House price prediction using computer vision

1. Problem statement

All people are supposed to have their own home and buying a home is an important step for a human being. Our house price needs to be affordable for the majority with decent income.

In reality, in Ontario for example, less than half of the population can afford a down payment for a house. This is the problem because of the rapid increase of the price in Ontario when the income doesn't increase at the same rate. So the problem of finding the best price for a house related to our income and savings is critical in Ontario.

In consequence, some data found that 57 per cent of people believed they won't ever afford a home, while more than half (54 per cent) of parents in Ontario do not plan on helping their children buy a home soon. Also 80 per cent of current Ontario homeowners are not planning to sell their homes in the next two-to-three years, compared to 77 per cent in 2021.

To find a way to reduce that impact, I decided to answer those question:

1. What can be the best estimated price of a house in Ontario based on the features of the house and image of the house to avoid overpricing by the seller?
2. Which areas are the most profitable to acquire a new property?

Unfortunately, access to Ontario house data was not public and will use datasets of Portland houses sold recently from redfin website to build a model that will be able to give us the most accurate estimated price possible. Also, we will have a look of meaningful features that have direct or some kind of effect on house price.

2. Data collection(~ 4 weeks)

For this problem, I decided to use the website redfin to scrape our tabular and image data of the houses. I first tried to scrape data on Zillow but the website was blocking the use of python code to scrape data on their website. I decided to use an alternative website called Redfin. To scrape data in redfin, I used python

codes using selenium and beautiful soup libraries. This codes help me to scrape 239 house data(URL,sold price,#bedroom,#bathroom,#square feet,address,type of the house,year of construction, estimated price and price per sqft). I put all this data in a CSV file called “houses”. Also, with this code, I was able to scrape ~5000 images of the different 239 houses. After getting those images, I created another python code to put all the images on the same size and especially cropped them because some of them had white space. I put all the clean images in the folder called “cropped houses”. Also note that our tabular data had some missing value on 5 features and we used the median to replace those missing values.

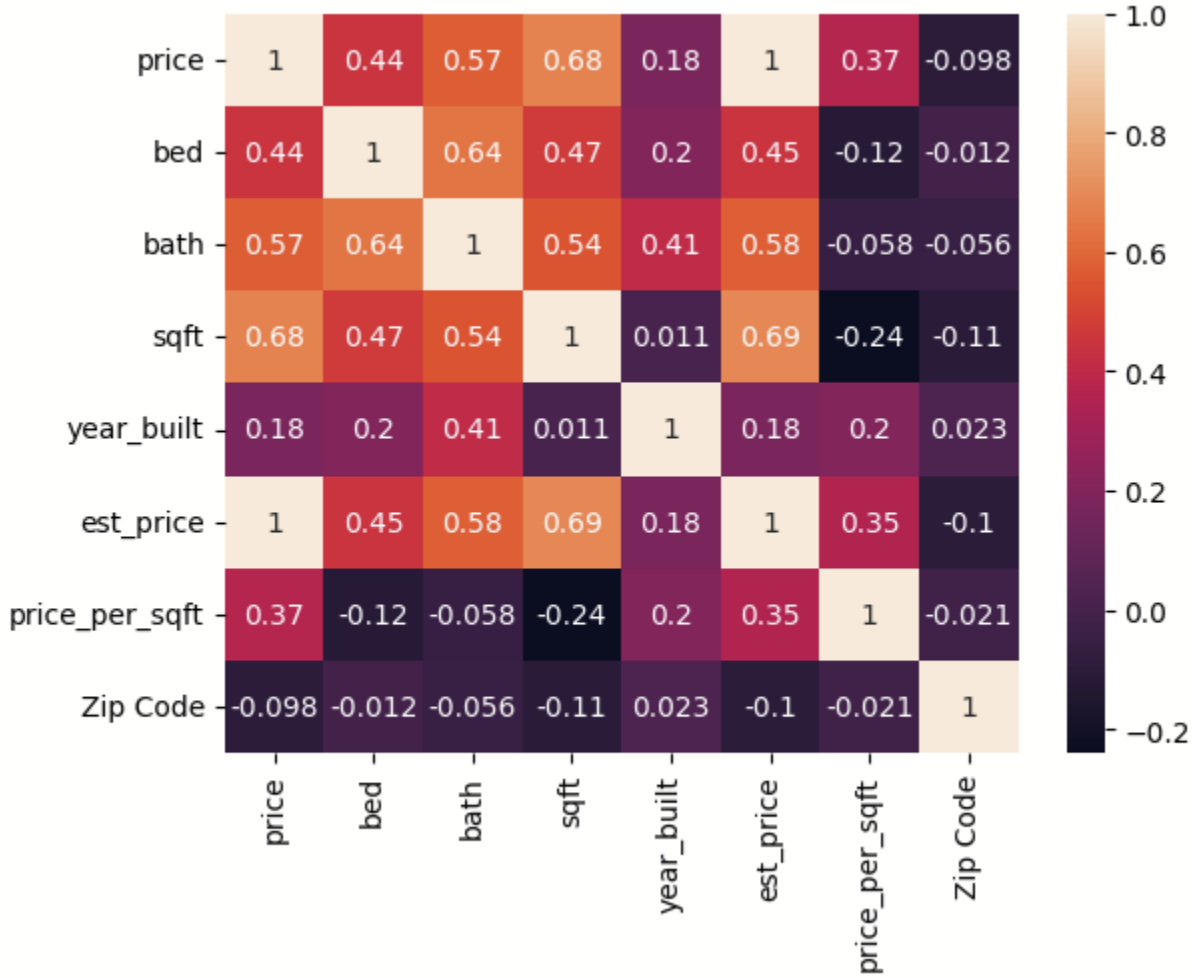
3. Data exploration and visualization(~ 0 to 1 week)

I started first to check the datatype of our tabular dataset houses. I found out that we have 11 features for 239 houses but I decided to use 208 houses because I had images for only 208 houses due to the fact that redfin blocked images for some houses. The types of features are 5 floats, 3 integers and 4 objects or categorical types. After that, we seek for more insights in the data as showing below:

	count	mean	std	min	25%	50%	75%	max
price	208.0	586873.293269	356533.198536	79985.0	414500.00	524500.0	650500.00	3577600.0
bed	208.0	3.110577	1.031992	1.0	3.00	3.0	4.00	6.0
bath	208.0	1.971154	0.804057	0.5	1.00	2.0	2.50	4.5
sqft	208.0	1966.144231	1046.910596	510.0	1267.75	1828.0	2400.00	9900.0
year_built	208.0	1957.403846	34.053530	1888.0	1927.00	1956.0	1982.00	2023.0
est_price	208.0	580618.225962	345461.468020	94599.0	411809.00	525390.5	643587.25	3517359.0
price_per_sqft	208.0	312.610577	90.985200	60.0	252.75	307.0	363.25	724.0
Zip Code	208.0	97219.370192	26.533408	97035.0	97211.00	97218.0	97229.00	97267.0

I can notice here for example that the average price of a house sold is $586,873 while the average sold price based on redfin estimator is $580,618 so a difference of $6,255 on average. We can notice that the houses sold have on average 3 bedroom and 2 bathroom so give us an insight of the importance of bedroom and bathroom for people in Portland,OR or maybe people looking for a house are more family than singles or couples. We can notice as well that the average year of construction of those houses is 1957 with 75% of the houses built between 1956 and 1982 with the newest home sold in 2023. We can also note that the average sq ft is1,966 and the average price is $312.61.

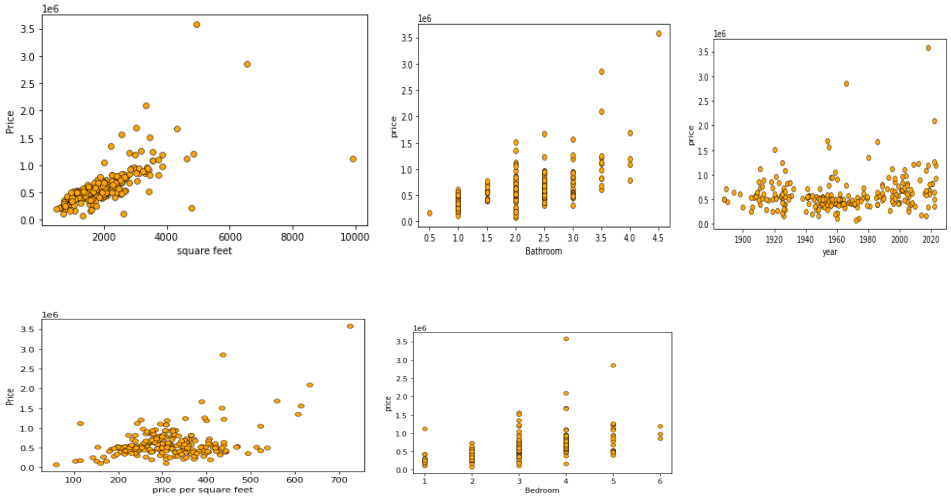
The next steps was to check the correlation between the features and the target variable as shown below:



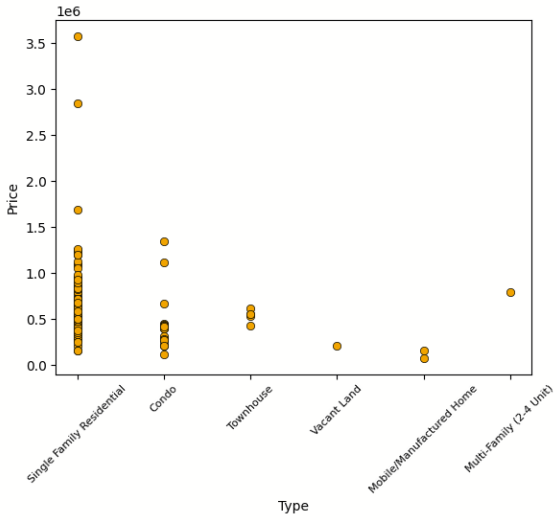
I can notice the features with strongest correlation with sold price with exception of estimated price are first sqft followed by number of bathrooms with correlation more than 0.5. The 2 next features with positive correlation with sold price are # bedroom and price per sqft. We can note for example that year_built and Zip code have low(positive and negative) or no correlation with the sold price. It can be explained by the fact people are looking more for space, number room , and how well maintained the house is.

One last thing I want to notice is the negative correlation between squarefeet and price per square feet that seems interesting even though it is not significant.

We can visualize them with the figures below:



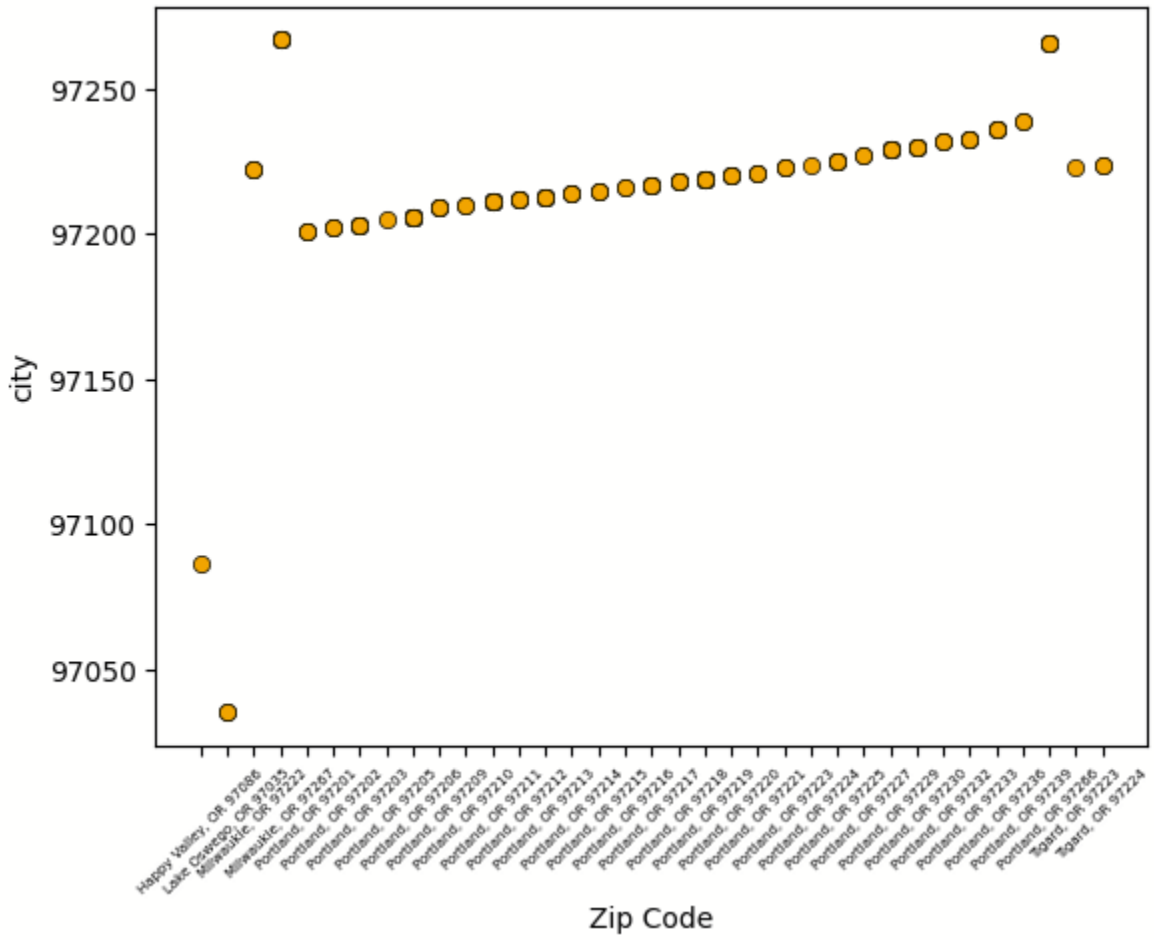
I can notice as well by the graph below that most of the houses sold are single family residential. This confirmed my assumption stated earlier that people buying the houses were more family.



4. Data preprocessing

In this section, I started first to check the unique values of each categorical value and dropped the categorical features that have too many unique values. So we

drop the URL and address. the graph below show correlation between city and zip code but did not have influence on XGboost:



We can notice that the majority of the houses are located in the zip code area between 97200 and 97250 . Also, the positive correlation was confirm by the chi-square test :

Chi-square statistic: 1248.0

p-value: 2.606664370697648e-149

The extremely low p-value indicates that the probability of observing such a large Chi-square statistic under the null hypothesis of no association is very low, so you can reject the null hypothesis and conclude that there is a significant relationship between the two variables.

The last features dropped are the estimated price has a correlation of 1 with price as we will use those estimated prices to see if our model can beat or be close to the estimation system of redfin and price per sqft as i realized that it increased bias.

The next step was to use the one-hot encoding to encode our categorical features.

Finally, we have split our data to 80% train and 20% test and also normalize them.

5. Model selection and training(0~1 week)

For my baseline model selection, I chose the xgboost regressor as my baseline model to improve on. So I started by using GridSearchCV() to search for the best model parameters in a parameter space provided by me.

- The parameter "max_depth" sets the maximum depth of a tree,
- "learning_rate" represents the step size shrinkage used in updating weights,
- "n_estimators" specifies the number of boosted trees to fit,
- "booster" determines which booster to use,
- "gamma" specifies the minimum loss reduction required to make a further partition on a leaf node of the tree,
- "subsample" is subsample ratio of the training instances; this subsampling will occur once in every boosting iteration,
- "colsample_bytree" specifies the subsample ratio of columns when constructing each tree,
- "colsample_bylevel" specifies the subsample ratio of columns for each split, in each level,
- "reg_alpha" is L1 regularization term, and
- "reg_lambda" is L2 regularization term

The gridsearch gave us those result below with cv=5 for our XGboost regressor:

Best parameters:

{'subsample': 0.3, 'reg_lambda': 3, 'reg_alpha': 33, 'n_estimators': 700, 'max_depth': 5, 'learning_rate': 0.009, 'gamma': 7, 'colsample_bytree': 0.7, 'colsample_bylevel': 0.5, 'booster': 'gbtree'}

then train our model based on those best parameters and then predict the mean absolute error(MAE).

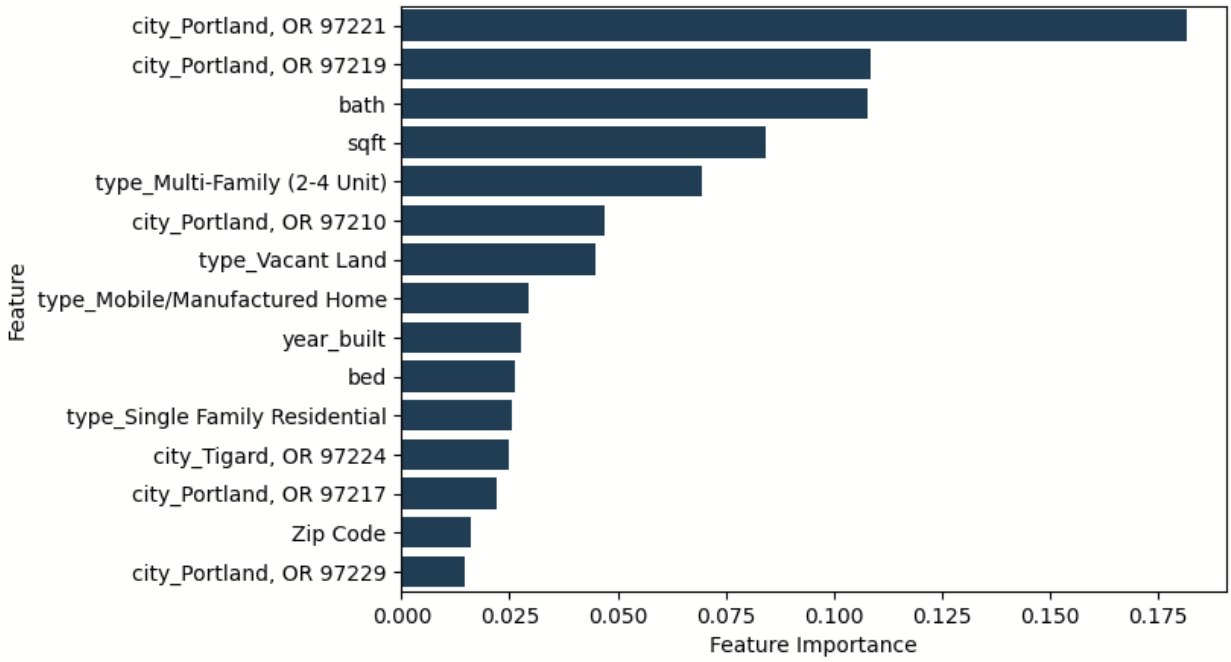
Our baseline MAE is ***XGBoost MAE = 86524.93563988095 or XGBoost MAPE = 16.836234478793596 that shows the percentage.***

I chose the mean absolute error (MAE) as our performance metric to evaluate and compare models. MAE presents a value that is easy to understand; it shows the average value of model error. For example, for our XGBoost model, its MAE is 86,524.93 which means that on average, XGBoost will predict a value that is bigger or smaller than the true value by 86,524.93. Now to understand how good this MAE is, we need to know the range and distribution of the data. In our case, we need to see the values of the target variable price which contains the actual house prices.

	price
count	1.660000e+02
mean	5.892890e+05
std	3.748423e+05
min	7.998500e+04
25%	4.147500e+05
50%	5.300000e+05
75%	6.443750e+05
max	3.577600e+06

We can see that the mean is 589,289 and the median is 530,000. We can see also that the first quartile is 414,750; this means that 75% of the data is larger than this number.Now looking at XGBoost error of 86,524.93, we can say that an error of about 86,524.93 is large for us for data whose mean is 589,289 and whose 75% of it is larger than 414,750. So I will try to improve by constructing a neural network using the image house and those text data.

We pull also the graph below to find out the important features for this model:



We can see that City_portland 97221 ,City_portland 97219, bath, Sqft respectively were the most important features to predict the house price.

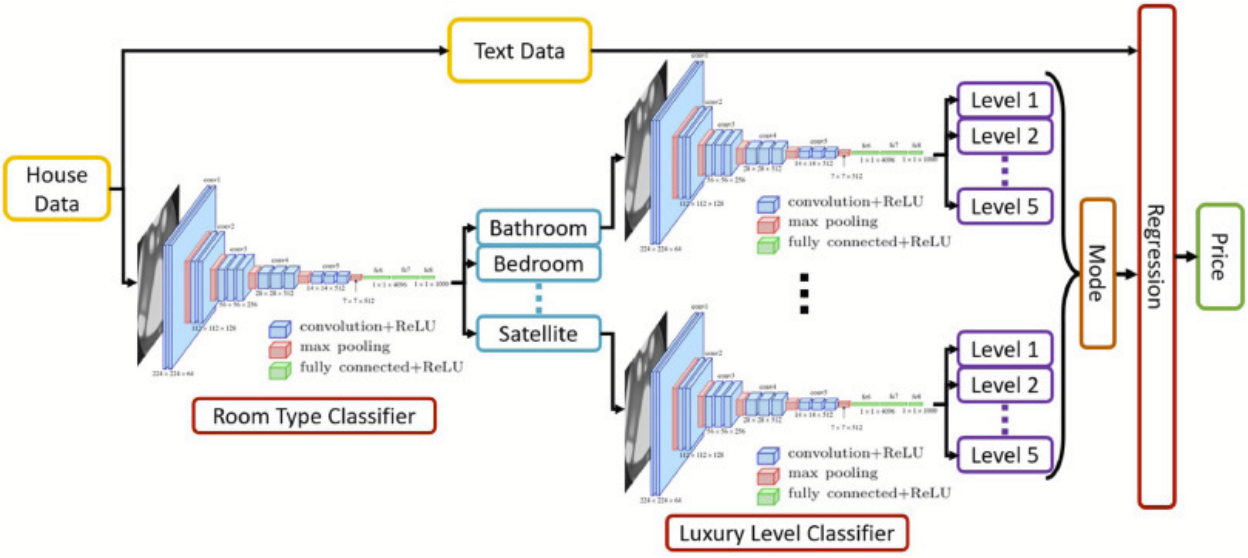
6. Improved Model based on the baseline model(~2 weeks)

Our reflection to improve our MAE or MAPE was to use the image of the houses concatenated with the tabular data using a neural network to see if it will improve our model. I did some research on a similar project. The Vision-based housing price estimation using interior, exterior & satellite images project done by Ali Nouriani and Lance Lemke based their approach:

- Collecting a large dataset of interior, exterior, and satellite images of houses.
- considering interior, exterior, and satellite visual attributes of the house.
- Deep convolutional neural network trained on a dataset to extract visual features of the houses and classify luxury levels.
- Comparing the efficiency of the proposed method with commercial services like Zestimate.

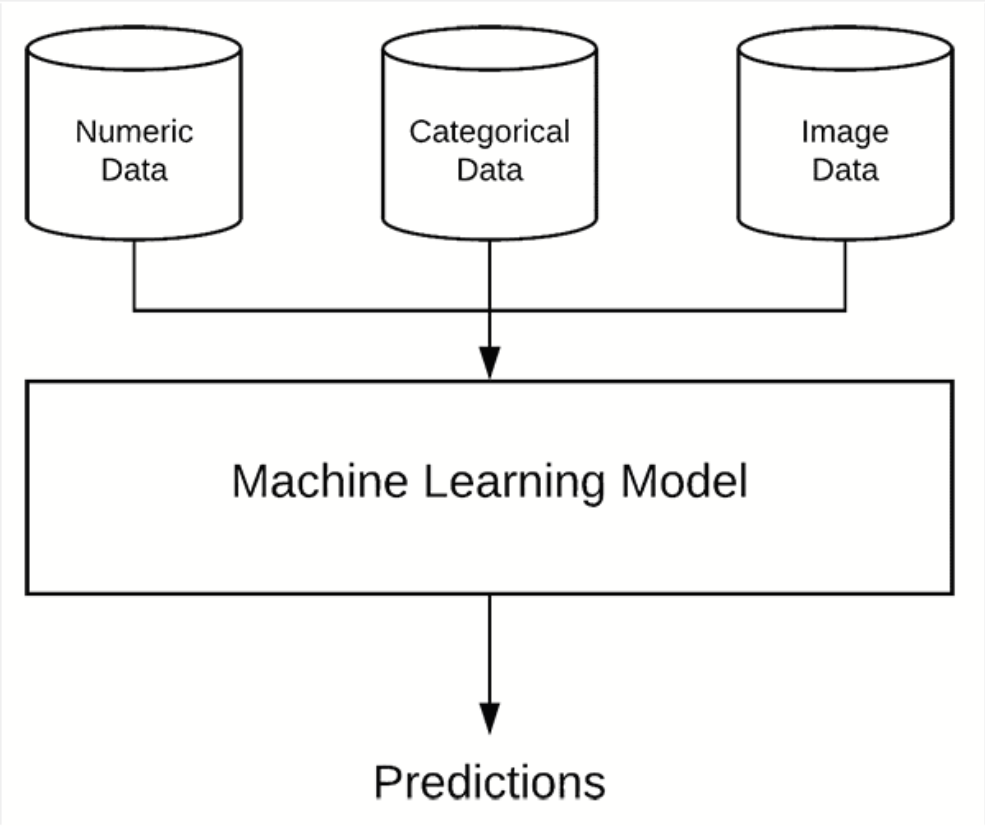
The textual methods have progressed so that they are commercialized in Zillow's "Zestimate" (Redfin, 2022) and Redfin's "Redfin estimate" (Zestimate, 2022). Both the Zestimate and Redfin estimates utilize statistical and machine learning models using textual data to make predictions of home prices. Zestimate and Redfin Estimate have a national median error percentage of 7.7% (Redfin, 2022) and 5.92% (Zestimate, 2022) respectively(Ali Nouriani & Lance Lemke, 2022).On the other hand, there are also vision-based research works like (Bessinger & Jacobs, 2016) that studies only exterior images of the apartments which extracts features using AlexNet, GoogLeNet, and VGG16/19.

For them, The limitation of these baseline works is that they do not include all aspects of the house, such as: Text information, street view, interior view and satellite images.



The picture above shows the architecture they have done to estimate the price with image and text.

In our end, our main idea will be to implemented that model below:



To do that architecture, I will need to create each branch at the time.

1. I will start with the text data so the numerical and categorical data first.

   I will input our numerical and categorical data. Then,I created the function process_house_attributes to normalize our tabular data.

2. Next step will be to process the image data. For that, I created the load_house_images function that has a function to input the image and group each image by house.

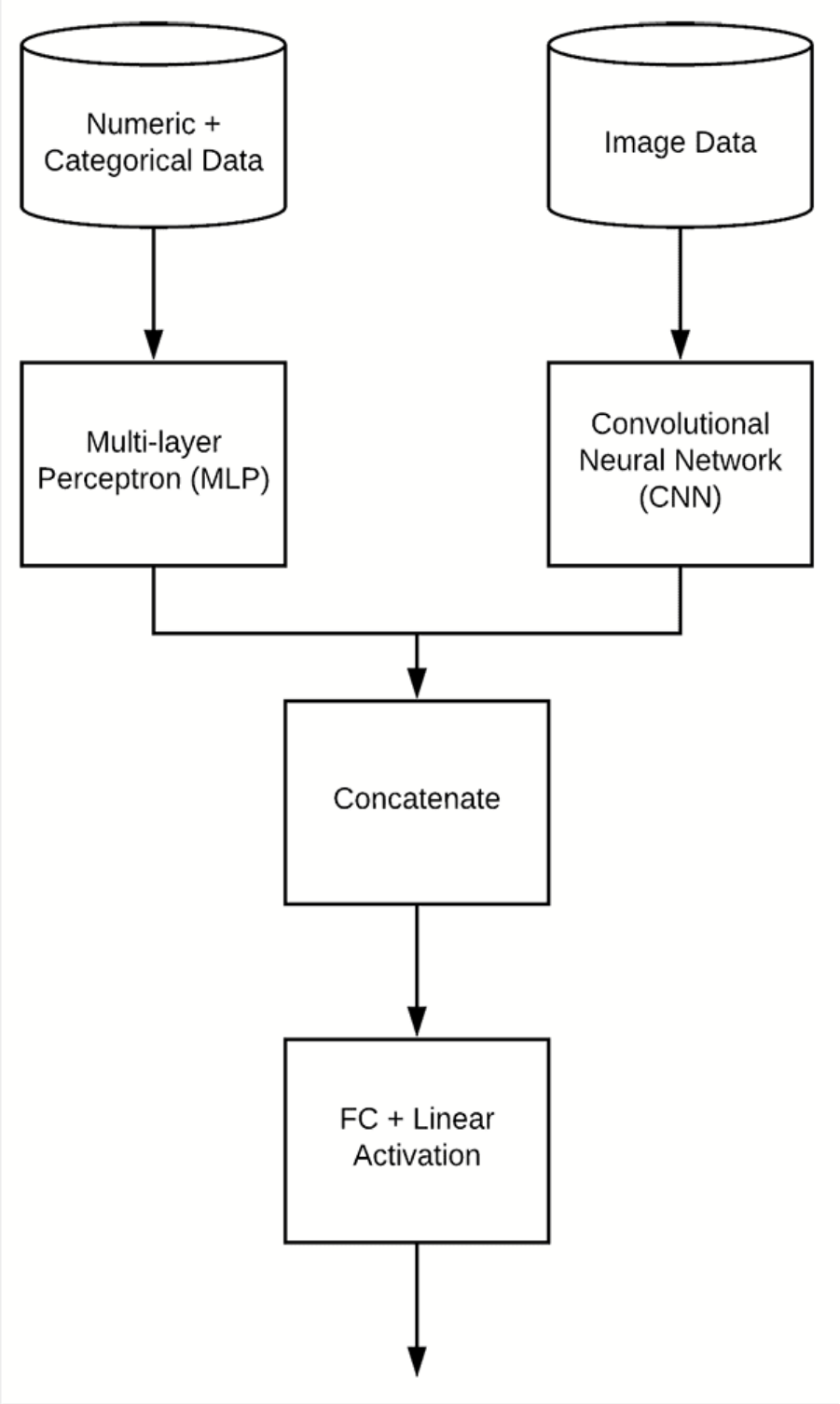
3. After, I create our MLP(Multi-layer Perceptron) function to process our tabular data but not to get the regression result, just to be a branch like the figure below.in the same way, we will create a CNN network to process our images.

4. Mix trained our data.

5. Concatenate the 2 models(CNN and MLP)

6. Create the final FC layers with the last one having a linear activation.

7. Predict and analyze the result.



7. Comparing result

I focused my mixed data neural network in 2 models. The first one was putting images per house and predicting the price based on the images and the text data. For that model, we got result as:

avg. house price: $586,873.29, std house price: $356,533.20
MAPE: 11.88%, std: 8.65%

The second model we did the same thing but isolating image per category(bedroom, bathroom,living room and front view) and we got the result as:

avg. house price: $586,873.29, std house price: $356,533.20
MAPE: 8.55%, std: 6.64%

We can summarize that in a table:

Models	MAPE
XGboost Regressor	16.84
Mix data	11.88
Mix data with categorized images	8.55