KNN

This repository include an implementation of the KNN algorithm in python.

Submitters:

Yishay Seroussi 305027948, Samuel Bismuth 342533064.

Python version:

3.9

Configuration

This repository includes an implementation of the knn alogithm in python.

We use the docker environement. Make sure docker is installed in you machine. That is the only dependency of the project.

According to your distribution, run:

```
sudo <yum/apt-get> install -y docker
```

Then, to run the script run:

```
bash start.sh
```

To enter in the container terminal (only for developement purpose):

```
bash bash.sh
```

If you don't want to use docker, you are able to run the code in any machine by folowing the next steps:

Install python3.9.

Install numpy by running:

```
 pip3 install numpy
pip3 install sklearn
```

Run the main python file:

```
python3 main.py
```

Code structure:

The code is composed of three folders.

- The data folder containing the txt file of data received for the assignment.
- The packages folder containing the requirement txt file with the pip lib we used (numpy, sklearn).
- The src folder containing the code source.
  - main.py -> The main file of the code. This is our entrypoint. This is also the file were the prints are done.
  - data.py -> The file handle the txt data to convert it into objects.
  - knn.py -> Here you can find the main algorithm with the computation of the final error and error.

Example of outputs:

```
##############################################

Train -> k: 1, p: 1, error: 0.0316923076923076
Test -> k: 1, p: 1, error: 0.46723076923076895

##############################################

##############################################

Train -> k: 1, p: 2, error: 0.027076923076923026
Test -> k: 1, p: 2, error: 0.46476923076923043

##############################################
```

```
#############################################

Train -> k: 1, p: inf, error: 0.026461538461538398
Test -> k: 1, p: inf, error: 0.46815384615384587

#############################################

#############################################

Train -> k: 3, p: 1, error: 0.22184615384615347
Test -> k: 3, p: 1, error: 0.43923076923076876

#############################################

#############################################

Train -> k: 3, p: 2, error: 0.23338461538461502
Test -> k: 3, p: 2, error: 0.4476923076923074

#############################################

#############################################

Train -> k: 3, p: inf, error: 0.23723076923076886
Test -> k: 3, p: inf, error: 0.4612307692307691

#############################################

#############################################

Train -> k: 5, p: 1, error: 0.2764615384615381
Test -> k: 5, p: 1, error: 0.4656923076923075

#############################################

#############################################

Train -> k: 5, p: 2, error: 0.2830769230769227
Test -> k: 5, p: 2, error: 0.46076923076923054

#############################################

#############################################

Train -> k: 5, p: inf, error: 0.2867692307692304
Test -> k: 5, p: inf, error: 0.45861538461538437

#############################################

#############################################

Train -> k: 7, p: 1, error: 0.30784615384615344
Test -> k: 7, p: 1, error: 0.4743076923076921

#############################################

#############################################

Train -> k: 7, p: 2, error: 0.3081538461538457
Test -> k: 7, p: 2, error: 0.4781538461538459

#############################################

#############################################

Train -> k: 7, p: inf, error: 0.3130769230769227
Test -> k: 7, p: inf, error: 0.48138461538461513

#############################################

#############################################

Train -> k: 9, p: 1, error: 0.340307692307692
Test -> k: 9, p: 1, error: 0.47384615384615336

#############################################

#############################################
```

```
Train -> k: 9, p: 2, error: 0.3370769230769226
Test -> k: 9, p: 2, error: 0.4763076923076919

############################################

############################################

Train -> k: 9, p: inf, error: 0.34784615384615347
Test -> k: 9, p: inf, error: 0.4787692307692305

############################################
```

Which parameters of k,p are the best?

When k=3 and p=1, we reach a minimal error.

Do you see overfitting?

err(train)/err(test) is high for each k and p. err(train)/err(test) is growing up with the number k. That is, the overfitting is smaller when k is equal to 9.

Work division

We worked on this code together using one computer as a pair programming. That is, we've handle and understand together the complexity of the knn implementation and the code design in python. There is nothing in this work that have been done only by one submitter. Notice that we worked only on one github account since we used only one computer.