Multimodal Property Price Prediction Project – Final Report

Name: Harshita Gupta
Enrollment Number: 23113065
Branch/Year: B.Tech. Civil 3rd Year
Date: 7 January 2026

Abstract

The goal of this project is to predict property prices by leveraging both satellite imagery and structured tabular data, such as area, number of bedrooms, and location. Combining these modalities allows the model to capture neighborhood characteristics along with property-specific features, providing a more accurate estimation of property values.

A multimodal deep learning model is employed for this purpose. A Convolutional Neural Network (CNN) processes satellite images, extracting spatial and neighborhood patterns, while a dense neural network processes normalized tabular features. The outputs from both networks are merged and trained together to predict property prices.

The performance of the multimodal model is compared against a tabular-only baseline, such as a Random Forest regressor. Experimental results indicate that the multimodal approach achieves higher error metrics (RMSE) andlower R² scores than the baseline, r.

This study highlights that combining satellite images with structured property data can improve predictive accuracy in real estate valuation and supports more informed decision-making for property pricing.

Introduction

Accurate property valuation is essential for buyers, sellers, and urban planners. Traditional models that rely solely on tabular data, such as area, number of bedrooms, and location, often fail to capture neighborhood characteristics that significantly influence property prices. These non-linear effects and spatial variations are difficult to model with tabular features alone.

Satellite imagery provides a visual representation of the surroundings of a property, capturing factors like greenery, road connectivity, building density, and proximity to commercial areas. These features are often difficult to quantify in tabular form but have a strong effect on property prices.

The objective of this project is to develop a multimodal model that combines tabular features with satellite images to predict property prices more accurately. By integrating visual information with structured data, the model aims to capture both property-specific and neighborhood-level factors affecting valuation.

Dataset Description

The dataset consists of two types of data: tabular features and satellite images.

Tabular Data:

- Features include: area, bedrooms, bathrooms, location, year_built, floor, and others.
- Training set contains 70% data, test set contains 15% data and validation set contains 15% data.
- Target variable: property price.

Satellite Images:

- Images are resized to 128×128 pixels with 3 color channels (RGB).
- Each property has a corresponding satellite image capturing its neighborhood context.
- Full datasets can be obtained using data_fetcher_train.ipynb and data_fetcher_test.ipynb for train.csv and test2.csv separately.

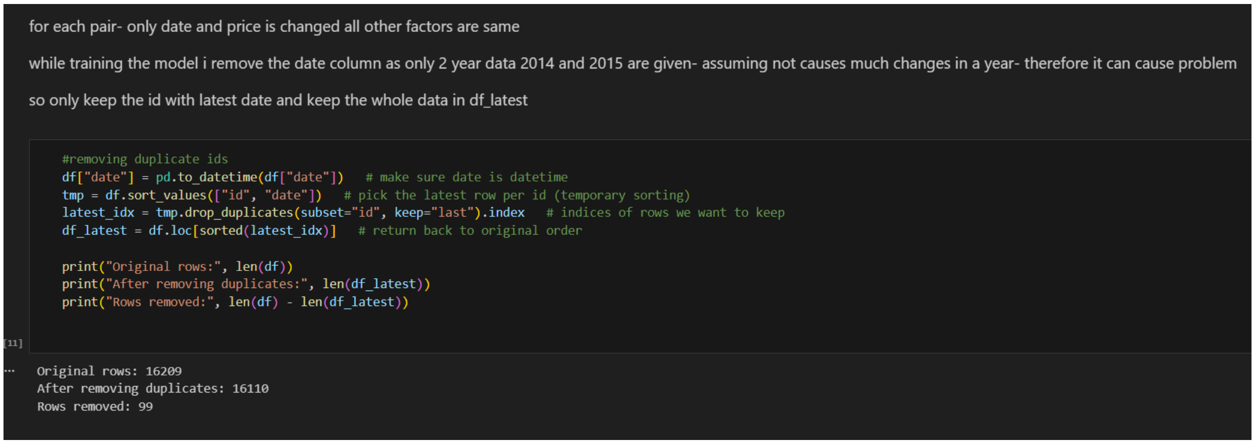
Downloading Satellite images (for both train.csv and test2.csv)

- I used Google Earth Engine to download satellite images for every property in train.csv.
- A function takes latitude, longitude, and id, searches NAIP imagery, and saves the image as: images/<id>
- The script loops through all rows and downloads images automatically.
- Missing images are handled safely (the code skips them instead of crashing).
- All the images are downloaded in a folder name images locally on google colab and then mounted on drive and then save it manually to the CDC_Project folder.
- Later, images were manually copied into the project folder because they were originally downloaded from another email account and GitHub couldn't store all of them.

Exploratory Data Analysis (EDA)

1. **Tabular Data Analysis**

    - There exist **duplicates ids** in train.csv. All duplicates are **removed**



for each pair- only date and price is changed all other factors are same

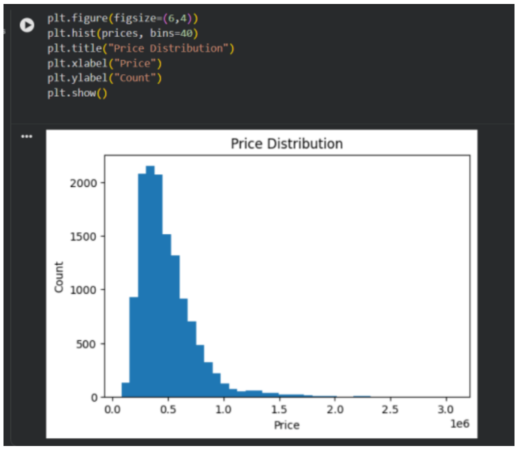
while training the model i remove the date column as only 2 year data 2014 and 2015 are given- assuming not causes much changes in a year- therefore it can cause problem

so only keep the id with latest date and keep the whole data in df_latest

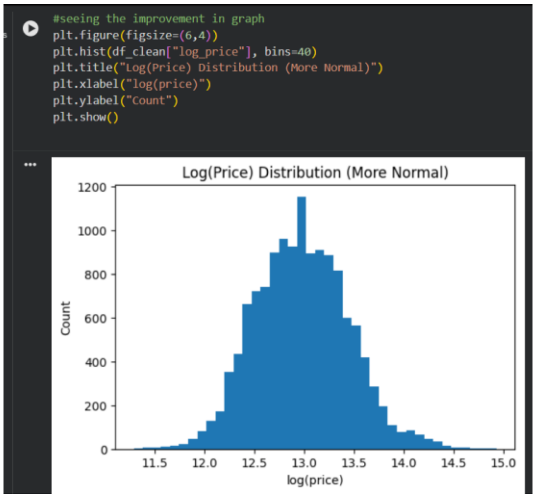
```
#removing duplicate ids
df["date"] = pd.to_datetime(df["date"])   # make sure date is datetime
tmp = df.sort_values(["id", "date"])   # pick the latest row per id (temporary sorting)
latest_idx = tmp.drop_duplicates(subset="id", keep="last").index   # indices of rows we want to keep
df_latest = df.loc[sorted(latest_idx)]   # return back to original order

print("Original rows:", len(df))
print("After removing duplicates:", len(df_latest))
print("Rows removed:", len(df) - len(df_latest))
```

```
Original rows: 16209
After removing duplicates: 16110
Rows removed: 99
```

- The distribution of property prices is **right-skewed**. Therefore, making targets log(price) instead of price should reduce the effect of very large prices (outliers) and make the distribution more normal (bell-shaped). But while training the complete model it takes too much time to get trained so **log(price) is not considered**. In the final model target is considered as price only.

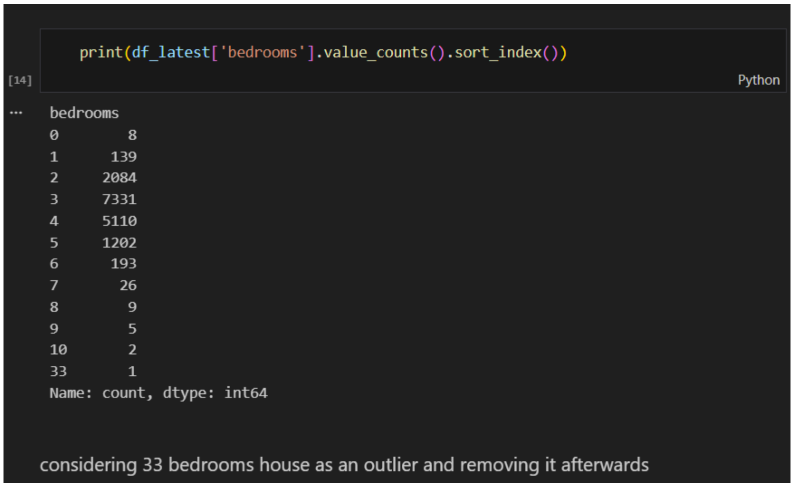


```
plt.figure(figsize=(6,4))
plt.hist(prices, bins=40)
plt.title("Price Distribution")
plt.xlabel("Price")
plt.ylabel("Count")
plt.show()
```



```
#seeing the improvement in graph
plt.figure(figsize=(6,4))
plt.hist(df_clean["log_price"], bins=40)
plt.title("Log(Price) Distribution (More Normal)")
plt.xlabel("log(price)")
plt.ylabel("Count")
plt.show()
```

- 33-bedroom outlier is removed.



```
print(df_latest['bedrooms'].value_counts().sort_index())
```

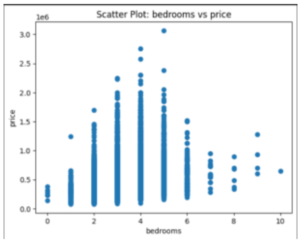
[14]

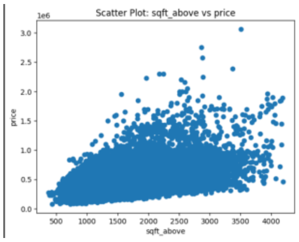
Python

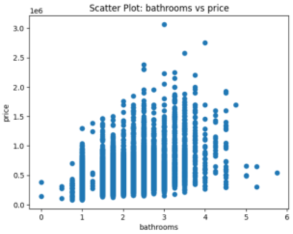
```
bedrooms
0        8
1      139
2     2084
3     7331
4     5110
5     1202
6      193
7       26
8        9
9        5
10       2
33       1
Name: count, dtype: int64
```

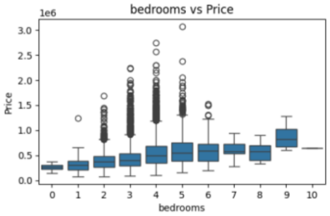
considering 33 bedrooms house as an outlier and removing it afterwards

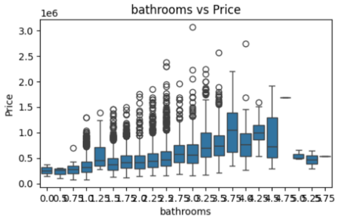
- Columns named ID, date(as only 2 year data is given ie 2014 and 2015) and zipcode(as the exact location is considered by lat and long) is not used for training.
- Area and number of bedrooms show positive correlation with price.
- Outliers exist in high-priced properties and they are removed

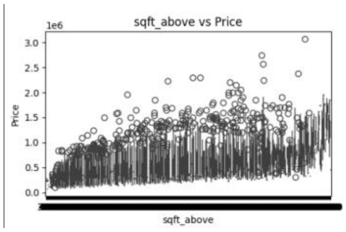


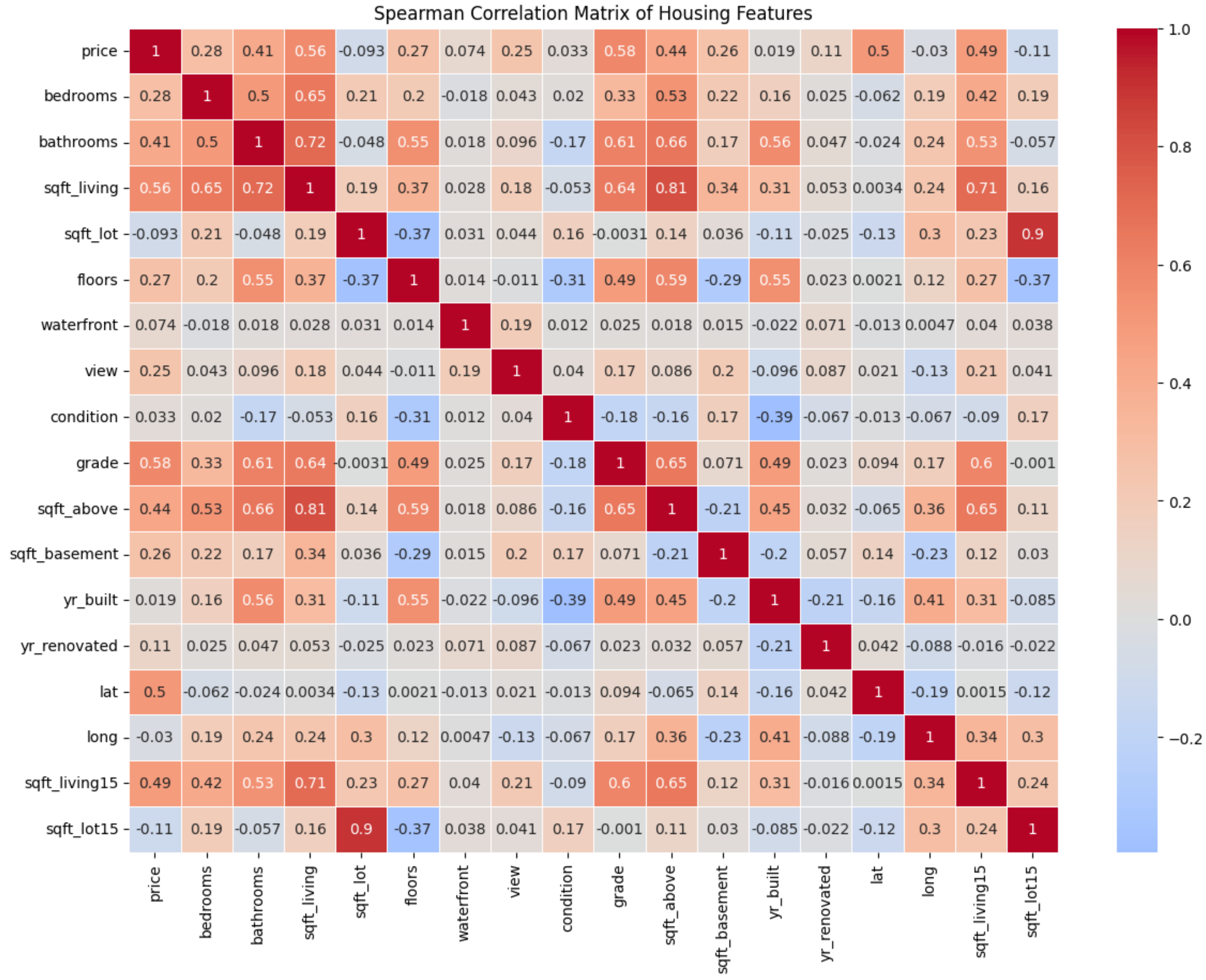












Spearman Correlation Matrix of Housing Features

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	lat	long	sqft_living15	sqft_lot15
price	1	0.28	0.41	0.56	-0.093	0.27	0.074	0.25	0.033	0.58	0.44	0.26	0.019	0.11	0.5	-0.03	0.49	-0.11
bedrooms	0.28	1	0.5	0.65	0.21	0.2	-0.018	0.043	0.02	0.33	0.53	0.22	0.16	0.025	-0.062	0.19	0.42	0.19
bathrooms	0.41	0.5	1	0.72	-0.048	0.55	0.018	0.096	-0.17	0.61	0.66	0.17	0.56	0.047	-0.024	0.24	0.53	-0.057
sqft_living	0.56	0.65	0.72	1	0.19	0.37	0.028	0.18	-0.053	0.64	0.81	0.34	0.31	0.053	0.0034	0.24	0.71	0.16
sqft_lot	-0.093	0.21	-0.048	0.19	1	-0.37	0.031	0.044	0.16	-0.0031	0.14	0.036	-0.11	-0.025	-0.13	0.3	0.23	0.9
floors	0.27	0.2	0.55	0.37	-0.37	1	0.014	-0.011	-0.31	0.49	0.59	-0.29	0.55	0.023	0.0021	0.12	0.27	-0.37
waterfront	0.074	-0.018	0.018	0.028	0.031	0.014	1	0.19	0.012	0.025	0.018	0.015	-0.022	0.071	-0.013	0.0047	0.04	0.038
view	0.25	0.043	0.096	0.18	0.044	-0.011	0.19	1	0.04	0.17	0.086	0.2	-0.096	0.087	0.021	-0.13	0.21	0.041
condition	0.033	0.02	-0.17	-0.053	0.16	-0.31	0.012	0.04	1	-0.18	-0.16	0.17	-0.39	-0.067	-0.013	-0.067	-0.09	0.17
grade	0.58	0.33	0.61	0.64	-0.0031	0.49	0.025	0.17	-0.18	1	0.65	0.071	0.49	0.023	0.094	0.17	0.6	-0.001
sqft_above	0.44	0.53	0.66	0.81	0.14	0.59	0.018	0.086	-0.16	0.65	1	-0.21	0.45	0.032	-0.065	0.36	0.65	0.11
sqft_basement	0.26	0.22	0.17	0.34	0.036	-0.29	0.015	0.2	0.17	0.071	-0.21	1	-0.2	0.057	0.14	-0.23	0.12	0.03
yr_built	0.019	0.16	0.56	0.31	-0.11	0.55	-0.022	-0.096	-0.39	0.49	0.45	-0.2	1	-0.21	-0.16	0.41	0.31	-0.085
yr_renovated	0.11	0.025	0.047	0.053	-0.025	0.023	0.071	0.087	-0.067	0.023	0.032	0.057	-0.21	1	0.042	-0.088	-0.016	-0.022
lat	0.5	-0.062	-0.024	0.0034	-0.13	0.0021	-0.013	0.021	-0.013	0.094	-0.065	0.14	-0.16	0.042	1	-0.19	0.0015	-0.12
long	-0.03	0.19	0.24	0.24	0.3	0.12	0.0047	-0.13	-0.067	0.17	0.36	-0.23	0.41	-0.088	-0.19	1	0.34	0.3
sqft_living15	0.49	0.42	0.53	0.71	0.23	0.27	0.04	0.21	-0.09	0.6	0.65	0.12	0.31	-0.016	0.0015	0.34	1	0.24
sqft_lot15	-0.11	0.19	-0.057	0.16	0.9	-0.37	0.038	0.041	0.17	-0.001	0.11	0.03	-0.085	-0.022	-0.12	0.3	0.24	1

2. Image Data Analysis

- Satellite images show variations in neighborhood density, greenery, and proximity to commercial areas.

- Images were resized and normalized before being input into the CNN.

Insights:

- Properties in greener or more connected neighborhoods tend to have higher prices.
- Visual patterns from satellite images can complement tabular features for better predictions.

Data Preprocessing

- **Missing Values:** There exists no missing value in the given data
- **Normalization:** Tabular features standardized using **Standardization** (also known as Z-score Normalization).
- **Images:** Resized to 128×128×3 and normalized to [0,1].
- **Data Split:** Training data split into training, test, and validation sets (70:15:15 ratio).

Methodology

1. **Tabular Model (Baseline)**

- Random Forest Regressor used as baseline.
- Input features: normalized tabular features.
- Output: predicted property price.

2. **Image Model (CNN)**

- Input: 128×128×3 satellite images.
- Architecture:
  - Conv2D → ReLU → MaxPooling
  - Conv2D → ReLU → MaxPooling
  - Flatten → Dense layers → Output
- Output: feature embedding representing visual neighborhood context.

3. **Multimodal Model**

- Outputs of CNN and tabular dense network are concatenated.
- Fully connected layers applied on concatenated features.
- Final output layer predicts property price.

4. **Training Details**

- Loss function: Mean Squared Error (MSE)
- Optimizer: Adam
- Metrics: RMSE, MAE, R^2
- Epochs: 10, Batch size: 32
- Hardware: CPU (done on Google Colab)

Figure to include: Model architecture diagram showing CNN + Tabular network → merged → output.

Results and Evaluation

Model	RMSE	R^2
Random Forest	94992.11	0.847
Multimodal CNN+Tabular	196494	0.4266

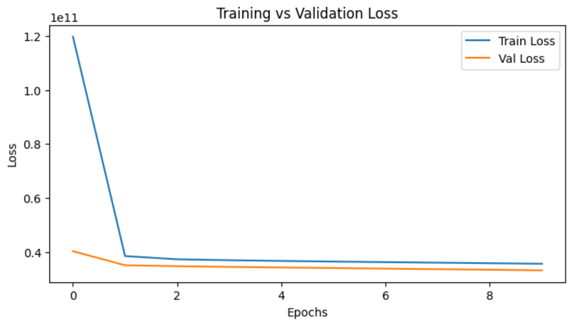
Model Comparison

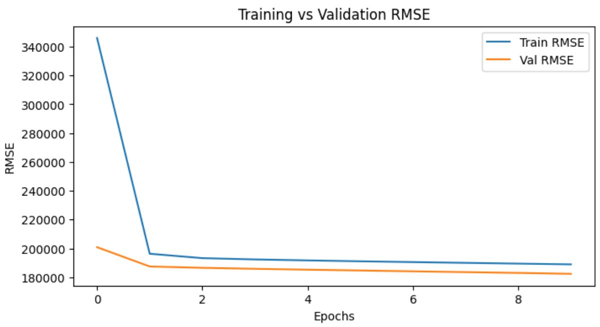
Random Forest

- Explains most of the price variation
- Very strong performance using only tabular data
- Handles non-linear relationships well

Multimodal CNN + Tabular

- Lower accuracy compared to Random Forest
- CNN may not have learned strong features from images yet
- Possibly needs more tuning, more training, or better images





Training Curves (Graphs)

Training vs Validation Loss

- Loss drops sharply in first few epochs
- Then stabilizes and becomes almost flat
- Training and validation loss are close
- Model is not heavily overfitting

- But final loss remains high, meaning predictions still have error

Training vs Validation RMSE

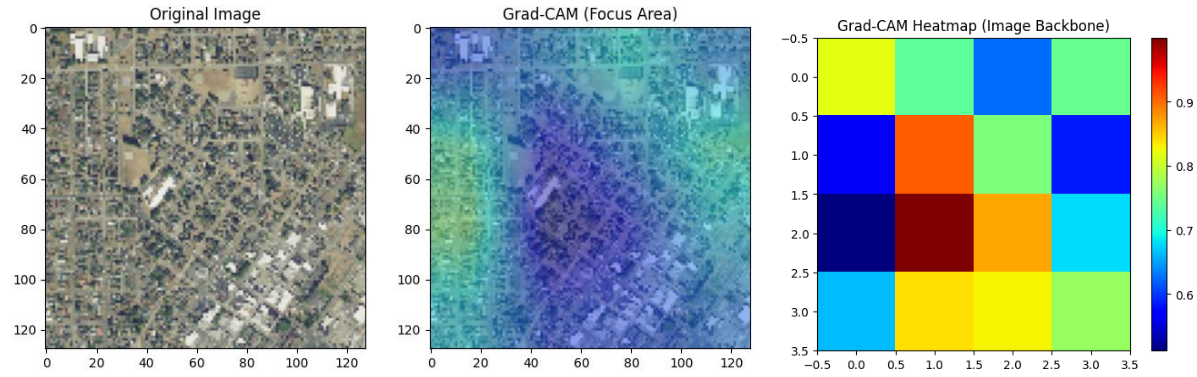
- RMSE decreases quickly at the start
- Then slowly improves over epochs
- Gap between training and validation RMSE is small
- Model generalizes reasonably well
- Still needs improvement to reduce final error

Why the Multimodal (CNN + Tabular) Model Did NOT Perform Better

- Deep learning models require significantly more data than classical models. Due to limited training images, the CNN was unable to learn strong visual representations.
- Several important price features are not visible in satellite imagery, reducing the benefit of adding the image branch.
- Low-resolution satellite crops limited the spatial context available to the CNN.
- The CNN backbone was likely under-trained and not fully optimized.
- The feature fusion stage may have diluted strong tabular signals instead of strengthening them.

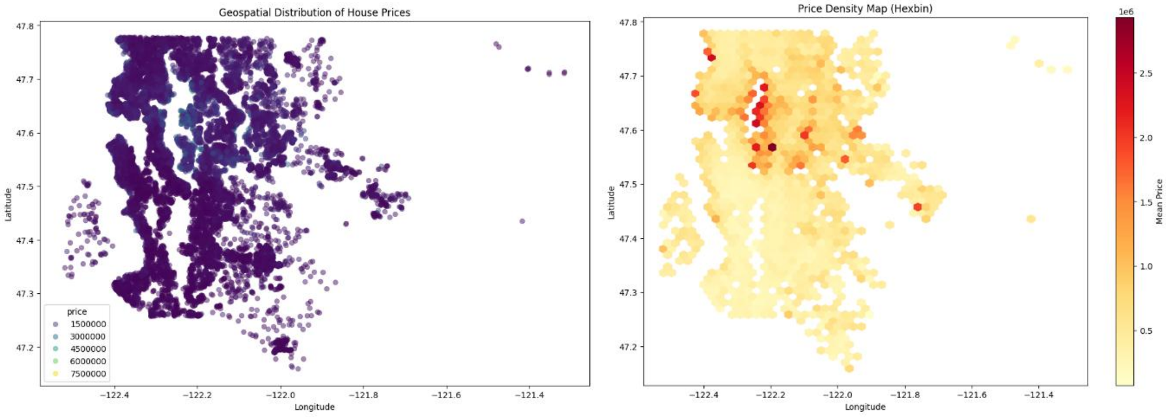
Grad-CAM: Understanding What the CNN Sees

- To understand how the image model (ResNet50 backbone) makes decisions, I applied **Grad-CAM (Gradient-Weighted Class Activation Mapping)**.
- Grad-CAM helps visualize **which parts of the satellite image the CNN focuses on** while predicting house price.
- I extracted the last convolution layer (conv5_block3_out) and generated a heatmap over the original image.
- Warmer colors (red/yellow) indicate **high attention regions**, while cooler colors (blue) indicate less important regions.
- From the Grad-CAM results, the model mainly focused on:
    - nearby roads and connectivity
    - surrounding building density
    - open spaces or greenery
- This confirms that the CNN is learning **contextual neighborhood information** instead of random noise.



Geospatial Analysis (Latitude–Longitude Maps)

- To analyze spatial distribution of prices, I created geospatial visualizations using latitude and longitude.
- Scatter plots colored by price show how house prices vary geographically.
- Higher prices cluster nearby:
  - central urban regions
  - regions with road accessibility
  - dense neighborhoods
- Lower prices are generally seen:
  - in peripheral/outer areas
  - low-density regions
- Hexbin density maps help identify **high-price hot-spots** and **low-price clusters** visually.
- These plots support that **location strongly influences house price**, which also explains why tabular features performed very well.



Conclusion

- A multimodal approach was developed combining **tabular property features** and **satellite images** to predict house prices.
- The **Random Forest (tabular-only)** model performed strongly and explained most of the variation in prices (high R^2 and low RMSE).
- The **Multimodal CNN + Tabular model** did not outperform Random Forest because the image branch could not extract strong, useful features with the available data.
- Important price factors such as age of building, renovations, interior quality, and legal factors are **not visible from satellite images**, reducing the benefit of image data.

  Llow-resolution crops, and under-trained CNN reduced visual learning capability.
- Grad-CAM results confirmed that the image model mainly looked at roads, greenery, and surrounding building density, which adds context but was not enough to beat the baseline model.
- Geospatial analysis (lat–long maps) showed that **location is the strongest driver of price**, supporting why tabular models worked well.
- Overall, the project demonstrates that:
  - multimodal learning is promising,
  - but requires more data, better resolution, and advanced tuning to surpass traditional ML models.
- Future improvements such as transfer learning, log-price training, better fusion layers, and larger datasets may allow multimodal models to outperform classical approaches.