COMP7107
Assignment 1 Report

ZHANG Hongyi
h1kari@connect.hku.hk
3036408959

February 13, 2025

1. Overview

Python was used to implement the programme and the code consists of two parts. `dataset.py` contains the implementation of the class that handles the input dataset and various queries. And `main.py` is the entry point to the programme. It will create an object from the dataset class and call various functions to complete tasks.

You can use the program as shown below (Python version: 3.8.20):

```
python main.py analysis.data.txt
```

2. Code

2.1. dataset.py

2.1.1. Attribute Mapping

The script maps categorical values to integer representations for easier processing:

- Seasons: “spring”, “summer”, “autumn”, “winter” ⟶ {0, 1, 2, 3}
- River Size: “small*”, “medium”, “large*” ⟶ {0, 1, 2}
- Flow Velocity: “low”, “medium”, “high” ⟶ {0, 1, 2}

Additionally, an attribute_types list defines the type of each attribute:

- 0 ⟶ Nominal (e.g., seasons)
- 1 ⟶ Ordinal (e.g., river size, flow velocity)
- 2 ⟶ Interval (e.g., numeric values)

2.1.2. Class Dataset

The Dataset class is designed to process and analyze the dataset.

1. **Initialization (__init__)**: Reads the dataset from a file (input_data method). Initializes min_values and max_values, which store the min/max values for numeric attributes.
2. **Data Preprocessing**
   - `fix_merged_values(value_with_points: str) → List[str]`: Handles cases where numeric values are incorrectly merged by extracting properly formatted floating-point numbers.
   - `parse_attribute(index: int, raw_value: str) → List[Any]`: Converts categorical values into integers using predefined mappings. Handles missing values (“XXXXXXX”) by returning −1. Parses numeric values, ensuring they are correctly formatted.
   - `input_data(filename: str) → List[Any]`: Reads the dataset line by line. Splits each line into attributes, parsing them based on their type. Handles missing or improperly formatted values. Stores the processed records in self.data.
3. **Data Retrieval & Statistics**
   - `get_data()`: Prints all dataset records.
   - `get_min_values() → List[Any]` / `get_max_values() → List[Any]`: Computes the minimum and maximum values for numeric attributes.
   - `missing_values() → int`: Counts the number of missing values (-1) in the dataset.
4. **Record Formatting**

- `get_record(index: int) → List[Any]` : Retrieves a formatted version of a record. Converts categorical values back to their original labels. Formats numerical values to five decimal places. Replaces missing values with "XXXXXXX".

5. **Similarity & Distance Computation**: The class provides methods to measure similarity and distance between records.
   1. Similarity Measures
      - `get_similarity(index1: int, index2: int) → float` : Computes similarity between two records. For categorical values: assigns 1 if they match, 0 otherwise. For ordinal values: normalizes differences over a scale of 2. For interval values: normalizes differences using min/max values.
      - `get_similarity_with_record(index1: int, record: List[Any]) → float` : Similar to get_similarity(), but compares a dataset record with an external record.
   2. Euclidean Distance Computation
      - `get_euclidean_distance(index1: int, index2: int) → float` : Computes the Euclidean distance between two records (ignores categorical values). Normalizes values using min/max before calculating squared differences.
      - `get_distance_with_record(index: int, record: List[Any]) → float` : Similar to get_euclidean_distance(), but compares a dataset record with an external record.

2.2. main.py

The script is designed to process a dataset using the Dataset class (from dataset.py). It provides three main functionalities:

- Loading and analyzing data (handling missing values and attribute types).
- Computing similarity between dataset records.
- Computing Euclidean distance between dataset records.

The script allows users to interactively select tasks via command-line input.

```
❯ python .\main.py .\analysis.data.txt
Tasks:
1)data loading, cleaning, and transformation
2)computation of similarity
3)computation of Euclidean distance

0)Exit
```

Figure 1: Screenshot of running main.py

3. Results

3.1. Outputs

```
Enter the task number: 1
type of attributes:  [0, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
minimum values per attribute:  [0, 0, 0, 5.6, 1.5, 0.222, 0.05, 5.0, 1.0, 1.0, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
maximum values per attribute:  [3, 2, 2, 9.7, 13.4, 391.5, 45.65, 990.16003, 564.59998, 771.59998, 110.456, 89.8, 72.6, 42.8, 44.6, 44.4, 77.6, 31.6]
total number of missing values:  33
```

Figure 2: Task 1 Output

```
maximum similarity: 0.9813726206923484; pair with maximum similarity: [36, 52]
object 36 = ['winter', 'small_', 'high__', '8.30000', '10.90000', '1.17000', '0.73500', '13.50000', '1.62500', '3.00000', '0.20000', '66.00000', '0.00000', '0.00000', '0.00000',
'0.00000', '0.00000', '0.00000']
object 52 = ['winter', 'small_', 'high__', '7.90000', '11.00000', '6.16700', '1.17200', '18.33300', '7.75000', '11.80000', '0.50000', '81.90000', '0.00000', '0.00000', '0.00000',
 '0.00000', '0.00000', '0.00000']
minimum similarity: 0.49706642196071893; pair with minimum similarity: [19, 152]
object 19 = ['spring', 'small_', 'medium', '7.79000', '3.20000', '64.00000', '2.82200', '777.59961', '564.59998', '771.59998', '4.50000', '0.00000', '0.00000', '0.00000', '44.600
00', '0.00000', '0.00000', '1.40000']
object 152 = ['autumn', 'medium', 'high__', '7.30000', '11.80000', '44.20500', '45.65000', '64.00000', '44.00000', '34.00000', '53.10000', '2.20000', '0.00000', '0.00000', '1.200
00', '5.90000', '77.60000', '0.00000']
highest similarity to input object: 0.9501762982029224; object with highest similarity: 198
object 198 = ['winter', 'large_', 'medium', '8.00000', '7.60000', 'XXXXXXX', 'XXXXXXX', 'XXXXXXX', 'XXXXXXX', 'XXXXXXX', 'XXXXXXX', '0.00000', '12.50000', '3.70000', '1.00000', '
0.00000', '0.00000', '4.90000']
```

Figure 3: Task 2 Output

```
maximum distance: 2.4242045466838524; pair with maximum distance: [19, 152]
object 19 = ['spring', 'small_', 'medium', '7.79000', '3.20000', '64.00000', '2.82200', '777.59961', '564.59998', '771.59998', '4.50000', '0.00000', '0.00000', '0.00000', '44.600
00', '0.00000', '0.00000', '1.40000']
object 152 = ['autumn', 'medium', 'high__', '7.30000', '11.80000', '44.20500', '45.65000', '64.00000', '44.00000', '34.00000', '53.10000', '2.20000', '0.00000', '0.00000', '1.200
00', '5.90000', '77.60000', '0.00000']
minimum distance: 0.07556167716507345; pair with minimum distance: [37, 52]
object 37 = ['spring', 'small_', 'high__', '8.00000', 'XXXXXXX', '1.45000', '0.81000', '10.00000', '2.50000', '3.00000', '0.30000', '75.80000', '0.00000', '0.00000', '0.00000', '
0.00000', '0.00000', '0.00000']
object 52 = ['winter', 'small_', 'high__', '7.90000', '11.00000', '6.16700', '1.17200', '18.33300', '7.75000', '11.80000', '0.50000', '81.90000', '0.00000', '0.00000', '0.00000',
 '0.00000', '0.00000', '0.00000']
smallest distance to input object: 0.04746367117171673; object with smallest distance: 198
object 198 = ['winter', 'large_', 'medium', '8.00000', '7.60000', 'XXXXXXX', 'XXXXXXX', 'XXXXXXX', 'XXXXXXX', 'XXXXXXX', 'XXXXXXX', '0.00000', '12.50000', '3.70000', '1.00000', '
0.00000', '0.00000', '4.90000']
```

Figure 4: Task 3 Output

3.2. Observations

From the pair with maximum similarity, we can see the attributes are mostly zero, which leads to large similarity. But for the pair with minimum similarity, there are few attributes are zero in common, leading to a small similarity.

For the euclidean distance, the situation is similar. We can see that the pair with maximum distance and the pair with minimum similarity is identical. At the same time, the object which has the smallest distance to input object is the one which has the highest similarity to the input object.