## **Predicting Home Resale Prices - Summary**

Research Objective: Using our extensive data science knowledge, we were meant to create a model that will help us accurately predict resale prices on homes, so that we can buy these homes, resell them using the predicted prices, and in the long run, make better decisions for our business.

Model Function: The function of our model is for it to confidently predict the 'resale prices' of houses; these resale prices that are predicted by our model, will then be used to help us in determining which homes to buy and then resell again.

Assumptions:

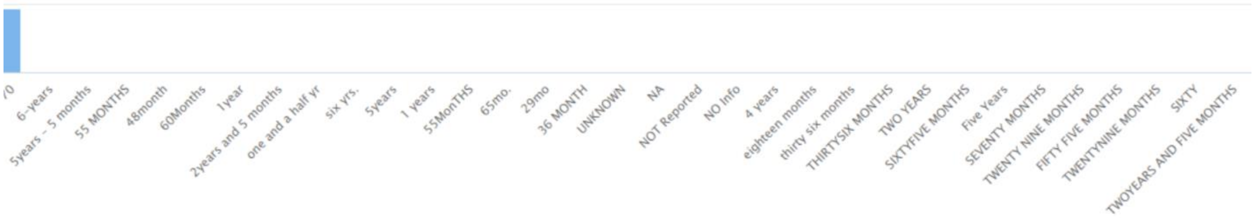
- Correlation between number of rooms and area of home size
- The resale price is represented in thousands USD

Data Summary:

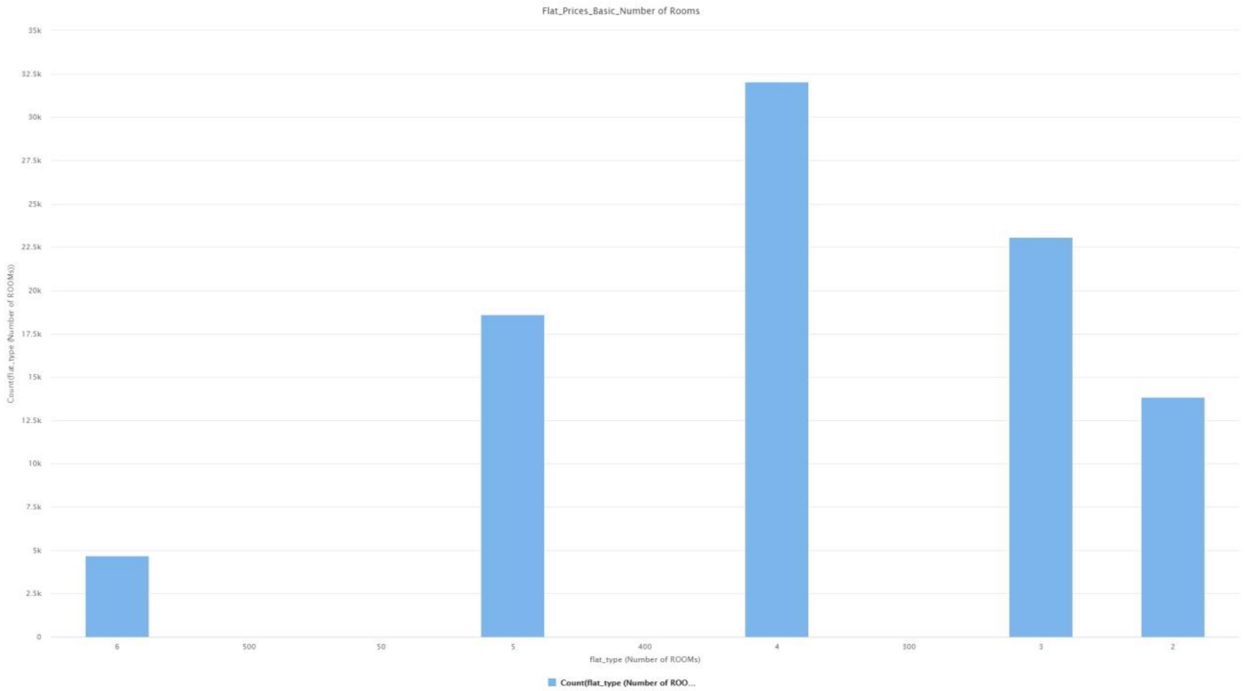
All data used for this project was extracted from four separate excel files.
Each dataset contained 92,270 rows.
*Flat Prices Basic* – contained columns with data like block number, street name, and town name, with the more significant columns being that of the number of rooms within the home, the area in square feet of the home, and the current resale price of the homes.
*Flat Model* – contained data regarding the type of home.
*Location and Storey Range* – contained data regarding the latitude of each house, and the range each house falls into pertaining to the amount of stories each home has.
*Lease Time* – contained the time remaining on lease for flats.

Data Prep:

The data we were given required extensive cleaning for it to be interpretable for a model. The flat prices dataset, for example, contained 19 columns, 13 of which contained missing values and 11 of which were empty, not representing a variable. We cleaned our data using RapidMiner turbo prep, where we used the transform feature to remove empty columns, as well as features that we decided were redundant. Before we accounted for observations with missing values, we also had to account for any data that was incorrectly entered into the datasets by converting that data into numeric values where it was often entered incorrectly as a string. Doing this before imputing for missing values allowed us to verify the mean for numeric data, and the mode for categorical data before we used those techniques for imputation.

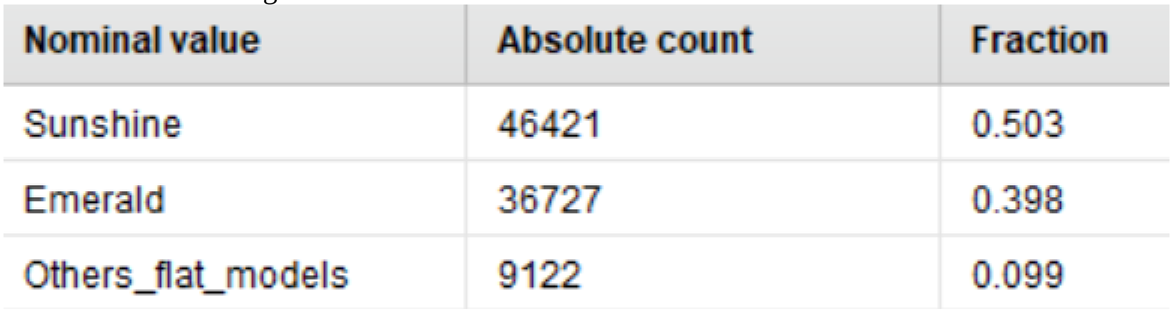


For example, in the remaining lease dataset, we used the visualization tool from the results view in RapidMiner to identify all the instances of incorrectly entered data. We then corrected each instance by utilizing the RapidMiner turbo prep replace feature in the transform section, and replaced each string of specific text with its corresponding lease as an integer. This was also the case in most of the datasets we were given, where there was text entered in records identifying a value as "unknown" or N/A, which we interpreted as a missing value. After converting all incorrectly entered numeric data into integers, we looked through the data to identify outliers.



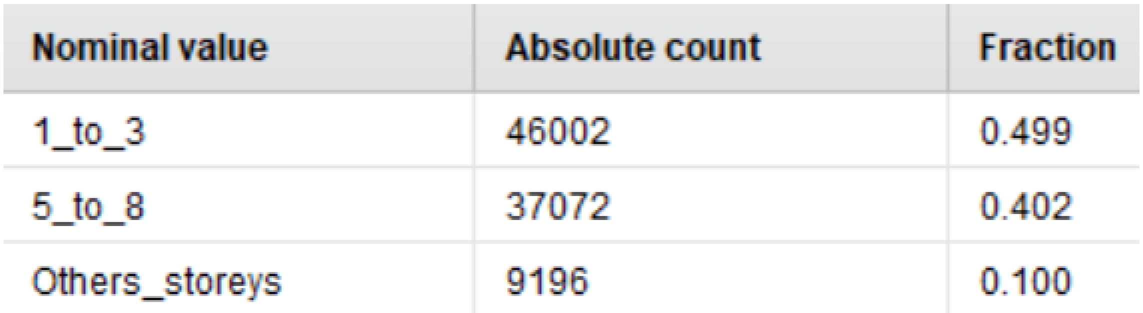
We identified 5 specific outliers in the flat_type(number of rooms) feature, as records that identified the number of rooms as being greater than 6. These outliers had values of 50, 300, 400 (2), and 500. The approach we took was to treat these as missing values and imputed them as the mean to the nearest whole value which was 4. Another approach could have been to identify these values as incorrectly inputted and contain the number of rooms that is represented by the first number of the value. For example, 500 would then be imputed to the 5 number of rooms category.

The datasets for flat_model and location_and_storey_range both contained categorical variables that needed to be cleaned and categorized.



| Nominal value | Absolute count | Fraction |
|---|---|---|
| Sunshine | 46421 | 0.503 |
| Emerald | 36727 | 0.398 |
| Others_flat_models | 9122 | 0.099 |

Both data sets contained two predominant values; in the case of the flat_model dataset, those were “Sunshine” and “Emerald,” also accompanied by an “others” category with additional unique flat models such as “Jade”. Our approach was to create three categories “Sunshine”, “Emerald”, and “others_flat_models,” where we would be able to add the rest of the unique values to the others category. However, we decided to keep the others category as it represented nearly 10% of the observations which is a significant portion of the data.



| Nominal value | Absolute count | Fraction |
|---|---|---|
| 1_to_3 | 46002 | 0.499 |
| 5_to_8 | 37072 | 0.402 |
| Others_storeys | 9196 | 0.100 |

We took the exact same approach when it came to categorizing the story range data; however, because in this case each category represented a range, we had to address data expressed as a specific value such as “1” or “one,” which would represent a flat with one story. We treated data represented like this as errors and replaced them with mode. Another approach could have been to label that data to its corresponding story range.

After having cleaned all the data, we underwent feature selection to determine what features we would like to have available for the model. In the flat_price_basic dataset we removed the following features: town (because each record contained the same value) street name and block (because there was no significant variation in the values) descr (because it was missing in the majority of the observations and the descriptions were long strings of text unfit for modeling) and month (because of the difficulty in trying to find value using that time series data). In Location_and_Storey_Range we removed the latitude feature because it was missing 10,663 records which represented a significant enough portion of the data with 11.5%.

We then underwent feature engineering, where we normalized the square feet, assigning it a normalized range of values between 0 and 1, due to the size of the values compared to rest of the predictor variables. For example, square feet ranged between 400 to 1800, whereas the number of rooms ranged from 1 to 6. Additionally, we encoded the categorial variables flat_model and storey_range to binary numeric variables, so they could be used as predictors in the model. Additionally, we used One Hot encoding, and later selecting the appropriate dummy variables, so that “others” would always be the omitted or base variable

After having undergone this data preparation we aggregated all the data we kept from the data sets and aggregated them into one datagram to undertake exploratory data analysis and allow for easier modeling later on.

Exploratory Data Analysis EDA:

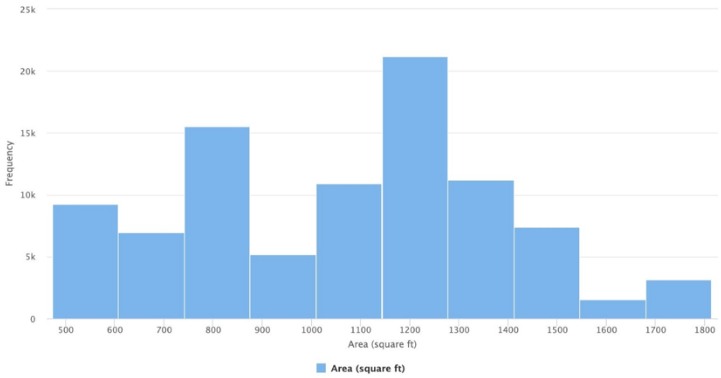
Summary Statistics Before Normalization:

| Variable | Min | Max | Mean | Stdev |
|---|---|---|---|---|
| **flat_type (Number of ROOMs)** | 2 | 6 | 3.752346 | 1.092607 |
| **Area (square ft)** | 473.6257 | 1814.548 | 1067.278 | 318.9416 |
| **resale_price (in Thousands USD)** | 2.577714 | 453.7725 | 216.5391 | 75.15181 |
| **flat_model = Emerald** | 0 | 1 | 0.398038 | 0.489496 |
| **flat_model = Sunshine** | 0 | 1 | 0.5031 | 0.499993 |
| **flat_model = Others_flat_models** | 0 | 1 | 0.098862 | 0.298478 |

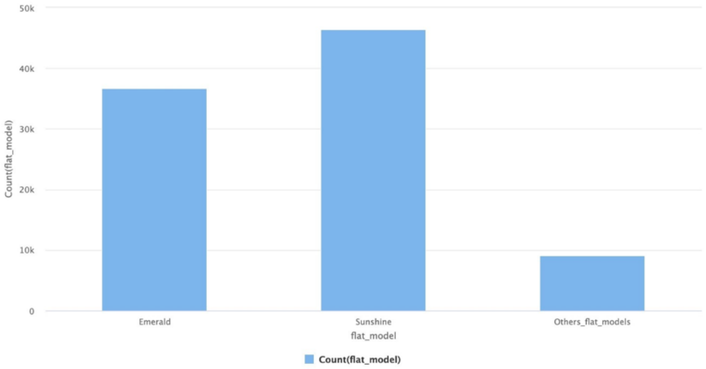
| | | | | |
|---|---|---|---|---|
| **remaining_lease** | 12 | 72 | 31.83118 | 19.67171 |
| **storey_range = Others_storeys** | 0 | 1 | 0.099664 | 0.299553 |
| **storey_range = 1_to_3** | 0 | 1 | 0.498559 | 0.500001 |
| **storey_range = 5_to_8** | 0 | 1 | 0.401777 | 0.49026 |

<u>Distribution (Histograms):</u>

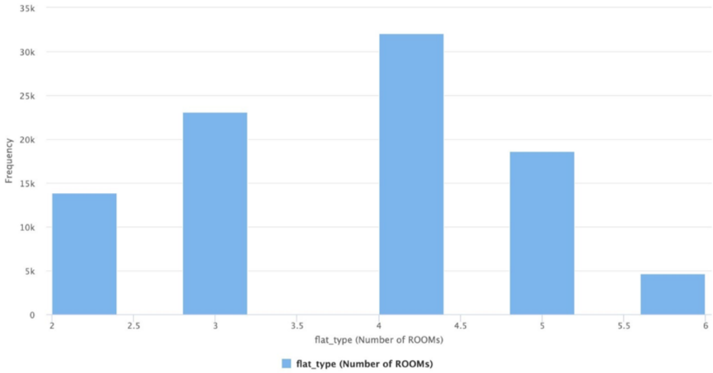
Area(Square ft)



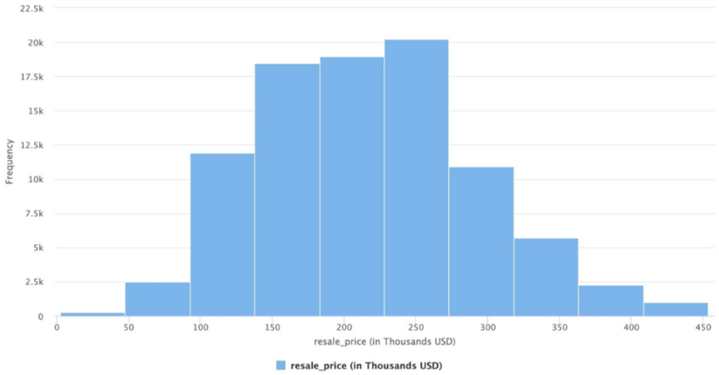
Flat_model



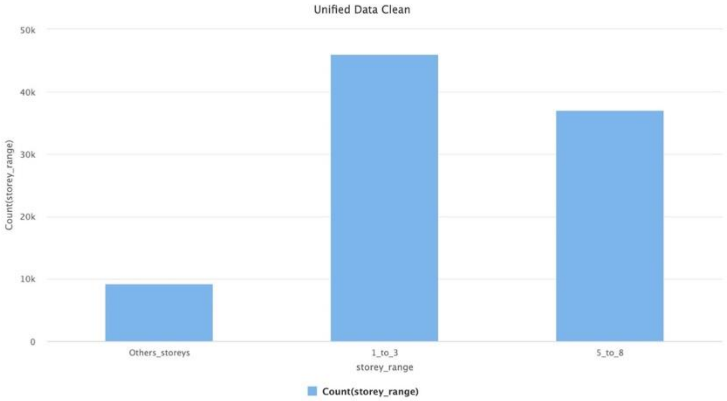
Flat_type (Number of Rooms)



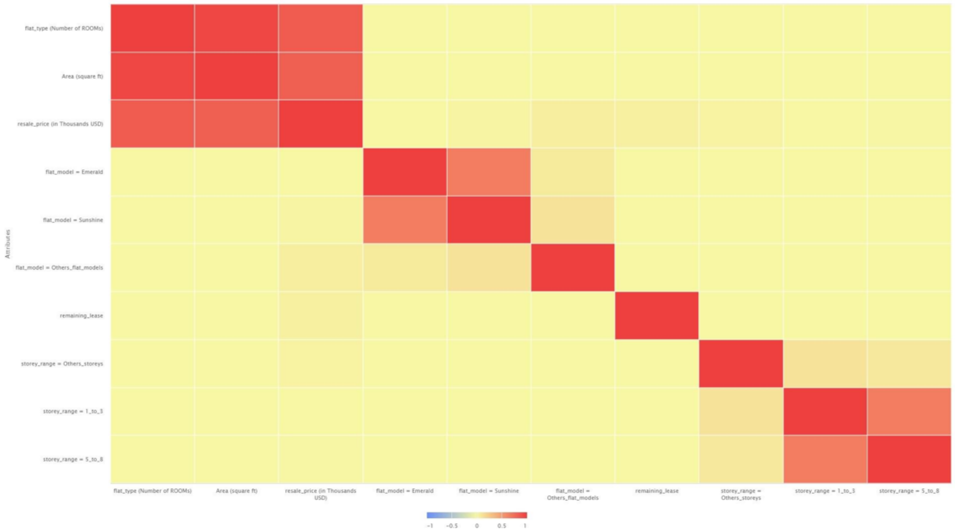
Resale_price (in Thousands USD)



Storey_range



Correlation Matrix:



We developed a correlation matrix to visualize the correlation between attributes. This visual depicts a very strong positive correlation between square feet and the number of rooms. Additionaly, it can be seen that square feet and number of rooms are both highly positively correlated with resale price, suggesting multicollinearity, which could overemphasize their importance in predicting resale price.

The columns and data types of the final table are shown below:

| **Variable** | **Data Type** |
|---|---|
| flat_type (Number of ROOMs) | Int |
| Area (square ft) | Float |
| resale_price (in Thousands USD) | Float |
| flat_model = Emerald | Object |
| flat_model = Sunshine | Object |
| flat_model = Others_flat_models | Object |
| remaining_lease | Int |
| storey_range = Others_storeys | Object |
| storey_range = 1_to_3 | Object |
| storey_range = 5_to_8 | Object |

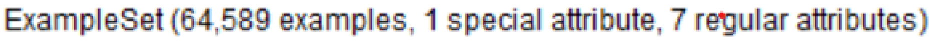
Training and testing datasets:

The dataset ended up having 8 columns total using the following attributes:

**[flat_type (Number of ROOMs), Area (sqaure ft), flat_model = Emerald**, **flat_model = Sunshine**, **remaining_lease**, **storey_range = 1_to_3**, **storey range = 5_to_8** , **resale_price (in Thousands USD)]**

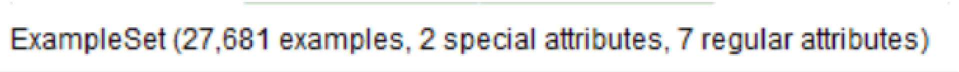
We split the dataset using random shuffling, where we split the data into training (70%) and test (30%), with our ***target variable*** as resale price.

Training Set:



ExampleSet (64,589 examples, 1 special attribute, 7 regular attributes)

Test Set:



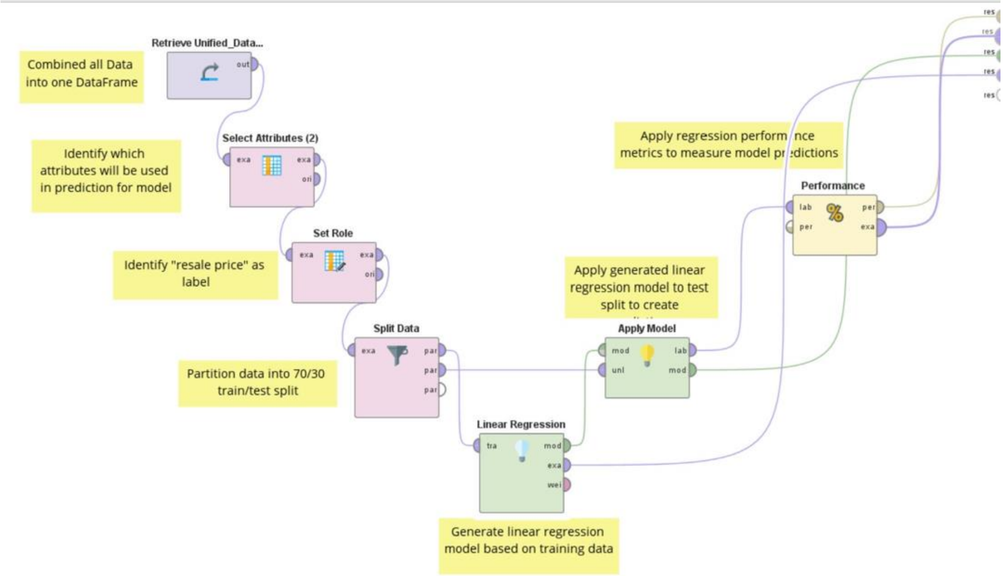
ExampleSet (27,681 examples, 2 special attributes, 7 regular attributes)

Our training set contained 64,589 records and the test contained 27,861 records. The additional special attribute from the test set is derived from the additional prediction column created when running the test for the model.

Algorithm tested:

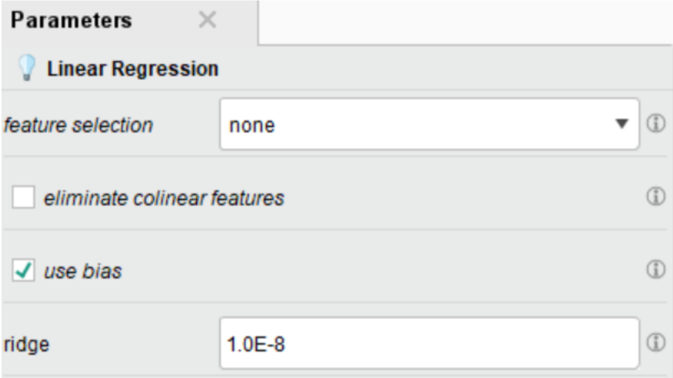
Linear Regression

Process:


Retrieve Unified_Data...
out
Combined all Data into one DataFrame
Select Attributes (2)
exa
exa
ori
Identify which attributes will be used in prediction for model
Set Role
exa
exa
ori
Identify "resale price" as label
Split Data
exa
par
par
par
Partition data into 70/30 train/test split
Linear Regression
tra
mod
exa
wei
Generate linear regression model based on training data
Apply Model
mod
unl
lab
mod
Apply generated linear regression model to test split to create
Performance
lab
per
per
exa
Apply regression perform nce metrics to measure model pr dictions
res
res
res
res
res

Initially, we selected the desired attributes for the model we were creating. Then, we identified the label or the variable we were trying to predict, which, in this case, was retail price. In order to deal with our category variables, we created a set of dummy variables from our encoded data. To avoid perfect multicollinearity between our parameters we omitted the “others” variable from each category and treated this as an implied base variable. We then split the data into a 70/30 train/test split in order to train the model on a shuffled sample of 70 percent of the data, then testing the model on the remaining 30 percent of data, which was unseen by the model. The model we created was a mullti-linear regression model. The hyperparameters we had set for the linear regression operator were: feature selection set to none, deselected eliminate colinear features, use bias selected, and used default ridge (1.0e^-8). Once the linear regression model was trained, we applied the model to the test set to create predictions. We then used these predictions to assess the performance of the model to determine the model’s ability to correctly make predictions in the future. The model performance metrics we used were the regression analysis metrics root mean squared error (RMSE) and squared correlation ($R^2$), where models having a lower RMSE and higher $R^2$ represent a better-performing model.

Model parameters:



Parameters

Linear Regression

feature selection: none

□ eliminate colinear features

☑ use bias

ridge: 1.0E-8

- Feature selection was set to none
- Deselected eliminate colinear features
- Use bias selected
- Used default ridge (1.0e^-8).

Measuring Performance:

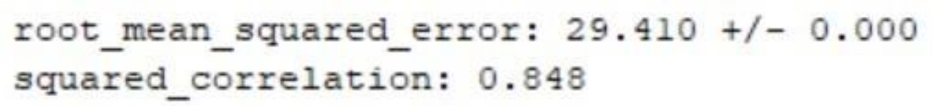
We measured the model's performance using the regression analysis metrics root, mean squared error (RMSE) and squared correlation ($R^2$), where models having a lower RMSE and higher $R^2$ represent a better-performing model.

Model 1:

Datasets Used: Flat_Prices_Basic

Predictors Used: Area sq ft (NORMALIZED), Number of rooms

Model Performance:



```
root_mean_squared_error: 29.410 +/- 0.000
squared_correlation: 0.848
```

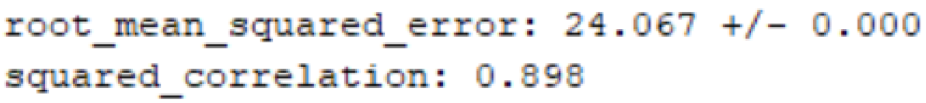
Model 2:

Datasets Used: Flat_Prices_Basic, Flat_Model

Model Type: Linear Regression

Predictors: Area sq ft (NORMALIZED), Number of rooms, Flat_Model = Sunshine, Flat_Model = Emerald

Model Performance:



```
root_mean_squared_error: 24.067 +/- 0.000
squared_correlation: 0.898
```

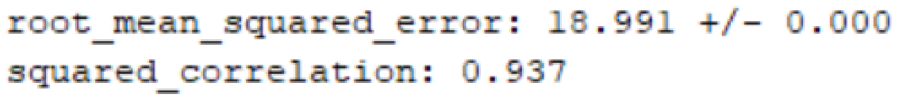
## Model 3:

Datasets Used: Flat_Prices_Basic, Flat_Model, Lease_Time

Model Type: Linear Regression

Predictors: Area sq ft (NORMALIZED), Number of rooms, Flat_Model = Sunshine, Flat_Model = Emerald, Lease_Time

Model Performance:



```
root_mean_squared_error: 18.991 +/- 0.000
squared_correlation: 0.937
```

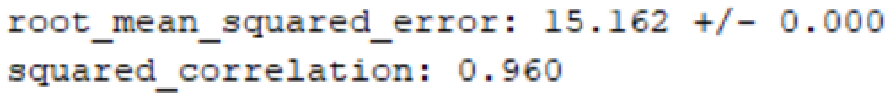
## Model 4:

Datasets Used: Flat_Prices_Basic, Flat_Model, Lease_Time, Location_and_storey_ranges

Model Type: Linear Regression

Predictors: Area sq ft (NORMALIZED), Number of rooms, Flat_Model = Sunshine, Flat_Model = Emerald, Lease_Time, , storey_range = 1to3, storey_range = 5 to 8

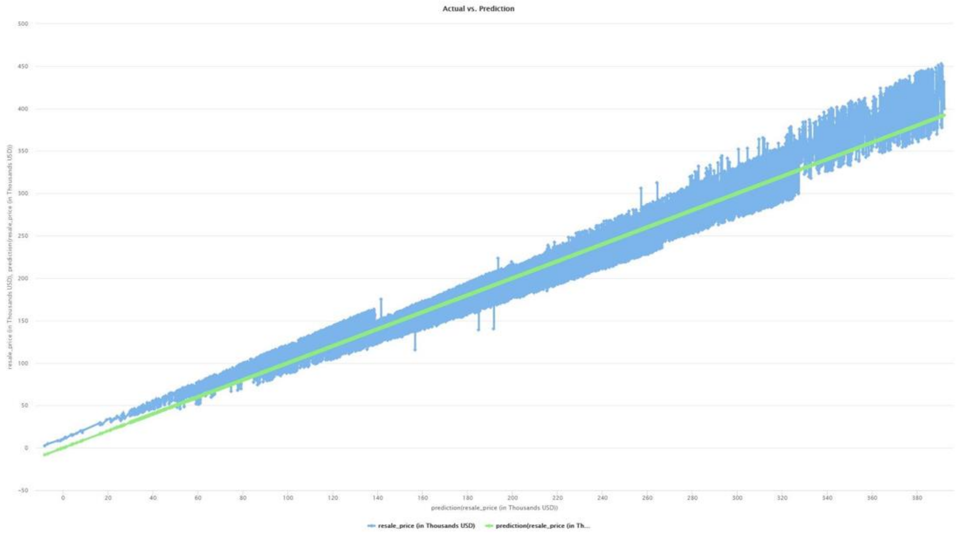
Model Performance:



```
root_mean_squared_error: 15.162 +/- 0.000
squared_correlation: 0.960
```

## Model Selection and Results:

The model we chose to select for deployment was Model 4, which utilized features obtained from all the data sets provided. Through the addition of these new features, the model's performance improved significantly seen through the decline of the RMSE and increase in $R^2$. However, the inability to account for the adjusted $R^2$ for the model is important to note, because this can help balance out the improvements generated from purely from adding new features to the model. Since from Model 1 to Model 4 we only added 5 new features, the impact should not be as significant compared to if we would have added a larger quantity of features. The computational time and power required to run Model 4, compared to the other models, was negligible, as the model we selected is a linear regression. This means that in this case, we can and are incentivized to deploy the most accurate model. Additionally, the ETL processing steps required for implementing new data are relatively straightforward and would not require intensive resource use to supply new data to continue to improve and run the model daily.



We can see the predictions of the model as represented through the straight green line and the actual results of the model shown in blue. Showing a clear linear trend that the model is able to pick up on, suggests ample opportunity to derive value from underpriced flats, as all data points below the green line represent an arbitrage opportunity If these specific flats are purchased at that resale price, we should, with confidence, have the capability to resell them at a higher price producing a profit, which would benefit our business.

## Conclusion:

For the purpose of making profit and creating a successful real estate business, we've used our extensive data science knowledge to test four separate models to determine which would be best at accurately predicting the resale prices of homes. We tested a standard linear regression model and added new features with each additional model that we tested to see which model would help us best predict resale prices, in order for us to choose which homes to invest in to resell. With each model, as stated previously, we increased the variables until we got to seven predictor variables. Our one predicted variable, resale price, was constant throughout each of the models. After testing each of our models, we determined that Model 4, which incorporated all seven predictor variables, was the most accurate in predicting the resale prices of the house data imputed, and would therefore, be the best choice in helping us in our business.