Satellite Imagery-Based Property Valuation Report

Name : Killada Dhanur Vishnu

Enrol : 22118037

College : IIT Roorkee

Opportunity provided by : CDC X YHills Open Projects 2025

1. Introduction:

The project outline is to evaluate the price of the property based on the two types of data context (both visual and tabular data) and observe which model is better at making better predictions of the prices. Visual data is directly not provided, but the latitude and longitude coordinates were provided through which, expected to construct a pipeline to fetch all the required satellite images which are used for the Multimodal training later. After comparing the performance of the models which are separately trained with only tabular data and tabular data combined with visual context, the results are properly explained at the end.

2. Fetching the Satellite image data

Since the given training and testing dataset contained two columns : 'lat' and 'long', we are expected to construct a pipeline to accurately download the image from the respective position and save it locally in the computer. Suggested APIs were Google Maps API, Mapbox API, and so on, but since I lacked proper bank details to create an account to access the API key, I had to use a different alternative to extract the images. I have used the ESRI website, which provides good satellite images without the requirement of logging in or submitting bank details.

I have used a zoom level of 18, which is sufficient enough to capture neighborhood context while also maintaining decent resolution during training. The process of downloading 16,110 training images were :

Collect the 'lat' and 'long' values for each sample → create an url with the mentioned zoom and positional coordinates → request those images from the server with a limited sleep time → once there is a response, save those images at the mentioned directory

The complication is that ESRI doesn't directly take latitude and longitude coordinates to get the images, it receives the x_tile and y_tile positions and then provides the images. I had to convert the latitude and longitude coordinates into tile coordinates, from which the ESRI website provided 16110 training images and 5404 testing images.

```
# Convert lat/lon → XYZ tile
tile = mercantile.tile(lon, lat, ZOOM)

# Get tile from the URL with set parameters
url = TILE_URL.format(z=tile.z, x=tile.x, y=tile.y)
```

Using the PIL for Image processing, all the images were successfully extracted and downloaded for training and testing.

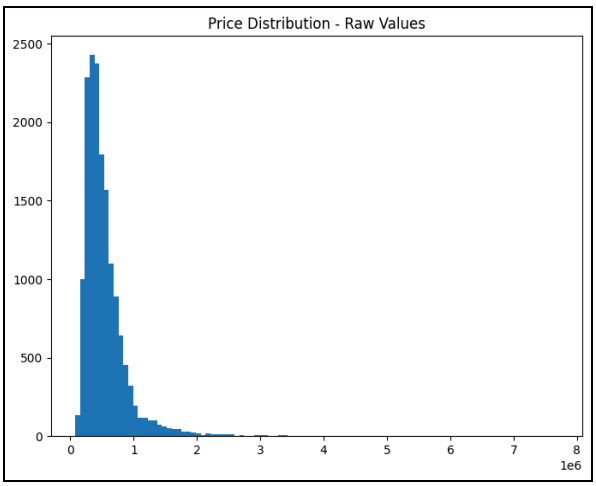
3. EDA and Preprocessing

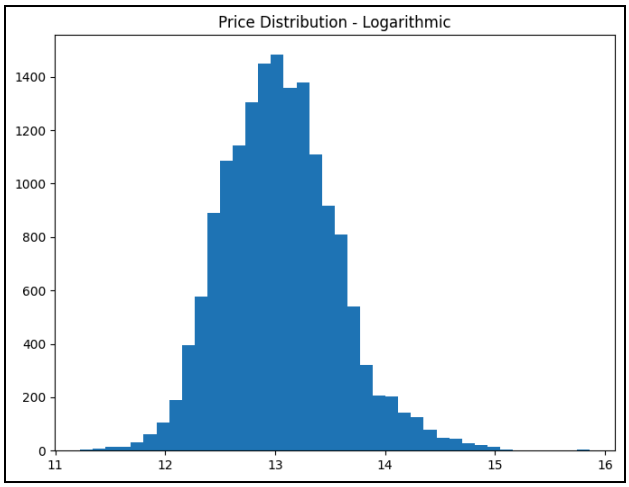
Now let's have a look at what the data is displaying. The training dataset consists of about 21 columns such as : "id", "date", "price", and many house characteristic features like : number of floors, number of bathrooms, view score, grade score and so on. Using the isnull() function, I checked if the data consisted of any Null values, but no Null values were found in the dataset.

Then I checked for unique ids and found out that there are 16110 unique ids, but the total number of samples is 16209, meaning 89 sample's ids were redundant. I have removed the duplicates and kept only the ids which are sorted by the recent date.

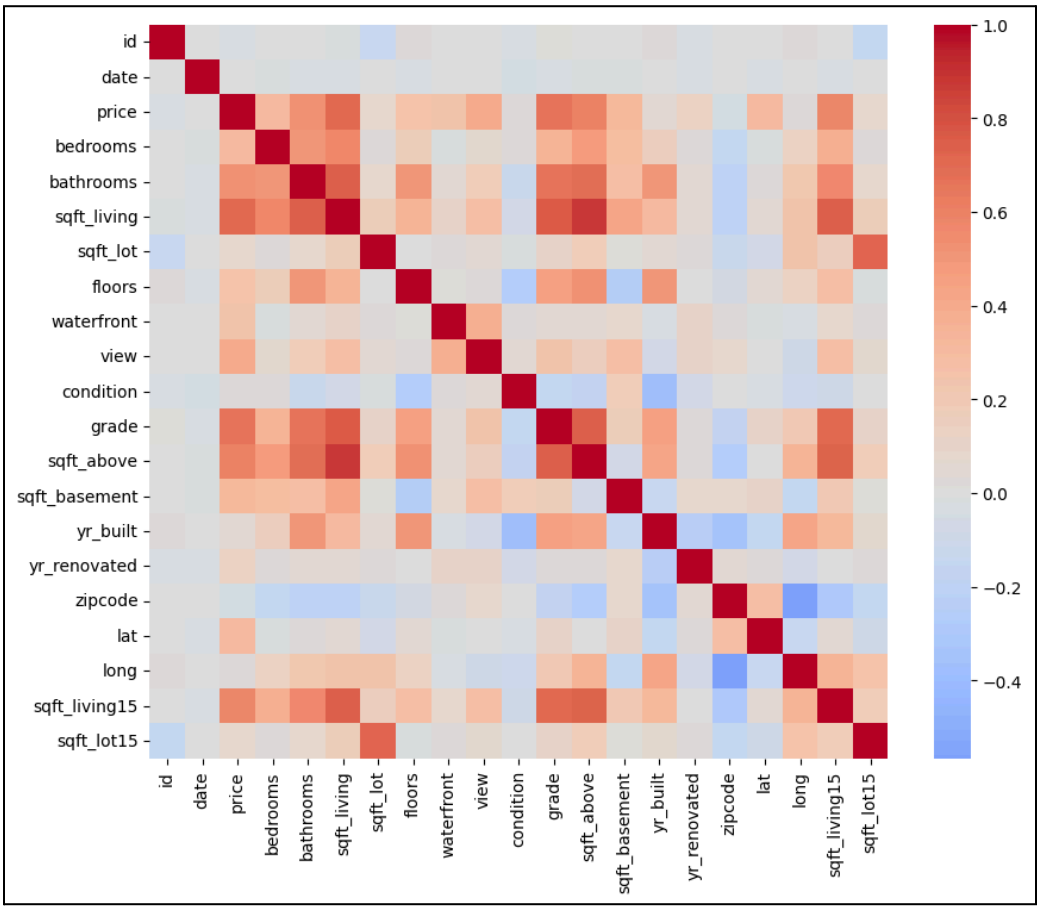
The "date" column and "id" column are not of the correct data type, so i converted the data type of "date" into 'datetime' and for "id" converted it into 'str' so that there won't be any value conversions.

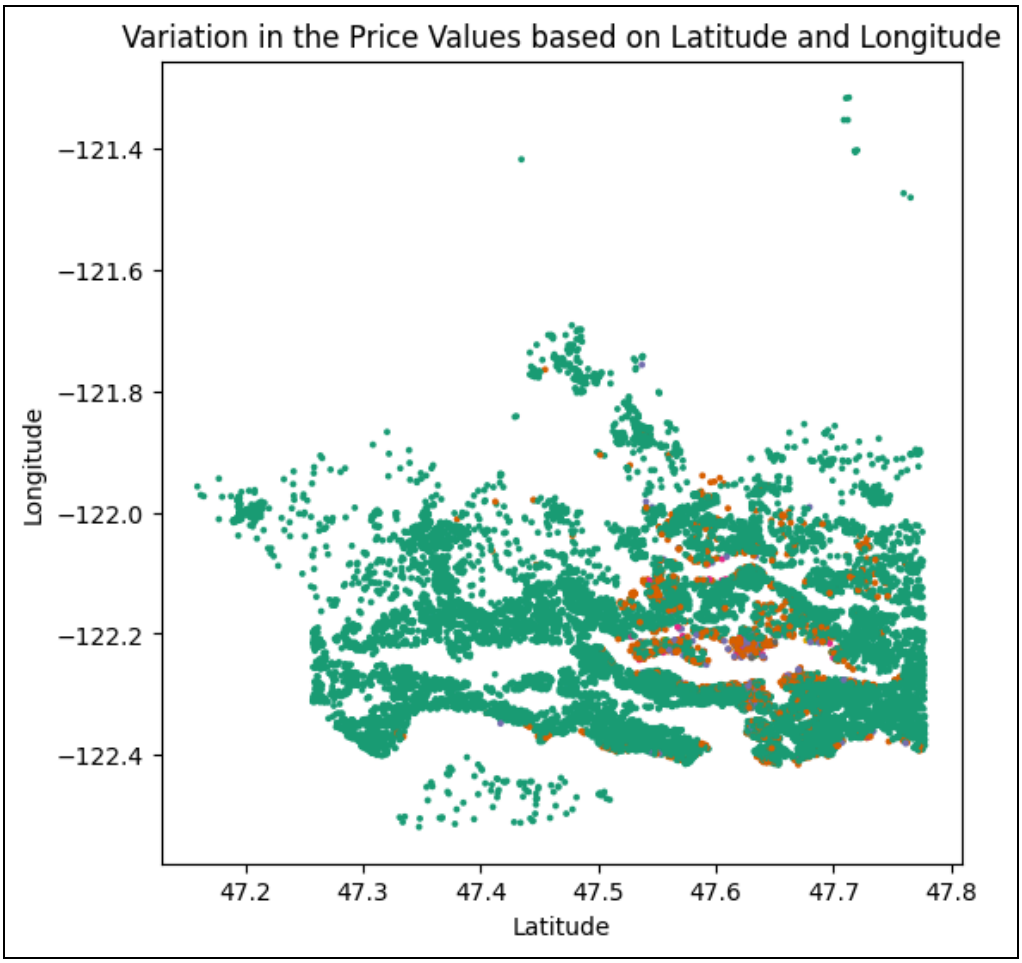
If we plot the frequency/count of the house according to their prices, we get a histogram that is right skewed in nature. This indicates the majority of the houses are on the medium and the lower end of the price range, compared to the very expensive houses which can be located at the very end of the tail. To avoid skewness in predictions also (since the dataset is imbalanced as well in terms of price), we apply logarithmic smoothing (np.log1p) so that the price values appear symmetrical, which are going to be fed into the model later.



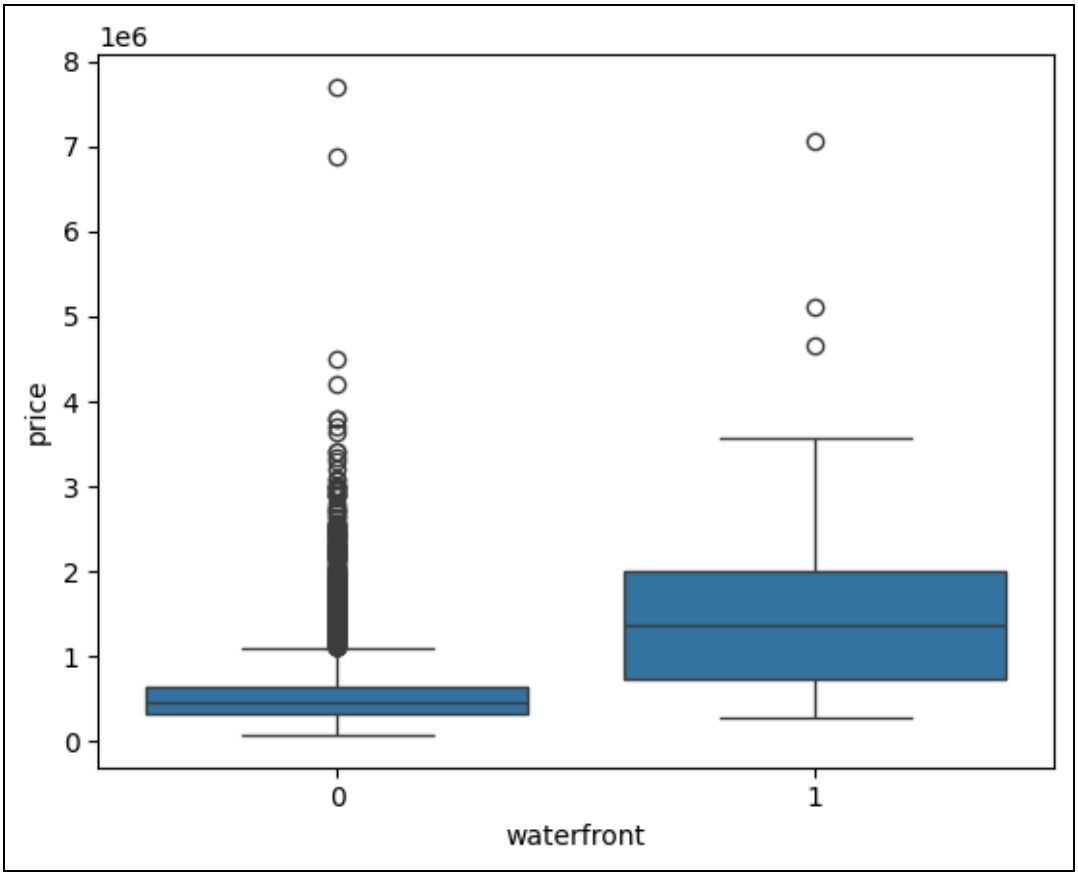


Next, we will look at the correlation of the features, to understand which features are highly correlated and which ones are mainly affecting the price column. If we look at this heatmap, we can see that the price column is mainly correlating with features like sqft_living, view, grade, sqft_living15 and so on. The “id” and the “date” column can be ignored for now, since they are going to be discarded after feature engineering for training purposes.





If we look at the previous plot (latitude vs longitude), we can see that the majority of the points are green (lower or medium price) and there are some points in orange/red color which indicate houses on the expensive range. If we consider this as a map, we can see that the empty white spaces are resembling that of water bodies. Houses which are closer to water bodies are having higher prices. And if we observe the density of the green and orange dots, green dots are more tightly packed, whereas the orange dots are somewhat spaced well. This can also be an underlying factor for price value evaluation.



This boxplot clearly explains that the majority of the higher price houses are having waterfront present within the boundary of their houses, leading to a higher IQR (observed in the plot). Whereas most of the houses which do not have any waterfront are on the lower end of the prices, but there are some exceptional outliers which have higher prices even if they don't have any waterfront. Possibility for more underlying features can be present which are affecting the price values.

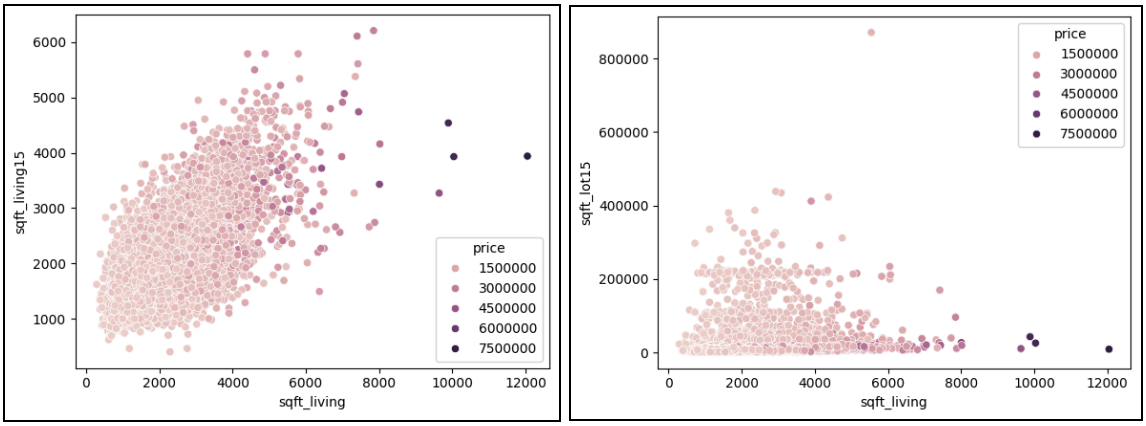
I tried to visually observe for any clear differences between the lower and higher priced houses, by sampling 9 images of each range.

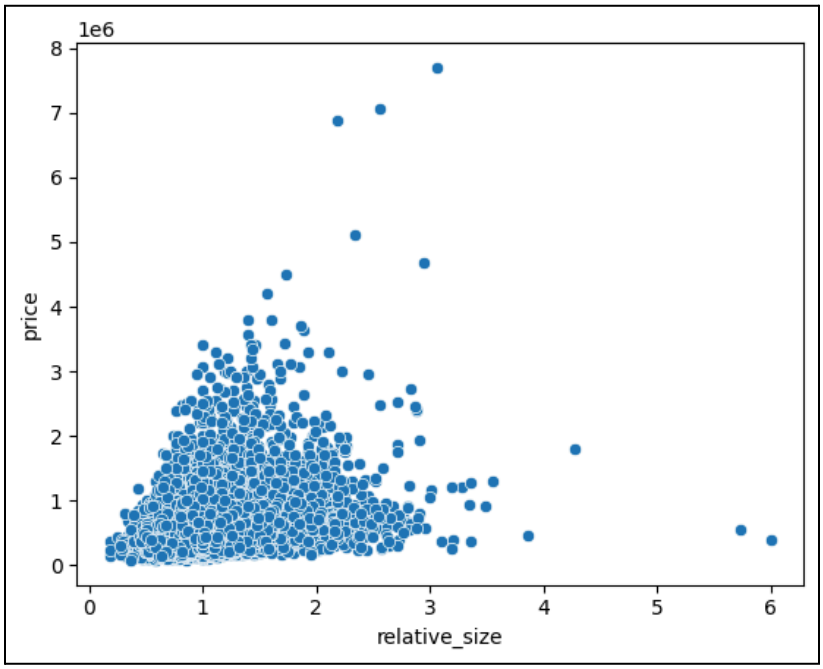
These houses which are of lower price, mostly contain houses which are tightly packed, with less free surrounding space around the houses. This congested layout of all the houses around decreases the price, and the less amount of greenery or waterbodies is playing a factor.

These images, which are on the higher price end, show a clear pattern of having lots of space surrounding their houses, and less packed compared to what we just observed earlier. We can even observe some water bodies in the above image, and lots of greenery surrounding the houses. The relaxed layout of the houses and presence of nature elements increases the price.

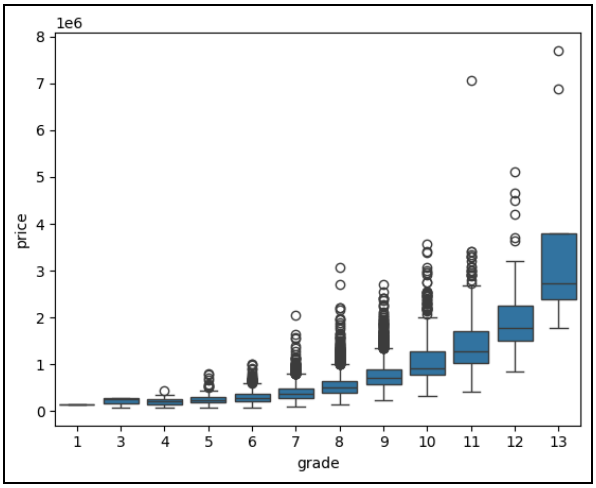
Let's have a look at the neighborhood density data and see if the surrounding houses affect the prices as well.

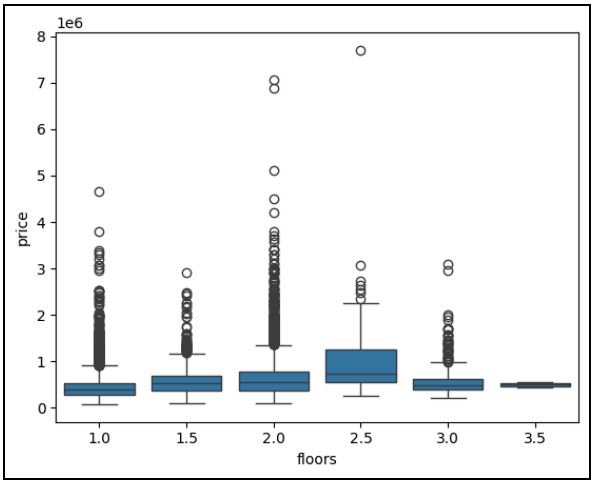
If we plot the sqft_living against the sqft_living15 (average area of surrounding 15 houses) and sqft_lot15 (average lot area of 15 houses), we can see that there is dependency on the surrounding lot space. If the lot space is much larger than the surrounding houses, then the price is slightly higher. Relatively, if the neighborhood houses have small lots, then the living space area causes the price fluctuations.

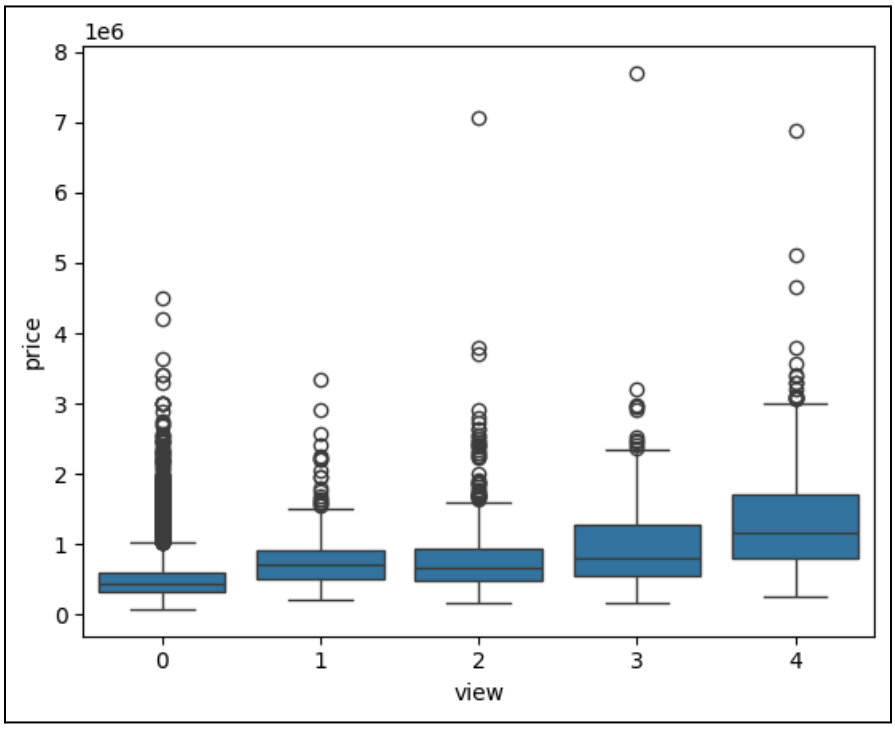




I created a feature which is the relative size of the house compared to the 15 neighborhood's houses. Here it is very evident that the relative size plays an important role in determining the price. If the neighborhood houses are much smaller than the current house, then the price will increase more.







If we look at the variation of price when plotted against “view”, “grade” and number of floors, grade and view are showing strong correlation, that if the view and grade of the houses are higher, then the house price is also higher. The number of floors is also slightly correlating with the prices, but it is not showing any drastic improvement in the price values.

4. Model Training - XGB, (CNNs + MLP)

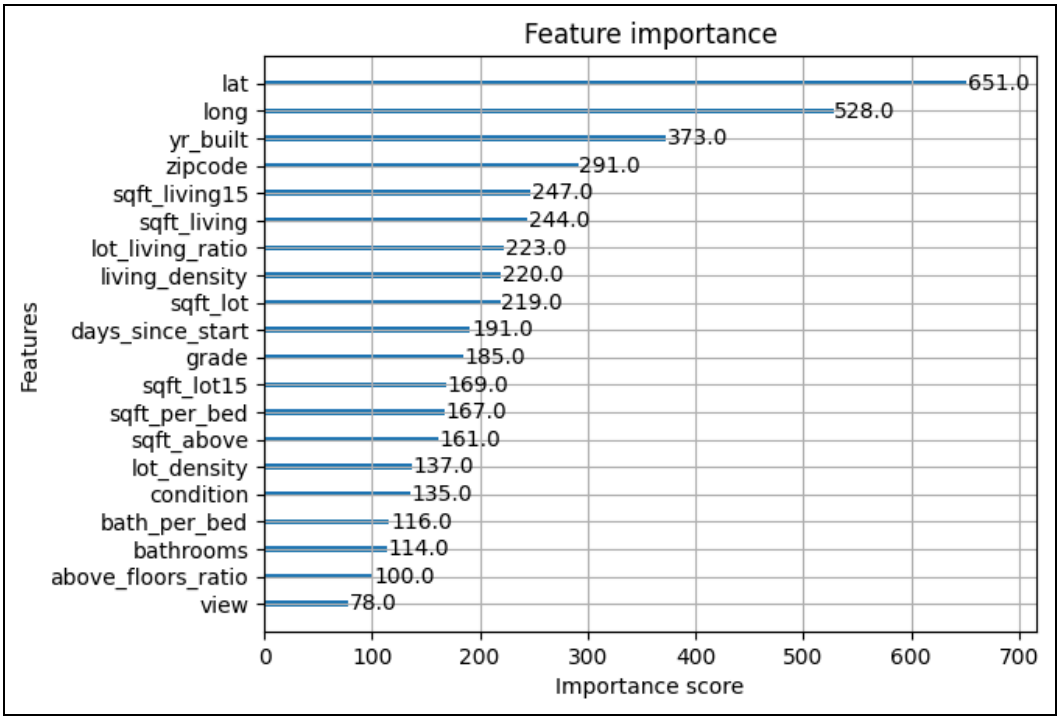
So firstly, I trained an XGB Regressor model. I split the training dataset into training and validation datasets to properly check if the XGB model is working as intended or not. I split about 20% of the training data as validation dataset, for evaluation purposes. I utilized the RandomizedSearchCV function to select the best hyperparameters for this XGB Regressor suitable for the training dataset.

```
params = {
    "n_estimators" : randint(300,800),
    "max_depth" : randint(3,10),
    "learning_rate": uniform(0.01, 0.1),
    "subsample" : uniform(0.6, 0.4),
    "min_child_weight" : randint(1,10),
    "gamma" : uniform(0,0.3)
}
```

```
xgb_model = XGBRegressor(
    objective = "reg:squarederror",
    n_jobs = -1,
    random_state = 42
)

random_search =  RandomizedSearchCV(
    estimator = xgb_model,
    param_distributions = params,
    n_iter = 40,
    scoring = "r2",
    cv = 3,
    verbose = 1,
    random_state = 42,
    n_jobs = -1
)
```

I ran the randomized search for 40 iterations, and once it was done training, I stored the validation set predictions to obtain the RMSE and R^2 score, which is explained in the later sections. The model was saved using ‘joblib’ module.



If we look at the feature dependencies, latitude and longitude, the year the house was built, surrounding neighborhood house area and their lot area ratios are the dominating features which strongly influence the price values.

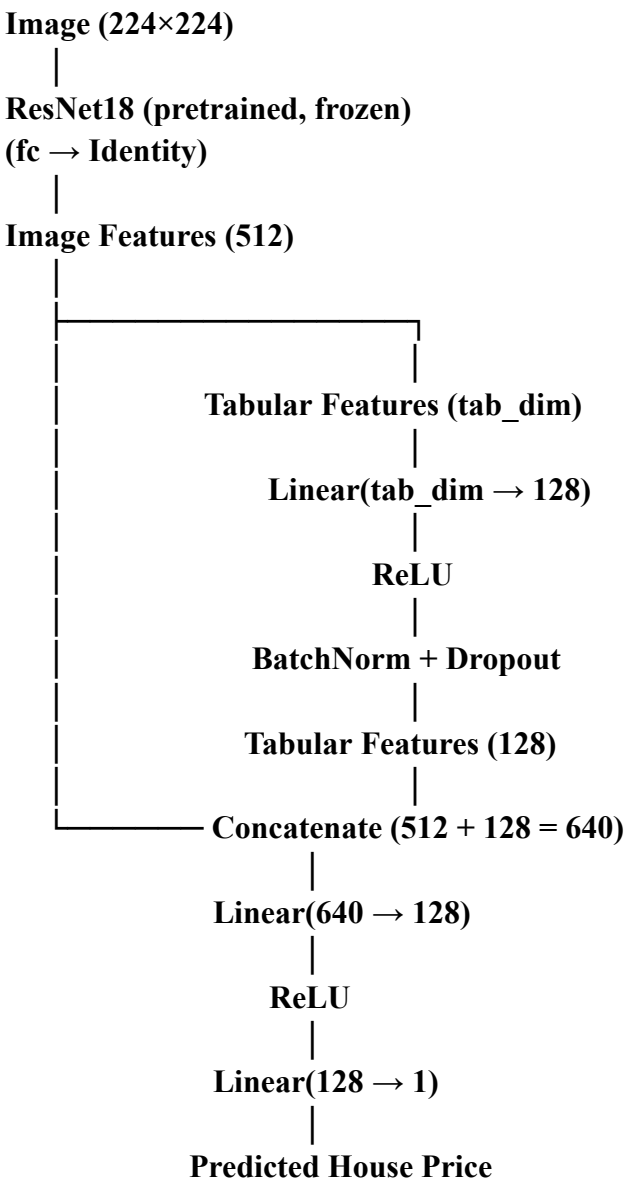
Next, for developing the MultiModal Regressor, I switched from VS Code to Google Colab, since it has the utility of using GPUs for faster training. Locally copied all the training images so that the model can extract visual features while training. In the first run, I did the same thing of dividing the training dataset into training sets and validation sets for evaluation purposes. Since neural networks are sensitive to large values, I scaled the input feature data using the StandardScaler() and did the same for the validation set also. For the target variable (y), I applied logarithmic operation and standardized it using the mean and standard deviation of the training set's price.

I created a class named "HouseDataset", which basically takes the tabular data and the image directory, from which it returns the image, the input feature data and the target value (for training) and index (for testing/evaluation) as a single output. Using a dataloader, with a batch size of 32, I loaded the training and validation datasets.

Now for the architecture of the Multimodal, I used resnet18 as the backbone of the CNN model (for operation) and freezed all the layers (since resnet18 is already pre-trained). For analyzing the tabular data, I constructed MLP with two layers (one input layer and one hidden layer). Both of those model's features are then concatenated and then fed into a regressor head consisting of 1

input layer, 1 hidden layer and 1 output layer. The final output of this regressor head is what the predicted price is going to be.

The architecture of the Multimodal is as follows:



Once I obtained satisfactory results in the validation set, I completely trained the model on the full training dataset and then obtained the predictions from the testing dataset. Before training and testing, I locally copied all 16110 training and 5404 testing images in Google Colab, for faster training, since if i set the path from Google Drive, there is a lot of latency which slowed down the training and testing process. I ran the training loop for 5 epochs and the average loss

was decreasing consistently. I saved the model using torch.save(), and saved the scaler of the x_train as well.

To get the predictions of the testing dataset, I appended all the predicted values and their respective ids in two separate lists. I undo-ed the Standardization which was done before training, and then undo-ed the Log operation by using np.expm1() function. All the predicted price values were stored in a different DataFrame and then saved using to_csv().

5. Results & GRAD CAM :

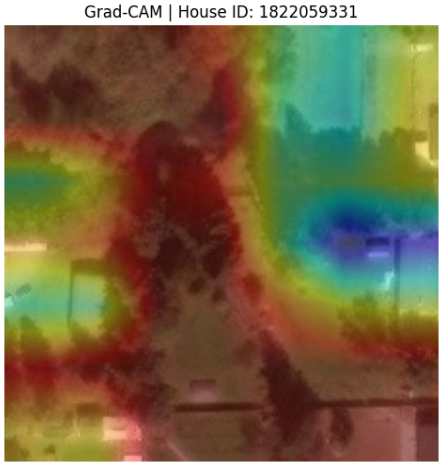
So let's have a look at the results that we obtained from the validation sets of both the XGB and MultiModal models.

Model Type	**RMSE**	**R^2 Score**
XGBoost (Tabular Only)	113,429.77	0.8909
Multimodal (Tabular + Satellite Images)	146,587.88	0.8178

From this table, we can see that the XGB model was performing quite well, and it has a good R^2 score compared to the Multimodal. The RMSE is also not too bad, since it is within the 20-30% error range. There are various reasons for which the Multimodal didn't do much better than XGBoost. Mainly because the Images were noisy, and didn't add much value to the model's performance. The RMSE has also increased slightly, probably due to some features providing misleading results, causing error in the predicted price values.

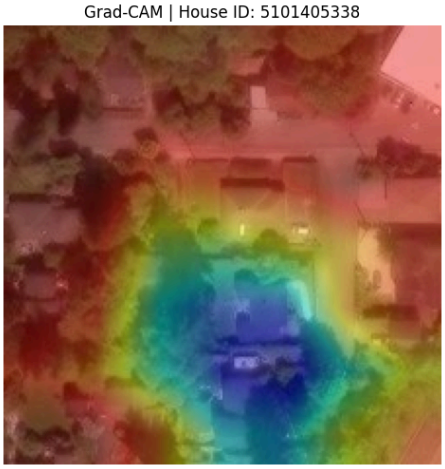
Even though the Multimodal has shown slightly poor stats and results, this approach has provided better explainability of the results such as presence of waterfront, greenery, street layout etc, which can be observed through GRAD CAM analysis.

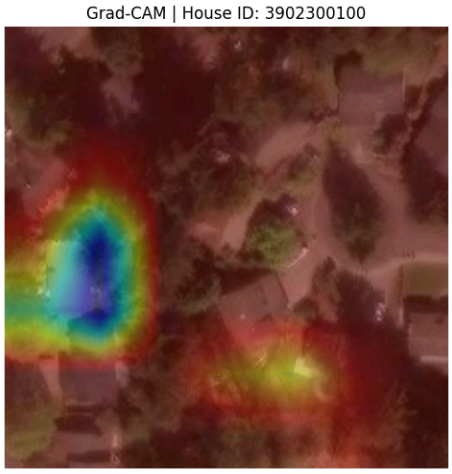
Grad-CAM | House ID: 1822059331

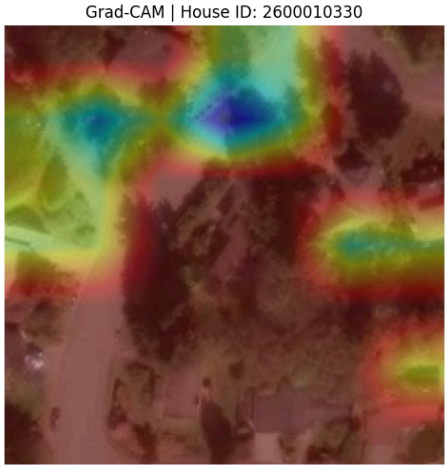


Grad-CAM | House ID: 4142450510

Grad-CAM | House ID: 5101405338







So in the training dataset, I split the samples into 5 categories of prices : very low, low, medium, high, very high. Based on this splitting, I sampled one image and id from each category and performed GRAD CAM analysis, which allows us to understand what the models observe and focuses on the most important visual features. The red highlighted regions are where the model focuses on and tries to gather important features such as neighborhood layout, house density, surrounding lot area, waterbodies and the amount of greenery present. The remaining blue and green highlighted areas are not important to the model, hence it strictly focuses on the red highlighted regions only.

The model is strictly assessing the boundaries of the house and tries to section it so that it can clearly check if the house should be priced higher or not. It learnt features such as neighborhood density, urban house layout, street/road exposure, open green spaces. This improves the interpretability of the image very well, if we just look at the tabular data which doesn't explain the variation in prices even if some parameters are worse compared to higher end prices. XGB Regressor performed better at predicting the house prices, but Multimodal performed better at explainability of the price criterion.