Final Project Report

Team The Conquistadors (StateSurplusLands)

Kaijie Zhou, Janice He, Tommy Lam, Athina Said, Murtaza Moiyadi

I. Project Description

The State Surplus Project aimed to efficiently manage surplus lands in Massachusetts in conjunction with the land equity bill. The goal was to maximize the benefit of building both affordable housing and houses for sale. Additionally, we wanted to find the regions of surplus land that can make the most profit to sell as well. We believe that the best locations for maximizing profit are in regions with the highest income, close proximity to wealthy neighborhoods, and those that possess functional home utility services such as water and electricity.

II. Project Progress

We started off the project with 1.8 million records approximately, and we applied a filter of luc to focus on properties with only **luc of 9xx**, because we believed these are government owned properties. We then standardized the names for the same given address using **FuzzyWuzzy**, then we found matches in our dataset with the official list of Massachusetts state agencies based on the **agency names**. We also used an alternative method to **match owner addresses** with AgencyList addresses, and only obtained 300 matches. But this method has proven inaccurate and inefficient through communicating with the other team, Ziba, and Rishab, so we decide to further expand on the previous method of matching owner names.

After we matched the **owner names** with the AgencyList agency names, we further applied a filter on luc for **91,92,97** and poly type for **Fee** and **Tax** on our dataset and further reduced the dataset size from 55000 to 7543.

Recently, we have merged our dataset with the other team. As both teams use similar filters methods of luc by **91x**, **92x**, and **97x**, and poly_type of **FEE** and **Tax,** we only have discrepancies of 100 rows. Next, we filtered out the merged dataset even more by separating properties into two categories, namely **Transportation** and **Housing**. This is indicated by the column TransportationOrHousing.

The last filtering layer we applied to our dataset is **removing unusable lands** from our dataset. This includes waterlands, conservation areas etc. We recently finished the functionality of converting addresses to Point, and check if Point inside polygons in those shape files. We add a new column "**removable**" to the dataset to indicate if this address is removable. Above, we mentioned all the filtering and data cleaning techniques that we utilized, and the result dataset "**FinalDataset.csv**" is what we obtained.

Other implementations that we added beside the filter dataset is using **ATTOM** to evaluate average land property values given the neighborhood the land is in. All of these implementations reside in **DataExtraction.py**.

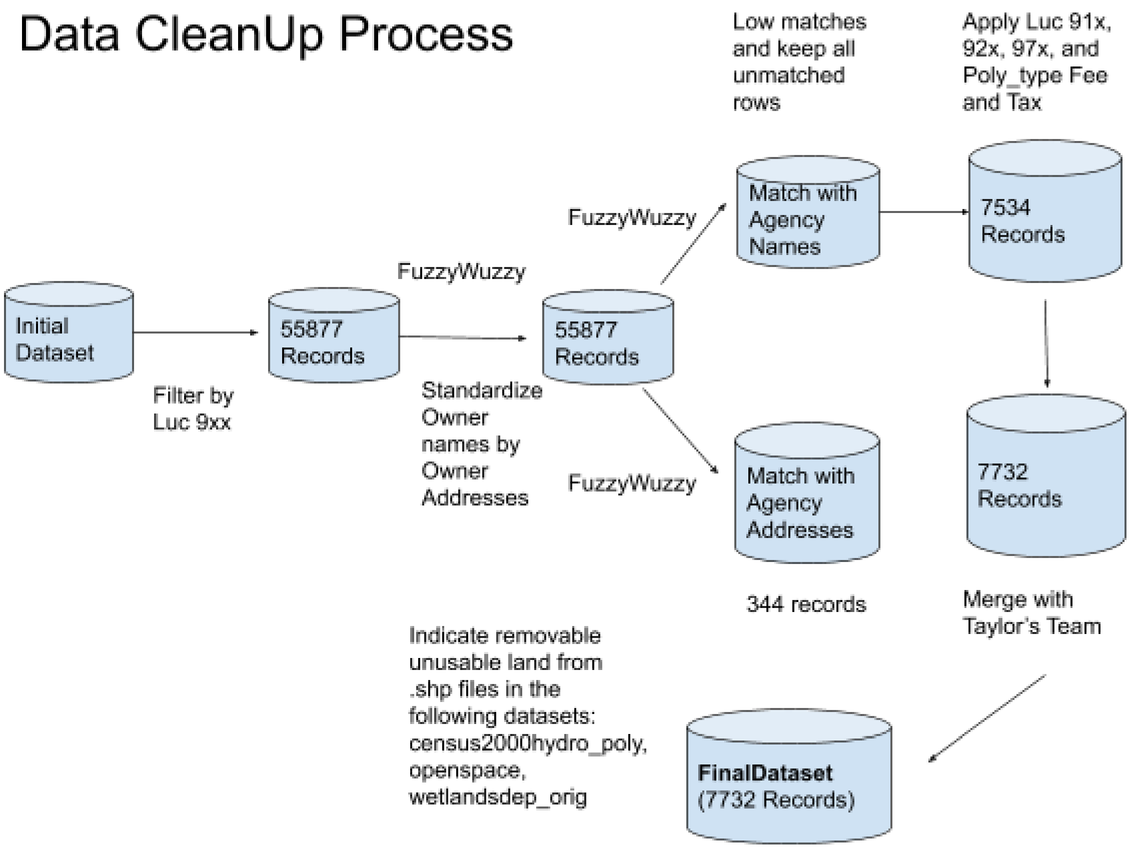
III. Challenges

A major difficulty we encountered was the data cleaning process for this project that became quite complex and time consuming as we progressed. Another difficulty that we came across was when we ran a land evaluation on the entire dataset is the **10 limited API Call per minute and 200 limited API Call every few days.** Depending on the size of the cleaned up dataset, land evaluation might take a long time to complete. As a result, we have not been able to perform evaluations on the entire dataset. One small sample of our dataset that we finished the evaluation on is **AttomEstimateResult.csv**.

Moreover, additional challenges that we faced during this project were the variety of uncertain columns that might contribute to the data cleaning process, the uncertainty of which method is best to use when cleaning the dataset, and the lack of communication with the client, which we could improve on if more time was allowed.

IV. Appendix

Appendix 1: Data CleanUp Process



Data CleanUp Process
Initial Dataset
Filter by Luc 9xx
55877 Records
FuzzyWuzzy
Standardize Owner names by Owner Addresses
55877 Records
FuzzyWuzzy
Match with Agency Names
Low matches and keep all unmatched rows
FuzzyWuzzy
Match with Agency Addresses
344 records
Apply Luc 91x, 92x, 97x, and Poly_type Fee and Tax
7534 Records
7732 Records
Merge with Taylor's Team
Indicate removable unusable land from .shp files in the following datasets: census2000hydro_poly, openspace, wetlandsdep_orig
FinalDataset (7732 Records)

Appendix 2: Python Files

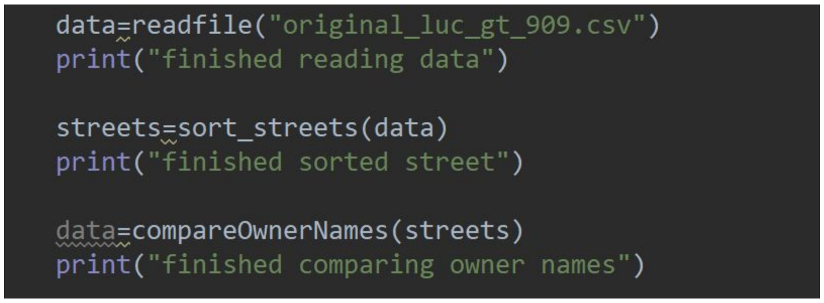
***Note running the below python files in order will produce the same FinalDataset.csv**

1. FilterDataset.py:

- Filter dataset by luc>909, result: **original_luc_gt_909.csv**

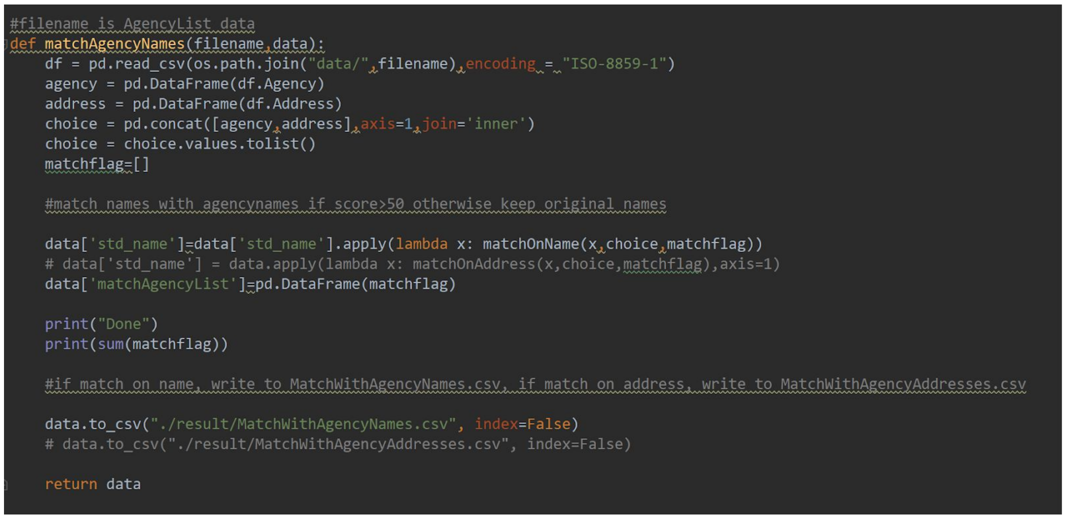
2. CleanUp.py:

- **Standardize owner name by owner addresses** :



```
data=readfile("original_luc_gt_909.csv")
print("finished reading data")

streets=sort_streets(data)
print("finished sorted street")

data=compareOwnerNames(streets)
print("finished comparing owner names")
```

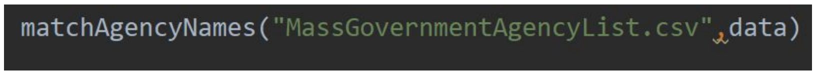
- **Match with Agency Names on Names**:



```
#filename is AgencyList data
def matchAgencyNames(filename,data):
    df = pd.read_csv(os.path.join("data/",filename),encoding = "ISO-8859-1")
    agency = pd.DataFrame(df.Agency)
    address = pd.DataFrame(df.Address)
    choice = pd.concat([agency,address],axis=1,join='inner')
    choice = choice.values.tolist()
    matchflag=[]

    #match names with agencynames if score>50 otherwise keep original names

    data['std_name']=data['std_name'].apply(lambda x: matchOnName(x,choice,matchflag))
    # data['std_name'] = data.apply(lambda x: matchOnAddress(x,choice,matchflag),axis=1)
    data['matchAgencyList']=pd.DataFrame(matchflag)

    print("Done")
    print(sum(matchflag))

    #if match on name, write to MatchWithAgencyNames.csv, if match on address, write to MatchWithAgencyAddresses.csv

    data.to_csv("./result/MatchWithAgencyNames.csv", index=False)
    # data.to_csv("./result/MatchWithAgencyAddresses.csv", index=False)

    return data
```

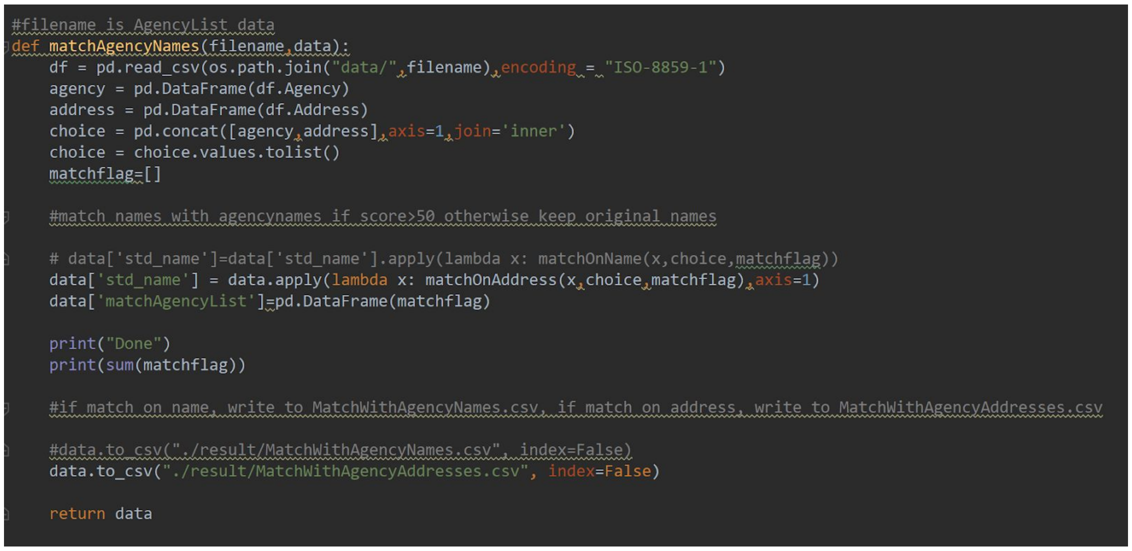
(make sure the matchOnAddress function is not being used)

- **Perform matching with MassGovernmentAgencyList:**



```
matchAgencyNames("MassGovernmentAgencyList.csv",data)
```

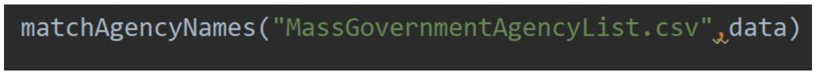
- **Match with Agency Names on Addresses**:



```
#filename is AgencyList data
def matchAgencyNames(filename,data):
    df = pd.read_csv(os.path.join("data/",filename),encoding = "ISO-8859-1")
    agency = pd.DataFrame(df.Agency)
    address = pd.DataFrame(df.Address)
    choice = pd.concat([agency,address],axis=1,join='inner')
    choice = choice.values.tolist()
    matchflag=[]

    #match names with agencynames if score>50 otherwise keep original names

    # data['std_name']=data['std_name'].apply(lambda x: matchOnName(x,choice,matchflag))
    data['std_name'] = data.apply(lambda x: matchOnAddress(x,choice,matchflag),axis=1)
    data['matchAgencyList']=pd.DataFrame(matchflag)

    print("Done")
    print(sum(matchflag))

    #if match on name, write to MatchWithAgencyNames.csv, if match on address, write to MatchWithAgencyAddresses.csv

    #data.to_csv("./result/MatchWithAgencyNames.csv", index=False)
    data.to_csv("./result/MatchWithAgencyAddresses.csv", index=False)

    return data
```

(make sure matchOnName function is not used)

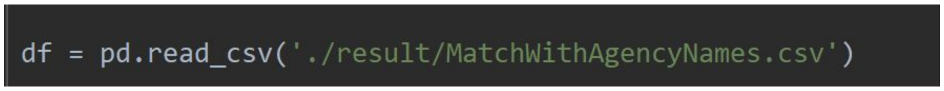
- **Perform matching with MassGovernmentAgencyList:**



```
matchAgencyNames("MassGovernmentAgencyList.csv",data)
```

3. State_surplus.py (given by Taylor's team):

- Filter out luc by 91x, 92x, 97x and Poly_Type by Fee and Tax
- Pass in the dataset to be filter:

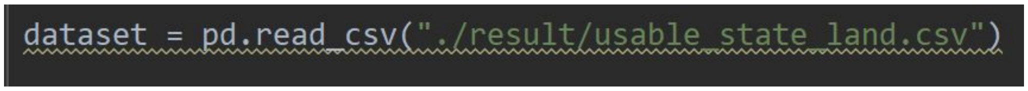


```
df = pd.read_csv('./result/MatchWithAgencyNames.csv')
```

- And run the whole file, the resulting data set is "**usable_state_land.csv**" which is stored in the **result** directory.

4. TransportationHousingDivider.py:

- Added a new column to the dataset to indicate whether property belongs to Transportation or Housing categories.
- Read the dataset through this line in the file:

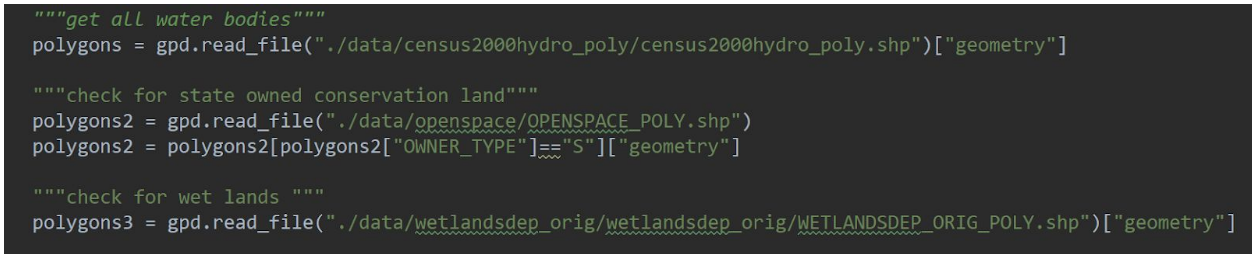


```
dataset = pd.read_csv("./result/usable_state_land.csv")
```

- Run the whole file, and the updated dataset will be store in the **result** directory

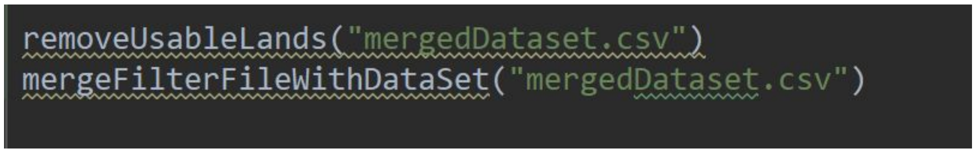
5. RemoveUnusableLand.py:

- Remove property with addresses in unusable lands such as waterbodies, conservation areas etc.



```
"""get all water bodies"""
polygons = gpd.read_file("./data/census2000hydro_poly/census2000hydro_poly.shp")["geometry"]

"""check for state owned conservation land"""
polygons2 = gpd.read_file("./data/openspace/OPENSPACE_POLY.shp")
polygons2 = polygons2[polygons2["OWNER_TYPE"]=="S"]["geometry"]

"""check for wet lands """
polygons3 = gpd.read_file("./data/wetlandsdep_orig/wetlandsdep_orig/WETLANDSDEP_ORIG_POLY.shp")["geometry"]
```

- The three lines above in the removeUsableLands function read the **shapefile** of **unusable land dataset**, and can be changed to any shapefile that we want our dataset to check against.

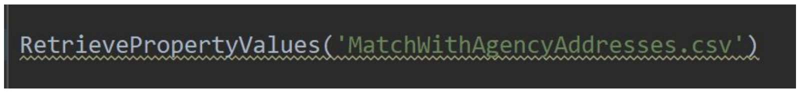


```
removeUsableLands("mergedDataset.csv")
mergeFilterFileWithDataSet("mergedDataset.csv")
```

- The two lines above feed in dataset of **mergedDataset.csv.** By running the first function, it produce **filter.csv** which contains addresses with a column to indicate whether addresses are removable. Then the second function merge the **filter.csv** back with the input dataset to produce **FinalDataset.csv.**

6. DataExtraction.py:

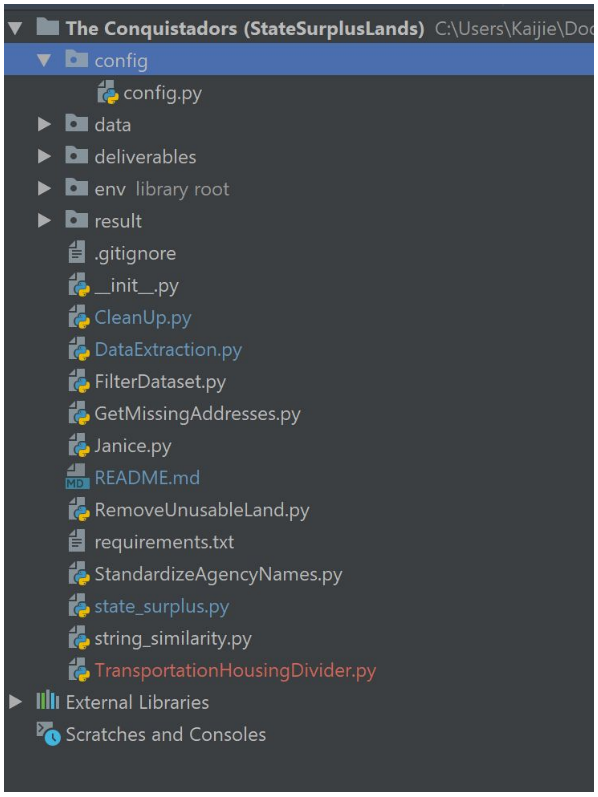
- This file uses ATTOM API integration to evaluate property addresses based on their corresponding neighborhood (more specifically county).
  *This file can simply be ran by inputting which file in the **result** directory to apply the evaluation on:



```
RetrievePropertyValues('MatchWithAgencyAddresses.csv')
```

- In order to run this file, an ATTOM API key is needed. Once key is obtained, put that key in a **config.py** inside **config** folder like below, and this file can be executed.

- The result dataset **AttomEstimateResult.csv** contains a new column **avgsaleprice** to indicate our estimation for the property value based on the neighborhood the property locates.



```
▼ The Conquistadors (StateSurplusLands) C:\Users\Kaijie\Do
    ▼ config
        config.py
    ▶ data
    ▶ deliverables
    ▶ env  library root
    ▶ result
      .gitignore
      __init__.py
      CleanUp.py
      DataExtraction.py
      FilterDataset.py
      GetMissingAddresses.py
      Janice.py
      README.md
      RemoveUnusableLand.py
      requirements.txt
      StandardizeAgencyNames.py
      state_surplus.py
      string_similarity.py
      TransportationHousingDivider.py
▶ External Libraries
  Scratches and Consoles
```

Appendix 3: Description of CSVs in "result" directory

***The CSVs described below are partial results of each data cleaning steps being applied**

1. Original_luc_gt_909.csv:

- Initialize dataset apply filter by luc >909.

2. MatchWithAgencyAddresses.csv:

- match owner addresses with MassGovernmentAgencyList addresses
- Columns:
    1. FullOwnerAddress: combined columns of owner_addr, owner_city and owner_state
    2. std_name: names of the AgencyList names that each address get matched to, reserve original names if no matching is found
    3. matchAgencyList: 1 if match from AgencyList is found, 0 otherwise

3. MatchWithAgencyNames.csv:

- match standardize owner names with agencylist agency names
- Columns:
    1. FullOwnerAddress: combined columns of owner_addr, owner_city and owner_state
    2. std_name: names of the AgencyList names that each address get matched to, reserve original names if no matching is found
    3. matchAgencyList: 1 if match from AgencyList is found, 0 otherwise

4. **Usable_state_land.csv:**

- MatchWithAgencyNames.csv applied with luc 90x, 91x, and 97x, and Poly_type of Fee and Tax, and applied **TransportationHousingDivider.py**
- Columns:
    1. FullOwnerAddress: combined columns of owner_addr, owner_city and owner_state
    2. std_name: names of the AgencyList names that each address get matched to, reserve original names if no matching is found
    3. matchAgencyList: 1 if match from AgencyList is found, 0 otherwise
    4. TransportationOrHousing: 1 for transportation, 0 for housing, -1 for neither

5. mergeDataset.csv:

- Merge **Usable_state_land.csv** dataset with Taylor's team's dataset , highlighted rows are rows with different "owner_name_std"(Taylor's team standardize name) and "std_name"(our standardize name).

- Columns:
    1. agency_name: Taylor's Team name matching with AgencyList
    2. FullOwnerAddress: combined columns of owner_addr, owner_city and owner_state
    3. std_name: names of the AgencyList names that each address get matched to, reserve original names if no matching is found
    4. matchAgencyList: 1 if match from AgencyList is found, 0 otherwise

6. filter.csv:

- Result of apply removeUnusableLands on **mergeDataset.csv**
- Columns:
    1. Address- property address
    2. Latitude- latitude of property address
    3. Longitude - longitude of property address
    4. Point - convert (lat,long) to epsg 26986 format (MassGIS format)
    5. Removable - true if land is unusable, false otherwise

7. FinalDataset.csv:

- Merge **filter.csv** with **mergeDataset.csv** to produce complete final dataset
- Columns:
    1. agency_name: Taylor's Team name matching with AgencyList
    2. FullOwnerAddress: combined columns of owner_addr, owner_city and owner_state
    3. std_name: names of the AgencyList names that each address get matched to, reserve original names if no matching is found
    4. matchAgencyList: 1 if match from AgencyList is found, 0 otherwise
    5. Latitude- latitude of property address
    6. Longitude - longitude of property address
    7. Point - convert (lat,long) to **epsg 26986** format (MassGIS format)
    8. Removable - true if land is unusable, false otherwise

8. AttomEstimateResult.csv:

- Input dataset is **MatchWithAgencyAddresses.csv,** and apply ATTOM land evaluation algorithm.
- Columns:
    1. Address: contains addresses from dataset that match base on the MassGovernmentAgencyList addresses
    2. avgsaleprice: contains ATTOM evaluation result for property

<u>Appendix 4: Python Files and Datasets Overview</u>

- Running these python files give these CSVs:

Python files	csv
FilterDataset.py with original csv	original_luc_gt_909.csv
CleanUp.py with **original_luc_gt_909.csv**	MatchWithAgencyNames.csv MatchWithAgencyAddresses.csv
State_surplus.py with **MatchWithAgencyNames.csv**	usable_state_land.csv
TransportationHousingDivider.py with usable_state_land.csv	usable_state_land.csv
CleanUp.py mergeDatasetWithOtherTeam() with **usable_state_land.csv**	mergedDataset.csv
RemoveUnusableLand.py with **mergedDataset.csv**	filter.csv FinalDataset.csv
DataExtraction.py with **MatchWithAgencyAddresses.csv**	AttomEstimateResult.csv