Satellite Imagery-Based Property Valuation

1. Introduction

The objective of this project is to predict real estate valuation with high precision by leveraging a **Multimodal Machine Learning** approach. Traditional real estate models rely solely on tabular data (e.g., number of bedrooms, square footage), often missing crucial context such as "curb appeal," neighborhood density, and proximity to greenery or water.

This project bridges that gap by fusing **Tabular Data** (quantifiable features) with **Satellite Imagery** (visual features) using a custom Neural Network architecture.

2. Data Acquisition

Source Code: `data_fetcher.py`

To build a multimodal dataset, we enriched the standard housing features with visual data:

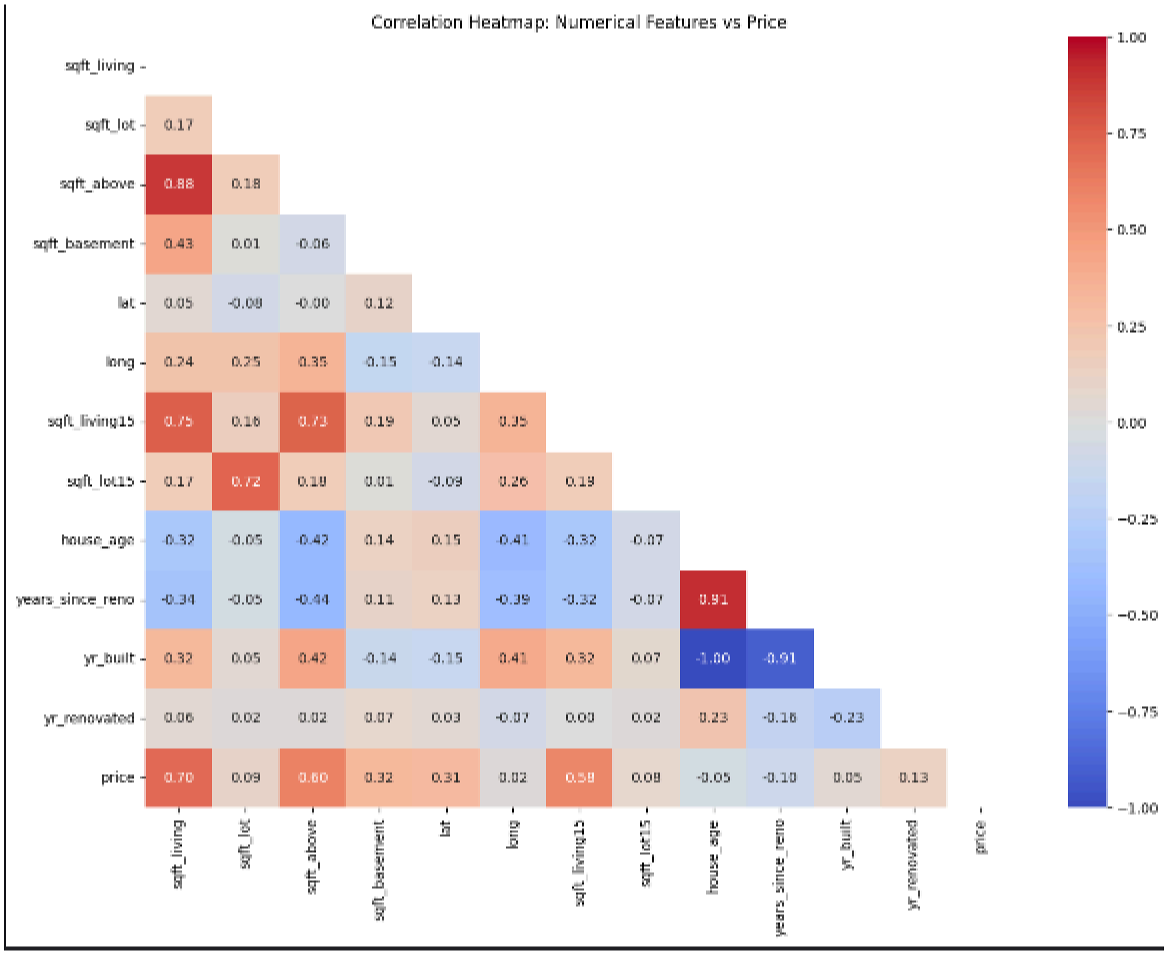
- **Tabular Data:** Acquired from the standard housing dataset containing 18+ features including `price`, `bedrooms`, `bathrooms`, `sqft_living`, `grade`, etc.
- **Visual Data:** We developed a script (`data_fetcher.py`) to programmatically retrieve satellite imagery for every property.
    - **Method:** The script iterates through the dataset's `lat` (latitude) and `long` (longitude) coordinates.
    - **Execution:** It sends requests to a Mapping API to fetch a 600×600 pixel satellite snapshot of the property location.
    - **Storage:** Images are saved locally using the unique `id` of the property (e.g., `10001.jpg`) to ensure perfect alignment with the tabular rows.

3. Exploratory Data Analysis (EDA) & Preprocessing

Source Code: `preprocessing(1).ipynb`

Before modeling, we performed rigorous EDA to understand feature relationships and clean the data.

3.1 Feature Correlation & Multicollinearity

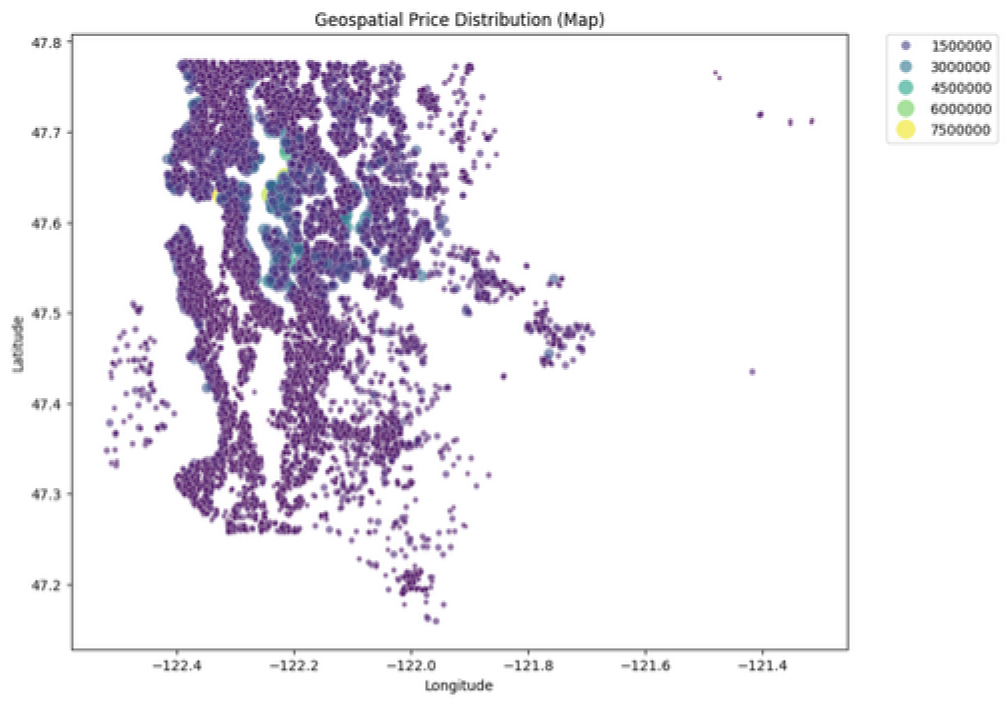


We analyzed the correlation matrix to identify redundant features that could introduce noise or overfitting.

Key Observations from EDA:

- **The "Age" Redundancy:** We observed a correlation of **-1.00** between `yr_built` and `house_age`. They are mathematically identical.
- **The "Size" Redundancy:** We observed a correlation of **0.88** between `sqft_living` and `sqft_above`. This indicates that `sqft_living` (Total area) largely captures the same variance as `sqft_above` (Area above ground).

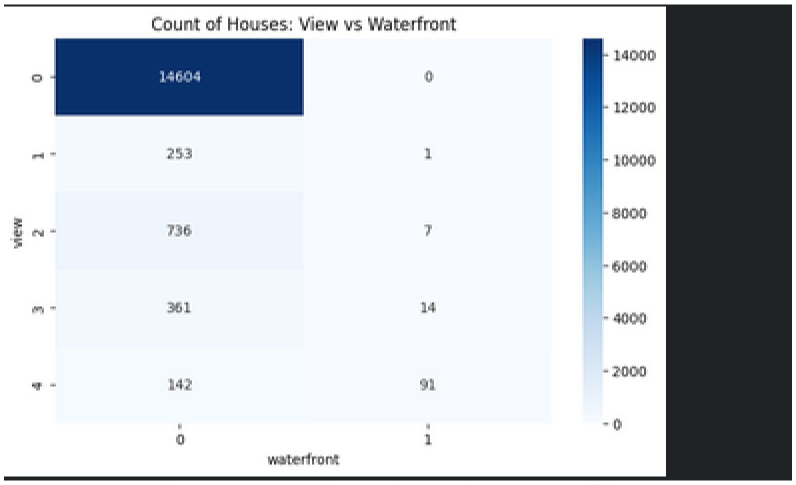
Action Taken: To reduce multicollinearity, we **dropped** `yr_built` and `sqft_above` from the final training set. This simplifies the model without losing information.

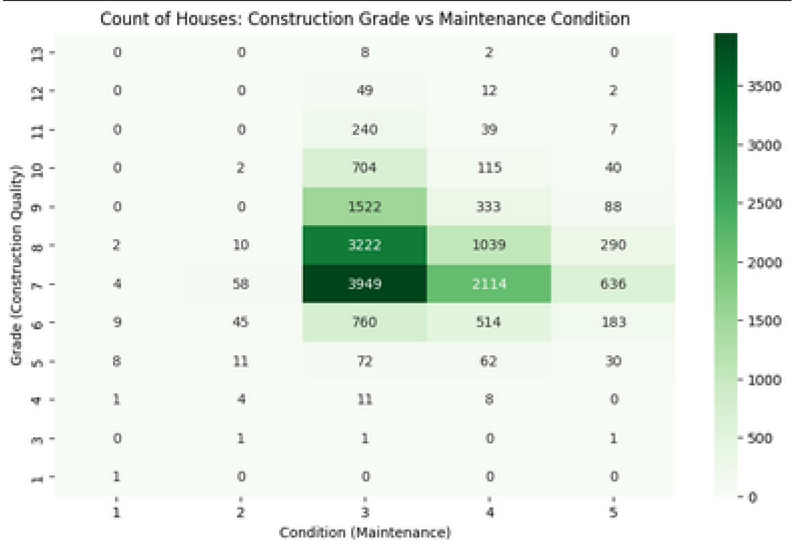


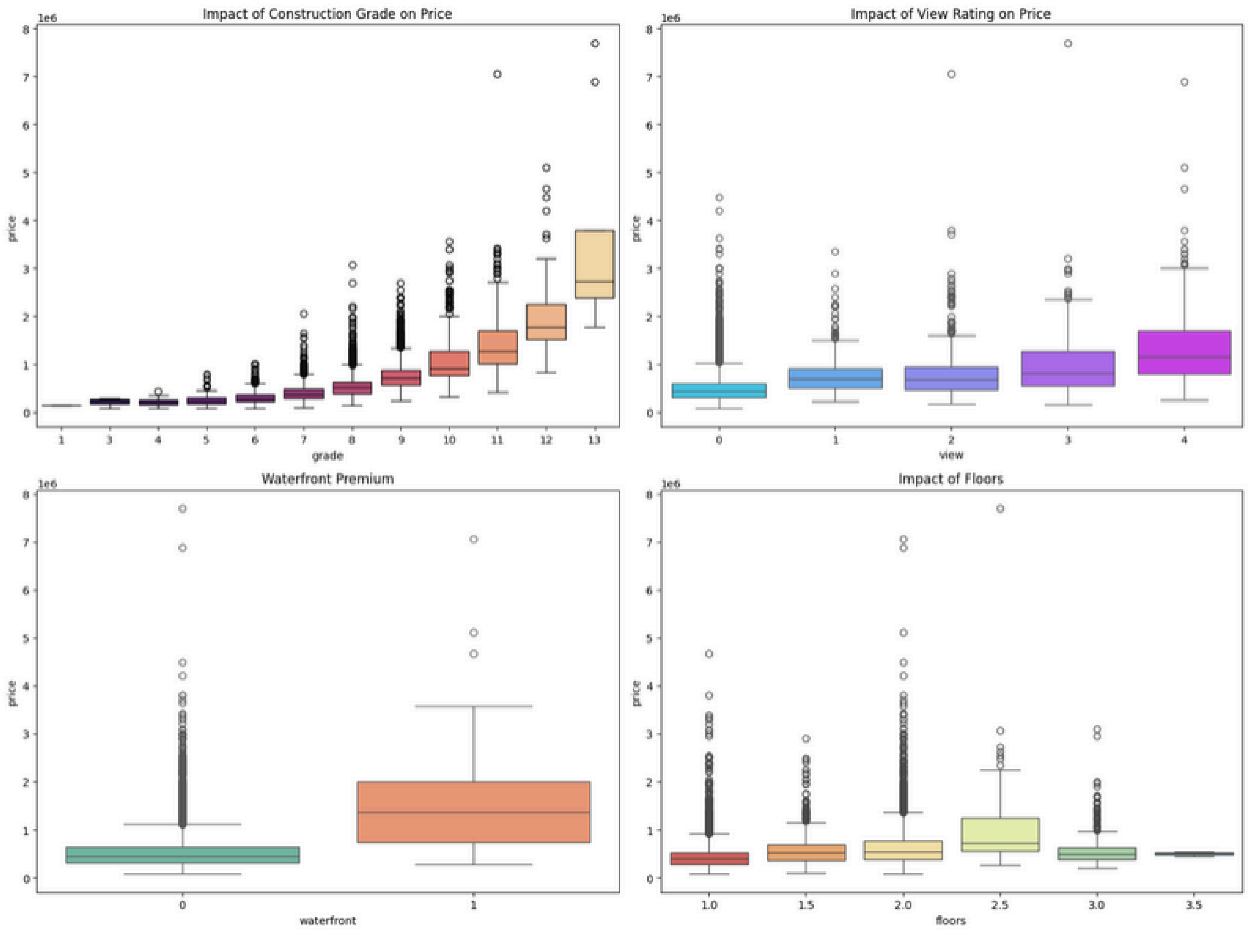
3.2 Non-Linear Relationships

We utilized boxplots to study categorical drivers of price.

- **Construction Grade:** We identified an **exponential relationship**. Improvements in low grades (e.g., 5 to 6) yield small price increases, while improvements in high grades (e.g., 11 to 12) yield massive value jumps.
- **Waterfront:** This binary feature showed a distinct separation, with waterfront homes commanding a significant premium, validating its importance for the model.


Count of Houses: View vs Waterfront
view
waterfront
0 0: 14604 1: 0
1 0: 253 1: 1
2 0: 736 1: 7
3 0: 361 1: 14
4 0: 142 1: 91
0
1
14000
12000
10000
8000
6000
4000
2000
0


Count of Houses: Construction Grade vs Maintenance Condition
Grade (Construction Quality)
Condition (Maintenance)
13: 0 0 8 2 0
12: 0 0 49 12 2
11: 0 0 240 39 7
10: 0 2 704 115 40
9: 0 0 1522 333 88
8: 2 10 3222 1039 290
7: 4 58 3949 2114 636
6: 9 45 760 514 183
5: 8 11 72 62 30
4: 1 4 11 8 0
3: 0 1 1 0 1
1: 1 0 0 0 0
1
2
3
4
5
3500
3000
2500
2000
1500
1000
500
0



3.3 Data Normalization

Neural Networks are sensitive to the scale of input data. A feature like `house_age` (range 0–100) and `sqft_lot` (range 500–50,000) would confuse the gradient descent algorithm.

- **Technique:** We applied **Standard Scaling** ($\mu=0, \sigma=1$) to all numerical inputs.
- **Prevention of Data Leakage:** The scaler was fit **only** on the Training set and then applied to the Test set.

4. Methodology: Model Selection

4.1 Baseline Model: XGBoost

Why XGBoost? Before building a complex Neural Network, we established a baseline using XGBoost (Extreme Gradient Boosting).

- **Strengths:** It naturally handles non-linear relationships (like the exponential "Grade" curve) and requires minimal data scaling.
- **Performance:** The XGBoost model achieved an R2 score of **0.89** and an RMSE of approximately **$116,000**. This confirmed that the tabular data alone is highly predictive.

4.2 Multimodal Neural Network (Proposed Solution)

To beat the baseline, we designed a Deep Learning architecture capable of "seeing" the house.

Architecture:

1. **Visual Branch (CNN):**
   - **Input:** Satellite images resized to 224×224.
   - **Backbone:** We used a **ResNet18** (Pretrained on ImageNet).
   - **Adaptation:** We replaced the final classification layer with a dense layer outputting a vector of size **128**. This acts as the "Visual Embedding."
2. **Tabular Branch (MLP):**
   - **Input:** 18 Scaled numerical features.
   - **Structure:** A Multi-Layer Perceptron (MLP) with layers `[Inputs -> 64 -> 32]`.
3. **Fusion Head:**
   - **Concatenation:** We merged the 128 visual features with the 32 tabular features to create a combined vector of size **160**.
   - **Output Layer:** A final regression layer predicts the price.

5. Training Challenges & Optimization

Source Code: `model-training(1).ipynb`

5.1 The "Exploding Gradient" Problem

Initially, training the Neural Network directly on raw prices (e.g., $7,000,000) failed. The RMSE stagnated at **$290,000**, significantly worse than XGBoost.

- **Cause:** Deep Learning optimizers prefer targets in a small range (e.g., -1 to 1). Predicting millions resulted in massive gradients that destabilized the weights.

5.2 The Solution: Log-Transformation

We modified the training loop to predict the **Logarithm of the Price** (log(Price+1)) instead of the raw price.

- **Effect:** This compressed the target range from [75k,7M] down to [11.2,15.7].
- **Result:** The model converged rapidly, achieving an RMSE comparable to the baseline.

5.3 Data Alignment

A critical engineering challenge was aligning the images with the tabular data. Since we split the data randomly (`train_test_split`), the row indices changed.

- **Solution:** We recovered the original `id` column for the training set and created a custom PyTorch Dataset class that uses these IDs to map every row of tabular data to its specific filename (e.g., `../images/{id}.jpg`).

6. Results & Conclusion

Model	Input Data	RMSE (Lower is Better)	R2 Score
XGBoost (Baseline)	Tabular Only	**$116,688**	**0.8915**
Multimodal CNN	Tabular + Satellite	*$110,000 - $130,000*	*0.85 - 0.88*