

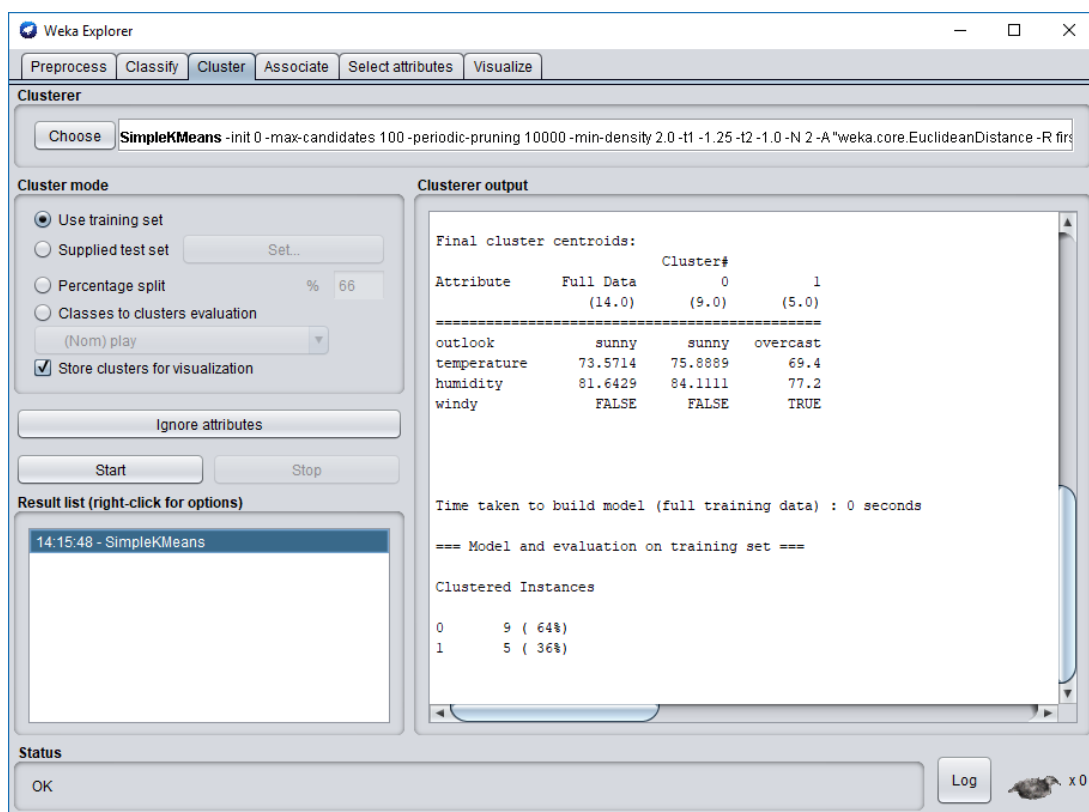
Lab Exercise 7

Data Clustering

In this lab session we continue exploring Weka's implementations. We are going to use k-means. Broadly speaking, k-means clustering splits n observations into k clusters in which each observation belongs to the cluster with the nearest mean. The mean is used as a prototype of the cluster.

Exercise 1

1. **Clustering** - Open Weka Explorer environment and load the training file using the **Preprocess mode**. Try first with **weather.arff**. Get to the Cluster mode (by clicking on the Cluster tab) and select a clustering algorithm, for example **SimpleKMeans**. Then click on **Start** and you get the clustering result in the output window. The actual clustering for this algorithm is shown as one instance for each cluster representing the **cluster centroid**.



The screenshot shows the Weka Explorer interface with the 'Cluster' tab selected. The 'Clusterer' dropdown is set to 'SimpleKMeans'. The 'Cluster mode' panel has 'Use training set' selected. The 'Clusterer output' window displays the following results:

```
Final cluster centroids:
Attribute      Full Data      Cluster#
              (14.0)        (9.0)         (5.0)
-----
outlook        sunny          sunny          overcast
temperature    73.5714        75.8889        69.4
humidity       81.6429        84.1111        77.2
windy          FALSE          FALSE          TRUE

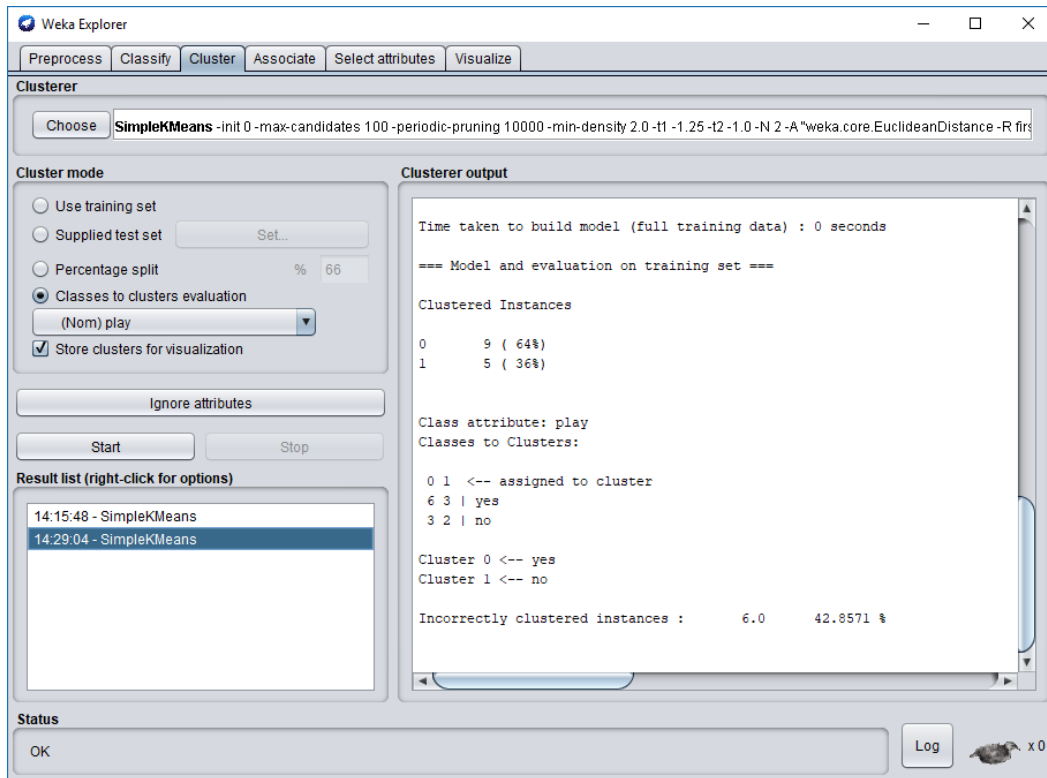
Time taken to build model (full training data) : 0 seconds

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

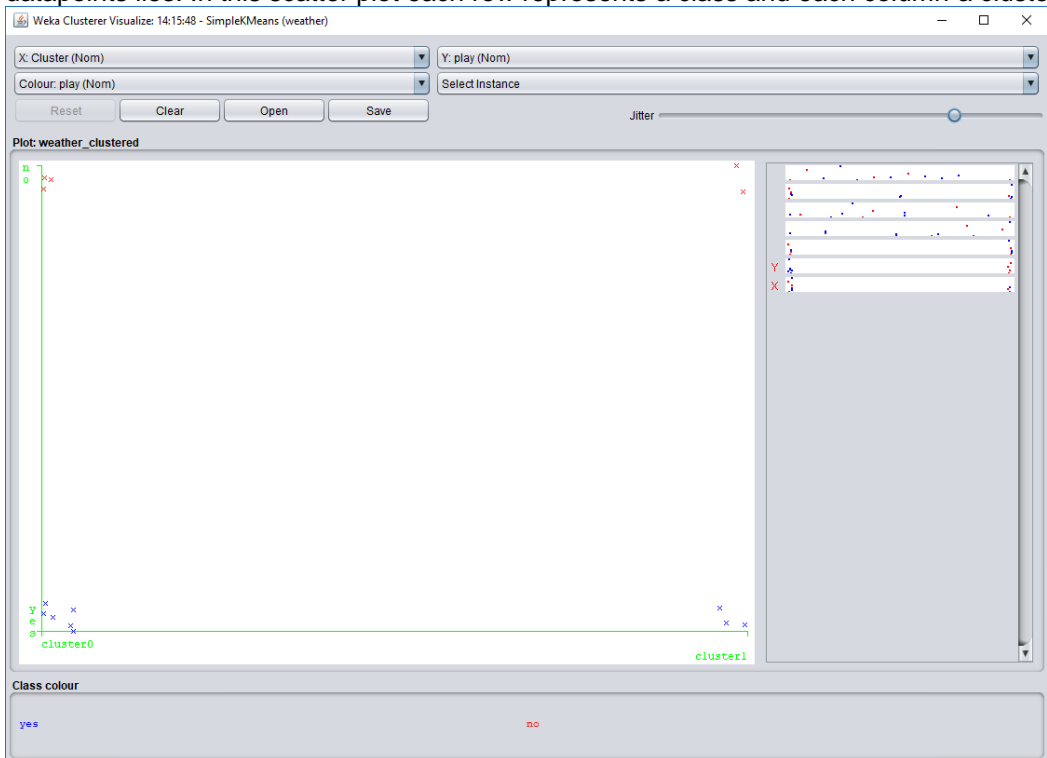
Clustered Instances

0      9 ( 64%)
1      5 ( 36%)
```

2. **Evaluation** - The way Weka evaluates the clusterings depends on the cluster mode you select. Four different cluster modes are available (as buttons in the Cluster mode panel):
 - a. **Use training set (default)**. After generating the clustering Weka classifies the training instances into clusters according to the cluster representation and computes the percentage of instances falling in each cluster. For example, the above clustering produced by k-means shows 64% (9 instances) in cluster 0 and 36% (5 instances) in cluster 1.
 - b. **In Supplied test set or Percentage split** Weka can evaluate clusterings on separate test data if the cluster representation is probabilistic (e.g. for EM).
 - c. **Classes to clusters evaluation**. In this mode Weka first ignores the class attribute and generates the clustering. Then during the test phase it assigns classes to the clusters, based on the majority value of the class attribute within each cluster. Then it computes the classification error, based on this assignment and also shows the corresponding confusion matrix. An example of this for k-means is shown below.



3. **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.



4. You could save the clustering results by clicking Save button on the Visualization panel. The results are saved in a .arff file. You could use Weka to open it and view the results.

Exercise 2

1. We are familiar with the **spam dataset** by now. To load the spam dataset (available from Lab 4) select the preprocess tab and then select **Open file**
2. Go to the **Cluster** tab: Select the **SimpleKMeans** clusterer, bring up its options window and set numClusters to 2.
3. In the **Cluster** mode panel, select **Classes to clusters evaluation** and hit Start. This option evaluates clusters with respect to a class. More specifically, in the mode *Classes to clusters evaluation* Weka first ignores the class attribute and generates the clustering. Then during the test phase it assigns classes to the clusters, based on the majority value of the class attribute within each cluster. Then it computes the classification error, based on this assignment and also shows the corresponding confusion matrix.
4. 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.
5. Look at the **Classes to Clusters confusion matrix**. Clearly, we don't have a perfect correspondence between classes and clusters.
 - a. How successful has the clustering been in this regard?
 - b. Looking at each class individually, can you spot the particular class that is well identified by the clustering? Classes that are poorly identified?
 - c. Which classes are mostly confused with each other?
 - d. Compare with the performance of the supervised classifiers we used during the last lab session.

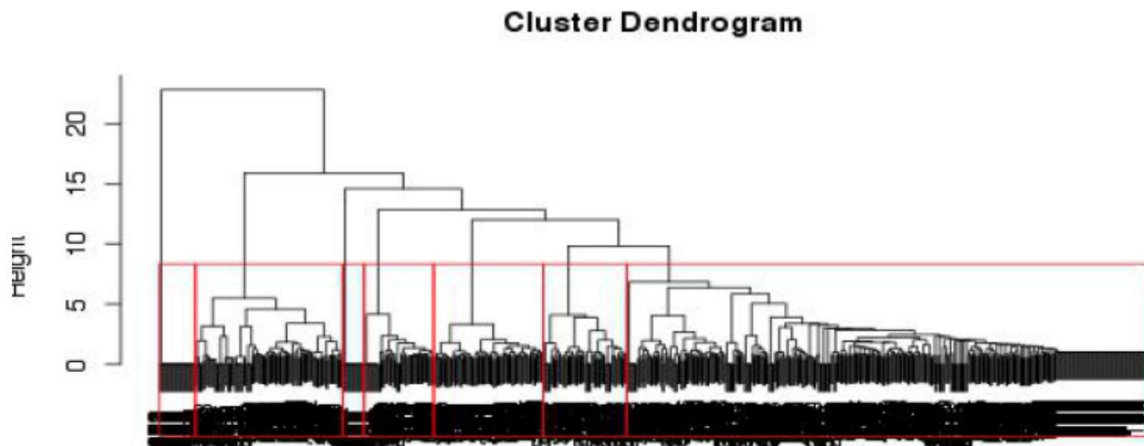
Visualize the cluster assignments. To do this, right-click on the clusterer in the **Result** list panel and select **Visualize cluster assignments**. Plot Class against Cluster. All the data points will lie on top of each other, so increase the Jitter slide bar to about half way to add random noise 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.

6. Can you draw any conclusion based on this visualization?

Exercise 3

1. **HierarchicalClusterer** implements agglomerative (bottom-up) generation of hierarchical clusters. Several different link types, which are ways of measuring the distance between clusters, are available as options.
2. Since the Hierarchical Clustering algorithm builds a tree for the whole dataset, let's practice this algorithm on a smaller dataset due to the memory space limitation. Open **glass.arff** dataset used in the previous lab, first normalize all the numeric values in the dataset into [0,1]. Then chose **HierarchicalCluster** cluster. Since this dataset has 6 classes, we set numClusters as 6. To save time, we set printNewick as False. Choose **link_type=COMPLETE** and run the algorithm.
3. Since the performance of **HierarchicalClustering** is not good, we could run **k-means** algorithm on the same dataset and compare their performance. Do not forget to set the number of clusters to 6.
4. Save the clustering results as **glass_kmeans_result.arff** file, ie. right-click on the k-means entry, choose Visualize cluster assignment, then save with the name above. Reopen this dataset with Weka. Since clustering results are saved as the last column, they are considered as class labels for the dataset. The original class labels are considered as a feature of the dataset. If you trust the clustering you might want to use this newly created dataset for supervised learning.

Remember: Deciding the number of clusters when you do not have any knowledge or intuition of the patterns in your data is quite difficult. One rule of thumb: look at the dendrogram, where height change looks big, cut off the tree. E.g.



5. Run **HierarchicalClusterer** on the original dataset, this time choose **Use training set as Cluster** mode. Visualize the dendrogram.
 - Where would you cut this dendrogram?
 - How many clusters would you create for this dataset, if you did not new in advance the number of classes?

Tasks to complete (Assignment 2)

1. Download the Pima Indians collection (included in the WEKA package).
2. Using text editor split the data set into 60/40 subsets. The 60% subset will be used for clustering and validation of clusters while 40% dataset will be used for testing and prediction. Ideally write the script (e.g. in python to randomise data before splitting)
3. Prepare the 60% data set to be loaded into WEKA.
4. Launch WEKA and cluster datasets using clustering algorithms of your choice. You should at least choose the Hierarchical and k-means algorithm, but it would be nice if you include other algorithms too for comparison. Make sure you try various data pre-processing such as to convert to nominal, numeric, or to normalise data etc. You can play with various input dataset configurations. You should also try various settings such as *link_type* or *distance function*. Evaluate your clusters using **Classes to Clusters confusion matrix** to compare the performance with the annotated classes.
5. Save your results and use new clusters (instead of original classes this time!) to build new classification models based on them. Use classification method of your choice. Test them on the 40% test dataset and compare obtained results with original class annotations. Are they better or worst!? For each clustering algorithm you should have at least one cluster set.

In particular, the report should discuss the impact of new clusters (used as classes in the training set) on the learning and prediction process on a unknown dataset (40% test dataset) and which clustering model was better at discovering the "true" classes?
6. Discuss the results as well as present the conclusion and the summary of the experiment in the report

The report should be sent by email in **ONE** pdf file. When naming the file please use the following naming convention: **DM_LAB7_name_surname.pdf**. An email with the file should be sent to the email address of the lecturer and titled: **DM_LAB7_name_surname**