Data Cleaning for NYC Property Data in Unsupervised Fraud Detection

Shreyash Kondakindi

September 21, 2025

1 Abstract

This report details the data cleaning steps applied to a New York City (NYC) property assessment dataset prior to performing an unsupervised fraud (anomaly) detection analysis. Key procedures included retaining extreme outlier values (to preserve potential anomalies), excluding certain records (notably government or institutional property owners unlikely to be fraud targets), and imputing missing values across nine important fields (full market value, assessed values, location, and structural characteristics). We describe the rationale and methods for each step, including grouping strategies for imputation designed to introduce only minimal distortion.

2 Methods

2.1 Outlier Retention

In contrast to many data preprocessing pipelines, we deliberately **retained outliers** in the data. The objective of the analysis was to identify properties with unusual valuations (potential fraud or assessment errors), which means the extreme values themselves are of primary interest. Removing or capping outliers would defeat the purpose, as it could eliminate the very cases we want the anomaly detection algorithm to find. Therefore, no filtering was applied to extreme values of financial fields (e.g., unusually high or low property values). By keeping these outliers, we ensured that genuinely aberrant valuations remain in the dataset to be detected by the unsupervised model.

2.2 Exclusions: Irrelevant Owners and Records

Some records were removed from the dataset because they do not represent the kind of properties relevant to fraud detection (for example, government-owned properties are not likely subjects of fraud and often have idiosyncratic valuations). We applied two exclusion rules:

- **Government easements:** Records where the *EASEMENT* code indicated a government property (code "U") were dropped. This affected only a negligible number of entries (1 record removed).
- **Owner name filter:** A list of owner name keywords was used to identify properties owned by government agencies, public authorities, or other entities outside our scope. The initial keyword list (`gov_list`) included terms such as *"DEPT"*, *"DEPARTMENT"*, *"UNITED STATES"*, *"GOVERNMENT"*, *" GOVT "*, and *"CEMETERY"*. Owner names containing any of these substrings

(and not containing the word "STORES", to avoid false matches like "Department Stores") were flagged. This automatically captured many city, state, and federal ownership entries (e.g., "POLICE DEPARTMENT", "CITY OF NEW YORK", "UNITED STATES GOVT").

The owner-based exclusion was refined iteratively. After the initial keyword filter, the set of flagged owner names was manually reviewed to ensure only true non-private owners were included. Additionally, the twenty most frequent owner names in the dataset were examined; many of these were large housing organizations or government-related (for example, *"NYC HOUSING PARTNERSHIP"*, *"NEW YORK CITY HOUSING"*, *"PORT AUTHORITY OF NY & NJ"*, *"MTA/LIRR"*, etc.). These high-frequency institutional owners were added to the removal list as well. In total, dozens of specific owner name patterns (covering various city agencies, state departments, public authorities, and similar bodies) were included in the final remove list. After applying all owner filters, any property whose *OWNER* field matched one of these patterns was removed from the dataset. This step eliminated a substantial number of records (26,501 records removed) corresponding to government or institutional properties that are not targets for fraud investigation. After these exclusions, the dataset was left with only privately owned properties of interest. Table 1 in the Results summarizes the number of records removed.

2.3 Missing Data Imputation

Nine fields were critical for our analysis and underwent missing data imputation: *FULLVAL* (full market value), *AVLAND* (assessed land value), *AVTOT* (assessed total value), *ZIP* (ZIP code), *STORIES* (number of stories), *LTFRONT* (lot frontage in feet), *LTDEPTH* (lot depth in feet), *BLDFRONT* (building frontage), and *BLDDEPTH* (building depth). In the raw data, missingness in these fields was indicated either by explicit empty/NA entries or by placeholder values (zeros, and in some cases ones) that are not plausible real values. We treated such placeholders as missing. For example, a lot or building dimension of 0 or 1 foot is not realistic and was interpreted as an absent measurement. Likewise, zero dollar values in valuation fields (where not conceptually valid) were considered missing data. Our imputation strategy was designed to be hierarchical: we imputed missing values in multiple passes, from more specific groupings of similar properties to more general groupings, stopping once a value was filled. The general principle was to use property characteristics (such as tax class, building category, and location) to compute typical values from comparable properties, thereby inserting a value that is "innocuous" (i.e., typical for that kind of property) rather than an arbitrary global default. This way, we minimize distortion to the data distribution and avoid artificially creating outliers. The specific approaches for different field categories are detailed below.

2.3.1 Valuation Fields (FULLVAL, AVLAND, AVTOT)

The full market value and assessed values had a small percentage of missing entries (where "missing" includes zeros). For **FULLVAL**, there were 10,025 cases of missing or zero. **AVLAND** had 10,027 and **AVTOT** had 10,025 such cases initially. We performed a three-tier imputation:

1. **By Tax Class + Borough + Building Class:** We grouped properties by their tax class, borough, and building class (a granular code for property type) and computed the median value within each group. The intuition is that properties of the same type in the same borough and tax category should have similar valuations. Missing values were first filled with the median of their $(\text{TaxClass}, \text{Borough}, \text{BldgClass})$ group. This first pass filled a portion of the missing values (e.g., for FULLVAL about 2,718 records were imputed in this step), but some remained unfilled if an entire group had been missing the value.

2. **By Tax Class + Borough:** For cases still missing after the first pass, we used a broader grouping. We grouped by tax class and borough only (ignoring building class) and filled remaining NAs with the

median value for that (TaxClass, Borough) combination. This leverages a larger pool of properties (all property types in the borough within the same tax class) to provide a typical value. This second pass imputed the majority of the remaining gaps (around 6,921 additional FULLVAL records).

3. **By Tax Class (overall):** Finally, any still-missing values were filled by using the median within just the tax class (across all boroughs). At this highest level, only very few records remained to be imputed (for FULLVAL, 386 cases were filled in this final step). After this, no FULLVAL entries remained missing. A similar pattern and count occurred for AVLAND and AVTOT, as these three fields tend to be missing together.

By progressively widening the grouping criteria, we ensured that wherever possible a property's missing valuation was filled in with a value characteristic of very similar properties; only if that failed did we resort to a broader average. This hierarchy aims to insert values that are as "normal" as possible for that property's profile. After all three steps, FULLVAL, AVLAND, and AVTOT had 0 missing entries remaining.

2.3.2 ZIP Code

ZIP codes were occasionally missing from the address data (20,431 properties initially lacked ZIP). We applied a two-step approach to infer missing ZIP codes:

1. **By Address lookup:** We created a composite key of street address and borough for each property and built a mapping from this key to ZIP code using all properties that had a known ZIP. Many NYC buildings have multiple units or lots associated with the same street address. Using this mapping, we filled in ZIP codes for properties whose address (street number/name and borough) matched an address where the ZIP was known. This recovered a significant number of ZIPs (approximately 2,832 filled).

2. **By neighbor inference:** For the remaining missing ZIP entries, we leveraged the sorted order of the dataset (essentially grouping nearby properties) to propagate ZIP codes where appropriate. We performed a forward-backward fill: for each run of missing ZIPs sandwiched between known ZIP values in the data, if the ZIP code above and below the gap were the same, we filled the entire gap with that ZIP (assuming the neighboring properties share the ZIP). This method effectively assigns a ZIP when all nearby context points to a consistent value. It accounted for the bulk of the remaining missing ZIPs (about 17,599 cases filled). For the handful of properties still without ZIP after this (cases at the very beginning or end of a borough's list, or isolated instances where neighbors had conflicting ZIPs), we applied a final pass using forward-fill/back-fill to ensure no ZIP was left blank.

After these two steps, every property had a ZIP code. The imputation here is innocuous in that it uses geographical or address continuity to guess the ZIP—essentially what one would do manually if addresses were known. By using actual addresses and neighbors, we avoid inventing implausible postal codes.

2.3.3 Building Stories

The number of stories (**STORIES**) was missing for some properties, especially land-only parcels or incomplete records (42,029 missing). Because number of stories is a discrete attribute that often takes a few common values (e.g., many residential homes have 2 stories, many apartment buildings might have 6, etc.), we based our fill on the mode (most frequent value) within property groupings:

1. **By Borough + Building Class:** Within each borough and building class category, we found the most common story count. We then filled missing STORIES with that modal value for the corresponding (Borough, BldgClass) group. For example, if most townhouse-style buildings (particular building class code) in Brooklyn are 2 stories, a Brooklyn property of that class with missing story count would be filled with 2. This step resolved about 4,108 missing entries.

2. **By Tax Class:** Some properties still had no story information after the first pass (in cases where an entire building class in a borough had no story data, or very unique property types). For these, we fell back on the broader tax class. We took the average (mean) number of stories within each tax class (class 1: one-family homes, class 2: multi-family, class 3: utilities, class 4: commercial, etc.) and used that to fill any remaining NAs. This filled the remaining 37,921 cases. After this second step, all properties had a story count imputed. (Using the mean at the tax class level can result in non-integer story counts, but since the number is only used for anomaly detection scaling and not as an exact count, this is acceptable.)

The grouping by building class captured the typical structure for similar buildings, while the fallback to tax class provided a reasonable generic estimate where needed. The result is that every property has a plausible story count, and the imputed values are unremarkable (often common low-story numbers for residential classes, etc.), ensuring that we did not introduce any outlandishly tall or short buildings via imputation.

2.3.4 Lot and Building Dimensions

The physical size fields **LTFRONT**, **LTDEPTH** (lot frontage and depth) and **BLDFRONT**, **BLDDEPTH** (building frontage and depth) had many missing or placeholder values initially. Zeros in these fields indicate not recorded – for instance, a condo unit might not have its own lot dimensions recorded, or a vacant lot might have no building to measure. We treated both 0 and 1 as missing, since a one-foot measurement is likely a dummy value. After this treatment, 161,133 records had missing lot frontage (LTFRONT), a similar number for lot depth, and a smaller number for building dimensions (building depth had 58 missing after zero/one replacement, and building frontage had a comparable order of magnitude). For lot size dimensions, we chose a relatively broad imputation grouping because lot sizes can vary widely even within a building class, but tend to be more consistent within broad property classes and locations. We grouped by **Tax Class and Borough** and computed the mean lot frontage and depth for each such group. Missing lot frontage and depth were then filled with the average for that class of property in that borough. This one-pass imputation filled essentially all missing lot size entries except for a tiny remainder. In fact, only 2 properties could not be imputed in this way for LTFRONT (and similarly for LTDEPTH) because they belonged to a category where no other comparable property had dimensions recorded (for example, an unusual property type with all instances missing). Those represented an exceedingly small fraction ($<0.0002\%$ of the data) and were left as missing since any guess would be arbitrary. (These cases were identified as peculiar vacant lots with no building and unique characteristics.) For building dimensions, missingness was much less prevalent. We imputed building frontage and depth using a more fine-grained grouping where possible, since building size correlates with building class. We first grouped by **Tax Class, Borough, and Building Class** and filled missing *BLDFRONT* and *BLDDEPTH* with the mean within each (TaxClass, Borough, BldgClass) group. Because the vast majority of buildings in a given class have their dimensions recorded, this step was able to fill nearly all gaps. After imputation, no building frontage or depth values remained missing in the dataset. The very few cases of properties with no building (for which building dimensions are conceptually not applicable) had been marked as missing; for analysis purposes we did not consider those as requiring imputation – in practice, those entries can be treated as having zero building size. In summary, by the end of cleaning all four dimension fields had no meaningful missing values impacting the analysis. This treatment of dimensions ensures that every property has reasonable lot and building size attributes for use in modeling. The use of averages within broad groupings (tax class, borough, and sometimes building class) provides typical sizes that are unremarkable for those types of properties. We especially note that setting 0/1 to missing and then imputing prevents those placeholder values from skewing any analysis – a 0 lot size or 1-foot building depth would have been extreme outliers if left as is, but after imputation each property has dimensions in a normal range for its category.

3 Results

Table 1 summarizes the extent of missing data in each of the nine key fields and how the imputation procedure addressed them. The table lists the total number of records that were initially missing (or had an invalid placeholder) for each field, the number of records imputed at each stage of the hierarchical filling process, and the number remaining missing at the end (if any).

Table 1: Missing Data Imputation Summary for Key Fields

Field	**Initial Missing**	**Filled Step 1**	**Filled Step 2**	**Filled Step 3**	**Final Missing**
FULLVAL	10,025	2,718	6,921	386	0
AVLAND	10,027	2,720	6,921	386	0
AVTOT	10,025	2,718	6,921	386	0
ZIP	20,431	2,832	17,599	–	0
STORIES	42,029	4,108	37,921	–	0
LTFRONT	161,133	161,131	–	–	2
LTDEPTH	161,133	161,131	–	–	2
BLDFRONT	∼58	∼58	–	–	0
BLDDEPTH	58	58	–	–	0

Notes: "Initial Missing" includes all records with NA, zero, or other placeholder values treated as missing. "Filled Step 1" refers to the first imputation pass (most specific grouping for that field: e.g., TaxClass+Borough+BldgClass or using address mapping for ZIP). "Filled Step 2" refers to the second pass (broader grouping or neighbor-based fill). "Filled Step 3" indicates the third pass (if applicable; for ZIP and STORIES only two steps were used, denoted by "–"). "Final Missing" are records still lacking a value after all imputation steps. For BLDFRONT, the initial missing count was very low (only on the order of tens; denoted ∼58 to indicate the same order as BLDDEPTH) and all were filled in one pass. As shown in the table, the imputation strategy was able to fill in essentially all missing values. The valuation fields and ZIP code achieved 100% completion. The stories field likewise had no missing values after the two-step fill. Lot dimensions had only 2 properties still missing (these were exceptional cases of unique vacant lots as noted). Building dimensions were fully populated after imputation (any originally missing building measurements were resolved, or considered effectively not applicable). In terms of data exclusions, the filtering removed a total of 26,502 records (1 due to easement code "U" and 26,501 due to the owner-name criteria) out of about 1,070,994 initial records, leaving roughly 1,044,492 records for analysis. This reduction primarily targeted properties unlikely to yield actionable fraud insights.

4 Discussion

The data cleaning process balanced completeness with the need to preserve genuine anomalies. By not removing outliers in the data, we ensure that extreme but potentially significant cases are available for the anomaly detection algorithm. At the same time, by excluding obvious non-fraud-related properties (like those owned by government entities), we focused the analysis on the domain of interest (private properties where irregular valuations might indicate fraud or error) and eliminated a source of systematic value differences that could confuse unsupervised detection. The missing data imputation was carefully designed to avoid introducing bias. The hierarchical scheme fills missing values with typical group medians or means, which means an imputed entry will look "normal" compared to its peers. This is crucial for unsupervised outlier detection: we do not want an imputed value itself to stand out as anomalous, nor do we want to overly homogenize data in a way that obscures real outliers. By using fine-grained groups (same property type and location) whenever possible, the imputed values are as close as possible to what that property's value would realistically be. Only when that fails do we widen the lens. For example, filling valuations by tax class citywide (the coarsest step) affected only a few hundred records and likely assigned them a very generic value (e.g., a mid-range market value for a given tax class) – those records were rare cases where no similar property had a recorded value, so this approach still should not create spurious anomalies. In the case of ZIP codes, using address-based and neighbor-based filling leverages the inherent structure of addresses and geography in the data; this method essentially recovers information that likely was known but unrecorded, with minimal risk of error (especially given that any remaining ambiguities were resolved by the final fill ensuring consistency). The lot and building size fields illustrate the principle of treating obviously invalid values (0 or 1) as missing and then using sensible averages to supply a value. For a downstream anomaly detection task, a vacant lot with no building should not have a "0" building size left in the data (as zero would be an extreme outlier in a distribution of building sizes); by marking it missing and filling it with, say, an average small building footprint for that class, that property will not be falsely flagged just for having an impossible zero entry, nor will it hide among outliers since a vacant lot might still be flagged by other features (e.g., value ratios). The two properties that remained with missing lot dimensions were essentially edge cases that could either be left as is (since two out of over a million records will not impact a robust PCA-based anomaly detection and could be manually examined if they turn up as outliers) or could be imputed with a very generic value without affecting results. In practice, these had no building and unique lots, so their missing lot size could be interpreted as "not applicable." These cases highlight that our strategy favored not forcing an imputation if it would be purely guesswork; leaving a couple of data points incomplete (or treating them separately) is preferable to injecting potentially misleading data. Overall, the cleaned dataset is complete (virtually no missing values aside from negligible exceptions) and contains only the relevant properties, with all original extreme values intact. This provides a solid foundation for the anomaly detection algorithm to run effectively, detecting unusual property valuations without interference from irrelevant data issues or undue bias from imputation.

Appendix: Code Excerpts

Owner Removal List Construction (Python snippet):

```
gov_list = ['DEPT ', 'DEPARTMENT', 'UNITED STATES', 'GOVERNMENT', ' GOVT ', 'CEMETERY']
remove_list = []
for owner_name in data['OWNER'].dropna().unique():
for key in gov_list:
if key in owner_name and 'STORES' not in owner_name:
remove_list.append(owner_name)
Manual additions based on inspection
additional_names = [
'THE CITY OF NEW YORK', 'CITY OF NEW YORK', 'NYS URBAN DEVELOPMENT',
'CULTURAL AFFAIRS', 'NY STATE PUBLIC WORKS', "NYC DEP'T OF HIGHWAYS",
'CITY WIDE ADMINISTRAT', 'PORT AUTHORITY', 'MTA', 'NYCTA',
'BOARD OF EDUCATION', 'NYC HOUSING', 'HOUSING PRESERVATION',
'DEPT OF TRANSPORTATION', 'NYC PARK DEPT', 'STATE OF NEW YORK',
'SANITATION', 'NYC TRANSIT', 'LIRR', 'DEUTSCHE BANK', 'LLC'
]
for name in additional_names:
remove_list.append(name)
remove_list = set(remove_list) # unique
Filter out records whose OWNER is in remove_list
data = data[~data['OWNER'].isin(remove_list)]
```

The above pseudocode compiles the keywords and specific names used to filter out owners. Note that actual code ensured no accidental removal of names containing those substrings in other contexts (e.g., excluding "STORES"). *ZIP Code Imputation via Neighbor Fill (Python snippet):*

```
Fill ZIP by forward/backward propagation where neighbors agree
zip_forward = data['ZIP'].fillna(method='ffill')
zip_backward = data['ZIP'].fillna(method='bfill')
mask = data['ZIP'].isna() & (zip_forward == zip_backward)
data.loc[mask, 'ZIP'] = zip_forward[mask]
```

This code uses pandas forward-fill and backward-fill to interpolate missing ZIPs. Only where the forward-filled and backward-filled values coincide (meaning the missing segment is bounded by identical ZIP codes) do we assign that ZIP to the missing entries.