ML Singapore! Project Report – Predicting Housing Prices in Singapore

Written by Team 24
Audrey Felicio Anwar (A0200761U), Austen Jeremy Sugiarto (A0200805W),
Dick Jessen William (A0200677H), Hubertus Adhy Pratama Setiawan (A0200816R),
Justin Tzuriel Krisnahadi (A0200794H), Mario Lorenzo (A0193781U)
National University of Singapore

Abstract

This project aims to train a machine learning model that can accurately predict the housing prices in Singapore, helping Singaporeans decide their housing given a certain budget. The effectiveness of several learning methods such as Random Forest, Gradient Boosted Trees, Artificial Neural Networks and Convolutional Neural Networks are tested and compared to achieve the best results.

Introduction

Singapore is the third country with the most expensive housing in the world. With this in mind, Singaporeans have to carefully consider whether to purchase a certain piece of property or not. They must also be mindful that property sellers may sell the property with a higher price, adding onto the already exorbitant rate associated with it.

However, people in general may not have sufficient knowledge to appraise the value of a particular piece of property. They might not be able to determine whether a house is worth the proposed price, overpriced, or even undervalued. Against this background, tools that could predict the price of a piece of property may come in handy. Property buyers or renters may use this tool to obtain a rough gauge of the property's price so that they have a better understanding on how the current trend is, which leads to better informed decision-making. Such a tool is also beneficial to investors who would like to maximize their profits in performing property transactions.

To construct a housing price predicting model, several learning techniques were used. In particular, ensemble learning methods such as Random Forest and Gradient Boosted Trees as well as deep learning methods such as Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) were chosen. The results from all the methods were then compared and analyzed.

Literature Review and Methodology

This section reviews various machine learning techniques, both ensemble learning methods and deep learning methods. Methods on finding the number of hidden layers and neurons are also covered.

Ensemble Machine Learning Models

Quang et al. (2019) suggested ensemble models for predicting housing prices. We decided to train three ensemble methods. The methods are Random Forest, Gradient Boosting Trees, and Stacking. To find the best parameters for these models, we used $GridSearchCV$ and $RandomizedSearchCV$ provided by $sklearn$.

Random Forest A Random Forest is a combination of numerous decision trees, where each tree differs slightly from the rest. Each tree is given a random vector sampled independently from the dataset with the same distribution, resulting in numerous trees that can learn the target concept in a slightly different manner. The idea behind this is to exploit the fact that a decision tree does a good job on predicting, but is prone to overfitting. Therefore, by combining numerous trees which overfit in different ways then averaging the result, the overall amount of overfitting can be reduced without loss of generalization power. For this project, $RandomForestRegressor$ from $sklearn$ was used.

Gradient Boosting Gradient Boosting techniques use numerous weak learners (e.g. shallow trees) and combine them into one strong learner in an iterative manner. The goal is to find an approximation that minimizes the empirical risk by applying steepest descent on each iteration. For this project, $GradientBoostingRegressor$ from $sklearn$ was used. On top of that, we also utilized Extreme Gradient Boosting $(XGBoost)$, which is a scalable gradient boosting system introduced by Tianqi Chen and Carlos Guestrin (2016). The system was built upon the traditional gradient boosting regression tree.

Stacking Stacking or Stacked Generalization is a method to combine multiple models fitting the data (level-0 models or base models) and a new meta-model (level-1 model) learning how to combine the predictions made by the base models. In this project, we used a package for stacking named $vecstack$ from GitHub. We built the most common architecture, which is 2-level stacking. The first level consists of Random Forest and Gradient Boosting, whereas the second level is $XGBoost$.

Deep Learning

According to Koleva (2020), Deep Learning is capable of dealing with data of high complexity. Therefore, we implemented two Deep Learning methods. The methods are Artificial Neural Network (ANN) and Convolutional Neural Network (CNN). We used the $Keras$ package from $Tensorflow$ to build the models and $RandomizedSearchCV$ from $sklearn$ to find the best parameters for both models.

Artificial Neural Network According to Wang (2003), Artificial Neural Network is a machine learning algorithm that is loosely modelled based on a human's brain that consists of an multiples layers of neurons: an input layer, several hidden layers, and an output layer. Each neuron in a layer is connected to every neuron in the adjacent layer, and each connection between neurons has a particular weight. The output of each layer is then passed to an activation function to introduce non-linearity into the neural network and to provide a bound for the value of the neuron. There are various types of activation functions, and one that is commonly used is the Rectified Linear Unit $(ReLU)$ activation function. It is defined as $ReLU(x) = max(0, x)$. Li Yu et al., (2017) concluded that $ReLU$ produced sparsity in the network, and hence reducing the degree of parameter independence.

Several researchers have attempted to use ANNs as their method to model a housing price predictor. Pai and Wang (2020) utilized several methods to predict real estate prices. Transaction data of real estates in Taichung, Taiwan was used as the dataset and 11 independent variables were taken into consideration. Several models were trained in the experiment, including Backpropagation Neural Networks (BPNN), and General Regression Neural Networks (GRNN). Mean Absolute Percentage Error (MAPE) was used to evaluate the performance of the model. Wang, et al. attempted to predict HDB prices using Artificial Neural Networks (ANN). Several intrinsic and extrinsic factors of public houses were taken into consideration. The results implied that ANNs are susceptible to overfitting.

Convolutional Neural Network Although the primary usage of Convolutional Neural Networks are for image-related tasks such as image classification and face recognition, it still performs magnificently on other machine learning problems (Albawi, Mohammed, Al-Zawi, 2017). The paper also states that another advantage of Convolutional Neural Networks is that it allows us to reduce the number of parameters on our model. Li Yu et al. (2018) gives an overview of a neural network consisting of a combination of an Input and Output Layer to take care of inputs and outputs of data, two Convolutional Layers as the main part of the architecture, a Pooling Layer to reduce overfitting, and a Full-connect Layer to flatten the matrix. Yu, et al. (2018) scraped housing price data in Beijing from several websites. They trained Convolutional Neural Networks (CNN), and Long-Short Term Models (LSTM). Both of them had significant errors due to several factors that could not be modelled such as macroeconomic and political factors.

Fixing the Number of Hidden Layers and Neurons Sheela and Deepa (2013) attempted to find an optimal number of hidden neurons in neural networks, and tested the result on an Elman network. The number of hidden neurons was proposed in form of the following equation: $N_h = \frac{4n^2+3}{n^2-8}$, where n is the number of inputs.

Data Collection, Analysis, and Preprocessing

This project used the data available online at https://www.srx.com.sg/. The data was collected by web scraping using the Python 3 $BeautifulSoup$ library. The following are the features scraped from the website:

- The year the property was built.
- The area of the property in square feet or square meters.
- The number of bedrooms in the property.
- The number of bathrooms in the property.
- The district area of the property (e.g., Chinatown, Geylang, etc).
- The type of the property (HDB, Condominium, or landed).
- The address of the property.
- The price of the property.

In total, we successfully scraped 33,036 data points. The features freshly scraped were not yet adequate. We improved on the features by performing the following:

1. Mapping addresses and district areas into latitude and longitude coordinates using Google's Geocoding API.
   In all our reference papers, latitude and longitude were used. We took inspiration from this, as latitude and longitude are more general features than address and district. Moreover, models tend to work better with numerical data than string data, such as addresses. One-hot encoding was not used for the addresses as they were unique. Lastly, by storing these coordinates we can generate other important features, such as the distance between the property to schools, since it is easy to compute the distance between two coordinates.
2. Standardizing the area of the property to square feet.
   Standardizing the dimensions allows the data to be consistent, which is crucial as models cannot distinguish units.
3. Acquiring the number of schools that are within 1 kilometer or 1 - 2 kilometers from the property address.
   Sumit et al. (2016) suggested that schools that were within 1 km as well as those within the 1 - 2 km zone affected housing prices in Singapore. Hence, our team collected the list of primary schools in Singapore from the MOE website, acquired their coordinates using Google's Geocoding API, and calculated their distance to each of the properties.

The features at this point can be seen in the two tables below.

Features	min	max	mean	std
year	1937	2035	2011.5	12.99
floor_area_sqft	23.969	2473.06	133.068	133.091
bedrooms	1	9	2.801	1.124
bathrooms	1	12	2.393	1.252
latitude				
longitude				
schools<1km	0	9	1.597	1.467
schools1-2km	0	15	3.878	2.496

Figure 1: Numerical Features

Features	Types	MC	LC
nearest_school	174	Cantonment	Admiralty
type	3	Condominium	HDB

Details: **MC**: Most Common, **LC**: Least Common

Figure 2: Categorical Features

Before we analysed the data, we performed data cleansing which consisted of:

- Removing rows that contained NaN for numerical features.
- Removing rows that contained an empty string in either "nearest_school" or "type" columns.
- Removing rows that contained 0 in either "year", "floor_area_sqft", "bedrooms", "bathrooms", "latitude", or "longitude" columns.

We considered using data imputation, but it might add noise rather than help our models to learn. As we had quite a lot of data, we performed removal instead.

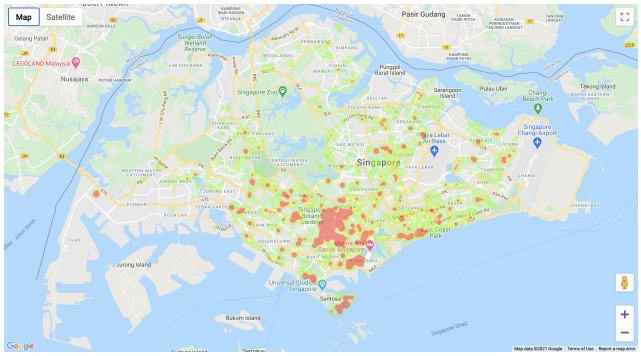

Figure 3: Heatmap of property prices in Singapore

On figure 3, the color is correlated to the property price. The redder the color of an area is, the more expensive the properties in it are. We can clearly see from the map that the area in the middle, which corresponds to the area between Singapore Botanic Gardens and Central Area, are redder compared to other regions of Singapore.

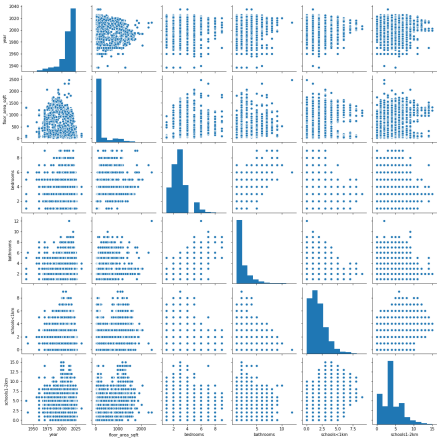

Figure 4: Pair Plot from features

From figure 4, we can see that most features had very small correlations with other features. Hence, all attributes were necessary for training our models.

Before training the models, we did additional preprocesssing. Shanker et al. (1995) suggested that normalizing continuous features increased the accuracy and the training time of neural network models. Another reason for normalization is that we wanted the value of the continuous features to lie near zero. As most activation functions, such as sigmoid and tanh, are linear near zero, the models will be less complex (satisfying Occam's Razor).

As mentioned previously, models work better with numerical data, so the remaining string data had to be converted. We used one-hot encoding instead of ordinal encoding due to the fact that the gap between each category might differ. For instance, in the "type" column, the gap between "HDB" to "Condominium" might differ from the between "HDB" and "Landed".

In the end, after implementing z-score normalization $\frac{x-\mu}{\sigma}$ for continuous features and one-hot encoding for categorical features, the final dataset consisted of 21,800 data points with 185 input features as well as 1 target feature, which is the price. In the next 2 sections, we compare the results of conventional ensemble machine learning models with deep learning models. For consistency, we split the data into training and test data with a ratio of $85:15$ and our metrics are Mean Absolute Error (MAE) and R^2 score.

Results

Random Forest

The parameters chosen for $RandomForestRegressor$:

- n_estimators : 4384
- max_depth : 21
- max_features : auto

The results of $RandomForestRegressor$:

- Validation MAE : 144659.1759689292
- Validation R^2 : 0.9835603299231612

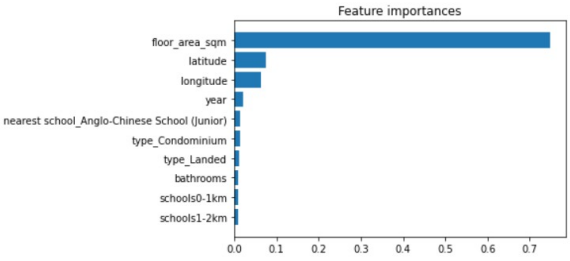

Figure 5: Important Features for Random Forest

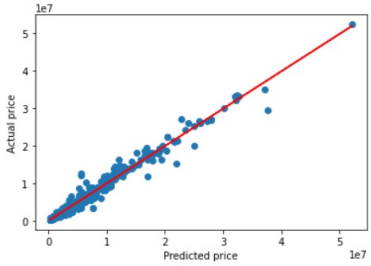

Figure 6: Prediction of Random Forest on test set

Gradient Boosted Trees

The parameters chosen for $GradientBoostingRegressor$:

- learning_rate : 0.13278489586128683
- loss : lad
- max_depth : 15
- n_estimators : 1064
- sub_samples : 0.9649398241605084

The results of $GradientBoostingRegressor$:

- Validation MAE : 138143.0657046704
- Validation R^2 : 0.9808268311629361

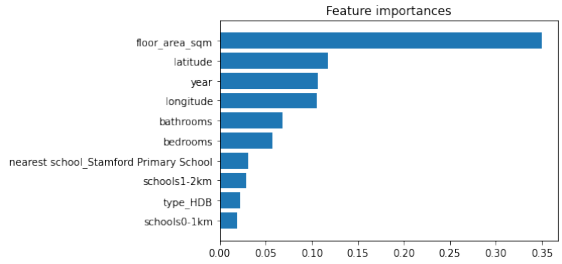

Figure 7: Important Features for Gradient Boosting

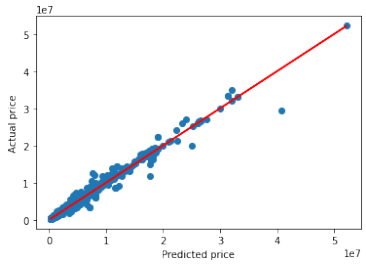

Figure 8: Prediction of Gradient Boosting on test set

The parameters chosen for $XGBoostRegressor$:

- learning_rate : 0.19930336770013027
- loss : lad
- max_depth : 9
- n_estimators : 2300
- sub_samples : 0.9276548808002751

The results of $XGBoost$:

- Validation MAE : 113933.71281536698
- Validation R^2 : 0.9853491501170271

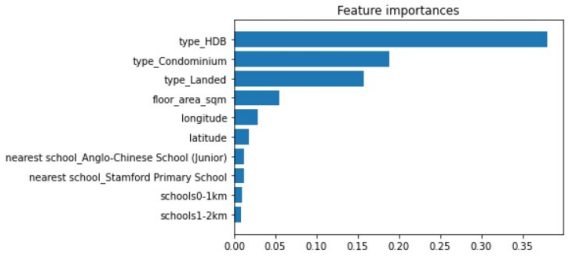

Figure 9: Important Features for XGBoost

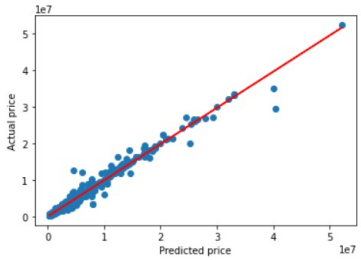

Figure 10: Prediction of $XGBoost$ on test set

Stacking

The results of $Stacking$:

- Validation MAE : 144431.05718654435
- Validation R^2 : 0.9769990624005662

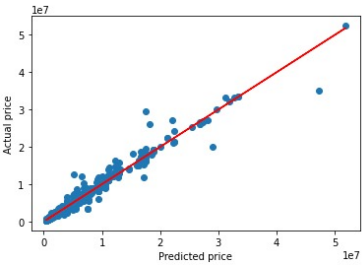

Figure 11: Prediction of Stacking on test set

Artificial Neural Network

Inspired by Wang et al. (2016), we decided to implement a 3-layer Neural Network. After some experimentation with the 3-layers neural networks, the model stuck at a very bad R^2 result. Hence, we decided to add another hidden layer. On

top of that, we used $Xavier$ initialization for the weights to help our model to converge faster as suggested by Glorot (2010). To prevent overfitting, we used early stopping to stop the training process if the loss doesn't drop for the last 100 iterations. Lastly, our activation function that we use is $ReLU$. The summary of the model can be seen below:

Layer (type)	**Output Shape**
Layer 1 (Dense)	(None, 4096)
Layer 2 (Dense)	(None, 4096)
Layer 3 (Dense)	(None, 4096)
Layer 4 (Dense)	(None, 1)

Figure 12: ANN architecture

After training the model for 5000 epochs, the results of our ANN model are as follows:

- Validation MAE : 184408.32780963302
- Validation R^2 : 0.9588249302804152

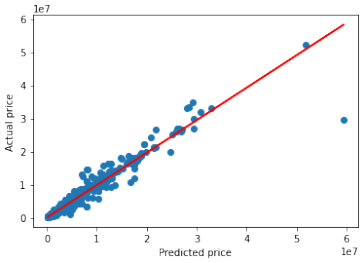

Figure 13: Prediction of ANN on test set

Convolutional Neural Network

Here is the summary of the final version of the model, which can also be seen with the model.summary() command in Python 3.

Layer (type)	**Output Shape**
main_input (Input Layer)	(None, 14, 14,1)
conv2d (Conv2D)	(None, 12, 12, 3)
activation (Activation)	(None, 12, 12, 3)
max_pooling2d (MaxPooling2D)	(None, 6, 6, 3)
conv2d_1 (Conv2D)	(None, 4, 4, 3)
activation_1 (Activation)	(None, 4, 4, 3)
max_pooling2d_1 (MaxPooling2D)	(None, 2, 2, 3)
flatten (Flatten)	(None, 12)
dense (Dense)	(None, 2048)
dropout (Dropout)	(None, 2048)
dense_1 (Dense)	(None, 2048)
dropout_1 (Dropout)	(None, 2048)
dense_2 (Dense)	(None, 2048)
dropout_2 (Dropout)	(None, 2048)
dense_3 (Dense)	(None, 2048)
dense_4 (Dense)	(None, 1)

Figure 14: Architecture for the CNN model

After training the model for 5000 epochs, here are the results:

- Validation MAE : 229640.284
- Validation R^2 : 0.967

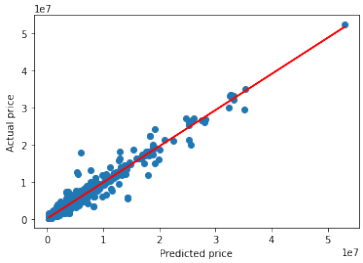

Figure 15: Prediction of CNN on test set

Comparison and Analysis

The results of all the methods used in the experiment is shown in Figure 16 below:

Method	**MAE**	**R^2**
Random Forest	144659.176	0.984
Gradient Boosting Regressor	138143.066	0.981
XGBoost Regressor	**113933.713**	**0.985**
Stacking	144431.057	0.977
Artificial Neural Network	184408.328	0.959
Convolutional Neural Network	229640.284	0.967

Figure 16: MAE and R^2 Score for all the trained methods

From figure 16, based on the score, $XGBoost$ performed the best compared to the other models. The top 3 most important features for $RandomForest$ and $GradientBoosting$ were area, latitude, and longitude. For $XGBoost$, however, the top 3 were type_HDB, type_Condominium, and type_Landed. According to figure 17, the most expensive type is Landed and the cheapest is HDB. $XGBoost$ had successfully captured this underlying distribution, thus producing better results. This comes as no surprise, as $XGBoost$ has been empirically proven to yields state-of-the-art results (Chen & Guestrin, 2016).

In this experiment, Neural Network based models failed to perform as well as we had thought. This can be attributed to some factors. Firstly, this can be caused by redundancy factors, according to Ghamesi et al. (2018). This happens as longitude and latitude have some sort of correlation with the number of schools. Secondly, Ghamesi also states that Deep Learning methods are most suitable in tasks involving lots of data. Here, the number of data points (21,800) is not very large, and this hinders the chance for neural network algorithms to shine.

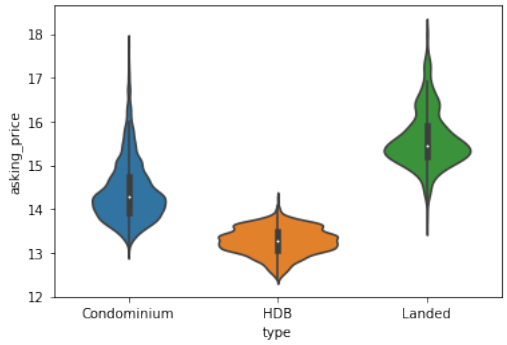

Figure 17: Price vs Property Type

Conclusion

From these experiments, we see that the state-of-the-art machine learning methods such as $XGBoost$, Neural Networks and Random Forest can produce a good model to predict housing prices in Singapore. We also learned to use techniques that can prevent overfitting such as Early Stopping and Dropout Layers to improve performance. Some possible ideas to continue this project are to find more data to improve accuracy and to diversify the input data to be able to generalize to housing outside of Singapore. We conclude that Machine Learning can indeed help Singaporeans predict housing prices conveniently with just a few features.

Acknowledgements

We would like to thank Assoc. Prof. Bryan Low and the entire CS3244 teaching team for making this project possible.

References

Quang, et al. 2019. Housing Price Prediction via Improved Machine Learning Techniques. Dept. of Computer Science, Texas Christian University, Forth Worth, United States.

Breiman, Leo. 2001. Random Forest. Statistics Department, University of California Berkeley.

Wolpert, David H. 1992. Stacked Generalization. Complex System Groups, Theoretical Division, and Center of Nonlinear Studies, Los Alamos.

S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey.

Wang, et al. 2016. Predicting Public Housing Prices Using Delayed Neural Networks. IEEE Region 10 Conference (TENCON)

Wang SC. 2003 Artificial Neural Network. In: Interdisciplinary Computing in Java Programming. The Springer International Series in Engineering and Computer Science, vol 743. Springer, Boston, MA. https://doi.org/10.1007/978-1-4615-0377-4_5

Glorot X. and Bengio Y. 2010 Understanding the difficulty of training deep feed-forward neural networks. InAistats, volume 9, pages 249–256.

Fahimeh Ghasemi, Alireza Mehridehnavi, Alfonso Perez-Garrido and Horacio Perez-Sanchez. 2018 Neural network and deep-learning algorithms used in QSAR studies: merits and drawbacks.

Chen, T., &; Guestrin, C. (2016). Xgboost. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. doi:10.1145/2939672.2939785

Appendix

Roles and Responsibilities

- Audrey Felicio Anwar: Web scraping, data cleaning, data preprocessing, Google API setup, report writing and proofreading.
- Austen Jeremy Sugiarto: ANN experimentation, web scraping, data preprocessing, data visualization, data cleaning, report writing.
- Dick Jessen William: CNN and $GradientBoostingRegressor$ experimentation, literature review for deep learning methods, report writing.
- Hubertus Adhy Pratama Setiawan: $RandomForestRegressor$ and $XGBoost$ experimentation, report writing and proofreading.
- Justin Tzuriel Krisnahadi: Stacking experimentation using $RandomForestRegressor$, $GradientBoostingRegressor$, and $XGBoost$, report writing and proofreading.
- Mario Lorenzo: Literature review for deep learning methods, ANN experimentation, report writing.