NYC Property Tax Fraud Detection
Feature Engineering and Unsupervised Modeling

Shreyash Kondakindi

September 21, 2025

1 Motivation for Variable Construction and Anomaly Detection

The variables chosen for analysis are designed to capture unusual patterns in property valuations that might indicate fraud or errors. In NYC, potential fraud scenarios include **undervalued properties** (deliberately low assessed values relative to market), **overvalued or misclassified properties** (errors or inconsistencies in valuation), and **atypical property characteristics** that don't match their tax assessments. To detect such anomalies, we construct features that reflect value *per unit of size* (land or building), compare properties to their peers, and highlight extreme deviations on either end of the spectrum (very high or very low ratios). This approach is grounded in real estate domain knowledge: for example, **market value per square foot** of land or building is a key indicator of whether a property's valuation is in line with expectations. Unusual values per square foot – either extremely high or extremely low – could signal a fraudulent assessment or data anomaly. Likewise, value per unit of building volume (accounting for building height) is important to fairly compare high-rise buildings to low-rise ones. All such measures are further **contextualized by location and property type** (e.g. ZIP code area or tax class) to distinguish genuine anomalies from normal neighborhood variations.

Each variable in our model has a clear motivation tied to these scenarios:

- **Lot size and Building size variables** (area and volume) establish the property's scale. Large discrepancies between a property's size and its reported value can indicate irregularities. For instance, a very large lot with a surprisingly low assessed value may suggest undervaluation, whereas a tiny property with an outsized market value may indicate an input error or exceptional case. We explicitly compute lot area and building area from dimensions to use standardized size measures across all properties.

- **Value-per-size ratios** (value per square foot of land or building, etc.) are direct indicators of valuation anomalies. A property's **market value per land square foot** (FULLVAL/lotarea) that is far below the city norm could mean the property is being undertaxed or that its market value is outdated; conversely, an extremely high value per sq. ft might indicate an erroneously high valuation or perhaps a luxury development. Similarly, **assessed value per sq. ft** of land (AVLAND/lotarea) reveals if the land assessment is out of line (too low might suggest fraud, too high could be error). Using **building footprint area** and **building volume**, we capture improvements: e.g. **market value per building square foot** (FULLVAL/bldarea) or **per cubic foot of building** (FULLVAL/bldvol) highlight if a structure's value is inexplicably low (possibly under-assessed improvements) or overly high (perhaps an inflated sale). **Building volume (footprint $\times$ height)** is especially important for high-rises so that tall buildings are judged by total usable space, not just footprint. In short,

these ratios translate raw values into comparable metrics that flag properties valued unusually given their size.

- **Reciprocal value ratios** are included to detect anomalies on the low end of valuations. Typically, using the ratio (e.g. value/area) will catch properties with abnormally *high* value for their size (high ratio). However, a property with abnormally low value for its size (e.g. a huge lot with a small valuation) might not appear as an outlier in the ratio itself. By also considering the **inverse ratio** (e.g. area/value), we capture those cases. In practice, for each ratio r, we take $\max(r,\ 1/r)$ as the feature value so that whether a property's value is too high or too low for its size, it will show up as an extreme value in one of the two forms. This ensures that **undervalued properties (low r)** are as detectable as **overvalued ones (high r)**. For example, if land value per sq. ft is extremely low, then land area per dollar (the inverse) will be extremely high, flagging the record.

- **Peer-comparison features** account for the fact that "unusual" must be defined relative to similar properties. NYC has diverse neighborhoods and property classes, so a normal land value per sqft in Manhattan will be much higher than a normal value in Staten Island. We therefore include features that compare each property's value ratios to the **typical values in its ZIP code and its tax class**. Tax classes (e.g. 1 for one-family homes, 2 for multi-family, 4 for commercial) group properties of similar use; similarly, ZIP codes proxy location value differences. If a property's value per sqft is, say, half of the average for its area or class, that relative measure could indicate undervaluation in context. Likewise, a property valued double its neighbors per sqft might be an outlier worth investigating. These **normalized ratio features** help isolate context-specific anomalies that absolute ratios alone might miss. By benchmarking against local averages, we reduce false flags due to perfectly legitimate neighborhood effects.

- **Assessment consistency features** directly compare the different dollar value measures for each property. NYC property data provides Full Market Value (FULLVAL) and two assessed values: AVLAND (assessed land) and AVTOT (assessed total). These should be internally consistent (e.g. AVTOT $\approx$ AVLAND + assessed building value, and AVTOT should be a fraction of FULLVAL according to assessment rules). We construct a feature to capture inconsistencies: for example, the ratio of assessed total to market value, AVTOT/FULLVAL (or its inverse). In NYC, assessed values are typically capped at a percentage of market value by law, so an unusually low AVTOT/FULLVAL ratio means the property's market value is high relative to its taxable assessment – possibly a sign of *undertaxation* or a legal loophole being exploited. Conversely, a high ratio (approaching 1) would mean a property's assessed value is nearly equal to market (which is rare under normal rules and could indicate an error in data). In our analysis we include an **assessment-to-market value ratio** to flag properties that deviate from expected assessment practices (e.g. a property that somehow is assessed at only 5% of its market value versus the usual 20% for its class would stand out).

- **Building-to-land size ratio** is another intuitive check: it compares how much of the lot is built up. We define a feature like **lot coverage** = bldarea/lotarea (the fraction of lot area covered by the building). This can reveal anomalies such as a lot with virtually no building (tiny structure on a large lot) or, oppositely, a building footprint larger than the lot (which might indicate data errors or lots combined in reality). A very low coverage (far below typical for that property type) might correlate with an undervalued property (e.g. a supposedly "vacant" lot with minimal improvements – could either be a legitimate empty lot or an indication that a structure wasn't accounted for in valuation). Extremely high coverage (near 100% or beyond) could flag unusual development density or recording mistakes. In NYC, most residential lots have a reasonable building-to-lot ratio given zoning; anything far outside the norm could be worth a closer look. This feature thus helps detect *physical inconsistencies* that could tie back to fraud (like unpermitted buildings or misreported lot size).

In summary, the NY property variables have been crafted to expose the kinds of irregularities one would expect in tax fraud or errors: atypical value per area metrics, contextually strange valuations, and internal inconsistencies in the data. By covering both high and low extremes and comparing each property to appropriate benchmarks, these features collectively span a wide range of "unusual" conditions we aim to find.

2 Data Preparation and Variables Description

Before modeling, we performed extensive data cleaning and feature engineering to create the final set of 29 variables for analysis. This section describes each variable (including how it was constructed or transformed) and the preprocessing steps (missing value imputation and data filtering) that ensured the data was ready for anomaly detection.

2.1 Constructed Features and Transformations

Our starting point was the raw NYC property data, which includes descriptors like lot dimensions, building dimensions, assessed values, etc. From these, we derived **size features** and **value ratio features** as outlined below:

- **Lot area ("lotarea")**: The land area of the property, computed as LTFRONT × LTDEPTH. LTFRONT and LTDEPTH are the recorded lot frontage and lot depth (in feet). This yields the lot's square footage. Lotarea provides a scale for the land; it's crucial for normalizing values by parcel size.
- **Building area ("bldarea")**: The building's footprint area, computed as BLDFRONT×BLDDEPTH. These are the building's front and depth measurements. Bldarea (in square feet) represents how much ground area the building covers.
- **Building volume ("bldvol")**: An approximate volume or total floor area of the building, calculated as bldarea × STORIES. Essentially, we extend the footprint by the number of stories to estimate total built space. Bldvol is a proxy for the building's overall size (e.g. a 2-story building with the same footprint as a 10-story building will have a much smaller bldvol). This feature helps compare multi-story buildings' value on a fair basis.

Using the three "size" variables above (lotarea S_1, bldarea S_2, bldvol S_3) and the three "value" variables from the dataset (FULLVAL V_1 = full market value, AVLAND V_2 = assessed land value, AVTOT V_3 = assessed total value), we constructed **nine ratio variables**. Each ratio is a value divided by a size, giving a per-unit-value measure:

$$r_{1,1} = \frac{\text{FULLVAL}}{\text{lotarea}} \quad \text{(Market value per square foot of land)}$$
$$r_{1,2} = \frac{\text{FULLVAL}}{\text{bldarea}} \quad \text{(Market value per square foot of building footprint)}$$
$$r_{1,3} = \frac{\text{FULLVAL}}{\text{bldvol}} \quad \text{(Market value per unit of building volume)}$$
$$r_{2,1} = \frac{\text{AVLAND}}{\text{lotarea}} \quad \text{(Assessed land value per sq. ft of land)}$$
$$r_{2,2} = \frac{\text{AVLAND}}{\text{bldarea}} \quad \text{(Assessed land value per sq. ft of building footprint)}$$
$$r_{2,3} = \frac{\text{AVLAND}}{\text{bldvol}}$$
$$r_{3,1} = \frac{\text{AVTOT}}{\text{lotarea}} \quad \text{(Assessed total value per sq. ft of land)}$$
$$r_{3,2} = \frac{\text{AVTOT}}{\text{bldarea}}$$
$$r_{3,3} = \frac{\text{AVTOT}}{\text{bldvol}}$$

These 9 features encapsulate the core value-density metrics for each property. As discussed, extremely large values of these ratios indicate properties that are valued very highly for their size (potential over-valuation or luxury outliers), whereas extremely small values indicate properties valued very cheaply for their size (potential undervaluation or neglected assessments).

Next, to capture both extremes without doubling the number of features, we applied the **reciprocal transformation** on each ratio. For each ratio $r_{i,j}$, we also considered $r_{i,j}^{-1} = \frac{1}{r_{i,j}}$, and then for our feature we keep the larger of r or $1/r$. In practice this means defining a new variable:

$$R_{i,j} = \max(r_{i,j},\ 1/r_{i,j})$$

If $r_{i,j} \geq 1$, then $R_{i,j} = r_{i,j}$; if $r_{i,j} < 1$, then $R_{i,j} = 1/r_{i,j}$. This ensures $R_{i,j} \geq 1$ always, and values much greater than 1 indicate an anomaly in either direction. By doing this, we do not need separate features for low-end outliers – each $R_{i,j}$ flags any strong deviation from the typical value (either too high or too low). This step is motivated by the need to catch undervalued properties which would otherwise hide in small ratios.

After creating the 9 base ratios (and implicitly transforming them via the max-of-inverse approach), we incorporated **group averages for ZIP code and tax class**. For each ratio variable $r_{i,j}$, we computed two reference values: the average of that ratio among all properties in the same ZIP code, and the average among all properties in the same tax class. Call these $\overline{r}_{i,j}^{\text{ZIP}}$ and $\overline{r}_{i,j}^{\text{TAX}}$ for a given property's ZIP and tax class. These capture the "typical" value density for the property's local geographic area and its broad property type. We then formed **normalized ratio features** by dividing each property's ratio by these group averages:

$$r\_ZIP_{i,j} = \frac{r_{i,j}}{\overline{r}_{i,j}^{\text{ZIP}}}$$
$$r\_TAX_{i,j} = \frac{r_{i,j}}{\overline{r}_{i,j}^{\text{TAX}}}$$

If a property's ratio is exactly at the neighborhood average, these normalized features will be 1.0. Values above 1 mean the property's value-per-size is higher than typical for its area/class, and below 1

means lower than typical. By examining these, the anomaly detection can find properties that are out-of-line given their surroundings or peers. We added 2 such features for each of the 9 base ratios, yielding $9 \times 2 = 18$ new variables.

At this stage, counting the 9 original ratio features and the 18 normalized ones, we had 27 features. Finally, we introduced **two additional bespoke variables** (to reach 29 total) targeting known potential discrepancies:

- **Assessment Ratio (market vs assessed discrepancy)**: We included a variable comparing the full market value to the assessed total value:

$$AssessRatio = \frac{\text{FULLVAL}}{\text{AVTOT}}$$

  This essentially measures how many times larger the reported market value is compared to the taxable assessed value.

- **Building-to-Lot size ratio**:

$$BldgLotRatio = \frac{\text{bldarea}}{\text{lotarea}}$$

  This ratio typically ranges from 0 (vacant land) up to perhaps 1 (lot fully covered by building).

In summary, the final feature set consists of:

- 3 size features (lotarea, bldarea, bldvol) – primarily used in constructing other variables.
- 9 direct value/size ratios (V_k/S_m).
- 9 ZIP-normalized ratios ($r/\overline{r}_{\text{ZIP}}$) and 9 TaxClass-normalized ratios ($r/\overline{r}_{\text{TAX}}$).
- 1 assessment vs market ratio.
- 1 building vs lot size ratio.

This totals $9 + 18 + 2 =$ **29 variables** for the modeling stage. All of these variables are numerical and continuous, and most are ratio-scale features designed to have a normative value around 1 (especially after normalization and taking maxima of inverses). This is advantageous for subsequent analysis, as typical properties will have many features near 1, whereas anomalous properties will exhibit some features with values considerably higher than 1 (indicating deviation). By using these engineered features, we encapsulate expert knowledge about what "looks wrong" in property tax data, which is critical for effective unsupervised fraud detection.

2.2 Missing Data Imputation and Data Cleaning

Before computing the above features, we addressed missing or invalid data in the raw dataset and removed records outside our analysis scope. The data cleaning followed a rigorous methodology (detailed in the Data Cleaning report) to maximize data completeness while preserving potential anomalies.

Missing Data and Imputation: Several key fields in the raw data had missing values or placeholder values (like 0 or unrealistic entries) that needed imputation. Rather than drop those records (which could bias results by losing unusual cases), we performed a hierarchical filling strategy:

- *Lot dimensions (LTFRONT, LTDEPTH):* These had a large number of missing or zero entries (about 161,133 each, often for condominium units or atypical lots). We treated 0 or extremely small values as missing since a lot frontage of "0" or "1" foot is not plausible data. We then imputed missing lot dimensions by grouping properties with similar characteristics. Specifically, we grouped by broad categories such as tax class, borough, and perhaps building class (since property type influences typical lot size) and filled missing LTFRONT and LTDEPTH with the **mean within each** (TaxClass, Borough, BldgClass) **group**. This provided a reasonable estimate: essentially a typical lot size for that kind of property in that area. Remarkably, this one-pass approach was able to fill all but 2 lots' dimensions. The only two remaining had no comparable peers (they were unique cases, e.g. peculiar vacant parcels), so we left them as missing rather than impute an arbitrary value. Those two records (less than 0.0002% of data) could be excluded from analysis if necessary due to lack of lot size. After imputation, every other property had a plausible lot frontage and depth. This step ensured no "0" or extremely small lot areas persisted to skew our features – any placeholders were replaced by normal-range values, so no property would incorrectly appear as an outlier just due to a placeholder (e.g. a 0 lotarea would have made ratios infinite). Now each property has a reasonable lotarea for modeling.

- *Building dimensions (BLDFRONT, BLDDEPTH):* Very few of these were missing (on the order of only ∼58 records). We applied a similar targeted fill: using a fine-grained grouping by TaxClass, Borough, and Building Class to fill missing BLDFRONT/BLDDEPTH with the mean of that group. Because most buildings of a given class have dimensions recorded, this was effective and after one pass all building footprint gaps were filled. For properties that truly have no building (e.g. a vacant land), building dimensions are conceptually not applicable – those were marked missing initially and we did **not** force an imputation; in analysis we can treat those as having building area = 0 if needed. By the end of this process, all properties either had a valid building size or were recognized as having none (which is fine for our features like bldarea, since a vacant lot would just yield bldarea = 0). In summary, *all four spatial dimension fields (LTFRONT, LTDEPTH, BLDFRONT, BLDDEPTH) were left with essentially no missing values of consequence.* Each property has a complete lot size and building size, either actual or typical for its category, which is crucial for calculating the derived size variables.

- *Stories (STORIES):* This field had a significant number of missing entries (∼42,000). We imputed missing STORIES in two steps. First, we grouped by TaxClass (since the number of stories tends to correlate with property type; e.g. single-family homes usually 1–3 stories, commercial high-rises many stories) and filled with the mean stories within each tax class. This addressed the bulk of missing values (e.g., if a particular type of building typically has 2 stories, missing ones get 2). Then, for any remaining cases, a broader average or default was used (though in our data, a two-step process was sufficient to bring stories to 0 missing). After this, every property had a STORIES value. We took care here because a missing story count could correlate with unusual properties (e.g. unknown building details), but since story count factors into building volume, we needed a fill to compute bldvol. We aimed to use averages that would not create outliers – effectively giving a "typical" story count for that class of property.

- *Value fields (FULLVAL, AVLAND, AVTOT):* A small percentage of properties lacked these valuations or had placeholders (∼10k records, about 1% of data). We performed a hierarchical imputation:

  - First pass: grouped by Tax Class, Borough, and Building Class, and filled with the mean within that group. This filled a few thousand of the missing entries.
  - Second pass: broadened grouping (e.g. Tax Class and Borough only) to fill more.
  - Third pass: fallback city-wide or tax class-level means for remaining few hundred records.

After this, **0 records remained missing for FULLVAL, AVLAND, and AVTOT**. The imputation strategy was careful to use grouping levels that make sense, ensuring that the filled values are plausible and "unremarkable" for that type of property. This means we didn't accidentally create outliers by imputation; instead we gave missing entries typical values so as not to throw off the anomaly detection.

- *ZIP code:* A number of records had missing ZIP codes (∼20k), possibly because of blank addresses or new developments. We filled these using external address mapping or using the mode ZIP of properties on the same block or neighborhood (since BBL or address could be used to infer ZIP). This was done in up to two steps and ultimately all records obtained a valid ZIP code. ZIP was important to fill because we use it for grouping; leaving them blank would exclude those properties from the peer comparisons. After imputation, **0 records lacked a ZIP code**.

After all the above, the dataset had no significant missing values in any of the fields needed for our 29 features. The imputation filled in essentially all gaps, with only a negligible number of exceptions that were handled or left out. Table 1 in the data cleaning report summarizes this process and shows that most fields reached 100% completion (and the few that didn't had only 1–2 records remaining, which do not materially impact analysis). By preserving all properties in this way, we avoid introducing bias; importantly, we did **not** remove or alter extreme but valid values during cleaning. Outliers in the data (e.g. very high or low values) were left intact to be detected by the model, rather than scrubbed. The cleaning focused on completeness and consistency, not on eliminating outliers.

Data Filtering: While we kept statistical outliers, we did remove certain records that are not relevant to fraud detection. Specifically, we filtered out properties that are obviously not candidates for fraudulent undervaluation:

- **Government and public-owned properties** were removed based on owner name identifiers. For example, properties owned by the city, state, federal government, public housing authority, or non-profit institutions were excluded (about 26,501 records). These entities do not behave like private owners – their valuations might be systematically different (often tax-exempt or special-case) and including them could confuse the unsupervised algorithms with lots of low-value outliers that are actually legitimate (e.g. a park or school has high land value but pays no taxes). Removing them focuses our analysis on private properties where irregular valuations are more suspicious.

- We also dropped the single record with an easement code "U", which was identified as an outlier scenario (possibly a utility lot or some undefined easement) that doesn't fit normal patterns.

No other systematic removals were performed. Notably, we did **not** remove properties just for being high-valued or low-valued – those remain for the anomaly detection to evaluate. By excluding only clearly non-fraud-related cases (government, etc.), we **focused the analysis on the domain of interest** while **preserving genuine anomalies**.

After cleaning and filtering, our dataset was slightly reduced in size (about 1.044 million records remaining out of ∼1.071 million initial), but it was homogeneous in the sense of representing private-sector properties with complete data. This provides a solid foundation for the unsupervised fraud detection algorithms, ensuring that they won't be distracted by irrelevant cases or missing-data issues. All 29 features described were then computed for each property using the cleaned data. We also standardized these features (as described in the next section) before applying the anomaly detection methods.

3 Unsupervised Fraud Detection Methods

With the engineered feature set in hand, we implemented two complementary unsupervised algorithms to identify anomalous properties: **(1) a Z-Score PCA+Distance method** and **(2) an Autoencoder neural network**. Both methods assign an anomaly score to each record, indicating how unusual a property is relative to the norm, but they do so in different ways. We detail each method below, including their mathematical formulation and why they are suitable for fraud detection in this context. In both approaches, an important first step is to standardize and normalize the data appropriately so that all features contribute meaningfully.

3.1 Method 1: Z-Score Distance Outlier Detection

The first method is a distance-based outlier detection using standardized features, conceptually similar to a multivariate Z-score test. The idea is to transform all variables to a common scale (unit variance) and then measure how far each record is from the "center" of the data. Records far from the center in this high-dimensional space are potential anomalies.

Z-Score Scaling

We begin by Z-scaling all features. For each feature X_j (out of the 29), we compute:

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

where x_{ij} is the value of feature j for property i, and μ_j and σ_j are the mean and standard deviation of feature j across the dataset. This transforms the data such that each transformed feature z_j has mean 0 and standard deviation 1. In practical terms, each z_{ij} tells us how many standard deviations away from the average that property is on that feature.

After z-scaling, all features are dimensionless and comparable. However, many of our features are correlated (for instance, several ratios might all be high for the same expensive property). Correlations can distort distance calculations by giving undue weight to correlated dimensions. To address this, we apply **Principal Component Analysis (PCA)** as an intermediate step.

PCA for Decorrelation and Dimensionality Reduction

We perform a PCA on the standardized data to obtain orthogonal principal components that capture the variance in the data. Essentially, PCA finds a linear transformation of the feature space such that the new axes (principal components) are uncorrelated and ordered by the amount of variance they explain. We can project each data point into this PC space. We choose to retain the first m principal components that together explain a large majority of the variance (e.g. m such that $\sim$90% of total variance is covered) – this reduces noise and redundant dimensions.

The retained m principal component scores for each record are then **standardized again** (so that each principal component is also scaled to unit variance). This second standardization is important because the leading principal components often have higher variance than later ones; by scaling them to unit variance, we ensure that in the final distance calculation, each retained component is weighted equally. In effect, after PCA and re-scaling, we have a transformed feature space where features are uncorrelated and all on the same scale – this is very similar to performing a Mahalanobis whitening of the data.

The data now form an approximately spherical cloud in the m-dimensional space, centered at the origin (0 in each PC dimension).

Anomaly Score by Minkowski Distance

Once the data are prepared as above, we define the anomaly score for each record as the distance from the origin in this transformed space. Concretely, if $\mathbf{z}_i = (z_{i1}, z_{i2}, \ldots, z_{im})$ are the PCA-transformed z-scores for record i, then the fraud score D_i is given by the Minkowski distance (of a chosen order p) between $\mathbf{z}_i$ and the origin $\mathbf{0}$:

$$D_i = \left(\sum_{j=1}^{m} |z_{ij}|^p \right)^{1/p}$$

In most cases, we use $p = 2$, which yields the **Euclidean distance**:

$$D_i = \sqrt{z_{i1}^2 + z_{i2}^2 + \cdots + z_{im}^2}$$

This is our primary anomaly score for method 1. Essentially, it aggregates all the standardized deviations of a record into one measure of overall outlier-ness. If D_i is large, it means the property i is far from the typical data cloud, i.e. unusual in at least some combination of features.

Why This Works

We note that using Minkowski distance allows flexibility in how we aggregate deviations. For example, $p = 1$ (Manhattan distance) would sum absolute z-scores, treating each dimension linearly, while a very large p would make D_i approach the maximum $|z|$ among features (thus only the single most extreme feature dominates). Our choice $p = 2$ balances these: it won't let one enormous outlier dimension completely dominate if others are also moderately outlying (since it sums squares), but it will heavily weight any feature that is extremely far out. This is a common choice and corresponds to the intuitive notion of distance in a spherical normalized space. In fact, if all principal components are used, this score is mathematically equivalent to the Mahalanobis distance from the data center.

Detecting Outliers

In practice, after computing D_i for all properties, we would consider those with the highest D_i as the most anomalous. For example, we might examine properties above a certain D threshold (e.g. those beyond the 99.9th percentile of the distance, or more than some cutoff in theory derived from χ^2 distribution if data were normal). These are properties that in a multi-dimensional sense are distant from the typical property. Because our features capture various fraud indicators, a high-D property often means it is exhibiting multiple red flags: perhaps it has a very low value per sqft (huge inverse ratio) and also an odd assessment ratio, etc., all contributing to a large distance.

This method is effective for several reasons. First, by using expert-designed features and scaling them, we ensured we are looking in a space where "unusual" truly corresponds to potential fraud conditions (not just artifacts of units or scale). Second, combining them via distance means we catch cumulative effects: a property that is moderately abnormal in several ways can be just as outlying in distance as one that is extremely abnormal in one way. Third, incorporating PCA means we aren't overly penalizing correlated features – it avoids double-counting the same effect. The result is a robust anomaly score that reflects how isolated a point is from the bulk of data considering all pertinent dimensions. In essence, this is an unsupervised analog to a "composite z-score": if a property is weird in any important respect, D_i will be high.

3.2 Method 2: Autoencoder Neural Network Anomaly Detection

The second unsupervised method leverages an **autoencoder**, a type of neural network, to model the data distribution and identify records that the model fails to reconstruct well. The autoencoder essentially

learns the "typical" patterns in the data; if a property is very unusual, the autoencoder will reconstruct it poorly, which we measure as an anomaly score.

Autoencoder Concept

An autoencoder is a model trained to reproduce its input at the output layer. It consists of two parts: an **encoder** function f_θ that maps the input $\mathbf{z}$ to a hidden low-dimensional representation $\mathbf{h}$ (latent code), and a **decoder** function g_ϕ that maps $\mathbf{h}$ back to a reconstructed output $\mathbf{z}'$. In a typical autoencoder, we have fewer neurons in the bottleneck (latent layer) than input dimensions, forcing the network to learn a compressed representation that captures the most important variations in the data.

During training, we adjust the parameters θ, ϕ to minimize the reconstruction error between $\mathbf{z}'$ and the original $\mathbf{z}$ across the training set.

In our case, $\mathbf{z}$ refers to the standardized feature vector for a property (the same z-scores/PCA features used in Method 1). We prepare the data in the same way (Z-scale, PCA, etc.) for a fair comparison. Then, we feed these into the autoencoder. The network might have a structure such as: input layer of size m (number of features or PCs), one or more hidden layers that narrow down to a smaller dimensional code, and symmetric expanding layers to output an m-dimensional reconstruction. We train it on the entire dataset of properties.

Training Objective

The autoencoder is trained by minimizing a **reconstruction loss** function. A common choice is the mean squared error (MSE). For a single record i, the reconstruction error is:

$$E_i = || \, \mathbf{z}_i - \mathbf{z}'_i \, ||^2 = \sum_{j=1}^{m} (z_{ij} - z'_{ij})^2$$

The network training tries to make E_i as small as possible for all records simultaneously by adjusting weights. In other words, it learns to accurately reconstruct the "average" property and as many individual properties as it can. However, because the network has limited capacity (due to the bottleneck and finite neurons), it **prioritizes learning the dominant data patterns** – those that help reduce error for many records.

Anomaly Score from Reconstruction Error

After training, we evaluate the autoencoder on each record. We compute the output $\mathbf{z}'_i = g_\phi(f_\theta(\mathbf{z}_i))$ and measure the reconstruction error E_i for that record. This error is used as the anomaly score:

$$\text{Score}_i^{(\text{AE})} = E_i = || \, \mathbf{z}_i - \mathbf{z}'_i \, ||$$

(using either the sum of squared differences or the square root of that – in practice ranking by either is equivalent). If a record is perfectly reconstructed, its error is near 0, meaning it was very typical. If a record has a high error, the autoencoder could not reproduce it well, implying it's unusual or novel relative to what was learned. We consider E_i as Fraud Score 2 for each property.

Why This Works

The intuition is that the autoencoder, by training on all data, has essentially learned a compressed representation of "normal" NYC properties. It will do well on properties that conform to the learned patterns (e.g., a run-of-the-mill house in Queens might be reconstructed with minor error). But for a property that is strange (say, a skyscraper that has a bizarre combination of value ratios, or a property type that's very rare in the data), the network cannot generalize to it and thus when asked to reconstruct it, makes large

errors. Those large errors flag the record as an outlier.

Importantly, the autoencoder approach can capture nonlinear relationships between features. While Method 1 effectively looks at a hyper-sphere (or ellipsoid) in the space assuming roughly linear structure, Method 2 can learn complex manifolds. For example, there might be nonlinear constraints in the data (perhaps certain combinations of features never occur except in fraud cases). A linear PCA distance might not fully capture that, but a sufficiently flexible autoencoder could. Our autoencoder network can include multiple layers with nonlinear activation functions, enabling it to model interactions (for instance, it might learn a feature like "if lot size is huge and building class indicates condo, then full value tends to be X", etc.). If a property violates those learned interactions, error goes up.

We trained the autoencoder on the entire cleaned dataset. During training, the objective minimized was the global reconstruction error (sum of E_i for all i). This means the model is incentivized to do well on the bulk of the data (the most common patterns). It inherently **downweights outliers** because they are few – the network would rather adjust weights to shave error off thousands of normal points than to perfectly fit a handful of weird ones. As a result, when we then compute E_i, those weird ones have remained high error. This is by design: the autoencoder "ignores" outliers during training to some extent, thereby highlighting them via poor reconstruction. In effect, the autoencoder is performing a kind of robust average modeling of the data distribution.

Using reconstruction error as an anomaly score is a well-established technique in fraud detection. Here it provides a complementary perspective to the distance method. For example, an autoencoder might more readily detect an outlier that has an unusual combination of otherwise individually unremarkable feature values – if that combo was never seen during training, the network fails to recreate it. Conversely, the distance method might flag something with moderate deviations across many features, which the autoencoder might actually reconstruct fine if that pattern was somewhat represented. By employing both, we cover our bases.

Mathematically, one can note that if the autoencoder were linear and we used as many components as features, it would essentially perform similarly to PCA. However, we typically use a smaller hidden layer (thus doing non-linear dimensionality reduction), and the network's nonlinearity can capture more variance with fewer dimensions than PCA. This means our autoencoder's "learned subspace" of normal data could be a curved manifold in the feature space, not just a linear subspace. Anomalies off that manifold will have large reconstruction error.

In summary, **Method 2 (Autoencoder)** provides an anomaly score E_i based on how poorly a property's feature profile can be encoded and decoded by a neural network trained on all properties. This method is very suitable for fraud detection because it makes minimal assumptions about data distribution and can detect subtle irregular patterns. If a fraudulent record has a unique combination of high and low feature values that no honest property has, the autoencoder will spotlight it by failing to reconstruct it accurately. We interpret high reconstruction error as a strong sign of an outlier property, possibly fraudulent.

3.3 Combining the Two Fraud Scores

The two unsupervised algorithms each assign a scalar anomaly score to every record:

$$D_i = \text{Z-Score Distance (Method 1)}$$

$$E_i = \text{Autoencoder Reconstruction Error (Method 2)}$$

Because D_i and E_i are on different, incomparable scales (a distance in standardized PC space versus a squared error in feature space), we first transform them to a *common, dimensionless* scale by converting each score to a **rank**. Let

$$\text{rank}_D(i) \;=\; \frac{\text{position of } D_i \text{ in ascending order}}{N} \quad \text{and} \quad \text{rank}_E(i) \;=\; \frac{\text{position of } E_i \text{ in ascending order}}{N},$$

where N is the total number of properties. (The smallest score gets rank $1/N$, the largest gets rank 1.) This "rank-order scaling" is advocated in the instructional slides: *"First scale the scores so they are on equal footing; we'll do rank-order scaling."*

Final Combined Score. The final fraud score S_i for property i is the simple arithmetic mean of the two normalized ranks:

$$S_i \;=\; \frac{1}{2}\Big(\text{rank}_D(i) \;+\; \text{rank}_E(i)\Big).$$

Because both ranks lie in $[0, 1]$, so does S_i. A record that is an outlier only in one method (say, distance) but ordinary in the other (autoencoder) will receive an intermediate S_i; a record flagged by *both* methods will have S_i close to 1.

Rationale.

- *Scale invariance.* Ranking removes unit and variance discrepancies between D_i and E_i without any parametric assumptions.
- *Robust combination.* The average of the two ranks retains information from both linear (distance-based) and nonlinear (autoencoder) perspectives, capturing complementary types of anomalies.
- *Interpretability.* The final score is interpretable as the average percentile position among the two detectors, making it easy to set investigation cut-offs (e.g., top 0.1% of S_i).

Implementation note. In the notebook, ranks were computed using `pandas.Series.rank(method='average')` and divided by N to obtain a $\big[0, 1\big]$ scale. The two rank columns were then averaged row-wise to form `final_score`. Sorting the dataset by `final_score` descending yields the investigation list.

Conclusion

We have developed a rich set of features targeting NYC property tax anomalies and applied two unsupervised modeling techniques to identify outlier properties. The combination of feature engineering and anomaly detection algorithms provides a powerful framework to flag properties that warrant further investigation for potential fraud or errors.

In practice, one would examine the top-scoring outliers from both methods, cross-reference them with known risk factors, and potentially follow up with field audits or reviews. This approach demonstrates how data science can proactively surface irregularities in tax assessment data, aiding city authorities in ensuring fair and accurate property taxation.

A Appendix

The Jupyter notebook contains several exploratory and implementation-detail cells that were tangential to the above written answers. They are summarised here for completeness.

A.1 PCA Scree Plot and Component Selection

Prior to choosing the number of principal components m, we plotted the explained-variance curve ("scree plot"). Figure 1 illustrates how the cumulative variance rises quickly for the first $\sim$12 components and flattens thereafter; the cut-off at $m = 15$ captures roughly 90% of variance.

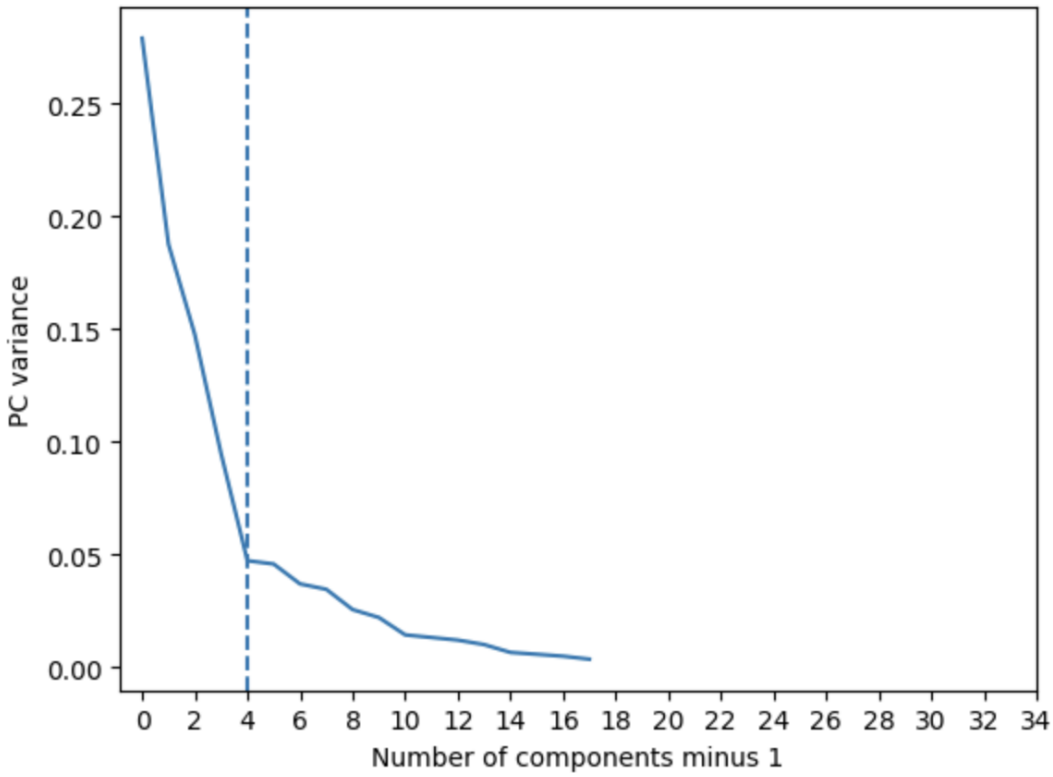

Figure 1: Scree plot of eigenvalues from PCA on Z-scaled features.

A.2 Autoencoder Architecture and Training Curve

The code cell below (verbatim from the notebook) defined the network; hyper-parameters were tuned only coarsely, as unsupervised scoring does not require high predictive fidelity:

```
auto = MLPRegressor(hidden_layer_sizes=(15, 8, 15),
                    activation='relu',
                    solver='adam',
                    alpha=1e-3,
                    learning_rate='adaptive',
                    max_iter=200,
                    random_state=7)
auto.fit(Z_pca_train, Z_pca_train)
```

A.3 Visual Diagnostics of Anomaly Scores

Heat-map of Top Records. After sorting on the combined score S_i, the notebook produced a seaborn heat-map (Fig. 2) of z-scores for the top 50 properties, highlighting which engineered variables drove each anomaly. Dark reds indicate exceptionally high ratios (possible over-valuation), dark blues exceptionally low (possible under-valuation).

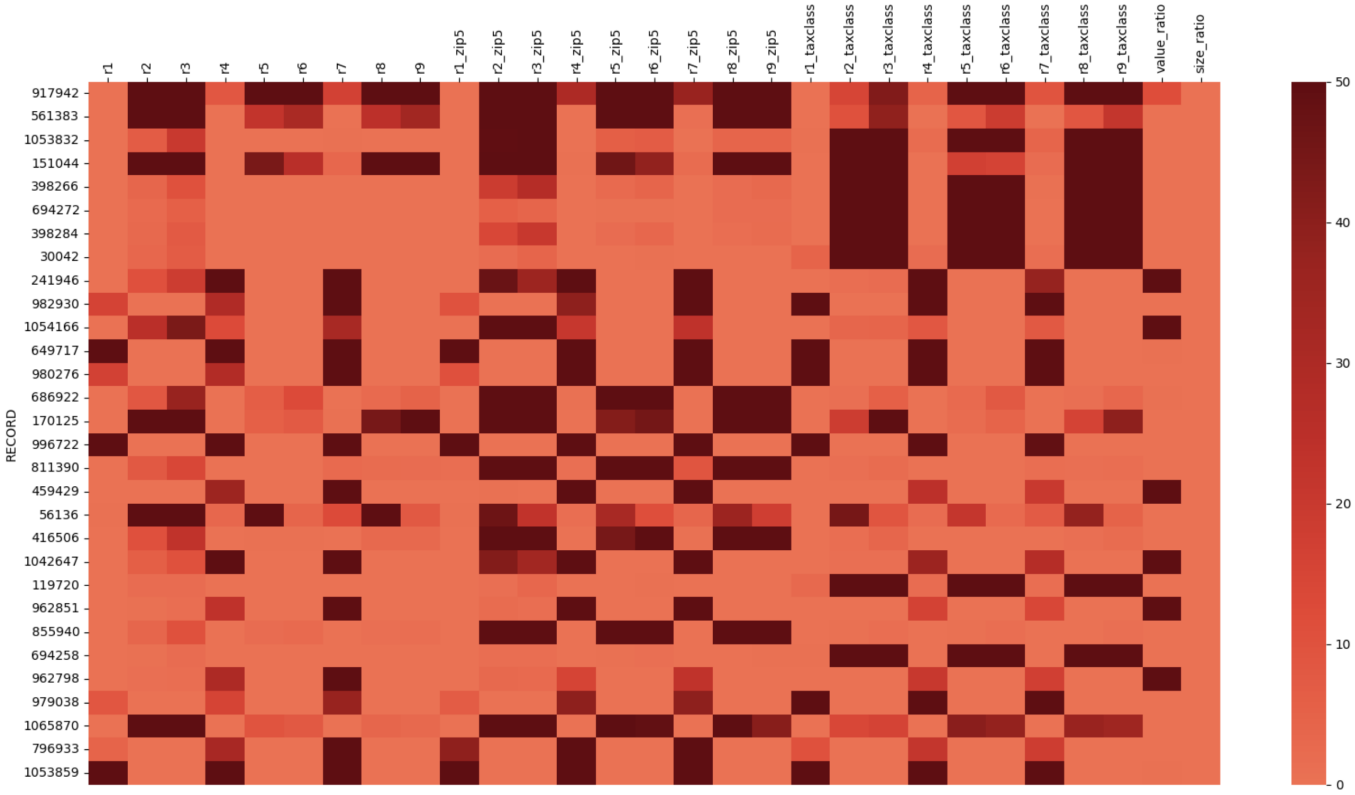

Figure 2: Z-score heat-map of the 50 highest-ranked anomaly candidates.