



# ReadMe

Status	In progress
Assign	

## NYC OpenData Data Mining

Team members: Yamini Ananth yva2002, Erin Liang ell2147

### About

This project extracts correlations between tree and temperature data in NYC by implementing the Apriori algorithm for finding association rules over an integrated dataset created from [NYC OpenData's Hyperlocal Temperature Monitoring](#) dataset—temperature data from neighborhoods with the highest heat mortality risk during the summers of 2018 and 2019—and the ever-so-popular [2015 Street Tree Census dataset](#) from the NYC Parks & Rec department. Although the Apriori algorithm consists of the bulk of the project implementation, the project's primary contribution is the integrated dataset itself and modifying the dataset to be Apriori-compatible. Details [on the selection, integration, and limitations of these datasets are discussed in the README.](#)

This project was completed as part of the Spring 2023 version of Columbia University's Advanced Database Systems course (COMS E6111) taught by Professor Luis Gravano at Columbia University.

### File Structure

```
proj3
├── lib
│   ├── data-cleaning.py
│   ├── main.py
│   └── AssociationRulesExtractor.py
├── INTEGRATED-DATASET.csv
├── example-run.txt
├── README.pdf <-- You're here now!
└── setup.sh
```

Filename	Description
<a href="#">setup.sh</a>	Bash script for setting up environment
<a href="#">data-cleaning.py</a>	Retrieving/downloading data, cleaning, processing, merging, and sampling data to product INTEGRATED-DATASET.csv.
<a href="#">INTEGRATED-DATASET.csv</a>	resulting cleaned + sampled data from sources
<a href="#">AssociationRulesExtractor.py</a>	Creates objects that extract frequent itemsets and high conf rules from INTEGRATED-DATASET.csv
<a href="#">main.py</a>	Executable file that handles input args. creates an AssociationRuleExtractor, and generates frequent itemsets/high association rules.
<a href="#">example-run.txt</a>	Run example for support = 0.09, confidence = 0.7

### How To Run



All commands necessary to install the required software, dependencies, and run the program.

### Installing Dependencies

- Note: It is advised that you run the setup scripts in a virtual environment to manage your python library versions. For creating and activating virtual environments with the specific VM instances for this class (Ubuntu 18.04 LTS), see [this guide](#).

Navigate to the repository:

```
cd <your/path/to/proj3>
```

Make sure the setup script is executable by changing the file permissions:

```
chmod +x setup.sh
```

From the top-level repository, run the setup script:

```
bash setup.sh
```

- This setup script will install all the requirements needed to run the program.

## Running The Program

Make sure you are in the base repository (which should be the case if following the library installation instructions). Navigate to the proj3 repository otherwise.

```
$ pwd
<should/be/your/path/to/proj3>
```

Now, run the main file with the desired parameters to yield frequent itemsets and strong association rules. We recommend piping the output to a file for later perusal as it is quite long.

```
python3 main.py <min_support> <min_conf> > <output_file.txt>
```

The following command uses `min_support = 0.09`, `min_conf = 0.7` and generates the file `0.09support_0.7confidence.txt`

```
python3 main.py 0.09 0.7 > 0.09support_0.7confidence.txt
```

## Data Processing

The below sections detail how we:

- select the dataset
- actually generate the final dataset ( `INTEGRATED-DATASET.csv` ) over which we run the apriori algorithm over
- explain why studying temperature and tree data is compelling

### a. NYC OpenData Datasets Used

We use two datasets from Open Data, free public data published by NYC agencies are used.

- [2015 Street Tree Census — Tree Data](#) 🌳
  - Provided by volunteers and staff from the NYC Parks and Rec department in 2015. Contains street tree data (tree species, diameter, health, location). [Cool treemap using this data here](#) where you can see all the street trees in NYC.
- [2021 Hyperlocal Temperature Monitoring](#) 🌡️
  - Provided by the NYC Parks and NYC Mental Hygiene departments as part of the Cool Neighborhoods Initiative. Contains hourly average measurements of street level temperature in the neighborhoods with highest heat mortality risk during the summers of 2018 and 2019.

Although these datasets are from different years, this gap shouldn't be a limitation because trees are slow-growing entities. 🌱

### b. Generating INTEGRATED-DATASET

- Download the trees and temps datasets from NYC OpenData to the empty `data` directory This can be done via the `wget` command in a virtual machine.

```
cd data
```

- Rename the datasets to `temps.csv` and `trees.csv` to work with the script to generate the final dataset.

```
wget https://data.cityofnewyork.us/api/views/qdq3-9eqn/rows.csv?accessType=DOWNLOAD
mv rows.csv?accessType=DOWNLOAD temps.csv
```

```
wget https://data.cityofnewyork.us/api/views/uvp1-gqnh/rows.csv?accessType=DOWNLOAD
mv rows.csv?accessType=DOWNLOAD trees.csv
```

Generate `INTEGRATED-DATASET.csv`

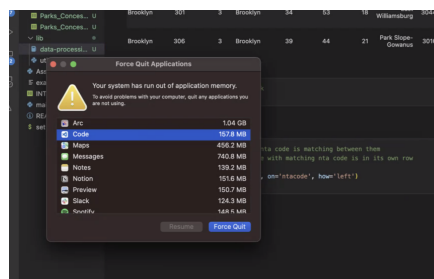
```
python3 lib/data-cleaning.py
```

This script will clean and filter the trees and temps datasets and create `INTEGRATED-DATASET.csv`. The high-level procedure for generating this csv is described below. For more detail see `lib/data-cleaning.py`.

## Dataset Cleaning

The script will do the following operations to make the datasets usable:

- **Drop columns that are not relevant to the task of extracting strong association rules.**
  - For the tree dataset, we drop columns that contain redundant information (e.g. `borocode` encodes same info as `borough`), columns that are too granular (e.g. `bin`), too broad (e.g. `state`) to be useful, and things that we don't care about (e.g. whether there are lights on the tree)
  - For the temps dataset, we similarly drop columns that are too broad to be useful (e.g. `Year`)
- **Sample down the 2.1M row hourly temperature dataset.** It is not feasible to analyze the entire dataset with the apriori algorithm. Our local machines ran out of memory even with using efficient csv analysis tools like Dask and Polars.



- We representatively sample this hourly time series data to create a more manageable dataset by taking the daily average temperature. Not every sensor has 100% uptime, which motivates this aggregation further.
- Further filter this data down by only including data from every 7th day of the week
- **Filter the trees dataset.** The temperatures dataset only includes temperatures measured from NYC neighborhoods with highest heat mortality risk, while the trees dataset includes trees from neighborhoods all over NYC. When we join these datasets in the next step, the trees that are not in these high heat mortality risk areas will be gone from the joined dataset, so it makes sense to prune these thousands of rows beforehand.

## Joining the datasets

- We inner join the temps and trees data on `ntacode`, which correspond to NYC's Neighborhood Tabulation Areas. These are medium-sized statistical geographical boundaries and roughly correspond with neighborhoods.
- These are still huge datasets, so we use `dask.dataframes` and create indexes on the join attribute to make the merging multithreaded and faster.
- There's still some cleaning we have to do after integrating the datasets:
  - **Downsample the dataset down to 50k rows.** Because apriori requires multiple passes over the data to generate all the large itemsets, using over 50k rows makes the program rather slow. Using 50k rows allowed us to run Apriori relatively quickly but still provided rich data from which to extract strong rules. As a note, even when using 5, or 15k data we extracted relatively similar quantities of rules and itemsets.
  - **Rename column values.** A column value might appear in multiple columns in the dataset or a column value might be hard to interpret without the context of the column name (e.g. in the problems column rename `None` → `Problems:None`). This was important so that the output of Apriori rules would be interpretable to the end user who had no familiarity with the initial data in the two datasets.
  - **Bin numerical data so we can extract stronger association rules from numerical data.** It does not make sense to distinguish 71 degrees and 70 degrees as distinct items. Without binning, it is not likely that a specific temperature value occurred more than a few times in the dataset.
    - For the temperature data, bin so 20% of the temperature data is separated into each bin. This will make it more likely that the temperature will appear in a frequent itemset.
    - Similarly, bin the tree diameter column.

## c. What makes INTEGRATED-DATASET compelling?

Our dataset is compelling for a number of reasons:

- **It is novel.**

- To our knowledge, there does not exist a dataset that explores the relationship between temperature and vegetation data in NYC. This dataset didn't exist before we did all this data cleaning!
- Additionally, the joining of the two datasets in question is *interpretable* (finding the temperatures around trees from tree sensors around a 3-4 yr timespan) and relatively easy to reproduce because both datasets share the NYC's `ntacode`.
- **It is sociopolitically interesting.**
  - Incorporating green infrastructure is advised as an effective way to reduce urban heat islands, a phenomenon closely linked to climate change. Structures such as buildings, roads, and other infrastructure absorb and re-emit the sun's heat more than natural landscapes such as forests and water bodies.
  - It is theorized that planting more trees would shade and deflect the heat that would otherwise directly heat up concrete surfaces and pavements in urban environments. What better urban environment to study than the concrete jungle itself?
  - Because of the nascency of green legislation, there has been no further research into what kinds of trees are best to plant. Are there specific species of trees better for cooling temperatures? Are bigger trees better? Would planting more trees within a block reduce its temperature?
  - Furthermore, we are limiting our data mining to the underserved high heat mortality risk areas (by nature of the locations of the temperature data collection)—the areas that matter the most.

## Design Descriptions

### Apriori Overview

- Apriori is an algorithm commonly used to learn association rules over relational databases in the field of data mining and exploratory data analysis. The goal of Apriori is to discover if there are any associations between items to find potentially useful and ultimately understandable patterns in the data.
- More formally, Apriori's goal is to mine good *association rules* that we can generate from a database of transactions.
  - Good association rules are defined as rules that have sufficient support and confidence ( `min_sup` , `min_conf` ) to make sure the rules have enough statistical significance and that the LHS is strongly associated with the RHS.
  - Importantly, these association rules do not imply causality. Moreso, they imply an "if `LHS_OF_RULE` occurs, it is also likely that `RHS_OF_RULE` occurs" relationship.
- The Apriori algorithm implemented is as described in Section 2.1 of the Agrawal and Srikant paper in VLDB 1994, minus the subset function using the hash tree section.

### Apriori Algorithm Implementation

- First, we generate all frequent large itemsets.
  1. Generate all possible singletons; prune by support to only frequent singletons.
  2. Generate all possible  $k > 1$  frequent itemsets, verifying that for each  $k$ -itemset, all  $k-1$  subsets are frequent.
  3. We terminate when no frequent  $k$ -itemsets are found for a given  $k$ .



Note: We convert all rows in the dataframe into sets upon initializing our AssociationRuleExtractor object. This way, checking if a subset exists is a constant-time operation. This significantly speeds up the process of generating all frequent large itemsets.

- From the large itemsets, compute and return the strong association rules.
  1. For each large itemset of size  $k > 1$ , we generate all possible combinations of size  $k-1$ . For each candidate combination, we can generate a candidate rule (itemset.subset( $k-1$ ), itemset-subset) which will have size ( $k-1$ ,  $k$ )
  2. Check the confidence of the candidate rule using the formula  $\text{support}(\text{full\_itemset}) / \text{support}(\text{subset}(\mathbf{k-1}))$ . If confidence  $\geq \text{min\_conf}$  as supplied in the user arguments, then a given rule is added to the output. Rules are stored in a dictionary with key=rule, value=confidence.

### Modifications to the Apriori Algorithm

- Instead of checking if every possible candidate  $k$ -itemset is frequent, we add an extra optimization to prune these candidate  $k$ -itemsets by checking that every subset of the candidate  $k$ -itemset is frequent
  - e.g. a 4-itemset is only frequent if all of the 3-itemsets and 2-itemsets and 1-itemsets for all possible also frequent.

## Running Apriori Algorithm on INTEGRATED\_DATASET.csv

### Determining Frequent Itemset min\_support threshold:

- We tried a variety of supports. Since we were highly interested in seeing interactions between bins of data from numerical attributes along with attributes from the non-numerical attributes, we wanted support > 0.05 (we binned data into bins of at least 5%, and wanted these bins to be frequent items).
- In the end, we used 0.09 as it provided a speedy enough runtime, and generated strong quantities of high quality itemsets (1764 frequent itemsets generated from our data).
- For each quantity by a factor of 0.01 that min\_support is decreased from 0.09, almost 1.5 full minutes are added to the program runtime.

## Determining Association Rule min\_conf threshold:

- We tried several min\_conf values in combination with 0.09 support. In the end, using 0.7 provided a robust set of rules that were interesting/compelling, but also well supported by the existing dataset.

## Interpreting the Resulting Association Rules

Looking into `example-run.txt`, which was run with min\_conf = 0.09, min\_sup = 0.7, we can examine the resultant frequent itemsets and strong association rules.

### Example (truncated) output

```
Parameters:
Minimum support      : <min_sup>
Minimum confidence   : <min_conf>
=====

+-----+-----+-----+
|                                     Frequent itemsets (min_sup=0.09)                                     |
+-----+-----+-----+
|                                     Itemset                                                         | Support % |
+-----+-----+-----+
|                                     Status: Alive                                                    | 97.2480 % |
|                                     OnCurb                                                            | 95.3040 % |
|                                     OnCurb, Status: Alive                                             | 92.6700 % |
|                                     . . .                                                            | ...      |
+-----+-----+-----+
Total number of frequent itemsets: <total_itemsets>
=====

Extracting association rules from dataset...

+-----+-----+-----+
|                                     Strong Association Rules (min_conf=0.7)                                     |
+-----+-----+-----+
|                                     Rule                                                             | Confidence % |
+-----+-----+-----+
|                                     ('Status: Alive',) => ('BX27',)                                     | 979.1381 % |
|                                     ('Status: Alive',) => ('BK81',)                                     | 975.7977 % |
|                                     ('OnCurb',) => ('BX27',)                                         | 959.5650 % |
|                                     . . .                                                            | ...      |
+-----+-----+-----+
Total number of strong association rules: <total_rules>
=====
```

## Frequent Itemsets


- With 0.09 support, we yield 1744 frequent itemsets of length k=1 to k=7
- We note a few “obvious”/control itemsets are correctly identified. These give credence to the fact that our algorithm is correctly identifying related/frequent itemsets.

| Bronx, BX27 | 9.9320 % | BX27 is an `ntacode` in the Bronx ✓

| London planetree, Platanus x acerifolia | 12.5740 % | This is the latin and common name for the same type of tree. ✓

## High Confidence Rules

- We note a few “obvious”/control rules that give credence to the fact that our algorithm is correctly identifying strong rules

 (Manhattan) => (10029)

- 10029 is a large zipcode in Manhattan, likely has a very high density of trees due to Central Park ✓
- It's also important to note that a lot of our “high confidence rules” were 100%. These demonstrate trivial relationships, which also validate the correctness of our algorithm



('Health: Good', 'Surveyor Type: NYC Parks Staff') => ('Status: Alive',)  
 ('Health: Good', 'honeylocust') => ('Status: Alive',)  
 ('Health: Good', 'Surveyor Type: TreesCount Staff') => ('Status: Alive',)

### Other interesting rule examples:

- Brooklyn has lots of healthy trees! With a confidence of 71.4957 %



('Brooklyn', 'Sidewalk: No Sidewalk: Damage') => ('Problems: None')

- Rule example with tree diameter to demonstrate that our binning is working as expected! Lots of healthy, alive, skinny trees



('Tree Diameter: (-0.001, 4.0]', 'OnCurb', 2019, 'Status: Alive') => ('Health: Good',). Confidence 77.1554%

('Tree Diameter: (-0.001, 4.0]', 'OnCurb', 'Surveyor Type: TreesCount Staff', 'Problems: None') => ('Health: Good',) Confidence 77.1554 %

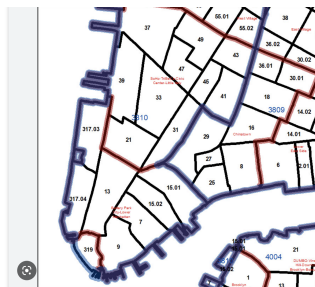
('Tree Diameter: (-0.001, 4.0]', 'OnCurb', 'Surveyor Type: TreesCount Staff', 'Status: Alive', 'Problems: None') => ('Health: Good',) Confidence 77.1554 %

We hoped to extract relations between, for instance, species names and temperature regions, or species names and boroughs; however, these were largely not associations present or mineable in the data for a few reasons. Namely, New York has a great variety of trees, so only about 3-4 tree types and species were even frequent enough to be supported as frequent items; as a result it was difficult to find any meaningful association rules that involved them. Furthermore, due to the binning of the data, any association we found would have been relatively general in any case.

## Future Work 🙌

Most of our time working on this project was spent trying to integrate the massive trees and temperatures datasets to be usable with the apriori algorithm, i.e. data cleaning and engineering. As a result, there are a number of areas in which we would like to improve on in future versions:

- **Implement the subset function with the hashtree data structure** as described in [Section 2.1.2 of the Agrawal and Srikant paper VLDB 1994](#). Currently, we generate the subsets of the candidate k-itemsets on the fly, which eats into runtime a bit.
- **Reduce the amount of sampling.** For practical purposes of this assignment, we traded of some degree of accuracy for some efficiency by sampling the otherwise huge dataset (~2M rows) down to 5k rows. This made it so the search for frequent itemsets can be completed in main memory.
- **“Bin” the tree species together.** The tree species collected in the trees datasets are quite granular (e.g. “Platanus x acerifolia” or “Gleditsia triacanthos var. inermis”). In the future, we could create stronger association rules by grouping some species together if they fall under the same botanical “genus” or “family” taxonomic umbrella. [Here's an example grouping for the Platanus x acerifolia species.](#)
- **Incorporate more external data to explore the urban heat island effect more.** There are many other factors that are hypothesized to contribute to the urban heat island effect, including building height, building materials (asphalt, concrete, astroturf), orientation of the street, and elevation. Incorporating more external data about the build environment of NYC could yield interesting association rules about how much these factors actually reduce temperature (and whether adding more green infrastructure would actually reduce the temperature of a block!)
- **Re-evaluate nta\_code usage to more accurately track a tree's impact on a block's temperature.** We joined the trees and temperature datasets on nta\_codes. The nta bounds of the lower tip of Manhattan are shown below.



- Realistically speaking, a tree in the upper left corner of the nta\_code=7 region would not impact the temperature measured in the bottom right corner of nta\_code=7 region. This is a side effect of nta\_code being a medium-sized statistical geographical boundary. In the future, we could consider creating bounds around trees by using both datasets' latitude and longitude attributes or trying to find a way to aggregate by block.

