

## U2 - Implementación KNN

The KNN is known as the algorithm of the nearest neighborhood that generally is used like an algorithm of classification starting from the assumption that similar points can be found close to each other and the classification is used for discrete values. It is commonly used for simple recommender systems, pattern recognition, data mining, financial market predictions, intrusion detection and more.

For classification problems, a class label is assigned based on a majority vote, i.e., the label that is most frequently represented around a given data point is used. While this is technically considered "majority voting," the term "majority vote" is more commonly used in the literature. The distinction between these terminologies is that "majority vote" technically requires a majority greater than 50%, which works primarily when there are only two categories. When you have multiple classes, e.g., four categories, you do not necessarily need 50 % of the vote to reach a conclusion on a class; you can assign a class label with a vote greater than 25 %.

Instead of using a loss function and optimization K-NN takes a "lazy learning" models, which means that it only stores a training data set instead of going through a training stage. This also means that all computation occurs when a classification or prediction is performed. Because it relies heavily on memory to store all of its training data, it is also referred to as an instance-based or memory-based learning method.

The distance metric used, the value of  $k$  (the number of neighbors to consider) and other hyperparameters are important aspects to consider when using the  $k$ -NN algorithm, but they do not involve a loss function or optimization as in other supervised algorithms. Instead,  $k$ -NN relies on a distance metric, such as Euclidean distance or Manhattan distance, to determine the similarity between instances and make decisions based on that similarity.

The  $k$ -NN algorithm has been used in a variety of applications, primarily within classification. Some of these packages include:

- -Data preprocessing: Data sets often have missing values, but the KNN algorithm can estimate those values in a process known as missing data imputation.
- -Recommendation engines : using website clickstream data, the KNN algorithm has been used to provide automatic recommendations to users for additional content. Shows that a user is assigned to a particular group and, based on the user behavior of that group, is given a recommendation. However, given the scaling issues with KNN, this approach may not be optimal for larger data sets.
- Healthcare: KNN has been applied within the healthcare industry, making predictions about the risk of heart attacks and prostate cancer. The algorithm works by calculating the most likely gene expressions.
  - Pattern recognition: KNN has also helped identify patterns, such as in text and digit classification. This has been particularly useful for identifying handwritten numbers that you may find on forms or mailing envelopes.

**Code:**

```
# 1. Import the necessary libraries: `pandas` for data manipulation
and `KNeighborsClassifier` from `sklearn.neighbors` for k-NN
classification.
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier

# 2. Create an example dataset:
#     - Define a dictionary `data` containing columns
'Consumption_Pizza_Mont', 'Consumption_Hamburguesa_Mont', and
'preferred_taste' with sample data.
data = {'Consumption_Pizza_Mont': [4, 2, 1, 5, 6, 7, 8],
        'Consumption_Hamburguesa_Mont': [2, 5, 4, 1, 3, 8, 10],
        'preferred_taste': ['Pizza', 'Hamburguesa', 'Hamburguesa',
'Pizza', 'Pizza', 'Hamburguesa', 'Hamburguesa']}

# 3. Create a pandas DataFrame:
#     - Use `pd.DataFrame(data)` to create a DataFrame `df` from the
sample data.
df = pd.DataFrame(data)

# 4. Split the features (X) and the target variable (y):
#     - Extract columns 'Consumption_Pizza_Mont' and
'Consumption_Hamburguesa_Mont' into `X`.
#     - Extract the 'preferred_taste' column into `y`.
X = df[['Consumption_Pizza_Mont', 'Consumption_Hamburguesa_Mont']]
y = df['preferred_taste']

# 5. Create a k-NN classifier with k=3:
#     - Initialize a k-NN classifier `knn` with 3 neighbors.
knn = KNeighborsClassifier(n_neighbors=3)

# 6. Train the classifier:
#     - Use `knn.fit(X, y)` to train the k-NN classifier using the
provided data.
knn.fit(X, y)

# 7. Create a new data point for predicting preferred taste:
#     - Define `new_point` as a list with a new data point, e.g., `[3,
4]`, representing pizza consumption of 3 and hamburguesa consumption
of 4 in a month.
new_point = [[3, 4]]

# 8. Make a prediction:
```

Student's name: Hernando Enrique Te Bencomo

Professor: Victor Alejandro Ortiz Santiago

Grade and group: 9°B

```
# - Use `knn.predict(new_point)` to predict the preferred taste
based on the new data point.
prediction = knn.predict(new_point)

# 9. Display the prediction:
# - Check the predicted value and print whether the person prefers
pizza or hamburguesa based on the prediction.
if prediction[0] == 'Pizza':
    print("The person prefers pizza more.")
else:
    print("The person prefers hamburguesa more.")
```