Project 2 Report

Mao-Ho Wang

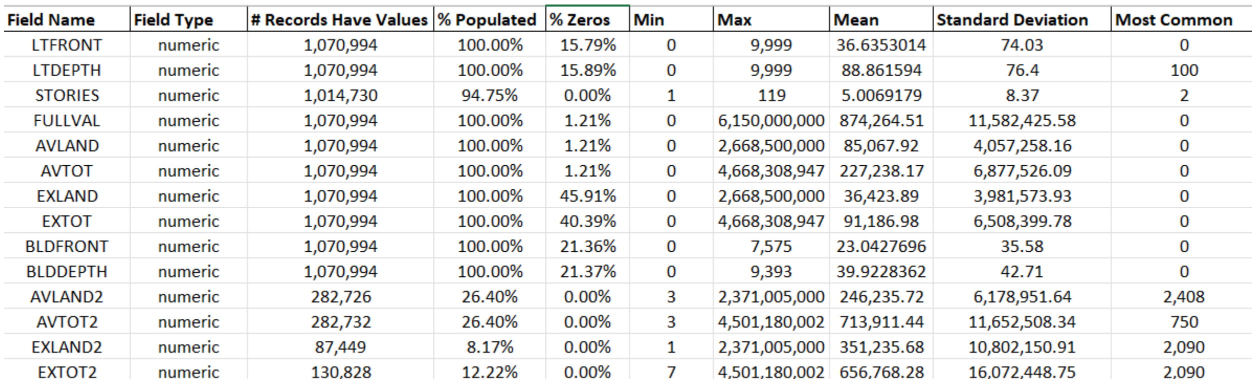
Executive summary:

The aim of this project is to develop an unsupervised model to identify tax fraud for property in NYC. The model will detect properties that show unusual characteristics and values. Looking at a list of the few hundred properties that the model gives a high score, further examination about tax fraud can be conducted.

Data Description:

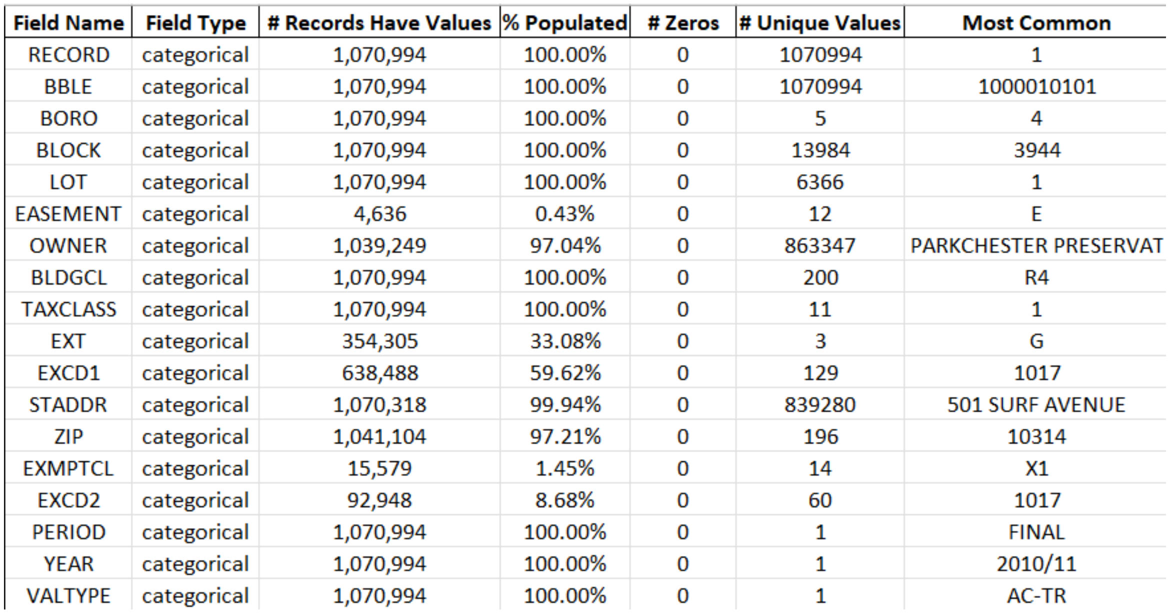
The dataset is NY Property Data, which contains Property valuation and assessment in New York City. The data came from NYC OpenData. There are 32 fields and 1,070,994 records.

Numeric Fields Table



Field Name	Field Type	# Records Have Values	% Populated	% Zeros	Min	Max	Mean	Standard Deviation	Most Common
LTFRONT	numeric	1,070,994	100.00%	15.79%	0	9,999	36.6353014	74.03	0
LTDEPTH	numeric	1,070,994	100.00%	15.89%	0	9,999	88.861594	76.4	100
STORIES	numeric	1,014,730	94.75%	0.00%	1	119	5.0069179	8.37	2
FULLVAL	numeric	1,070,994	100.00%	1.21%	0	6,150,000,000	874,264.51	11,582,425.58	0
AVLAND	numeric	1,070,994	100.00%	1.21%	0	2,668,500,000	85,067.92	4,057,258.16	0
AVTOT	numeric	1,070,994	100.00%	1.21%	0	4,668,308,947	227,238.17	6,877,526.09	0
EXLAND	numeric	1,070,994	100.00%	45.91%	0	2,668,500,000	36,423.89	3,981,573.93	0
EXTOT	numeric	1,070,994	100.00%	40.39%	0	4,668,308,947	91,186.98	6,508,399.78	0
BLDFRONT	numeric	1,070,994	100.00%	21.36%	0	7,575	23.0427696	35.58	0
BLDDEPTH	numeric	1,070,994	100.00%	21.37%	0	9,393	39.9228362	42.71	0
AVLAND2	numeric	282,726	26.40%	0.00%	3	2,371,005,000	246,235.72	6,178,951.64	2,408
AVTOT2	numeric	282,732	26.40%	0.00%	3	4,501,180,002	713,911.44	11,652,508.34	750
EXLAND2	numeric	87,449	8.17%	0.00%	1	2,371,005,000	351,235.68	10,802,150.91	2,090
EXTOT2	numeric	130,828	12.22%	0.00%	7	4,501,180,002	656,768.28	16,072,448.75	2,090

Categorical Fields Table



Field Name	Field Type	# Records Have Values	% Populated	# Zeros	# Unique Values	Most Common
RECORD	categorical	1,070,994	100.00%	0	1070994	1
BBLE	categorical	1,070,994	100.00%	0	1070994	1000010101
BORO	categorical	1,070,994	100.00%	0	5	4
BLOCK	categorical	1,070,994	100.00%	0	13984	3944
LOT	categorical	1,070,994	100.00%	0	6366	1
EASEMENT	categorical	4,636	0.43%	0	12	E
OWNER	categorical	1,039,249	97.04%	0	863347	PARKCHESTER PRESERVAT
BLDGCL	categorical	1,070,994	100.00%	0	200	R4
TAXCLASS	categorical	1,070,994	100.00%	0	11	1
EXT	categorical	354,305	33.08%	0	3	G
EXCD1	categorical	638,488	59.62%	0	129	1017
STADDR	categorical	1,070,318	99.94%	0	839280	501 SURF AVENUE
ZIP	categorical	1,041,104	97.21%	0	196	10314
EXMPTCL	categorical	15,579	1.45%	0	14	X1
EXCD2	categorical	92,948	8.68%	0	60	1017
PERIOD	categorical	1,070,994	100.00%	0	1	FINAL
YEAR	categorical	1,070,994	100.00%	0	1	2010/11
VALTYPE	categorical	1,070,994	100.00%	0	1	AC-TR

Data cleaning:

Exclusion:

Since I am only interested in private properties, I first excluded government easement properties.

```
#remove the records with easement type as goverment
data = data[data["EASEMENT"] != "U"].reset_index(drop=True)
numremoved = numrecords - len(data)
print('# records removed:', numremoved)
```

Later, I created a list of words that are related to government or cemetery to further filter properties that owned by non-private organizations.

```
# create some words for the owner name that might be goverment or a cemetery
gov_list = ['DEPT ', 'DEPARTMENT', 'UNITED STATES','GOVERNMENT',' GOVT ', 'CEMETERY']
owner = list(set(data['OWNER'].to_list()))
owner.pop(0) #remove the nan
remove_list = []
print("Total owner number before removing is ", len(owner))

for i in owner:
   for g in gov_list:
    if g in i and 'STORES' not in i:
        remove_list.append(i)
```

```
# add some others to also be removed
remove_list2.append('THE CITY OF NEW YORK')
remove_list2.append('NYS URBAN DEVELOPMENT')
remove_list2.append('CULTURAL AFFAIRS')
remove_list2.append('NY STATE PUBLIC WORKS')
remove_list2.append("NYC DEP'T OF HIGHWAYS")
remove_list2.append('CITY WIDE ADMINISTRAT')
remove_list2.append('NEW YORK CITY')
remove_list2.append('THE PORT OFNY & NJ')
remove_list2.append('NEW YORK STATE DEPART')
remove_list2.append('CITY AND NON-CITY OWN')
remove_list2.append('SANITATION')
remove_list2.append('NYS DOT')
remove_list2.append('NEW YORK CITY TRANSIT')
remove_list2.append('PORT AUTHORITY OF NY')
remove_list2.append('NEW YORK STATE OWNED')
remove_list2.append('NYC PARK DEPT')
remove_list2.append('PORT OF NEW YORK AUTH')
remove_list2.append('NYC PARK DEPT')
remove_list2.append('LIRR')
remove_list2.append('NY STATE PUBLIC SERV')
remove_list2.append('STATE OF NEW YORK')
remove_list2.append('NYC HIGHWAY DEPT')
```

Here I removed some words that are private properties in the list.

```python
# rremove some of the removes...
remove_list.remove('YORKVILLE TOWERS ASSO')
remove_list.remove('434 M LLC')
remove_list.remove('DEUTSCHE BANK NATIONA')
remove_list.remove('561 11TH AVENUE TMG L')
remove_list.remove('MH RESIDENTIAL 1, LLC')
```

Finally, I filter properties that owned by government or cemetery from the dataset.

```python
numrecords = len(data)
removed = data[data['OWNER'].isin(remove_list)].reset_index(drop=True)
data = data[~data['OWNER'].isin(remove_list)].reset_index(drop=True)
numremoved = numrecords - len(data)
print('# records removed:', numremoved)
```

Field Imputation:

I first conducted imputation on zip column. I created a new column that combined staddr and boro value so that I can use it to match rows that have missing zip with others properties in the same zip.

```python
# concatenate the 'staddr' and 'boro' columns into a new 'staddr_boro' column
data['staddr_boro'] = data[data['STADDR'].notnull()]['STADDR'] + '_' + data[data['BORO'].notnull()]['BORO'].astype(str)
data['staddr_boro']
```

I further filled in the missing zip if those rows can match with other rows that have zip

```python
staddr_boro_zip = {}
for index, staddrboro in data['staddr_boro'].items():
    if staddrboro not in staddr_boro_zip :
        staddr_boro_zip [staddrboro] = data.loc[index, 'ZIP']


# fill in by mapping with street addrees boroughs
data['ZIP'] = data['ZIP'].fillna(data['staddr_boro'].map(staddr_boro_zip))
```

To deal with the left missing zip, I assumed that the data is already sorted by zip. And if the zips before and after are the same, I fill in the missing zip with nearby zip value.

```python
# Assume the data is already sorted by zip. If a zip is missing,
# and the before and after zips are the same, fill in the zip with that value
# There should be a more efficient way to calculate this rather than this slow loop...
for i in range(len(missing_zips)):
    if(data.loc[missing_zips[i]+1,'ZIP'] == data.loc[missing_zips[i]-1,'ZIP']):
        data.loc[missing_zips[i],'ZIP'] = data.loc[missing_zips[i]-1,'ZIP']
```

For the rest that are still missing zip, I filled in with the previous row's zip

```
for i in range(len(missing_zips)):
    data.loc[missing_zips[i],'ZIP'] = data.loc[missing_zips[i]-1,'ZIP']
```

I later conducted imputation on FULLVAL column. I first examined if there are any properties that have zero market value. Since it is unlikely that properties don't have zero market value, and I observed no rows have null FULLVAL, I assigned those zero market value's row with null.

```
len(data[data['FULLVAL']==0])
```

```
10025
```

```
data['FULLVAL'].isnull().sum()
```

```
0
```

```
data['FULLVAL'].replace(0, np.nan, inplace=True)
data['FULLVAL'].isnull().sum()
```

I grouped all records by the combination of TAXCLASS, BORO and BLDCL, calculate the average of FULLVAL for each group, and replace the missing value in that group with the average of the group.

```
data["FULLVAL"] = data.\
                         groupby(['TAXCLASS','BORO','BLDGCL'])['FULLVAL'].transform(lambda x: x.fillna(x.mean()))
data['FULLVAL'].isnull().sum()
```

I then grouped all records by the combination of TAXCLASS and BORO, calculate the average of FULLVAL for each group, and replace the missing value in that group with the average of the group.

```
data["FULLVAL"] = data.\
                         groupby(['TAXCLASS','BORO'])['FULLVAL'].transform(lambda x: x.fillna(x.mean()))
data['FULLVAL'].isnull().sum()
```

Finally, I grouped all records by TAXCLASS, calculate the average of FULLVAL for each group, and replace the missing value in that group with the average of the group.

```
data["FULLVAL"] = data.\
                         groupby(['TAXCLASS'])['FULLVAL'].transform(lambda x: x.fillna(x.mean()))
data['FULLVAL'].isnull().sum()
```

Later, I apply the same approach to AVLAND and AVTOT

```
len(data[data['AVLAND']==0])
```

```
10027
```

```
data['AVLAND'].isnull().sum()
```

```
0
```

```
data['AVLAND'].replace(0, np.nan, inplace=True)
data['AVLAND'].isnull().sum()
```

```
10027
```

```
data["AVLAND"] = data.\
                        groupby(['TAXCLASS','BORO','BLDGCL'])['AVLAND'].transform(lambda x: x.fillna(x.mean()))
data['AVLAND'].isnull().sum()
```

```
7307
```

```
data["AVLAND"] = data.\
                        groupby(['TAXCLASS','BORO'])['AVLAND'].transform(lambda x: x.fillna(x.mean()))
data['AVLAND'].isnull().sum()
```

```
386
```

```
data["AVLAND"] = data.\
                        groupby(['TAXCLASS'])['AVLAND'].transform(lambda x: x.fillna(x.mean()))
data['AVLAND'].isnull().sum()
```

```
len(data[data['AVTOT']==0])
```

```
10025
```

```
data['AVTOT'].isnull().sum()
```

```
0
```

```
data['AVTOT'].replace(0, np.nan, inplace=True)
data['AVTOT'].isnull().sum()
```

```
10025
```

```
data["AVTOT"] = data.\
                        groupby(['TAXCLASS','BORO','BLDGCL'])['AVTOT'].transform(lambda x: x.fillna(x.mean()))
data['AVTOT'].isnull().sum()
```

```
7307
```

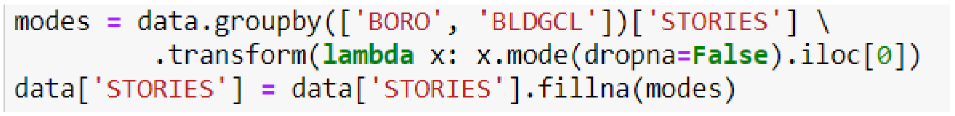
```
data["AVTOT"] = data.\
                        groupby(['TAXCLASS','BORO'])['AVTOT'].transform(lambda x: x.fillna(x.mean()))
data['AVTOT'].isnull().sum()
```

```
386
```

```
data["AVTOT"] = data.\
                        groupby(['TAXCLASS'])['AVTOT'].transform(lambda x: x.fillna(x.mean()))
data['AVTOT'].isnull().sum()
```

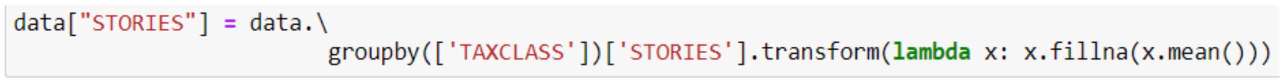
```
0
```

I conducted the imputation on STORIES column. I first filled in the missing values with the most frequent value that have same BORO and BLDGCL.



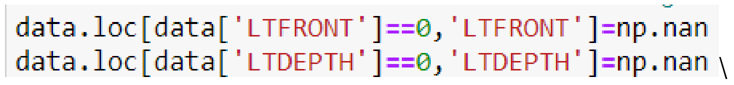
```
modes = data.groupby(['BORO', 'BLDGCL'])['STORIES'] \
          .transform(lambda x: x.mode(dropna=False).iloc[0])
data['STORIES'] = data['STORIES'].fillna(modes)
```

Later, I filled in the rest with the mean value that have same taxclass.



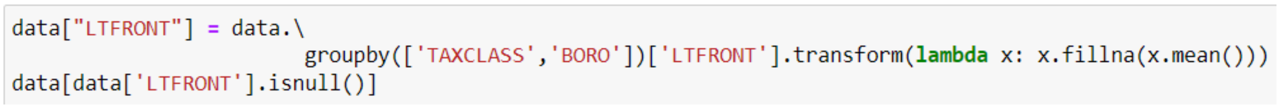
```
data["STORIES"] = data.\
                        groupby(['TAXCLASS'])['STORIES'].transform(lambda x: x.fillna(x.mean()))
```

After that, I started dealing with LTFRONT, LTDEPTH, BLDDEPTH, BLDFRONT. I first convert LTFRONT and LTDEPTH that have zero value to null as it's unlikely these two columns have zero value.



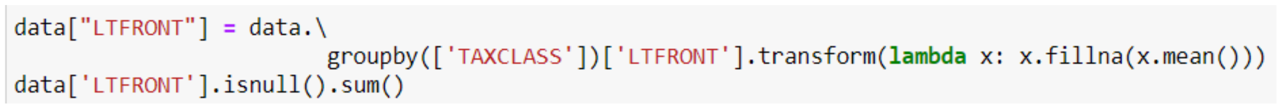
```
data.loc[data['LTFRONT']==0,'LTFRONT']=np.nan
data.loc[data['LTDEPTH']==0,'LTDEPTH']=np.nan \
```

To do imputation on LTFRONT, I grouped all records by the combination of TAXCLASS and BORO, calculated the average of LTFRONT for each group, and replaced the missing value in that group with the average of the group.



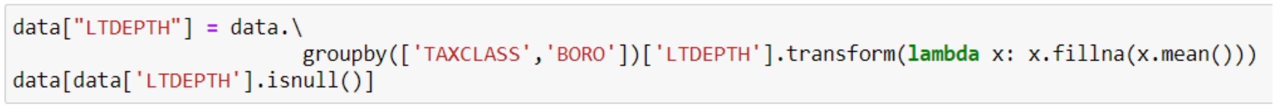
```
data["LTFRONT"] = data.\
                        groupby(['TAXCLASS','BORO'])['LTFRONT'].transform(lambda x: x.fillna(x.mean()))
data[data['LTFRONT'].isnull()]
```

I then grouped all records by TAXCLASS, calculated the average of LTFRONT for each group, and replace the missing value in that group with the average of the group.

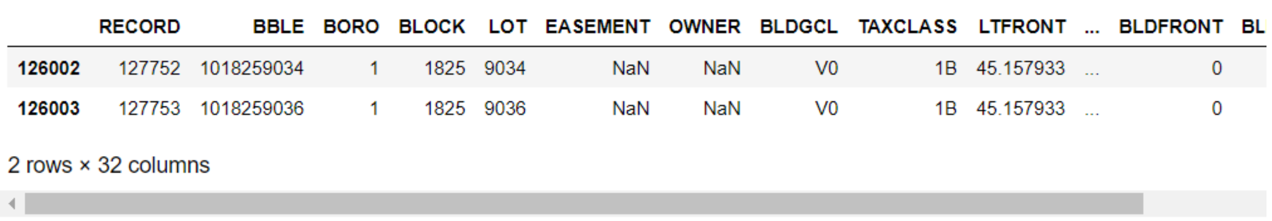


```
data["LTFRONT"] = data.\
                        groupby(['TAXCLASS'])['LTFRONT'].transform(lambda x: x.fillna(x.mean()))
data['LTFRONT'].isnull().sum()
```

I applied same approach to LTDEPTH

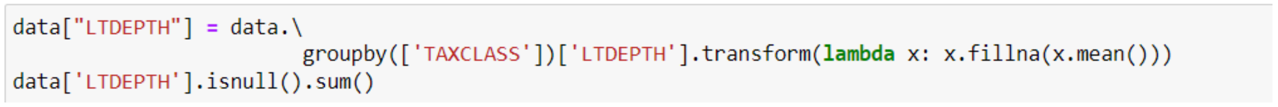


```
data["LTDEPTH"] = data.\
                        groupby(['TAXCLASS','BORO'])['LTDEPTH'].transform(lambda x: x.fillna(x.mean()))
data[data['LTDEPTH'].isnull()]
```



	RECORD	BBLE	BORO	BLOCK	LOT	EASEMENT	OWNER	BLDGCL	TAXCLASS	LTFRONT	...	BLDFRONT	BL
126002	127752	1018259034	1	1825	9034	NaN	NaN	V0	1B	45.157933	...	0	
126003	127753	1018259036	1	1825	9036	NaN	NaN	V0	1B	45.157933	...	0	

2 rows × 32 columns



```
data["LTDEPTH"] = data.\
                        groupby(['TAXCLASS'])['LTDEPTH'].transform(lambda x: x.fillna(x.mean()))
data['LTDEPTH'].isnull().sum()
```

I grouped all records by the combination of TAXCLASS, BORO and BLDCL, calculate the average of BLDFRONT for each group, and replace the missing value in that group with the average of the group

```
data['BLDFRONT'] = data.\
                    groupby(['TAXCLASS','BORO','BLDGCL'])['BLDFRONT'].transform(lambda x: x.fillna(x.mean()))
data['BLDFRONT'].isnull().sum()
```

I then grouped all records by the combination of TAXCLASS and BORO, calculate the average of BLDFRONT for each group, and replace the missing value in that group with the average of the group.

```
data['BLDFRONT'] = data.\
                    groupby(['TAXCLASS','BORO'])['BLDFRONT'].transform(lambda x: x.fillna(x.mean()))
data['BLDFRONT'].isnull().sum()
```

Finally, I grouped all records by TAXCLASS, calculated the average of BLDFRONT for each group, and replace the missing value in that group with the average of the group.

```
data['BLDFRONT'] = data.\
                    groupby(['TAXCLASS'])['BLDFRONT'].transform(lambda x: x.fillna(x.mean()))
data['BLDFRONT'].isnull().sum()
```

I applied the same approach to BLDDEPTH.

```
data['BLDDEPTH'].isnull().sum()
```

```
0
```

```
data['BLDDEPTH'] = data.\
                    groupby(['TAXCLASS','BORO','BLDGCL'])['BLDDEPTH'].transform(lambda x: x.fillna(x.mean()))
data['BLDDEPTH'].isnull().sum()
```

```
0
```

```
data['BLDDEPTH'] = data.\
                    groupby(['TAXCLASS','BORO'])['BLDDEPTH'].transform(lambda x: x.fillna(x.mean()))
data['BLDDEPTH'].isnull().sum()
```

```
0
```

```
data['BLDDEPTH'] = data.\
                    groupby(['TAXCLASS'])['BLDDEPTH'].transform(lambda x: x.fillna(x.mean()))
data['BLDDEPTH'].isnull().sum()
```

Variables Creation:

I started creating as many variables as possible. I first created three variables about size as price per size is important to detect potential fraud. I also add 0.0001 to the value to make sure when divide by these variables, no error will generate.

```
# epsilon is an arbitrary small number to make sure we don't divide by zero
epsilon = .0001
data['ltsize'] = data['LTFRONT'] * data['LTDEPTH'] + epsilon
data['bldsize'] = data['BLDFRONT'] * data['BLDDEPTH'] + epsilon
data['bldvol'] = data['bldsize'] * data['STORIES'] + epsilon
```

Later, I created 9 variables using the market value, land value, and total value divided by the size variables to examine the price per size.

```
data['r1'] = data['FULLVAL'] / data['ltsize']
data['r2'] = data['FULLVAL'] / data['bldsize']
data['r3'] = data['FULLVAL'] / data['bldvol']
data['r4'] = data['AVLAND'] / data['ltsize']
data['r5'] = data['AVLAND'] / data['bldsize']
data['r6'] = data['AVLAND'] / data['bldvol']
data['r7'] = data['AVTOT'] / data['ltsize']
data['r8'] = data['AVTOT'] / data['bldsize']
data['r9'] = data['AVTOT'] / data['bldvol']
```

I also inverse the newly created variable because fraud usually occurs when the price per size is extremely low. However, it's hard to detect the outlier without inversing it as the small price per size values are still within a few standard deviations. Therefore, I inverse it to make small price per size become extremely large.

```
# add in the inverse of all the 9 primary variables. See slides for why.
for col in data.columns[36:]:
    data[col+'inv'] = 1/(data[col] + epsilon)
```

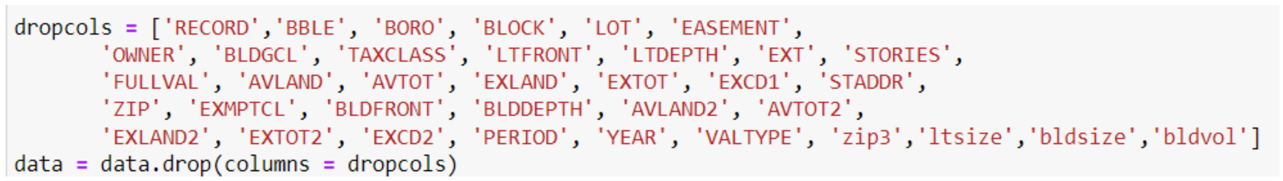
Later, I calculated the mean of previous 18 variables grouped by ZIP AND TAXCLASS, and divided each of the 18 ratio variables by the mean grouped by ZIP AND TAXCLASS. These columns can find out the price per size difference between the property and other properties within the same group.

```
# Standardized Ratio by Group
vars18 = ['r1','r2','r3','r4','r5','r6','r7','r8','r9',
          'r1inv','r2inv','r3inv','r4inv','r5inv','r6inv','r7inv','r8inv','r9inv']
zip5_mean = data.groupby('ZIP')[vars18].mean()
taxclass_mean = data.groupby('TAXCLASS')[vars18].mean()
data = data.join(zip5_mean, on='ZIP', rsuffix='_zip5')
data = data.join(taxclass_mean, on='TAXCLASS', rsuffix='_taxclass')
rsuffix = ['_zip5', '_taxclass']
for var in vars18:
    for r in rsuffix:
        data[str(var)+r] = data[var] / data[str(var)+r]
```

I also created two more variables that could be useful for the model. The first variable compares the 3 $ value measure. The second variable compares the building to lot sizes.

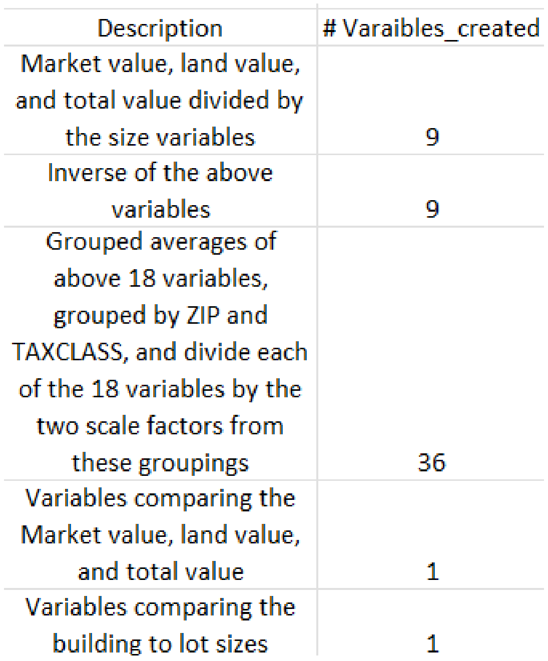
```
# include two more possibly interesting variables
data['value_ratio'] = data['FULLVAL']/(data['AVLAND']+data['AVTOT'])
data['value_ratio'] = data['value_ratio']/data['value_ratio'].mean()
data['value_ratio'] = np.where(data['value_ratio'] < 1, 1/(data['value_ratio']+epsilon), data['value_ratio'])
data['size_ratio'] = data['bldsize'] / (data['ltsize']+1)
```

Finally, I drop columns that will not be used for modeling.



```
dropcols = ['RECORD','BBLE', 'BORO', 'BLOCK', 'LOT', 'EASEMENT',
       'OWNER', 'BLDGCL', 'TAXCLASS', 'LTFRONT', 'LTDEPTH', 'EXT', 'STORIES',
       'FULLVAL', 'AVLAND', 'AVTOT', 'EXLAND', 'EXTOT', 'EXCD1', 'STADDR',
       'ZIP', 'EXMPTCL', 'BLDFRONT', 'BLDDEPTH', 'AVLAND2', 'AVTOT2',
       'EXLAND2', 'EXTOT2', 'EXCD2', 'PERIOD', 'YEAR', 'VALTYPE', 'zip3','ltsize','bldsize','bldvol']
data = data.drop(columns = dropcols)
```

Model Variable table:



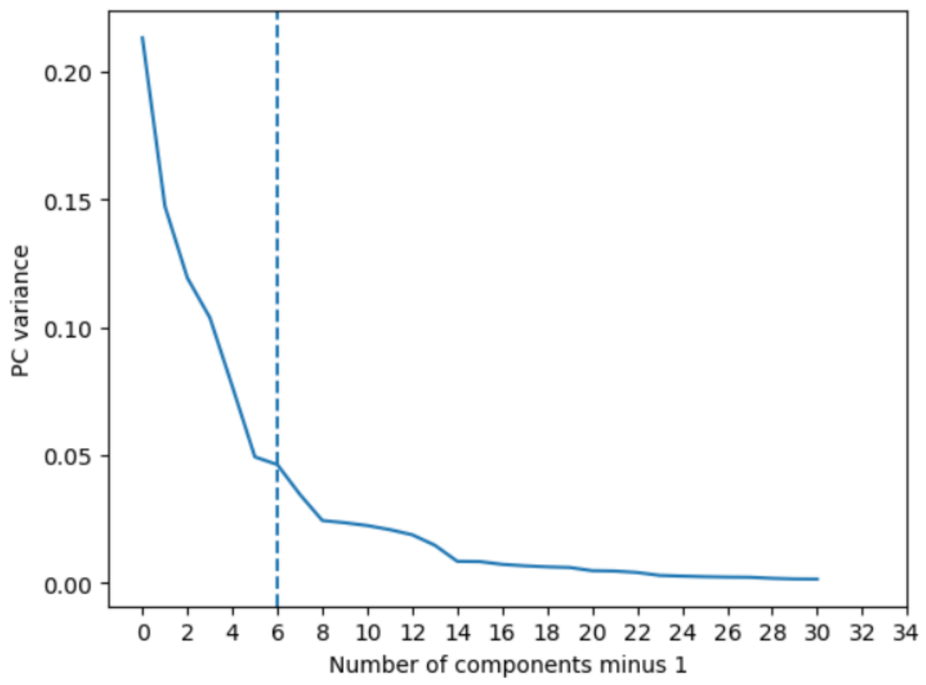
Description	# Varaibles_created
Market value, land value, and total value divided by the size variables	9
Inverse of the above variables	9
Grouped averages of above 18 variables, grouped by ZIP and TAXCLASS, and divide each of the 18 variables by the two scale factors from these groupings	36
Variables comparing the Market value, land value, and total value	1
Variables comparing the building to lot sizes	1

Dimensionality Reduction

To reduce the dimensionality for the unsupervised model, I conducted PCA for the variables. Before doing it, I first z scale all variables so all dimensions have the same scaling and the PCA can accurately detect new dimensions that have high variance.

```
# zscale all the variables
data_zs = (data - data.mean()) / data.std()
data_zs_save = data_zs.copy()
data_zs.describe().transpose()
```

After z scaling, I started conducting PCA and created a scree plot to measure how many dimensions I should keep.



As the result shows, the top 4 dimensions have high PC variance, but the variance started decreasing slowly after that. I decided to keep only the top 4 dimensions for the model.

```
data_pca.head(5)
```

	PC1	PC2	PC3	PC4
0	-0.293868	0.866511	0.633140	-0.115214
1	-0.300357	1.365950	1.104043	-0.078121
2	-0.314117	2.405358	2.061290	-0.004450
3	-0.282661	0.032445	-0.153859	-0.169897
4	-0.287792	0.403883	0.196843	-0.148305

Since I only keep 4 PCs, I decided to use z scale again to make all four PCs equally important when developing fraud score.

```
data_pca_zs = (data_pca - data_pca.mean()) / data_pca.std()
data_pca_zs.describe()
```

	PC1	PC2	PC3	PC4
count	1.044493e+06	1.044493e+06	1.044493e+06	1.044493e+06
mean	4.796928e-16	1.403844e-15	1.768613e-16	-1.385506e-16
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
min	-4.370107e+00	-3.050259e-01	-2.159868e+02	-4.891476e+01
25%	-5.115316e-02	-5.712310e-02	-8.996535e-02	-5.457550e-02
50%	-2.539389e-02	-4.418371e-02	-2.957867e-02	-4.369926e-02
75%	-8.645789e-03	-2.629883e-02	2.617214e-02	-2.164053e-02
max	5.647748e+02	6.923782e+02	6.107024e+02	4.169278e+02

Anomaly Detection Algorithms

After doing PCA, I developed two scoring methods for fraud detection.

For the first method, I used two Minkowski distances to calculate the distance to the origin for each point. This can help detecting data points that are far away from the group, which is outlier.

```
p1 = 2
p2 = 2
ntop = 1000
```

```
oop1 = 1/p1
score1 = (((data_pca_zs).abs()**p1).sum(axis=1))**oop1
score1.head(10)
```

For the second method, I built an autoencoder using a neural network (one layer, 3 nodes) to output the original vector input. As neural network will minimize the error on the whole dataset, records that are different to the normal records won’t be reproduced well. Thus, I used the error from an autoencoder as the second score.

```
NNmodel = MLPRegressor(hidden_layer_sizes=(3),activation='logistic',max_iter=50,random_state=1)
NNmodel.fit(data_pca_zs,data_pca_zs)
```

```
# calculate score 2 as the error of an autoencoder
pca_out = NNmodel.predict(data_pca_zs)
error = pca_out - data_pca_zs
oop2 = 1/p2
score2 = ((error.abs()**p2).sum(axis=1))**oop2
```

After getting both scores, I ranked data based on the score they got.

```
scores['score1 rank'] = scores['score1'].rank()
scores['score2 rank'] = scores['score2'].rank()
scores.head(20)
```

	score1	score2	RECORD	score1 rank	score2 rank
0	0.400539	0.106112	9	970724.0	750298.0
1	0.645725	0.207066	10	998565.0	958874.0
2	1.159632	0.411705	11	1022236.0	1004541.0
3	0.123781	0.081817	12	665076.0	696793.0
4	0.190450	0.034974	13	815448.0	332249.0
5	0.176590	0.035030	14	774480.0	333166.0
6	0.933790	0.325206	15	1014565.0	993302.0
7	0.904465	0.313225	16	1013358.0	991244.0

Finally, I used average rank from two methods as the final score. I used average because there's no label and I couldn't know which method works better.

```
scores['final'] = (.5*scores['score1 rank'] + .5*scores['score2 rank'])
scores_sorted = scores.sort_values(by='final', ascending=False)
scores_sorted.head(20)
```

	score1	score2	RECORD	score1 rank	score2 rank	final
897398	725.633541	696.124310	917942	1044493.0	1044493.0	1044493.0
1040913	638.546816	616.786517	1067360	1044492.0	1044492.0	1044492.0
110100	565.764235	540.352530	111420	1044491.0	1044491.0	1044491.0
934505	525.283790	500.581446	956520	1044490.0	1044490.0	1044490.0
641107	412.042949	387.635286	658933	1044488.0	1044489.0	1044488.5

Results:

By sorting the final score from high to low, we can examine the price per size variables of data points on the top to find out the reason that makes the score high. For instance, a significantly high r1inv could result in a high fraud score. This could be a potential tax fraud because the property that has high lot size shouldn't have very low land value.

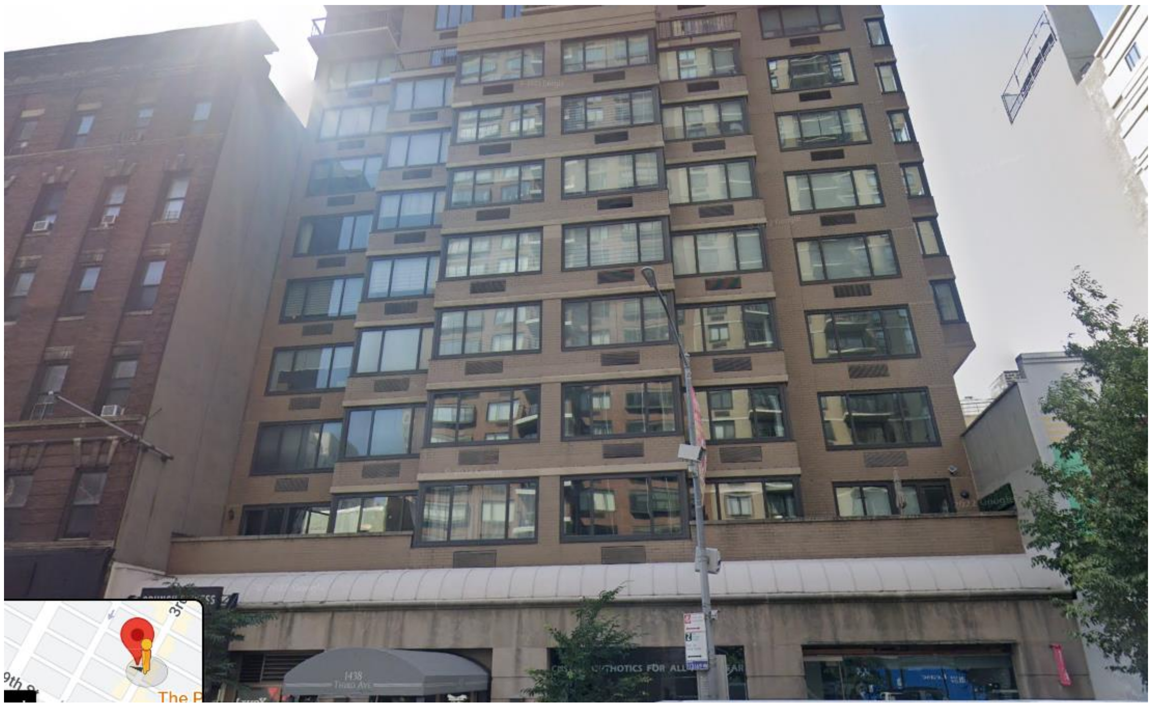
I provided five interesting cases that have high fraud score below:

Case 1:

Record: **170125**

OWNER	BOXWOOD FLTD PARNTERS	LTFRONT	75
ADDRESS	1438 3 AVENUE	LTDEPTH	93
FULLVAL	296508	BLDFRONT	7575
AVLAND	22896	BLDDEPTH	9393
AVTOT	133429	STORIES	31

FULLVAL too low



The property is suspicious because it has multiple stories but low market value. This causes the high r2inv and r3inv values. The land value is also low, which results in high r5inv. Therefore, the model detects it as potential fraud.

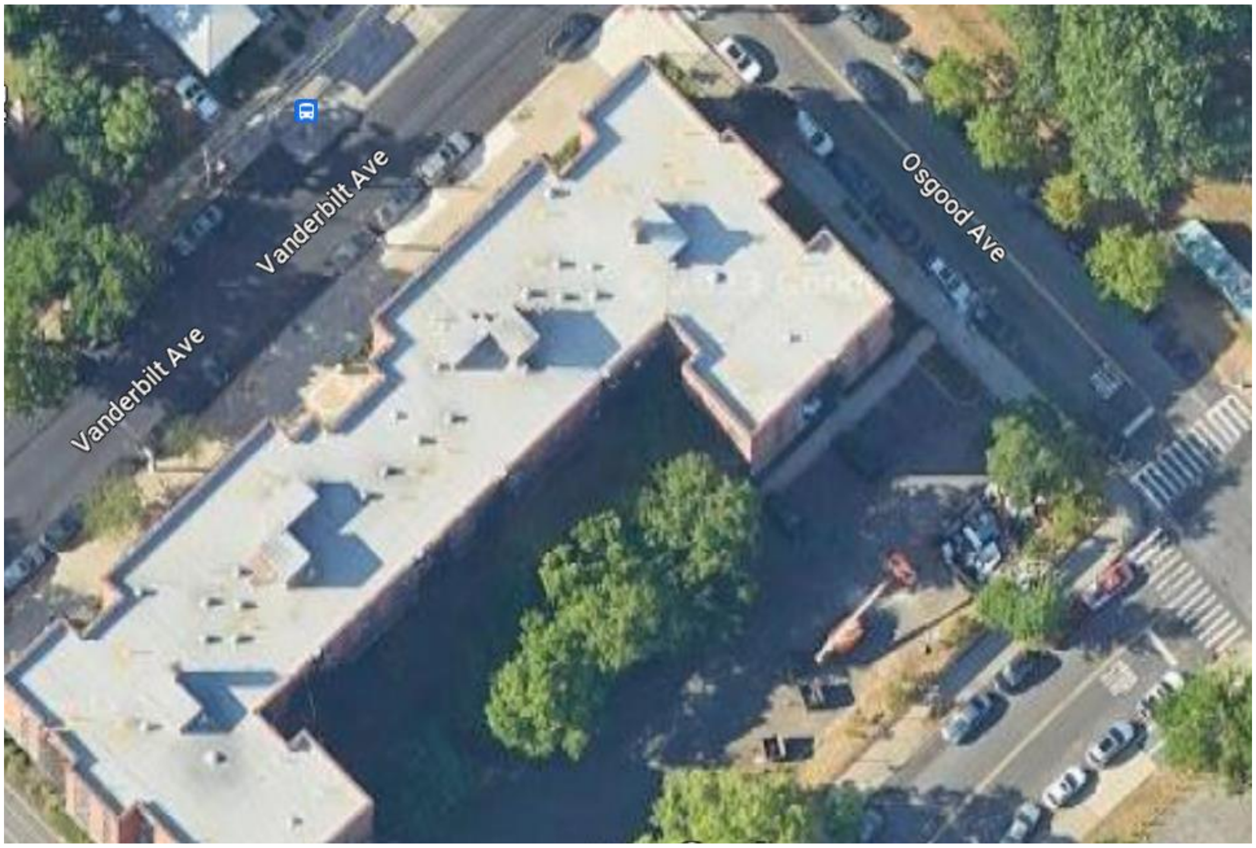
Therefore, it is recommended to examine the market price of this property again to prevent fraud.

Case 2:

Record: **996722**

OWNER	IMPERIAL COURT HOMEOW	LTFRONT	300
ADDRESS	OSGOOD AVENUE	LTDEPTH	400
FULLVAL	250	BLDFRONT	0
AVLAND	1	BLDDEPTH	0
AVTOT	1	STORIES	None

LTFRONT and LTDEPTH are high, but FULLVAL, AVLAND and AVTOT are extremely low.



The property could be a potential fraud because its Lot width and depth are quite high. However, the market value and land value are surprisingly low. This causes the high r4inv and r7inv values. It's implausible that a property in New York has this low value.

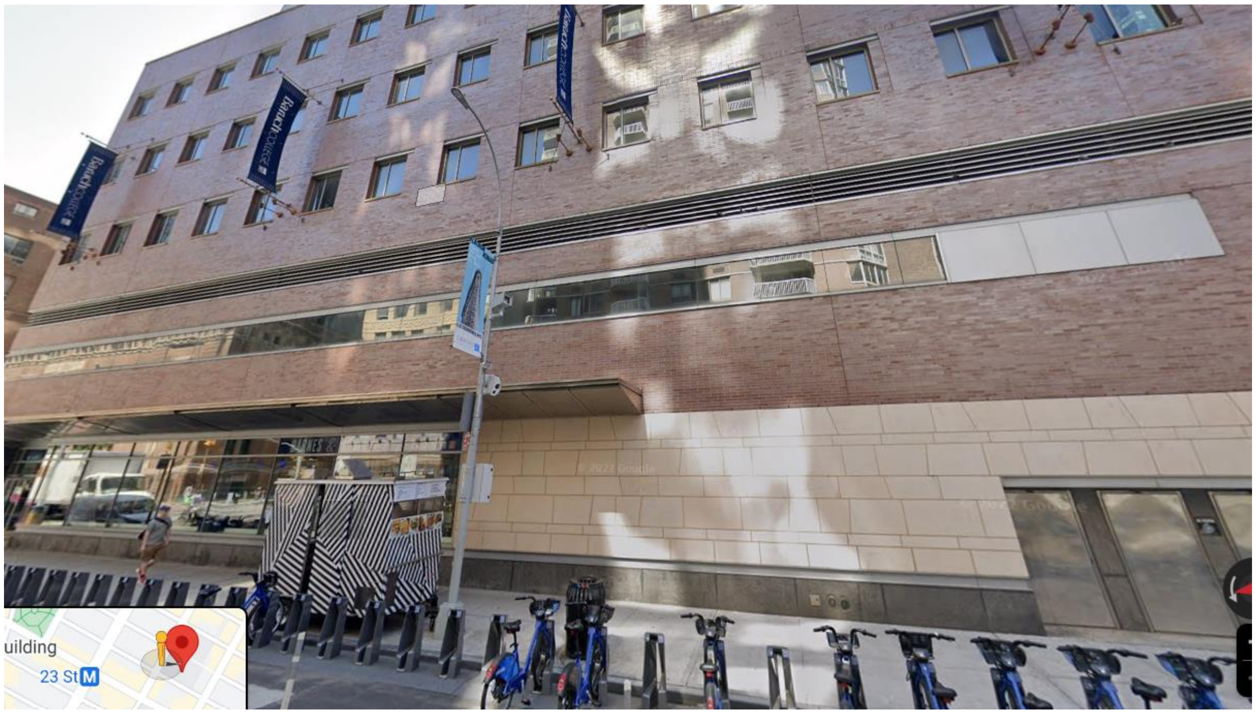
Therefore, it is recommended to examine the market value and land value of this property again to prevent fraud.

Case 3:

Record: **47984**

OWNER	BERKOWTIZ, ULWT LOUIS	LTFRONT	39
ADDRESS	49 LEXINGTON AVENUE	LTDEPTH	50
FULLVAL	138000000	BLDFRONT	39
AVLAND	11025000	BLDDEPTH	50
AVTOT	62100000	STORIES	2

LTFRONT and LTDEPTH are low.



The property has been identified as a potential fraud because its Lot width and Lot depth are too low. However, the market value and land value are quite high. This causes a high r7 value.

Therefore, it is recommended to examine the Lot width and Lot depth of this property again to prevent fraud.

Case 4:

Record: **561383**

OWNER	YILDIZ HOLDING A.S.	LTFRONT	930
ADDRESS	5120 AVENUE U	LTDEPTH	650
FULLVAL	258000000	BLDFRONT	0
AVLAND	40590000	BLDDEPTH	0
AVTOT	116100000	STORIES	2

Value to size ratios are too high in that zip code.

The property has been identified as a potential fraud. Even though the value and size look normal, its price per size are higher than other properties within the same zip code, which causes high r2_zip5, r3_zip5, r4_zip5, r5_zip5, r6_zip5, r7_zip5, r8_zip5, r9_zip5

Therefore, it is recommended to examine the value and size of this property again to prevent fraud.

Case 5:

Record: **116647**

OWNER	MF ASSOCIATES OF NEW	LTFRONT	25
ADDRESS	1849 2 AVENUE	LTDEPTH	75
FULLVAL	161000000	BLDFRONT	70
AVLAND	19215000	BLDDEPTH	456
AVTOT	72450000	STORIES	35

Value to size ratios are too high in that zip code and tax class.

The property has been identified as a potential fraud. Even though the value and lot size look normal, its price per lot size are higher than other properties within the same zip code and same tax class, which causes high r1_zip5, r4_zip5, r7_zip5, r1_taxclass, r4_taxclass, and r7_taxclass**.**

Therefore, it is recommended to examine the value and size of this property again to prevent fraud.

Summary:

To sum up, the purpose of this project is to develop an unsupervised model that can detect potential tax fraud related to properties in NYC. The primary focus of the model is to identify properties that show abnormal characteristics and land values, which could indicate fraudulent activity. The model can generate a list of several hundred properties with high fraud scores. These properties can then undergo a more thorough investigation to determine if tax fraud has indeed occurred.

To achieve the goal, I first created a data quality report (attached in appendix) for better understanding of the dataset. Later, I conducted data cleaning by addressing exclusion and applying various methods for imputation. I later created various price-per-size related variables for the model. Further, I applied Z-scale, PCA, and Z-scale to reduce the dimensionality, where I used scree plot to determine the number of PCs. I further developed two scoring methods for fraud detection. One is Minkowski distances of the data points to origin after PCA, another one is the error from an autoencoder built by a neural network. I determined the final score for fraud detection using the average ranking of two scoring methods within the whole data set. Looking at properties that have high fraud score, we can further check the price per size variables of them to find out the reason that makes the score high and whether they are tax fraud or not.

The model can be modified after looking at the top records. If those records aren't what experts consider tax fraud nor the type of properties they are interested in, the algorithm can be adjusted by improving exclusions and variables until the list of the top records look like the potential tax fraud that experts are interested in.

Appendix:

Data Quality Report

1. Data Description

The dataset is **NY Property Data**, which contains **Property valuation and assement** in New York City. The data came from NYC OpenData. There are **32 fields** and **1,070,994 records**.

2. Summary Tables

Numeric Fields Table

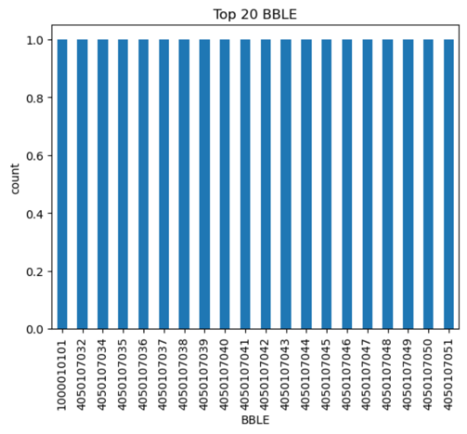
Field Name	Field Type	# Records Have Values	% Populated	% Zeros	Min	Max	Mean	Standard Deviation	Most Common
LTFRONT	numeric	1,070,994	100.00%	15.79%	0	9,999	36.6353014	74.03	0
LTDEPTH	numeric	1,070,994	100.00%	15.89%	0	9,999	88.861594	76.4	100
STORIES	numeric	1,014,730	94.75%	0.00%	1	119	5.0069179	8.37	2
FULLVAL	numeric	1,070,994	100.00%	1.21%	0	6,150,000,000	874,264.51	11,582,425.58	0
AVLAND	numeric	1,070,994	100.00%	1.21%	0	2,668,500,000	85,067.92	4,057,258.16	0
AVTOT	numeric	1,070,994	100.00%	1.21%	0	4,668,308,947	227,238.17	6,877,526.09	0
EXLAND	numeric	1,070,994	100.00%	45.91%	0	2,668,500,000	36,423.89	3,981,573.93	0
EXTOT	numeric	1,070,994	100.00%	40.39%	0	4,668,308,947	91,186.98	6,508,399.78	0
BLDFRONT	numeric	1,070,994	100.00%	21.36%	0	7,575	23.0427696	35.58	0
BLDDEPTH	numeric	1,070,994	100.00%	21.37%	0	9,393	39.9228362	42.71	0
AVLAND2	numeric	282,726	26.40%	0.00%	3	2,371,005,000	246,235.72	6,178,951.64	2,408
AVTOT2	numeric	282,732	26.40%	0.00%	3	4,501,180,002	713,911.44	11,652,508.34	750
EXLAND2	numeric	87,449	8.17%	0.00%	1	2,371,005,000	351,235.68	10,802,150.91	2,090
EXTOT2	numeric	130,828	12.22%	0.00%	7	4,501,180,002	656,768.28	16,072,448.75	2,090

Categorical Fields Table

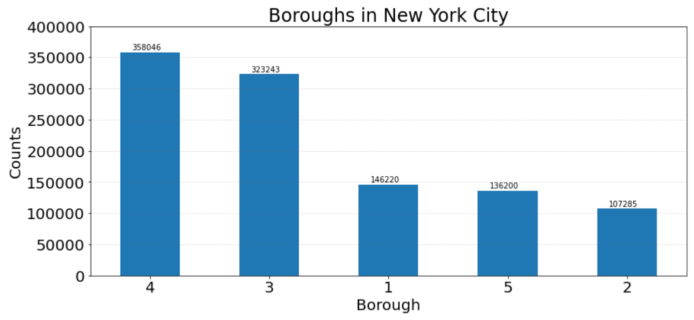
Field Name	Field Type	# Records Have Values	% Populated	# Zeros	# Unique Values	Most Common
RECORD	categorical	1,070,994	100.00%	0	1070994	1
BBLE	categorical	1,070,994	100.00%	0	1070994	1000010101
BORO	categorical	1,070,994	100.00%	0	5	4
BLOCK	categorical	1,070,994	100.00%	0	13984	3944
LOT	categorical	1,070,994	100.00%	0	6366	1
EASEMENT	categorical	4,636	0.43%	0	12	E
OWNER	categorical	1,039,249	97.04%	0	863347	PARKCHESTER PRESERVAT
BLDGCL	categorical	1,070,994	100.00%	0	200	R4
TAXCLASS	categorical	1,070,994	100.00%	0	11	1
EXT	categorical	354,305	33.08%	0	3	G
EXCD1	categorical	638,488	59.62%	0	129	1017
STADDR	categorical	1,070,318	99.94%	0	839280	501 SURF AVENUE
ZIP	categorical	1,041,104	97.21%	0	196	10314
EXMPTCL	categorical	15,579	1.45%	0	14	X1
EXCD2	categorical	92,948	8.68%	0	60	1017
PERIOD	categorical	1,070,994	100.00%	0	1	FINAL
YEAR	categorical	1,070,994	100.00%	0	1	2010/11
VALTYPE	categorical	1,070,994	100.00%	0	1	AC-TR

3. Visualization of Each Field

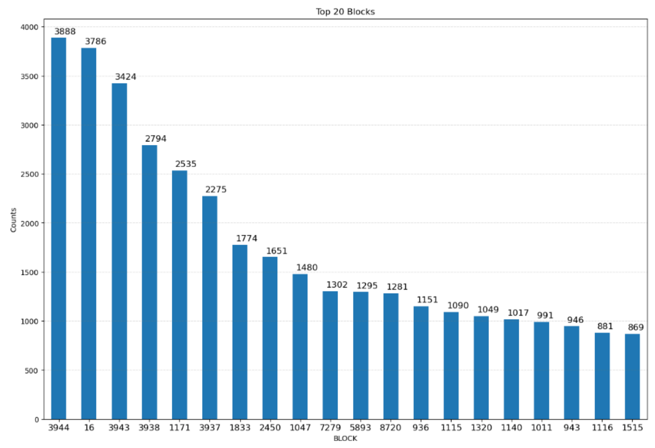
1) **Field Name: RECORD**
   Description: Ordinal unique positive integer for each property valuation and assessment, from 1 to 1,070,994
2) **Field Name: BBLE**
   Description: Combination of Boro, Block, Lot, and Easement code. The plot shows the top 20 values of BBLE. There are 1,070,994 unique values.



3) **Field Name: BORO**
Description: Borough. The distribution shows the number of boroughs in New York City. The distribution shows the 5 field values of Borough. The most common value is 4, with a total count of 358,046.

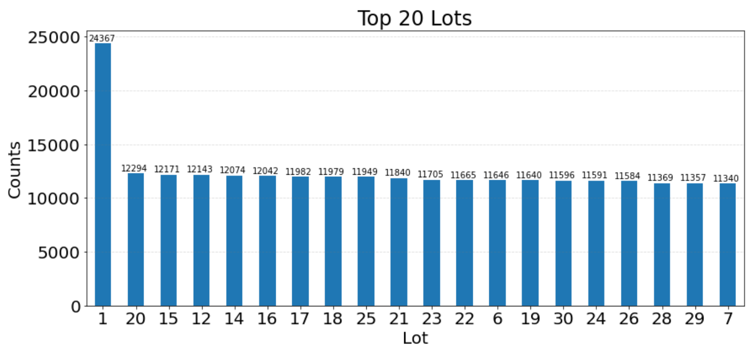


4) **Field Name: BLOCK**
Description: Valid block ranges by borough. The distribution shows the top 20 field values of blocks. The most common value is 3944, with a total count of 3,888.



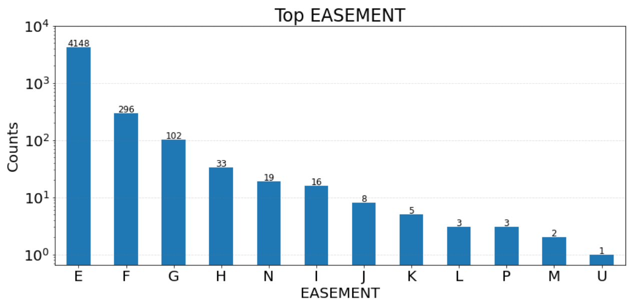
5) **Field Name: LOT**

Description: Lot by block. The distribution shows the top 20 field values of lot. The most common value is 1, with a total count of 24,367.



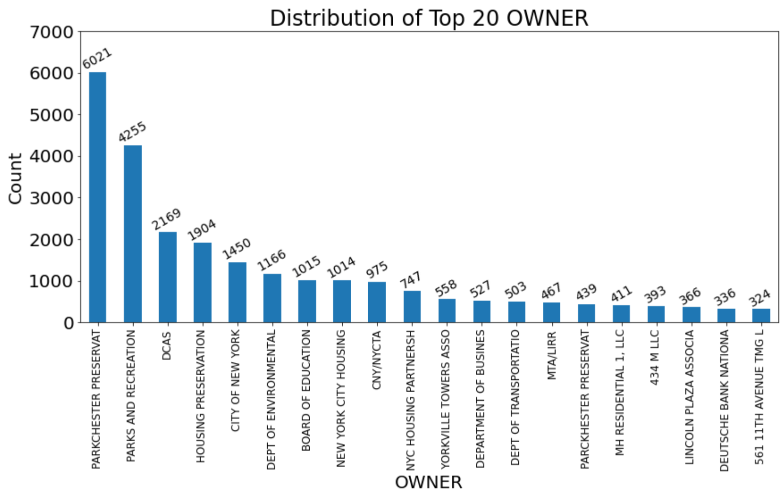
6) **Field Name: EASEMENT**

Description: Alphabet indicating easement. The distribution shows the field values of the easements. The most common value is 'E', with a total count of 4,148.



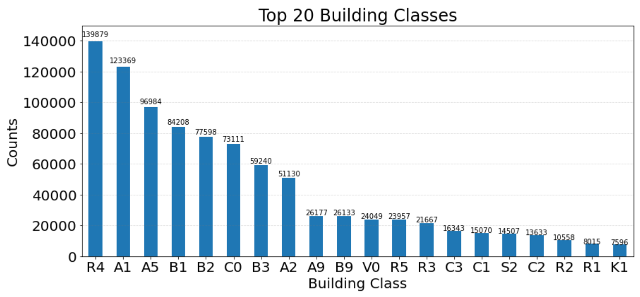
7) **Field Name: OWNER**

Description: Owner name of the property. The distribution shows the top 20 field values of owner name. The most common value is 'PARKCHESTER PRESERVAT', with a total count of 6,021.



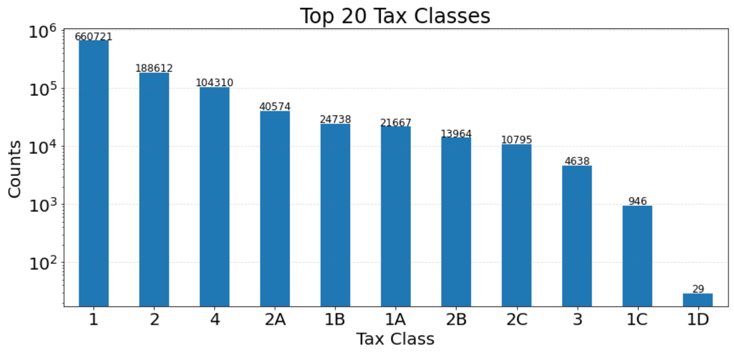
8) **Field Name: BLDGCL**

Description: Building Class. The distribution shows the top 20 field values of building classes. The most common value is 'R4', with a total count of 139,879.



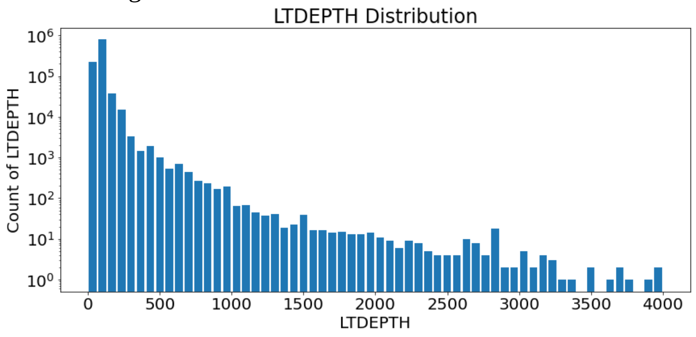
9) **Field Name: TAXCLASS**

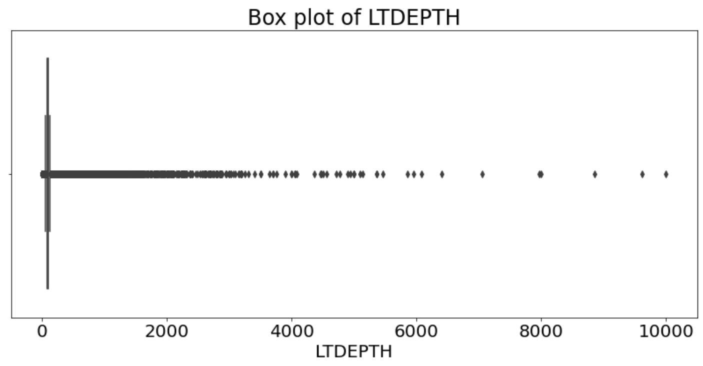
Description: Tax Class. The distribution shows the field values of the tax classes. The most common value is 1, with a total count of 660,721.

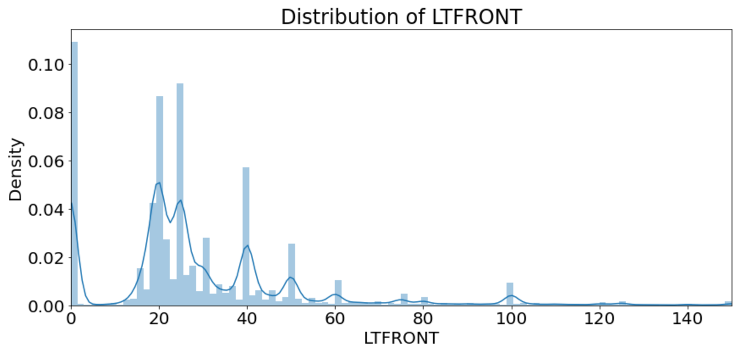


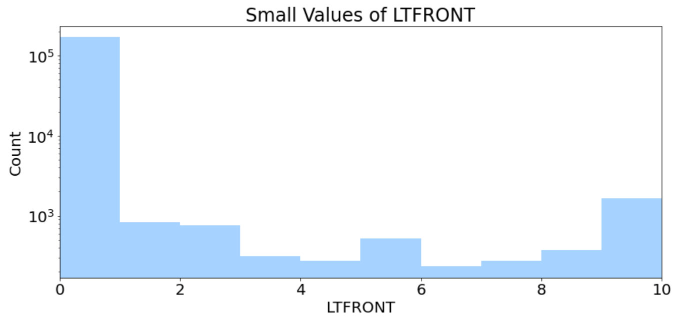
10) **Field Name: LTFRONT**

Description: Lot Width. The first distribution shows the histogram distribution of the lot widths. The second distribution shows the box plot of the overall distribution. The third distribution shows the most relevant range. The last distribution shows the distribution of the small numbers in log scale.



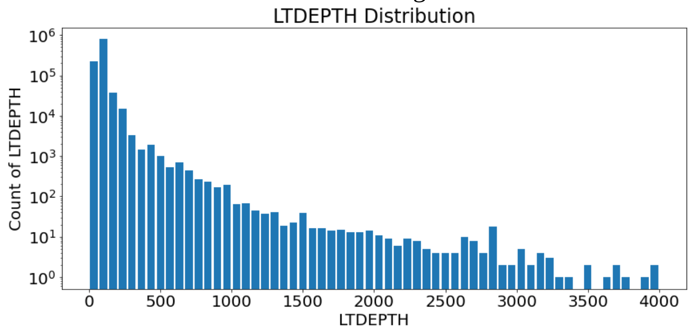


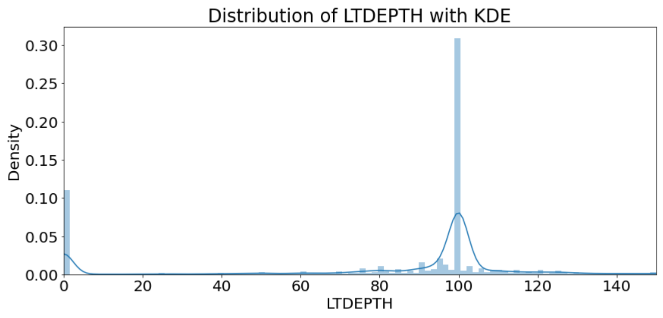


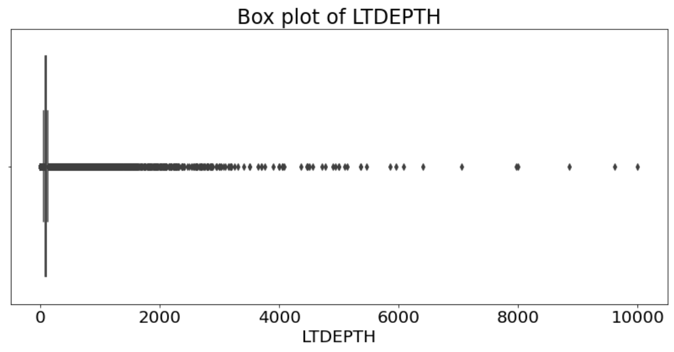


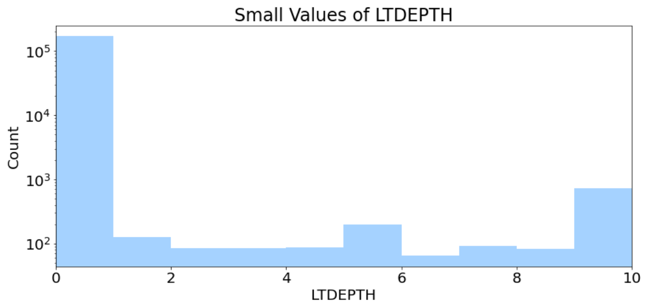
11) Field Name: LTDEP

Description: Lot Depth. The first plot shows the overall distribution of the lot depth. The second plot shows the box plot of the lot depth. The third distribution shows the most relevant range. The last plot shows the distribution of the small numbers in log scale.

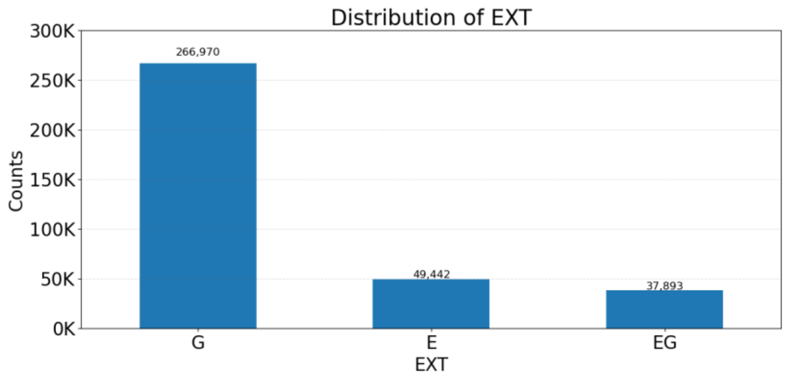




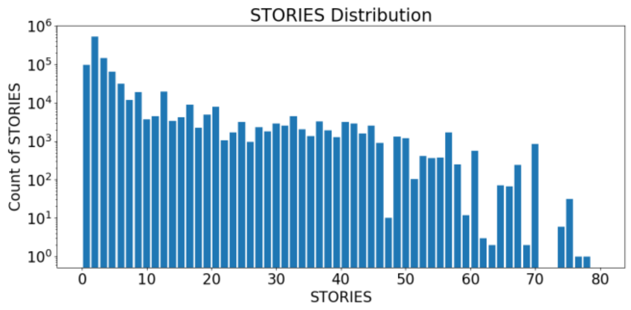


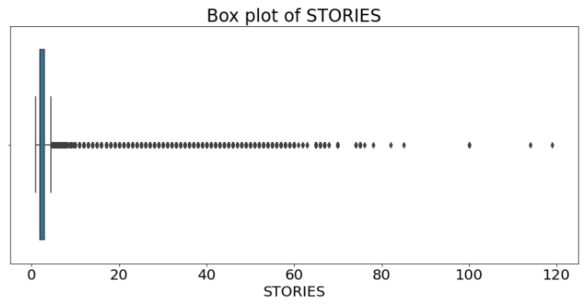


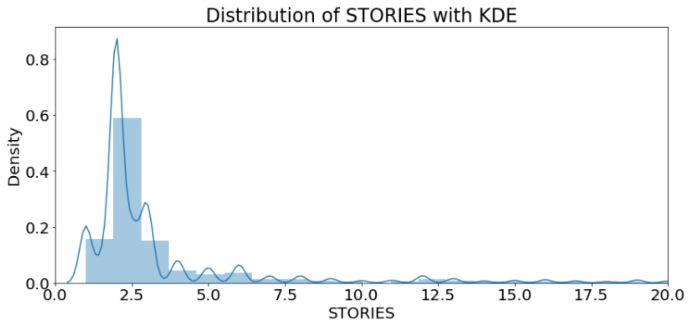
12) **Field Name: EXT**
Description: Extension Indicator. The total count of EXT = ‘G’ is 266,970, EXT = ‘E’ is 49,442, EXT = ‘EG’ is 37,893.



13) **Field Name: STORIES**
Description: The number of stories for the building. The first plot shows the overall distribution of the stories. The second plot shows the box plot of the stories. The third distribution shows the most relevant range.

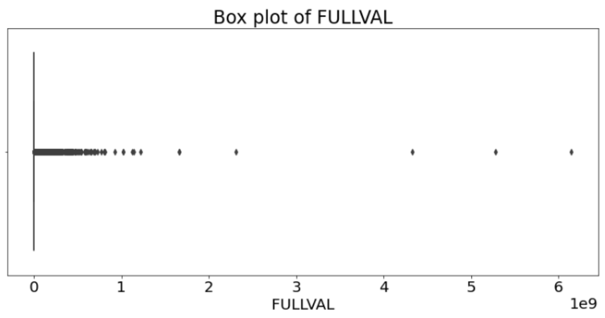


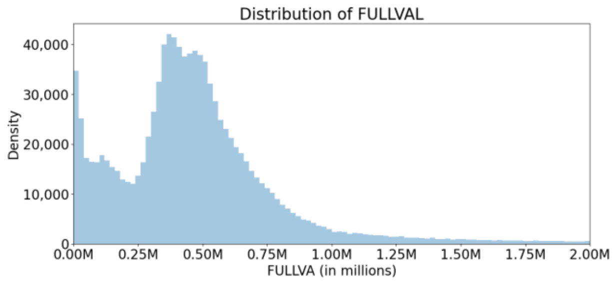




14) **Field Name: FULLVAL**

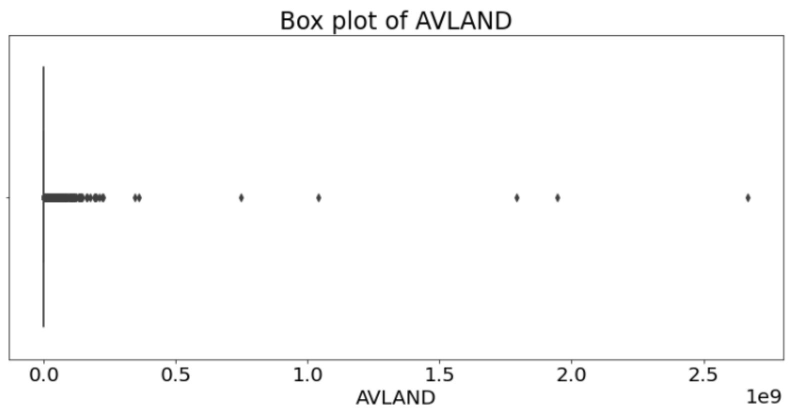
Description: Total market value. The first plot shows the box plot of the stories. The second plot shows the overall distribution of the market value.

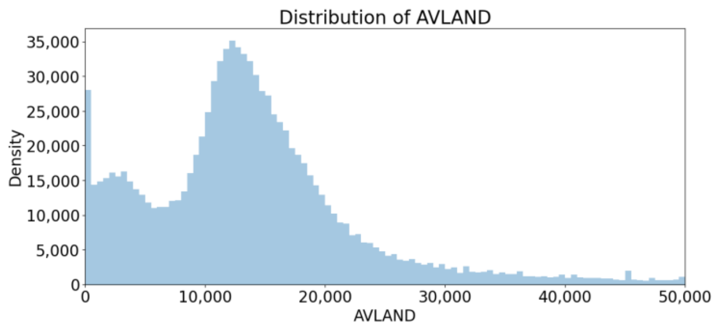




15) **Field Name: AVLAND**

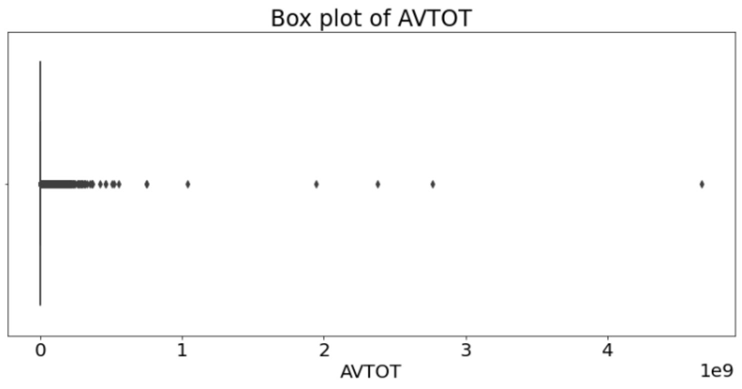
Description: Actual land value. The first plot shows the box plot of the actual land value. The second plot shows the overall distribution of the actual land value.

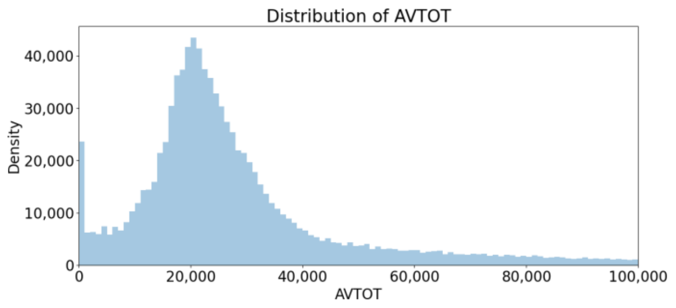




16) **Field Name: AVTOT**

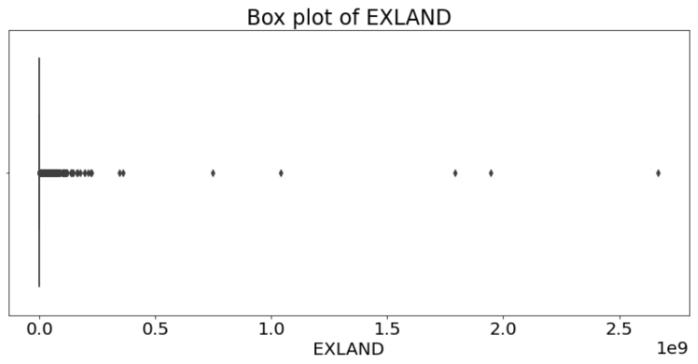
Description: Actual Total Value. The first plot shows the box plot of the actual total value. The second plot shows the overall distribution of the actual total value.

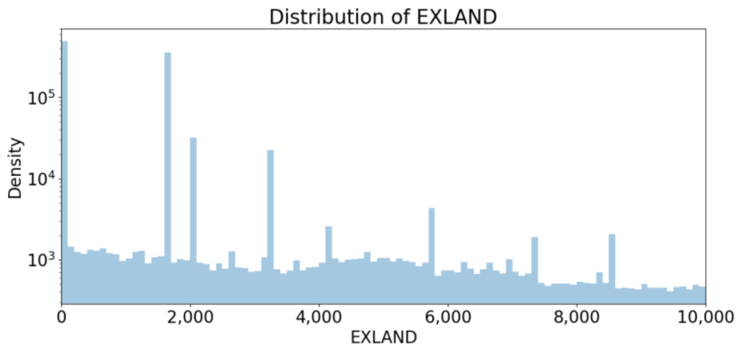




17) **Field Name: EXLAND**

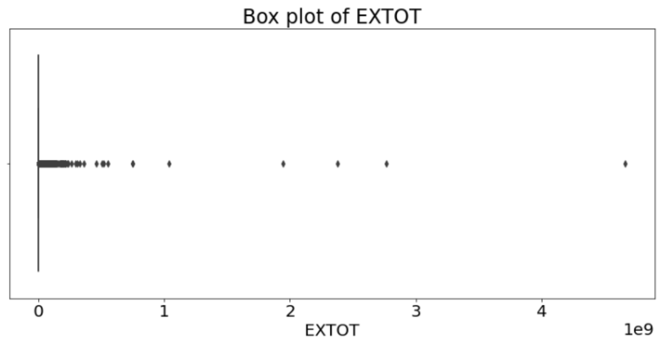
Description: Actual Exempt Land Value. The first plot shows the box plot of the actual exempt land value. The second plot shows the overall distribution of the actual exempt land value.

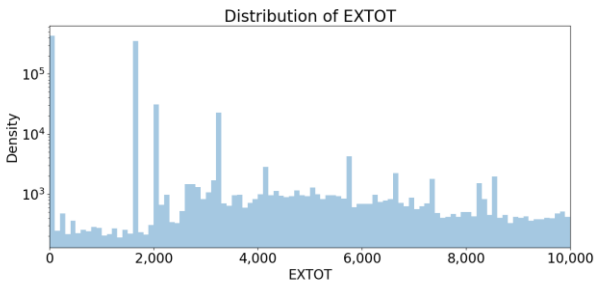




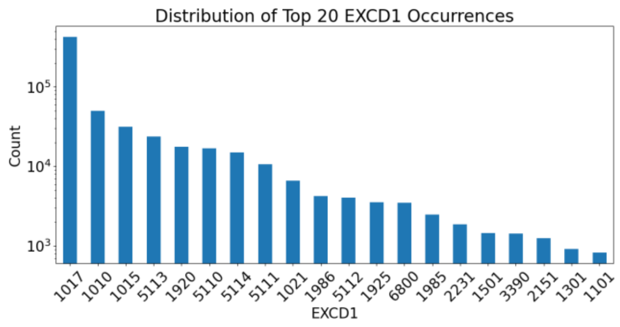
18) **Field Name: EXTOT**

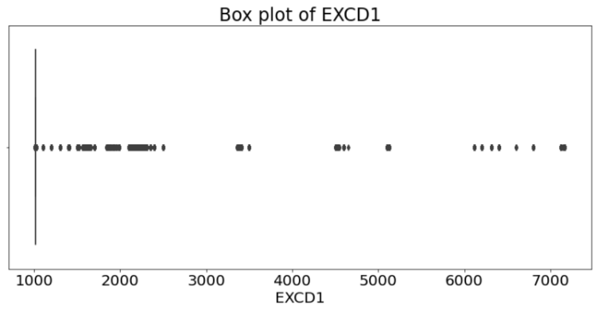
Description: Actual Exempt Land Total. The first plot shows the box plot of the actual exempt land total value. The second plot shows the overall distribution of the actual exempt land total value.



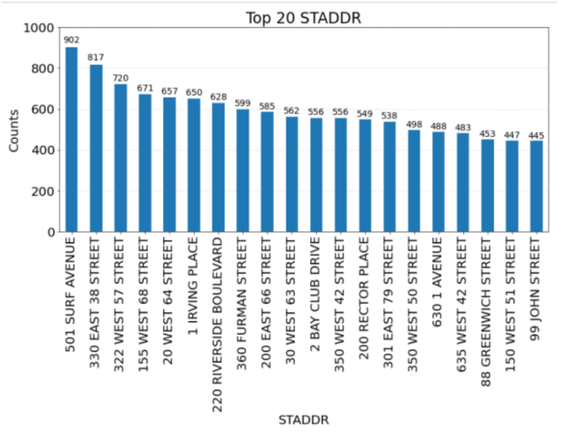


19) **Field Name: EXCD1**
Description: Exemption Code. The distribution plot shows the top 20 exemption codes. The most common code is 1017. The second plot shows the boxplot of the exempt code.

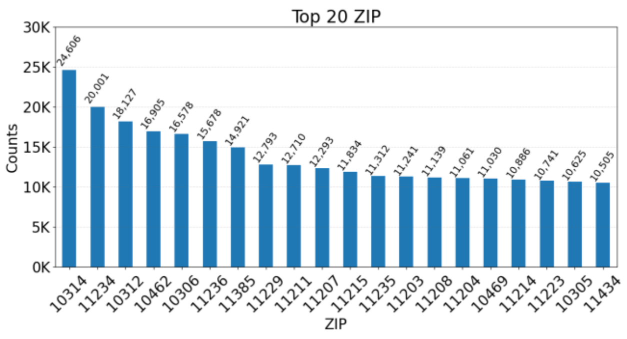




20) **Field Name: STADDR**
Description: Street Address. The plot shows the top 20 street address. The most common value is '501 SURF AVENUE', with a total count of 902.

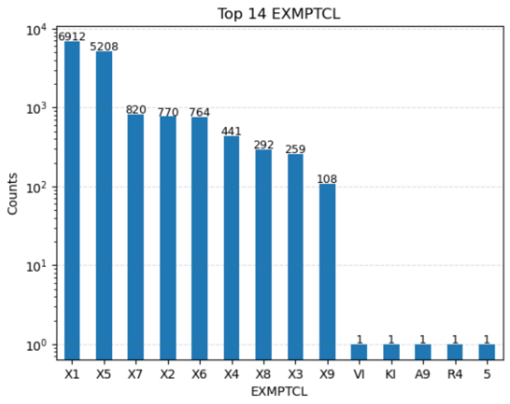


21) **Field Name: ZIP**
Description: Zip code. The distribution plot shows the top 20 zip code. The most common value is 10314, with a total count of 24,606.



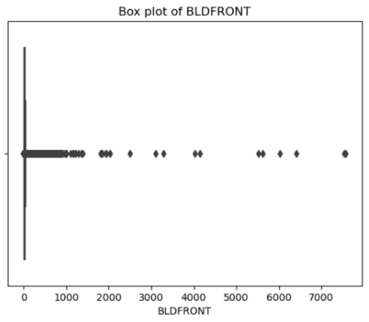
22) Field Name: EXMPTCL

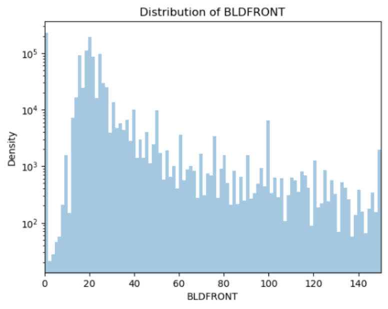
Description: Exempt Class. The distribution shows the top 14 field values of EXMPTCL. The most common number is X1, with a total count of 6,912.

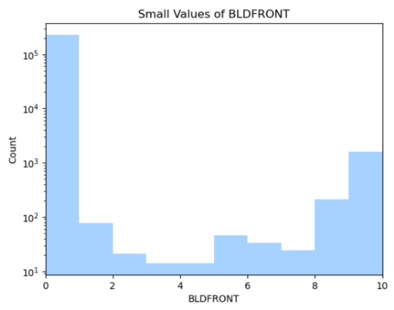


23) Field Name: BLDFRONT

Description: Building Frontage in feet. The first distribution shows the box plot indicating the overall distribution. The second plot shows the density distribution in log scale. The last graph shows the distribution of the small numbers in log scale.

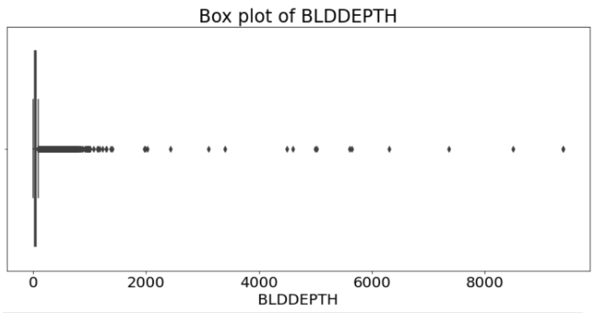


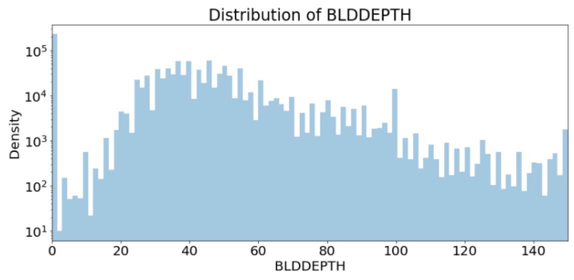


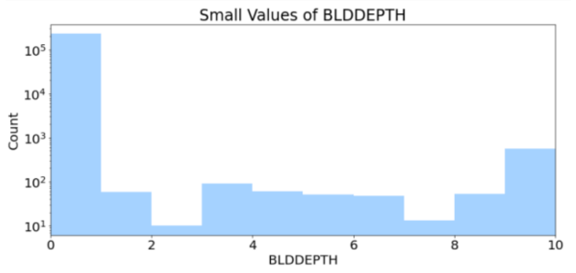


24) **Field Name: BLDDEPTH**

Description: Lot Depth in feet. The first distribution shows the box plot indicating the overall distribution. The second plot shows the density distribution in log scale. The last graph shows the distribution of the small numbers in log scale.

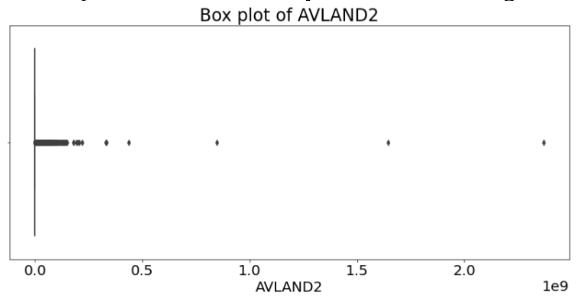


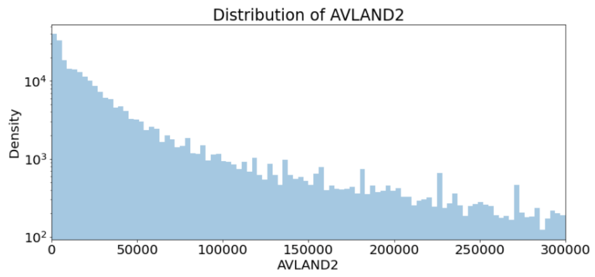




25) **Field Name: AVLAND2**

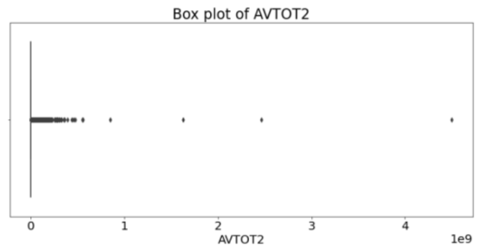
Description: Land value. The first distribution shows the box plot indicating the overall distribution. The second plot shows the density distribution in log scale.

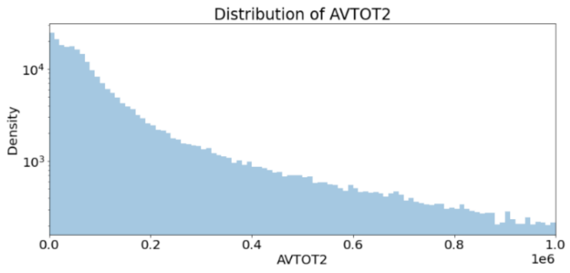




26) Field Name: AVTOT2

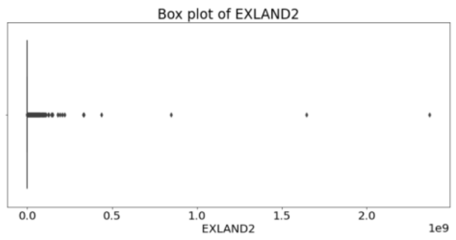
Description: Total value. The first distribution shows the box plot indicating the overall distribution. The second plot shows the density distribution in log scale.

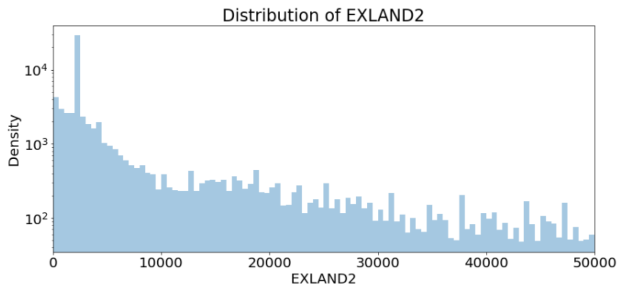




27) Field Name: EXLAND2

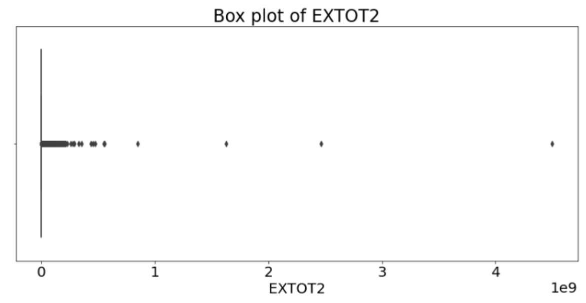
Description: Exemption land value. The first distribution shows the box plot indicating the overall distribution. The second plot shows the density distribution in log scale.

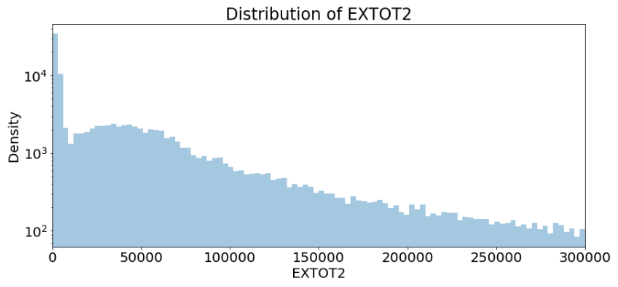




28) Field Name: EXTOT2

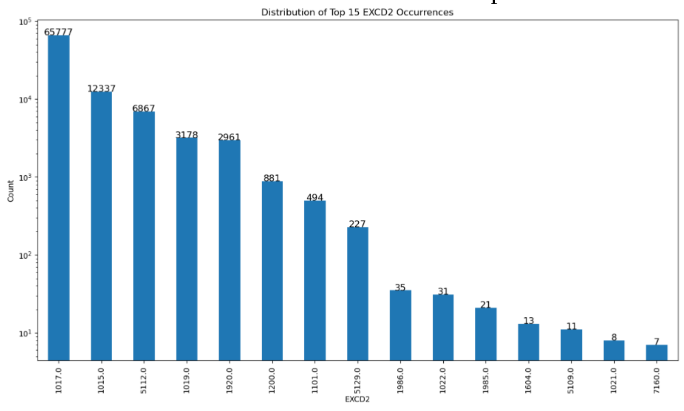
Description: Exemption land total. The first distribution shows the box plot indicating the overall distribution. The second plot shows the density distribution in log scale.

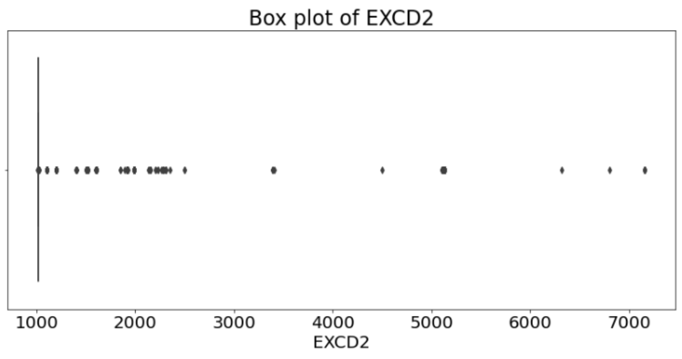


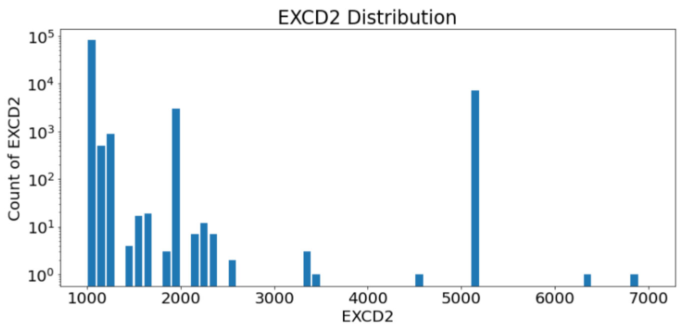


29) Field Name: EXCD2

Description: Exemption Code 2. The first distribution shows the top 15 field values of EXCD2. The most common number is 1017, with a total count of 65,777. The second figure shows the box plot indicating the overall distribution. The third distribution and plot show the histogram distribution in log scale.

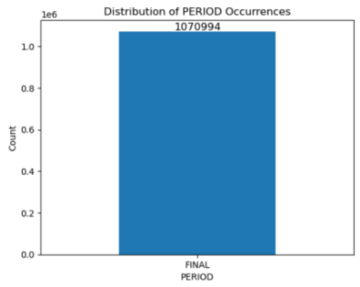






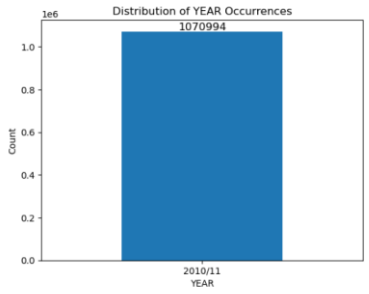
30) **Field Name: PERIOD**

Description: Assessment and evaluation period. The total count of PERIOD = ‘FINAL PERIOD’ is 1,070,994.



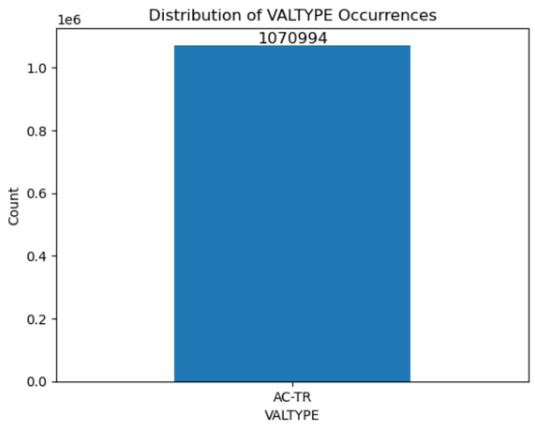
31) **Field Name: YEAR**

Description: Assessment and evaluation year. The total count of YEAR = ‘2010/11’ is 1,070,994.



32) **Field Name: VALTYPE**

Description: Value type. The total count of VALTYPE = ‘AC-TR VALTYPE’ is 1,070,994.


Distribution of VALTYPE Occurrences
1e6
Count
AC-TR
VALTYPE