Report for Singapore Housing Prices Kaggle Competition

Introduction

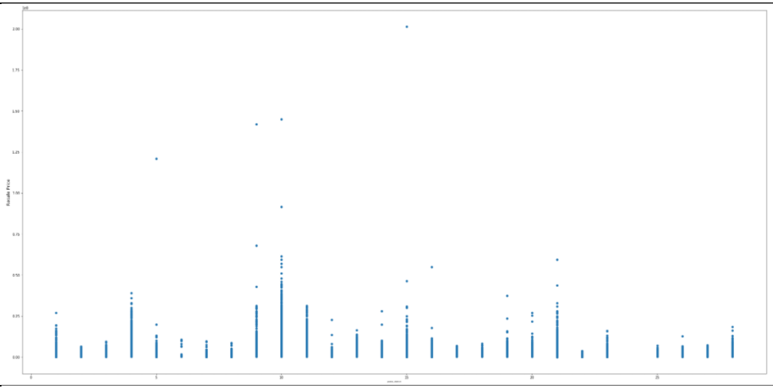
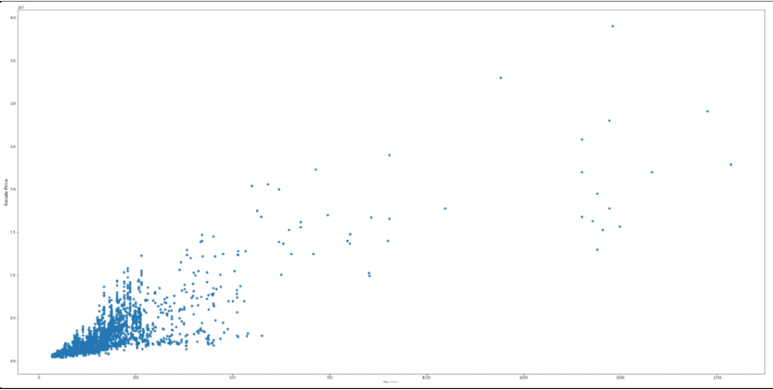
The goal of this Kaggle competition is to predict the house prices in Singapore. The dataset was divided into 1) HDB and, 2) private housing.

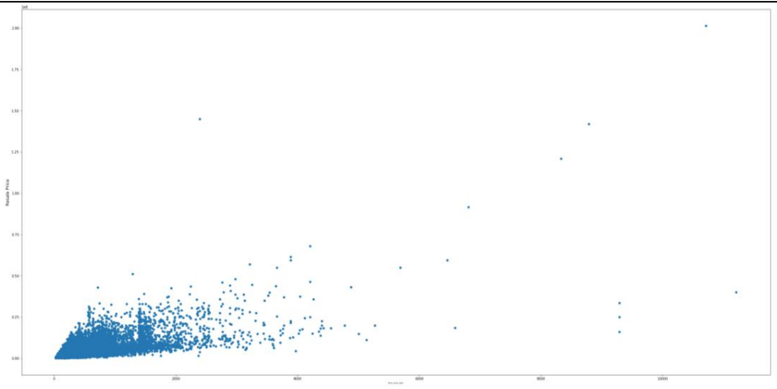
Understandably, I expect these two datasets to come from different distributions since HDB prices, although we are only considering resale, are controlled by the government. Private housing prices are still subjected to the free market of demand and supply despite some cooling measures by the government. I expect to build two separate models to fit these two distributions.

We are only using the provided datasets which contains location details of a usual property listing. For example, we have floor area (sqm), type of land, block, lease length, built date, sold date, apartment floor level, etc. We are not allowed to use external datasets. If we could, we can further improve the model with location of branded schools, train stations and bus interchanges.

Exploratory Data Analysis

Fortunately, we were provided with a very clean dataset. Off the bat, we could easily see correlations between some of the variables and price.

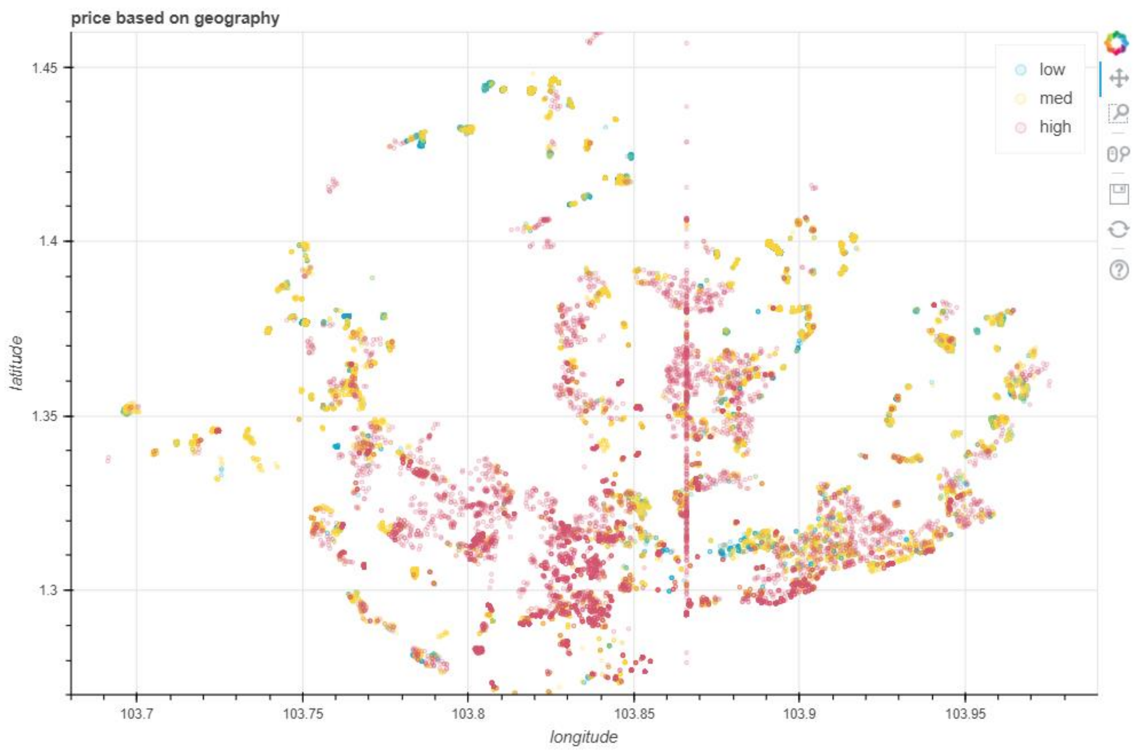
	(A) This is a **scatter plot of price(y) against postal district (x).** Some districts are located in the central region, has more prestigious schools or higher accessibility to other parts of Singapore. E.g. You can clearly see a peak in distinct 10 which contains Ardmore, Bukit Timah, Holland Road and Tanglin.
	(B) This is a **scatter plot of price(y) against floor area in sqm in 2010**. As expected, the higher floor area fetches higher prices. Interesting to observe that the price spread grows wider as the square area increase.

	(C) This is a **scatter plot of price(y) against floor area in sqm for the whole dataset**. You can see that the plot still observes the same pattern as (B). The point to be made here is that by plotting all the years, it seems the pattern is not as clear as looking at just a single year.

Some of the variables like floor area in sqm, postal district and floor number have very clear correlation with the price. However, over the past 20-30 years, house prices in Singapore have changed quite a lot. Intuitively, variables that related to years should be treated differently.

Location, Location and Location

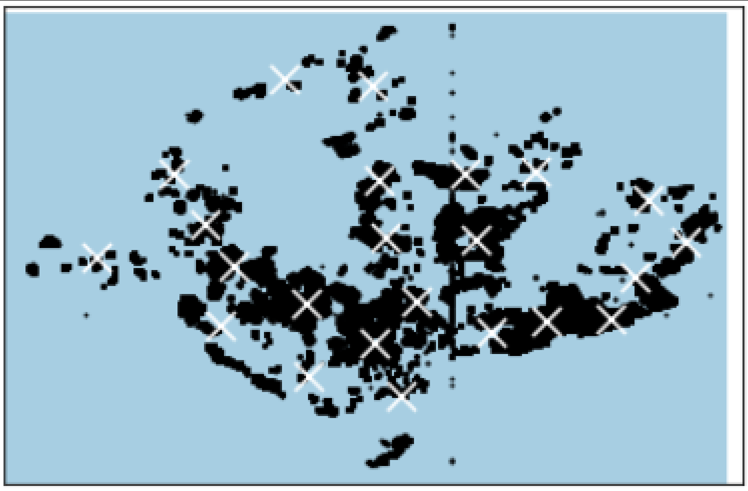
We are also provided with the latitude and longitude of the houses. If left untreated, this variable is not very useful to be used in the model.



This is a **plot of latitude(y) against longitude(x) for year > 2015**. We can see a rough outline of Singapore. The colour coded dots refers to different prices bands (red – high, orange – medium, blue – low) that I have arbitrarily defined. The main point is that the latitude and longitude of the house can tell us a lot about the house prices. E.g. You can see a lot of red/pinkish dots around the CBD, district 10, East Coast and Bishan. The prices of the houses away from the city fringes are mostly low-medium priced. (There are a small number of anomalies that resulted in a straight line on the plot. Later I removed them since there are only a few.)

Feature Engineering

- **Cluster**: To deal with the **latitude** and **longitude**, I used K-means to segment these points in k clusters. Through trial and error and plotting the clusters, I decided to use 20 (HDB) and 23 (Private) clusters which gave me this plot.

	Generally I am quite satisfied with the clusters because they covered the major clusters on the map except for the area around Sentosa which I could not capture no matter how low/high k I used. Nevertheless, these cluster labels will be fed into the model as a categorical variable.

- **Year Sold**: I used regular expression to extract the year sold from “2016-11” in **month** variable.
- **Lease Length**: I used regular expression to extract the lease length from the **tenure** variable which can be 99, 999, 9999 or even NA. Most of the NAs are actually freehold. I filled the NAs with 9999 since the difference is negligible (assumed).
- **Lease Start Year:** I applied regular expression again on **tenure** to get the lease start year. This can be fed in the model as a categorical variable. Also it helps to create the next variable.
- **Lease Length Left:** This is also an important variable because it is basically the life span of the apartment. I create this using this formula **(Lease length – (year sold – lease start year)).** (I calculated age of apartment instead of lease length left for the HDB dataset.)

For all categorical variables, I used **label encoding** from **scikit-learn** package. I chose to use label encoding instead of one hot encoding because it is suitable for the model that I will be using. Also it helps to reduce the amount of memory needed to store and process this dataset.

Model Architecture

I will be using the **keras** deep learning framework because it is fast and easy to use. As for the model, I am using a very simple architecture which can be summarized in the diagram below.

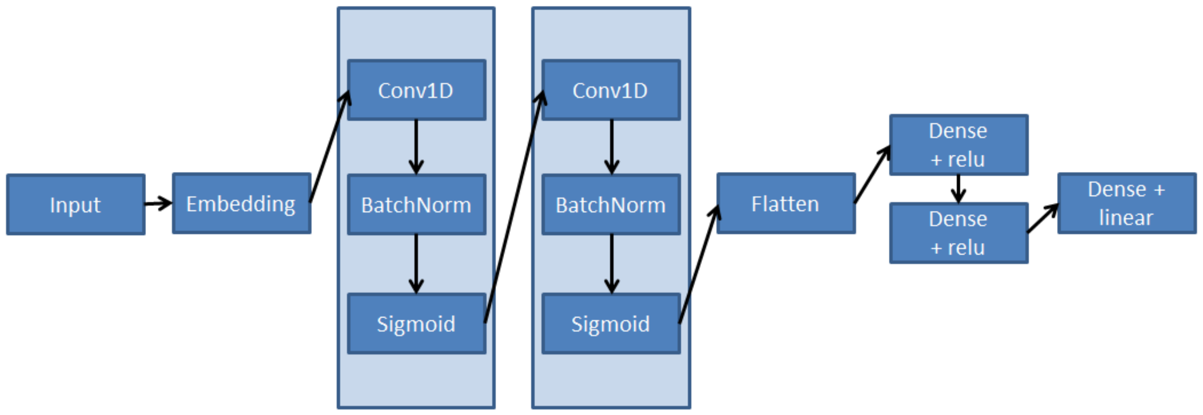


Figure 1 Private Housing Dataset Model Architecture

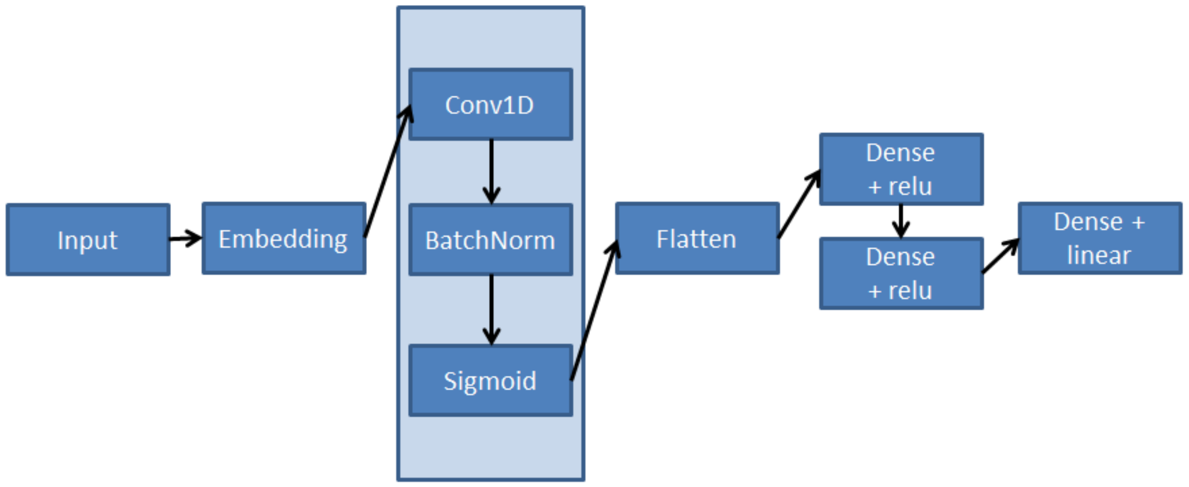


Figure 2 HDB Model Architecture

Firstly, I decided to use the **embedding** layer because I have a lot of categorical variables that contains very helpful information. I did not want to use a one-hot encoding and feed it in a usual MLP because it will be too slow and I cannot guarantee a good result. With an embedding layer, I effectively summarize all the information contained in thousands of categories with just 64 dense embedding.

Subsequently, I used a **1 dimension convolution** layer to look through the 64 dense embedding. I have had good results with embedding followed by convolution in the past. I also prefer CNN over RNN or MLP because CNN training is usually quite speedy because of parameter sharing.

The **BatchNorm** layer normalizes all the neurons which can help the model to converge much faster and reduce overfitting.

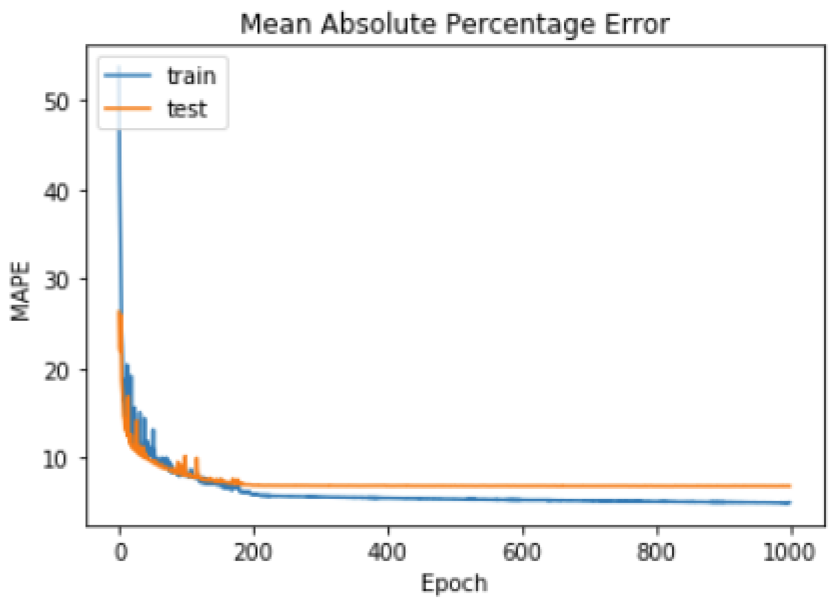
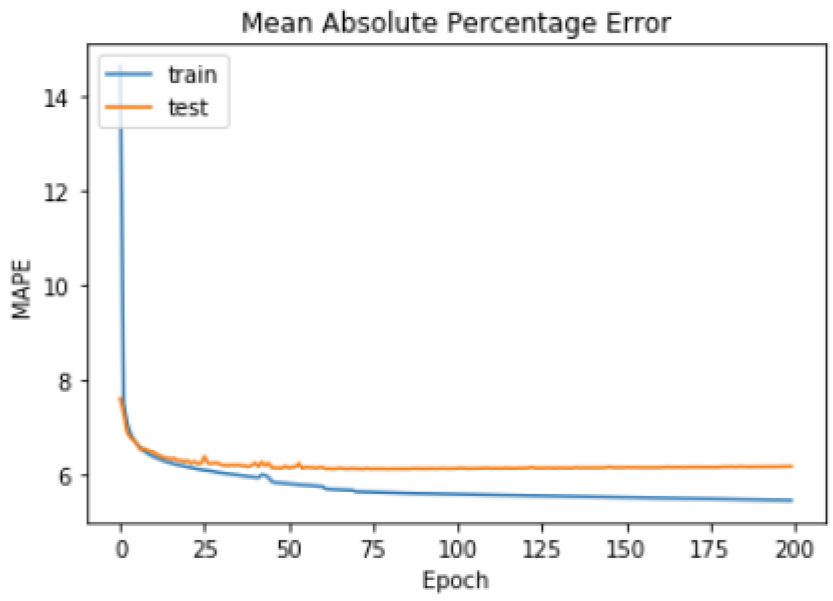
I used the **Sigmoid** activation layer instead of the popular relu activation. My model could not converge if I used relu.

Next few layers are pretty much the same as any modern deep learning networks. For the last dense layer, I used **linear** activation function since we are predicting a certain value instead of a probability.

We were warned that the private housing dataset could have different distributions for different sale types. Instead of building 3 separate models, I decided to label encode them and let the model deal with it. To give the model a helping hand, I gave it one extra convolution block compared to the HDB model architecture.

The model was able to train and converge pretty fast, running only on CPU. It takes only around 27s for each epoch (128 batch size).

I was able to achieve ~6% MAPE for both private and HDB validation datasets.

 Figure 3 MAPE for Private Dataset	There is some over-fitting after 200 epochs. We can try adding a dropout layer between the convolution layers to reduce over-fitting.
 Figure 4 MAPE for HDB Dataset	There is some over-fitting after 20 epochs. It may be better to reduce the number of parameters in the layers to reduce this over-fitting

For both models, I stopped the training prematurely using early stopping to prevent too much overfitting.

I was able to achieve 6.31974% on the public leaderboard.