Hong Kong Baptist University

Department of Computer Science

COMP 7990 Principles and Practices of data analytics (2022-23)

Lab 1: Data Mining using Weka

Introduction

Data mining is a process to discover patterns and rules from a large data set. It is an important tool in many areas of research and industry. The ultimate goal of data mining is to create a model that can convert data into information and make prediction.

What are the differences between data mining and machine learning? *Data mining* is the application and *machine learning* is the algorithms we use. We can use machine learning algorithms for the purpose of data mining. We can design algorithms that can self-learn from data to gain knowledge, and to apply the learned knowledge on new (unseen) data. There are some applications benefit from ML, such as language translation and chatbots. And there are two main categories of machine learning: **supervised learning** and **unsupervised learning**.

For **Supervised learning,** we can train a ML model using labeled data, the model learns the relationship between the attributes of the data and its outcome, and finally we make prediction on the new data for which the label is unknown. **Unsupervised learning** builds model from data without a predefined outcome or label, it aims at extracting structure or pattern from the data, data instances are grouped together by similarity.

A) Supervised Learning:

1. **Regression**: it is able to predict <u>a **continuous** value</u> for an instance. E.g., stock price, housing price and temperature. *Linear regression* is one of the algorithms

2. **Classification**: it is able to predict and classify each instance into a set of predefined <u>**discrete** values.</u> *Support Vector Machine (SVM), k-nearest neighbors (KNN), Neural networks* are common algorithms

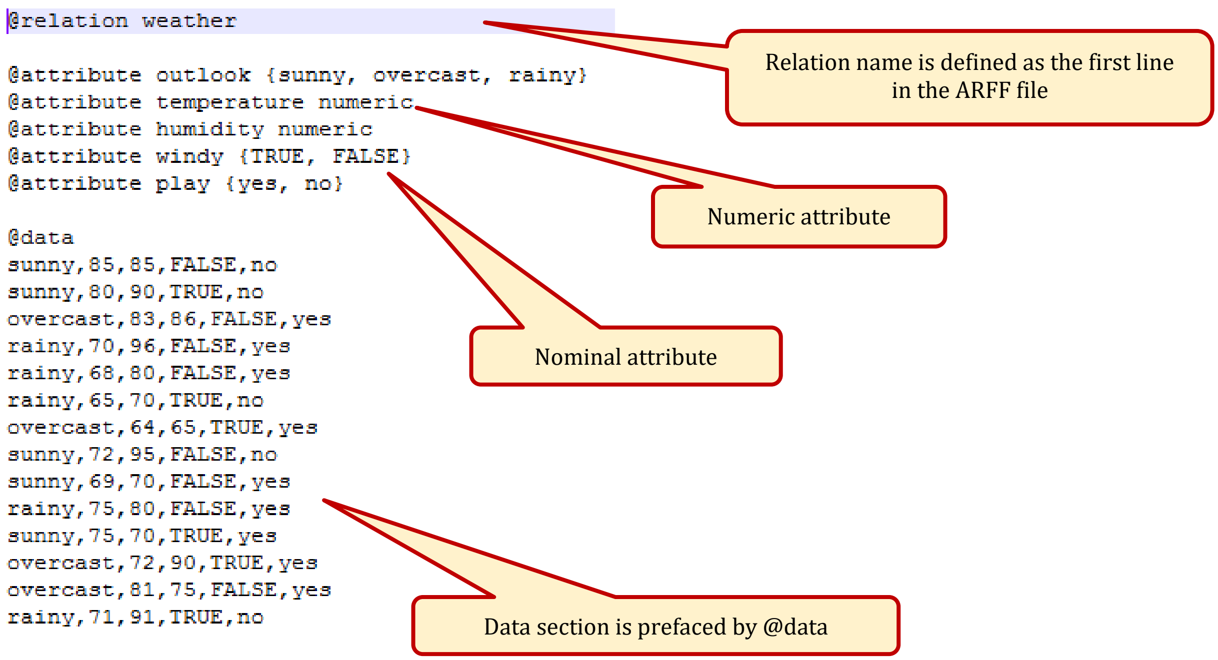
B) Unsupervised Learning:

1. **Clustering**: It groups the data instances together based on similarity, there is no predefined outcome. Example of algorithms: *KMeans.*

What is a Weka?

Waikato Environment for Knowledge Analysis (WEKA) is the product of the University of Waikato (New Zealand) and was first implemented in 1997. It is a *free* and *open source* data mining workbench which provides machine learning algorithms for data mining tasks (with 100+ algorithms for classification, 75 for data processing, 20 algorithms for clustering etc). Weka is issued under the *GNU General Public License*. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

Weka is written in the Java™ language and can run on any computer. It accepts data in **.arff** file format (Attribute-Relation File Format) and it provides a handy tool to load CSV files and save them in ARFF. In Weka, a row of data is called an **instance**, or **observation**. A column of data is called a **feature** or **attribute**.



Download the program

https://www.cs.waikato.ac.nz/~ml/weka/downloading.html (Latest stable version: 3.8.5)

Source files:

Many data sources are found in C:\Program Files\ Weka-3-8-5\data, you can open them to perform data mining tasks in this tutorial. Download the file **Lab1-Weka.zip** from BUelearning. **Unzip** it and place the them on Desktop.

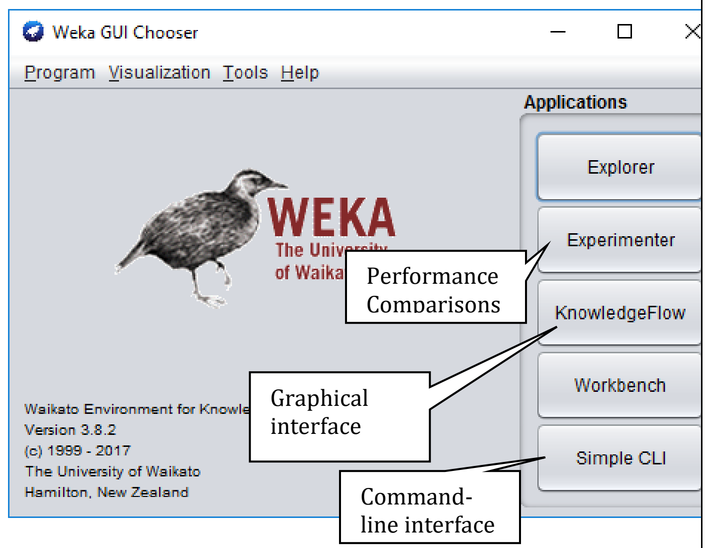
Learning Outcome

1. Introduce data mining and machine learning
2. Exploring the Explorer in Weka
3. How to use Weka for Classification?
4. How to use Weka for Clustering?

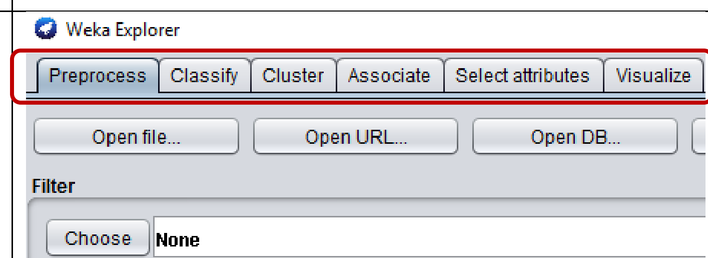
Exploring the Explorer in Weka

1. Open the **Weka** program

2. Select **Explorer** (This option is more than sufficient for everything we need)

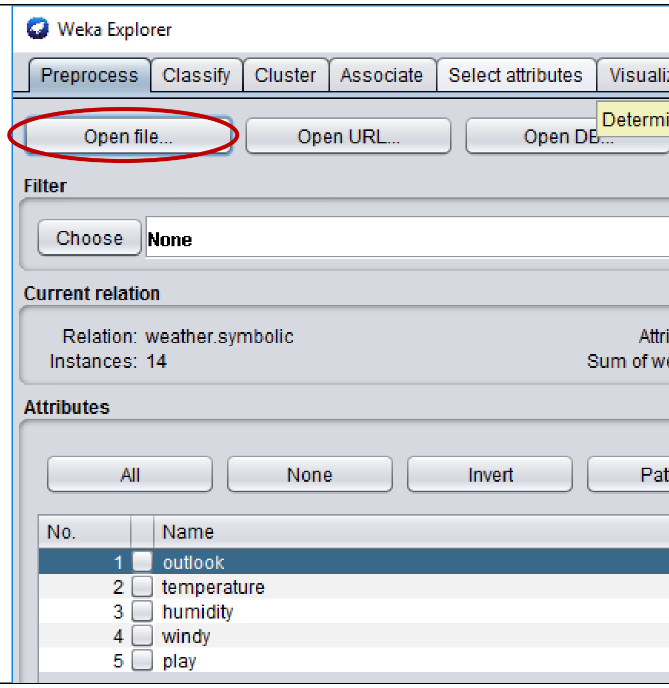


3. There are six panels at the top: **Preprocess, Classify, Cluster, Associate, Select attributes** and **Visualize**

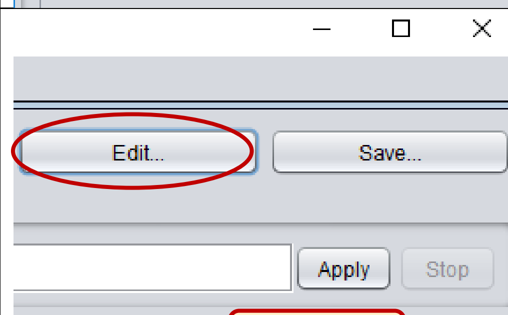


- **Preprocess**: Choose the dataset and modify it in various ways
- **Classify**: Train learning schemes that perform classification or regression and evaluate them
- **Cluster**: Learn clusters for the dataset
- **Associate**: Learn association rules for the data and evaluate them
- **Select attributes**: Select the most relevant aspects in the dataset
- **Visualize**: View different two-dimensional plots of the data and interact with them

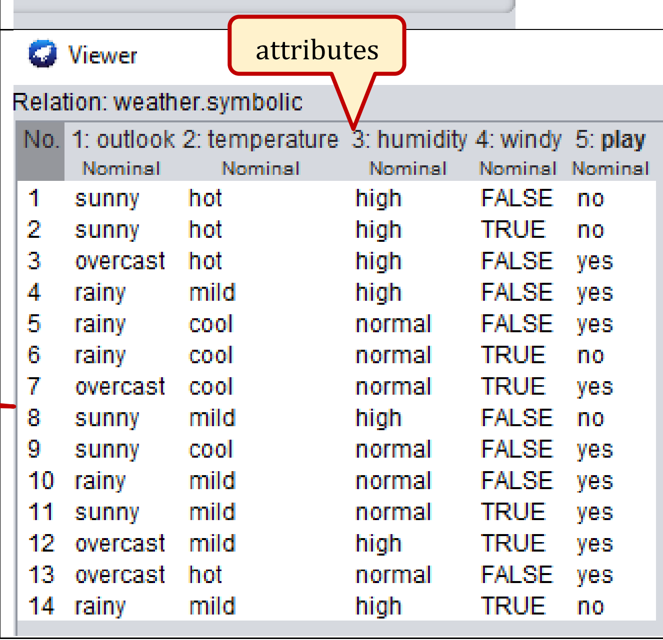
4. Select **Preprocess** panel. Choose **Open file** to open the file **weather.nominal.arff**
   There are **14 instances**, **5 attributes**, the last attribute *play* is a target variable/class attribute. There are **2 possible values** for this class attribute. [yes/no](specify whether or not we are going to play a particular game)



5. Select **Edit** button. (at the top right corner)

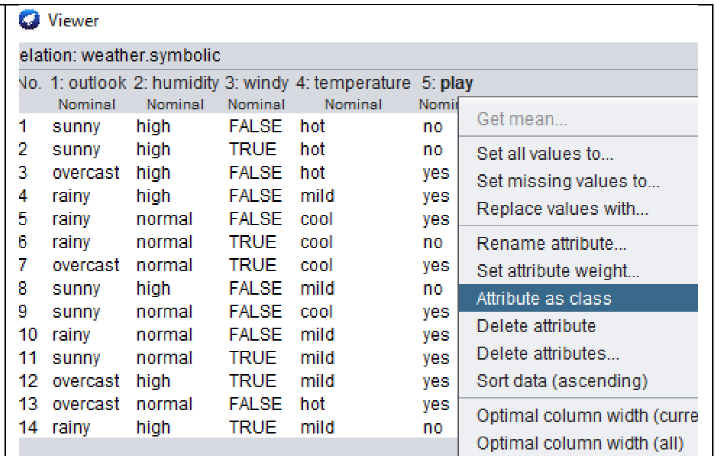


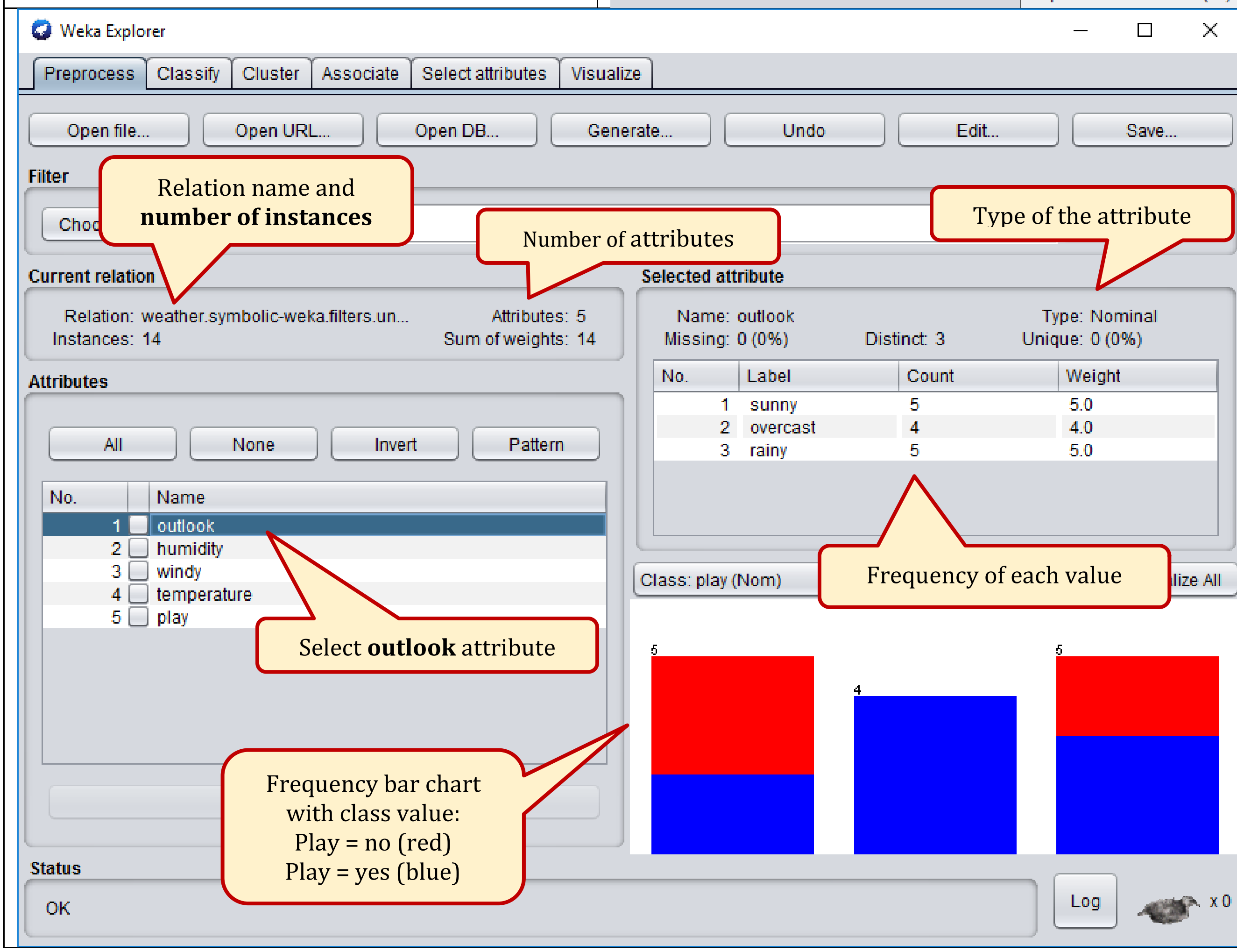
6. You can change the values, delete or add instances here. (But you need to save the data set by clicking **OK**, then **Save** button) [But now, you do not need to make any change!]



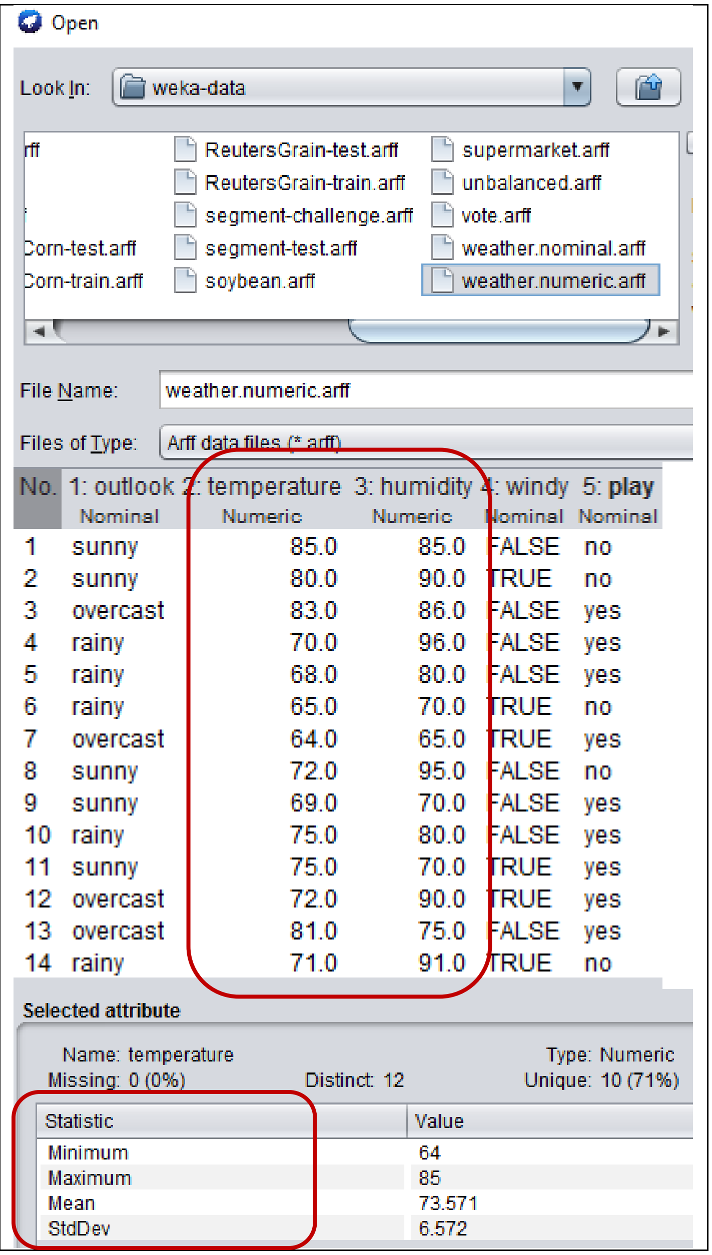
No.	1: outlook (Nominal)	2: temperature (Nominal)	3: humidity (Nominal)	4: windy (Nominal)	5: play (Nominal)
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

7. You can **right click** a certain attribute and assign it as **class attribute.**





8. Open the file **weather.numeric.arff**
   The attributes ***temperature*** or ***humidity*** contain numeric values, and there are some statistics for them. (**Min, Max, Mean**, and **standard deviation** are shown)

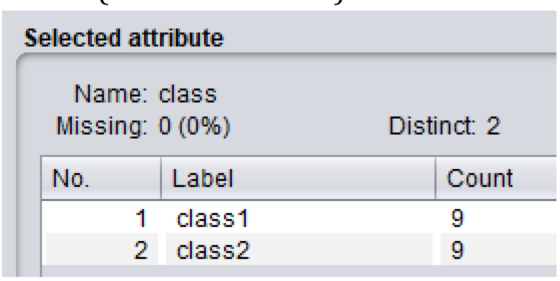


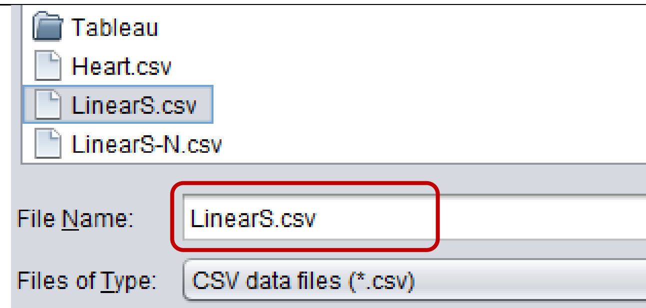
How to use Weka for Classification (Support Vector Machine)?

Classification allows us to predict and classify each instance into a set of predefined **discrete** values

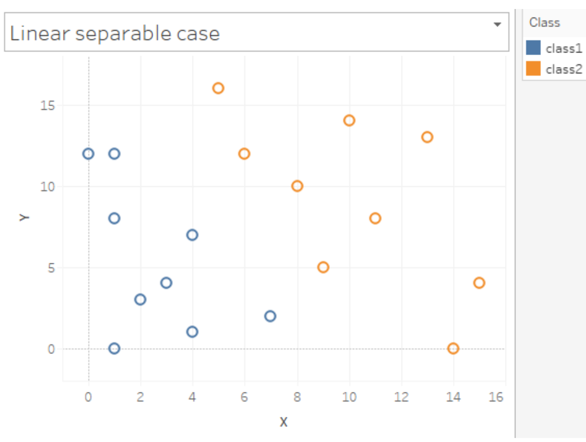
A **support vector machine (SVM)** is a *supervised* learning algorithm. It supports *classification* and *regression* problems. It works by finding a hyperplane with maximum margin. It is mostly developed for binary classification problems.

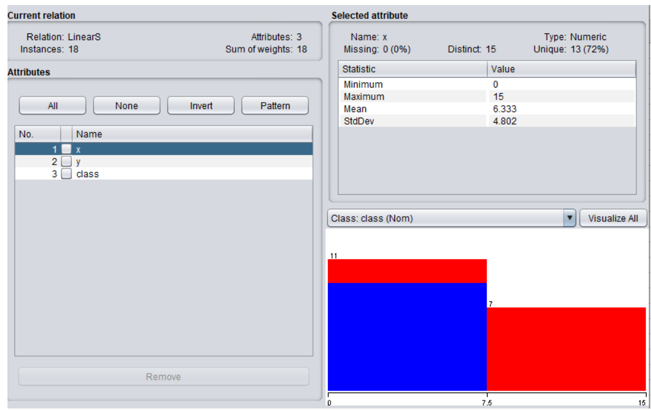
1. Open the file **LinearS.csv**
2. **Target variable** is **class** with 2 possible values (**class1** or **class2**).



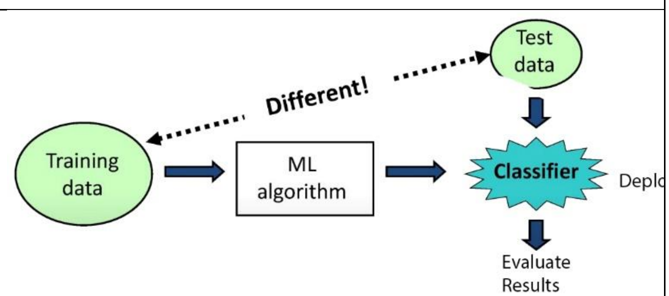


3. **Save** the file as **LinearS.arff**





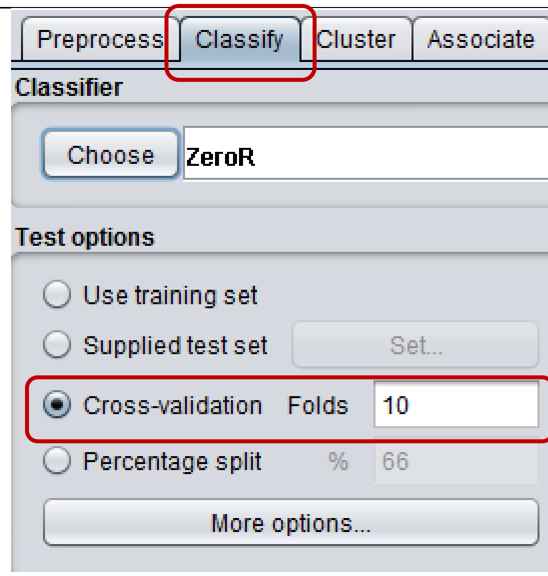
4. How to build classifier?
   a) Feed in training data to produce a classifier
   b) Test the classifier by using test data
   c) Evaluate the results
   d) Deploy the classifier to make prediction

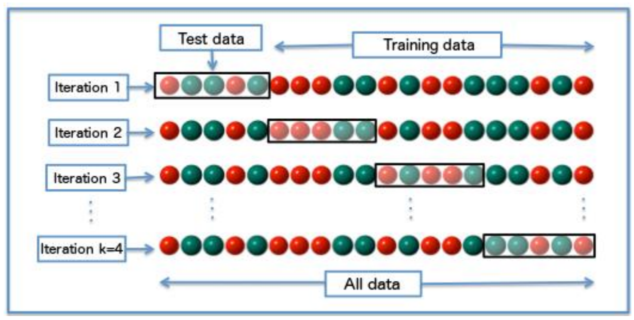


5. Select **Classify** panel. There are several ways to evaluate the classifier. Keep the setting **Cross-validation Folds 10** under **Test options**.
   [*Cross-validation folds 10* divides a dataset into 10 pieces ("folds"), then hold out each piece in turn for testing and train on the remaining 9 pieces. This gives 10 evaluation results, which are averaged.]

   Supplied test set, allows you to supply an independent test set.
   Percentage split allows you to hold out a certain percentage of the data for testing
   Use training set means using all instances to build the model, and test the model

6. Click **Choose** button.



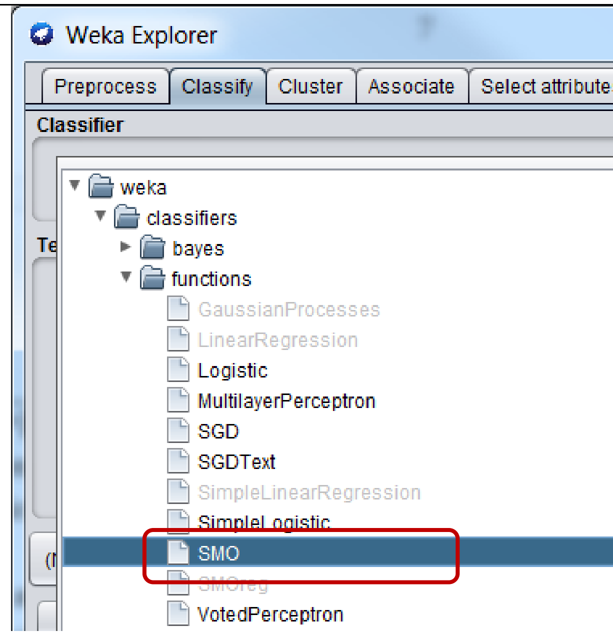


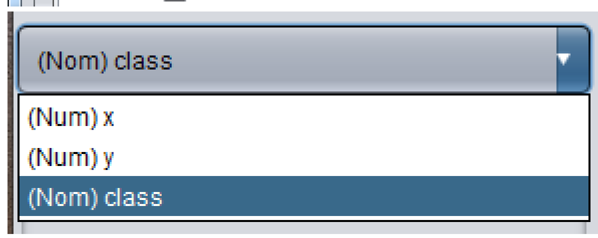
<table>
<tr>
<td>

7. Select **functions → SMO** (Sequential Minimal Optimization/SMO is Weka's implementation of the SVM algorithm) (for binary classification)

8. Select the last attribute as dependent variable (the column to be predicted) [by default **(Nom) class** is selected]

</td>
<td>

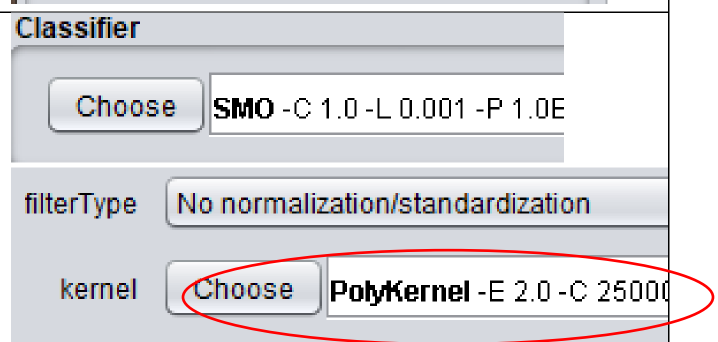




</td>
</tr>
<tr>
<td>

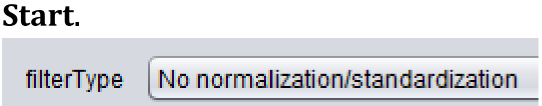
9. Click the classifier **SMO** to review the algorithm configuration. The default kernel is **PolyKernel**. That is polynomial kernel. It will separate the classes using a curved or wiggly line, the higher the polynomial, the more wiggly the line.

</td>
<td>

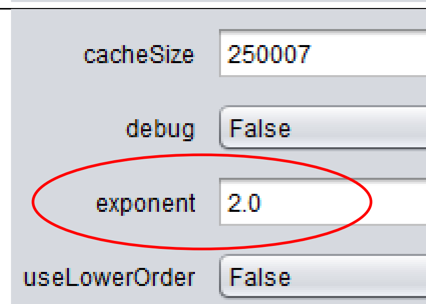


</td>
</tr>
<tr>
<td>

10. Click on the kernel **PolyKernal** and set **exponent** to **2**. Press **OK**.
11. Change the filterType to **No normalization/standardization** (to easily translate the results). Press **OK** and then **Start**.

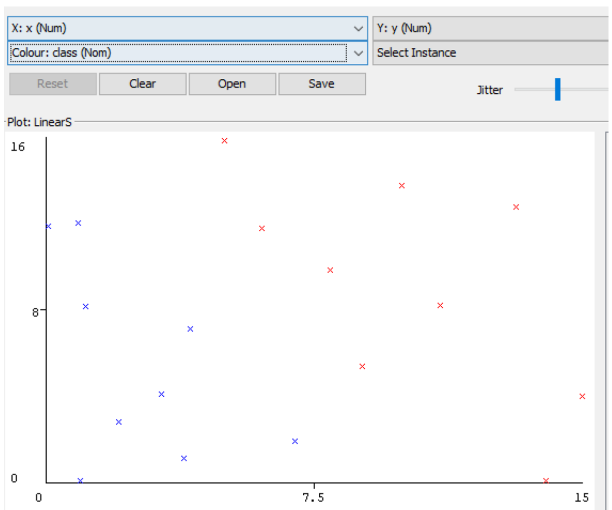


</td>
<td>

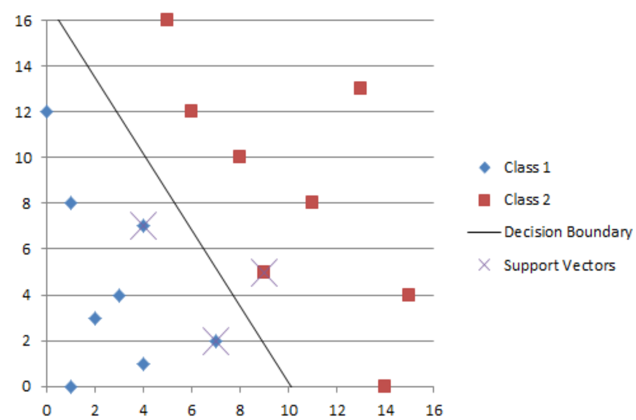


</td>
</tr>
</table>

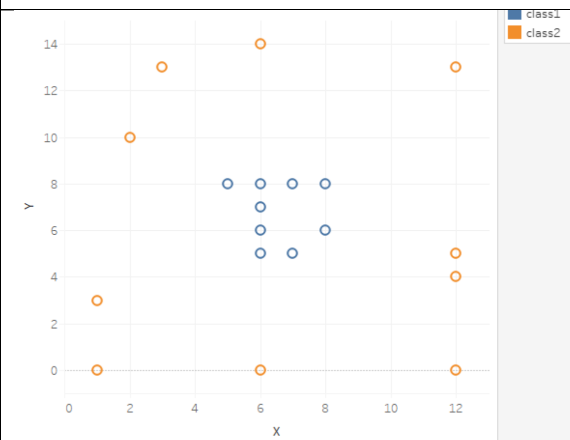
12. There are 3 support vectors. You may use **Visualize** tab to display the instances.



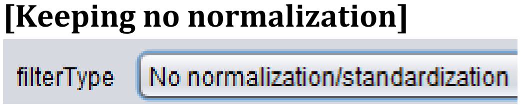

```
Kernel used:
  Poly Kernel: K(x,y) = <x,y>^2.0

Classifier for classes: class1, class2

BinarySMO

-        0.0005 * <7 2 > * X]
+        0.0006 * <9 5 > * X]
-        0.0001 * <4 7 > * X]
-        2.7035

Number of support vectors: 3
```



13. Open the file **NonlinearS.csv.** Here is the distribution of the instances.



14. Click the classifier **SMO** to review the algorithm configuration. Change the Kernel to **RBFKernel (Radial Basic Function) [Keeping no normalization]**

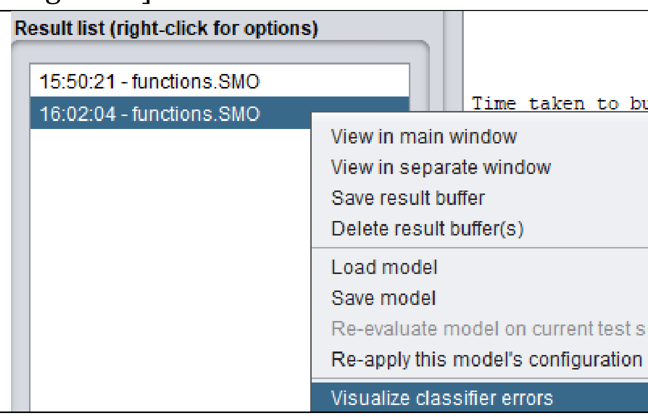


RBFKernel is a nonlinear kernel that makes linear models work in nonlinear settings
Press **OK** and then **Start**.

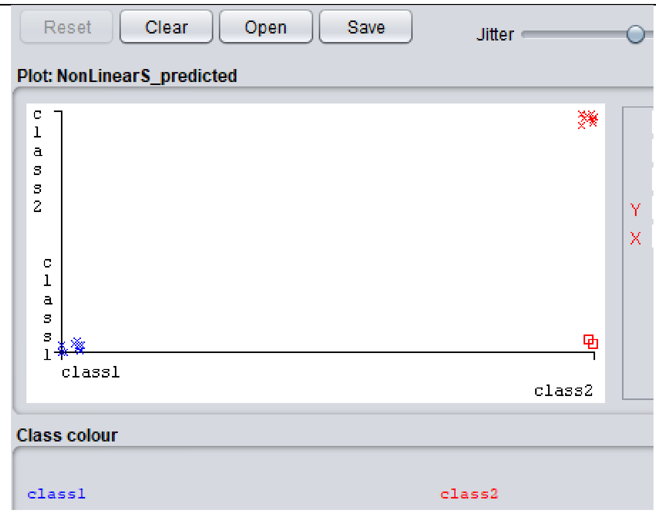

```
Kernel used:
  RBF Kernel: K(x,y) = exp(-0.01*(x-y)^2)
```

15. Correctly Classified Instances is **90%. Correctly Classified Instances = 18**	`Correctly Classified Instances 18 90 %`
	`Incorrectly Classified Instances 2 10 %`
	`Kappa statistic 0.802`
	`Mean absolute error 0.1`
	`Root mean squared error 0.3162`
	`Relative absolute error 20 %`
	`Root relative squared error 62.9629 %`
	`Total Number of Instances 20`

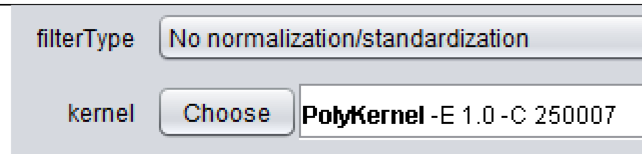
16. The prediction results can be summarized in the **confusion matrix**. One axis of a confusion matrix is the label that the model predicted, and the other axis is the actual label.	`=== Confusion Matrix ===`
	
	`a b <-- classified as`
	`9 0 | a = class1`
	`2 9 | b = class2`
	
	Top left, 9, are things your model thinks are "a" which really are "a" [true positives]
	Bottom left, 2, are things your model predicts are "a" but which are really "b" (error) [false positives]
	Top right, 0, are things that predicted as b but actutally a (another error) [false negatives]
	Bottom right, 9, are things that your model thinks are "b" which really are "b" [true negatives]

17. You can show the classifier errors on a two-dimensional plot. **Right click** the classifier in the **Result list**, select **Visualize classifier errors**. Here you can see which instances are misclassified.	

18. Adjust the **jitter** by moving the slider to the right.
19. **Right click** on the **red rectangles** near the bottom right hand corner. 2 instances are misclassified.



20. **Click the classifier SMO** to review the algorithm configuration. Change the kernel to **PolyKernel**. (No normalization). Click **OK** and **Start**.

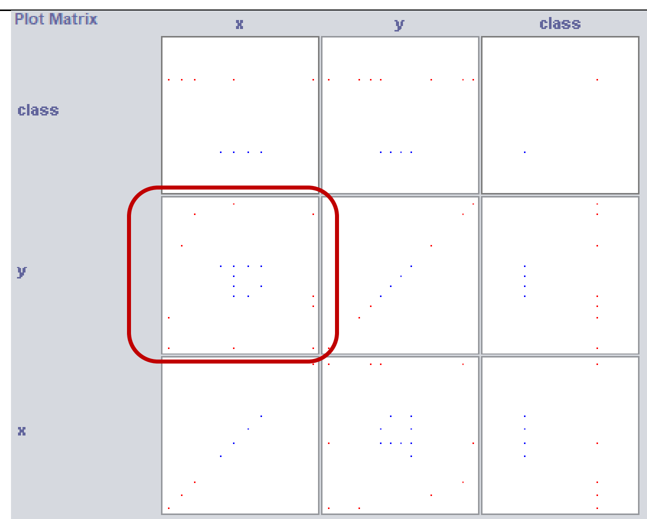


21. It is a linear Kernel. **Correctly Classified Instances = 40%.** So, using **RBF Kernel is better for non-linear separable case**.

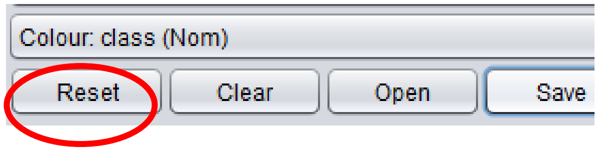
```
Kernel used:
  Linear Kernel: K(x,y) = <x,y>
```

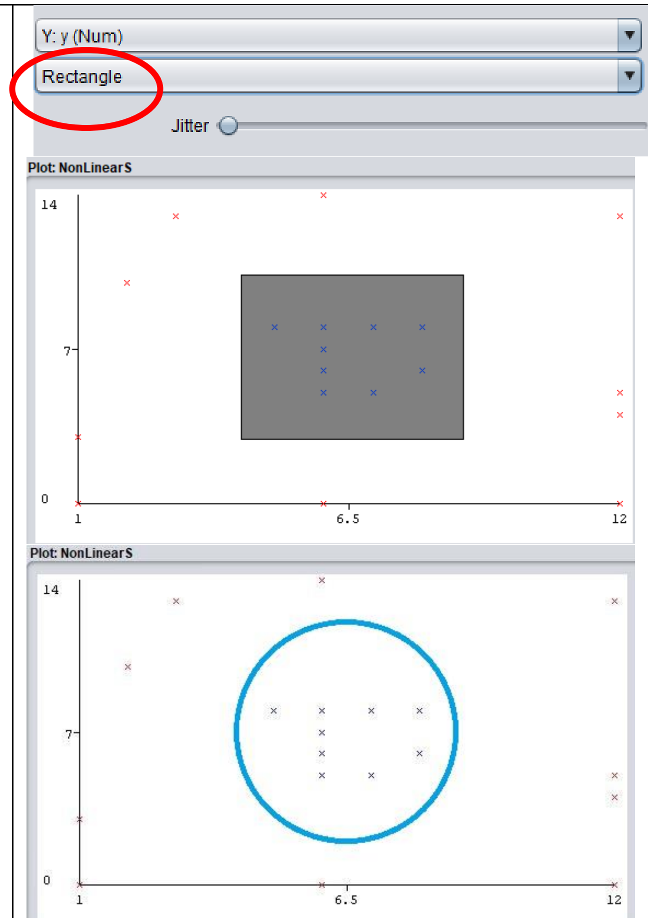
```
Correctly Classified Instances           8               40      %
Incorrectly Classified Instances        12               60      %
Kappa statistic                         -0.2371
Mean absolute error                      0.6
Root mean squared error                  0.7746
Relative absolute error                120      %
Root relative squared error            154.2269 %
Total Number of Instances               20
```

22. Select **Visualize** tab, try to visualize the relationship between x & y by clicking the matrix shown here. You may change the color of the classes.

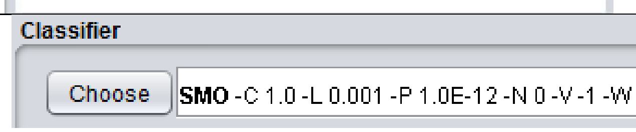


23. Select "**Rectangle**" from the "**Select Instance**" menu. Then select few instances to **Submit** and **Save** (as another arff file)
24. Then reset the settings and close the **Weka Explorer**.





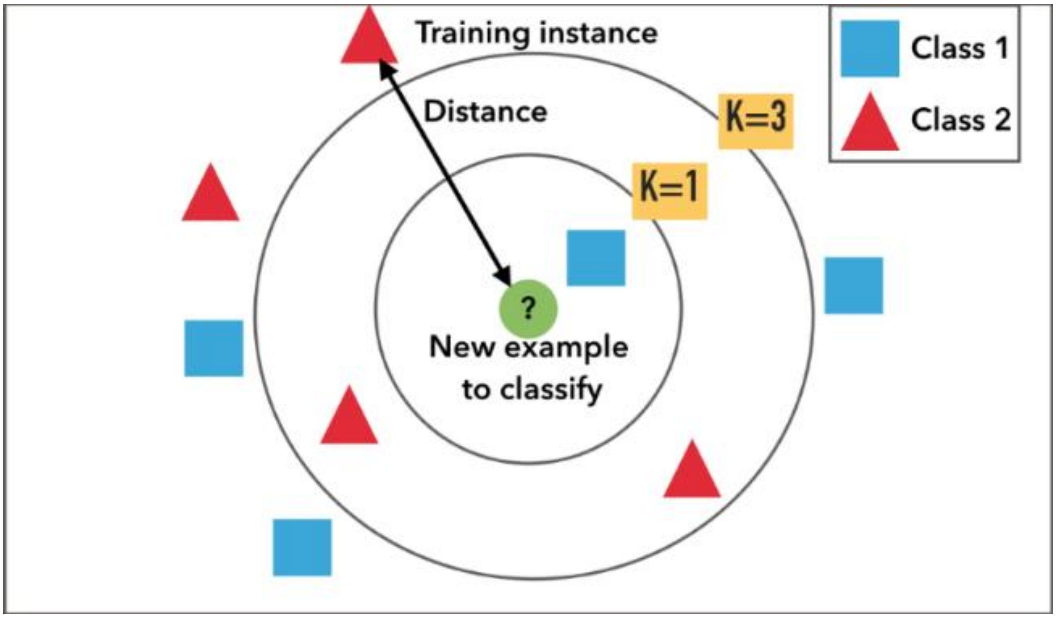
25. **In-class exercise A:**
    - Open the file **iris.arff**
    - Go to **Classify** tab and choose **SMO** algorithm. Click on **SMO** algorithm to review the algorithm configuration.
    - Use "**PolyKernel**" [**No normalization/Standardization**] (try exponent 1 & 2 respectively).
    - Use "**RBFKernel**" [**No normalization/Standardization**].
    - Open the file **lab1-inclass-ans.docx** and fill in the answers.



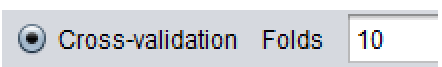
How to use Weka for Classification (kNN)?

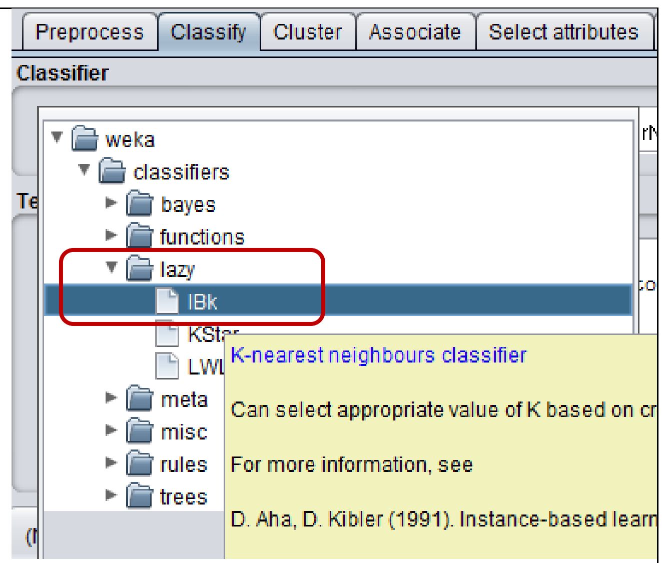
Nearest Neighbor Learning (also known as **Instance-based Learning**) is a useful data mining technique that allows you to use your past data instances, with known output values, to predict an unknown output value of a new data instance. It is often very accurate. It is a kind of lazy learning.

It searches for the K observations in the training data that are nearest to the measurements of the unknown instance. The k-nearest neighbors algorithm is called kNN for short. It supports both *classification* and *regression*. When making predictions on classification problems, **KNN will choose majority class among several neighbors** (k of them).



1. Open the file **labor.arff**, go to **Classify** panel, select **Choose**.
2. Select **lazy → IBk** (K-nearest neighbours classifier)
3. Keep the Test option **Cross-validation Folds 10**.

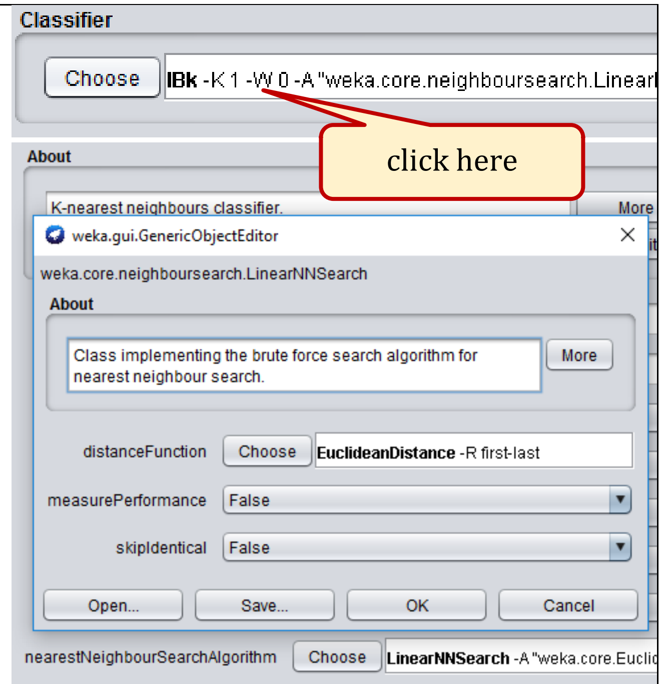




4. Click **Start**. It shows **Correctly Classified Instances** is **82.4561 %.** The **default KNN value = 1**. (If k = 1, then the object is simply assigned to the class of that single nearest neighbor)

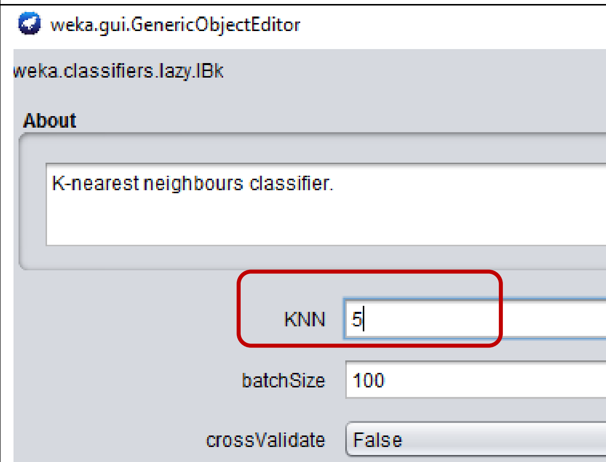
```
Correctly Classified Instances          47               82.
Incorrectly Classified Instances        10               17.
Kappa statistic                          0.6235
Mean absolute error                      0.1876
Root mean squared error                  0.4113
Relative absolute error                 41.0144 %
Root relative squared error             86.1487 %
Total Number of Instances               57
```

5. Click on the classifier to change the algorithm configuration.
   - The default value for **nearestNeigbourSearchAlgorithm** is a **LinearNNSearch.**
   - Clicking the name of this search algorithm and you can see another parameter **distanceFunction**. By default, **Euclidean distance** is used, it is used to calculate the distance between instances by calculating sum of squares of differences.

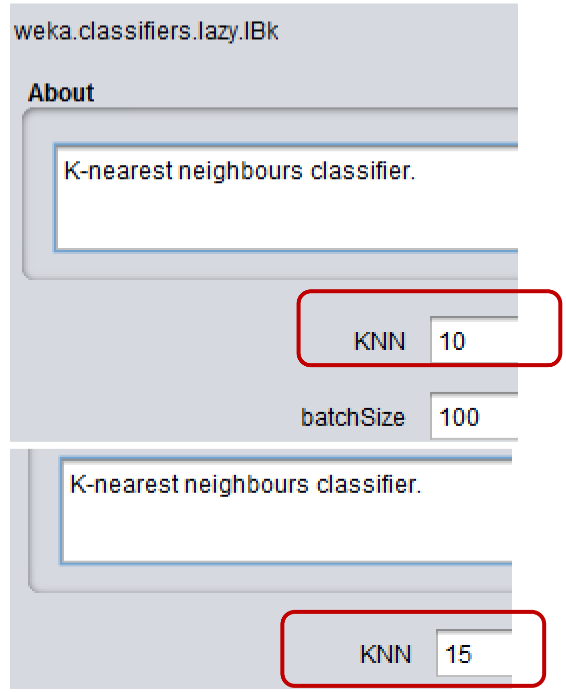


https://en.wikipedia.org/wiki/Euclidean_distance

6. Press **OK** and change the **KNN value to 5** and press **OK**. Click **Start** under Test options.
7. **Correctly Classified Instances** = 85.9649 %



8. **In-class exercise B:**
   - Repeat the steps before and try different KNN values (10 & 15).
   - Open the file **lab1-inclass-ans.docx** and fill in the answers.
   - Try different Test option (Percentage split) and fill in the answers.
9. **Close** the **Weka Explorer**.

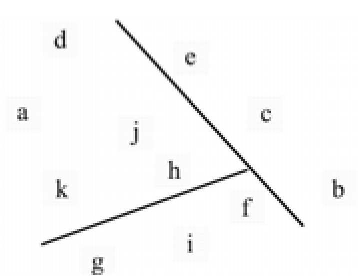


How to use Weka for Clustering (KMeans)?

Clustering allows a user to make groups of data to determine patterns from the data. Clustering has its advantages when the data set is defined and a general pattern needs to be determined from the data. Every attribute in the data set will be used to analyze the data. A major disadvantage of using clustering is that the user is required to know ahead of time how many groups he wants to create. Clustering differs from classification and regression because it does not produce a single output variable. (No class attribute)

There are four popular **cluster types** that are supported in Weka. They are **Disjoint sets**, **Overlapping sets**, **Probabilistic clusters** and **Hierarchical clusters**.

1. Disjoint sets

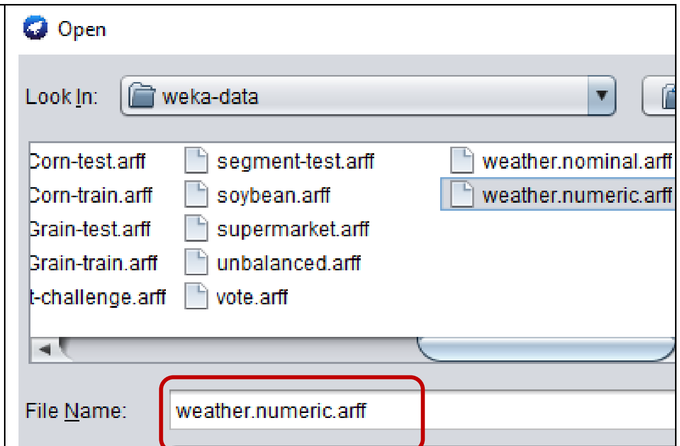
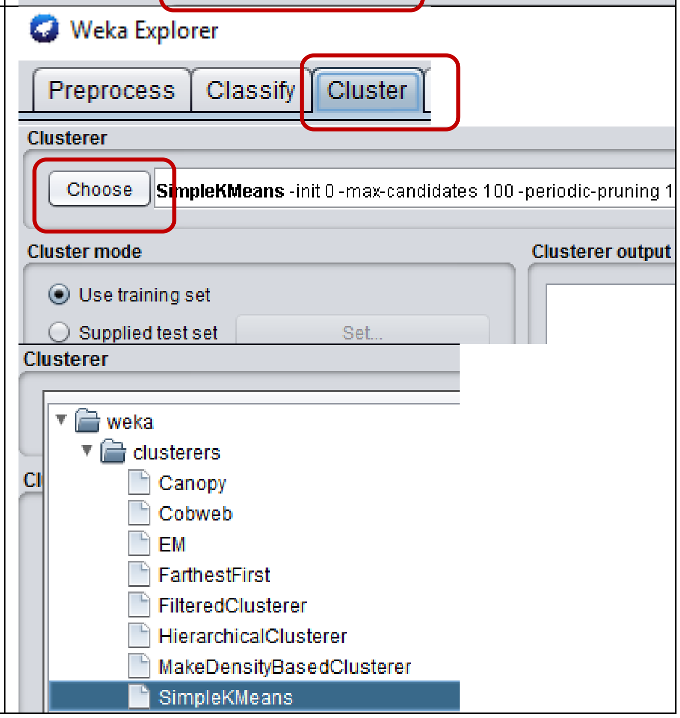


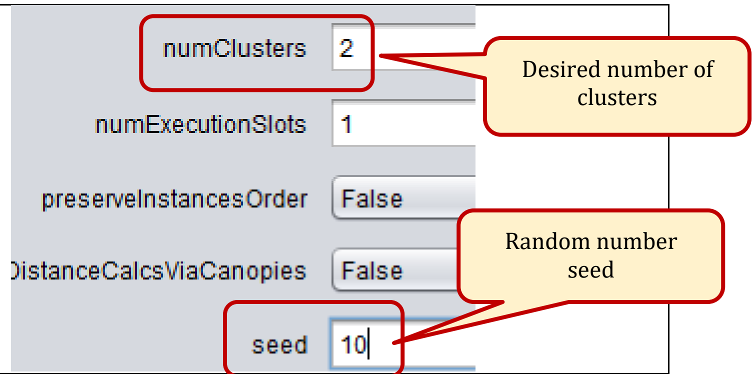
KMeans is belong to **disjoint set** cluster type. It takes the instance space and divide it into sets such that each part of the instance space is in just one cluster. Here are the procedures of using **KMeans** clustering:

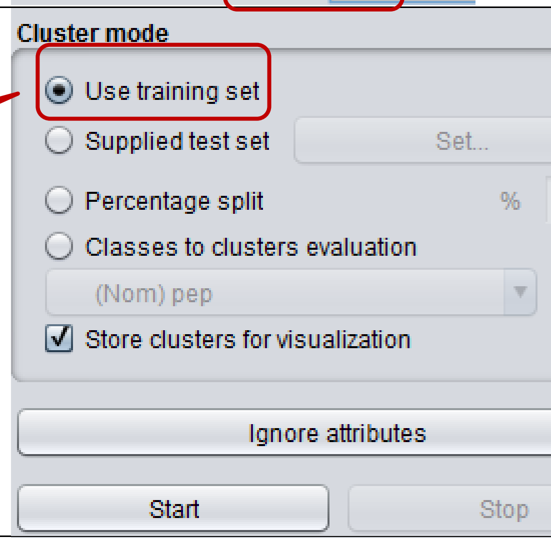
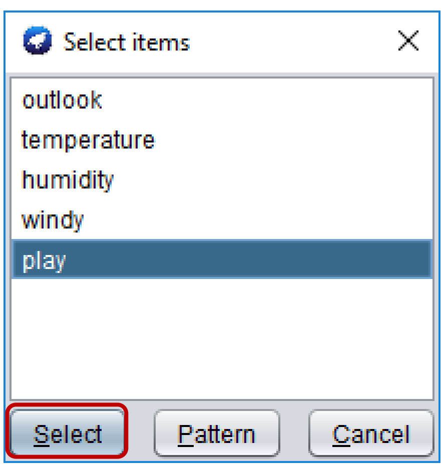
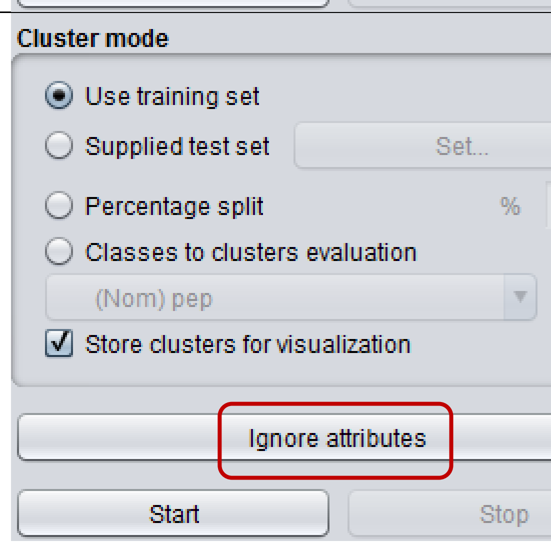
- Specify k, the desired number of clusters
- Choose k points at random as cluster centers

- Assign instances to their closest cluster center
- Calculate the centroid (i.e., mean) of instances in each cluster. These centroids are the new cluster centers
- Repeat until the cluster centers don't change

KMeans is always dependent on the initial assignment of the cluster centers
This algorithm minimizes the total squared distance from instances to their cluster centers. (but it is a local minimum, not global minimum)

1. Open the **Weka Explorer** again. Use **Preprocess** panel to open the file **weather-numeric.arff**.	
2. To perform **KMeans** clustering, select the **Cluster** panel → **Choose** 3. Select **SimpleKMeans** under **clusterer** (*SimpleKMeans* algorithm uses Euclidean distance measure to compute distances between instances and cluster centers)	

<table>
<tr>
<td>4. Click on the classifier to change the algorithm configuration. Keep number of clusters = 2, seed = 10 (default), press OK. (seed is used for generating initial cluster centers)</td>
<td>
</td>
</tr>
<tr>
<td>5. Make sure that in the Cluster mode, the Use training set option is selected.

Use training set means using all instances in the arff file for clustering</td>
<td>
</td>
</tr>
<tr>
<td>6. Click Ignore attributes button. Select the class attribute (play) to be ignored. Then click Start.

</td>
<td>
</td>
</tr>
</table>

7. Two clusters are created. The figure on the right shows the initial random centroids. Number of iterations = 3 and sum of squared errors = 11.24

```
Number of iterations: 3
Within cluster sum of squared errors: 11.237456311387234

Initial starting points (random):

Cluster 0: rainy,75,80,FALSE
Cluster 1: overcast,64,65,TRUE

Missing values globally replaced with mean/mode
```

8. This figure shows the **centroid of final clusters.** One instance for each cluster representing the cluster centroid.

```
Final cluster centroids:
                           Cluster#
Attribute      Full Data          0          1
                  (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
```

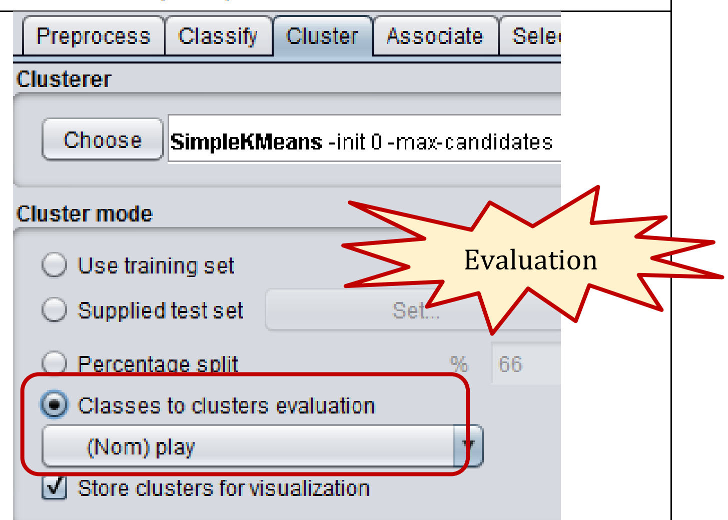
9. This figure shows how many instances in each cluster. 9 instances are in cluster 0 & 5 instances are in cluster 1.

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

Clustered Instances

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

10. Evaluate the model using **Classes to clusters evaluation** under **Cluster mode.** 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.

11. Select the attribute **(Nom)play** to be ignored.

12. Click **Start** button.

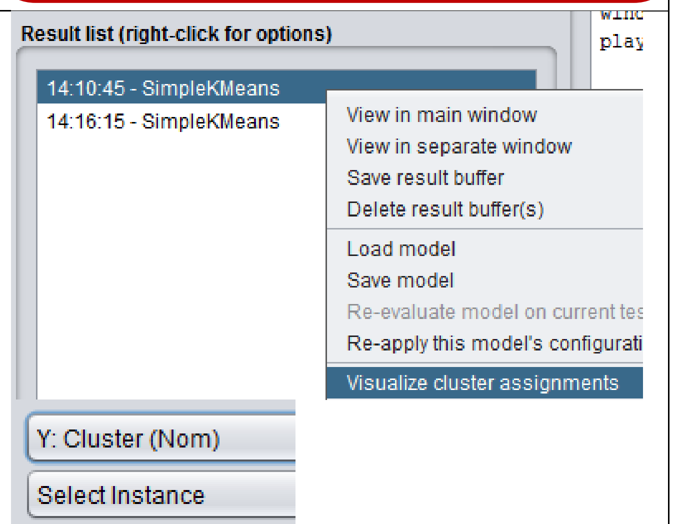


13. It computes the classification error, and shows the evaluation results. **Incorrectly clustered instances** (%) = 42.8571%.

```
Ignored:
              play
Test mode:    Classes to clusters evaluation on training dat

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

Clustered Instances

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


Class attribute: play
Classes to Clusters:

 0 1  <-- assigned to cluster
 6 3 | yes
 3 2 | no

Cluster 0 <-- yes
Cluster 1 <-- no

Incorrectly clustered instances :       6.0      42.857
```

6 instances are correctly assigned to "yes" cluster (cluster 0), 2 instances are correctly assigned to "no" cluster (cluster 1)

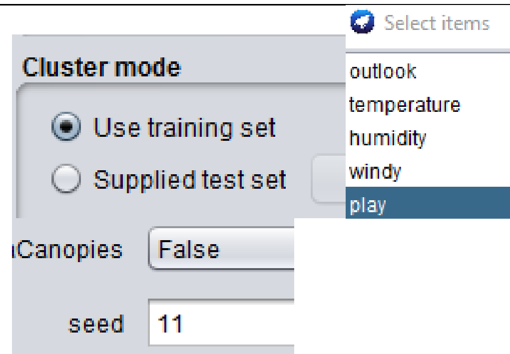
14. **Right click** the **second SimpleKMeans model** in **Result list** and choose **Visualize cluster assignments**. Set **Y: Cluster (Nom)**. Then press **Save** button. Name the file as **kmeanscluster.arff**. Use Notepad to open the file.



15. Evaluation clustering result is like this:
- Number of yes in cluster 0 is 6. (TP)
- Number of no in cluster 0 is 3. (FN)
- Number of yes in cluster 1 is 3. (FP)
- Number of no in cluster 1 is 2. (TN)
- 6 instances are wrongly classified. (same as the previous results)

```
Cluster 0 <-- yes
Cluster 1 <-- no
```

```
@data
0,sunny,85,85,FALSE,no,cluster0
1,sunny,80,90,TRUE,no,cluster0
2,overcast,83,86,FALSE,yes,cluster0
3,rainy,70,96,FALSE,yes,cluster0
4,rainy,68,80,FALSE,yes,cluster0
5,rainy,65,70,TRUE,no,cluster1
6,overcast,64,65,TRUE,yes,cluster1
7,sunny,72,95,FALSE,no,cluster0
8,sunny,69,70,FALSE,yes,cluster0
9,rainy,75,80,FALSE,yes,cluster0
10,sunny,75,70,TRUE,yes,cluster1
11,overcast,72,90,TRUE,yes,cluster1
12,overcast,81,75,FALSE,yes,cluster0
13,rainy,71,91,TRUE,no,cluster1
```

<table>
<tr>
<td>16. In-class exercise C:
• Open the file lab1-inclass-ans.docx and fill in the answers
• By using the same file weather.numeric.arff, try different seed value (i.e. 11), we can get different sum of square errors and starting centroids. The final centroids will be different too.
• Capture the screenshots in each case</td>
<td>
</td>
</tr>
</table>

Take home assignment

Open another file **lab1-assignment-ans.docx**, complete it individually.

Submission

Submit the following files to buelearning website:

- lab1-inclass-ans.docx (In-class exercise)
- lab1-assignment-ans.docx (Take home assignment)

References

1. More Data Mining with Weka MOOC -Material. (n.d.). Retrieved from http://www.cs.waikato.ac.nz/ml/weka/mooc/moredataminingwithweka
2. Data Mining Algorithms [Video file]. (n.d.). Retrieved from https://www.youtube.com/playlist?list=PLea0WJq13cnCS4LLMeUuZmTxqsqlhwUoe
3. How To Estimate The Performance of Machine Learning Algorithms in Weka https://machinelearningmastery.com/estimate-performance-machine-learning-algorithms-weka/
4. Support Vector Machine - How Support Vector Machine Works https://www.youtube.com/watch?v=TtKF996oEl8
5. Multilayer Perceptron https://weka.sourceforge.io/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html
6. How To Use Classification Machine Learning Algorithms in Weka https://machinelearningmastery.com/use-classification-machine-learning-algorithms-weka/