DMBA ASSIGNMENT

**AY 2021/2022 Oct Semester**

**DECLARATION**

I declare that I am the originator of this work and that all other original sources used in this work have been appropriately acknowledged.

I understand that plagiarism is the act of taking and using the whole or any part of another person’s work and presenting it as my own without proper acknowledgement.

I also understand that plagiarism is an academic offence and that disciplinary action will be taken for plagiarism.”

I Agree

|  |  |
| --- | --- |
| **My Information** | |
| Name (as in matriculation card) | Edzran Bin Hisham |
| Admin Number | 2004986B |
| Practical Group (e.g. P01) | P03 |
| I am submitting \_\_\_\_\_ level work | Advance |

Yes, I would like to receive qualitative feedback for my submitted work. I understand that my teacher will only be providing feedback and not grades or marks.

**Background**

In this assignment, we will make use of the HDB resale data retrieved from Data.Gov.Sg. No modifications were made to the datasets. We will make use of the data to train predictive models that can satisfactorily predict resale prices for given HDB units.

**About the data**

I will be working with the following five datasets:

* resale-flat-prices-based-on-approval-date-1990-1999
* resale-flat-prices-based-on-approval-date-2000-feb-2012
* resale-flat-prices-based-on-registration-date-from-mar-2012-to-dec-2014
* resale-flat-prices-based-on-registration-date-from-jan-2015-to-dec-2016
* resale-flat-prices-based-on-registration-date-from-jan-2017-onwards

The datasets contain information about HDB flats transacted on the resale market and other information such as the flat type, flat model, flat size, transacted price as well as lease commencement date.

Link to data: <https://data.gov.sg/dataset/resale-flat-prices>

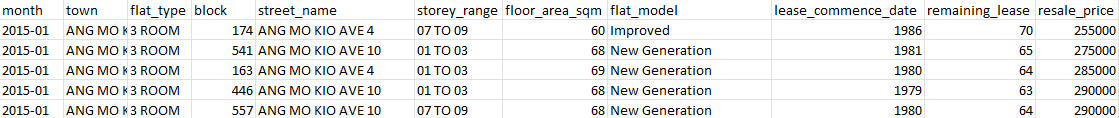
# Data Exploration and Preparation

**Missing remaining\_lease columns**

Taking a look at each dataset in excel, I can see that datasets:

* resale-flat-prices-based-on-registration-date-from-jan-2015-to-dec-2016
* resale-flat-prices-based-on-registration-date-from-jan-2017-onwards

have an extra column – remaining\_lease. Which is missing from the other datasets.

****

Remaining lease calculates the limited period remaining on a HDB lease, depending on their age. For example, an HDB flat built-in 1980 has 59 years remaining on its lease in 2020. According to Singapore law, after 99 years, the flat comes back to the state.

Generally, older resale HDB flats have lower market values because once the lease expires, the property is surrendered to the government making it invaluable. However, it's not always that straightforward; older resale HDB's are attractive to buyers looking for properties in mature estates. These buyers are willing to pay more for these properties.

With that being said, the remaining\_lease column could be a useful feature in predicting our target (resale prices) and thus I will try to derive a column calculating the remaining lease in the other datasets as well.

1. Firstly calculate new “year” column that only contains month.

The new *year* column will replace old *month* column. This is so I can calculate *remaining\_lease* column easier. Year column also helps generalize our data more.

Table

Description automatically generated

1. Formula to calculate remaining lease: 99 - (*year – lease\_commence\_data)*

*Table

Description automatically generated*

From above, I have calculated new *remaining\_lease* column in one of the dataset that did not originally contain it. I will repeat this process and calculate *remaining\_lease* column for the rest of the datasets as well.

**Remaining lease column from string to numerical**

In file “*resale-flat-prices-based-on-registration-date-from-jan-2017-onwards*” I can see that the data are in nominal format. This is further clarified from the automatic assignment of data type in SAS EM as shown:

****

As our models that we will be using requires only numerical inputs, I will transform this column into numerical. One way I can achieve this is by calculating the number of total years left instead of years and months. This is to also standardize with the values of remaining\_lease column in “resale-flat-prices-based-on-registration-date-from-jan-2015-to-dec-2016” which values are in years:

*Jan-2017-onwards file Jan-2015-to-dec-2016 file*

Graphical user interface, text, application, chat or text message

Description automatically generated Table

Description automatically generated

I can do this by using the same formula used previously to calculated *remaining\_lease*.

Formula to calculate remaining lease: 99 - (*year – lease\_commence\_data)*

*A picture containing graphical user interface

Description automatically generated*

**Merge all 5 datasets**

Now that all 5 separate datasets contain the same types of columns, I can merge them together. I can achieve this using python in jupyter notebook

1. Import packages and set working directory which contains all the files

Logo, company name

Description automatically generated

1. Use glob to match the pattern ‘csv’



1. Combine all files in the list and export as CSV

A picture containing text

Description automatically generated

After combining all 5 datasets together, my dataset now contains 854,132 rows.

**todo list**

* Merge all 5 datasets together
* Do some research on factors that could affect house prices
* Drop block column
  + Reasoning: Not useful. Higher block number does not necessarily mean higher prices. Block number can be repeated and used in other town areas.

**Data Exploration**

* Missing values?
* Duplicates?
* Outliers?

**Feature Engineering**

* Price/sqm
* Distance to mrt
* Time to walk to nearest mall

**Deriving new column *price/sqm.***

Why calculate *price/sqm*? It’s the easiest way to compare apple to apples, especially when we take into account the underlying value of the property itself in a specific market. It is an indicator used by investors to determine the value of the property, and the ability of an investor to add value. Thus calculating *price/sqm* could be a useful feature in predicting home prices.

Formula for *price/sqm*: *resale\_price / floor\_area\_sqm*

Table

Description automatically generated

**Distance to Mrt**

It is common for a property’s value affected by its proximity to an MRT station: in general, the closer your home is to an MRT station, the higher its value. This relationship plays directly into the idea that location is key when it comes to property valuation. Thus, I would include this as a feature in my dataset.

Research source: <https://www.propertyguru.com.sg/property-guides/mrt-effect-on-property-prices-39498>

1. Derive new *address column* that can be inputted into OneMap API

*Address* column will contain the combination of block and street name.

This can be easily achieved in Excel using the formula shown below:Table

Description automatically generated

After acquiring the addresses of all the datapoints, I will delete all other columns only leaving address column which would be inserted into onemapsg API later on. I will save this file as a new file – *address.xlsx.* This file contains all the addresses from 1990-2021 basically.

1. Extract longtitude and latitude from onemap.sg

Next I will write code to extract the coordinates of each address. I can do this by making use of onemapsg API. This API provides searching of address data for a given search value. It returns search results with both *latitude*, *longitude* and *x*, *y* coordinates of the searched location.

Graphical user interface, text, application

Description automatically generated

Firstly I will import all necessary modules. Set the current working directory to where my *address.xlsx* dataset is stored. Open a new worksheet and set sheet to be set to the first sheet in the workbook(sheet1).

The for loop will extract each address from the workbook and store the result in variable *query\_address.*

Text

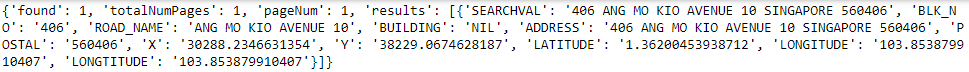
Description automatically generated with medium confidence

Next I will pass in results in *query\_address* into the query string as a parameter for the API call. Then store the JSON result as a Python object in *data* variable.

Graphical user interface, text, application

Description automatically generated

Next, I will try storing the query results in different columns in my excel sheet via the codes under “try:”. If there is no information on a particular address, API will return a null result which would then print “error”, then continues calling the API with the next query address. A successful search query result is shown below:



However one issue I encountered was that the entire process can take a very long time. In *address.xlsx*, I have 854133 rows of data! This means I will have to call the API 854,133 times which would take a while. I have also noticed that the search results also get progressively slower.

****

In view of time constraint, I will extract a subset from *address.xlsx*. Instead of extracting *lat* and *long* information from entire dataset which contains 800k rows. I will take a subset of it and extract information from only latest 5 years of resale data.

Graphical user interface

Description automatically generated with low confidenceUsing the excel filter function, I will filter only data from 2017-2021 and delete the rest of data.

I am now left with 113,960 rows of data which I feel is still plenty enough for me to perform modelling and analysis with. If I had more time, I would love to continue this project and figure out a way I can extract all *lat* and *long* info from whole 800k rows of data better and create models using the entire dataset from 1990.



Using the dataset of addresses from 2017-2021, and extracting information using onemap.sg API still took awhile but managed to complete. As I only want lat and long information, I will delete the rest of the columns that contained other information from the search query.

The results are shown below(first few rows):

Graphical user interface, text

Description automatically generated

1. Find out MRT stations and their respective coordinates

Doing a quick google search, I found a dataset that contains all MRT station names and their respective coordinates as of Dec 2020.

Table

Description automatically generated

[*https://data.world/hxchua/train-stations-in-singapore*](https://data.world/hxchua/train-stations-in-singapore) *(MRT coordinates dataset)*

I will only keep *STN\_NAME, Latitude & Longitude*.

1. Compute MRT distance between address and closest MRT station

I can achieve this by creating a function in jupyter notebook. Firstly I will load in the 2 datasets – *address-2017-2021-csv* (addresses and respective lat & long coordinates) and *mrtsg*(mrt station names and respective lat & long coordinates)

Text

Description automatically generatedText

Description automatically generated

Next, create a function that will compute distance between lat and long coordinates of addresses and closest mrt stations and apply the function:

Text

Description automatically generated

1. Finally, export the results into csv – *address\_to\_nearest\_stn\_dist.csv*:



First few rows in exported csv - *address\_to\_nearest\_stn\_dist.csv*:

Table

Description automatically generated

Finally, I can now append the new column to our original dataset(*resale-priced-combined-2017-2021*) which contain all the other columns:

*A picture containing chart

Description automatically generated*

**Distance to nearest shopping mall**

Living near a shopping mall can be a convenience which also comes with a higher price tag due to good location. Thus, it may be a good feature to analyze and thus I will include it in my dataset.

Research source: <https://www.propertyguru.com.sg/property-guides/living-near-shopping-malls-in-singapore-37517>

1. Get full list of Shopping Malls in Singapore

To get started, I have found a list of all the shopping malls available in Singapore from Wikipedia. (<https://en.wikipedia.org/wiki/List_of_shopping_malls_in_Singapore>). I then copy and pasted all the Shopping Mall information into an empty excel file as shown:

Graphical user interface, text, application, chat or text message, email

Description automatically generatedThere are a total of 173 shopping malls available in Singapore.

1. Find coordinates for each shopping mall

I can use the same python code previously used to search for information on addresses, to search for information on the shopping mall list I have just created. I will only tweak the excel workbook that will be used which is now *shopping\_mall.xlsx.*

**

And ensure that I save into the same excel workbook:



Here are the results:

Table

Description automatically generated with medium confidence

I can see that the onemap API could not retrieve data for some shopping malls. Thus I will manually google the coordinates for the observations with missing information. Some shopping mall coordinates cannot be retrieved as they are rebranded into a different name. For example: “PoMo” mall is now “GR.ID” mall.



Text

Description automatically generated

Shaw House and Shaw Centre are 2 different malls but connected. However, if inputted into onemap API together as shown below there will be no results. For this case I will just take the coordinates for either one as they are both connected to each other.



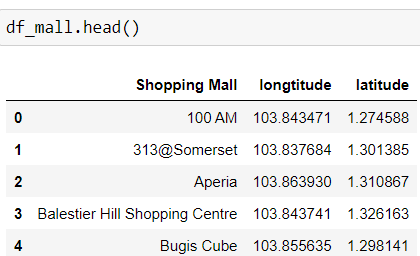
After manually inputting the missing values, I will delete all other columns, leaving only the shopping mall names and respective coordinates:

Table

Description automatically generated

1. Compute distance from address to nearest shopping mall

I will load in the 2 datasets – *address-2017-2021-csv.csv* (addresses and respective lat & long coordinates) and *shopping\_malls.csv* (mrt station names and respective lat & long coordinates).

Graphical user interface, text

Description automatically generated

Next, use the previously created functions to compute distance:

Text

Description automatically generated

1. Finally, export the results into csv – *address\_to\_nearest\_mall .csv* and merge with full dataset

After about 2-3hrs.. the calculation for each address to nearest mall is done. I can export the data into csv into my local machine using the following code:



Results:

Graphical user interface, text, application

Description automatically generated

Finally, I can now append the new column to our original dataset(*resale-priced-combined-2017-2021*) which contain all the other columns by copy and pasting the entire column to my dataset which contains all other columns:

Table

Description automatically generated

Values are rounded up to 2 decimal places for simplicity.

**Region**

Generally, properties in Central Business District regions tend to fetch higher resale prices than properties in neighbourhood regions due to the location.

1. Extract region data from <https://en.wikipedia.org/wiki/Planning_Areas_of_Singapore>

Table

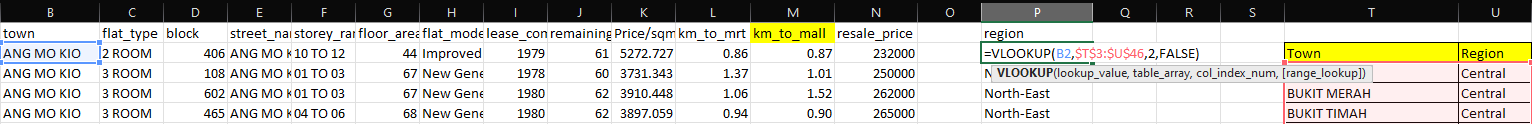
Description automatically generated

I will manually extract the names of each planning area and their respective region into an excel worksheet and create a VLOOKUP table as such:

Table

Description automatically generated *Half of the VLOOKUP table containing Town and Region data*

1. Create new “Region” column and populate column using VLOOKUP table



Using the VLOOKUP() function, I set the lookup value to be observations in “Town” column, the table\_array to be the created VLOOKUP table containing town and each respective region, column index to be the 2nd and FALSE for exact match.

I then populate the entire column by copying the formula down.

1. Manually fill in NA values

Table

Description automatically generated

Filtering the column to only include NA values, I discovered that certain towns contain NA values for Region column as they do not have respective data to be referenced to in the VLOOKUP table:

CENTRAL AREA



KALLANG/WHAMPOA



I will add 2 new rows into my VLOOKUP table to ensure that these towns have data to be referenced to:

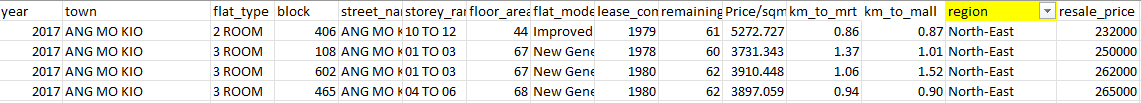


After repeating the steps again and ensuring I include the two extra rows in the VLOOKUP function, I can populate the column again. This time I have no more NA values:

Text

Description automatically generated

Here is the result:



Derived

* price/sqm
* km\_to\_mrt
* km\_to\_mall
* regions
* remaining\_lease for few datasets
* number of top schools in area

**Top secondary schools in area within 2km distance**

Living near top schools also be a factor in value of resale prices. Families may live closer and could pay higher prices for the convenience of enrolling their child into a top school for better education. Thus, it is one factor I could include in my dataset.

Research source: <https://www.propertyguru.com.sg/property-management-news/2016/5/124914/are-properties-near-famous-schools-more-expensive>

1. Download all dataset containing all schools in Singapore

I will first download the linked dataset from data.gov.sg which contains general information of all primary and secondary schools in Singapore. Most important information I want is the school names for geocoding later.

Link: <https://data.gov.sg/dataset/school-directory-and-information>

Graphical user interface, text, application

Description automatically generated

1. Load dataset into Jupyter Notebook and extract school name

I will take all values in column “school\_name” which are all the names of primary and secondary schools in Singapore and store it in variable – school\_name

Code: Result:

Text

Description automatically generatedText

Description automatically generated

1. Extract coordinates from onemap API

I can then input all school names into onemap API which will return results of various information. The information I will focus on is the latitude and longitude. Here is an example of the return result:

Graphical user interface, text, application, email

Description automatically generated

As I have many school names to be inserted into onemap API, I will automate a task by creating a for loop:

Graphical user interface, text, application, email

Description automatically generated

Results are stored in variable school\_geoloc.

1. Cleaning school name

Text

Description automatically generated These are the school names that did not return any results when inputted into the API. Thus these school names were not stored in variable school\_geoloc.

This can be seen by creating a set and subtracting both sets for school names in school\_data and newly created variable school\_geoloc.

The errors that I identified that caused it was when school names include:

Fullstop(.) Commas(,)

Graphical user interface, text

Description automatically generated Text

Description automatically generated

Thus, I will clean the school names column and repeat the geocoding for loop again.

school\_data['school\_name\_clean'] = [val.replace("St.","St").replace("’s","").replace("s’","").replace(".","").replace(",","") for val in school\_data['school\_name']]

school\_data['school\_name\_clean']

Result:

Graphical user interface, text, application

Description automatically generated

1. Text, table

   Description automatically generatedGet distance between all addresses and schools

Import file used previously (address 2017-2021).

I will also remove any duplicated address which as there are same addresses over the years in the dataset.

Graphical user interface, text, application

Description automatically generated

Next I will create a for-loop function to calculate distance to nearest school in for each address, for each school. Then export to csv as checkpoint.

Code: Results in csv:

Graphical user interface, text, application, chat or text message

Description automatically generatedText

Description automatically generated

1. Calculate top schools within 2km of each address

According to this website: <https://schoolbell.sg/secondary-school-ranking/#Express_O-Level_Secondary_School_Ranking>

* Extract the top 10 secondary school’s name according to rank and insert into an excel sheet.
* Clean school name using previous code
* Insert the cleaned top 10 secondary school names into onemap API to extract coordinates

And finally, calculate new column called *num\_top\_school\_2km:*

* Firstly, import the top 10 secondary school names into the jupyter notebook
* Create a new variable *flats\_with\_top\_school* which will store the results from a dataframe which satisfy 2 conditions:

1. 
2. 

* Lastly, name the 2 new columns in variable *flats\_with\_top\_school* as “address” and “num\_top\_school\_2km”:



\*Note *flat\_sch\_distance\_df* was previously made and contains distance to nearest school in for each address, for each school name. \*

1. Merge column back to original dataset

* Import original dataset(*resale-priced-combined-2017-2021.csv*) into jupyter notebook
* Merge new *num\_top\_school\_2km* column with original dataset based on “*address”* column

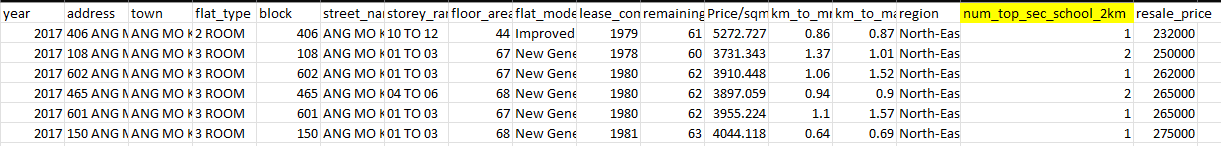


* Fill any NA values with 0



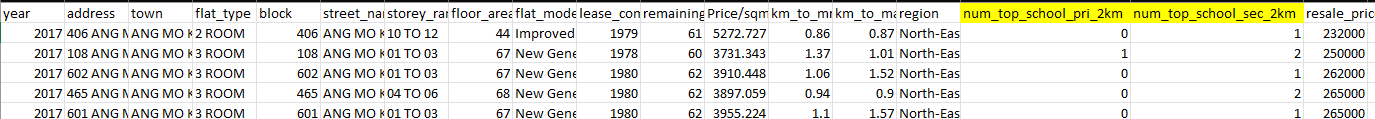
* Export final dataset as csv:

Result:



Note that this column is representing number of **top secondary schools within 2km radius**. Thus, I will repeat the steps above with **top 10 primary school names** to calculate another column representing primary schools.

Final:



# Data Preprocessing

To set a baseline score which can be compared to, I will be performing modeling on the dataset without any new engineered and supplemented columns. However Linear Regression model can only take in numerical values, thus these are the preprocessing methods I will use for each feature:

**One-hot encode**

Although linear regression does not get affected from the *Curse of Dimensionality*. I will still only One-Hot encode features that do not have high number of unique values for better interpretability of my features later.

One-hot encoding using *pd.get\_dummies* in python.

* Town
* Flat\_model

**Ordinal encode**

Ordinal encoding is performed when features have a natural ranking order. Performing ordinal encoding will help machine learning model to properly interpret these features.

* Flat\_type
  + '1 ROOM':0, '2 ROOM':1, '3 ROOM':2, '4 ROOM':3, '5 ROOM':4, 'MULTI-GENERATION':5, 'EXECUTIVE':6
* Storey\_range
  + Based on this article <https://tinyurl.com/ycxyuhwu> . Higher floors tend to fetch higher prices. Thus higher floor, higher natural ranking.

**Drop column**

I will either drop columns that I feel are not insightful based on domain knowledge or have too many unique values(hundreds to thousands)

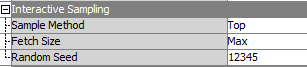
* Block
* Street\_name
* Address

# Modeling (benchmark)(Linear Regression)

In this segment, I will be performing modeling on SAS EM. One

Graphical user interface, text, application, chat or text message

Description automatically generated

Preferences

Fetch Size: Max

File node settings

* Target: resale\_price
* Input: All other features
* Role: Train
* Name Row: Yes
* Delimiter: (,)

A picture containing background pattern

Description automatically generatedData Partition settings

* Train: 70%
* Test: 30%
* Random Seed: 12345 (for repeatability)

Regression Node settings

Regression Type: Linear Regression

# Metrics to be used in evaluating Regression problem

**Adjusted R Square**

R Square is calculated by the sum of squared of prediction error divided by the total sum of the square which replaces the calculated prediction with mean. R Square value is between 0 to 1 and a bigger value indicates a better fit between prediction and actual value.

Compared to R Squared which can only increase, **Adjusted R Squared** has the capability to decrease with the addition of less significant variables, thus resulting in a more reliable and accurate evaluation. Thus, I will use Adjusted R Squared as the metric to focus on when evaluating my models.

**RMSE (Root Mean Square Error)**

MSE(Mean Squared Error) gives you an absolute number on how much your predicted results deviate from the actual number. However, the value given can be too big to be interpreted. Therefore, I will use RMSE.

Root Mean Square Error(RMSE) is the square root of MSE. It is used more commonly than MSE because firstly sometimes MSE value can be too big to compare easily. Secondly, MSE is calculated by the square of error, and thus square root brings it back to the same level of prediction error and makes it easier for interpretation. In our case, RMSE will show the average value of how far off our model predicted the resale price to be.

Example:



Baseline Linear Regression model results:



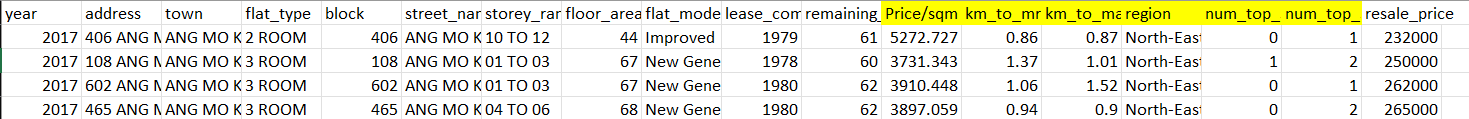
Takeaway: 0.8621 Adjust R-Square is quite good for the model without any new features. This will be the benchmark score to be beaten.



Takeaway: Delving deeper, I can see that the model RMSE value was $59052.20 for training set and $59020.66 for testing set. In the context of predicting HDB resale price values, it is not too bad. However, if this model was to be deployed and actually used by a company to help families to plan their finances for buying homes, $50k difference could be a lot of money to some families. Thus, let me see if the newly created features will help in improving my model in the later section.

# Preprocess new features/columns

Now I will try to create a similar model using the dataset with new features in hopes of outperforming the benchmark model created. However, before I can start performing machine learning on the new dataset, I will need to perform preprocessing on the new features. \*New features are highlighted\*



**One-hot encode column “region”**

Only 1 new feature would need to be transformed. I will need to convert the feature(*region*) to a numerical value. As the column only has a small number of unique values and region normally does not a have natural ranking, I will perform one-hot encoding on the column in python to allow the feature to be fed into a Linear Regression model. Results after one-hot encoding region shown below:

Background pattern

Description automatically generated with low confidence

**Normalizing features**

Why normalize data?

* My dataset contains features with different ranges.
* When we do further analysis, like multivariate linear regression, for example, the attribute with larger range will intrinsically influence the result more due to its larger value. But this doesn’t necessarily mean it is more important as a predictor.
* Therefore, Normalization helps to bring the values of numeric columns in the dataset to a **common scale**, without distorting differences in the ranges of values.
* Normalization scales each input variable separately to the range **0-1**, which is the range for floating-point values where we have the most precision.

I can perform normalization using python. The steps taken are as such:

1. Import MinMaxScaler from sklearn.preprocessing package
2. From pandas import DataFrame (to convert normalized array into a dataframe for us to export to csv)
3. Split dataset into x(features) and y(target)
4. Apply normalization using MinMaxScaler onto x dataset which only contains the features as such

Text, letter

Description automatically generated

1. Merge x and y back together and import as csv.

Here is a screenshot of part of the final dataset:

Table

Description automatically generated

All features are now normalized with a common scale of 0 to 1, eliminating any biasness.

# Modeling with new features/columns(Linear Regression)

In this segment, I will be performing the same Linear regression modeling using the dataset that contains the newly engineered and added features. Then later compare the results with the benchmark results. I will also further different selection models for linear regression such as:

1. Backward
   1. Fit a full model by slowly remove terms one at a time, starting with the term with the highest *p*-value.
2. Forward
   1. Instead of starting with a full model, we start with a model containing only the intercept. Then we slowly add terms to the model, one at a time, starting with the predictor with the lowest *p*-value. This continues until all the remaining terms that are not included in the model are above a specified *p*-value threshold.
3. Stepwise
   1. Method that iteratively examines the statistical significance of each independent variable in a linear regression model. The forward selection approach starts with nothing and adds each new variable incrementally, testing for statistical significance

For all Regression nodes, I will set the following settings:

* Regression Type: Linear Regression
* Target: resale\_price
* Input: All other feature

And finally, assess and determine the best model using Model Comparison node.

Model Comparison node settings:

* Selection Statistic: Mean Squared Error
* Selection Table: Test

Outline of diagram shown below:Diagram

Description automatically generated

In the Model Comparison node, I can see that the champion model is the regular regression node without any selection model method.

Table

Description automatically generated

Diving deeper, under Fit Statistics I can see that this model was chosen as the champion model as it has the lowest Mean Square Error(MSE) and Root Mean Square Error(RMSE).

Graphical user interface, text, application, chat or text message

Description automatically generatedText, table

Description automatically generated with medium confidenceThe RMSE using the new dataset with new features is quite good as it tells me that there is an average deviation of $23514.01 from the actual value vs predicted value.

That it almost a decrease of 50% from the RMSE value of the benchmark model. Thus, using the dataset with new features has improved our performance.

Linear Regression Model performance with new features:

Adjusted R-square: 0.97

Table

Description automatically generated

RMSE:

* Training set: 23708.48
* Testing set: 23514.01



Takeaway:

Performing linear regression using the dataset which included new features has significantly improved the overall performance. Adjusted R-Square has increased from 0.86 -> 0.97 while RMSE has decreased from 59020.66 -> 23514.01.

Effects Plot & Score Rankings Overlay(mean predicted of resale\_price)

|  |  |
| --- | --- |
| Chart  Description automatically generated with medium confidence | Chart  Description automatically generated |

Top 3 influential features with highest coefficient:

1. Price\_sqm (engineered feature)

Price\_sqm feature has the highest absolute coefficient value of 1019513. It also has a positive effect on the target variable which means the higher the value of price\_sqm, the higher the value of the predicted resale price.

1. Floor\_area\_sqm

Floor\_area\_sqm feature has the second highest absolute coefficient value of 1006657. It also has a positive effect on the target variable which means the higher the value of floor\_area\_sqm, the higher the value of the predicted resale price. This makes sense as normally a larger property would normally fetch higher prices as well.

1. Flat\_model\_2\_rm

Lastly, Flat\_model\_2\_rm represents flats that have 2 rooms. This feature has the 2nd highest effect on predicted value with an absolute coefficient value of 132756.8. It has a negative effect on predicted value. This indicates that hdb houses that have model types of 2 rooms would have a lower predicted price. Properties with only 2 rooms are normally properties with not much space and thus it makes sense that this feature would pull the predicted price lower.

Unimportant features (based on P values > 0.05):  
Table

Description automatically generated

Summary

An overall adjusted r-square score of 0.97 may be too good to be true. I suspect one of my features is causing a target leakage. Researching on target leakage, I found that target leakage happens when I train an [algorithm](https://www.datarobot.com/wiki/algorithm/) on a dataset that includes information that would not be available at the time of prediction. Since the model already knows the actual outcomes due to the target leakage present in the dataset, the model’s results will be unrealistically [accurate](https://www.datarobot.com/wiki/accuracy/) for the training data, like bringing an answer sheet into an exam.

In my context, since I am predicting resale prices of HDBs, the feature price\_sqm may be a target leakage as it is somewhat of an indicator of the resale price. If this prediction model was to be deployed to predict HDB resale prices, I do not think that users would know information of price per square meter of a specific property beforehand. Therefore, I will try to remove this feature and perform modeling again.

# Modeling – 2nd iteration (Linear Regression)

In this iteration of modeling, I will remove the suspected target leakage – price\_sqm. I will also remove lease\_commence\_date as I feel it is similar to remaining\_lease column. Thus I will remove it as well for better model interpretability. In the regression node, I can simply change the setting to NOT use price\_sqm and lease\_commence\_date as such:





Performance after dropping price\_sqm

**Adjusted R-Square**

Text, letter

Description automatically generated

**RMSE**

Training set: 55911.01

Testing set: 55971.74



**Analysis of Variance**

**Text

Description automatically generated**

Probability > F value of **<.0001** indicates model as a whole is significant.

Takeaway:

After removing price\_sqm, the model performance significantly drops. Adjusted R-Square score decreased from 0.97 to 0.87 while RMSE value increased from 23514.01 -> 55971.74. The performance now is much more realistic as compared to performance in 1st iteration. 0.97 Adjusted R-Square score was indeed too good to be true and the performance reduction after removing the feature is evident that price\_sqm was indeed a target leakage feature.

Compared to the performance in my benchmark model, there was a slight increase of 0.1% in Adjusted **R-Square score of 0.86 -> 0.87** and **decrease of about 3k in RMSE value of 59000.66 ->55971.74**. Therefore, the additional features that are present in the new dataset has helped to slightly increase prediction performance. Let us take a look at the important factors in predicting resale price through analyzing effects plot.

Effects Plot & Score Rankings Overlay(mean predicted of resale\_price)

|  |  |
| --- | --- |
|  |  |

Top 3 influential features

1. Floor\_area\_sqm

After removing target leakage, floor\_area\_sqm feature is now the most important feature in determining resale prices. This stays true as the floor area square meter is one of the common metric people see when buying and comparing properties.

1. Remaining\_lease

The 2nd most important feature in determining resale prices is the amount of remaining lease years available. It has a positive effect, which means the higher the value of this feature, the higher the predicted resale price value.

According to one study, one possible reasoning for this is that all HDBs have a 99-year or shorter leasehold. As HDBs leasehold increases, the value at which it was initially bought for may increase as well. However, owners may see the HDB value start falling as early as from the 45th year onwards.

Therefore, the higher remaining lease a HDB property has, the likelier that the resale price would be higher as well, up to a certain point.

Chart, line chart

Description automatically generated

Study source: <https://lifefinance.com.sg/whats-the-value-of-my-leasehold-hdb-3-dealing-with-lease-decay/>

1. Flat\_model\_2\_rm

HDBs that have 2 room models is still one of the top important factors when determining resale prices in the 2nd model iteration. This feature has a negative effect which means observations with this feature will pull the predicted resale price lower.

Top influential additional features

1. Km\_to\_mrt

Km\_to\_mrt is the 6th most important feature in predicting resale price. This feature calculates the distance of an address of a property to the nearest MRT station. It has a negative effect meaning when one variable increases, the other decreases and vice versa. In other words, feature km\_to\_mrt and target resale\_price negative correlates with each other. The closer the address of a property is to the MRT, the higher the resale price value of the property would be.

1. Region\_Central

This feature is the 21st important feature in predicting resale price. This feature indicates properties that are generally in the central area. It has a positive effect meaning it has a positive correlation with the target resale price.

1. Region\_North\_East

Lastly, this feature is the 22nd important feature in predicting resale price. This feature indicates properties that are in the North East region of Singapore. It also has a positive correlation with the target resale price.

Other insights are:

* Higher storey range, higher resale price (positive effect)
* HDBs in towns such as Marine Parade and Bukit Timah likelier to have higher resale price (positive effect)
* HDB in towns such as Sengkang and Punggol likelier to have lower resale price (negative effect)

# Modeling (Neural Network for Regression) – 1st iteration

Diagram

Description automatically generated

* A supervised machine learning technique that is based on how the human brain works
* Proper term is Artificial Neural Network (**ANN**).

Regression ANNs predict an output variable as a function of the inputs. The input features (independent variables) can be categorical or numeric types, however, for regression ANNs, I require a numeric dependent variable. If the output variable is a categorical variable (or binary) the ANN will function as a classifier.

ANN settings

Target: resale\_price

Exclude variables: price\_sqm, lease\_commence\_date

Input: All other variables

Model selection criteria: Average Error

* As we are using ANN for a Regression task, it is more appropriate to use Average Error as the criteria.



Hidden units: 3

Max iterations: 50

Performance statistics

**RMSE**



Takeaway:

RMSE value for training set is 71798.74 and 71250.07 for testing set. This is a decrease in performance from our Linear regression model previously. That being said, no optimization was done to the ANN model.

Researching on factors that affect the performance of ANNs 2 things steps I can take to improve the model is:

1)Increase hidden units

Increasing the number of hidden units and/or layers may lead to overfitting because it will make it **easier** for the neural network to memorize the training set, that is to learn a function that perfectly separates the training set but that does not generalize to unseen data

2)Reduce dimensionality

The current model has 184 features. Reducing the dimension of my dataset can also reduce the complexity which further mitigates the risk of overfitting.



# Modeling (Neural Network for Regression) – 2nd iteration

In this segment, I will perform the 2 optimization steps previously mentioned above. I will start with **reducing the dimensionality** of my dataset by making use of the Regression node which contained the **Stepwise model selection** setting and connecting to it.

Diagram

Description automatically generated with low confidence

In a way, I use Stepwise regression as a feature selection method as stepwise method selects only important features in the final model. Thus, limiting the number of variables and reducing the dimensionality of the dataset which is then used for my ANN model.

Next, I will increase the hidden units to 24 as I feel it is sufficient to ensure no overfit and acceptable training time.



Performance statistics

**RMSE**

Training set: 40568.12

Testing set: 40978.55



Takeaway:

Overall, using the 2 optimization steps of reducing dimensionality and increasing number of hidden units have increased performance. Comparing RMSE of testing sets, I can see a decrease of approximately 30k from 71250.07 in 1st ANN iteration to 40978.55 in 2nd ANN iteration.

The ANN also outperforms the Linear regression model of 55971.74 -> 40978.55(approx. 15k decrease). Therefore, the champion model here would be the ANN model based on lowest RMSE value.

# Summary/Recommendation

How can a resale price prediction model be used?

One way these prediction models can be used is **to help people who plan to buy a house so they can know the price range in the future**, then they can plan their finance well. For example, if this model was to be deployed, future home buyers could key in specific home details which our model would then use as features and compute the estimated house price for families to refer to and plan their financials.

In addition, house price predictions are also beneficial for property investors to know the trend of housing prices in a certain location.

A realtor could also make use of the prediction model as a supplementary tool to make valuations on a property by simply keying in details of the property instead of physically inspecting and spending time analyzing the market to gauge the price.

What are some insights?

From our linear regression model, important insights identified was

* The size of a property is the biggest influencer on price
* However, there are also other factors such as location, age of property and and distance to MRT
* Least popular model type for a property is a 2 room HDB resale.
* Houses in the central area are more likely to have higher resale price
* Houses nearer to the MRT are also likely to have higher resale price

One thing to note is that there are many other factors that affect the resale price as well such as economic conditions, government regulation and also the proportion of working adults in respective towns that may play apart in the final resale price. Therefore, these factors can also be added into the model for future improvements.

Personal preferences also play apart when buying resale homes. Such as location preferences, whether closer to working area or whether they do not mind traveling. With these intangibles, therefore I feel that this model should only be used as a supplementary tool of analysis for realtors, or users when conducting price valuations as there are many other factors that determine price that is not covered by the models as well.

The champion model was the neural network with optimization from reduced dimensionality and increased hidden units. It had the lowest RMSE of 40978.55. Given the nature of HDB resale prices ranging from hundreds of thousands, a deviation of 40k from actual value and predicted value is good in this case. However, more improvements can be made to the model to optimize it more for better performance or other Machine Learning models could be used. Currently only 1 engineered feature has the biggest influence on resale price which was the distance to MRT. Therefore, an improvement I would make in the future is to learn to engineer better more impactful features.

Another improvement I would make to my project is to also work with a larger dataset. As I previously took only latest 5 years due to time’s sake. A larger dataset would allow my machine learning model to be trained on more data which in turn improves the ML model. More trends and patterns could also be discovered on a bigger dataset.

**References**

<https://www.onemap.gov.sg/docs/#search> (onemap API documentation)

<https://data.world/hxchua/train-stations-in-singapore> (MRT coordinates dataset)

<https://www.freecodecamp.org/news/how-to-combine-multiple-csv-files-with-8-lines-of-code-265183e0854/> (combining multiple csv files in python)

<https://en.wikipedia.org/wiki/List_of_shopping_malls_in_Singapore> (list of shopping malls)

<https://schoolbell.sg/primary-school-ranking/> (top 10 primary schools)

<https://schoolbell.sg/secondary-school-ranking/#Express_O-Level_Secondary_School_Ranking> (top 10 secondary schools)

<https://data.gov.sg/dataset/school-directory-and-information> (primary and secondary school general information)

<https://towardsdatascience.com/what-are-the-best-metrics-to-evaluate-your-regression-model-418ca481755b> (evaluation metrics for regression problems)

<https://www.datarobot.com/wiki/target-leakage/> (target leakage)

<https://sixsigmastudyguide.com/main-effects-plot/#:~:text=The%20main%20effects%20are%20the,response%20value%20increases%20or%20decreases.&text=Negative%20effect%3A%20Increase%20in%20the%20independent%20variable%20decreases%20the%20dependent%20variable>. (interpreting effect plots)

<https://documentation.sas.com/doc/en/emref/15.1/p1gqtpy080di3yn1d4i0xwlbe91h.htm> (SAS documentation on nodes)