# House Price Dataset Preprocessing Report – Group 3

This report details all the changes and transformations applied to the raw house price dataset, covering missing value handling, column dropping, feature engineering, categorical encoding, and numerical scaling/normalization. These steps were designed to prepare a clean, engineered, and preprocessed dataset for machine learning model training.

## Data Cleaning and Column Removal

The initial phase involved cleaning the dataset, specifically by addressing missing values and removing irrelevant columns:

* **Handling Missing Values:** The column **'No of Times Visited'** was identified as having a high percentage of missing data and was therefore dropped from the dataset.
* **Dropping Columns:** Several columns deemed to have low relevance for the house price prediction task were removed. These include:
  + 'Waterfront View'
  + 'Living Area after Renovation (in Sqft)'
  + 'Area of the House from Basement (in Sqft)'
  + 'Renovated Year'
  + 'Lot Area after Renovation (in Sqft)'
  + 'No. of Times Visited'

## Feature Engineering

The initial step in feature engineering involved examining the raw dataset to identify implicit information that could be transformed into explicit and more informative features for a machine learning model predicting house sale prices. Factors considered included the relationships between existing columns and how they might collectively represent underlying characteristics of the houses and their locations. For instance, while individual features like Zipcode, Latitude, and Longitude provide geographical information, their raw values might not directly capture the nuances of location quality or neighborhood value.

Existing features often have limitations that feature engineering can help address. For example, Zipcode is a categorical variable with a high cardinality (many unique values). Using it directly in some machine learning models can be challenging and might not effectively capture the spatial relationship between different zip codes. Similarly, Flat Area (in Sqft) and Sale Price individually are useful, but their ratio (price\_per\_sqft) can provide a standardized metric for comparing the value of properties regardless of their size.

We aim to provide the model with more relevant and discriminative information, ultimately leading to more accurate predictions of house sale prices. This involves thinking about what factors a human might consider when valuing a house and trying to represent those factors numerically.

**LocationQuality:** This feature attempts to synthesize geographical information (Zipcode, Latitude, Longitude) into a single metric that reflects the perceived value or desirability of a location. By incorporating the average sale price of a zipcode, it leverages the collective market valuation of properties in that area. Combining this with latitude and longitude allows for finer-grained spatial information to be included. This approach is more likely to capture complex spatial patterns related to price than simply using raw coordinates or a high-dimensional one-hot encoding of zipcodes.

**Price\_per\_sqft**: This feature provides a normalized measure of value that is independent of the absolute size of the house. It's a commonly used metric in real estate analysis and directly reflects how much value is packed into each square foot of a property. This is expected to be a strong predictor of sale price and helps in comparing properties of different sizes on a level playing field.

In summary, the chosen features were selected because they are intuitively related to house valuation, are expected to have a strong correlation with the target variable and offer a more compact and potentially less overfitting-prone representation of location and value density compared to some alternative methods. These features address limitations of the raw data by creating more informative and directly relevant variables for the machine learning model.

|  |  |
| --- | --- |
| **New Feature** | **Rationale and Calculation** |
| **price\_per\_sqft** | Provides a standardized, size-independent metric for comparing property values. Calculated by dividing **'Sale Price'** by **'Flat Area (in Sqft)'**. |
| **LocationQuality** | A composite feature capturing the property's desirability and value. It was created by first calculating the **average sale price for each unique Zipcode** and then summing this average price with the property's specific **Latitude** and **Longitude** coordinates. |

In addition to these two numerical features, a new categorical feature, **LocationQuality\_Grade**, was created by rounding the numerical LocationQuality values and assigning letter grades (F, D, C, B, A) based on predefined bins, effectively discretizing the location quality.

## Scaling, Normalization, and Encoding

The final stage involved transforming the numerical and categorical features into a format suitable for machine learning algorithms.

**Scaling and Normalization:**

**Purpose:** The goal was to standardize the range of numerical features and handle skewed distributions. Machine learning algorithms often perform better when numerical features are on a similar scale and have distributions closer to normal.

**Process:** We applied Min-Max Scaling to most numerical columns (excluding 'ID', 'Sale Price', and the LocationQuality features) to transform their values into a fixed range, specifically between 0 and 1. For features identified as potentially skewed (like 'Flat Area', 'Lot Area', and 'Basement Area'), we also explored Log Normalization (using np.log1p) to reduce the impact of extreme values and make their distributions more symmetrical.

**Outcome:** The summary statistics and visualizations of the scaled data showed that the numerical features were successfully brought into the [0, 1] range, making them comparable and suitable for distance-based or gradient-descent-based models. Log normalization helped in transforming the skewed distributions.

**Encoding Categorical Data:**

**Purpose:** Machine learning models require numerical input. This phase converted our categorical features ('Condition of the House' and 'LocationQuality\_Grade') into a numerical format while preserving their inherent order.

**Process:** We used Ordinal Encoding because the categories in these columns ('Bad' to 'Excellent' for condition, and 'F' to 'A' for Location Quality Grade) have a clear, ranked order. We defined the specific order for each feature, and the encoder assigned a unique integer to each category based on its position in the defined order (e.g., Bad=0, Okay=1, Fair=2, etc.).

**Outcome:** The categorical features were successfully transformed into numerical integers. This allows our machine learning model to interpret the ordered nature of these features correctly. We confirmed that there were no nominal categorical features remaining that would require one-hot encoding.

In summary, these steps were crucial in preparing the data by making numerical features comparable through scaling and normalization and by converting categorical features into a machine-readable numerical format while retaining their ordinal information. The dataset is now well-prepared for the next phase of model building.

### Numerical Feature Transformation (Scaling and Normalization)

Numerical features, which operate on vastly different scales (e.g., price vs. bedrooms), were transformed to ensure variables contribute equally to the model and to handle skewness.

|  |  |  |
| --- | --- | --- |
| **Transformation** | **Goal** | **Application** |
| **Scaling (Min-Max)** | To equalize the range of features, typically confining them to the **[0, 1]** range, which prevents large-magnitude variables from dominating the model. | Applied to most numerical features, such as square footage metrics. |
| **Normalization (Log)** | To adjust the shape of the distribution by reducing the impact of skewness and outliers, particularly in highly skewed data. | Applied to potentially skewed features, including monetary and size attributes like **area**, **lot sizes**, and the target variable **price**. |

### Categorical Feature Encoding

Categorical features were converted into numerical representations, preserving their inherent order where applicable.

|  |  |  |
| --- | --- | --- |
| **Encoding Method** | **Features** | **Rationale** |
| **Ordinal Encoding** | **'Condition of the House'** and **'LocationQuality\_Grade'** | Used because these features have an **intrinsic order or hierarchy** (e.g., Poor -> Excellent), allowing the model to recognize increasing quality. |
| **One-Hot Encoding** | This technique is used when categories have no natural order.  Each unique category becomes a separate binary column with values 0 or 1. | Used for nominal (unordered) features to prevent the model from assuming an arbitrary numerical relationship or order between categories. |

**Summary**

This report details the preparation of the house price dataset for machine learning. This involved three major steps: **data cleaning**, **feature engineering**, and **transformation**.

For **cleaning**, several irrelevant columns were dropped, and the feature 'No of Times Visited' was removed due to excessive missing data. In **feature engineering**, two key variables were created: **price\_per\_sqft** for a standardized valuation metric, and **LocationQuality** to synthesize geographical information and average neighborhood price, which was then categorized into a new **LocationQuality\_Grade** feature (A-F). Finally, for **transformation**, **Min-Max Scaling** was applied to numerical data to normalize ranges, **Log Normalization** was used on skewed features to handle outliers, and **Ordinal Encoding** was used on features like 'Condition of the House' and 'LocationQuality\_Grade' to preserve their inherent ordered structure.