## CS513: Theory & Practice of Data Cleaning

## Individual Submission: Nadia Wood (nadiaw2)

## Team115

## Git repo: <https://github.com/nadiawoodninja/cs513-data-cleaning>

## Tableau Dashboard: <https://public.tableau.com/app/profile/nadia.wood/viz/cs513/Dashboard1>

# Dataset of interest:

* **Initial dataset**: farmersmarkets.csv. The source of this data is <https://www.ams.usda.gov/local-food-directories/farmersmarkets>
* There is an API available to download the farmers market data at this URL: https://www.usdalocalfoodportal.com/api/farmersmarket/
* According to the website “The Farmers Market Directory lists markets that feature two or more farm vendors selling agricultural products directly to customers at a common, recurrent physical location. Maintained by the Agricultural Marketing Service, the Directory is designed to provide customers with convenient access to information about farmers market listings to include: market locations, directions, operating times, product offerings, accepted forms of payment, and more.”
* **Output dataset:**
  + **farmersmarkets\_output.csv**
  + **farmersMarket\_location.csv**
  + **farmeresmarkets\_payments.csv**
  + **farmersmarkets\_products.csv**

# Initial Data Profiling:

I used the Pandas profiling library to develop a profile of the data to understand it better.

**Link to the data profiling report**: <https://htmlpreview.github.io/?https://github.com/nadiawoodninja/cs513-data-cleaning/blob/main/DataProfiling/farmersDataStats.html>

Graphical user interface, application

Description automatically generated

Names of all the columns in the csv

Text

Description automatically generated

# Project Use Cases

# U1 (main target)

Given the popularity and usage of credit cards (Apple Pay, Android Pay etc.), an interesting use case to develop would be to identify the markets that accept credit card in a certain geo location. A heatmap could be created to show which geo locations are using credit cards most and which ones are using least. This can be done with some data cleaning efforts given the current dataset.

# U0 use case that requires “zero data cleaning”:

A possible use case without data cleaning would be to determine the most and least popular products sold by markets by summing the 'Y' for each product's column. This can be pivoted in various other columns such as location of the market.

# U2 is a use case data “never (good) enough”:

Any use cases surrounding the Season2, Season3 and Season4 columns are never going to be useful due to a lot of missing data. No amount of wrangling, cleaning will be able to give us any insights into the data.

# Initial Assessment of the dataset

There are 8665 entries in the dataset and 59 columns. Further details can be found in the data profiling report generated by pandas\_profiling report.

Table

Description automatically generated Table

Description automatically generated

# Columns and their description:

* FMID – a 7 digit integer unique identifier for each farmers' market
* MarketName - a string containing the name of the farmers' market
* Website, Facebook, Twitter, Youtube, OtherMedia - a string containing URL or other information that identifies the social media information
* street, city, County, State, zip - strings containing data corresponding to the column name that identifies the location of the farmers' market
* Season1Date, Season1Time, Season2Date, Season2Time, Season3Date, Season3Time, Season4Date, Season4Time - date fields representing the start date and end date for the given farmers' market or the times in which the farmers' markets are opened
* x, y - latitude and longitude coordinates
* location - a string describing the location of the farmers' market
* Credit, WIC, WICcash, SFMNP, SNAP - Y/N (boolean) character to indicate whether or not a given payment method is accepted
* Organic, Bakedgoods, Cheese...PetFood, Tofu, WildHarvested (30 columns) - Y/N (boolean) column to indicate whether or not a given product is offered

# Data quality problems

***U1****: Given the popularity and usage of credit cards (Apple Pay, Android Pay etc.), an interesting use case to develop would be to identify the markets that accept* ***credit card*** *in a certain* ***geo location****.*

Graphical user interface

Description automatically generated with medium confidenceLet’s observe the missing values by looking at the nullity matrix of the data generated by pandas\_profiling. The nullity matrix is a data-dense display which allows one to quicky visually pick out patterns in data completion.

Doing a quick visual analysis, we can see that there are some missing data fields in the **street**, **city**, **county** and **zip** columns. Credit column does not have any missing data and seems to have either true or false values for each entry. Let’s deep dive into these columns.

**Street Column:** It has 285 missing values. To generate a geo heatmap it would this column is not crucial to have. We can also look at other markets which may have similar address and infer the street address.

**Graphical user interface

Description automatically generated with medium confidence**

**City:** This column has 40 missing values. We would have to look at other location related columns to determine the city.

Graphical user interface, text, application

Description automatically generated

**County:** There is 6.2% missing data in this column. Again, looking at other location columns, the data would have to be imputed.

Graphical user interface

Description automatically generated with medium confidence

**Zip:** There is 10.9% of the data missing in this column. Again, the data would have be imputed based on other location columns.

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

# Data Cleaning methods and process

* 1. Zip Code: Records were filtered where zip was missing and geo attributes (Longitude & Latitude) available. TomTom’s Reverse Geocode API was used to derive zip code. (See farmersmarket\_impute\_zip.ipynb). After this step, I had only 0.3% missing zip code vs. 10.9%. The resulting dataset only had 3 records where it did not have any geographics attributes such as street, city, state or longitude, latitude.

Graphical user interface, text, application

Description automatically generated

* 1. I used OpenRefine to clean the data. Please note that the tool works for smaller datasets and not large datasets
  2. Cleaning started with MarketName column by first trimming the leading and trailing

whitespace and then collapsing any consecutive whitespaces. Then a text facet was used and clustering to group similar MarketNames together. As seen below, the key collision method and the fingerprint keying function was used.

Graphical user interface, application

Description automatically generated

* 1. Next, removed some of the columns that are **irrelevant** to both main use case and other potential use cases. The social media data quality was very poor and so those columns were also removed: Website, Facebook, Twitter, Youtube, OtherMedia. The time and date columns for Season2 onwards were also removed because there was very little data for these columns.

Graphical user interface, application, table

Description automatically generated

* 1. Next. Switching the focus to the location columns - street, city, County, State, and zip. For street, GREL expressions were used to remove any special characters and substitute the ampersand with 'AND':

value.replace(/[%@#!.?;:,"']/, '').replace(/-\[\]\(\)/, '').replace("&", 'AND')

Graphical user interface, text, application, email

Description automatically generated

* 1. Then, the leading and trailing whitespace were trimmed and collapsed any consecutive whitespaces and converted to uppercase.

Graphical user interface, application

Description automatically generated

* 1. Next step was to merge any logical clusters together using the key collision method and

fingerprint keying function followed by ngram-fingerprint keying.

Graphical user interface, application

Description automatically generated

Graphical user interface, application, email

Description automatically generated

* 1. Same steps were used (remove special characters, trim and collapse whitespace, convert to uppercase, clustering) for the city, County, and State columns. After this process, the address information is much cleaner and more consistent.

Graphical user interface, application

Description automatically generated

* 1. Next step was to switch focus to Season1Date and Season1Time. These columns were removed as they are not relevant to the use case in focus.
  2. The x and y columns were renamed to latitude and longitude respectively, and then convert to numeric. The Location column was removed as it is not helpful for our purposes and is generally blank.
  3. The values in the updateTime column were converted to ISO format using the GREL expression: value.toDate('d/M/y H:m:s') after trimming and collapsing whitespace.

# Final Dataset

# Relational Database Schema

The following Entity Relationship shows the schema developed for the final dataset. The cleaned dataset was broken into three separate tables (found in 2CleanedData): location, payments, and products, with the FMID as the primary key for all of them. This ER diagram was generated using DBVisualizer after loading the separate tables into sqllite.

Diagram

Description automatically generated

Then, a few integrity constraints were developed using in sql-lite/sql-lite.ipynb notebook.

• Ensure that FMID is an appropriate primary non-null and unique key

• Ensure that data for my use case is non-null (specifically latitude, longitude, state)

• Ensure latitude is between [0,90] and longitude is between [-180, 180]

• Ensure FMID has unique address (street, City, County, State, zip) if it exists

# Modeling the Workflow

I modeled the workflow using or2yw tool, and the instructions found at <https://pypi.org/project/or2ywtool/>, I was able to create a workflow model of our data cleaning process very easily. I used the operations history json file from the OpenRefine Data Cleaning steps. Here the commands that I used:

1. pip install of the or2yw tool:

>pip install or2ywtool

2.Then, we generate both a serial and parallel yw file

>or2yw -i farmersmarkets\_OperationHistory.json -o farmersmarkets\_serial.yw

>or2yw -i farmersmarkets\_OperationHistory.json -o farmersmarkets\_parallel.yw -t parallel

3.Install graphviz:

>brew install graphviz

and then generate the model using the following commands:

>or2yw -i farmersmarkets\_OperationHistory.json -o farmersmarkets\_parallel.png -ot png -t

parallel

> or2yw -i farmersmarkets\_OperationHistory.json -o farmersmarkets\_linear.png -ot png

Graphical user interface, text, chat or text message

Description automatically generated

A picture containing graphical user interface

Description automatically generated

# Overview of changes

|  |  |
| --- | --- |
| FMID |  |
| MarketName | 392 leading/trailing whitespaces  trimmed; 43 whitespace collapsed;653  clustered |
| Website, Facebook, Twitter, Youtube,  OtherMedia | columns removed |
| street, city, County, State, zip | street: 3175 cells had special characters  removed/replaced; 305 had whitespaces  trimmed, and 108 had whitespace  collapsed, and 8301 were converted to  uppercase, and 84 total were clustered  city: 917 had whitespaces trimmed and 2  had whitespace collapsed; 68 total  clustered cells clustered, 24 had "-"  replaced,  County: 126 cells had punctuation/special  characters removed or replaced; 8140  were converted to uppercase  State: all converted to uppercase  zip: |
| Season1Date, Season1Time, Season2Date,  Season2Time, Season3Date, Season3Time,  Season4Date, Season4Time | Removed |
| x, y | columns renamed to latitude and  longitude, and 8658 cells converted to  Numeric |
| location | removed |
| Credit, WIC, WICcash, SFMNP, SNAP | No change |
| Organic, Bakedgoods, Cheese...PetFood,  Tofu, WildHarvested (30 columns) | 5043 cells had "-" replaced in Organic  column |
| updateTime | 219 cells had whitespace collapsed; 8384  cells changed to ISO date |

# Conclusion

Now that we have a clean dataset, we can dive into our focus use case. I used tableau to create a dashboard (<https://public.tableau.com/app/profile/nadia.wood/viz/cs513/Dashboard1>) with drop downs (State, Zip Code and Credit card yes or no) . This will allow us to explore the data.

From visualization, we can see that East coast, West coast and some parts of Midwest are dense with credit card acceptance.

Map

Description automatically generated

Here is another visualization of the zip codes that do not accept credit card. Comparing the two visualizations, it is hard to determine the difference between the two cases. So let’s break it down by state.

Map

Description automatically generated

Here we observe just the state of Minnesota. The places closer to bigger cities like Minneapolis and Rochester accept credits cards vs. some of the smaller cities do not.

Map

Description automatically generated

Within Minnesota, even within Hennepin County, there are markets which do not accept credit card. See below.

Map

Description automatically generatedTimeline, map

Description automatically generated

# Future Considerations

The data can allow to slice and dice information based on products, credit cards, city, zip code etc. The data can also be enriched by population data. There are many different permutations of the data that can be done to understand the data. Another thing to note that Tableau itself, gives the ability to clean and manipulate the data as well. The downside of cleaning the data in tableau is that there is no tracking history of changes unlike OpenRefine and Yesworkflow. These tools are great to be able to reproduce the steps taken to clean and manipulate the data.