Machine Learning Lab

Experiment – 5

Ankith Motha

21BAI1284

k-NN Clustering

**Dataset:** Heart Attack Dataset

**Input Variables:** age, sex, cp, trtbps, chol, fbs, restecg, thalachh, exng, oldpeak, slp, caa, thall.

**Target Variable:** Output

**About this dataset**

* Age : Age of the patient
* Sex : Sex of the patient
* exang: exercise induced angina (1 = yes; 0 = no)
* ca: number of major vessels (0-3)
* cp : Chest Pain type chest pain type
  + Value 1: typical angina
  + Value 2: atypical angina
  + Value 3: non-anginal pain
  + Value 4: asymptomatic
* trtbps : resting blood pressure (in mm Hg)
* chol : cholestoral in mg/dl fetched via BMI sensor
* fbs : (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
* rest\_ecg : resting electrocardiographic results
  + Value 0: normal
  + Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
  + Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria.
* thalach : maximum heart rate achieved
* target :
  + 0 - less chance of heart attack;
  + 1 - more chance of heart attack

**Overview:**

The k-nearest neighbours (KNN) algorithm is a non-parametric, lazy learning algorithm that is used for both classification and regression. It works by finding the k most similar instances in the training set to a new instance, and then assigning the new instance to the class that is most common among the k nearest neighbours.

The k value is a hyperparameter that must be chosen by the user. A higher value of k will give more weight to the more distant neighbours, while a lower value of k will give more weight to the closer neighbours. The optimal value of k will depend on the specific dataset and the desired accuracy.

KNN is a simple algorithm to understand and implement, and it is often used as a baseline algorithm for comparison with other machine learning algorithms. However, it can be computationally expensive to calculate the distances between all pairs of instances in the training set, and it can be sensitive to noise in the data.

Here are the steps on how KNN works:

1. Choose the k value. The k value is a hyperparameter that must be chosen by the user. A higher value of k will give more weight to the more distant neighbours, while a lower value of k will give more weight to the closer neighbours. The optimal value of k will depend on the specific dataset and the desired accuracy.
2. Find the k nearest neighbours. For each new instance, find the k instances in the training set that are most similar to the new instance. This can be done using any distance metric, such as the Euclidean distance or the Manhattan distance.
3. Predict the class of the new instance. The class of the new instance is predicted by the majority vote of its k nearest neighbours. For example, if 3 of the k nearest neighbours are of class A and 2 are of class B, then the new instance is predicted to be of class A.

Here are some of the advantages of KNN:

* It is a simple algorithm to understand and implement.
* It is a non-parametric algorithm, which means that it does not make any assumptions about the underlying distribution of the data.
* It can be used for both classification and regression problems.
* It is a robust algorithm that is not sensitive to noise in the data.

Here are some of the disadvantages of KNN:

* It can be computationally expensive to calculate the distances between all pairs of instances in the training set.
* It can be sensitive to the choice of the hyperparameter k.
* It can be slow to predict new instances, especially for large datasets.

Overall, KNN is a simple and powerful machine learning algorithm that can be used for a variety of tasks. It is a good choice for beginners who are learning about machine learning, and it can also be used as a baseline algorithm for comparison with other machine learning algorithms.

1. **Workflow of KNN**

Initially, Let’s consider 20 samples from the training set of our heart disease dataset and let’s apply the KNN algorithm with K-neighbours = 3 on a testing sample.

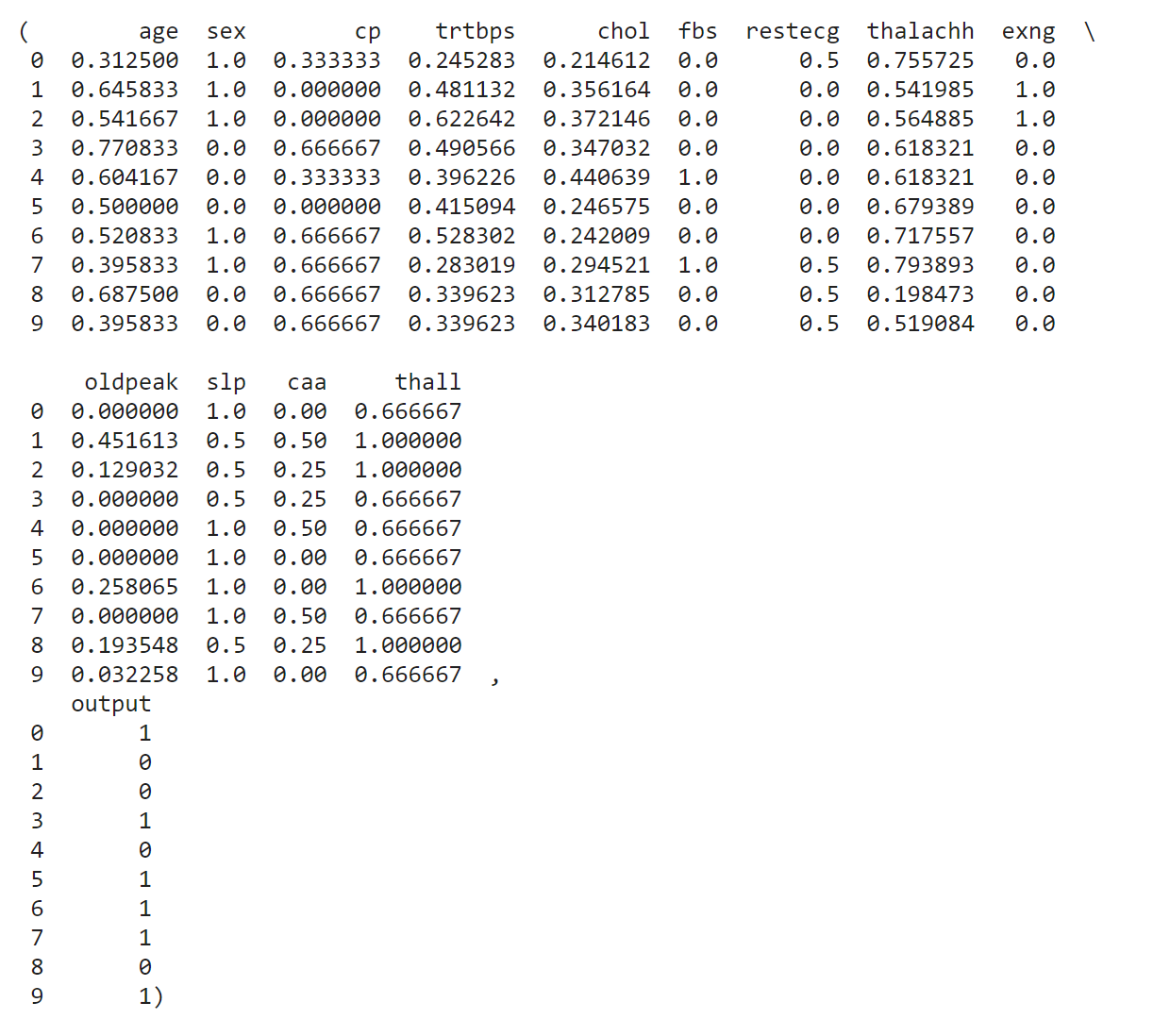
**Test Sample:**

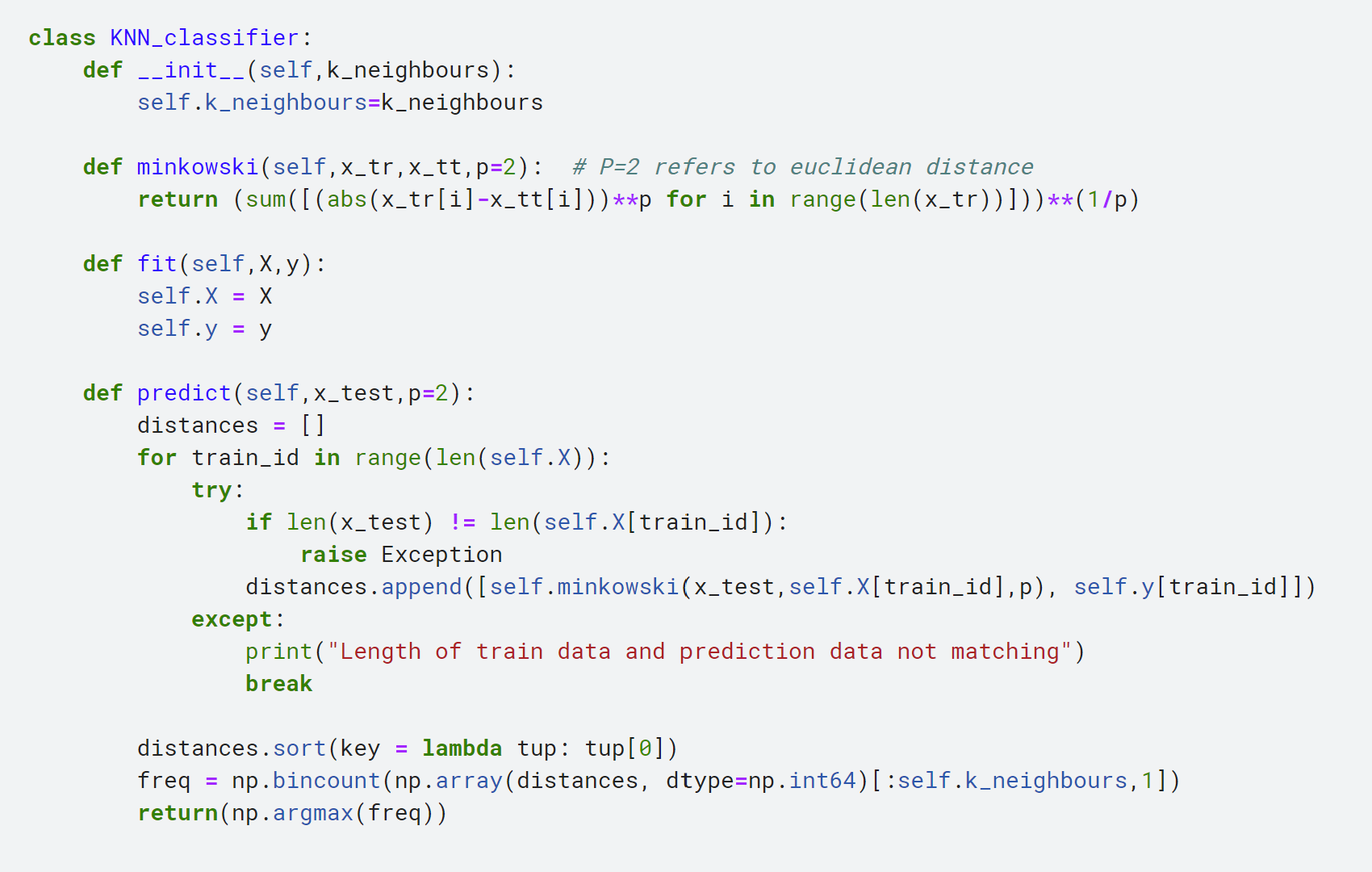
**Input:** [0.47916667, 1., 1., 0.22641509, 0.1369863, 0., 0., 0.90839695, 0., 0., 0.5, 0., 0.33333333]

**Expected Result:** 1

**Training Data for workflow from Training set**



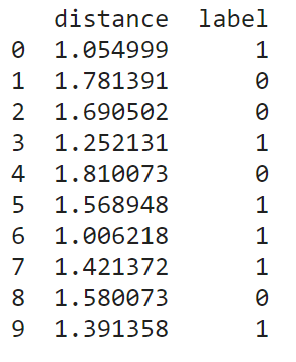




As we pass the testing sample as input to the classifier for prediction,

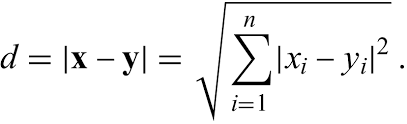
[0.47916667, 1., 1., 0.22641509, 0.1369863, 0., 0., 0.90839695, 0., 0., 0.5, 0., 0.33333333]

We first calculate the distances of the input test vector with the other training data



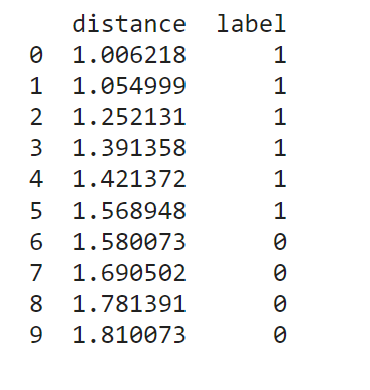
The distances of the test sample with all the other 10 training samples is calculated using

Euclidean distance,



Where x(i) is a feature value of test sample and y(i) is the corresponding feature value of the train sample. “n” is the total number of features in the input.

Next, sort the distances in ascending order and chose the ‘**k’** nearest neighbours of the testing data based on the sorted distance**.**



It can be seen that the distances have been sorted in ascending order.

We need to now choose the k nearest neighbours, i.e, the first 3 training points which are closest to test point.

**Output**:

[[1.00621776, 1.]

[1.0549994, 1.]

[1.25213052, 1.]]

Using the principle of **MAX VOTING**, we will choose the labels which are more frequently appearing among the **k nearest neighbours**. This max label will be assigned to the test data.

**Output:**

[0 3]

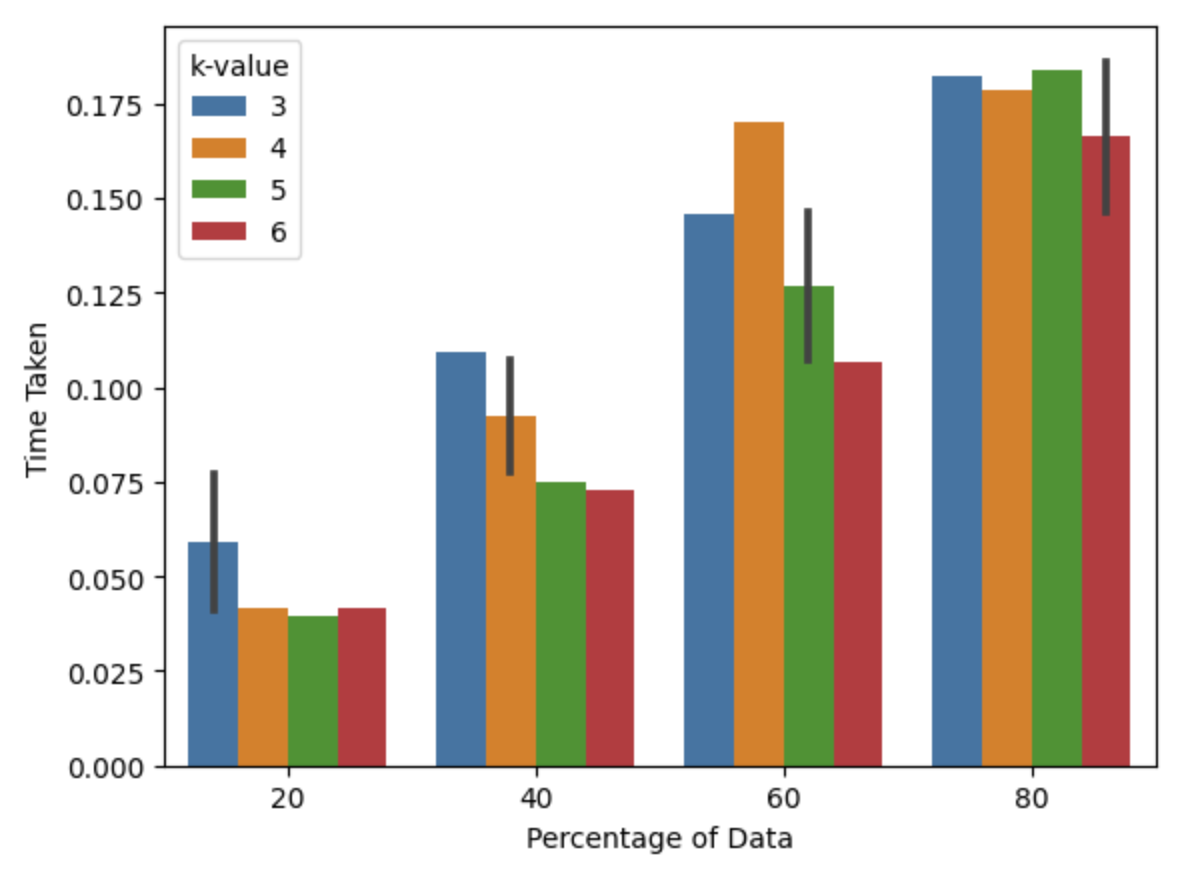
The above results conveys the meaning that there are 3 tuples with Label 1 as answer, and 0 tuples with Label 0. Hence by max voting, the test data will get 1 as the label.

On comparing the true label and predicted label, we find that both are the same **(Label: 1).**

1. **Comparative Analysis**

**1&2) Time Analysis for Train Data vs K along with 30% Test Data(in secs):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Size of Training Data \ K Neighbours | K = 3 | K = 4 | K = 5 | K = 6 |
| 20% | 0.041011325 | 0.041398121 | 0.039613125 | 0.041703245 |
| 40% | 0.077000163 | 0.077383058 | 0.074809444 | 0.072773152 |
| 60% | 0.109130311 | 0.107329887 | 0.107174437 | 0.106888516 |
| 80% | 0.145766027 | 0.169836267 | 0.146234051 | 0.146342354 |
| 100% | 0.182187114 | 0.178715758 | 0.183862835 | 0.186098923 |



**3) Testing Data with Different k-values:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment** | **Precision** | **Recall** | **F1-Score** | **Accuracy** |
| K = 3 | 0.7778 | 0.8571 | 0.8155 | 0.7912 |
| K = 4 | 0.8261 | 0.7755 | 0.8 | 0.7912 |
| K = 5 | 0.7885 | 0.8367 | 0.8119 | 0.7912 |
| K = 6 | 0.8163 | 0.8163 | 0.8163 | 0.8022 |

