CS513 Theory and Practice of Data Cleaning

Final Project Report

Apoorva Hoysala Srinivasa- apoorva6

Individual Submission

Data cleaning is an essential step in data analytics; and has to be carried out before the actual analysis takes place. Any analysis that is done with bad/dirty data does not carry any meaning or, carries false interpretations about the data. Thus, the goal of this project is to clean up data so that is it suitable for accurate and meaningful analysis. First, we begin by discussing what is in the dataset and what are its discrepancies. We then establish use cases that can be developed before and after the cleaning process. Cleaning is then carried out using a powerful tool called OpenRefine, followed by some cleaning done in Python language. Then, a relational database schema is built in SQLite and a check for Integrity constraints is carried out in SQLite. Following this a workflow model is built using the YesWorkflow tool to showcase all the changes made to the dataset.

About the dataset:

The dataset chosen for the CS513 final project is a U.S Farmer's Market dataset. Its source is https://www.ams.usda.gov/local-food-directories/farmersmarkets. A Farmer's market is a business run by the farmers themselves, where-in they sell their produce directly to the customers.
The dataset consists of information about various markets throughout the U.S, such as, location of the market, its social networking accounts, active months, mode of payment, goods sold etc.

The dataset has a total of 8687 observations with its primary key being FMID.

We can see the dataset has some groups inherent in it. We will consider these groups as entities for now and attributes define these entities. For example, Social media can be an entity; Facebook, Twitter, YouTube can be its attributes. Similarly, products can be an entity and Cheese, crafts, flowers, eggs can be its attributes.

- FMID -This is a unique key that defines every Farmer's market distinctly.
- Website – A website link for every farmer's market
- Social Media – Links to various social media accounts or account name
  - Facebook
  - Twitter
  - YouTube
  - OtherMedia
- Located – Information about where the Farmer's markets are situated within the U.S
  - street

    - city
    - county
    - State
    - zip
    - x - Longitude
    - y – Latitude
    - Location
- Active periods – Time during the year when the farmer's markets are active. This is divided by the four seasons.
    - Season1Date, Season1Time, Season2Date, Season2Time, Season3Date, Season3Time, Season4Date, Season4Time
- Mode of Payment – Which types of payments are accepted by these markets
    - Credit
    - WIC
    - WICcash
    - SFMNP
    - SNAP
- Goods – Variety of goods sold in these markets:
    - Organic, Bakedgood, Cheese, crafts, flowers, eggs, Seafood, Herbs, Vegetables, Honey, Jams, Maple, Meat, Nursery, Nuts, Plants, Poultry, Prepared, Soap, Trees, Wine, Coffee, Beans, Fruits, Grains, Juices, Mushrooms, PetFood, Tpfu, WildHarvested
- updateTime – date and time when the observation was added/updated in the dataset.

Data discrepancies:

1) The dataset contains NULL values in most of its columns. Specially the columns - Season2Date, Season2Time, Season3Date, Season3Time, Season4Date, Season4Time have more than 75% NULL values. Thus, these columns do not help us much with analysis because we cannot accurately say whether they are NULL because most markets were inactive at the time or they are NULL because of data entry issues.
2) Columns do not maintain consistent formatting within them. For example,
    - Some values in Facebook, Twitter, YouTube are given as a website address while some are mentioned as account names.
    - Season1Date has inconsistent formatting.
    - Column names such as MarketName, county have some observations represented completely in uppercase while other represented in lowercase
3) There is also some mix-up of the attribute entries. For example, in one of the twitter columns there is Facebook address provided.

The dataset can be used as is for the following analysis:

1) Which product is sold in most of the markets? Which product is sold the least? (In other words - Which are the products that are most and least popular among farmer's

markets) {“Sold” here does not mean bought, it means items the market is selling}. Top 5 items can also be analyzed.
2) Which method of payment is used by majority of the markets?
3) Which state has the highest number of farmer’s markets?
4) Count of Farmer’s markets that have a website
5) Number of farmers markets that have a Facebook/Twitter/YouTube/Instagram account

Hypothetical use cases:

1) Top 5 cities or counties that have the highest number of farmer’s markets (This can be shown as a drilldown in SQL using “group by” statement).
2) Which are the most popular months for Season1 among the Farmer’s market?
(Note: It does not make sense to repeat these for seasons 2-4 because most of the rows are null for these columns.)
3) On converting latitude and longitude column to numeric, we can plot the points in a geographical map.
4) Since the zipcodes are null for certain observations, we can derive them from their respective latitudinal and longitudinal information. However, there are 29 records which have null values for latitude and longitude as well. We have no choice but to exclude these from our analysis. By obtaining zipcodes for missing observations from latitude and longitude information, if a customer wishes to find a farmer’s market near his house/office, this can be done by querying – i.e filtering for a particular zip code.

Data cleaning using OpenRefine

OpenRefine is a tool that allows us to perform various data cleaning activities. Although for our hypothetical use cases we do not need to make use of all the columns in the dataset, we will still clean the entire dataset to an extent where it will be easy to comprehend.

Before any cleaning is carried out, there are two fundamental steps:
- Trim leading and trailing whitespace
- Collapse consecutive white space

The results are shown below:

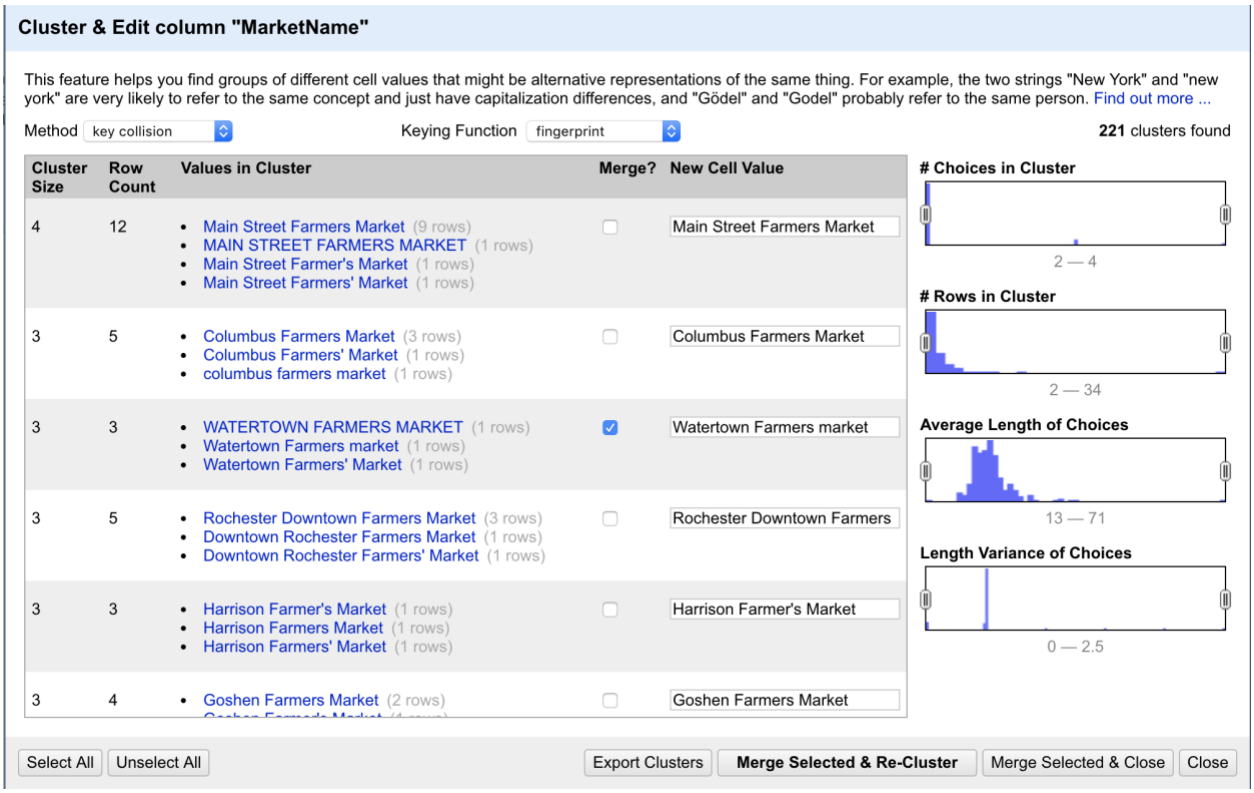
Name of column	# obs affected by trimming leading and trailing whitespace	# rows affected by collapsing consecutive white space
MarketName	392	43
Website	21	0
Facebook	32	3
Twitter	11	0

YouTube	4	0
OtherMedia	15	18
street	303	87
city	917	2
updateTime	0	219

Now we inspect and clean the columns individually one after the other:

Market name

- Use text facet and cluster similar texts by using key collision method and fingerprint keying function. All the text in this column is converted to small case with the exception of first letter of each word, which is uppercase. (653 rows affected by this change)



Facebook

- Transform the entire column to text
- (Note- For our use case, we only need to count the number of markets that have a Facebook account. For this reason, whatever observation that says they don't have an account will be made null.)
  Change all "n/a", "no" and "None" to BLANKS (7 rows affected). If the column reads "yes" then we retain it as is. This transformation is done within OpenRefine using python language
- We see that some rows provide direct link to their Facebook account and some only have the account name. We make them similar by adding the prefix- "https://facebook.com/" to the ones that contain only account names. (525 rows affected). This too is done in OpenRefine using Python.

Twitter

- Transform the entire column to text
- Change all "n/a", "NA", "no", "No twitter" and "None" to BLANKS (19 rows affected). Again, this transformation is done within OpenRefine using python
- Similar to Facebook column, some observation in Twitter have only account name while some have entire address. This difference is eliminated. Also, there are some twitter names that start with "@". This symbol is removed before adding "https://twitter.com/" before the names. (255 observations changed)

YouTube

- Transform the column to text
- Change all "n/a", "NA", "no" and "None" to BLANKS (22 rows affected). Again, this transformation is done within OpenRefine using python
- Observations that have only channel names are given a prefix "www.youtube.com"

  There are lot of discrepancies in the Facebook, YouTube and Twitter columns in the way they are displayed. But since our use case does not require us to fix this, we will not dive deep into the issues.

city and County

- Use text facet and cluster similar texts by using key collision method and fingerprint keying function. All the text in this column is converted to small case with the exception of first letter of each word, which is uppercase. (561 cells affected in city and 734 cells affected in the county column)
- In the city column, replace "-" and "AV±asco" to NULL

Cluster & Edit column "city"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. Find out more ...

Method key collision Keying Function fingerprint **89** clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
3	8	• Woodstock (5 rows) • woodstock (2 rows) • WOODSTOCK (1 rows)	☐	Woodstock
2	43	• Brooklyn (41 rows) • BROOKLYN (2 rows)	☐	Brooklyn
2	7	• Akron (6 rows) • AKRON (1 rows)	☐	Akron
2	2	• FOUNTAIN (1 rows) • Fountain (1 rows)	☐	Fountain
2	32	• Los Angeles (30 rows) • LOS ANGELES (2 rows)	☐	Los Angeles
2	10	• St. Louis (8 rows) • St Louis (2 rows)	☐	St. Louis
2	14	• Canton (13 rows) • canton (1 rows)	☐	Canton

 # Choices in Cluster

2 — 3

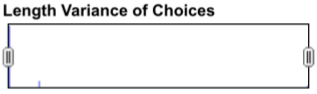
 # Rows in Cluster

2 — 43

Average Length of Choices

4 — 16

Length Variance of Choices



0 — 5

Select All | Unselect All | Export Clusters | **Merge Selected & Re-Cluster** | Merge Selected & Close | Close

Cluster & Edit column "County"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. Find out more ...

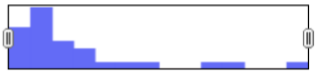
Method key collision Keying Function fingerprint **28** clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
3	14	• DeKalb (11 rows) • Dekalb (2 rows) • DEKALB (1 rows)	☐	DeKalb
2	11	• Santa Barbara (9 rows) • SANTA BARBARA (2 rows)	☐	Santa Barbara
2	94	• Washington (93 rows) • WASHINGTON (1 rows)	☐	Washington
2	5	• Mobile (4 rows) • MOBILE (1 rows)	☐	Mobile
2	14	• Fresno (13 rows) • FRESNO (1 rows)	☐	Fresno
2	125	• Los Angeles (121 rows) • LOS ANGELES (4 rows)	☐	Los Angeles
2	17	• Humboldt (16 rows) • HUMBOLDT (1 rows)	☐	Humboldt

 # Choices in Cluster

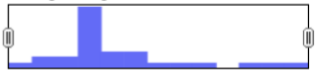
2 — 3

 # Rows in Cluster



0 — 130

Average Length of Choices



3 — 15

Select All | Unselect All | Export Clusters | **Merge Selected & Re-Cluster** | Merge Selected & Close | Close

Season1Date

- Spilt the start and end dates as two columns using GREL. For our use case we pull out the months from the two dates. While some cells are in date format, some just have month given in words. We convert these word representations to numbers. (For example “July” is changed to “07”).
- The original column is dropped and we create two columns called Season1StartMonth and Season1EndMonth

Season2Date, Season3Date, Season4Date, Season2Time, Season3Time, Season4Time

- These columns were dropped from the dataset because they have lot of NULL values in them which makes it unsuitable for any analysis. (8236 rows out of total 8687 rows are NULL for Season2Date, 8604 rows out of total 8687 rows are NULL for Season3Date and 8679 rows out of total 8687 rows are NULL for Season4Date.)

Longitude(x) and Latitude(y)

- Convert the entire column to numeric (8658 rows affected)

Organic

- Replace “-“ to Blank using GREL (5043 rows affected)

Further Cleaning using Python

In our hypothetical use case we discussed that we can derive the zipcodes using Latitude and Longitudinal values. This will be carried out in Python by adding an extra column called derived_zip.
Also, in the original zip column there are two different kinds of representations- XXXXX and XXXXX-XXXX. We will keep the format XXXXX so as to make our future analysis easier.

Relational Database Schema

Below is the ER diagram for the Farmer’s Market dataset. As we are considering just one table, we will consider the dataset as the entity itself and all the variables that define the dataset as attributes.

Farmer's Market	
FMID (Primary key)	INTEGER
MarketName	TEXT
Webiste	TEXT
Facebook	TEXT
Twitter	TEXT
YouTube	TEXT
OtherMedia	TEXT
street	TEXT
city	TEXT
County	TEXT
State	TEXT
zip	INTEGER
derived_zip	INTEGER
Season1StartMonth	NUMERIC
Season1EndMonth	NUMERIC
Season1Time	NUMERIC
x	REAL
y	REAL
Location	INTEGER
Credit	INTEGER
WIC	INTEGER
WICcash	INTEGER
SFMNP	INTEGER
SNAP	INTEGER
Organic	INTEGER
Bakedgoods	INTEGER
Cheese	INTEGER
Crafts	INTEGER
Flowers	INTEGER
Eggs	INTEGER
Seafood	INTEGER
Herbs	INTEGER
Vegetables	INTEGER
Honey	INTEGER
Jams	INTEGER
Maple	INTEGER
Meat	INTEGER
Nursery	INTEGER
Nuts	INTEGER
Plants	INTEGER

Poultry	INTEGER
Prepared	INTEGER
Soap	INTEGER
Trees	INTEGER
Wine	INTEGER
Coffee	INTEGER
Beans	INTEGER
Fruits	INTEGER
Grains	INTEGER
Juices	INTEGER
Mushrooms	INTEGER
PetFood	INTEGER
Tofu	INTEGER
WildHarvested	INTEGER
updateTime	INTEGER

(The code to create a database schema from the original spreadsheet is attached in the project submission folder)

Checking integrity constraints using SQLite

After loading the csv output from Python into a relational database using SQLite, we will check to see if the clean dataset now satisfies minimum integrity constraints.
On finding any records that violate the constraints, the table will be updated to make sure all the conflicts are resolved. This process of updating is also carried out using SQLite.

Integrity Constraint 1:

Check if all the rows are accurately migrated to the relational database from the original csv.

Code:
```
select count(FMID) from farmersmarket;
```

Output:
8687

Integrity Constraint 2:

The primary key, FMID should not contain any NULL values.

```
Code:
select count(*) from farmersmarket where FMID is NULL;

Output:
0
```

Integrity Constraint 3:

The primary key FMID has to be unique for every observation in the dataset.

```
Code:
select count(distinct FMID) from farmersmarket;

Output:
8687
```

Integrity Constraint 4:

Season1 start month cannot be more than Season1 End month.

```
Code:
select FMID, Season1StartMonth, Season1EndMonth  from farmersmarket where
Season1StartMonth>Season1EndMonth;

OR

select count(*)  from farmersmarket where Season1StartMonth>Season1EndMonth;

Output:
168
```

There are 168 observations that have end month before the start month. This, obviously, is a discrepancy and does not make sense if used in our analysis. Thus these observations are made NULL.

```
Code:
 update farmersmarket
 Set Season1StartMonth = Season1EndMonth = NULL
 where Season1StartMonth > Season1EndMonth;
```

Code:
Select count(*) from farmersmarket where
Season1StartMonth > Season1EndMonth;

Output:
0

Integrity Constraint 5:

Since we derived zipcode from latitude and longitude, zipcode must not be NULL except for 29 rows(because these 29 rows do not have latitude and longitude information).

Code:
Select count(*) from farmersmarket where derived_zip is NULL;

Output:
29

Also, if our derivation of zipcode from latitude and longitude is successful, the values derived should match with the values originally provided in the dataset in the zipcode column.

Code:
Select count(*) from farmersmarket where derived_zip <> zip;

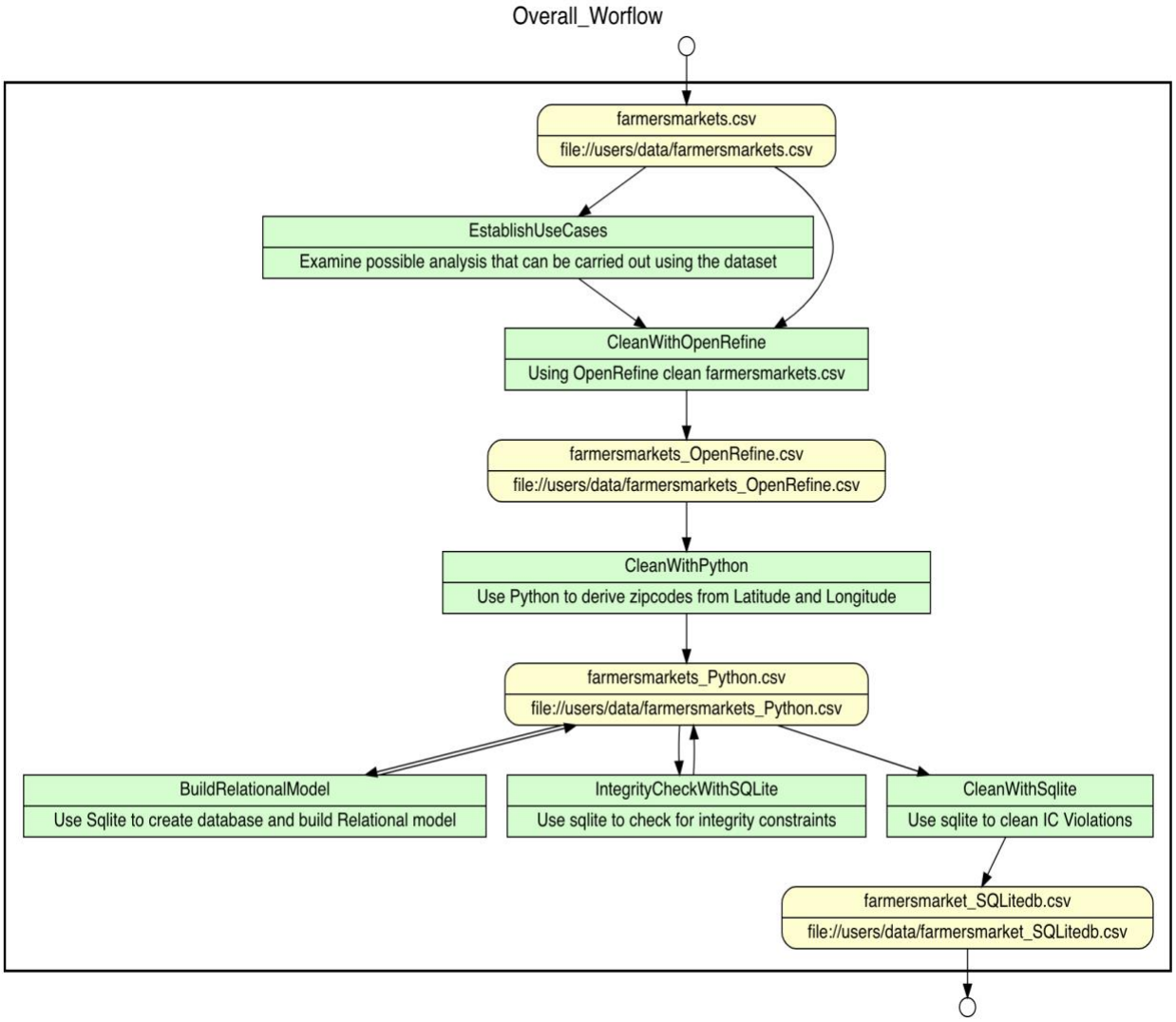
Output:
3663

A whooping count of 3663 rows do not match between derived zip and zipcode provided in the dataset. Among these 945 rows will not be equal because their original zipcodes are NULL. The remaining 2718 rows do not have matching zipcodes. On closely observing these values, we see that for most for them there is an offset by +/- 1 . This could occur because the zipcodes were filled in wrong originally, or, the x and y values are wrong.

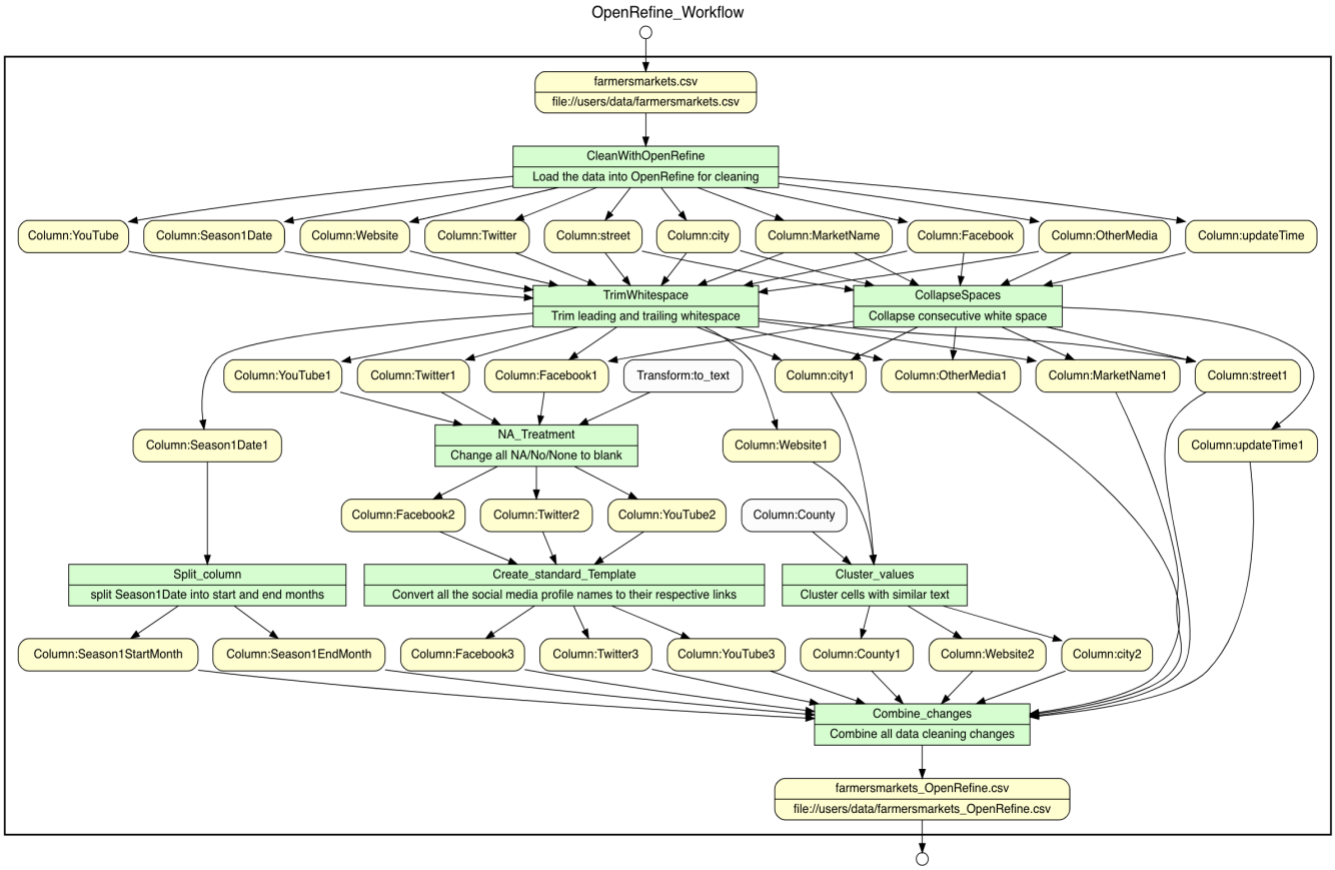
Workflow model

In order to make the dataset suitable for our analysis, lot of changes were made to the original dataset and it is hard to remember all the changes without a proper documentation. Further, developing a workflow model for the process is much more easier to understand. This is now carried out using a tool called YesWorFlow.

First, we will develop a workflow showcasing all the steps carried out throughout this project i.e the Overall_Worflow. This is shown below:



Due to the multiple changes made to the csv using OpenRefine tool, we will make a separate workflow for the process as well.



After all these cleaning steps the data is good enough to serve well for our use cases. However, there are many loopholes/faults in the data which makes it impossible for certain other analysis unless a human intervention is made.

References:

1) US Zip codes - https://uszipcode.readthedocs.io/index.html
2) Importing csv into database - https://www.quackit.com/sqlite/tutorial/import_data_from_csv_file.cfm