Project 3

Association Rule Mining

Note

Project code: `project3.py`

Authors

- Evan Ting-I Lu (tl3098)
- Pitchapa Chantanapongvanij (pc2806)

Files

1. `project3.py` (main program)
2. `README`
3. `example-run.txt`
4. The pre-processed dataset for our program to run on: `./data/INTEGRATED-DATASET.csv`
5. The original dataset for reference: `./data/raw-INTEGRATED-DATASET.csv`

How to run the program

1. Create a new confda environment and install python (tested on Python 3.9)

```
conda create --name 6111_ADB_proj_3 python=3.9
```

2. Activate the conda env

```
conda activate 6111_ADB_proj_3
```

3. Verify Python version

`python --version` (should see `Python 3.9.12`)

4. Install `pandas`

```
conda install pandas
```

5. Clone our project from GitHub (our repo is set to private so please inform us if need access)

```
git clone https://github.com/evantilu/SP22-6111-ADB-Projects.git
```

6. `cd` to project folder

```
cd SP22-6111-ADB-Projects/project_3/
```

Now the program is ready to run

7. Issue `python project_3.py -h` to check options

- Example: run on our INTEGRATED-DATASET.csv with min_sup = 0.01 and min_conf = 0.5

```
python project3.py ./data/INTEGRATED-DATASET.csv 0.01 0.5
```

8. The program will output the result in `output.txt` under the same directory as the program

Description of project

a) NYC Open Data used:

We used the 'Property Valuation and Assessment Data' collected by the City Government. The dataset is a real estate assessment property data which contains the values of properties. This is collected from the Department of Finance when calculating the property tax bills.

This is the link to access the raw dataset:

```
https://data.cityofnewyork.us/City-Government/Property-Valuation-and-Assessment-Data/yjxr-fw8i
```

b) High-level procedure used to map the original NYC Open Data data set into INTEGRATED-DATASET file:

First we removed all the NaN, 0, and empty values from the dataset. That is for any cell that contains either Nan, 0, or empty the entire row will be removed.

Most columns in this dataset is of numeric values. We discretized and categorized the values into buckets (as will be described later) so that it's proper for the Apriori algorithm to run on.

Out of all columns, we selected 16 interesting columns to work with and to extract potential relationships. We bucketized them in order for Apriori to generate interpretable association rules. Below are the 16 columns and the bucketization*:

1. `BORO`: 5 boroughs in New York City included Manhattan, Bronx, Brooklyn, Queens, and Staten Island
   Bor_1: Manhattan
   Bor_2: Bronx
   Bor_3: Brooklyn
   Bor_4: Queens
   Bor_5: Staten Island
2. `STORIES`: Number of stories of a property
   Str_1: Less than or equal to 6 stories
   Str_2: More than 6 and less than or equal to 12 stories
   Str_3: More than 12 stories
3. `FULLVAL`: The value of a property
   Val_1: Less than or equal to 82000
   Val_2: More than 82000 and less than or equal to 1384000
   Val_3: More than 1384000
4. `YEAR`: Assessment year
   Yrs_1: Earlier than or equal to 2013

Yrs_2: Later than 2013 and earlier or equal to 2016
Yrs_3: Later than 2016

5. `LOT`: Number of parking spaces
   Lot_1: Less than or equal to 61
   Lot_2: More than 61 and less than or equal to 1126
   Lot_3: More than 1126
6. `BLDGCL`: Building class
   Bcl_*original building class* ranging from C1 to Z4
7. `AVTOT2`: Transitional total value
   Transval_1: Less than or equal to 35000
   Transval_2: More than 35000 and less than or equal to 560000
   Transval_3: More than 560000
8. `Community_Board`: Community board of the building
   cmb_*original 3 digit numbering*
9. `Council_District`: Council district of the building
   cdb_*original 1 to 2 digit numbering*
10. `Lot_Area`: A new column that we appended to the dataset. It is calculated from LTFRONT * LTDEP which is lot width and lot depth from the original dataset respectively.
    Lotarea_1: Less than or equal to 10000
    Lotarea_2: More than 10000 and less than or equal to 50000
    Lotarea_3: More than 500000
11. `Building_Area`: A new column that we appended to the dataset. It is calculated from BLDFONT * BLDDEPTH which is building width and building depth from the original dataset respectively.
    Buildingarea_1: Less than or equal to 10000
    Buildingarea_2: More than 10000 and less than or equal to 50000
    Buildingarea_3: More than 50000
12. `Census_Tract`: Number of individuals living in the property
    Census_1: Less than or equal to 1000
    Census_2: More than 1000 and less than or equal to 6000
    Census_3: More than 6000
13. `AVLAND2`: Transitional land value
    Avland2_1: Less than or equal to 10000
    Avland2_2: More than 10000 and less than or equal to 10000
    Avland2_3: More than 10000
14. `EXLAND2`: Transitional exemption land value
    Exland2_1: Less than or equal to 50000
    Exland2_2: More than 50000 and less than or equal to 100000
    Exland2_3: More than 100000
15. `EXTOT2`: Transitional exemption land total
    Extot2_1: Less than or equal to 2000
    Extot2_2: More than 2000 and less than or equal to 5000
    Extot2_3: More than 5000
16. `EXCD2`: Exemption code 2
    Excd2_1: Less than or equal to 2000
    Excd2_2: More than 2000 and less than or equal to 5000
    Excd2_3: More than 5000

The result after data processing is the `INTEGRATED-DATASET.csv` under the `data/` folder.

*bucketization: For bucketization, we referenced quantile values to decide the splitting points for the buckets; thus making sure the transacction distribution is uniform throughout each bucket. e.g. to bucketize `Lotarea` into three buckets, we check out its 1/3 and 2/3 quantiles and round these values up or down as the splitting points. Therefore all three buckets will have similar amount of transactions after bucketizing.

c) Why did we choose this dataset?

We are interested in real estate and housing market in New York City and want to learn more about the larger real estate market as well as discover relationships between different entities in housing data. Some examples are: the relationship between boroughs and value of property, borough and number of parking spaces, value of property and the building area, ect.

d) Description of internal design of project (what apriori algorithm we used)

For this project we followed the a-priori algorithm described in Section 2.1 of the Agrawal and Srikant paper in VLDB 1994 to compute the frequet itemsets.

In the main program project3.py there are the following functions:

1. `Main()`
   The main function reads the INTEGRATED-DATASET as well as takes in the input parameters min_sup and min_conf. After reading the raw data, we need to convert the data frame into a list of sets so that **each row of the csv is interpreted as one transaction**. Then, we initialize the itemset L which is a dictionary of dictionaries, using the notation in the paper. e.g. L[2] is the 2-itemset that contains all 2-item tuples and corresponding support counts. As in the paper, for k=1, we initialize L[1] to contain all items in the market basket. For k = 2 and beyond we followed the standard procedure in the paper to check if the support for the items is more than or equal to the minimum support (specified as input params).

   Subsequently, `gen_assoc_rules` is called with params `L`, `min_sup`, and `min_conf` to generate association rules. Finally, the result which includes the frequent itemset and confident association rules is printed in decreasing order to the output file, `output.txt`.

2. `join()`
   The join function performs the apriori algorithm as described in the paper section 2.1.1. Given the list of items of length k the function outputs a list of item lists of length k+1. Specifically, candidate item lists of length k+1, or C_{k+1} as per the paper's notation.

3. `prune()`
   This function follows the implementation in the paper's section 2.1.1. It takes in 2 inputs including C, a list of candidate item lists of length k and L_pre, the k-itemset. The function will output the pruned list of item lists L_k+1. We do so by deleting all the itemsets c in C_k if any permutation of len(k-1) does not appear in the (k-1)-itemset, as described in the paper.

4. `apriori_gen()`
   This function takes in the current list L of k-itemsets and outputs the next list of k+1 itemsets. It first

calls join(L) to get a list of items of length k+1, i.e., the candidates list. Then pass the candidates list to `prune()`, where unqualified candidates will be deleted.

5. `gen_assoc_rules()`
   This function takes in the current list L of k-itemsets, min_sup, and min_conf. It outputs a list of association rules. We do so by computing the confidence for every permutation in every itemset tuple and the rules of the items in the tuple (for all items whose confidence is more than or equal to min_conf).

Command line specification of compelling sample run

In our example-run.txt we ran the following support and confidence values: min_sup = 0.02 min_conf = 0.5

> Note: It shouldn't be surprising that because of our bucketizing process, many "items" will have pretty high supprt ratio. This may be different in nature from most "market sales" dataset, where items generally have very low support rates. Nonetheless, This doesn't affect our adpotion of the Apriori algorithm and such a transformation actually shows the flexiility of the algorithm to be run on various datasets after a bit of modification as we did on our dataset. As a side note, **you can run our program with a higher min_sup value such as 0.1 and will still get meaningful relations.**

Why is it compelling?

The association rules listed in the example-run.txt file show many interesting and useful relationships. There are some obvious ones, such as `['val_1'] => ['str_1'] (Conf: 100.0%, Supp: 2.1%)`, which implies properties that have lower values (<$82000) tend to have less then 6 stories. And some rules speak of census. Such as the association between community board and boroughs. For example, `['cmb_208'] => ['bor_2'] (Conf: 100.0%, Supp: 2.8%)` indicates that community board cmb_208 implies the property is located in Bronx. Likewise, `['cmb_107'] => ['census_1'] (Conf: 100.0%, Supp: 8.5%)` suggests that if a property is associated with community board cmb_107 then it has less than or equal to 1000 residents. We also generated many high confidence and high support association rules such as `['buildingarea_1'] => ['census_1'] (Conf: 79.1%, Supp: 43.8%)`. This shows that a property with building area of less than or equal to 10000 implies less than or equal to 1000 residents living in the building. Some relations pertaining to the property were also extracted. As an example, we found that properties assessed after 2016 are likely to have less than 61 parking lots (captured by `['yrs_3'] => ['lot_1'] (Conf: 78.6%, Supp: 26.6%)`)

We also generated complex association rules with high confidence such as `['exland2_3'] => ['avland2_3', 'bcl_D4', 'extot2_3'] (Conf: 76.9%, Supp: 37.3%)`. This shows that Transitional exemption land value of more than 100000 implies transitional land value of more than 10000, original building class D4, and transitional exemption land total of more than 5000.

All in all, the results we produced are compelling since we are able to draw connections and associations between different entities in the New York City real estate market. We are fulfilled by this is wonderful learning journey in terms of both algorithm and NYC housing market.