Ex3.DBScanClustering

January 16, 2024

1 3rd exercise: Do DBScan clustering for anomaly detection

- Course: AML
- Lecturer: Gernot Heisenberg
- Author of notebook: Finn Heydemann
- Date: 28.10.2023

GENERAL NOTE 1: Please make sure you are reading the entire notebook, since it contains a lot of information on your tasks (e.g. regarding the set of certain paramaters or a specific computational trick), and the written mark downs as well as comments contain a lot of information on how things work together as a whole.
GENERAL NOTE 2: * Please, when commenting source code, just use English language only. * When describing an observation please use English language, too * This applies to all exercises throughout this course.

1.0.1 DESCRIPTION:

This notebook allows you for using the DBScan clustering algorithm for anomaly detection.

1.0.2 TASKS:

The tasks that you need to work on within this notebook are always indicated below as bullet points. If a task is more challenging and consists of several steps, this is indicated as well. Make sure you have worked down the task list and commented your doings. This should be done by using markdown. Make sure you don't forget to specify your name and your matriculation number in the notebook.

YOUR TASKS in this exercise are as follows: 1. import the notebook to Google Colab or use your local machine. 2. make sure you specified you name and your matriculation number in the header below my name and date. * set the date too and remove mine.

3. read the entire notebook carefully * add comments whereever you feel it necessary for better understanding * run the notebook for the first time. 4. take the three data sets from exercize 1 and cluster them 5. read the following article for getting help estimating eps and minPts * https://stats.stackexchange.com/questions/88872/a-routine-to-choose-eps-and-minpts-for-dbscan 6. describe your findings and interpret the results ——————————————————————

[1]:
```
from sklearn.cluster import DBSCAN
import numpy as np
from numpy.random import randn
np.random.seed(1)
random_data = np.random.randn(50000,2)  * 20 + 20
```

The output of the below code is 94. This is the total number of noisy points. SKLearn labels the noisy points as (-1). The downside with this method is that the higher the dimension, the less accurate it becomes. You also need to make a few assumptions like estimating the right value for eps which can be challenging.

[2]:
```
# hyperparameters
minPts = 2
eps = 3

outlier_detection = DBSCAN(min_samples = minPts, eps = eps)

clusters = outlier_detection.fit_predict(random_data)

list(clusters).count(-1)
```

[2]:
```
94
```

[47]:
```
import pandas as pd

def outlier_detection_dbscan(data: np.ndarray, minPts: int, eps: int):
    if isinstance(data, pd.Series):
        data = data.to_numpy()
    if data.ndim == 1:
        data = data[:, np.newaxis]
    clusters = DBSCAN(min_samples=minPts, eps=eps).fit_predict(data)
    return clusters, list(clusters).count(-1)
```

[63]:
```
data = pd.read_csv("data/exercise_1/StudentsPerformance.csv")

data.head()
```

[63]:
```
   gender race/ethnicity parental level of education         lunch  \
0  female        group B           bachelor's degree      standard
1  female        group C                some college      standard
2  female        group B             master's degree      standard
```

```
3    male        group A          associate's degree  free/reduced
4    male        group C                some college      standard

  test preparation course  math score  reading score  writing score
0                    none          72             72             74
1               completed          69             90             88
2                    none          90             95             93
3                    none          47             57             44
4                    none          76             78             75
```

[83]:
```
clusters, n_outliers = outlier_detection_dbscan(data["math score"], 4, 1)
```

[88]:
```
data[clusters == -1]
```

[88]:
```
     gender race/ethnicity parental level of education         lunch  \
17   female        group B           some high school  free/reduced
59   female        group C           some high school  free/reduced
145  female        group C               some college  free/reduced
338  female        group B           some high school  free/reduced
787  female        group B               some college      standard
842  female        group B                high school  free/reduced
980  female        group B                high school  free/reduced

    test preparation course  math score  reading score  writing score
17                     none          18             32             28
59                     none           0             17             10
145                    none          22             39             33
338                    none          24             38             27
787                    none          19             38             32
842               completed          23             44             36
980                    none           8             24             23
```

Because the math score is an integer an eps smaller than or equal to 0.5 clusters all students with that score except the one which don't have a "partner score". To detect outliers in this case it makes more sense to put eps to 1. 4 as minPts to classify a point as core points works well to group everything except the low mat score. Pretty equal to exercise 1 and 2

[89]:
```
data = pd.read_csv("data/exercise_1/city_temperature.csv")
data.drop(data[data["AvgTemperature"] == -99].index, inplace=True)
data["AvgTemperature"] = (data["AvgTemperature"] - 32) * 5/9
data["date"] = pd.to_datetime(data[["Year", "Month", "Day"]])
data = data.set_index("date").groupby(["City", pd.
 ↪Grouper(freq="D")])["AvgTemperature"].mean()
data = data.reset_index()
```

```
/tmp/ipykernel_63562/3201813435.py:1: DtypeWarning: Columns (2) have mixed
types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv("data/exercise_1/city_temperature.csv")
```

[102]:
```
data_brasilia = data[data["City"] == "Brasilia"]
clusters, n_outliers =␣
 ↪outlier_detection_dbscan(data_brasilia["AvgTemperature"], 10, 0.5)
n_outliers
```

[102]: 7

[103]:
```
data_brasilia.iloc[clusters == -1]
```

[103]:

	City	date	AvgTemperature
406288	Brasilia	1996-06-29	13.666667
406289	Brasilia	1996-06-30	14.333333
407080	Brasilia	1998-09-06	30.000000
408563	Brasilia	2002-10-26	30.944444
409549	Brasilia	2005-07-10	14.333333
414262	Brasilia	2018-06-18	13.388889
414713	Brasilia	2019-09-21	29.555556

[104]:
```
data_pyongyang = data[data["City"] == "Pyongyang"]
clusters, n_outliers =␣
 ↪outlier_detection_dbscan(data_pyongyang["AvgTemperature"], 10, 0.5)
n_outliers
```

[104]: 15

[109]:
```
data_pyongyang[clusters == -1]
```

[109]:

	City	date	AvgTemperature
2010165	Pyongyang	2000-08-17	31.000000
2010309	Pyongyang	2001-01-11	-17.388889
2010310	Pyongyang	2001-01-12	-18.055556
2010311	Pyongyang	2001-01-13	-17.611111
2010312	Pyongyang	2001-01-14	-19.666667
2010313	Pyongyang	2001-01-15	-20.388889
2010314	Pyongyang	2001-01-16	-19.722222
2011396	Pyongyang	2004-01-21	-17.277778
2014160	Pyongyang	2012-02-02	-16.055556
2014495	Pyongyang	2013-01-02	-16.166667
2014496	Pyongyang	2013-01-03	-16.500000
2016334	Pyongyang	2018-01-24	-15.944444
2016523	Pyongyang	2018-08-01	31.666667
2016524	Pyongyang	2018-08-02	31.500000
2016525	Pyongyang	2018-08-03	31.888889

In comparison to the former alogrithms, this algo finds outliers also in non-normal distrubted data. This is because it is independant from the complete dataset (e.g. median, mean etc.) but rather compares the current point to its neigbors.

[ ]: