Laboratory assignment

Component 3 - Machine Learning Techniques

Authors: Leordean Ada Alexandra, Selegean Victor
Group: 242

January 14, 2026

1 Supervised Learning - k-Nearest Neighbors (KNN)

The classifier used is k-Nearest Neighbors (KNN), a non-parametric, instance-based supervised learning method. It does not construct an explicit model, making predictions by comparing a new example to those already seen in training. As such, we assume that songs with similar feature values tend to share the same popularity label.

KNN relies on a distance function (Euclidian distance, in this case) to measure similarity between songs.

1.1 Input Space

Each song is represented by some numerical audio features. For this project, only the following are used:

$$\mathbf{x} = (x_1, x_2, x_3, x_4) = (\text{danceability, energy, tempo, valence}) \tag{1}$$

Thus:

$$x \in \mathbf{R}^4 \tag{2}$$

1.2 Output Space

$$y \in \{0, 1\} \tag{3}$$

Where $y = 1$ means the track's popularity is above 70% and $y = 0$ otherwise.
As such, there is a learning function:

$$h : \mathbf{R}^4 \to \{0, 1\} \tag{4}$$

1.3 Formal Target Definition

There exists an unknown true mapping:

$$f^*(x) : \mathbf{R}^4 \to \{0, 1\} \tag{5}$$

which determines whether a song is objectively popular.
The goal of supervised learning is to approximate

$$h(x) \approx f^*(x) \tag{6}$$

based on labeled examples.

1.4 Learning Hypothesis

The hypothesis of KNN is that similar songs tend to have the same popularity level.

As such, for a new song x:

$$h(x) = majority(y_i : x_i \in N_k(x)) \tag{7}$$

Where $N_k(x)$ is the set of k-nearest neighbours of x. The distance similarity between them is calculated using the Euclidian distance equation:

$$d(x, x_i) = \sqrt{\sum_{j=1}^{4} (x_j - x_{ij})^2} \tag{8}$$

and the prediction through:

$$h(x) = \arg \left(\max_{c \in \{0,1\}} \sum_{x_i \in N_k(x)} \mathbf{1}(y_i = c) \right) \tag{9}$$

Which is meant to count how many neighbours have label 0 or 1 and then choose the most frequent answer.

1.5 Learned Function Representation

Unlike regression or neural networks, KNN does not create a parameterized function. Instead, it uses (9) to create a model that stores a training matrix X with the label vector y and a hyperparameter k. Thus, the model representation is:

$$M = (X, y, k) \tag{10}$$

1.6 Algorithm

1.6.1 Training Phase - Lazy Learning

Training consists of storing the data

$$X_{train} = X, \qquad y_{train} = y \tag{11}$$

with little preprocessing by substituting zeros to prevent division by 0 in the future.

1.6.2 Prediction Phase

For each test sample x, we compute the distances to all training samples

$$d_i = d(x, x_i) \tag{12}$$

and take the indices of the k nearest ones

$$I_k = argsort(d_i)[1 : k] \tag{13}$$

After this, we extract the labels of neighbours

$$Y_k = y_i : i \in I_k \tag{14}$$

where we perform the majority on and then output the predicted result.

2 Unsupervised Learning - K-Means Clustering

Tracks from the dataset will be clustered into groups using the K-Means Clustering method.

The method begins by choosing a natural number K[IBM]. In our case, K the algorithm will be tried with different values. Our options include using 35, the number of distinct genres present in the dataset. Another option is 21, the number of distinct genres present in the dataset with a share of more than 1% of the entries. From that point on, K random points from the dataset will be chosen as initial centroids. Using the Euclidean distance function, each point in the dataset will get assigned to the centroid closest to itself. A two-step iterative process will commence, where the points in each cluster will be used to compute the cluster's new centroid and then will get reassigned to the clusters around these new points. The algorithm stops when the centroids stop moving from one iteration to another. We will say at that point that our system has converged.

Considering the fact that the K-Means Clustering only converges to a local optimum [MIT], the experiment will be repeated a number of times to minimize the potential effect of unlucky starting positions.

2.1 Input Space

The input space will consist of track features. All features will be scaled and shifted to follow a normal distribution. This ensures that distance-based comparisons are meaningful.

$$
\begin{aligned}
x = \ & (\\
& \text{acousticness},\ \text{danceability},\ \text{energy}, \\
& \text{instrumentalness},\ \text{key},\ \text{liveness},\ \text{loudness}, \\
& \text{mode},\ \text{speechiness},\ \text{tempo},\ \text{time\_signature}, \\
& \text{valence},\ \text{duration\_ms} \\
&) \in \mathbf{R}^{13}
\end{aligned}
$$

2.2 Output Space

The output will consist of a single natural number uniquely associating the input track to one of the clusters. Each cluster will have a number associated at random from the range $\overline{1, K}$.

$$y \in \{1,\ 2,\ ...,\ K\}$$

2.3 Formal Target Definition

The complete system should be able to decide whether or not two given tracks, x_i and x_j, are part of the same genre.

$$
\begin{aligned}
& f : \mathbf{R}^{13} \times \mathbf{R}^{13} \to \{\text{true}, \text{false}\} \text{ with} \\
& f(x_i, x_j) = \text{true if } x_i \text{ and } x_j \text{ are part of the same genre, false otherwise}
\end{aligned}
$$

2.4 Learning Hypothesis

The K-Means hypothesis is that musical tracks that are similar in their audio features belong to the same conceptual group. In our case, we evaluate wether the clusters correspond to known musical genres.

Thus, clustering aims to assign each sample x_i to the cluster whose centroid μ_k minimizes the distance:

$$h(x_i) = \arg\min_{k=\overline{1,K}} d(x_i, \mu_k)$$

where $d(\cdot)$ is the Euclidean distance. The algorithm assumes clusters are convex and roughly spherical in shape.

2.5 Learned Function Representation

The learned model produced by K-Means consists of the final cluster centroids:

$$M = \{\mu_1, \mu_2, ..., \mu_K\}$$

Each centroid $\mu_k \in \mathbf{R}^{13}$ represents the mean of all samples assigned to that cluster.

For any new data point x, the predicted cluster assignment is the index of the closest centroid:

$$h(x) = \arg\min_{k=\overline{1,K}} d(x, \mu_k)$$

2.6 Algorithm

2.6.1 Initialization

Randomly select K distinct points from the dataset as the initial centroids:

$$\mu_1, \mu_2, ..., \mu_K$$

2.6.2 Assignment Step

Assign each sample x_i to the nearest centroid:

$$c_i = \arg\min_{k=\overline{1,K}} d(x_i, \mu_k)$$

2.6.3 Update Step

Recompute each centroid as the mean of all points assigned to it:

$$\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$$

where C_k is the set of points assigned to cluster k.

2.6.4 Convergence

Repeat the assignment and update steps until convergence, when centroids stop changing between iterations. This convergence is guaranteed to happen, albeit not to a global optimum.

References

[IBM] IBM Presentation on K-Means Clustering. accessed December 2025, online at https://www.ibm.com/think/topics/k-means-clustering.

[MIT] MIT 6002, Lecture 12 from 2016: Clustering. accessed November 2025, online at https://www.youtube.com/watch?v=esmzYhuFnds.