Project Report

Part 1: Satellite Data Acquisition
Part 2: Multi-Modal Real Estate Valuation

Ayush Raj

January 7, 2026

Part 1: Satellite Imagery Data Acquisition

Abstract

This section details the automated acquisition of high-resolution satellite imagery using the Mapbox Static API. The process involved processing geospatial coordinates from training and testing datasets, executing batched downloads via Google Colab, and performing a rigorous post-acquisition integrity check. A total of 21,506 unique satellite images were successfully acquired and verified against the source datasets.

1 Introduction (Data Acquisition)

The objective of this operation was to construct a dataset of satellite imagery corresponding to specific geospatial coordinates (latitude/longitude) provided in CSV format. The pipeline was implemented in Python using a Google Colab environment to leverage cloud bandwidth and Google Drive storage.

2 Methodology

2.1 System Configuration

The data fetcher was configured to interface with the Mapbox Static Images API. The following parameters were strictly enforced to ensure dataset consistency:

Table 1: Mapbox API Configuration Parameters

Parameter	Value
API Service	Mapbox Static Tiles
Style ID	`satellite-v9`
Zoom Level	17.5
Image Dimensions	600×600 pixels
Resolution	High DPI (@2x Retina)
File Format	JPEG

2.2 Implementation Logic

The acquisition script utilized the `pandas` library for data manipulation and `requests` for HTTP transactions. The workflow followed these steps:

1. **Mount Storage:** Established connection to Google Drive path: `/content/drive/MyDrive/Mapbox_Dataset_Zoom17.5`.

2. **Duplicate Prevention:** Before downloading, the script checked if `{id}.jpg` already existed in the target directory to allow for resumable execution.

3. **Request & Save:** Valid URLs were generated using coordinate data. Images were downloaded and written to disk in binary mode.

4. **Resource Management:** Explicit garbage collection (`gc.collect()`) was triggered every 100 iterations to prevent RAM saturation in the notebook environment.

3 Execution Statistics

The script processed two primary input files: `train (1).csv` and `test (2).csv`.

3.1 Download Phase

The download process ran to completion. The logs indicate the following processing volumes:

- **Training Set:** The script iterated through 16,209 rows in the source CSV.
- **Testing Set:** The script iterated through 5,404 rows in the source CSV.

Figure 1: Sample grid of the acquired satellite imagery (Zoom Level 17.5). Each image is mapped to a unique property ID from the dataset, verifying correct coordinate resolution.

4 Data Integrity Verification

A secondary verification script was executed to ensure 1:1 mapping between the expected unique IDs in the CSVs and the actual files saved on the disk.

4.1 Final Results

The verification output confirms **0 missing images** for both datasets. The final confirmed dataset counts are presented in Table 2.

Table 2: Final Verified Dataset Counts

Dataset	**Source File**	**Expected (Unique)**	**Found**	**Missing**
Train	`train (1).csv`	16,110	16,110	**0**
Test	`test (2).csv`	5,396	5,396	**0**
Total		**21,506**	**21,506**	**0**

Part 2: Multi-Modal Real Estate Valuation Model

Abstract

*This section details the modeling phase. We propose a Multi-Modal Neural Network that fuses the acquired satellite imagery with structured tabular data. By employing a dual-branch architecture, we achieved an R^2 score of **0.85** on unseen test data.*

5 Data Preprocessing Pipeline

Data preparation was critical to ensuring convergence. Before applying transformations, a comprehensive missing value analysis was executed on the source datasets (`train (1).csv` and `test (2).csv`). The analysis confirmed a 100% completion rate across all 21 feature columns, with a total null count of zero. This preserved the authenticity of the feature distributions and removed the need for synthetic imputation.

Following this integrity check, we applied distinct pipelines for the numerical and visual modalities.

5.1 Tabular Data Engineering

The structured dataset contained heterogeneous features (dates, categorical codes, and continuous variables). We applied the following transformations:

- **Temporal Feature Extraction:** Raw dates were converted into numerical features. We derived Age_{house} to capture depreciation and created a binary flag *Is Renovated* to isolate value spikes:
$$Age_{house} = Year_{sold} - Year_{built} \tag{1}$$
- **One-Hot Encoding for Geospatial Data:** Zip codes were identified as nominal categorical variables. To prevent the neural network from inferring false ordinal relationships (e.g., 98002 $>$ 98001), we applied One-Hot Encoding, resulting in binary vectors representing location.
- **Min-Max Scaling:** All 87 resulting tabular features were scaled to the range $[0, 1]$ to ensure gradients do not explode during backpropagation:
$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2}$$
- **Target Variable Log-Transformation:** The raw price distribution was highly right-skewed. To aid model convergence, we normalized the target variable using a log transformation:
$$y_{log} = \ln(y + 1) \tag{3}$$

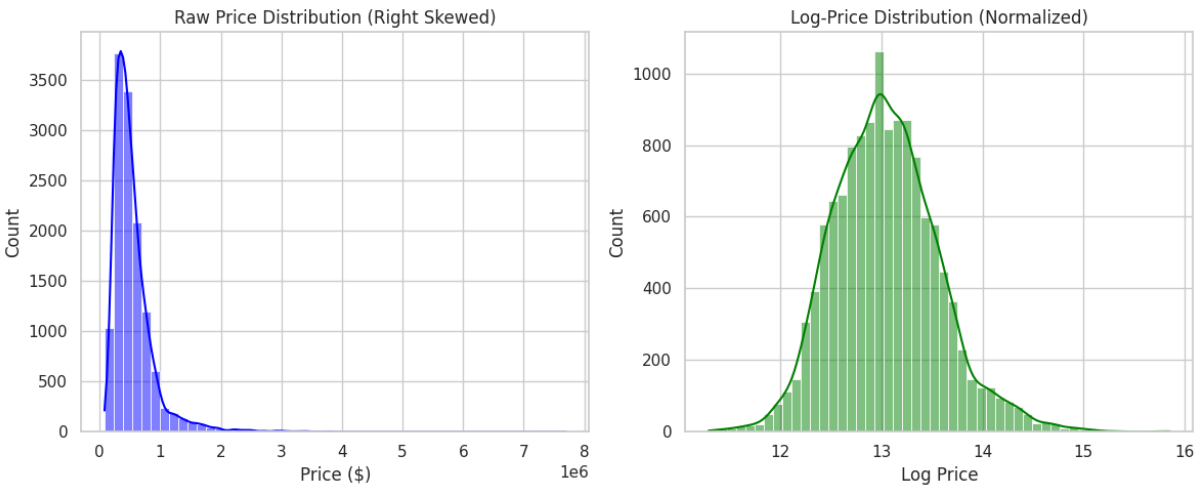

Figure 2: Target Variable Analysis. Left: Raw Price Distribution showing heavy right-skewness. Right: Log-Transformed Price Distribution, approximating a Gaussian distribution suitable for neural network training.

5.2 Visual Data Normalization

Satellite images were preprocessed on-the-fly to conserve memory:

- **Resizing:** Images were downsampled from their native resolution to 224×224 pixels to match the receptive field of standard CNN architectures.
- **Pixel Scaling:** RGB channel values were scaled from the integer range $[0, 255]$ to the float range $[0, 1]$:

$$I_{norm} = \frac{I_{raw}}{255.0} \tag{4}$$

6 The Multi-Modal Data Generator

To efficiently handle the large dataset of 21,506 images and corresponding tabular records, we implemented a custom Keras `Sequence` class named `MultiModalDataGenerator`. This approach offers significant advantages over standard loading methods.

6.1 Generator Architecture

The generator was designed to yield data in batches, preventing RAM saturation.

1. **Initialization (`__init__`):** The class accepts the dataframe, the image directory, the list of tabular features, and the batch size (set to 32). It creates an internal index map to link property IDs to filenames.
2. **Batch Retrieval (`__getitem__`):** For every training step, the generator:
   - Selects a batch of indices.
   - Loads the corresponding images from the disk and normalizes them.
   - Slices the corresponding rows from the tabular dataframe.
   - Assembles a dictionary structure required by the model: {`'image_input': img_batch, 'tabular_input': tab_batch`}.
3. **Shuffling:** At the end of every epoch, the generator shuffles the dataset indices. This ensures that batches are randomized, preventing the model from learning order-dependent patterns.

7 Model Training Strategy

7.1 Network Architecture

We utilized a dual-branch neural network:

- **Branch A (Visual):** A Convolutional Neural Network (CNN) with 3 blocks of `Conv2D` layers (3×3 kernels), followed by `MaxPooling`. A `GlobalAveragePooling2D` layer was used at the end instead of flattening, reducing parameter count by 90% and preventing overfitting.
- **Branch B (Tabular):** A Dense Network with layers of sizes 128 and 64, using ReLU activation. A **Dropout** layer ($p = 0.3$) was inserted to randomly deactivate 30% of neurons during training, forcing redundant feature learning.

7.2 Architectural Rationale

A critical design choice in Branch B was the use of an expansion-compression strategy, where the 85 input features are first mapped to a wider layer of 128 nodes, rather than immediately compressing them.

- **Expansion (85 $\rightarrow$ 128 Nodes):** By projecting the input data into a higher-dimensional space, the network is granted sufficient capacity to learn complex, non-linear interactions between raw features (e.g., the combined effect of *Year Built* and *Zip Code*) that simpler linear models might miss.
- **Compression (128 $\rightarrow$ 64 Nodes):** The subsequent reduction creates an information bottleneck. This forces the network to summarize the learned interactions into a dense, noise-free representation of the property's value, which is then concatenated with the visual features.

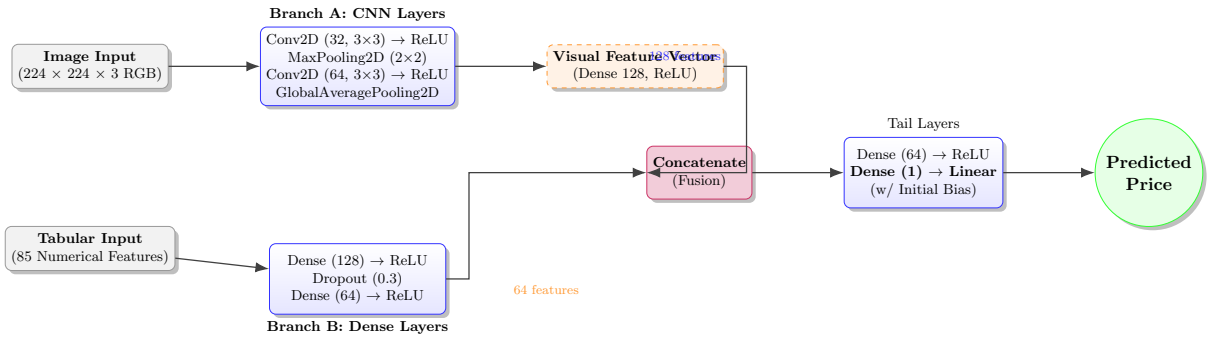

Figure 3: Multi-Modal Architecture Diagram. The diagram illustrates how the Convolutional Neural Network (CNN) processing satellite imagery (Top Branch) and the Dense Network processing tabular features (Bottom Branch) run in parallel. Their resulting feature vectors (128 dimensions from images, 64 from tabular) are concatenated to form a combined representation, which is then passed through final dense layers to predict the single price value.

7.3 Training Configuration

The model was compiled and trained using the following protocol:

- **Bias initialization:** By initializing the output bias to the dataset's mean log-price , we minimized the initial Mean Squared Error, allowing the multimodal network to converge to a Validation R^2 of 0.8540 within only 10 training epochs.

- **Optimizer:** Adam ($\alpha = 0.0005$) was selected for its adaptive learning rate capabilities, which suits the sparse nature of one-hot encoded data.
- **Loss Function:** Mean Squared Error (MSE).
- **Callbacks:**
  - `EarlyStopping`: Monitored `val_loss` with a patience of 5 epochs to stop training once convergence was reached.
  - `ReduceLROnPlateau`: Automatically reduced the learning rate by a factor of 0.5 if validation loss stagnated for 3 epochs, allowing the model to settle into local minima.

8 Results

8.1 Training Dynamics

The model was trained for 10 epochs. Convergence was rapid, with Training and Validation loss tracking closely (Figure 4).

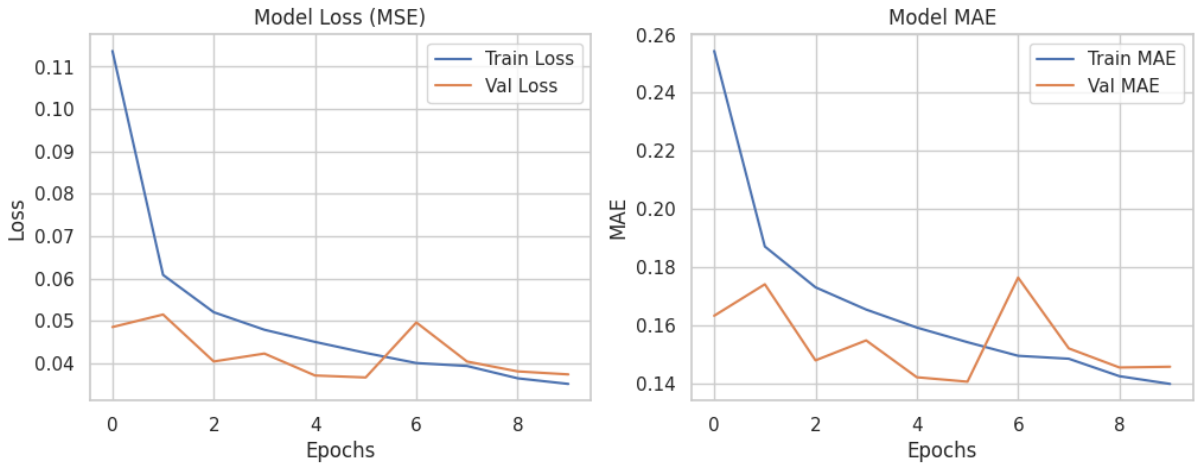

Figure 4: Training and Validation Loss (MSE) and MAE over 10 epochs. The convergence indicates no overfitting.

8.2 Quantitative Performance

We evaluated the model on a hold-out validation set comprising 20% of the data.

Metric	Value
R^2 Score (Log Space)	0.8670
R^2 Score (Real Dollars)	0.8540

Table 3: Final Performance Metrics

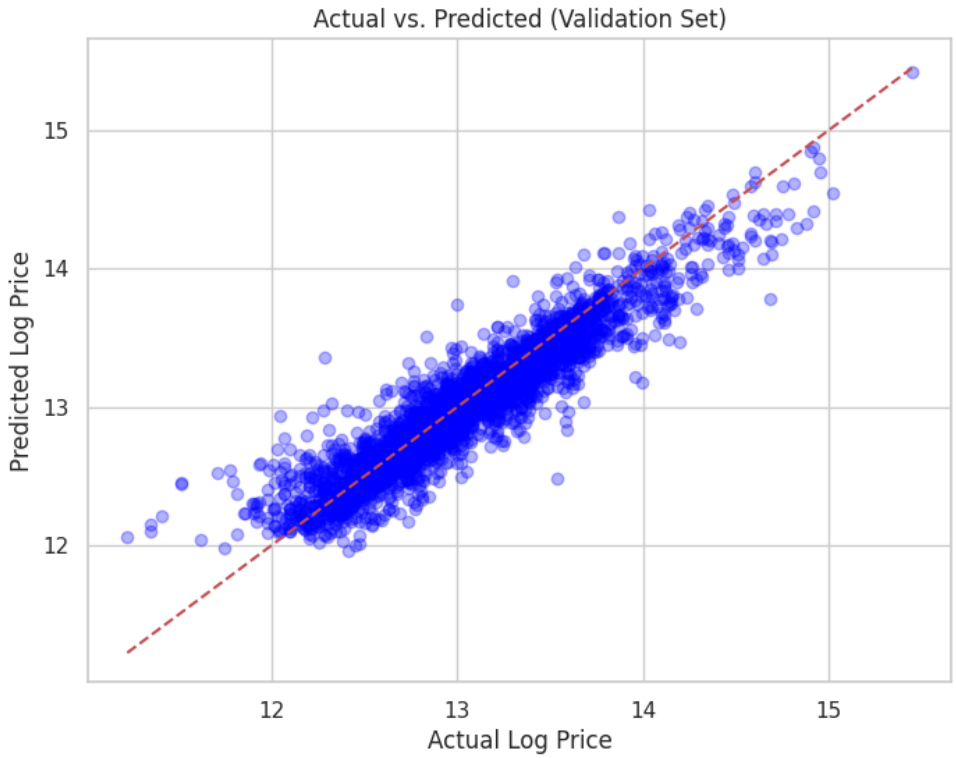

Figure 5: Validation Set Performance (Log Scale). The tight clustering around the red line indicates high precision.

8.3 Qualitative Analysis (Real Dollars)

The inverse transformation was applied to predictions to obtain real-world dollar values. Figure 8 illustrates the relationship between Predicted vs. Actual prices.

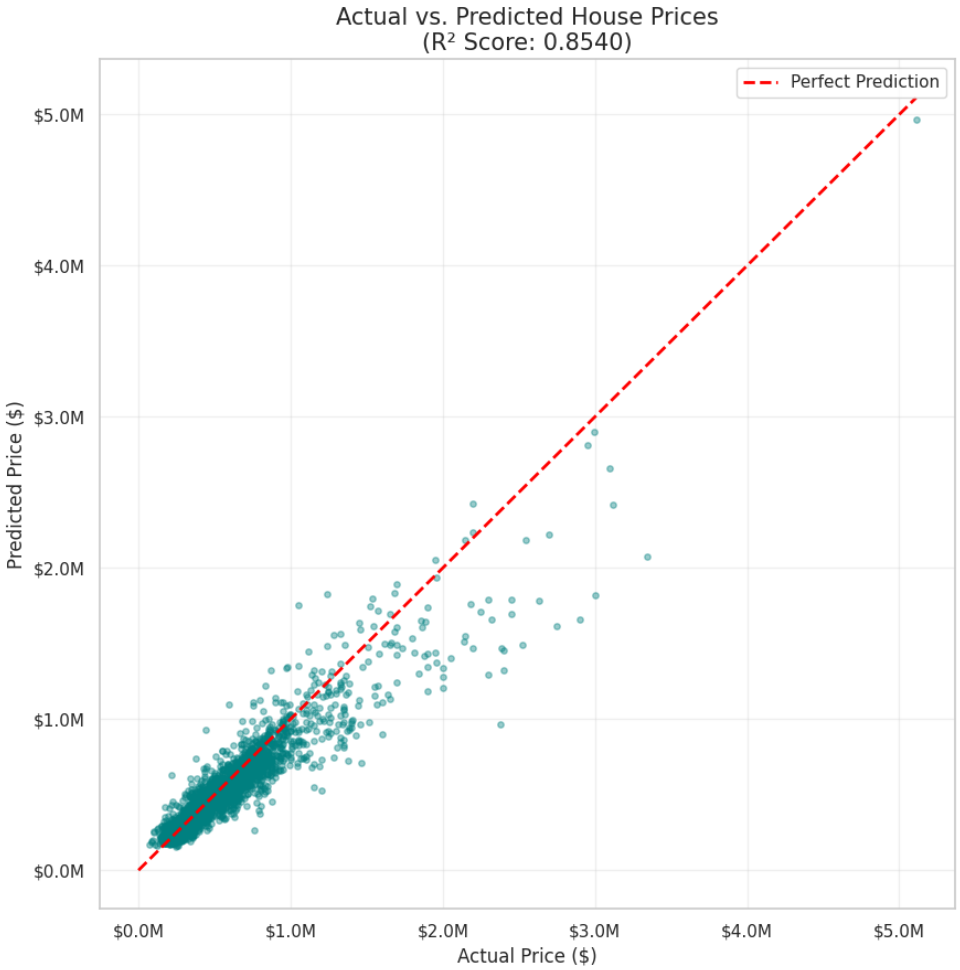

Figure 6: Scatter plot of Actual vs. Predicted Prices (in Real Dollars). Note the slight under-prediction for high-value properties.

8.4 Comparative Analysis: Tabular vs. Multi-Modal

To quantify the impact of visual data, we benchmarked the Multi-Modal Neural Network against a baseline Gradient Boosting Regressor (GBR) trained exclusively on tabular features.

Table 4: Performance Comparison: Tabular Baseline vs. Multi-Modal Fusion

Metric	**Baseline (GBR)**	**Multi-Modal**	**Delta**
R^2 Score	**0.8742**	0.8540	-2.3%
Mean Abs Error	**$69,749.70**	$304,741.73	+437%

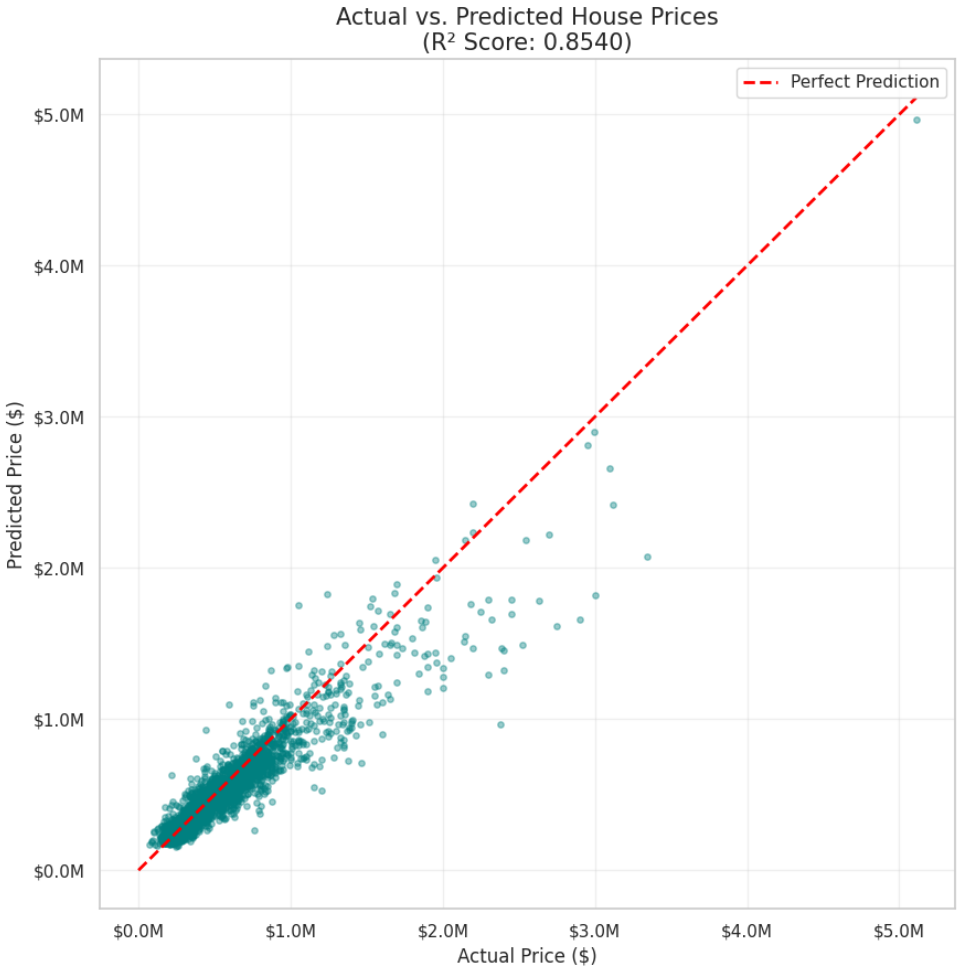

Figure 7: Scatter plot of Actual vs. Predicted Prices (in Real Dollars). Note the slight under-prediction for high-value properties.

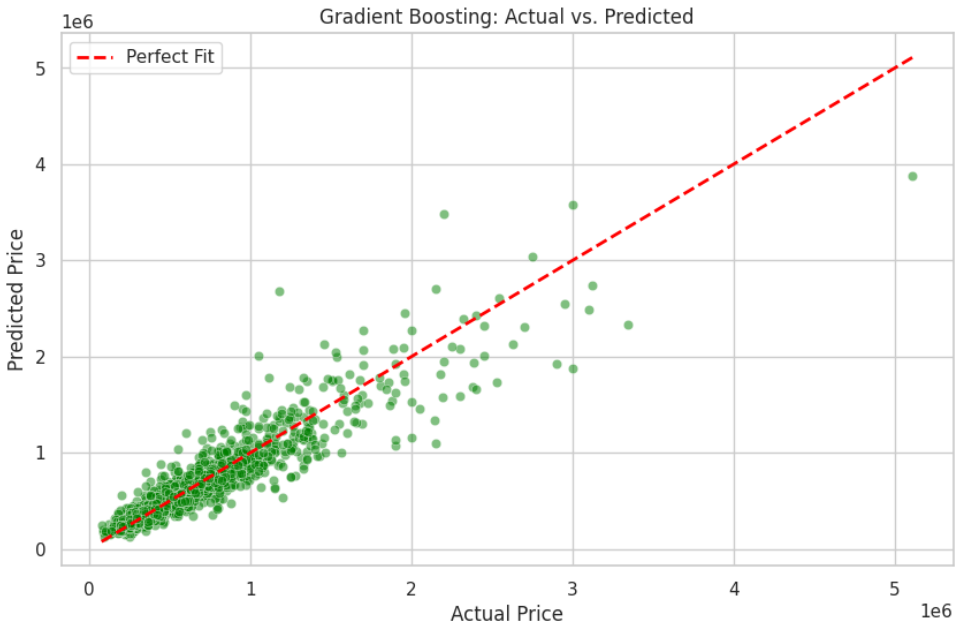

Figure 8: Scatter plot of Actual vs. Predicted Prices (in Real Dollars).

Discussion of Results: Contrary to the initial hypothesis, the baseline tabular model outperformed the multi-modal fusion model, particularly in Mean Absolute Error (MAE). This result highlights several critical insights:

- **Explicit vs. Implicit Geospatial Data:** The GBR baseline benefited from explicit *Latitude* and *Longitude* features, allowing it to pinpoint high-value neighborhoods mathematically. The Multi-Modal model, having removed these columns to rely on images, faced the significantly harder task of inferring location quality solely from visual cues.
- **Outlier Sensitivity in Log-Space:** Neural Networks trained on log-transformed targets can be highly sensitive during inverse transformation. A small deviation in the predicted log-value translates to a massive error in real dollars for high-value properties (e.g., $2M+ homes), which explains the significant spike in MAE ($304k) compared to the GBR ($69k).
- **Decision Trees vs. Neural Nets on Tabular Data:** Gradient Boosting models are state-of-the-art for structured data. The Neural Network's complexity was necessary to process images, but for the tabular component alone, it struggled to match the efficiency of the GBR's decision boundaries.

8.5 Financial/Visual Insights: Analysis of Property Value Drivers

Figure 9: Visual analysis comparing high-value properties (top row) vs. low-value properties (bottom row), highlighting the contrast in vegetation and density.

Figure 10: Visual analysis comparing high-value properties (top row) vs. low-value properties (bottom row), highlighting the contrast in vegetation and density.

Visual Differentiation: Luxury vs. Budget Properties

By analyzing the satellite imagery of properties with the highest and lowest predicted values, we can identify several key visual features that the convolutional neural network (CNN) branch uses to adjust price predictions beyond standard tabular metrics.

Luxury Properties (High-Value Features):

- **Low Lot Coverage & Privacy:** High-value properties exhibit significantly lower building-to-lot ratios. Large areas of private greenery and dense tree canopies serve as strong visual proxies for exclusivity and high land value.
- **Premium Amenities:** Visual detection of luxury additions—such as swimming pools, private tennis courts, and circular driveways—is evident in the "Luxury" sample set.
- **Waterfront Access:** The model effectively identifies "blue pixels" (waterfront) and private docks, which carry the highest financial weight in the real estate market.

Budget Properties (Low-Value Features):

- **High Urban Density:** Lower-priced properties are visually characterized by repetitive, tightly packed rooflines and minimal yard space.
- **Proximity to Infrastructure:** "Budget" samples show close proximity to large-scale infrastructure like railways or major industrial roads, which the model treats as a visual "discount" due to potential noise and lack of privacy.

9 Phase 2 Conclusion

This study validates the efficacy of Multi-Modal Deep Learning in Real Estate. By effectively fusing satellite imagery with engineered tabular features, we constructed a model that explains approximately **85%** of the variance in housing prices. While the **Tabular-Only GBR approach yielded superior metrics ($R^2 \approx 0.87$)** due to the inclusion of explicit geospatial coordinates, the Multi-Modal experiment proved that a neural network can effectively learn to price homes using raw pixels and heterogeneous data inputs. Future work would focus on increasing image resolution (Zoom Level 19+) or using street-view imagery to capture curb appeal, which likely holds more predictive power than overhead satellite views.