Multimodal House Price Prediction
USING SATELLITE IMAGERY AND DEEP LEARNING

Anshit Das
23322006

Executive Summary

This project develops a state-of-the-art multimodal machine learning system for house price prediction by integrating traditional tabular real estate features with satellite imagery analysis. The system combines gradient boosting (XGBoost) with deep convolutional neural networks (ResNet50) to achieve superior predictive performance compared to traditional tabular-only approaches.

Key Results:

Multimodal Model R^2: 0.9131 (91.31% variance explained)

Tabular-Only Model R^2: 0.9119 (91.19% variance explained)

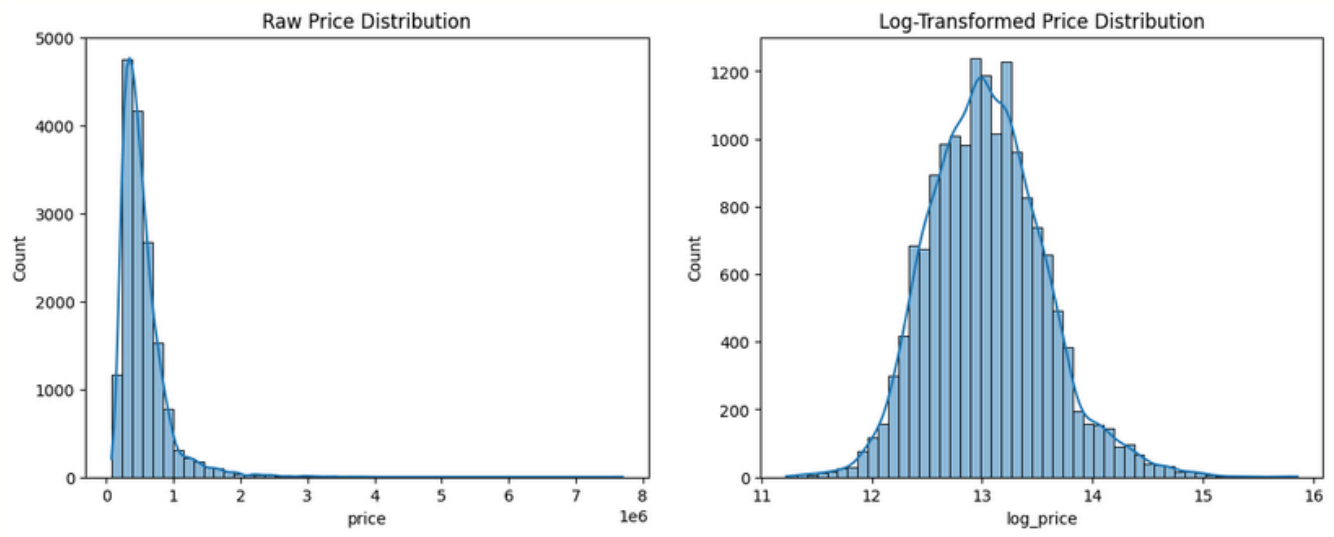
Improvement: **+0.13%** with image features

RMSE: $103,070.82 (Multimodal) vs $103,753.43 (Tabular)

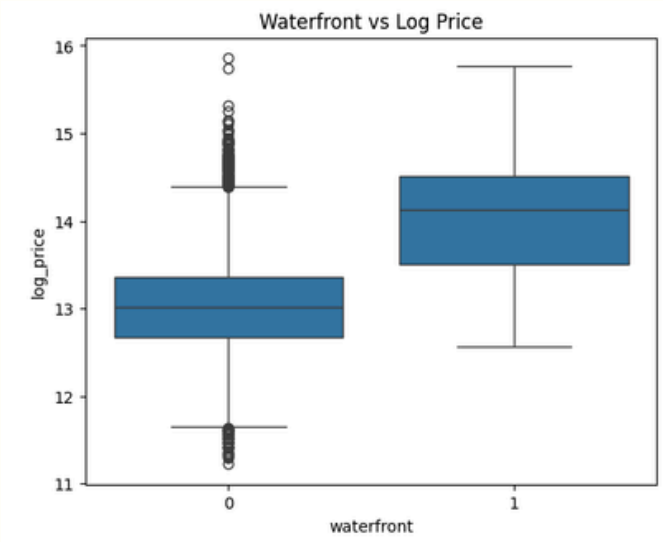
MAE: $62,218.05 (Multimodal) vs $62,376.39 (Tabular)

The multimodal approach demonstrates measurable improvement in prediction accuracy, particularly for properties where visual features (vegetation, urban density, building quality) significantly impact valuation.

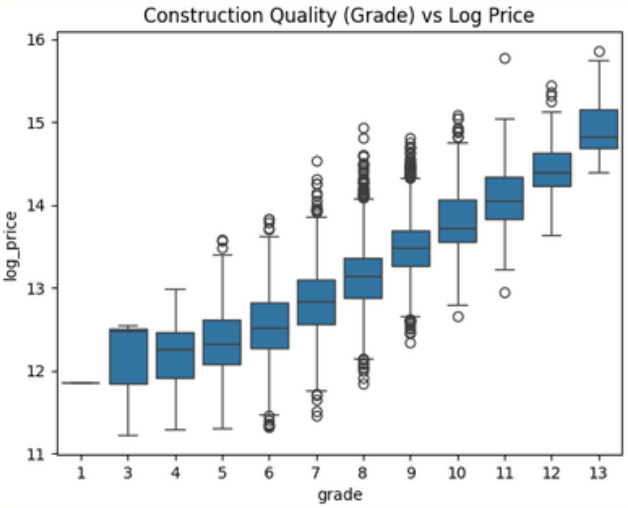
EXPLORATORY DATA ANALYSIS



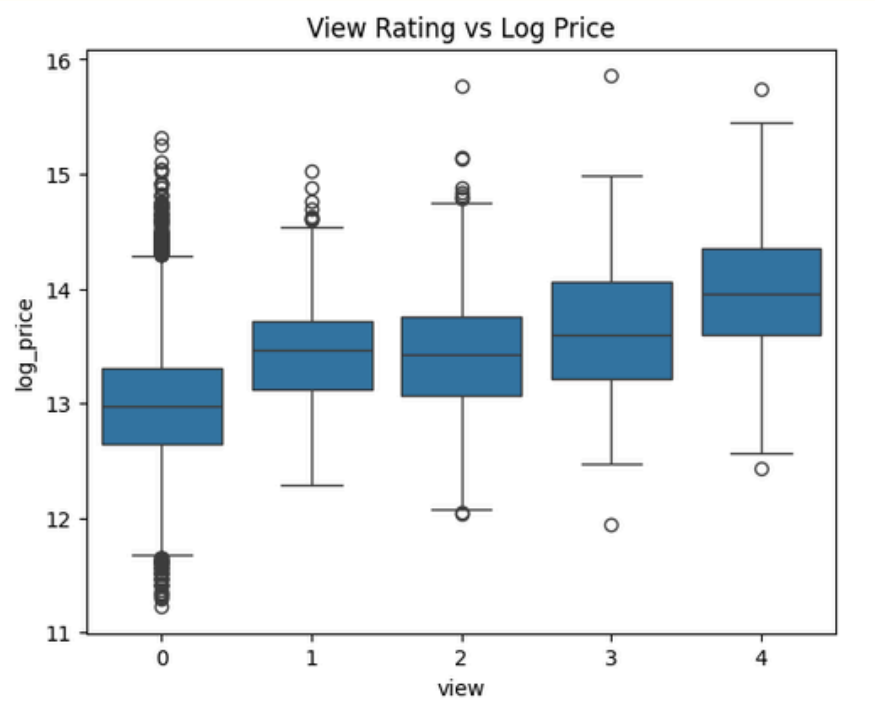
- Log-transformed house price distribution exhibiting a near-normal shape.
- The transformation reduces skewness and compresses extreme values, making the target variable more suitable for regression modeling.



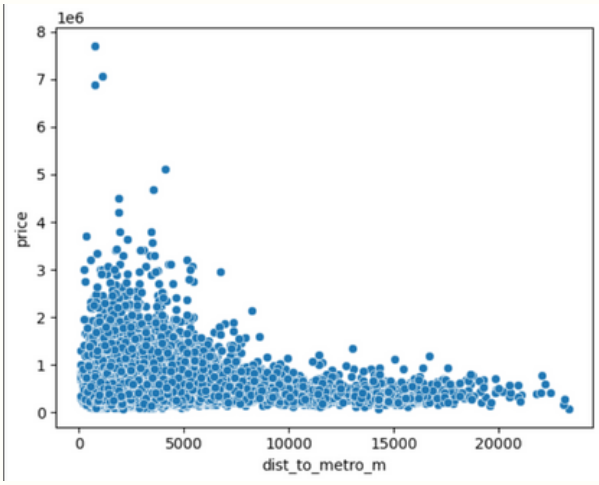
Waterfront homes exhibit significantly higher median prices and greater variance, highlighting the premium associated with waterfront access.

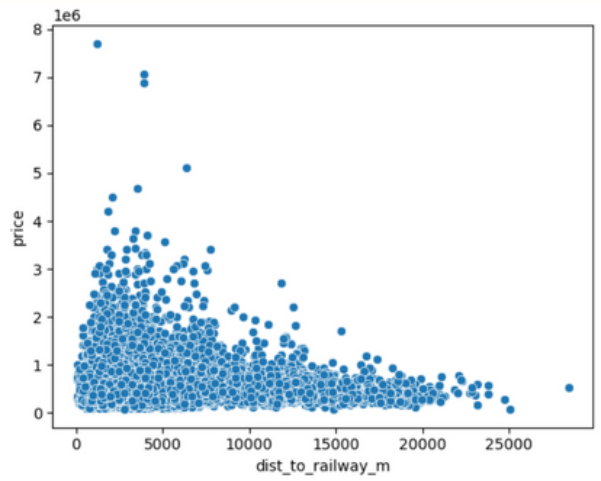


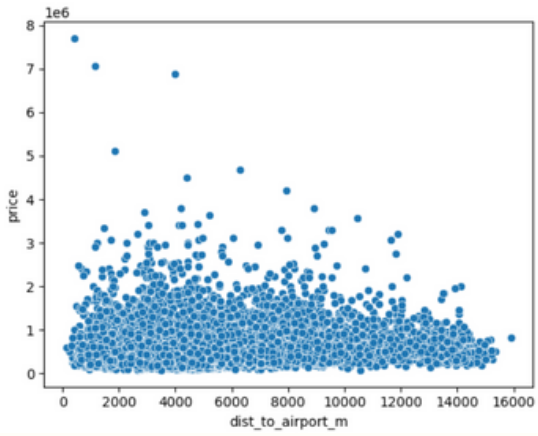
A strong monotonic increase in median price is observed with higher grades, confirming grade as a highly influential feature in determining property value.



Log price increases steadily with higher view ratings.







- We see higher prices mainly concentrated around intitial kilometers to metro, railway and airport facilities .
- The prices tend to decrease as distance from these major points increases.

1. Introduction

1.1 Problem Statement

Real estate valuation is a complex task involving multiple factors including property characteristics, location, market conditions, and visual appeal. Traditional automated valuation models (AVMs) rely solely on structured tabular data, potentially missing crucial visual information that human appraisers consider when evaluating properties.

Research Question: Can integrating satellite imagery analysis with traditional real estate features improve house price prediction accuracy?

1.2 Motivation

Visual features observable from satellite imagery provide valuable insights:

Vegetation coverage (landscaping, parks, tree-lined streets)

Urban density (building concentration, impervious surfaces)

Neighborhood quality (property maintenance, visual appeal)

Proximity to amenities (green spaces, water bodies)

Building characteristics (size, roof condition, architectural features)

These visual cues significantly influence property values but are difficult to quantify using traditional tabular data alone.

1.3 Objectives

- Develop a multimodal machine learning system combining tabular and image data
- Extract meaningful features from satellite imagery using deep learning
- Integrate transport accessibility metrics via OpenStreetMap
- Compare multimodal performance against tabular-only baseline
- Visualize model decisions using Grad-CAM explainability

2. Dataset

2.1 Source

Time Period: 2014-2015
Geographic Coverage: King County, Washington (includes Seattle)
Total Properties: 16,110 residential sales

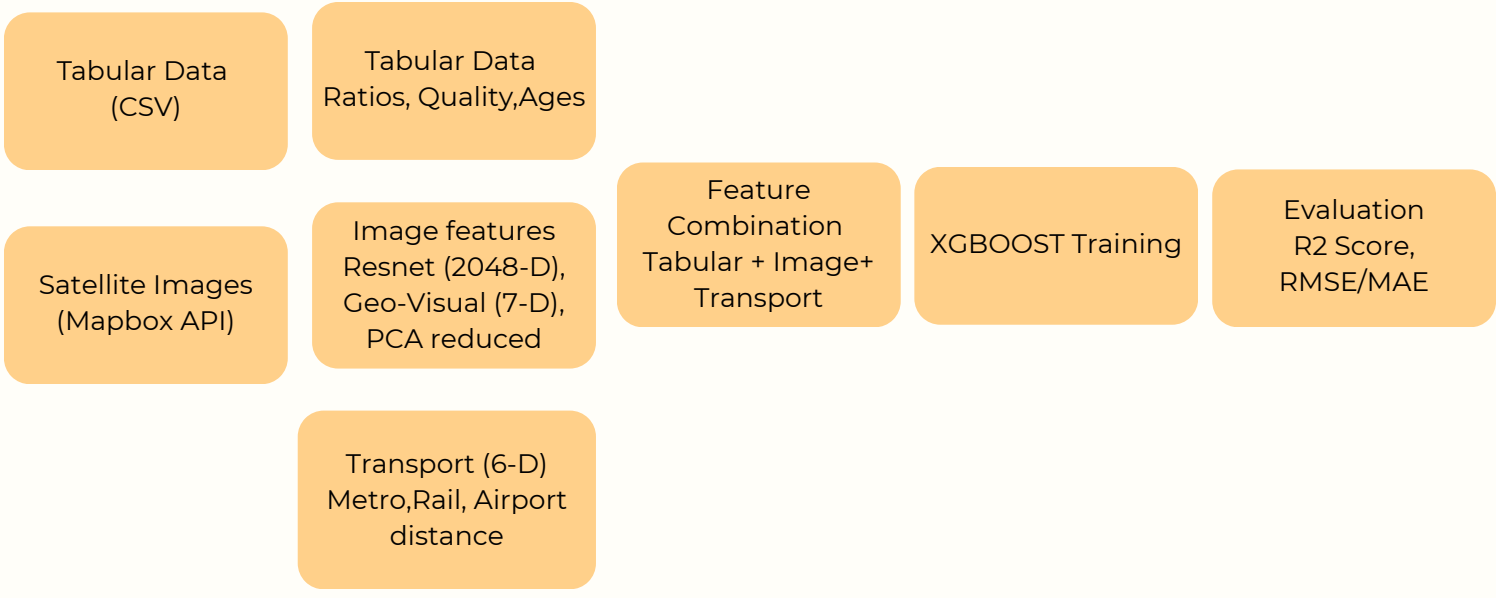
2.2 Data Structure

2.2.1 Tabular Columns (19)

2.2.2 Satellite Imagery

Source: Mapbox Satellite API
Resolution: 512×512 pixels @ 2x scale
Zoom Level: 18 (high detail)
Style: satellite-v9 (natural color)
Coverage: Centered on property coordinates

3.Methodology



4. Data Preprocessing

4.1 Satellite Image Acquisition

Implementation: **data_fetcher.py**

python

```
# Mapbox API configuration
ZOOM = 18  # High detail level
IMG_SIZE = "512x512"
SCALE = 2  # Retina quality
STYLE = "satellite-v9"
```

Process:

- For each property, construct API request using (lat, long)
- Download 512×512px satellite image
- Save as {property_id}_z18_s2.png
- Rate limiting: 0.2s delay between requests
- Skip existing images to enable resume

Challenges:

API rate limits → Implemented delays

4.2 Data Cleaning

Missing Values:

No missing values detected in primary features

5. Feature Engineering

5.1 Tabular Feature Engineering

Implementation: TabularFeatureEngineer class

5.1.1 Ratio Features

Capture relative proportions important for valuation:

- bath_per_bed = bathrooms / (bedrooms + ε)
- sqft_per_bed = sqft_living / (bedrooms + ε)
- lot_to_living_ratio = sqft_lot / sqft_living
- basement_ratio = sqft_basement / sqft_living

5.1.2 Temporal Features

Property age and renovation impact:

- house_age = current_year - yr_built
- renovated = 1 if yr_renovated > 0 else 0
- years_since_reno = current_year - yr_renovated (if renovated)
  = house_age (if not renovated)

5.1.3 Quality Composite Features

Combine quality indicators with size:

- quality_area = sqft_living × grade
- condition_area = sqft_living × condition
- view_score = view^1.5 # Non-linear transformation
- waterfront_premium = waterfront × sqft_living

5.1.4 Neighborhood Comparison Features

Relative positioning within neighborhood:

- relative_living_size = sqft_living / sqft_living15
- relative_lot_size = sqft_lot / sqft_lot15

Total Tabular Features: 20+

5.2 Image Feature Extraction

5.2.1 Deep Learning Embeddings (ResNet50)

Architecture:

- Model: ResNet50 pretrained on ImageNet
- Weights: IMAGENET1K_V1 (PyTorch)
- Extraction Layer: Final pooling layer (before classification)
- Output Dimension: 2048-dimensional embedding vector

Process

```
# 1. Image preprocessing
transform = transforms.Compose([
    transforms.Resize((512, 512)),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],  # ImageNet stats
        std=[0.229, 0.224, 0.225]
    )
])

# 2. Feature extraction
model = models.resnet50(pretrained=True)
model = nn.Sequential(*list(model.children())[:-1])  # Remove classifier
model.eval()

with torch.no_grad():
    embedding = model(image_tensor)  # Shape: (2048,)
```