NYCU

National Yang Ming Chiao Tung University

Data Mining
HW3

Sylvain Thong (312551818) **sthong.cs12@nycu.edu.tw**

28/05/2024 - Version 1.1

Table of contents :

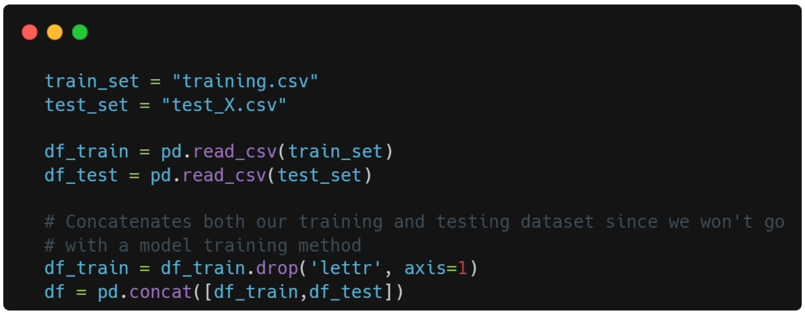
I - Implementation choice

With this homework 3, we decided to go with the unsupervised K-Nearest Neighbors method in order to distinct outliers from our original 6 letters UCI Letter Image Recognition training dataset.
This has the advantage of not requiring any form of model training as it is solely based on the chosen distance metric and the number of neighbors we consider for evaluating this distance. All the while providing great results for this use case.

It should be considered that this method is great for smaller datasets and can be really **computationally expensive on larger ones with a quadratic complexity** when computing our distance matrix!

A) Preprocessing

As the unsupervised KNN method does not require any form of training, we concatenate both our training and testing dataset including the outliers.



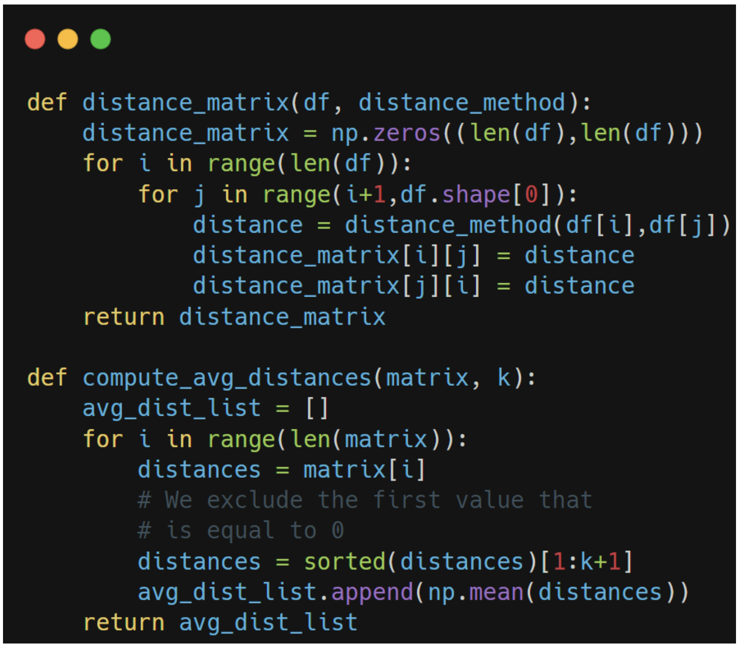
```
train_set = "training.csv"
test_set = "test_X.csv"

df_train = pd.read_csv(train_set)
df_test = pd.read_csv(test_set)

# Concatenates both our training and testing dataset since we won't go
# with a model training method
df_train = df_train.drop('lettr', axis=1)
df = pd.concat([df_train,df_test])
```

Afterwards, in order to take into account the range of values that are not equivalent amongst the features, **we normalize our values per column.**

B) Distance matrix and average distance

As our whole dataset “only” comprises 5200 rows, we can compute a distance matrix that registers the distances between each data point. The operation is quite straightforward but we also need to decide **which distance metric we are going to adopt for the best performance (which we will do in the next part).**

For each data point, we select the k-nearest data points and take the mean value of their distances. This will be our value that allows us to determine whether a data point is an outlier or not when selecting a threshold.



```python
def distance_matrix(df, distance_method):
    distance_matrix = np.zeros((len(df),len(df)))
    for i in range(len(df)):
        for j in range(i+1,df.shape[0]):
            distance = distance_method(df[i],df[j])
            distance_matrix[i][j] = distance
            distance_matrix[j][i] = distance
    return distance_matrix

def compute_avg_distances(matrix, k):
    avg_dist_list = []
    for i in range(len(matrix)):
        distances = matrix[i]
        # We exclude the first value that
        # is equal to 0
        distances = sorted(distances)[1:k+1]
        avg_dist_list.append(np.mean(distances))
    return avg_dist_list
```

C) Distance metric selection

In order to determine which distance metric is the most suitable for our outliers detection's task, we need to consider the number of features we have as it describes the dimensionality of our problem and also their natures!

Since our data points are represented as real-valued vectors, we can choose the Minkowski's distance as a base :

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}.$$

We then need to determine the value of p that is best suited to our problem in order to achieve best performances.

II - Area under the ROC Curve or F1 Score ?

Our predictions are evaluated by Kaggle via the Area under the ROC Curve (AUC) metric over the F1 Score that was used in the previous homework. But why is that ?

The purpose of this homework is to detect anomalies/outliers amongst our testing dataset that comprises “regular” data and outliers. **Therefore, we need to correctly assert the number of misclassified instances but also those that are correctly classified.**

A) F1 Score

The F1 Score is computed by combining both the Recall and the Precision metrics :

$$\text{Precision} = \frac{\text{Relevant retrieved instances}}{\text{All retrieved instances}}$$

$$\text{Recall} = \frac{\text{Relevant retrieved instances}}{\text{All relevant instances}}$$

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

However, due to how the F1 Score is computed, **it doesn't take into account the number of true negatives** (instances that were classified as negative/outliers and that are actual outliers). This is problematic for an anomaly detection system evaluation since it is our main target for assessing the performance of such a classifier…

B) AUC

On the other hand, AUC is computed in a way that it should better assess an anomaly detector's performance : in a nutshell, **it determines whether the system will rank a randomly chosen positive instance higher than a randomly chosen outlier instance,** straight to an anomaly detector's main purpose. Moreover, **it does take into account the number of true negatives (TN)** as the ROC Curve is based on the false positive rate (FPR) :

$$FPR = \frac{FP}{FP + TN}$$

Therefore, it is the most prized choice for evaluating an anomaly's detector nowadays.

III - Semi-supervised or unsupervised learning ?

In this homework, the training dataset was made of labeled instances whereas the testing one only had unlabeled entries. This could have suggested that we could take a semi-supervised learning approach in order to tackle the anomaly detection's task. But what are the main differences with an unsupervised approach ?

A) Key differences

On one hand, unsupervised learning involves training algorithms on **datasets that do not have labeled responses** : therefore, the primary goal is to find hidden patterns in the input data to be able to cluster the data.
On the other hand, semi-supervised learning falls between supervised and unsupervised learning : it involves **using a small amount of labeled data along with a large amount of unlabeled data.** The goal is to leverage the labeled data to guide the learning process and improve the performance on the unlabeled data.

B) Pros and cons

Due to their difference in nature, both approaches can lead to different results and tackle different problems efficiently.

- Unsupervised learning is ideal when no labeled data is available and the goal is to explore the data structure when we don’t know about the nature/number of clusters to find.
- Semi-supervised learning is beneficial when there is some labeled data available but not enough to train a robust model, allowing the leveraging of a large pool of unlabeled data to improve model performance.

In our case, leveraging our labeled training data did not really have a purpose with the use of a KNN method to cluster our dataset. Hence why, we chose to get rid of our label in the training dataset in our preprocessing pipeline.