**Lab 7: Use Multilayer perceptron algorithm for the following data.**

**Introduction:**

Multilayer Perceptron (MLP) is a type of artificial neural network widely used in classification and regression tasks. It consists of an input layer, one or more hidden layers, and an output layer. Each neuron uses an activation function (usually sigmoid) to introduce non-linearity, making it capable of learning complex patterns. In data mining, MLP is a supervised learning algorithm that learns from labeled data by minimizing the error between predicted and actual class labels using back propagation.

**How does Multilayer Perceptron work?**

The working of the MLP algorithm can be outlined in the following steps:

1. Input Layer: Receives the input features (e.g., outlook, temperature, humidity, windy).
2. Forward Propagation:
   * Each input is passed to the neurons of the hidden layer.
   * Neurons compute weighted sums and apply activation functions.
   * The result is passed to the output layer.
3. Output Layer: Produces the predicted class (e.g., "play" = yes/no).
4. Error Calculation: Compares predicted output with actual class label and calculates error.
5. Back propagation: Adjusts weights using gradient descent to minimize the error.
6. Repeat Training: Continues for a number of epochs until the model converges.

**Multilayer Perceptron in WEKA:**

Weka provides a direct implementation of Multilayer Perceptron under the “Classify” tab.

Steps to implement:

1. Load a dataset (.arff format).
2. Go to the “Classify” tab.
3. Select function> MultilayerPerceptron.
4. Click Start to train the model.
5. WEKA shows the confusion matrix, accuracy, and detailed class performance metrics.

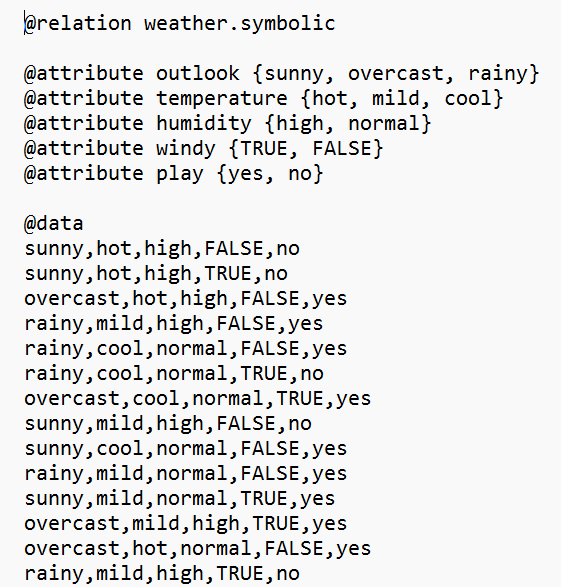


Figure 7.1: .arff file

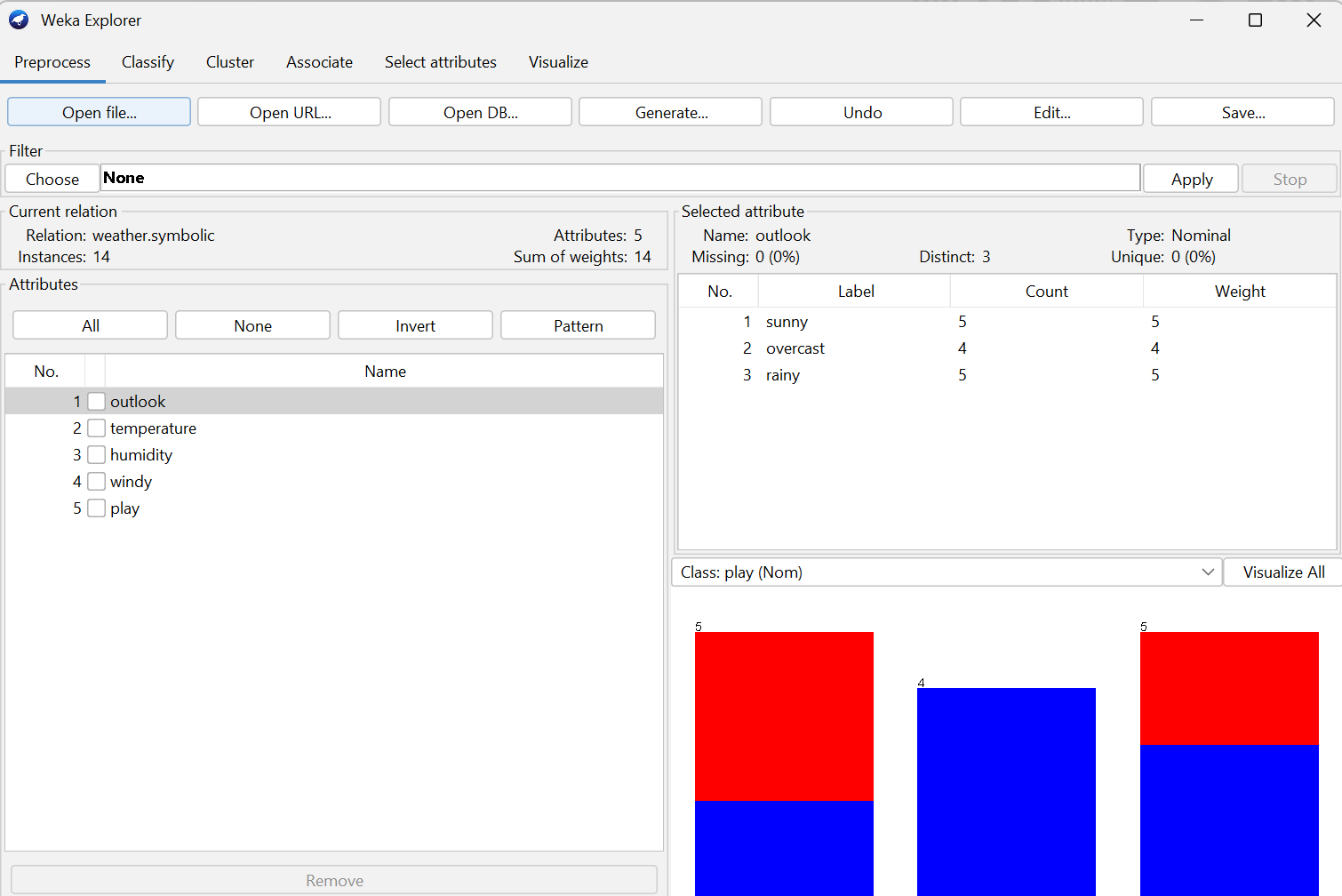
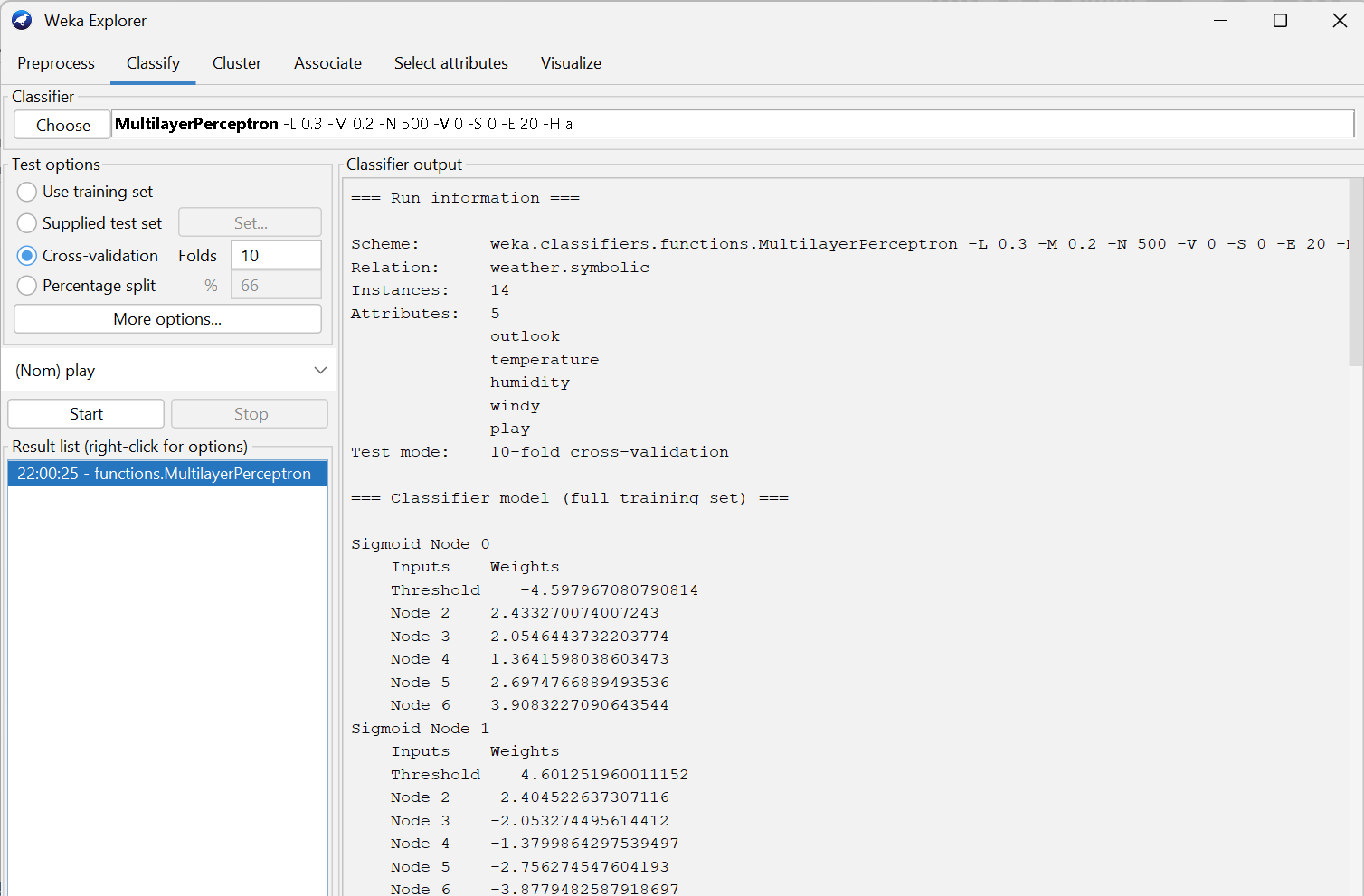


Figure 7.2: Pre-process data



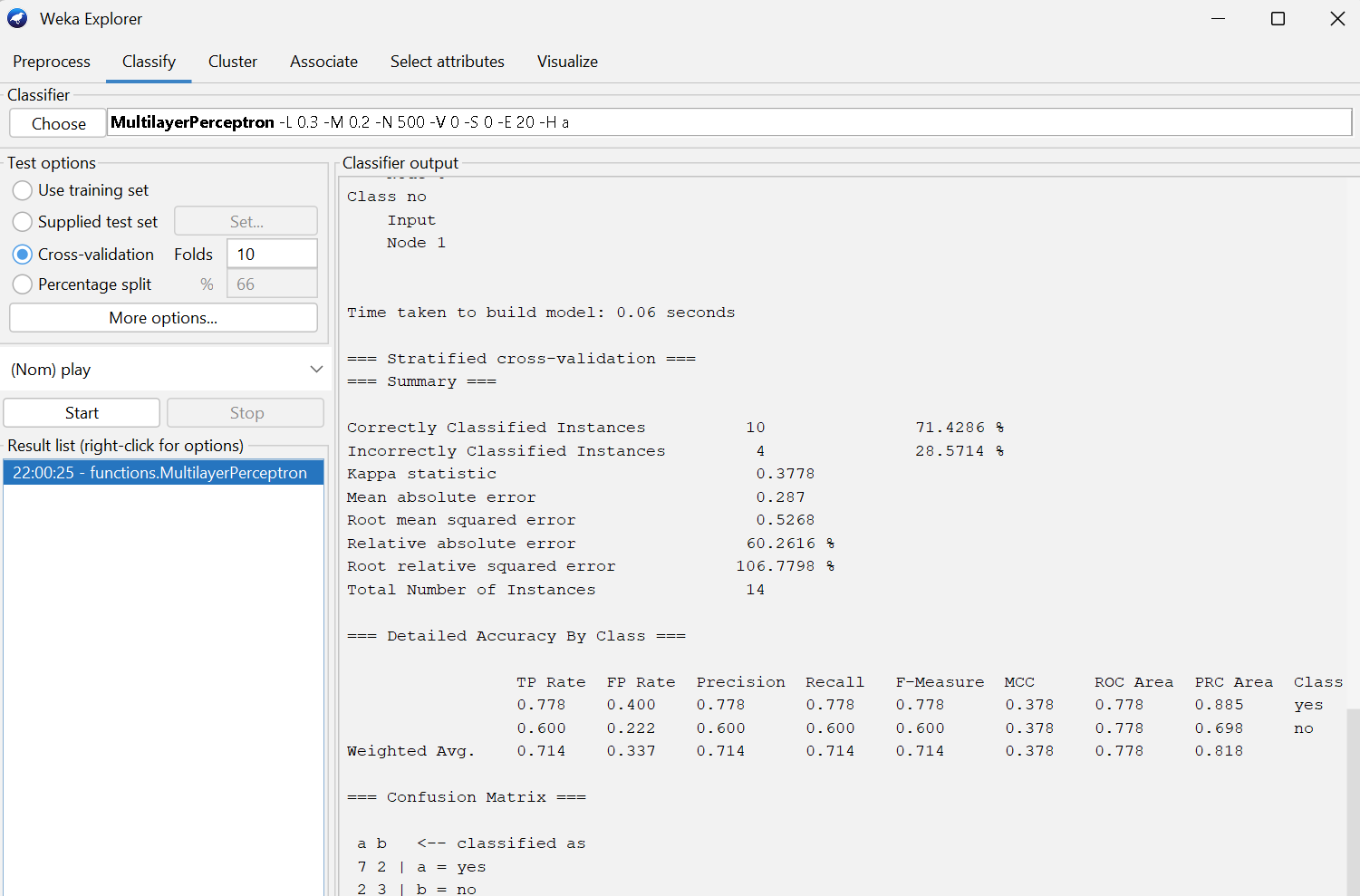


Figure 7.4: Output results

**Conclusion:**

Multilayer Perceptron algorithm effectively learns patterns in data through layered neural processing, making it well-suited for classification tasks like predicting outcomes from weather conditions.

**Lab 8: Use Simple K-means clustering for the following data.**

**Introduction:**

Simple K-means is an unsupervised machine learning algorithm used for clustering data into groups (clusters) based on similarity. It works by dividing data points into *k* clusters, where each point belongs to the cluster with the nearest centroid. It is commonly used to identify patterns or groupings in unlabeled data by minimizing the distance between data points and their assigned cluster centers (centroids). In data mining, K-Means is commonly applied for pattern discovery, customer segmentation, and anomaly detection, where it helps uncover hidden structures in unlabeled data.

**How does Simple K-means work?**

The working of the K-means algorithm can be summarized in the following steps:

1. Choose the number of clusters (k).
2. Initialize k centroids randomly.
3. Assign each data point to the nearest centroid based on distance (usually Euclidean).
4. Recalculate centroids as the mean of all points in each cluster.
5. Repeat steps 3–4 until centroids no longer change significantly or a maximum number of iterations is reached.

Since K-means is unsupervised, it does not use class labels (like "play") in the clustering process.

**Simple K-means clustering in WEKA:**

Weka provides a direct implementation of Simple K-means clustering under the “Cluster” tab.

Steps to implement:

1. Open WEKA and load a dataset (in .arff format).
2. Go to the "Cluster" tab.
3. Choose clusterers> SimpleKMeans .
4. (Optional) Set variables
5. Click Start to train the model.
6. WEKA displays the decision tree, accuracy, and confusion matrix.

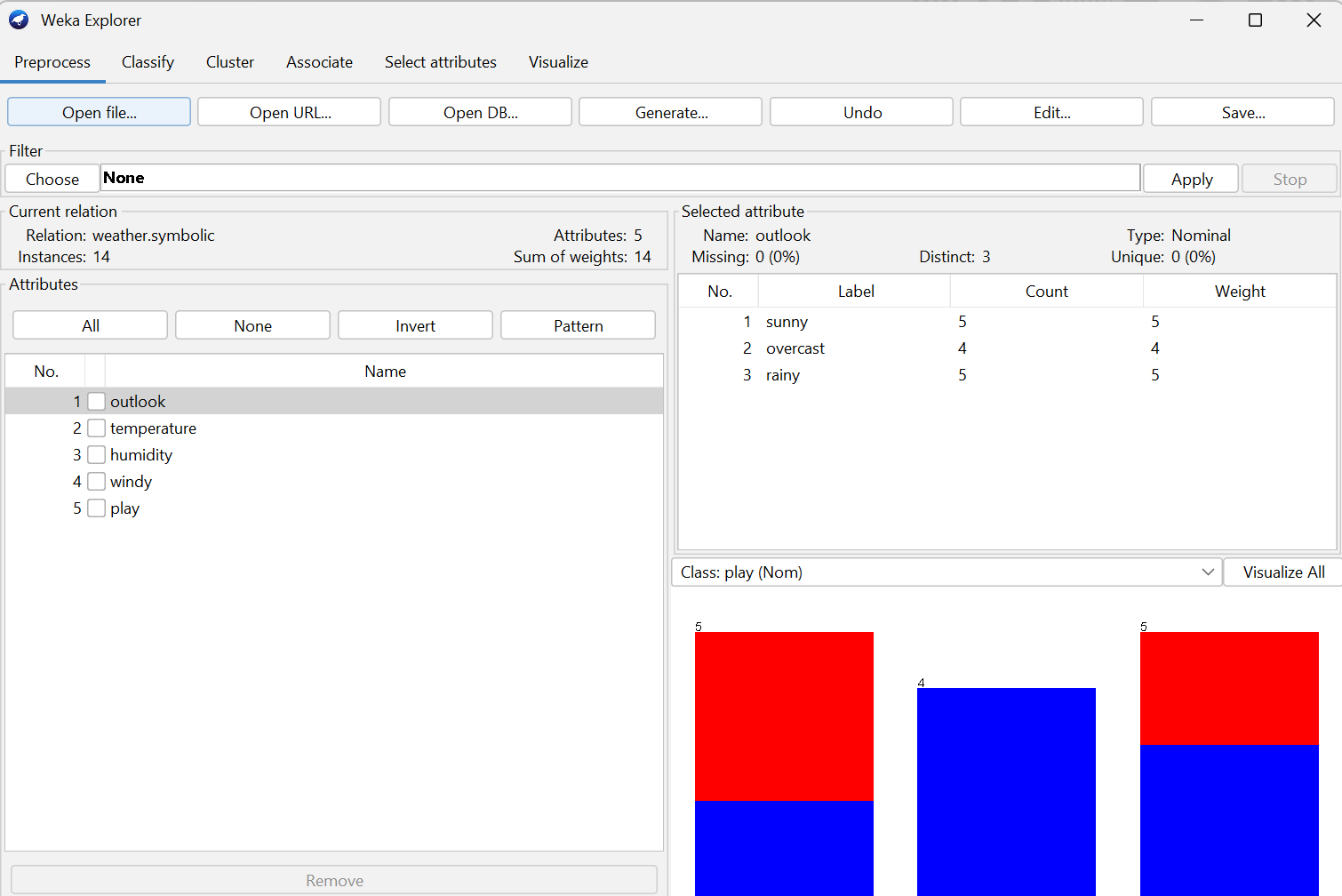


Figure 8.1: Pre-process data

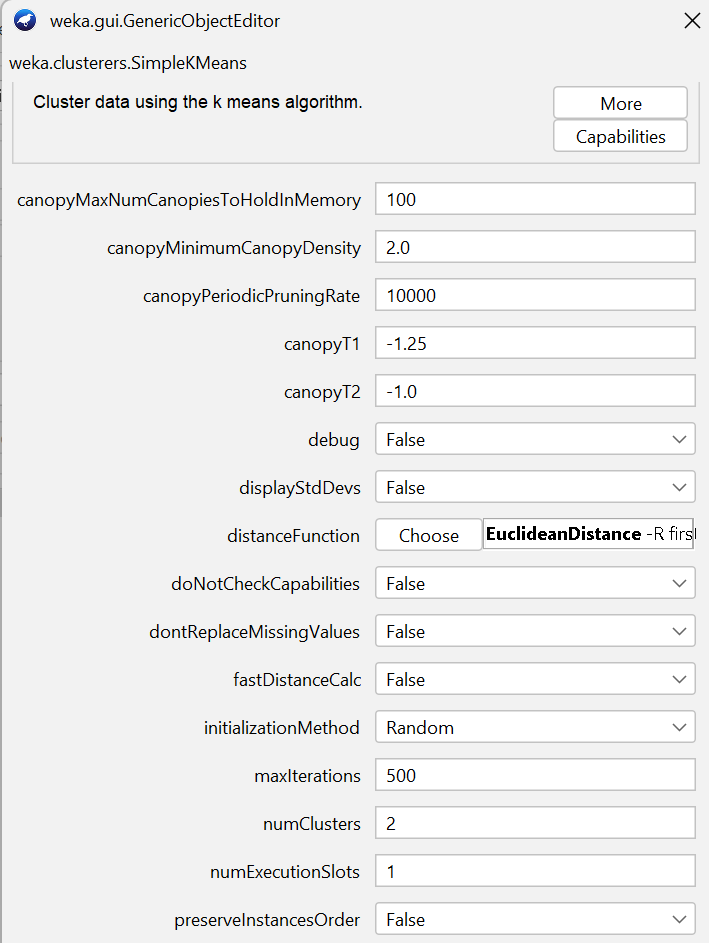


Figure 8.2: Configuration

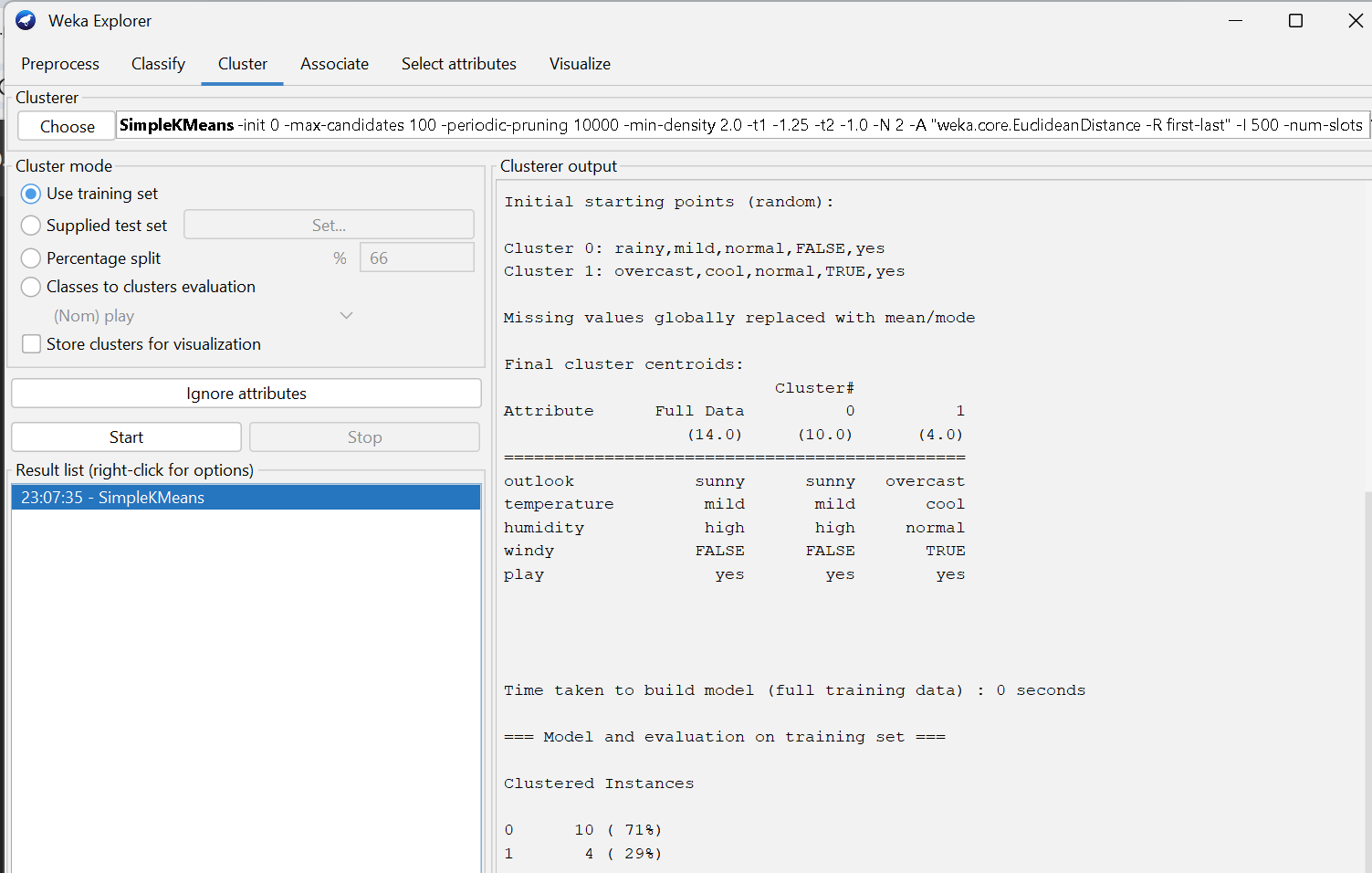


Figure 8.3: Output result

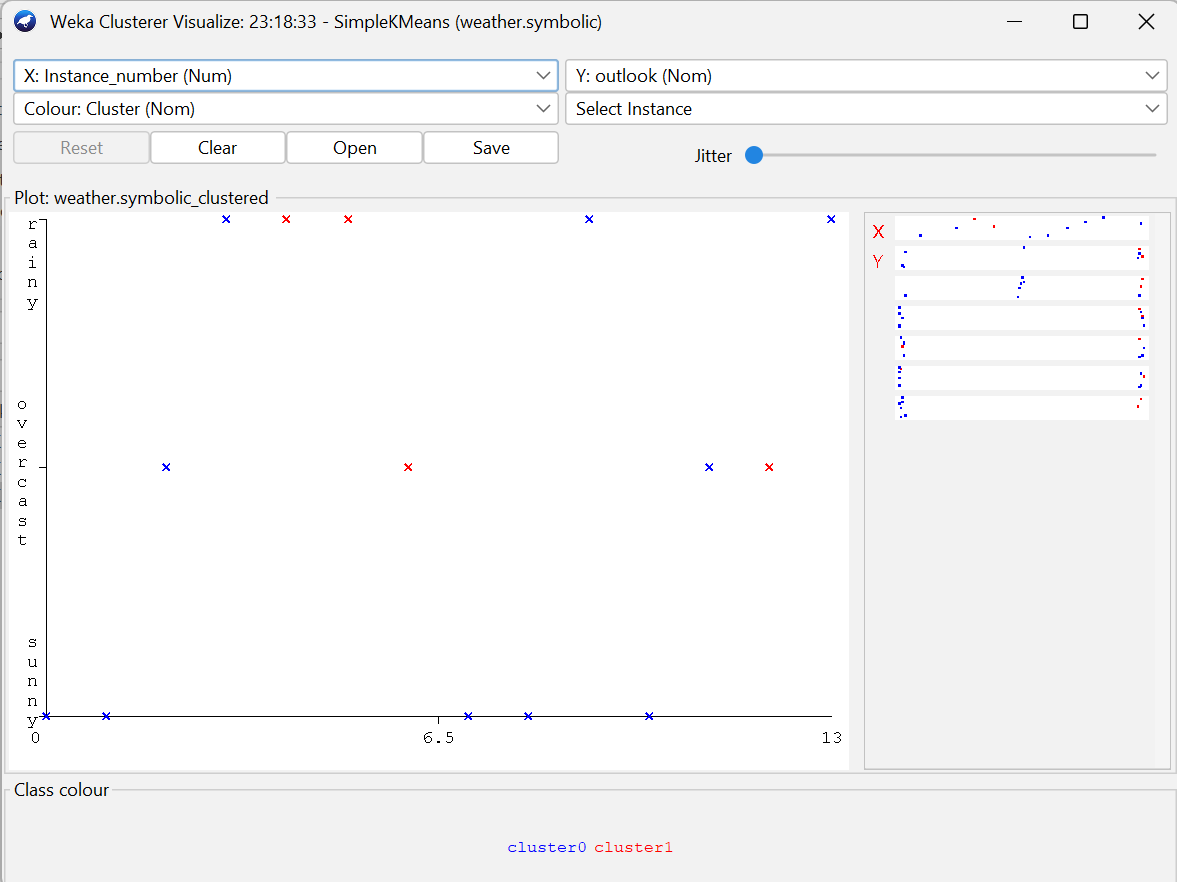
****

Figure 8.4: Visualizing the clusters

**Conclusion:**

Simple K-means groups similar data points into clusters by minimizing the distance to cluster centroids. It is effective for discovering natural groupings and hidden patterns in unlabeled data.

**Lab 9: Use Hierarchical clustering for the following data.**

**Introduction:**

Hierarchical Clustering is an unsupervised machine learning algorithm used to build a hierarchy of clusters without predefining the number of groups. It works by either recursively merging smaller clusters into larger ones (*agglomerative approach*) or by recursively splitting larger clusters into smaller ones (*divisive approach*). The results are often represented using a dendrogram, a tree-like diagram that shows the order and distance at which clusters are merged or divided. In data mining, hierarchical clustering is applied in taxonomy, gene expression analysis, and social network analysis, where it helps identify nested structures and relationships within unlabeled data.

**How does Hierarchical Clustering work?**

The working of hierarchical clustering can be summarized in the following steps:

1. Start with each data point as its own cluster.
2. Calculate distances between all pairs of clusters.
3. Merge the two closest clusters based on a linkage method (e.g., single, complete, average).
4. Repeat merging steps until all data points are combined into one large cluster or the desired number of clusters is reached.
5. The result is visualized as a dendrogram, showing how clusters were formed.

**Hierarchical clustering in WEKA:**

Weka provides a direct implementation of Hierarchical clustering under the “Cluster” tab.

Steps to implement:

1. Open WEKA and load a dataset (in .arff format).
2. Go to the "Cluster" tab.
3. Choose clusterers> HierarchicalClusterer.
4. (Optional) Set variables
5. Click Start to train the model.
6. WEKA displays the decision tree, accuracy, and confusion matrix.

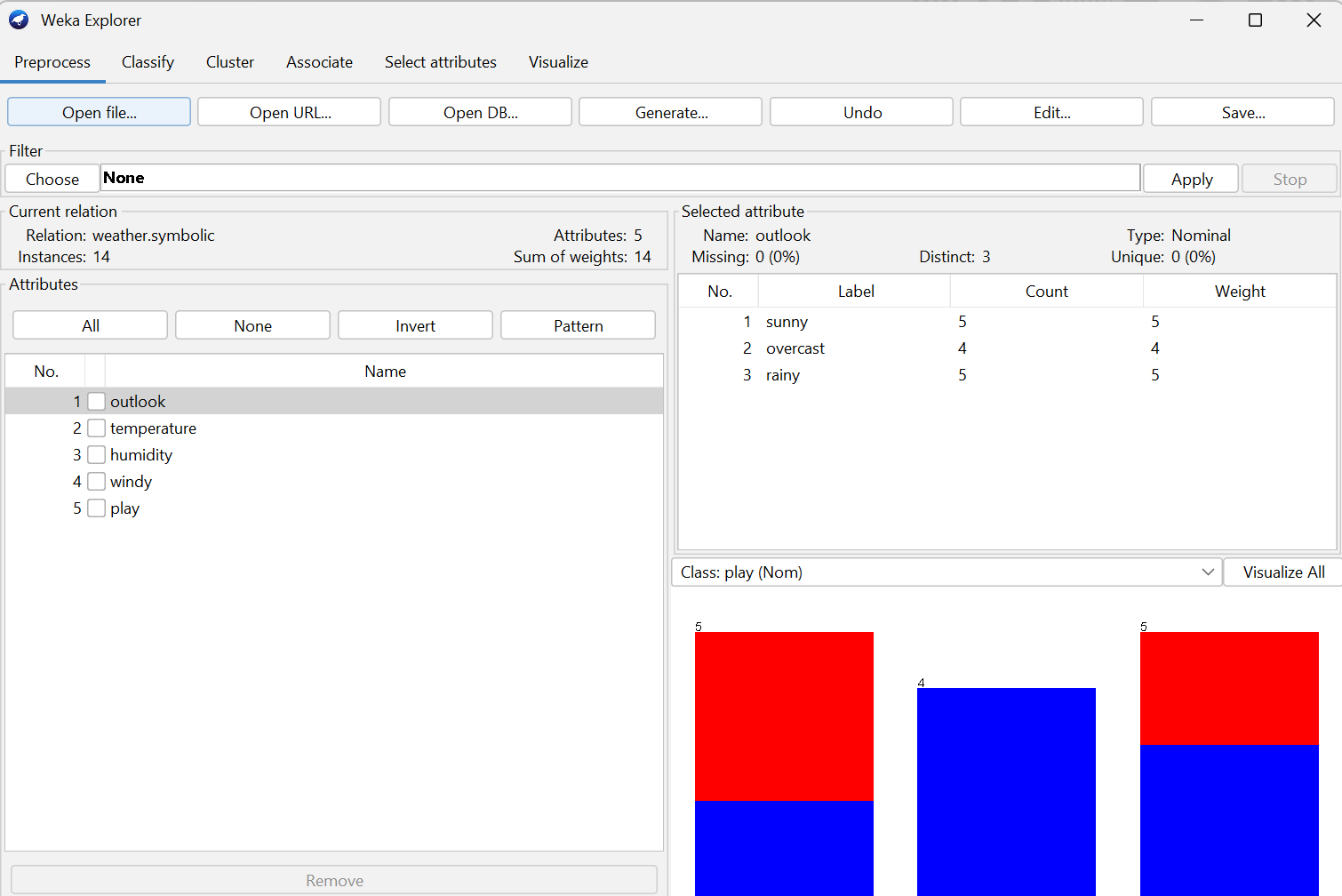


Figure 9.1: Pre-process data

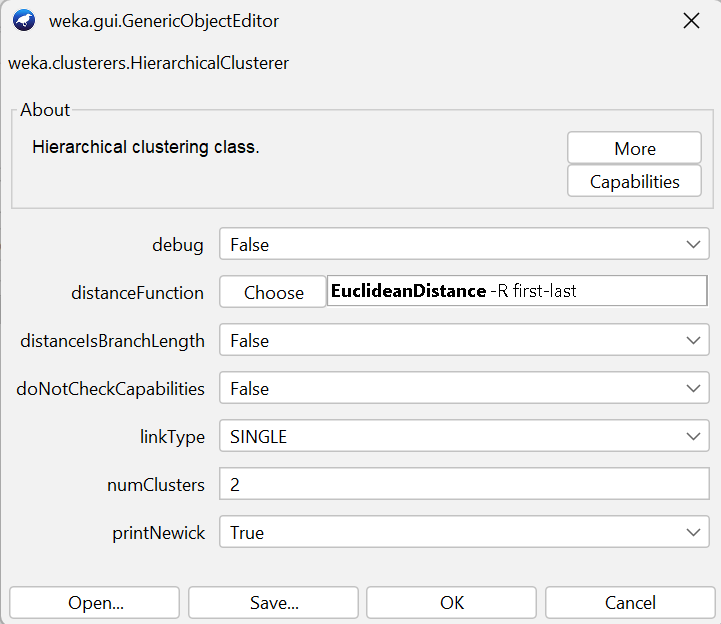


Figure 9.2: Configuration

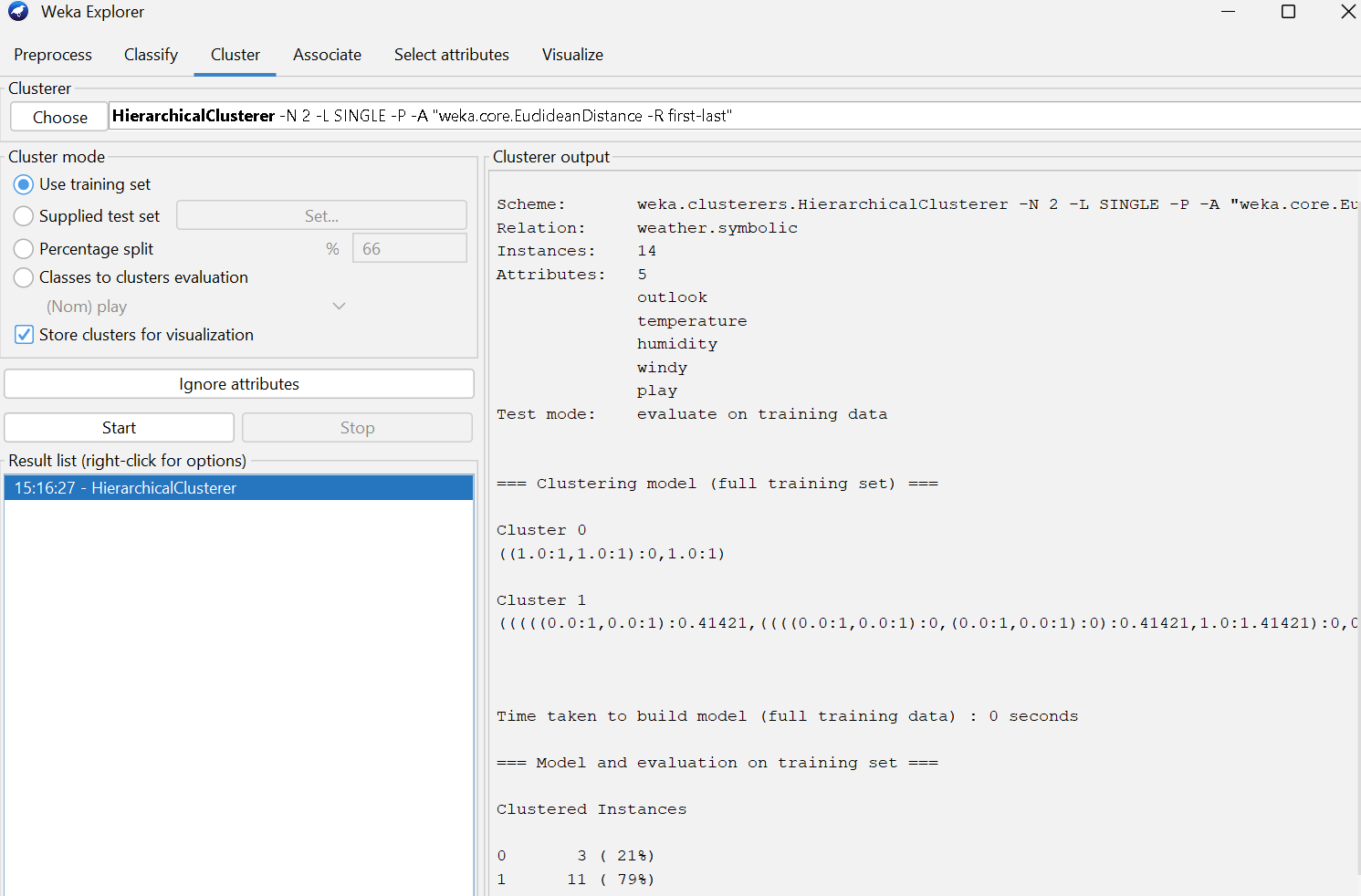


Figure 9.3: Output result

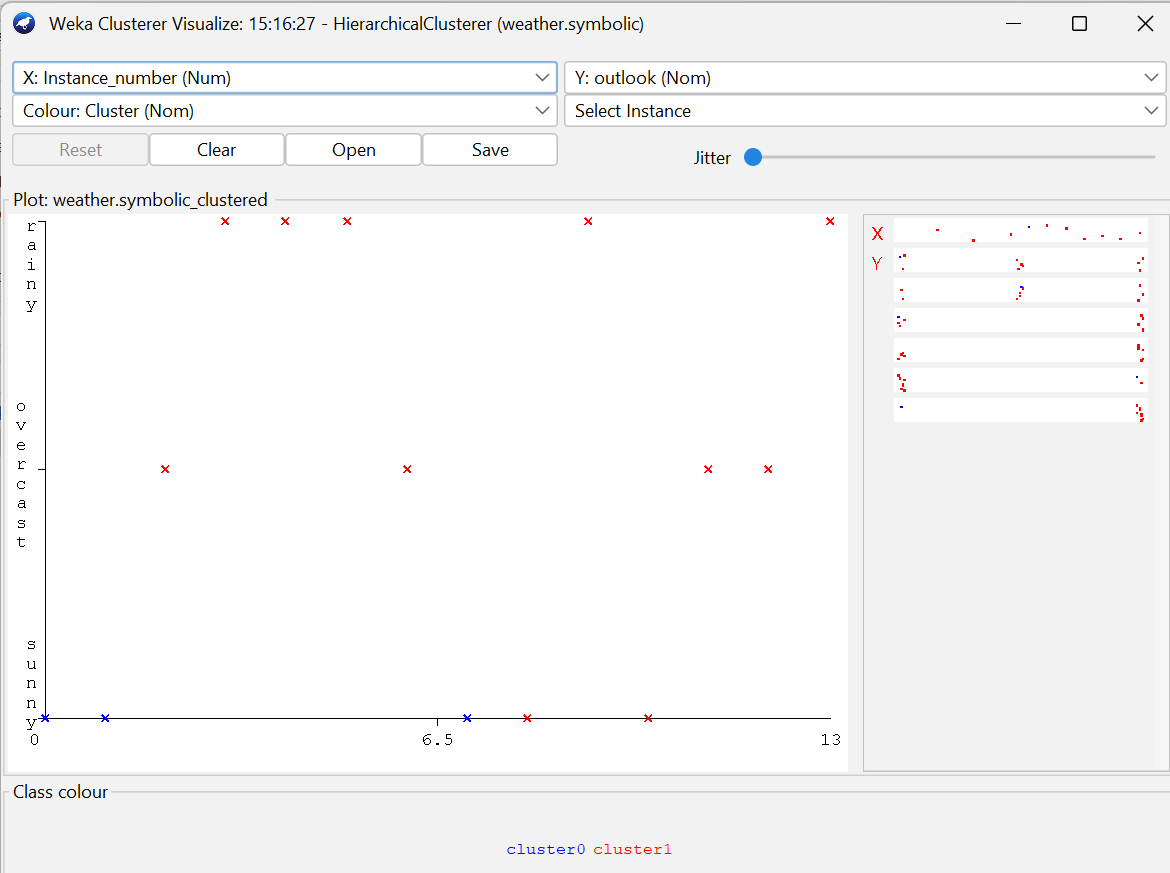


Figure 9.4: Visualizing the clusters

**Conclusion:**

Hierarchical clustering creates a tree-like structure of nested clusters, making it useful for understanding relationships in data and exploring natural groupings without predefining the number of clusters.

**Lab 10: Density based clustering**

**Introduction:**

Density-based clustering is an unsupervised machine learning algorithm that groups data points based on regions of high density separated by regions of low density. Unlike partitioning methods, it does not require specifying the number of clusters beforehand and can discover clusters of arbitrary shapes. The most common algorithm, DBSCAN, defines clusters using two parameters: the neighborhood radius (*epsilon*) and the minimum number of points required to form a dense region. Points in dense regions are assigned to clusters, while points in sparse regions are treated as noise or outliers. In data mining, density-based clustering is widely used for spatial data analysis, anomaly detection, and image segmentation due to its ability to handle irregular cluster structures and noisy data.

**How does Density Based Clustering work?**

The key idea of density-based clustering (e.g., DBSCAN) can be summarized in these steps:

1. Select a radius (ε) and a minimum number of points (MinPts).
2. Identify core points: points that have at least MinPts within ε distance.
3. Form clusters by connecting core points and including directly reachable neighbors.
4. Label remaining points that don’t fit into any cluster as noise or outliers.
5. Clusters grow based on density rather than distance from centroids.

**Density Based clustering in WEKA:**

Weka provides a direct implementation of Hierarchical clustering under the “Cluster” tab.

Steps to implement:

1. Open WEKA and load a dataset (in .arff format).
2. Go to the "Cluster" tab.
3. Choose clusterers> MakeDensityBasedClusterer.
4. (Optional) Set variables
5. Click Start to train the model.
6. WEKA displays the decision tree, accuracy, and confusion matrix.

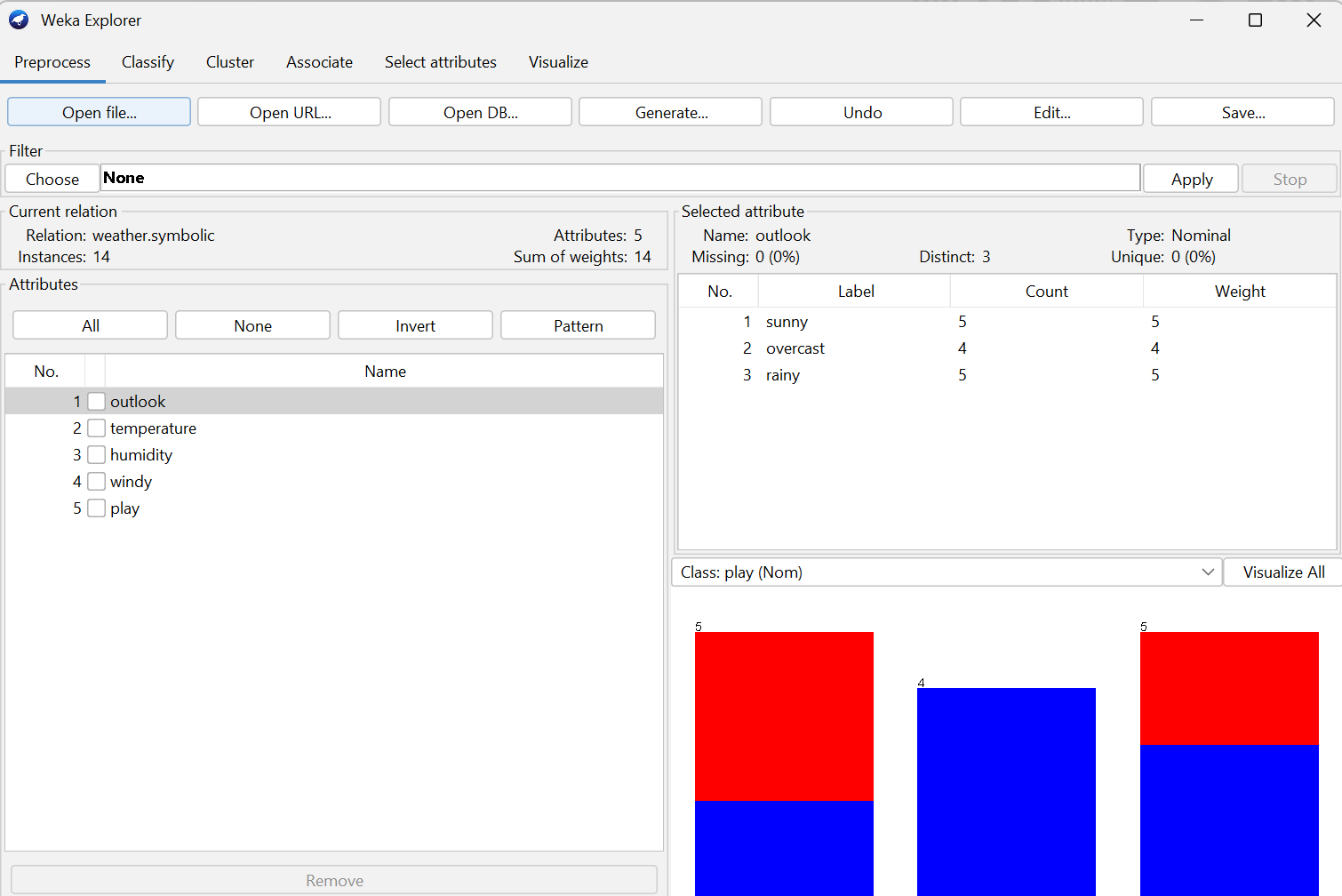


Figure 9.1: Pre-process data

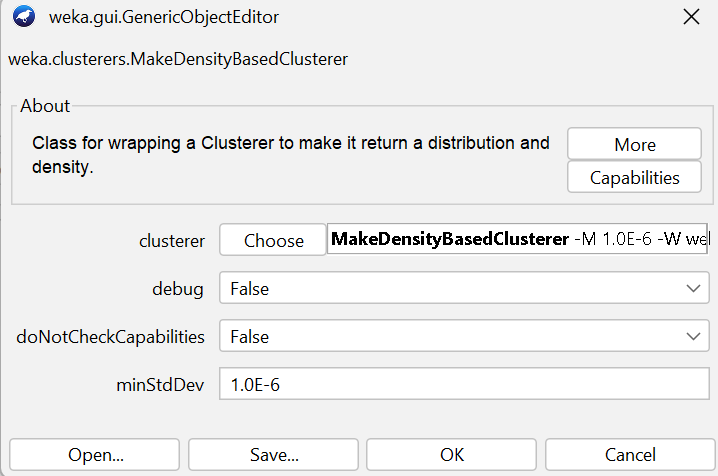


Figure 9.2: Configuration

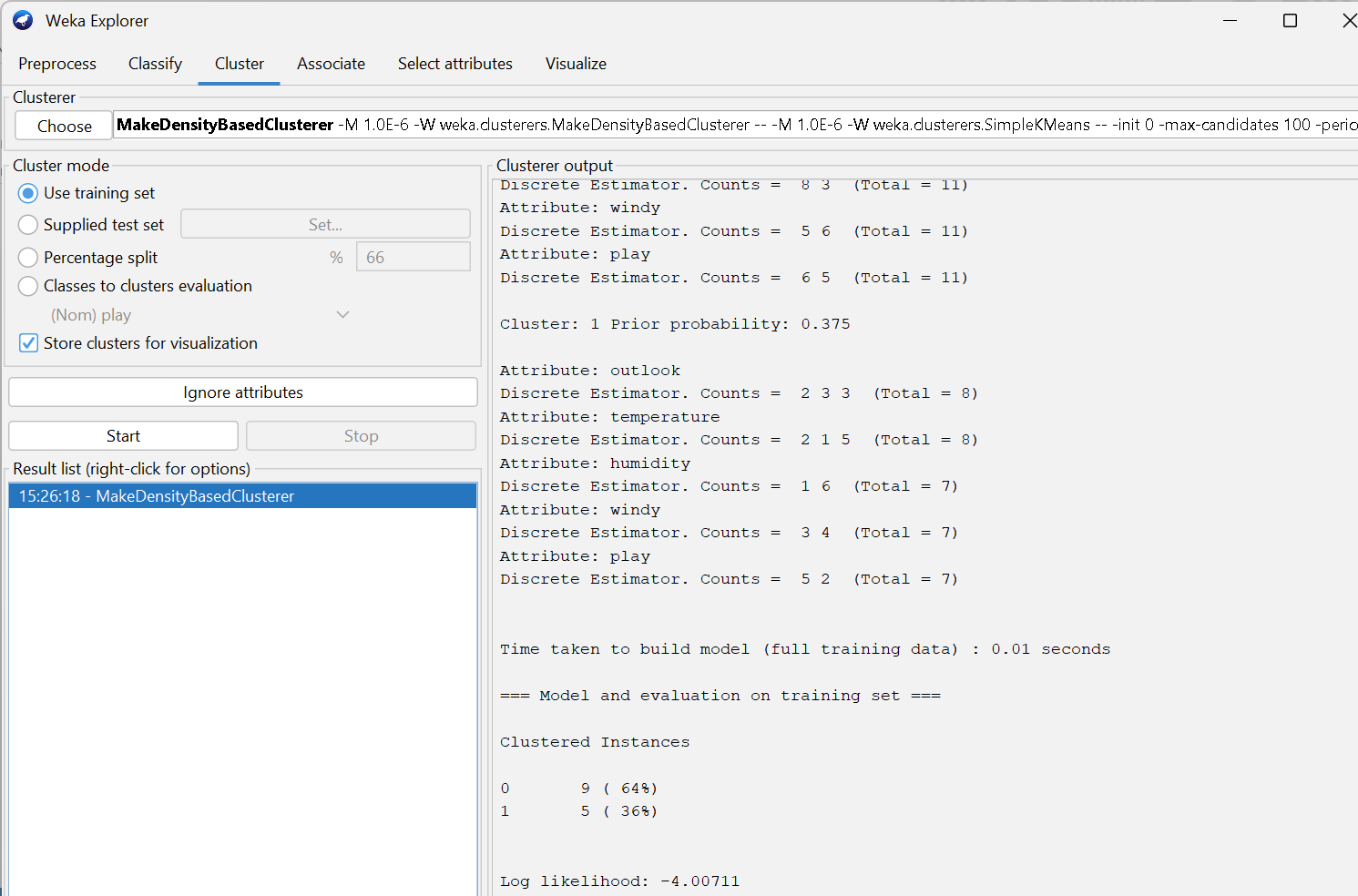


Figure 9.3: Output result

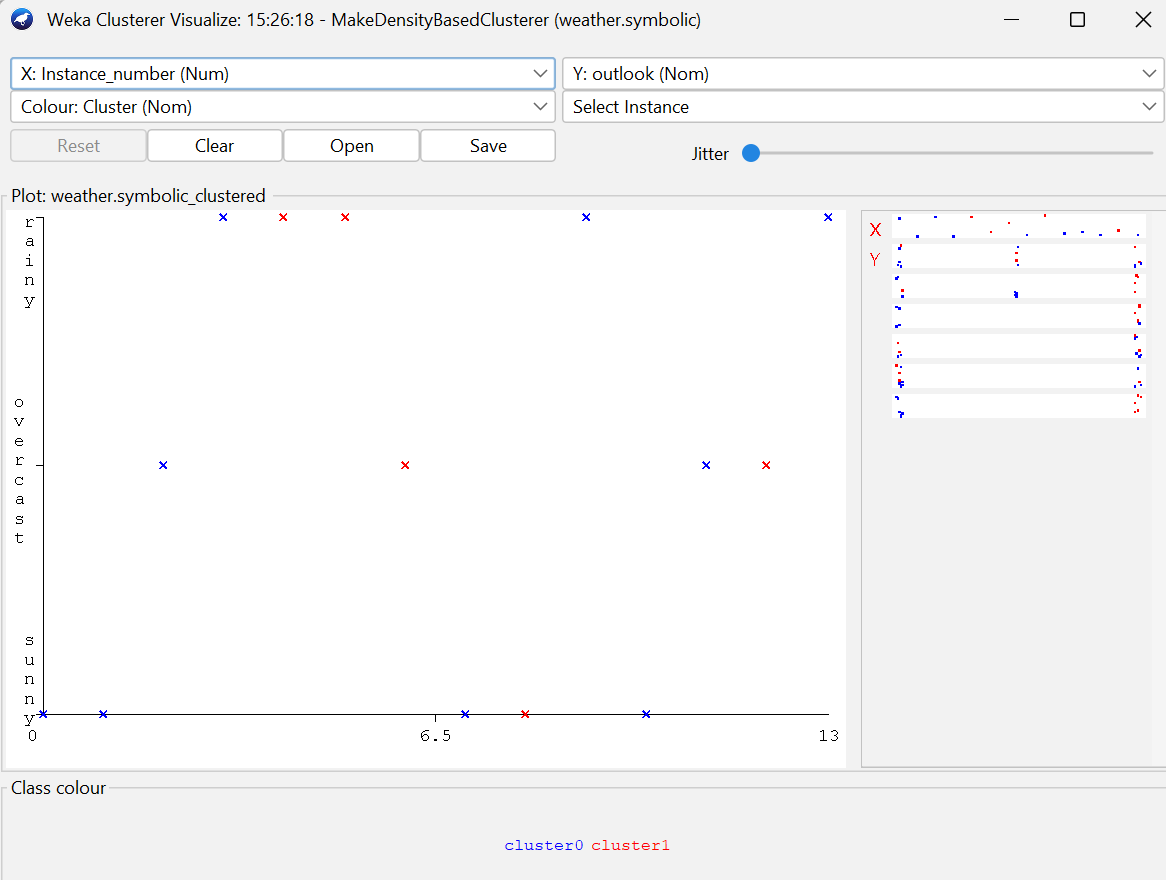


Figure 9.4: Visualizing the clusters

**Conclusion:**

Density-based clustering identifies clusters based on data density, allowing it to find arbitrarily shaped clusters and detect outliers effectively—making it ideal for complex and noisy datasets.

**COLLEGE OF APPLIED BUSINESS AND TECHNOLOGY**

Gangahity, Chabahil, Kathmandu



**Laboratory assignment report of**

**CSC410: Data Warehousing and Data Mining**

**Submitted by:**

Shreya Pathak

Roll no. 28655/078

BSc.CSIT 7th Semester

**Submitted to:**

Mr. Tushar Maharjan

Department of Computer Science and Information Technology

August 10, 2024