Team The Conquistadors (StateSurplursLands)

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

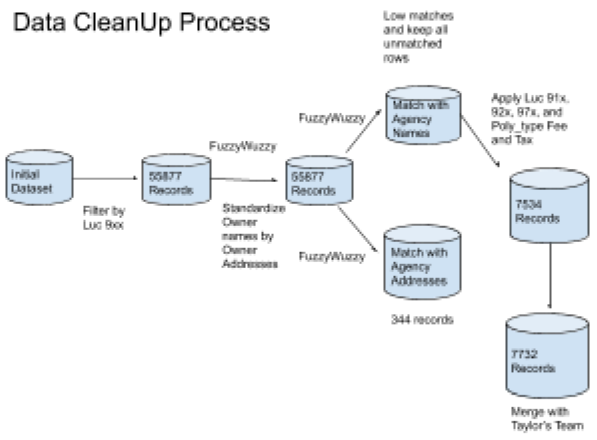
Deliverable 2/3

The main goal of our team for this project is to clean up the dataset as much as we can, since our data cleaning process has become quite complex and time consuming. We will mention the tasks we have completed so far, and what remains to be done. We start off the project with 1.8 million records approximately, and we apply 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 use 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 address with the AgencyList agency names, we further applied a filter of luc code of **91,92,97** and poly type of **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. Currently, we are still working on cleaning and finalizing this dataset between both teams. Finally, we filter out the dataset even more about separating properties into two categories, namely Transportation and Housing. This is indicated by the column TransportationOrHousing.

Another filtering layer we are working on to apply to our dataset is **remove 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 to the dataset to indicate if this address is removable. Once our dataset is finalized, we can easily integrate this.

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. Since at the time of writing this, our merge dataset between 2 teams is not finalized, we did not perform this land evaluation on the dataset yet. However, one difficulty that we might see when we run 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.

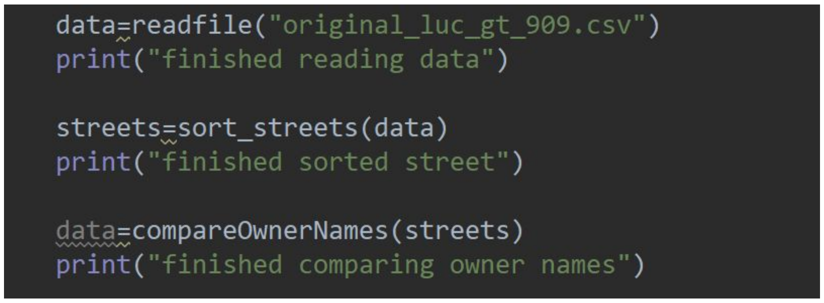


Important Files To Note:

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

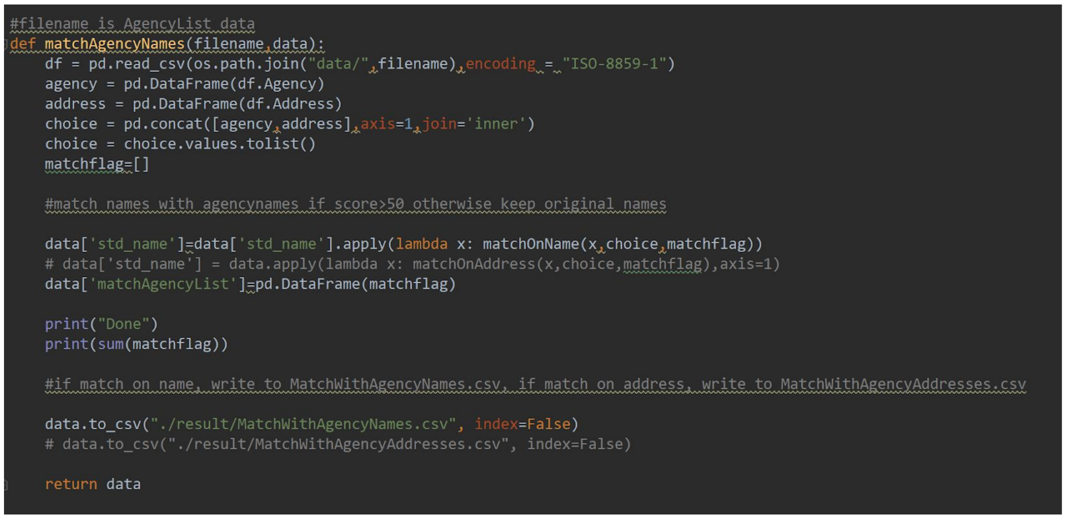
CleanUp.py:

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



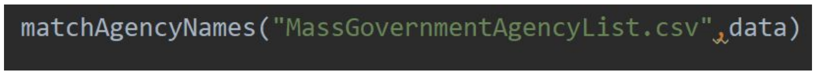
```
data=readfile("original_luc_gt_909.csv")
print("finished reading data")

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

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

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



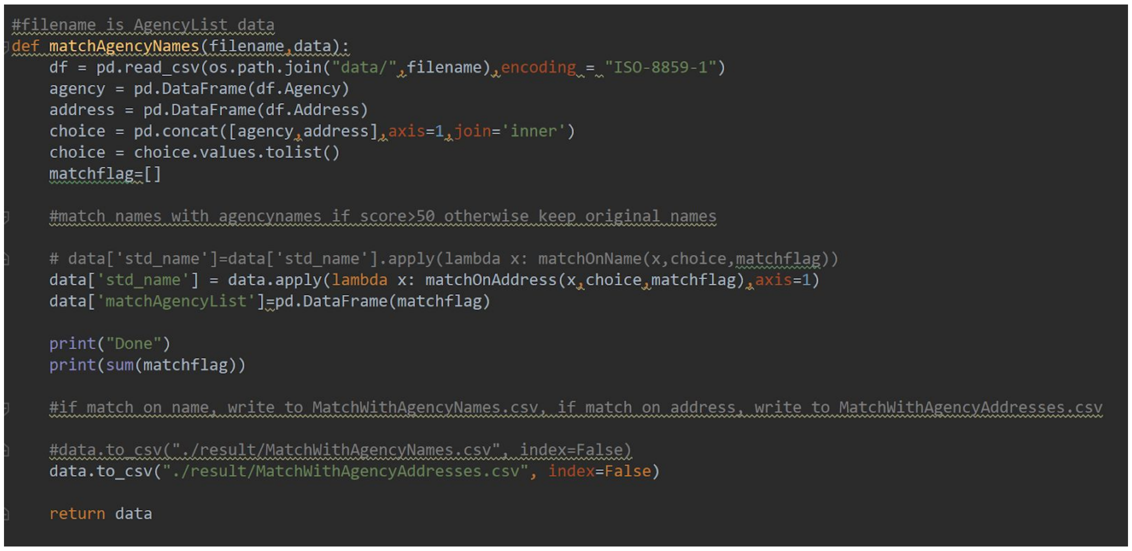
```python
#filename is AgencyList data
def matchAgencyNames(filename,data):
    df = pd.read_csv(os.path.join("data/",filename),encoding = "ISO-8859-1")
    agency = pd.DataFrame(df.Agency)
    address = pd.DataFrame(df.Address)
    choice = pd.concat([agency,address],axis=1,join='inner')
    choice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



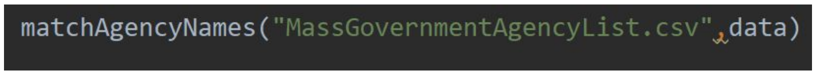
```python
matchAgencyNames("MassGovernmentAgencyList.csv",data)
```

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



```python
#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)

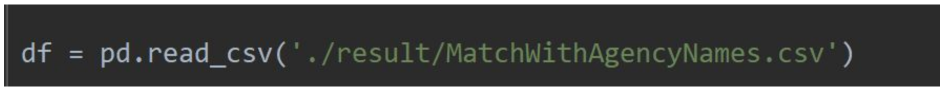


```python
matchAgencyNames("MassGovernmentAgencyList.csv",data)
```

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:

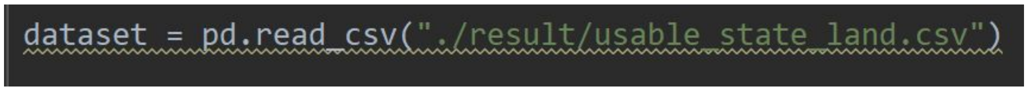


```
df = pd.read_csv('./result/MatchWithAgencyNames.csv')
```

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

TransportationHousingDivider.py - add a new column to dataset to indicate whether property belongs to Transportation or Housing categories.

Read the dataset through this line in the file:



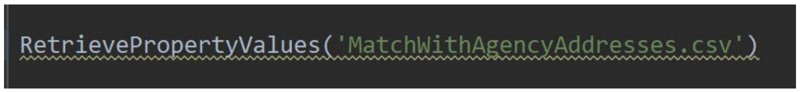
```
dataset = pd.read_csv("./result/usable_state_land.csv")
```

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

DataExtraction.py:

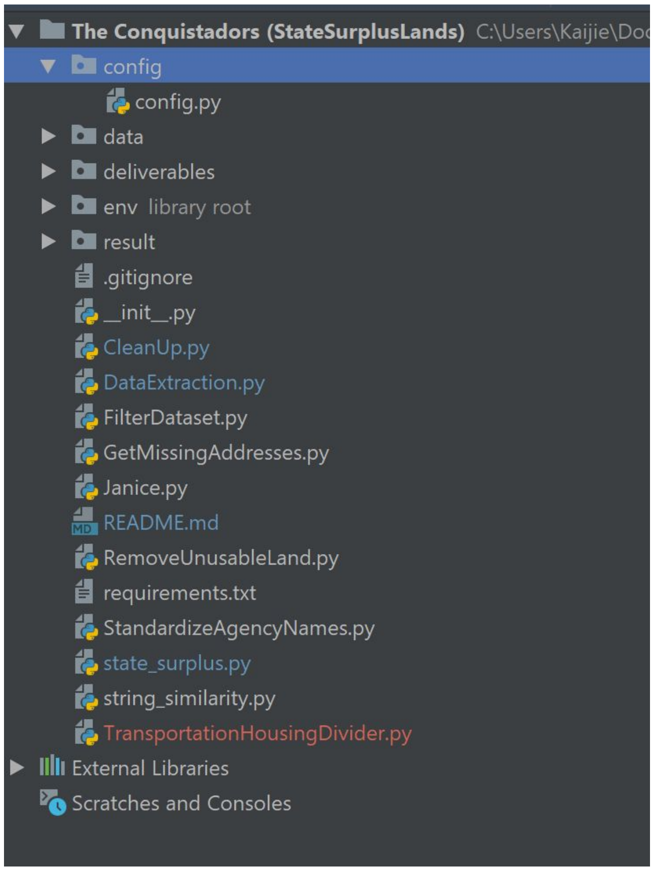
- This file uses ATTOM API integration to evaluate property addresses based on their corresponding neighborhood (more specifically county).

  *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 Conquistadors (StateSurplusLands) C:\Users\Kaijie\Doc
  ▼ 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
```

RemoveUnusableLand.py:

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

```
hydro = gpd.read_file("./data/census2000hydro_poly/census2000hydro_poly.shp")
```

```
data=pd.read_csv("./result/AttomEstimateResult.csv")
```

The first line reads the **shapefile** of **unusable land dataset**, and can be changed any shapefile that we want our dataset to check against.

The second line reads the **input dataset** that we want to filter, and can be changed to any other csv that has **address** column.

Finally, running this file will generate a new column “removable” to indicate whether the property belongs to unsable lands.

Description of CSVs in “result” directory:

-**AttomEstimateResult.csv**:
Address: contains addresses from dataset that match base on the AgencyList addresses,
avgsaleprice: contains ATTOM evluation result for property

-MatchWithAgencyAddresses.csv:
- match owner addresses with agencylist addresses

FullOwnerAddress: combined columns of owner_addr, owner_city and owner_state
Std_name: names of the AgencyList names that each address get matched to, reserve original names if no matching is found
matchAgencyList: 1 if match from AgencyList is found, 0 otherwise

MatchWithAgencyNames.csv:
- match standardize owner names with agencylist agency names

FullOwnerAddress: combined columns of owner_addr, owner_city and owner_state
Std_name: names of the AgencyList names that each address get matched to, reserve original names if no matching is found
matchAgencyList: 1 if match from AgencyList is found, 0 otherwise
mergeDataset.csv:
Merge dataset with Taylor’s team , highlighted rows are rows with different “owner_name_std”(Taylor’s team standardize name) and “std_name”(our standardize name).

FullOwnerAddress: combined columns of owner_addr, owner_city and owner_state

Original_luc_gt_909.csv:
Initialize dataset apply filter by luc >909.

Usable_state_land.csv:
MatchWithAgencyNames.csv applied with luc 90x, 91x, and 97x, and Poly_type of Fee and Tax.