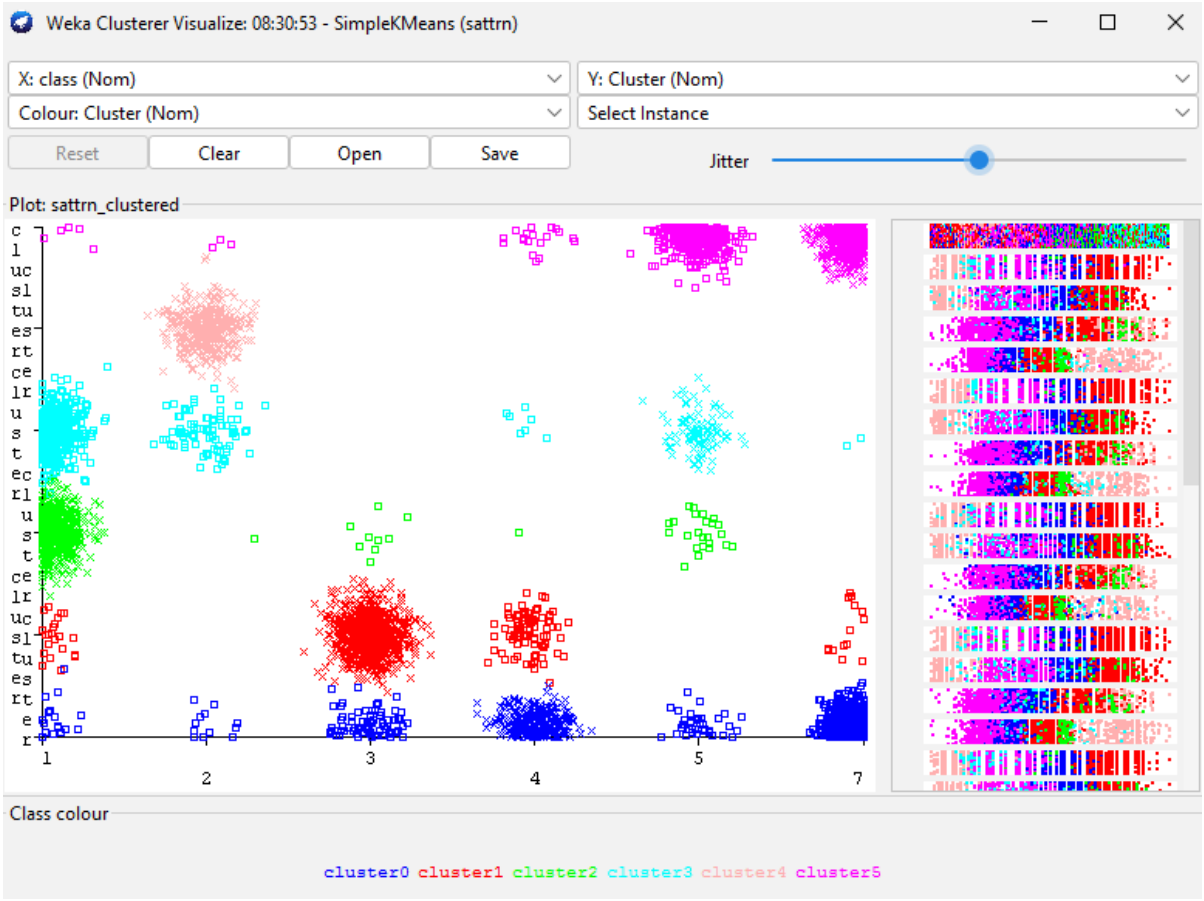
- Looking at each class individually, can you spot particular classes that are well identified by the clustering? Classes that are poorly identified?

Well identified by the clustering: 7 (not good, but better than others).

Poorly identified by the clustering: 5, 2, 3, 4, 1.

- Which classes are **mostly** confused with each other? ***5 and 7.***
- Does this relate to the observations you made in Lab 2? Have a look at the *Visualization* tab again if you need to.
  No, I do not observe it.

4. Visualize the cluster assignments. To do this, right-click on the clusterer in the *Result list* panel and select *Visualize cluster assignments*. Plot *Class* against *Cluster*. All the data points will lie on top of each other, so increase the *Jitter* slide bar to about half way to add random noise to each point. This allows us to see more clearly where the bulk of the datapoints lies. In this scatter plot each row represents a class and each column a cluster.



2. PCA

We now consider PCA. Instead of selecting a subset of attributes, PCA allows us to construct a new set of features based on a linear combination of the attributes. We will use the Landsat satellite imaging set. Acquaint yourself with this by reading this description. Load up the Landsat data

1. On the *Attribute Selection* tab, choose the *PrincipalComponents* attribute evaluator. Use the *Ranker* search method, but set the *numToSelect*option to -1. Hit *Start* and observe the output.
   - How many features does the algorithm select?

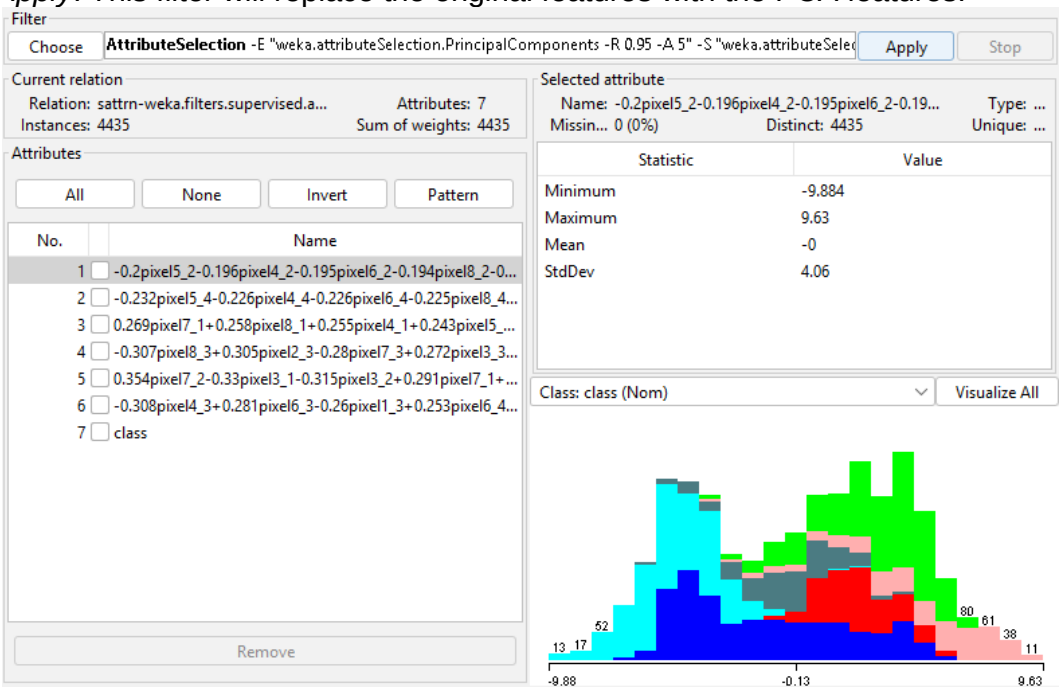
```
Ranked attributes:
 0.5878    1 -0.198pixel5_2-0.194pixel4_2-0.193pixel6_2-0.193pixel8_2-0.192pixel2_2...
 0.229     2 -0.227pixel5_4-0.222pixel4_4-0.221pixel6_4-0.22pixel8_4-0.217pixel2_4...
 0.1627    3 0.555class=1-0.263class=3-0.226class=2-0.194pixel7_1-0.188pixel8_1...
 0.1342    4 -0.707class=5+0.622class=7-0.238class=4+0.153class=2-0.08class=3...
 0.1073    5 -0.878class=4+0.336class=3+0.306class=5+0.084class=7-0.065class=2...
 0.0859    6 -0.303pixel8_3+0.301pixel2_3-0.271pixel7_3+0.27 pixel3_3+0.263pixel1_3...
 0.0696    7 -0.342pixel7_2+0.328pixel3_2+0.315pixel3_1-0.31pixel7_1+0.262pixel2_2...
 0.0548    8 0.294pixel4_3-0.284pixel6_3-0.254pixel3_3-0.254pixel3_4-0.254pixel6_4...
 0.0441    9 -0.61class=3+0.503class=7+0.484class=5-0.166class=1+0.115pixel1_1...

Selected attributes: 1,2,3,4,5,6,7,8,9 : 9
```

- On what basis does it select these? Could we select fewer? More? *Hint:* Bring up the *PrincipalComponents* options window and click on *More* to read a synopsis of the method's implementation
  Performs a principal components analysis and transformation of the data. Use in conjunction with a Ranker search. Dimensionality reduction is accomplished by choosing enough eigenvectors to account for some percentage of the variance in the original data---default 0.95 (95%). Attribute noise can be filtered by transforming to the PC space, eliminating some of the worst eigenvectors, and then transforming back to the original space.

2. On the *Result list* (at the bottom-left corner of the *Attribute Selection* tab), right-click on the PCA run and select *Visualize transformed data*.
3. Look at the scatter plots of the PCA attributes against each other.
   - Can you make any salient observations?
     Yes, I see that the features have become more independent, I do not observe correlations.
   - Do you think that the classes are more separable with the PCA features than with the original set of attributes?
     Of course, because this is the hallmark of the PCA method.

   We can consider training a classifier with the PCA feature representation. Do you expect the resulting model to achieve better performance than when trained on the original set of features?
   Yes, this may be because the PCA method is often used to avoid the curse of dimensionality; however, the power of the original feature set is not so great.
4. If you have time, go to the *Preprocess* tab and select *Choose* > *filters* > *supervised* > *attribute* > *AttributeSelection*. Bring up the *AttributeSelection* options window and set the *evaluator* field to *PrincipalComponents* and the *search* field to *Ranker*. Click *OK* and then *Apply*. This filter will replace the original features with the PCA features.



5. Train a Naive Bayes model using the PCA representation and 5-fold cross-validation. Then reload the Landsat data and train a Naive Bayes model using the original representation (again use 5-fold CV). Compare the performance of the resulting models. Was this your guess in question 4? ***YES, we have reduced the chance of dimension curse.***

Naive Bayes model using the PCA representation:

Classifier output

```
Correctly Classified Instances        3706               83.5626 %
Incorrectly Classified Instances       729               16.4374 %
Kappa statistic                          0.796
Mean absolute error                      0.0628
Root mean squared error                  0.1995
Relative absolute error                 23.292  %
Root relative squared error             54.363  %
Total Number of Instances             4435

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,965    0,013    0,960      0,965    0,963      0,951    0,996     0,992     1
                 0,967    0,005    0,963      0,967    0,965      0,960    0,999     0,993     2
                 0,907    0,038    0,869      0,907    0,888      0,856    0,984     0,949     3
                 0,402    0,044    0,487      0,402    0,441      0,391    0,910     0,439     4
                 0,681    0,033    0,708      0,681    0,694      0,659    0,960     0,751     5
                 0,818    0,067    0,788      0,818    0,802      0,741    0,967     0,909     7
Weighted Avg.    0,836    0,035    0,829      0,836    0,832      0,799    0,975     0,886

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 1035    2   13    0   22    0 |    a = 1
    0  463    1    1   14    0 |    b = 2
   13    5  872   44   25    2 |    c = 3
    3    4   85  167   25  131 |    d = 4
   27    6    0   21  320   96 |    e = 5
    0    1   32  110   46  849 |    f = 7
```

Naive Bayes model using the original representation:

Classifier output

```
Correctly Classified Instances        3525               79.4814 %
Incorrectly Classified Instances       910               20.5186 %
Kappa statistic                          0.7485
Mean absolute error                      0.0685
Root mean squared error                  0.2559
Relative absolute error                 25.4218 %
Root relative squared error             69.7181 %
Total Number of Instances             4435

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,789    0,029    0,897      0,789    0,840      0,796    0,971     0,925     1
                 0,894    0,001    0,991      0,894    0,940      0,934    0,996     0,981     2
                 0,891    0,031    0,889      0,891    0,890      0,859    0,982     0,925     3
                 0,631    0,073    0,470      0,631    0,539      0,490    0,901     0,456     4
                 0,738    0,070    0,557      0,738    0,635      0,592    0,929     0,725     5
                 0,757    0,039    0,857      0,757    0,804      0,751    0,956     0,877     7
Weighted Avg.    0,795    0,037    0,820      0,795    0,803      0,764    0,962     0,855

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 846   1  36   1 188   0 |   a = 1
  18 428   0   3  28   2 |   b = 2
  17   0 856  82   2   4 |   c = 3
   8   0  63 262  10  72 |   d = 4
  54   3   0  13 347  53 |   e = 5
   0   0   8 196  48 786 |   f = 7
```

3. PERFORMANCE ASSESSMENT #2

We will continue our performance assessment on the Splice data set which was originally introduced in Lab 3. As a reminder: the classification task is to identify *intron* and *exon* boundaries on gene sequences. Read the description at the link above for a brief overview of how this works. The class attribute can take on 3 values: N, IE and EI. Now download the data sets below, converted into ARFF for you, and load the training set into Weka:

- splice_train.arff: training data
- splice_test.arff: test data

1. Set *KNN* to the value that provides the greatest percent of correctly classified instances in Lab 3, Question 5 and re-run the classifier on the held out test dataset *splice_test.arff*.

   K=130:

Classifier output

```
Time taken to build model: 0.01 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.58 seconds

=== Summary ===

Correctly Classified Instances         233               91.3725 %
Incorrectly Classified Instances        22                8.6275 %
Kappa statistic                          0.8567
Mean absolute error                      0.2861
Root mean squared error                  0.3213
Relative absolute error                 71.6471 %
Root relative squared error             73.0859 %
Total Number of Instances              255

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,866    0,000    1,000      0,866    0,928      0,853    0,997     0,998     N
                 1,000    0,089    0,743      1,000    0,852      0,823    0,992     0,970     EI
                 0,963    0,020    0,929      0,963    0,945      0,931    0,997     0,991     IE
Weighted Avg.    0,914    0,022    0,932      0,914    0,916      0,864    0,996     0,991

=== Confusion Matrix ===

   a   b   c   <-- classified as
 129  16   4 |   a = N
   0  52   0 |   b = EI
   0   2  52 |   c = IE
```

2. We wish to understand the output from WEKA with respect to TP rate, FP rate, Precision, Recall etc. Note that these are output on a per-class basis. As the Splice data is a three-class problem, we can consider each of the classes (N, EI, IE) as the "positive" class, and lump the remaining two together as the negative class. Thus to compute TP rate etc we need to reduce the 3 x 3 confusion matrix to a 2 x 2 confusion matrix. So if the positive class is EI, this will mean lumping together the results for the N and IE classes.

Reduce the 3 x 3 confusion matrix to a 2 x 2 confusion matrix using EI as the positive class.

$$\begin{matrix} \mathbf{52} & \mathbf{0} \\ \mathbf{16+2} & \mathbf{129+4+52} \end{matrix} = \begin{matrix} \mathbf{52} & \mathbf{0} \\ \mathbf{18} & \mathbf{185} \end{matrix}$$

To check your reasoning, compute the TPR = TP/(TP+FN), FPR = FP/(FP+TN) and Precision = TP/(TP+FP) and check them against the results output by WEKA.

$$\textbf{\textit{TPR(recall)}} = \frac{52}{52+0} = 1 \textbf{\textit{ (all EI samples are classified correctly);}}$$

$$\textbf{\textit{FPR(recall)}} = \frac{18}{18+185} = \frac{18}{203} = 0.089.$$

Is the proportion of points correctly classified a sufficient measure of performance? *Hint:* Consider the other data sets that we've looked at. What does a false positive mean in identifying spam? Is this as bad as a false negative?

It depends on our goal. If we want to filter non-EI samples, then everything is fine because on EI our algorithm will not say that it is not EI, but we will, of course, lose 18 non-EI samples (compare with 185 defined correctly).