Machine Learning-Based Application for HDB valuation

Tan Jun Heng (A0216050X)[1], Derek Ng Wei Kang (A0215125U)[2], Er Song Heng (A0216099X)[3], Josiah Lee En (A0216329E)[4], Ong Han Yang (A0217422N)[5]

National University of Singapore

E0538151@u.nus.edu[1], E0535078@u.nus.edu[2], E0538200@u.nus.edu[3], E0538430@u.nus.edu[4], E0543458@u.nus.edu[5]

Abstract

Housing Development Board (HDB) is the most common housing choice in Singapore, and their valuation is often inaccurate or will incur additional costs. Here, we explore different Machine Learning algorithms to aid in the valuation of HDB flats across the island. Notably, we have finalised to use a Random Forest Regression model, and obtained an absolute mean percentage error of around 6.7%, with 84% of our predictions lying within 10% percentage error for samples drawn from the 8 HDB towns that were used for training. This is better than SRX, the current predictor available, which had an absolute mean percentage error of around 8.2% and only had 69% of their predictions lying within 10% percentage error.

Motivation

In Singapore, Housing Development Board (HDB) flats account for roughly 80%[1] of housing for Singaporeans and the resale market accounts for more than half of the housing transactions in Singapore. Currently, resale buyers must first negotiate with the seller on the selling price. This process is often time-consuming as the buyers and sellers need to understand the current resale market for a fair valuation.

Buyers and sellers are able to visit the HDB website to aid with their property valuation. However, this website only shows past retail prices of blocks in that area and it may not be useful for prediction of the current value of the flats. An alternative would be the SRX website. However, this website tends to give a conservative estimate for their valuation.

As such, most buyers and sellers would often engage a property agent to aid them with their negotiations, at the cost of up to 2% commission fees[2].

We aim to develop a Machine Learning-Based Application for HDB valuation to help buyers and sellers agree to a proper price before HDB valuation. Our model will try to account for any human biases and will provide a fair, objective and accurate pricing for both parties. With knowledge about current and future HDB prices, both buyers and sellers can start on the same page with a recommended price as a basis for negotiation, leading to quicker negotiations and a fairer price for both parties.

Description

The main target audience for our machine learning application will be

1. HDB sellers
2. HDB buyers
3. Property Agents

The main hallmark of our ML application will be the inclusion of distance to amenities such as schools, hawker centres, gyms, parks and Mass Rapid Transit (MRT) stations. We included these because we believe that during most negotiations, the distance to these amenities would often have a large impact on the valuation of the HDB. Granted, the inclusion of these additional attributes may be hard to account for manually. As such, we would aim to use a Machine Learning model that can exploit high-dimension input data.

Our model will be trained using historical HDB resale data obtained from *data.gov.sg* with the location data used from *data.gov.sg* and Google geocode. The distances to the nearest amenities are calculated using a greedy approach, calculating the distances to all amenities and selecting the shortest distance. We also chose the 8 towns *Jurong West, Tampines, Woodlands, Bukit Batok, Bukit Merah, Toa Payoh, Clementi* and *Sengkang* to cover all areas of Singapore to predict accurate HDB prices for all areas of Singapore (North, South, East, West and Central).

Our target is to achieve a better prediction than SRX for at least predicting the resale value for the 8 towns specified above so that all three parties in the transaction can have a more accurate and unbiased estimate of future property value.

Research Methodology

The general workflow of our machine learning project is shown below (Figure 1). This is to provide ease of navigation of the report.

[1] https://www.hdb.gov.sg/about-us/our-role/public-housing-a-singapore-icon

[2] https://www.99.co/singapore/insider/commission-property-agents/

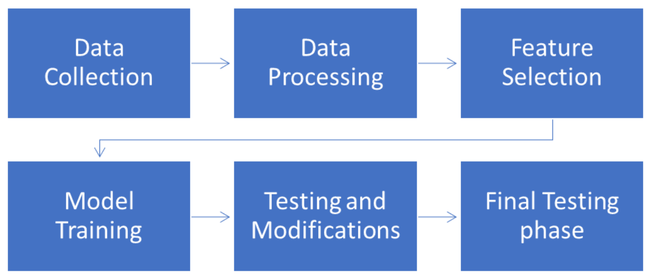

Figure 1. Flow chart of general workflow of the project.

Data Collection

Our data was extracted from the public government website *data.gov.sg*, Google's Geocode API as well as the SRX website.

Types of Data Extracted

HDB Data:

Firstly, we extracted the address and details of HDB flats in Singapore. To attain the geolocation of all the HDBs, we used the python library "BeautifulSoup" to iteratively web scrape the data from Google's geocode API.

Amenities Data:

Secondly, we extracted details of amenities in Singapore. These include data about kindergartens, primary schools and parks etc. Similarly, we extracted the geo-coordinates of each of the amenities.

SRX Price Data:

Lastly, we extracted data from the SRX website to serve as our benchmark for final testing. This was done with the python library "Selenium" to write a program that automates this process.

Limitations

Some of the data we gathered was not up to date. For example, our test data was last updated on April 8, pre-school locations were obtained from August 12 2020, retail pharmacies from March 18 2019, hawker centres from September 1 2021 and so on. There may have been new amenities in recent years which may have affected the predictions of our model, that were not in the amenities data collected.

The original HDB data also did not contain the unit number, which is required to get the predicted price from SRX. As such, we standardized across different HDBs and allocated them the same unit number when scraping. The SRX site went down while we were scraping, thus we were only able to scrape about ⅔ of the required data. As such when using SRX as our benchmark we were only able to attain 400+ data entries.

Data Processing

Data Cleaning

As previously mentioned, for purposes of demonstration, we decided to train our model on the following 8 towns: *Jurong West, Tampines, Woodlands, Bukit Batok, Bukit Merah, Toa Payoh, Clementi* and *Sengkang.* This is done by filtering the dataset using the town variable.

Preprocessing the Data

Besides the variables found in the HDB dataset, we also added a few other features.

First, we added columns (Lat, Lon) to the dataset. These are the latitude and longitude of the resale flats, which were scraped from Google API. For Lat Lon values that are missing or outside of Singapore, we updated the Lat Lon values manually.

Next, we added "*dist_*" columns. They are the shortest distance from a HDB unit to a preschool, gym, kindergarten, park, retail pharmacy, MRT Exit and hawker centre respectively. To do this, we calculated the distance of two points using the Haversine formula. To speed up calculations, we only considered Lat Lon values within range [x-k, x+k] and [y-k, y+k], where (x,y) are the Lat Lon values of the HDB unit and k is some offset.

To ensure the shortest distance is found, the offset must be large enough so that the max(*dist_*) < distance between (x,y) and (x+k, y). After some trial and error, an offset of k=0.03 satisfied the above constraints (max(dist_) is *dist_gym* = 2861m, distance between (x,y) and (x+0.03, y) = 3336m)

Next, we added the column *storey_index*. Upon inspection of the *storey_range* column, we noted there were two different representations of *storey_range*, either in groups of 5 (eg "6 TO 10") or in groups of 3 (eg "7 TO 9"). We decided to randomly split groups of 5 into groups of 3. For example, there were 1128 instances with storey_range "06 TO 10". We randomly split these instances with the following proportions:

- "04 TO 06": 226 instances (⅕)
- "07 TO 09": 677 instances (⅗)
- "10 TO 12": 225 instances (⅕)

Then, we converted the storey_range into a *storey_index* with a 1 to 1 mapping. "01 TO 03" is 1, "04 TO 06" is 2 and so on.

Finally, we added a column for the price index. While the Resale Price Index (RPI) provided by HDB is useful, it is not updated regularly enough. As of writing, this index only includes data up to 2021-Q3. After some testing, we decided to use a three-month-average as a substitute index. For any month x, we calculated the average resale price of units sold in the past 3 months, including x. We then divided this by the value found in February 2009, the same base that RPI uses. The comparison between our price index and RPI is shown below:

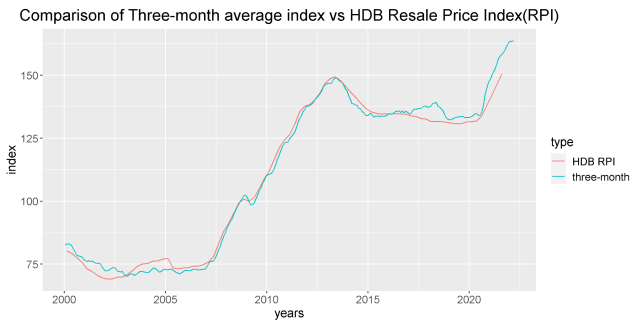

Figure 2. 3-month average index vs RPI

Additionally, below are some examples of columns that were altered:

Columns	Preprocessing method
● *flat_model* ● *flat_type* ● *town*	One-hot encoding was used to relabel these categorical variables into dummy variables.
● *block_number* ● *street_name* ● *lease_commence_date*	Removed as they are directly correlated with *lat lon* and shortest distances found earlier. *lease_commence_date* is directly correlated with years and *remaining_lease*

Feature Selection

Methods Used

For feature selection, we used two methods to narrow down which variables in our data are relevant.

Variance Inflation Factor

In our selection of the variables, we used the Variance Inflation Factor to check for multicollinearity.

```
           years     townBUKIT MERAH          townCLEMENTI      townJURONG WEST
        7.200994           32.894970              6.971514             6.808130
   townSENGKANG        townTAMPINES          townTOA PAYOH        townWOODLANDS
       45.921944          119.565634             21.280202            50.069275
             Lat                 Lon flat_modelApartment        floor_area_sqm
       81.831472          133.309539              7.551191            19.705457
 flat_type4 ROOM     flat_type5 ROOM  flat_typeEXECUTIVE                  <NA>
        8.078452           22.451324             22.997357                   NA
           index
        6.354960
```

Figure 3. Variance Inflation Factor values for the variables

We found there is serious multicollinearity between some variables. Hence, we removed *flat_type*, *years*, *lat* and *lon*.. This reduced VIF for all regressors to below 5.

Forward Selection

In our selection of the variables, we used Akaike's Information Criterion (AIC) to decide which variables to leave out at each step. This was done in R code. Below are the results of the selection (Figure 4).

```
               Df  Sum of Sq        RSS     AIC
<none>                       5.4828e+14 5364765
- dist_kinder   1 2.3963e+10 5.4830e+14 5364773
- dist_gym      1 3.6841e+11 5.4865e+14 5364930
- dist_park     1 7.6800e+11 5.4905e+14 5365112
```

Figure 4. Results of backward selection

Although the inclusion of all regressors increased the AIC, the increase by adding *dist_kinder* is very minimal. We decided to remove *dist_kinder* due to Occam's Razor.

Conclusion and Assumptions

It is important to note that these methods rely on the assumption that each regressor has a linear relation with the response variable (*resale_price*). We decided this was a reasonable assumption for all regressors other than *lat* and *lon*. Thus, we added *lat* and *lon* to train non-linear models.

Model Training

Training Methodology

Overview

The dataset used dates from 2001 to 2022, with more than 200,000 entries. However, the benefits of the datasets 20 years ago when they were used to predict future prices in 2022 may not be the best. For example, the closest amenity we calculated may not have been built when the unit was sold. Hence, our group explored trading off more data against correct data, by making a novel revision to train the various models based on different year ranges to find out which year range is the most accurate.

The data obtained was up till March 2022. We have chosen to split the data into pre-2022 and post-2022. The data from pre-2022 (ie 2001 to 2021) is known as D1 from now on, while the data from post-2022 is split further into 2 parts: S1.1 and S1.2 (collectively known as S1)

Testing for the Various Year Ranges

Each model is tested with data ending in the year 2021. Sequentially, 2 years worth of data will be added for each time the model is trained, leading to 11 different year ranges per model.

For instance, the first iteration will be initially trained with 2021 data. The next iteration will be trained with data from 2019-2021 and so on.

Process

We have 3 parts to our project.

In Part 1, D1 will be used to train all the different models, evaluating them based on k-fold cross-validation, as well testing against S1.1. After which, the best algorithm and year range will be chosen to proceed to part 2.

In Part 2, the best year range will be combined with S1.1 to produce D2, the dataset used to train the selected algorithm from part 1. We would also tune the hyperparameters of the selected algorithm. Finally, multiple iterations of training this algorithm with the selected hyperparameters is done, testing against S1.2 each iteration. The fold in the iteration that produces the lowest percentage error will be chosen as our final model.

Finally, to evaluate the accuracy and related scores of our final model, we also collected data from April 2022 from *data.gov.sg* late into the project to use as our final test set. This dataset is known as S2.

Rationale and Assumptions of the Algorithms

Linear Regression

We considered using linear regression as the basis of comparison as our dataset was expected to be linearly separable.

We use two diagnostic plots, a residual vs fitted plot to check linearity and a Normal Q-Q to check normality of errors.

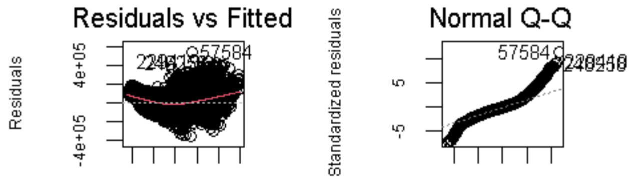

Figure 5: Diagnostic plots before box-cox transformation

Figure 4 shows that there are issues with these assumptions. To solve this, we used box-cox transformation of the response variable *resale_price* with lambda=0.3434.. calculated using the MASS package in R.

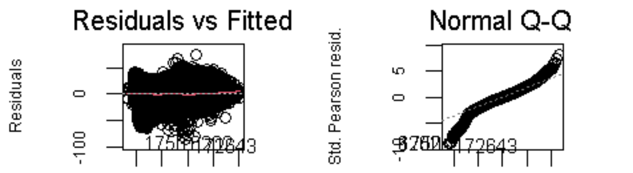

Figure 6: Diagnostic plots after box-cox transformation

The diagnostic plots show an improvement in linearity and normality.

While linear regression works well for extrapolation, it is prone to overfitting, outliers and multicollinearity. Thus, ridge and lasso regression was also considered.

Ridge Regression

To prevent overfitting from linear regression, we tried using ridge regression, which does not require unbiased estimators, avoiding the issue of having large and inaccurate variances.

Lasso Regression

In addition to preventing overfitting like ridge regression, Lasso regression penalises less important features, thus the selection of variables is more vigorous and variables that have lower contribution are sieved out.

K Nearest Neighbour Regression (k-NN)

As k-NN is a non-parametric algorithm, it may be useful as our dataset may not follow theoretical assumptions. Since our dataset has many features, k-NN could effectively estimate instances based on feature similarity. We also recognise that k-NN requires homogeneous features, which we have attempted to tackle through feature selection.

Support Vector Regressor (SVR)

We also considered using SVR as it is effective in high dimensional spaces like our dataset. Despite also handling outliers well, we noted that SVR does not work well with large datasets because of more ambiguous margins of separation between classes.

Decision Tree Regression

As a vast majority of our variables are categorical, we considered using models like Decision Tree Regression that do not assume that our variables have a linear relationship.

Random Forest Regression

To counteract overfitting of decision trees, we also opted for random forest regression where we take the results of many different decision trees.

Testing and Modifications

Part One

Test set: S1.1 (half of 2022 Jan - Mar)

Each of the models with the different year ranges will be validated with its own training data with k-fold validation. Afterwards, each model will also be tested against S1.1. The Sum of Squared Errors (SSE) is the criterion used to judge the goodness of fit of the various models. Below is one of the results of each model, showing the year ranges for the lowest average test SSE.

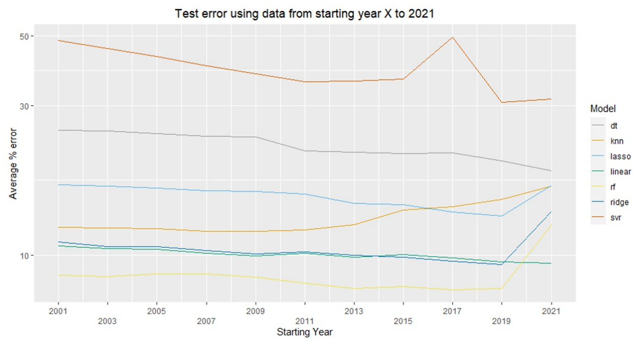

Figure 7: Results of the test error of testing models using data from starting year X to 2021

Results

After comparing the various models, it was observed that amongst the models the random forest regressor performed the best, more specifically the model that trained on the year range 2017-2021(Figure 5).

Part Two

Test set: S1.2 (half of 2022 Jan - Mar)

During this stage of testing, we performed hyperparameter tuning to select the best hyperparameters. The best model from part one (Random Forest Regression), along with the year range (2017-2021) was used. Additionally, the test set S1.1 was included in the training dataset. Afterwards, the model will be tested against S1.2.

With these, we performed k-fold cross-validation to obtain the average validation and test percentage differences, for use in hyperparameter tuning.

After which, the best hyperparameters were used to train the regression multiple times to produce the best model selected by the lowest SSE between predicted and actual resale price. K-folds were also used here to allow for some additional variation in the models generated.

The final model was then saved for evaluation against novel data.

Hypertuning of Parameters of the Selected Model

As we are using Scikit's Random Forest Regressor[3], we have chosen to iterate through hyperparameters such as:

- *n_estimators* (number of trees, from 50 to 100),
- *criterion* (squared_error, poisson),
- *max_depth* (9 to 25, in intervals of 1)
- *max_features* (auto, sqrt, log2)
- *min_sample_leaf* (1 to 101, in intervals of 10)
- *min_weight_leaf* (0 to 0.5, in intervals of 0.1)
- *min_samples_split* (2 to 102, in intervals of 10)

The following graph (Figure 6) displays one of our findings for the hyperparameter tuning:

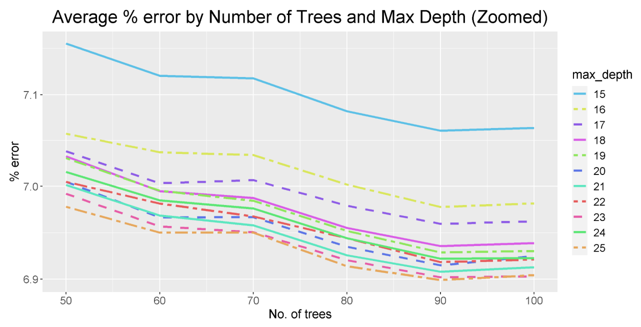

Figure 8: Average % error by n_estimators and max_depth

Ultimately, the hyperparameters we chose were:

- *n_estimators* of 90
- *criterion* of squared_error
- *max_depth* of 18
- *max_features* of auto
- *min_sample_leaf* of 1
- *min_weight_leaf* of 0
- *min_samples_split* of 2

3 https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

Choosing a Model

After hyperparameter tuning, we trained the model for several iterations. The iteration that led us to choose a particular model has the following results:

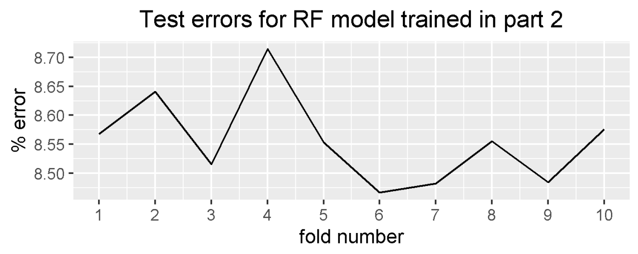

Figure 9: Test error of one iteration of finding a suitable RF model

The 6th fold returned the lowest SSE value, as such it was saved as the model to use for our final evaluation.

Final Testing Phase

We obtained recent data from *data.gov.sg* (April 2022 data) for our dataset S2 to be used for evaluation. Note that this only included towns and data that we were able to scrape from the SRX website.

Using our model obtained from part 2, we tested on the dataset and have obtained the following results:

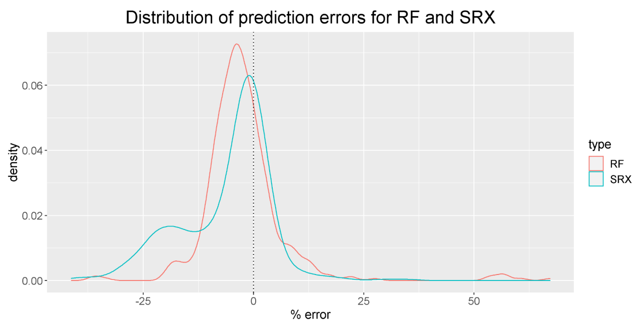

Figure 10: Distribution of errors of our RF model and SRX

Tabulated, the results have the following:

Model	**Ours (RF)**	**SRX**
Mean Percentage Error (Absolute)	6.66525 %	8.17477 %
Percentage of predictions within 10% of actual price	84.2%	69%

Discussion

Ethical Impact

One of our concerns for this project is the users' reliance on the price predictions of our model on resale HDBs. We bear the ethical responsibility of ensuring that our model remains accurate and up to date as users may use it as a benchmark. We plan to do this by displaying the accuracy of our model to inform users of the risks of referencing the predicted prices. We also intend to conduct regular updates to our model to include future transactions such that our model will always predict prices based on the latest transactions. This will help users to make informed decisions and negotiate for a fair price in the valuation of their HDBs.

Limitations

Our model also does not account for some external influences that affect the resale price. For example, the flat model DBSS (Design, Build and Sell Scheme) was built under private developers. Hence, despite having similar attributes (i.e. distance to gyms) as other input instances, the actual resale price is higher than expected, and hence our model produces a conservative prediction.

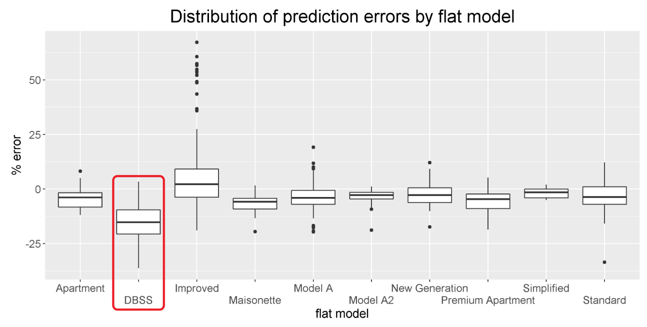

Figure 11: Distribution of errors by flat model

Furthermore, our training dataset has some limitations like not having the exact flat unit and floor number. If the model were to be used by real estate companies, we might have access to this data which may increase the accuracy of the model.

During data processing, we did not account for differences in timing between amenities and HDB resale. For instance, a HDB flat could have been sold before a certain amenity X was built or even announced, but we drew correlations between them during our data processing. The root cause of this is the lack of incremental amenity data. For future exploration, it can be beneficial to tag amenity data to a specified time to allow for a more accurate prediction of how amenities affect resale price.

Member Roles

Members	**Project roles (mainly)**
Josiah Lee	Project Lead *(Planning, Data Preparation, Data Visualization and Linear Regression)*
Er Song Heng	Member *(Technical tasks, Training models, Report writing)*
Derek Ng	Member *(Technical tasks, Web scraping, Data cleaning, Report Writing)*
Tan Jun Heng	Member *(Technical tasks, Training models, Report writing)*
Ong Han Yang	Member *(Technical tasks, Training models, planning, Report refinement)*

Reflections

Josiah:

I learnt how difficult yet important feature engineering is. The process of planning new features required creative inputs from all team members. Then, external data had to be sourced and merged and processed with our main dataset. This was a tough yet enriching experience.

Song Heng:

From this project, I have learnt that machine learning is not as straightforward as feeding data into the models. In fact, a lot of pre-processing and data manipulation is required before training the model. Also, I was surprised by the effectiveness of the Random Forest Regressor in predictions since I was under the impression that it was more of a classification based model.

Derek:

I was mainly assigned to web scraping of data and occasionally training the models. During the process of web scraping, I grew to appreciate the pains and struggles of having to extract large amounts of data. These pains include the time required to write a bot that handles exceptions smoothly for instance.. However, it was a valuable experience as I picked up a soft skill that will be useful for me in the future.

Jun Heng:

From this project, I realised the importance of data cleaning and data manipulation in machine learning workflow. Similarly, I didn't expect the tree-based regression model to beat more advanced models such as SVR which is surprising to me.

Han Yang:

This being the first time I've embarked on a ML project, I did not expect the extensiveness of the data cleaning and gathering required. I felt helpless sometimes as I had minimal knowledge on those areas. I learnt a lot about how the input to the models must fit within certain guidelines according to what has been trained. Particularly for categorical data, data processing had to be done to convert them into dummy variables. The actual training and testing portions of the project were comparatively much simpler and surprising to me as well, given the amount of theory that had gone into the formulation of the models.