EE4802 - Learning From Data - House Price Prediction Project Report

Aditi Chadha – A0226612M
National University of Singapore

Contents

Problem statement: Propose a model that can help the **general public** determine a fair price for a resale flat that he/she plans to buy/sale **based on data** available in the url below. Note that transaction prices from as early as 1990 are available. https://data.gov.sg/dataset/resale-flat-prices

Abstract: There are multiple factors that affect the selling price of HDB flats in Singapore. By using regression, I am interested in finding out how the selling price of a HDB resale flat changes based on characteristics such as Flat type, Storey Range, Floor area in sqm, Remaining years of lease, Flat Model, etc.
In this report, I will present how I implement the knowledge learnt in EE4802 course, to do data pre-processing, data analysis, model training and complete the prediction of the selling price of HDB flats in Singapore.

Data and data preparation: Please refer to the python notebook "EE4802_Project_1.ipynb" alongside.
To build this model, I am using all the data-sets provided on this link - https://data.gov.sg/dataset/resale-flat-prices, that is, the data from 1990 all the way to 2023. This data is provided in five .csv files which I load into the "data" dataframe. There are multiple reasons why I chose to include all the data files. Some of them are:

1. **Improve accuracy**- More data can lead to a more accurate model because it provides a more comprehensive representation of the problem domain, patterns and relationships within data.
2. **Reduce overfitting**- Using more data helps to reduce overfitting as it provides a larger and more diverse set of examples for the model to learn from. This will prevent the model from adjusting to the training data too well such that it does not result in poor performance on the new data.
3. **Improved robustness**- The robustness of a model is improved by including more data as then it is more likely to have learned to handle variations and noise in the data.

```
import pandas as pd
import glob
data = pd.concat(map(pd.read_csv, glob.glob("/content/drive/MyDrive/Data Files/*.csv")))
# The above .csv files include data from 1990 to 2023.
# This data is contained in a total of 5 files namely:
# 1990-1999.csv
# 2000-feb-2012.csv
# Mar-2012-to-Dec-2014.csv
# Jan-2015-to-Dec-2016
# Jan-2017-onwards.csv
```

```
data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 894203 entries, 0 to 369650
Data columns (total 11 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   month                894203 non-null  object
 1   town                 894203 non-null  object
 2   flat_type            894203 non-null  object
 3   block                894203 non-null  object
 4   street_name          894203 non-null  object
 5   storey_range         894203 non-null  object
 6   floor_area_sqm       894203 non-null  float64
 7   flat_model           894203 non-null  object
 8   lease_commence_date  894203 non-null  int64
 9   remaining_lease      185153 non-null  object
 10  resale_price         894203 non-null  float64
dtypes: float64(2), int64(1), object(8)
memory usage: 81.9+ MB
```

Exploratory Data Analysis – Once, all the data of the above 5 files is loaded into the "data" data frame, I do some exploratory data analysis to know more about the data fields contained in the dataFrame. I use the .info() method from Python's pandas library to obtain a summary of the "data" dataFrame's structure and content.

Feature Engineering: After observing the data, some features need to be encoded or transformed and some new features need to be added after learning from the already given information. In this section, we focus on transformation and encoding of existing features.
For a feature to be useful, it must have a mathematical relationship to the target (here - 'resale_price) for models to learn. The following pre-processing is done to the data-sets before training the model –

1. The 'lease_remain_years' column is added to the DataFrame which contains the remaining years of lease in a numerical form and this value is calculated for each row using this formula- 99 – (2023-data['lease_commencement_date']). Refer to the code in the python notebook.

2. Since the ‘storey_range’ data field in the dataFrame is of the data type ‘object’, I convert it into a numerical value by finding the median of the ‘storey_range’ values for each house. The calculated value is stored in the new column - ‘storey_median’ in the ‘data’ dataframe.

```
[11] data['storey_median'] = data['storey_range'].apply(lambda x: get_median(x))
     data
```

3. The ‘data’ dataframe has the following data fields after updation.

```
data.head(5)
```

	month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	remaining_lease	resale_price	lease_remain_years	storey_median
0	2015-01	ANG MO KIO	3 ROOM	174	ANG MO KIO AVE 4	07 TO 09	60.0	Improved	1986	70	255000.0	62	8.0
1	2015-01	ANG MO KIO	3 ROOM	541	ANG MO KIO AVE 10	01 TO 03	68.0	New Generation	1981	65	275000.0	57	2.0
2	2015-01	ANG MO KIO	3 ROOM	163	ANG MO KIO AVE 4	01 TO 03	69.0	New Generation	1980	64	285000.0	56	2.0
3	2015-01	ANG MO KIO	3 ROOM	446	ANG MO KIO AVE 10	01 TO 03	68.0	New Generation	1979	63	290000.0	55	2.0
4	2015-01	ANG MO KIO	3 ROOM	557	ANG MO KIO AVE 10	07 TO 09	68.0	New Generation	1980	64	290000.0	56	8.0

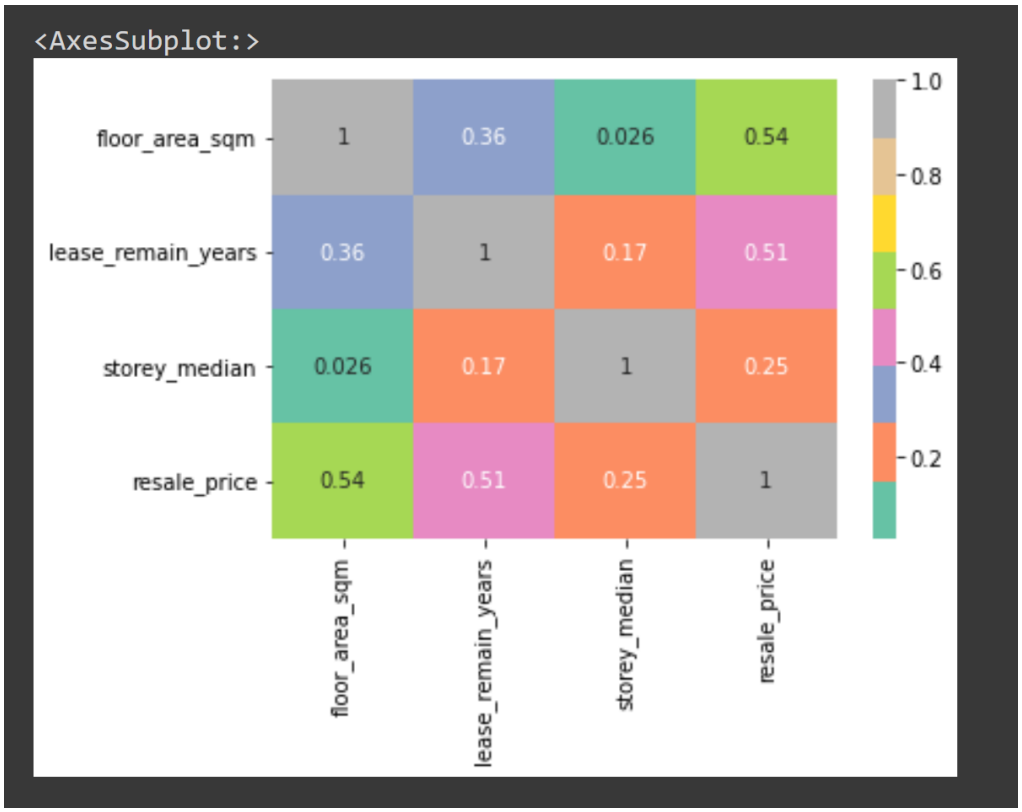
4. The two columns ‘block’ and ‘street_name’ are dropped from the dataFrame as the data contained in these columns is not of much significance to us. This is because we are not looking at location specific amenities such as closeness to MRT etc.
   Since, the ‘lease_remain_years’ column with numerical data is already added to the dataFrame, the ‘remaining_lease’ column with ‘object’ type data is dropped and the newly created dataFrame is stored as ‘trim_data’.

```
trim_data = data.drop(["block", "street_name", "remaining_lease"], axis=1)
trim_data
```

5. To better analyse the data, I create another DataFrame – ‘trim_data_correlation’ from ‘trim_data’ using the relevant columns (columns that contain numerical data) as shown below:-

```
trim_data_correlation = trim_data[['floor_area_sqm','lease_remain_years','storey_median','resale_price']]
trim_data_correlation
```

   This DataFrame is used to find the correlation between the various variables in the DataFrame and most importantly the correlation between the features and target variable. The correlation values will help decide of the data features we want to include while training our model. In essence, correlation being a statistical measure helps us analyse how strongly two variables are related to each other. Correlation values range from -1 to 1, where 0 indicates no correlation while +1 and -1 indicate perfect positive and perfect negative correlation. Correlation measures the degree to which a change in one variable is associated with a change in the other variable. The code uses the ‘seaborn’ library to create a heatmap of the correlation matric of ‘trim_data_correlation’ DataFrame. You may refer the code on the python notebook. The heatmap obtained is as follows-



The heatmap was useful in identifying patterns in data and determining which variables are correlated.

6. After observation, it was noted that the data in the 'flat_model' column is inconsistent as the same values differ by the upper case and smaller case. Example – '2-room' and '2-ROOM'. The data in this column was made consistent by changing all values to lower case.
7. To perform regression, we need all of the data to be numerical. The non-numerical data in our current data-set is contained in the following data-fields: ['month', 'flat_type', 'storey_range', 'town', 'flat_model'].

   With the help of encoders, the non-numerical data is transformed into numerical data as follows:

```
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
pipeline = ColumnTransformer([
("o", OrdinalEncoder(), ["month","flat_type","storey_range"]),
("n", OneHotEncoder(), ["town", "flat_model"]),
], remainder='passthrough')
X = pipeline.fit_transform(trim_data.drop(["resale_price"], axis=1))
y = trim_data["resale_price"]
```

The data under 'month' were transformed using the ordinal encoder. For example, 1990-01 is converted into 0 and 1990-02 is converted into 1. Similarly, the data under 'flat_type' and 'storey_range' was transformed using ordinal encoder. The python code to view the values that were under these respective data fields and the sequence in which they are encoded can be found in the python notebook.
The data under 'town' and 'flat_model' were transformed using the one-hot encoder (binary type conversion). For these fields, an additional column is included for every unique data value. For example, since there are 27 different HDB towns, 27 columns were added. The python code to view the values that were under these respective data fields and the sequence in which they are encoded can be found in the python notebook.

Models and Model Selection -
Since, the data is now ready (in numerical form) we perform Linear and Decision tree regression and the following result is obtained –>
Linear Regression -

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X, y)
print("Intercept: ", lr.intercept_)
print("Coefficient: ", lr.coef_)
print("R-sq of model: ", lr.score(X, y))
```

```
Intercept:  -6334162.456204808
Coefficient:  [    845.43504934   41040.73349139    5582.07109283   17950.30133105
    7521.54862231   70289.91509922  -26712.24350379   76751.04059092
  -71297.67283541  102156.74814075   63058.51847715  -71688.20742874
   32309.93437463   29638.96891507  -25025.94230027  -26012.015541
  -57398.96460136   43572.95594543   -7545.29987161  103819.12402913
  -26160.51643569  -60601.98040557   76276.71142158 -108113.9927657
  -73478.7367901     6415.01398081   -3142.90447368   47994.08366223
  -84954.3564969   -35622.03114007  -29442.60517649    2677.2168179
  -66344.57377618  -58283.62995009   91753.30427272  -76888.45563992
   10083.90267484  -44813.0439939   -69904.95328862  -27879.42920211
  -93374.25946249 -105296.90782535  -68941.05248274  -68056.05672553
  102429.92772683   -3218.36056242  -82826.36810485  -75821.45152706
  147231.99285721  253753.69787582  263161.10549284    1899.78281385
    3139.81895261]
R-sq of model:  0.8287881265494992
```

Decision tree Regression –

```
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor(criterion='friedman_mse', max_depth=12, min_samples_leaf=1, random_state=0).fit(X, y)
print("R-sq of model: ", dtr.score(X, y))
```

```
R-sq of model:  0.9211873379843262
```

Model validation -Based on the above results, Decision tree Regression models seems a better choice for the model as it has a higher R^2 value. But since, a higher R^2 value does not guarantee better predictions, we will use data to train and test the model and evaluate the model's prediction accuracy.

The code below uses 'train_test_split' function from the Scikit-learn library to split a data-set into training and testing sets for machine-learning. The function returns four arrays – X_train, y_train which contain the training data, and X_test and y_test which contain the testing data. The test_size parameter is set to 0.25 which means 25% of the data will be used for testing while the remaining 75% will be used for training.

```
[ ] from sklearn.model_selection import train_test_split
    import numpy as np

    X_input=np.asarray(X_with_resale_price.toarray()[:,:-1])
    Y_input=np.asarray(X_with_resale_price.toarray()[:,-1])

    X_train, X_test, y_train, y_test=train_test_split(X_input,Y_input,random_state=42,test_size=0.25) #considering 25% of data for prediction purpose
```

Model 1- We perform Linear Regression again but this time with the training and testing data, the following values are observed: (For the code, you may refer the python notebook).

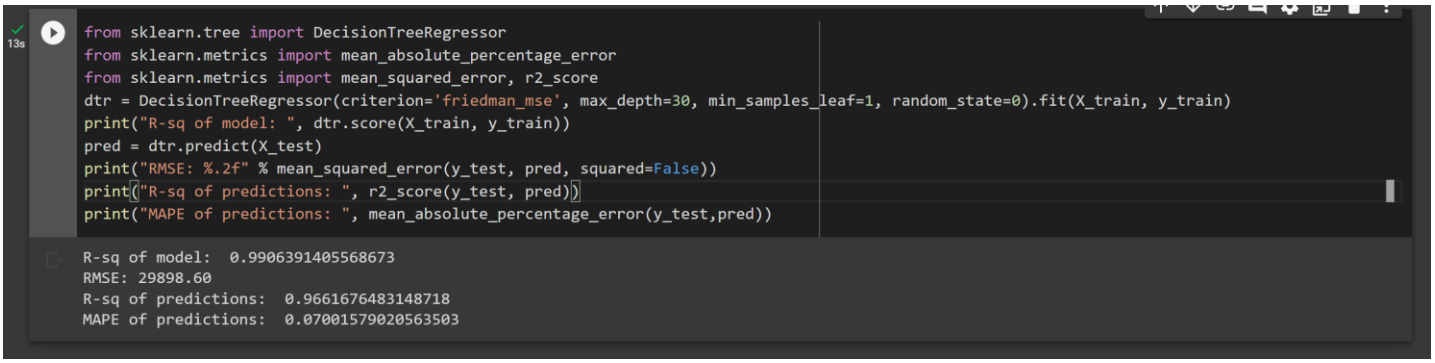
```
R-sq of model:  0.8288982181215597
RMSE: 67285.94
R-sq of predictions:  0.8286519529833725
MAPE of predictions:  0.21506023009369243
```

Model 2-Next we perform, decision tree regression with the training and testing data and the following values are noted for a decision tree with max_depth=12:

```
R-sq of model:  0.9215745706527093
RMSE: 46047.28
R-sq of predictions:  0.9197511923683699
MAPE of predictions:  0.1041027645438105
```

Model 3- To find out the best model, different values of max_depth were tried to create different decision tree regression models using the training and testing data and the best values were obtained using the max_depth = 30. In this case, the (coefficient of determination) R^2 of the model and R^2 of predictions was maximized, while the Mean Absolute Percentage error was minimized (Refer to the screenshot below).

Hence, this model was chosen to be the final proposed model for this project.



Final Model – Decision Tree Regression Model with criterion = 'friedman_mse' and max_depth = 30 is used to obtain a model to predict the house prices in Singapore with a mean absolute percentage error of 0.070015.

Discussions– While coming up with the models, I tried to do geocoding to make use of the 'block' and 'street_name' and combined them as 'address' data field. This address data field was used to get the latitude, longitude and postal code of the flat using the following open-source API - https://www.onemap.gov.sg/docs/#onemap-rest-apis.

However, since the data-set was extremely huge (1990-2023) so all the API calls could not be made due to the time constraints. The code ran for around 7 hours but still the data for all the data points was not fetched.

The aim to find the latitude, longitude and postal address was to find the proximity of a HDB to the nearest MRT station and include that data feature in the training model to make the model more precise, as the costs of HDb also depend on their proximity to the MRT station. However, due to timing issues, it could not come forth.

Refer to this link to check my work on geocoding - https://colab.research.google.com/drive/192bcVnAi_BZtwZzq_bnBZPKSjhNIdgvE?usp=sharing
(This is not the complete code with the final model as the location for all the data points was not fetched, but it can demonstrate how I intend to fetch the exact location of the flat and use it as a data-field for the model.)

How to use proposed model to determine fair price:

To use the proposed model to determine the fair price, please download the submitted "EE4802_Project_1.ipynb" notebook and upload it to your google colab workspace. Then you may click on "Runtime" => "Run all" to run all the cells.
Next, traverse to the last cell and change the values of the input array corresponding to values of the respective house you are looking for, to get the predicted price. You may have to refer to the encoding sequence to find the index of the respective values.

```
#FINAL MODEL

from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_error, r2_score
dtr = DecisionTreeRegressor(criterion='friedman_mse', max_depth=30, min_samples_leaf=1, random_state=0).fit(X_train, y_train)
print("R-sq of model: ", dtr.score(X_train, y_train))
pred = dtr.predict(X_test)
print("RMSE: %.2f" % mean_squared_error(y_test, pred, squared=False))
print("R-sq of predictions: ", r2_score(y_test, pred))
print("MAPE of predictions: ", mean_absolute_percentage_error(y_test,pred))

#Change values below, Refer to instructions on the top

import numpy as np
input = np.zeros(53)
input[0] = 331   # month
input[1] = 3     # flat_type
input[2] = 4     # storey_range
input[24] = 1    # town
input[38] = 1    # flat_model
input[51] = 100  # floor_area
input[52] = 1999 # lease_commencement_data
print(input)
print("Estimated resale price (DT regression)=> ", dtr.predict([input]))
```

You may look for the details on the python notebook.
The indices of the input array represent the following with respect to the house -

Index	Representation w.r.t house	Data Type
0	month	Numerical
1	Flat_type	Numerical
2	Storey_range	Numerical
3-29	Town	Binary
30-50	Flat_model	Binary
51	Floor_area	Numerical
52	Lease_commencement_data	Numerical

Conclusion – This proposed model predicts the house prices of HDB flats in Singapore using Decision Tree Regression with max_depth = 30 and also provides the user details about the model such as –

1. The coefficient of determination (R^2 of the model)
2. R^2 of predictions
3. Root mean squared error
4. Mean Absolute Percentage error.