**Data Science and Machine Learning (MSc)**

DAMA60: Algorithmic Techniques and Systems for Data Science and Machine Learning

Academic Year: 2023–2024

|  |  |  |
| --- | --- | --- |
| #1 Written Assignment | | |
| Submission Deadline | Wed, 15 November 2023, 11:59 PM | |
| Name/StudentId | Kritikos Antonios | Std157978 |

# Remarks

The deadline is definitive.

An indicative solution will be posted online upon the return of the graded assignments.

The assignment is due via the STUDY submission system. **You are expected to turn in a document (.DOC, .ODT, .PDF) and a file containing the Python program:**

* 1 document file (this document) with the answers to all the questions, along with the Python code snippets for Topic 5.
* 1 file with the complete Python program that answers Topic 5 and should execute with the IDLE system.

**You should not make any changes in the written assignment file other than providing your own answers.** You should also type all of your answers into Word and not attach any handwritten notes as pictures into your work, otherwise points will be taken off. Make sure to name all the files (DOC file and Python program file) with **your last name first followed by a dash symbol and the names of each component at the end**. For example, for the student with last name Aggelou the files should be named as follows: Aggelou-HW1.doc (or Aggelou-HW1.pdf), and Aggelou-Topic5.py.

|  |  |  |
| --- | --- | --- |
| Topic | Points | Grades |
| 1. **Online Quiz** | 40 |  |
| 1. **Decision Trees – Gain Ratio** | 15 |  |
| 1. **Association Rules - FP Growth** | 15 |  |
| 1. **Clustering** | 15 |  |
| 1. **Python** | 15 |  |
| **TOTAL** | **100** | **/100** |

# Topic 1: Online Quiz

**(40 points)** Complete the corresponding online quiz available at:

https://study.eap.gr/mod/quiz/view.php?id=26520

You have **one effort** and unlimited time to complete the quiz, up to the submission deadline.

# Topic 2: Decision Trees – Gain Ratio

**(15 total points)** The following training set contains 20 instances with 3 categorical attributes (Gender, Region, Occupation), 1 discretized continuous attribute (Income) and the Class (Has Laptop).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Gender** | **Region** | **Occupation** | **Income** | **Has Laptop** |
| 1 | male | city | student | ≤ 9000 | no |
| 2 | female | city | teacher | > 21000 | no |
| 3 | male | countryside | banker | > 21000 | yes |
| 4 | male | countryside | teacher | > 21000 | no |
| 5 | male | city | student | ≤ 9000 | no |
| 6 | female | city | banker | 9000…21000 | yes |
| 7 | female | city | student | ≤ 9000 | yes |
| 8 | male | city | student | ≤ 9000 | yes |
| 9 | female | countryside | teacher | 9000…21000 | no |
| 10 | female | city | student | ≤ 9000 | yes |
| 11 | male | city | student | 9000…21000 | no |
| 12 | female | city | student | 9000…21000 | yes |
| 13 | female | countryside | banker | 9000…21000 | yes |
| 14 | male | countryside | banker | > 21000 | no |
| 15 | male | city | student | ≤ 9000 | yes |
| 16 | male | city | officer | > 21000 | yes |
| 17 | male | city | student | ≤ 9000 | yes |
| 18 | female | countryside | officer | > 21000 | yes |
| 19 | male | city | teacher | 9000…21000 | no |
| 20 | male | city | banker | > 21000 | no |

**(a) (8 points)** In the following table fill in **the missing values** (“**?**”) using an accuracy of 4 decimal digits.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Information Gain | Split Information | Gain Ratio |
| Gender | **0.0803** | 0.9710 | **0.0827** |
| Region | 0.0031 | **0.8813** | **0.0035** |
| Occupation | **0.3368** | 1.8150 | **0.1856** |
| Income | 0.0458 | **1.5813** | **0.0290** |

**(b) (3 points)** Which **attribute** will be used as the root node of the decision tree, based on the gain ratio? **Please provide one sentence as a justification of your answer.**

**Answer: The attribute selected as the root node is the one with the highest information gain or gain ratio, which in both cases here is the attribute Occupation.**

**(c) (4 points)** Draw the first level of the decision tree after deciding which attribute will be used as the root node.

**Answer:**

# Topic 3: Association Rules – FP Growth

**(15 total points)** Use the FP-Growth algorithm and the transactional database below to answer the following questions. Assume a support count threshold of s = 4 transactions.

|  |  |
| --- | --- |
| TID | Transaction |
| 1 | {A, B, D} |
| 2 | {A, D, F} |
| 3 | {B, D, F} |
| 4 | {B, E, F} |
| 5 | {A, B, E, F} |
| 6 | {B, C, D, F} |
| 7 | {A, B, C, D, F} |
| 8 | {B, C, D, E, F} |

**(a) (1 point)** Complete the following table with the support count of each item.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Item** | **A** | **B** | **C** | **D** | **E** | **F** |
| **Support count** | 4 | 7 | 3 | 6 | 3 | 7 |

**(b) (1 point)** Sort items in each transaction by decreasing support count, after discarding infrequent items. If more items within a transaction have the same support count, sort them lexicographically.

|  |  |
| --- | --- |
| TID | Transaction (only frequent items, sorted by decreasing support count) |
| 1 | B D A |
| 2 | F D A |
| 3 | B F D |
| 4 | B F |
| 5 | B F A |
| 6 | B F D |
| 7 | B F D A |
| 8 | B F D |

**(c) (3 points)** Draw the frequent pattern tree (FP-Tree) from the table above.

**(d) (10 points)** Apply the FP-Growth Algorithm in a step–by–step fashion by following the instructions below in order to calculate all frequent itemsets ending in D. Answers should be provided in the empty spaces below.

i) **(4 points)**



|  |  |
| --- | --- |
| **Draw the prefix paths ending in D** | **Draw the conditional FP-tree for D** |
|  |  |
| **List the frequent itemsets generated at this step, along with their support count:**  **B F D with support count 4**  **B F with support count 6**  **B D with support count 5**  **F D with support count 5** | **From the conditional FP-Tree for D, we can conclude that we need to determine whether two (2) 2-itemsets and one (1) 3-itemset are frequent. These are, in lexicographical order:**  1st 2-itemset to check: B D  2nd 2-itemset to check: D F  1st 3-itemset to check: B D F |

ii) **(6 points)**

|  |  |
| --- | --- |
| **From the itemsets identified in the previous question, list the frequent itemsets along with their support count:** | **B D with support count 5, D F with support count 5, B D F with support count 4** |

# Topic 4: Clustering

**(15 total points)** In this exercise you will apply the DBSCAN algorithm to a set of data points. Consider the following 10 two-dimensional vectors (points) of data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***P1*** | (3.3, 3.6) |  | ***P6*** | (1.0, 4.1) |
| ***P2*** | (0.7, 4.7) |  | ***P7*** | (3.7, 0.6) |
| ***P3*** | (1.6, 1.5) |  | ***P8*** | (1.3, 3.5) |
| ***P4*** | (4.5, 0.9) |  | ***P9*** | (3.1, 0.8) |
| ***P5*** | (0.8, 3.3) |  | ***P10*** | (4.2, 1.1) |

**(a)** **(3 points)** Using the Euclidean distance, the following distance matrix is calculated. Provide **the six (6) missing elements** (denoted by “**?**”) using an accuracy of 4 decimal digits.

Distance matrix:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***P1*** | ***P2*** | ***P3*** | ***P4*** | ***P5*** | ***P6*** | ***P7*** | ***P8*** | ***P9*** | ***P10*** |
| ***P1*** | 0 | 2.8231 | 2.7019 | 2.9547 | 2.5179 | 2.3537 | 3.0265 | 2.0025 | 2.8071 | 2.6571 |
| ***P2*** |  | 0 | 3.3242 | 5.3740 | 1.4036 | 0.6708 | 5.0804 | 1.3416 | 4.5793 | 5.0210 |
| ***P3*** |  |  | 0 | 2.9614 | 1.9698 | 2.6683 | **2.2847** | 2.0224 | 1.6553 | 2.6306 |
| ***P4*** |  |  |  | 0 | 4.4102 | **4.7424** | 0.8544 | 4.1231 | 1.4036 | **0.3606** |
| ***P5*** |  |  |  |  | 0 | 0.8246 | 3.9623 | 0.5385 | 3.3971 | 4.0497 |
| ***P6*** |  |  |  |  |  | 0 | 4.4204 | **0.6708** | 3.9115 | 4.3863 |
| ***P7*** |  |  |  |  |  |  | 0 | 3.7643 | **0.6325** | 0.7071 |
| ***P8*** |  |  |  |  |  |  |  | 0 | 3.2450 | **3.7643** |
| ***P9*** |  |  |  |  |  |  |  |  | 0 | 1.1402 |
| ***P10*** |  |  |  |  |  |  |  |  |  | 0 |

**(b)** **(8 points)** Assume the parameters Eps = 1.4 and MinPts = 4 for the DBSCAN algorithm. Using these values and the distance matrix apply the DBSCAN algorithm and complete the following matrix. If a point is found to be a **core point** put a **C** in the corresponding position in the matrix for that point. Similarly, if a point is a **border point** put a **B** and if a point is a **noise point** put an **N** in the corresponding position in the matrix for that point.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Point** | ***P1*** | ***P2*** | ***P3*** | ***P4*** | ***P5*** | ***P6*** | ***P7*** | ***P8*** | ***P9*** | ***P10*** |
| **Type of point** | **N** | **B** | **N** | **B** | **B** | **C** | **C** | **C** | **B** | **C** |

**(c)** **(4 points)** Using integer numbers (0, 1, 2, 3, …) as cluster labels complete the following matrix. Use the same cluster label to designate points belonging to the same cluster. Use a value of **-1** to denote points that have not been assigned to any cluster.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Point** | ***P1*** | ***P2*** | ***P3*** | ***P4*** | ***P5*** | ***P6*** | ***P7*** | ***P8*** | ***P9*** | ***P10*** |
| **Cluster** | **-1** | **0** | **-1** | **1** | **0** | **0** | **1** | **0** | **1** | **1** |

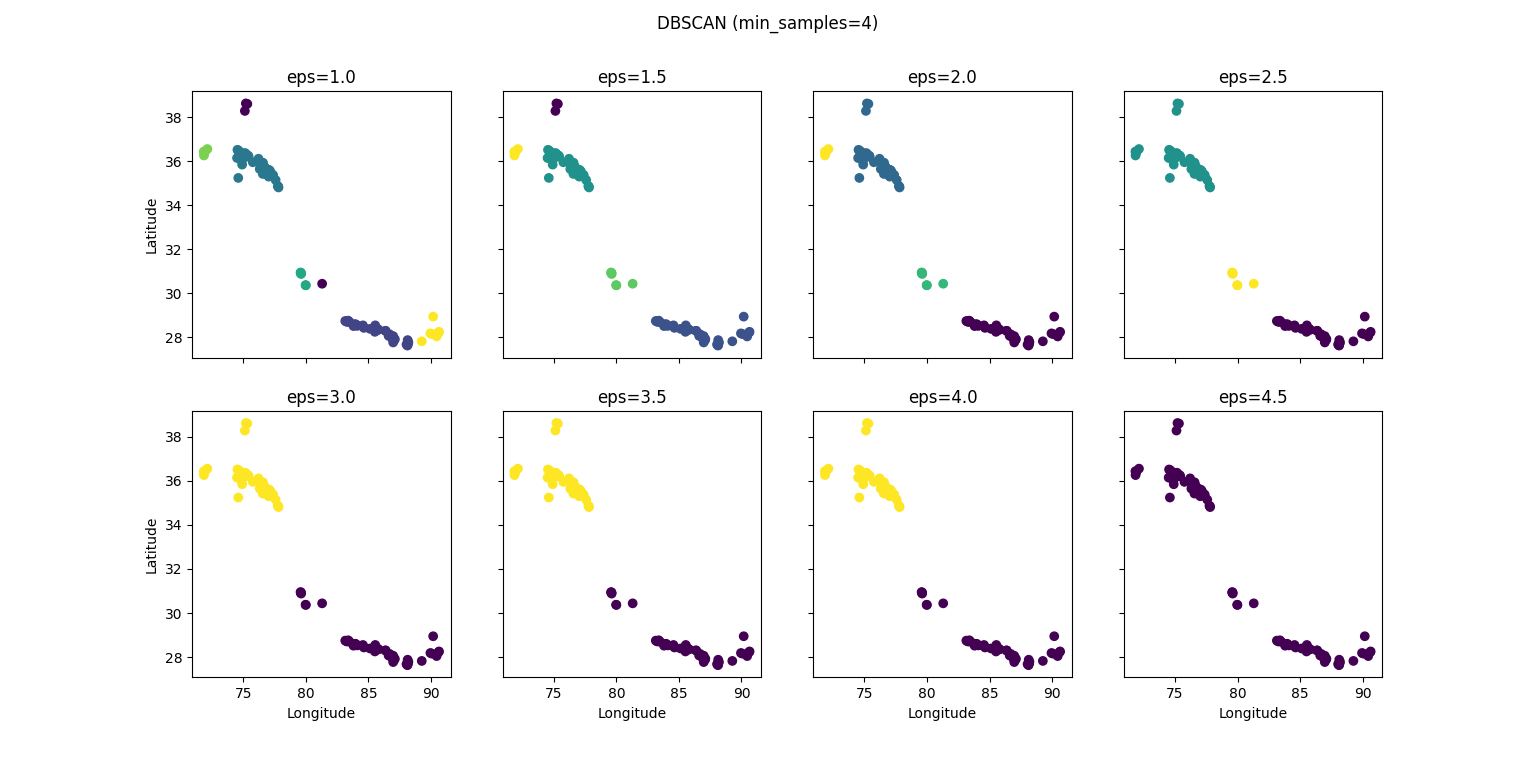
# Topic 5: Python

**(15 total points)** In this exercise you will use libraries of Python to apply DBSCAN to a given set of data and graphically present the results. The data is provided in the “Comma-Separated Values (CSV)” file “mountains.csv”. The file contains information regarding the highest mountain peaks of the Himalayas. Each line in the file includes information for one mountain peak. Despite the name of the format of the input file, fields in each line are allowed to be separated with any character, which is the semicolon (;) in our case. The file can be opened and examined with any text editor.

In the context of this exercise, we are interested in the position of each mountain peak, which is represented through the “longitude” and “latitude” fields of the input file. The purpose of the exercise is to create clusters of mountain peaks that are close together and graphically present the results of the applied clustering algorithm.

The libraries that will be used to complete the exercise are the following: pandas, scikit-learn, matplotlib and math. You are given an incomplete program (named Topic5.py) and must add the missing statements, according to the instructions provided. The pandas library will be used to read the input file and easily extract the required columns of information. The scikit-learn library contains an implementation of DBSCAN that will be used for the exercise. It will be fed with the appropriately formatted data extracted from the input file and the remaining parameters required for DBSCAN. The library matplotlib will be used to graphically present the results. Finally, some mathematical functions from math will be used to calculate parameters for the graphical representation of the results.

To facilitate understanding of the final goal, we provide an image of the results that the completed program must generate:



**(a) (3 points)** Complete the following Python code so as to read the input file into a pandas data frame named “dataframe”. Furthermore, extract the longitude and latitude columns of the data frame and store them in variables x and y respectively.

**Incomplete Python program**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import DBSCAN

# Read input data from file "mountains.csv" into a pandas dataframe

dataframe = pd.read\_csv("mountains.csv", sep=";")

# Extract coordinate x (longitude) and y (latitude) from the dataframe

x = dataframe['longitude']

y = dataframe['latitude']

**(b) (4 points)** Using the appropriate function from the matplotlib library, plot the initial data in a scatter plot. The title of the plot should be 'Highest mountain peaks of the Himalayas'. The labels of the x and y axes should be ‘Longitude’ and ‘Latitude’ respectively. Finally, make the plot appear on the screen.

**Incomplete Python program**

# Plot initial data as a scatter plot

plt.scatter(x, y) # Plot the data in x and y as a scatter plot

plt.title('Highest mountain peaks of the Himalayas')# Add the title

plt.xlabel('Longitude') # Add the label for the x axis

plt. ylabel('Latitude') # Add the label for the y axis

plt.show() # Show the plot

**(c) (2 points)** Complete the missing statements according to the provided comments.

**Incomplete Python program**

# Using variables x and y create a list of corresponding (x,y)

# tuples to represent 2-D coordinates. The resulting list will

# be provided as input to DBSCAN.

data = list(zip(x, y))

# A total number of 8 different values of parameter eps will be tested.

# All results will be presented together in a set of 8 subplots.

# Subplots should be arranged on a 2x4 grid. The following lists will be

# used to create all values of parameter eps and to iterate over all subplots.

i\_list = [i for i in range(0, 2)]

j\_list = [j for j in range(0, 4)]

# Create grid of subplots

fig, ax = plt.subplots(2, 4, figsize=(12, 6))

fig.suptitle('DBSCAN (min\_samples=4)')

**(d) (6 points)** Complete the missing statements according to the comments given.

**Incomplete Python program**

for i in i\_list:

for j in j\_list:

# Calculate parameter eps (starting with a value of 1.0 and

# increasing by 0.5 for each iteration).

e = 1.0 + 0.5 \* (i \* len(j\_list) + j)

# Set desired parameters of DBSCAN algorithm.

# Ensure that the same value of parameter MinPts (= 4) is

# used for each run and that only the value of eps differs.

dbscan = DBSCAN(eps=e, min\_samples=4)

# Apply algorithm with desired parameters to the input data.

dbscan. fit(data)

# Plot clusters of the current iteration as a scatter plot

# in the corresponding subplot.

ax[i, j].scatter(x, y, c=dbscan. labels\_)

ax[i, j].set\_title('eps=' + str(e))

# Create a dictionary to count how many times each cluster label appears,

# i.e., how many points are assigned to each cluster.

for a in ax.flat:

# Set labels for left and bottom plots.

a.set(xlabel='Longitude', ylabel='Latitude')

# Hide x labels and tick labels for top plots and y ticks for right plots.

a.label\_outer()

plt.show() # Show the plot