Predictive Modelling of HDB Resale Prices:

Leveraging Machine Learning for Market Insights and Decision Support

An AI ML Capstone Project

By: Roslan M. Amin (Sep 2024)

Table of Contents

1. Project Overview

This project focuses on the application of machine learning to predict HDB resale prices and providing valuable insights for various stakeholders. It highlights the dual goals of developing a predictive model and offering decision support, making it clear and comprehensive.

2. Introduction

The Housing and Development Board (HDB) flats are a cornerstone of Singapore's public housing policy, providing affordable housing to over 80% of the population. With the dynamic nature of the real estate market, predicting HDB resale prices has become increasingly important for buyers, sellers, and policymakers. This project aims to develop a predictive model for HDB resale prices using historical data and advanced machine learning techniques. By understanding the factors influencing resale prices, stakeholders can make informed decisions, ensuring a fair and transparent market.

3. Objectives

- **Develop a Predictive Model**

  Create a robust model to accurately predict HDB resale prices based on historical data.

- **Identify Key Factors**

  Determine the most significant factors affecting resale prices, such as location, flat type, floor area, and remaining lease.

- **Enhance Decision-Making**

  Provide insights to buyers, sellers, and policymakers to facilitate informed decision-making.

- **Market Analysis**

  Analyse trends and patterns in the HDB resale market to understand its dynamics and future directions.

4. Analysis Goals

- **Data Cleaning and Preprocessing**

  Ensure the dataset is clean and ready for analysis by handling missing values, outliers, and transforming categorical variables into numerical formats.

- **Exploratory Data Analysis (EDA)**

  Perform EDA to uncover initial insights and visualize relationships between variables.

- **Model Development**

  Use various machine learning algorithms (e.g., linear regression, random forest, gradient boosting) to develop and validate the predictive model.

- **Model Evaluation**

  Assess the model's performance using metrics such as R-squared, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE).

- **Feature Importance**

  Identify and rank the importance of different features in predicting resale prices.

5. Detailed Aspects and Analysis of the Project

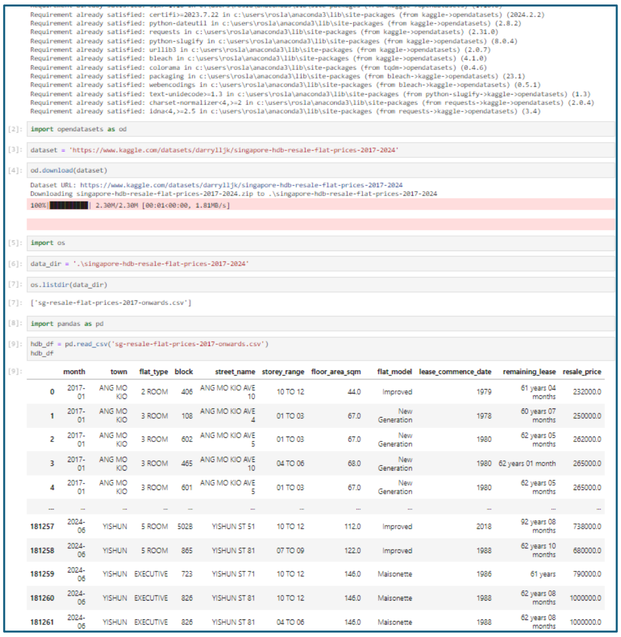
5.1 Data Source and Preparation

- **Dataset**

  This dataset includes resale price information for HDB flats in Singapore, covering the period from January 2017 to June 2024. It contains records from reliable sources such as the URA, and other relevant databases. The dataset is comprehensive, showing details such as transaction month, town, flat type, block, street name, storey range, floor area, flat model, lease commencement date, remaining lease, and resale price.

  For this project, the dataset was downloaded from Kaggle using Python code, which originally sourced it from data.gov.sg.

[Singapore Resale Flat Prices (2017-2024) (kaggle.com)]



```
Requirement already satisfied: certifi>=2023.7.22 in c:\users\rosla\anaconda3\lib\site-packages (from kaggle->opendatasets) (2024.2.2)
Requirement already satisfied: python-dateutil in c:\users\rosla\anaconda3\lib\site-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: requests in c:\users\rosla\anaconda3\lib\site-packages (from kaggle->opendatasets) (2.31.0)
Requirement already satisfied: python-slugify in c:\users\rosla\anaconda3\lib\site-packages (from kaggle->opendatasets) (8.0.4)
Requirement already satisfied: urllib3 in c:\users\rosla\anaconda3\lib\site-packages (from kaggle->opendatasets) (2.0.7)
Requirement already satisfied: bleach in c:\users\rosla\anaconda3\lib\site-packages (from kaggle->opendatasets) (4.1.0)
Requirement already satisfied: colorama in c:\users\rosla\anaconda3\lib\site-packages (from tqdm->opendatasets) (0.4.6)
Requirement already satisfied: packaging in c:\users\rosla\anaconda3\lib\site-packages (from bleach->kaggle->opendatasets) (23.1)
Requirement already satisfied: webencodings in c:\users\rosla\anaconda3\lib\site-packages (from bleach->kaggle->opendatasets) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in c:\users\rosla\anaconda3\lib\site-packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\rosla\anaconda3\lib\site-packages (from requests->kaggle->opendatasets) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\rosla\anaconda3\lib\site-packages (from requests->kaggle->opendatasets) (3.4)

[2]: import opendatasets as od

[3]: dataset = 'https://www.kaggle.com/datasets/darrylljk/singapore-hdb-resale-flat-prices-2017-2024'

[4]: od.download(dataset)

Dataset URL: https://www.kaggle.com/datasets/darrylljk/singapore-hdb-resale-flat-prices-2017-2024
Downloading singapore-hdb-resale-flat-prices-2017-2024.zip to .\singapore-hdb-resale-flat-prices-2017-2024
100%|██████████| 2.30M/2.30M [00:01<00:00, 1.81MB/s]

[5]: import os

[6]: data_dir = '.\singapore-hdb-resale-flat-prices-2017-2024'

[7]: os.listdir(data_dir)

[7]: ['sg-resale-flat-prices-2017-onwards.csv']

[8]: import pandas as pd

[9]: hdb_df = pd.read_csv('sg-resale-flat-prices-2017-onwards.csv')
hdb_df
```

	month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	remaining_lease	resale_price
0	2017-01	ANG MO KIO	2 ROOM	406	ANG MO KIO AVE 10	10 TO 12	44.0	Improved	1979	61 years 04 months	232000.0
1	2017-01	ANG MO KIO	3 ROOM	108	ANG MO KIO AVE 4	01 TO 03	67.0	New Generation	1978	60 years 07 months	250000.0
2	2017-01	ANG MO KIO	3 ROOM	602	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	262000.0
3	2017-01	ANG MO KIO	3 ROOM	465	ANG MO KIO AVE 10	04 TO 06	68.0	New Generation	1980	62 years 01 month	265000.0
4	2017-01	ANG MO KIO	3 ROOM	601	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	265000.0
...	...	...	...	...	...	...	...	...	...	...	...
181257	2024-06	YISHUN	5 ROOM	502B	YISHUN ST 51	10 TO 12	112.0	Improved	2018	92 years 08 months	738000.0
181258	2024-06	YISHUN	5 ROOM	865	YISHUN ST 81	07 TO 09	122.0	Improved	1988	62 years 10 months	680000.0
181259	2024-06	YISHUN	EXECUTIVE	723	YISHUN ST 71	10 TO 12	146.0	Maisonette	1986	61 years	790000.0
181260	2024-06	YISHUN	EXECUTIVE	826	YISHUN ST 81	10 TO 12	146.0	Maisonette	1988	62 years 08 months	1000000.0
181261	2024-06	YISHUN	EXECUTIVE	826	YISHUN ST 81	04 TO 06	146.0	Maisonette	1988	62 years 08 months	1000000.0

Click below to see code shown on the left. This is to download the source datafile from Kaggle.

kaggle_download.ipynb

181262 rows × 11 columns

[12]:
```python
# for data manipulation

import numpy as np
import pandas as pd
```

•[13]:
```python
# Finding out the content of the dataset

hdb_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 181262 entries, 0 to 181261
Data columns (total 11 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   month                181262 non-null  object
 1   town                 181262 non-null  object
 2   flat_type            181262 non-null  object
 3   block                181262 non-null  object
 4   street_name          181262 non-null  object
 5   storey_range         181262 non-null  object
 6   floor_area_sqm       181262 non-null  float64
 7   flat_model           181262 non-null  object
 8   lease_commence_date  181262 non-null  int64
 9   remaining_lease      181262 non-null  object
 10  resale_price         181262 non-null  float64
dtypes: float64(2), int64(1), object(8)
memory usage: 15.2+ MB
```

To find out the content and manipulate the dataset, I imported the Pandas and Numpy libraries, and used the command code hdb_df.info() (see above). This reveals that the dataset contains **181,262 entries** and **11 feature columns**. These columns are: 'month', 'town', 'flat_type', 'block', 'street_name', 'storey_range', 'floor_area_sqm', 'flat_model', 'lease_commence_date', 'remaining_lease', and 'resale_price'.

All feature columns, except for three, contains categorical datatypes. The features 'floor_area_sqm' and 'resale_price' are of the float datatype, while the feature 'lease_commence_date' is of the integer data type.

Importantly, the entries to all the features are complete ie there are no null entries or missing values.

- **Data Cleaning**

<u>Missing values</u>
Since all entries are complete, there is no adjustment needed at this stage. This ensures a smooth transition to the next steps in the data preparation and analysis process.

<u>Feature removal</u>
In the initial stages of this data preparation, features deemed inconsequential to the resale price will be removed. The selection is currently based on a judgement call, leveraging my familiarity and understanding of the housing

market. Later, a correlation analysis will be conducted to determine which of the remaining features significantly impact the resale price.

The features to remove are ‘street_name’, ‘flat_model’ and ‘block’.

Shown below is the code to remove the features stated (line 7). I have at this point also added the code ‘pd.set_option('display.max_columns', None)’ and imported matplotlib together with the seaborn libraries for data visualization. The code in line 1 is to allow full display of columns in the dataset. Also at this point, I have renamed the code file to ‘hdb_dataset_dropped_features.ipynb’

```
 9   remaining_lease       181262 non-null  object
 10  resale_price          181262 non-null  float64
dtypes: float64(2), int64(1), object(8)
memory usage: 15.2+ MB
```

[20]:
```
1  pd.set_option('display.max_columns', None)
2
3  # for visualization
4  import matplotlib.pyplot as plt
5  import seaborn as sns
6
7  hdb_df = hdb_df.drop(hdb_df[['street_name', 'flat_model', 'block']], axis=1)
8
9  hdb_df
10
```

[20]:

	month	town	flat_type	storey_range	floor_area_sqm	lease_commence_date	remaining_lease	resale_price
0	2017-01	ANG MO KIO	2 ROOM	10 TO 12	44.0	1979	61 years 04 months	232000.0
1	2017-01	ANG MO KIO	3 ROOM	01 TO 03	67.0	1978	60 years 07 months	250000.0
2	2017-01	ANG MO KIO	3 ROOM	01 TO 03	67.0	1980	62 years 05 months	262000.0
3	2017-01	ANG MO KIO	3 ROOM	04 TO 06	68.0	1980	62 years 01 month	265000.0
4	2017-01	ANG MO KIO	3 ROOM	01 TO 03	67.0	1980	62 years 05 months	265000.0
...	...	...	...	...	...	...	...	...
181257	2024-06	YISHUN	5 ROOM	10 TO 12	112.0	2018	92 years 08 months	738000.0
181258	2024-06	YISHUN	5 ROOM	07 TO 09	122.0	1988	62 years 10 months	680000.0
181259	2024-06	YISHUN	EXECUTIVE	10 TO 12	146.0	1986	61 years	790000.0
181260	2024-06	YISHUN	EXECUTIVE	10 TO 12	146.0	1988	62 years 08 months	1000000.0
181261	2024-06	YISHUN	EXECUTIVE	04 TO 06	146.0	1988	62 years 08 months	1000000.0

181262 rows × 8 columns

Click below to see code shown on the left.

hdb_dataset_dropped_features.ipynb

<u>Determining outliers</u>

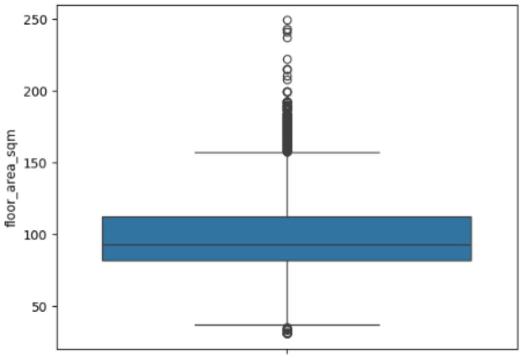
The dataset is now left with 8 feature columns with 2, ‘floor_area_sqm’ and ‘resale_price’, having numerical datatypes. Performing outlier analysis on these 2 features would reveal any outliers that exist within them. Outliers are caused by variability in the data and by a possible experimental measurement error. This would in turn result in various problems in the statistical analysis and may also cause a significant impact to the results ahead.

The boxplot analysis below for ‘floor_area_sqm’ indicates significant outliers, with many entries exceeding 160 square meters and falling below 45 square meters. Upon reviewing the smallest and largest floor areas for HDB flats in Singapore, which range from 45 to 186 square meters, it is reasonable to exclude entries below 45 square meters and above 186 square meters. This adjustment will help ensure the dataset accurately represents typical HDB flat sizes.

Furthermore, there are only a total 976 out of 181262 entries (refer to second figure below), strengthening the rationale for their removal.

```
[9]: # Outlier analysis on 'floor_area_sqm'

sns.boxplot(y='floor_area_sqm', data=hdb_df)
```

[9]: <Axes: ylabel='floor_area_sqm'>




```
[ ]:
```

```
[23]: hdb_df_above_186 = hdb_df[hdb_df['floor_area_sqm'] > 186]
hdb_df_above_186
```

[23]:

	month	town	flat_type	storey_range	floor_area_sqm	lease_commence_date	remaining_lease	resale_price
2187	2017-02	WOODLANDS	EXECUTIVE	10 TO 12	192.0	1994	75 years 11 months	760000.0
4060	2017-03	WOODLANDS	EXECUTIVE	07 TO 09	189.0	1994	76 years 01 month	800000.0
5890	2017-04	WOODLANDS	EXECUTIVE	07 TO 09	192.0	1994	75 years 09 months	729000.0
7854	2017-05	WOODLANDS	EXECUTIVE	10 TO 12	189.0	1994	75 years 08 months	765000.0
8868	2017-06	KALLANG/WHAMPOA	3 ROOM	01 TO 03	215.0	1972	54 years 01 month	830000.0
...	...	...	...	...	...	...	...	...
169006	2023-12	WOODLANDS	EXECUTIVE	04 TO 06	192.0	1994	69 years 02 months	1038000.0
171559	2024-01	WOODLANDS	EXECUTIVE	04 TO 06	192.0	1994	69 years 02 months	930000.0
172801	2024-02	KALLANG/WHAMPOA	3 ROOM	01 TO 03	208.0	1972	47 years 06 months	1280000.0
175798	2024-03	WOODLANDS	EXECUTIVE	04 TO 06	189.0	1994	69 years 03 months	980000.0
179561	2024-05	KALLANG/WHAMPOA	3 ROOM	01 TO 03	189.4	1972	47 years 03 months	1150000.0

135 rows × 8 columns

```
[33]: hdb_df_below_90 = hdb_df[hdb_df['floor_area_sqm'] < 45]
hdb_df_below_90
```

[33]:

	month	town	flat_type	storey_range	floor_area_sqm	lease_commence_date	remaining_lease	resale_price
0	2017-01	ANG MO KIO	2 ROOM	10 TO 12	44.0	1979	61 years 04 months	232000.0
195	2017-01	BUKIT MERAH	2 ROOM	07 TO 09	34.0	1971	53 years 06 months	218000.0
196	2017-01	BUKIT MERAH	2 ROOM	07 TO 09	34.0	1971	53 years 06 months	230000.0
383	2017-01	GEYLANG	2 ROOM	07 TO 09	42.0	1971	53 years	205000.0
384	2017-01	GEYLANG	2 ROOM	10 TO 12	42.0	1971	53 years	215000.0
...	...	...	...	...	...	...	...	...
180896	2024-06	BUKIT MERAH	2 ROOM	07 TO 09	43.0	1970	44 years 10 months	250000.0
181013	2024-06	JURONG WEST	2 ROOM	10 TO 12	38.0	2019	94 years	330000.0
181065	2024-06	PUNGGOL	2 ROOM	19 TO 21	38.0	2018	93 years 06 months	332000.0
181102	2024-06	SEMBAWANG	2 ROOM	07 TO 09	38.0	2019	94 years	327000.0
181227	2024-06	YISHUN	2 ROOM	04 TO 06	38.0	2019	94 years 04 months	322888.0

841 rows × 8 columns

The figures below show the dataset after removal of these outliers, leaving 180286 rows.

```
[44]: # removing outliers by dropping the impacted rows

hdb_df = hdb_df.drop(index=hdb_df_above_186.index)
hdb_df
```

[44]:

	month	town	flat_type	storey_range	floor_area_sqm	lease_commence_date	remaining_lease	resale_price
0	2017-01	ANG MO KIO	2 ROOM	10 TO 12	44.0	1979	61 years 04 months	232000.0
1	2017-01	ANG MO KIO	3 ROOM	01 TO 03	67.0	1978	60 years 07 months	250000.0
2	2017-01	ANG MO KIO	3 ROOM	01 TO 03	67.0	1980	62 years 05 months	262000.0
3	2017-01	ANG MO KIO	3 ROOM	04 TO 06	68.0	1980	62 years 01 month	265000.0
4	2017-01	ANG MO KIO	3 ROOM	01 TO 03	67.0	1980	62 years 05 months	265000.0
...	...	...	...	...	...	...	...	...
181257	2024-06	YISHUN	5 ROOM	10 TO 12	112.0	2018	92 years 08 months	738000.0
181258	2024-06	YISHUN	5 ROOM	07 TO 09	122.0	1988	62 years 10 months	680000.0
181259	2024-06	YISHUN	EXECUTIVE	10 TO 12	146.0	1986	61 years	790000.0
181260	2024-06	YISHUN	EXECUTIVE	10 TO 12	146.0	1988	62 years 08 months	1000000.0
181261	2024-06	YISHUN	EXECUTIVE	04 TO 06	146.0	1988	62 years 08 months	1000000.0

181127 rows × 8 columns

```
[45]: # removing outliers by dropping the impacted rows

hdb_df = hdb_df.drop(index=hdb_df_below_90.index)
hdb_df
```

[45]:

	month	town	flat_type	storey_range	floor_area_sqm	lease_commence_date	remaining_lease	resale_price
1	2017-01	ANG MO KIO	3 ROOM	01 TO 03	67.0	1978	60 years 07 months	250000.0
2	2017-01	ANG MO KIO	3 ROOM	01 TO 03	67.0	1980	62 years 05 months	262000.0
3	2017-01	ANG MO KIO	3 ROOM	04 TO 06	68.0	1980	62 years 01 month	265000.0
4	2017-01	ANG MO KIO	3 ROOM	01 TO 03	67.0	1980	62 years 05 months	265000.0
5	2017-01	ANG MO KIO	3 ROOM	01 TO 03	68.0	1981	63 years	275000.0
...	...	...	...	...	...	...	...	...
181257	2024-06	YISHUN	5 ROOM	10 TO 12	112.0	2018	92 years 08 months	738000.0
181258	2024-06	YISHUN	5 ROOM	07 TO 09	122.0	1988	62 years 10 months	680000.0
181259	2024-06	YISHUN	EXECUTIVE	10 TO 12	146.0	1986	61 years	790000.0
181260	2024-06	YISHUN	EXECUTIVE	10 TO 12	146.0	1988	62 years 08 months	1000000.0
181261	2024-06	YISHUN	EXECUTIVE	04 TO 06	146.0	1988	62 years 08 months	1000000.0

180286 rows × 8 columns

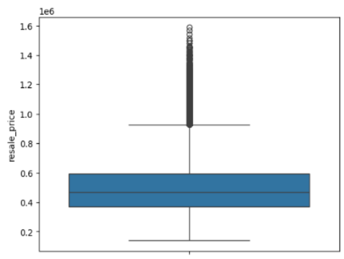
Click below to see code shown on the left.

hdb_dataset_outlier_removal.ipynb

Below is the outlier analysis for ‘resale_price’. The boxplot reveals numerous data points outside the upper bound of Q3 + 1.5 x IQR. Further investigation (see the second figure below) indicates a total of 3858 entries in this category. A check on the maximum transacted price (third figure below) reveals that the highest transaction is at $1588000, which although seemingly high, accurately reflects the current market trend, particularly on the higher end model. As a result, I have decided to retain all these outliers.

```
[15]: # Outlier analysis on 'resale_price'

sns.boxplot(y='resale_price', data=hdb_df)
```

[15]: <Axes: ylabel='resale_price'>



Click below to see code shown on the left.

hdb_dataset_price_
outlier_removal.ipyn

```
[15]: hdb_df_above_up = hdb_df[hdb_df['resale_price'] > 927500]
      hdb_df_above_up
```

[15]:

	month	town	flat_type	storey_range	floor_area_sqm	lease_commence_date	remaining_lease	resale_price
310	2017-01	CENTRAL AREA	5 ROOM	49 TO 51	105.0	2011	93 years	1108000.0
664	2017-01	KALLANG/WHAMPOA	5 ROOM	25 TO 27	119.0	2011	93 years	1005000.0
1329	2017-02	BISHAN	5 ROOM	40 TO 42	120.0	2011	93 years 07 months	1180000.0
1397	2017-02	BUKIT MERAH	5 ROOM	19 TO 21	115.0	2013	95 years 03 months	955000.0
1398	2017-02	BUKIT MERAH	5 ROOM	34 TO 36	114.0	2013	95 years 03 months	980000.0
...	...	...	...	...	...	...	...	...
181178	2024-06	TAMPINES	EXECUTIVE	01 TO 03	146.0	1989	64 years	960000.0
181195	2024-06	TOA PAYOH	4 ROOM	16 TO 18	93.0	2019	94 years 08 months	940000.0
181197	2024-06	TOA PAYOH	5 ROOM	13 TO 15	153.0	1984	59 years 02 months	1130000.0
181260	2024-06	YISHUN	EXECUTIVE	10 TO 12	146.0	1988	62 years 08 months	1000000.0
181261	2024-06	YISHUN	EXECUTIVE	04 TO 06	146.0	1988	62 years 08 months	1000000.0

3858 rows × 8 columns

```
[14]: hdb_df['resale_price'].max()

[14]: 1588000.0
```

- **Feature Engineering**

The dataset contains several categorical features that need to be converted to numerical data types or one-hot encoding.

For a start, I will convert the ‘remaining_lease’ into months to ensure it is suitable for machine learning models. The ‘remaining_lease’ feature is significant as it directly impacts buyer behaviour. Thereafter it will be checked for reasonableness, considering that Singapore’s HDB lease is 99 years or 1188 months.

These other categorical features such as ‘town’, ‘flat_type’, and ‘storey_range’, will be dealt with at a later stage.

The figure below shows the conversion of the ‘remaining_lease’ feature from years-and-months format to a months-only format, resulting in a new feature named ‘lease_remaining’, and the removal of ‘remaining_lease’ column. The second figure (refer to figure below it) shows that none of the values in the ‘lease_remaining’ exceeds 1188 months.

[21]:
```
# redefining 'remaining_lease' feature from years and months into months.
# creating a new feature column 'lease_remaining'

def convert_to_months(remaining_lease):
    parts = remaining_lease.split()
    years = 0
    months = 0

    for i in range(len(parts)):
        if 'year' in parts[i]:
            years = int(parts[i-1])
        elif 'month' in parts[i]:
            months = int(parts[i-1])

    total_months = years * 12 + months
    return total_months

hdb_df['lease_remaining'] = hdb_df['remaining_lease'].apply(convert_to_months)

hdb_df = hdb_df.drop('remaining_lease', axis=1)

hdb_df
```

[21]:

	month	town	flat_type	storey_range	floor_area_sqm	lease_commence_date	resale_price	lease_remaining
1	2017-01	ANG MO KIO	3 ROOM	01 TO 03	67.0	1978	250000.0	727
2	2017-01	ANG MO KIO	3 ROOM	01 TO 03	67.0	1980	262000.0	749
3	2017-01	ANG MO KIO	3 ROOM	04 TO 06	68.0	1980	265000.0	745
4	2017-01	ANG MO KIO	3 ROOM	01 TO 03	67.0	1980	265000.0	749
5	2017-01	ANG MO KIO	3 ROOM	01 TO 03	68.0	1981	275000.0	756
...	...	...	...	...	...	...	...	...
181257	2024-06	YISHUN	5 ROOM	10 TO 12	112.0	2018	738000.0	1112
181258	2024-06	YISHUN	5 ROOM	07 TO 09	122.0	1988	680000.0	754
181259	2024-06	YISHUN	EXECUTIVE	10 TO 12	146.0	1986	790000.0	732
181260	2024-06	YISHUN	EXECUTIVE	10 TO 12	146.0	1988	1000000.0	752
181261	2024-06	YISHUN	EXECUTIVE	04 TO 06	146.0	1988	1000000.0	752

180286 rows × 8 columns

[23]:
```
# check if any of lease remaining is above 1188 months ie greater than 99 years

hdb_df_above_1188 = hdb_df[hdb_df['lease_remaining'] > 1188]
display(len(hdb_df_above_1188))
```

0

[ ]:

Click below to see code shown on the left.

hdb_dataset_remaining_lease_redefine

With the redefinition of ‘remaining_lease’, the features ‘month’ and ‘lease_commence_date’ have become redundant and are removed from the dataset (refer to figure below).

[24]:
```
# dropping the 'month' and 'lease_commence_date' from dataset

hdb_df = hdb_df.drop(['month', 'lease_commence_date'], axis=1)
hdb_df
```

[24]:

	town	flat_type	storey_range	floor_area_sqm	resale_price	lease_remaining
1	ANG MO KIO	3 ROOM	01 TO 03	67.0	250000.0	727
2	ANG MO KIO	3 ROOM	01 TO 03	67.0	262000.0	749
3	ANG MO KIO	3 ROOM	04 TO 06	68.0	265000.0	745
4	ANG MO KIO	3 ROOM	01 TO 03	67.0	265000.0	749
5	ANG MO KIO	3 ROOM	01 TO 03	68.0	275000.0	756
...	...	...	...	...	...	...
181257	YISHUN	5 ROOM	10 TO 12	112.0	738000.0	1112
181258	YISHUN	5 ROOM	07 TO 09	122.0	680000.0	754
181259	YISHUN	EXECUTIVE	10 TO 12	146.0	790000.0	732
181260	YISHUN	EXECUTIVE	10 TO 12	146.0	1000000.0	752
181261	YISHUN	EXECUTIVE	04 TO 06	146.0	1000000.0	752

180286 rows × 6 columns

Click below to see code shown on the left.

hdb_dataset_remaining_lease_redfined_final.ipynb

5.2 Exploratory Data Analysis (EDA)

- **Descriptive Statistics**

```
[33]: hdb_df.describe()
```

	floor_area_sqm	resale_price	lease_remaining
count	180286.000000	1.802860e+05	180286.000000
mean	97.344420	4.987337e+05	895.699084
std	23.628913	1.718718e+05	167.015426
min	45.000000	1.400000e+05	498.000000
25%	82.000000	3.700000e+05	759.000000
50%	93.000000	4.688880e+05	894.000000
75%	112.000000	5.930000e+05	1060.000000
max	186.000000	1.588000e+06	1173.000000

<u>Key Insights</u>

Floor Area:

The majority of the flats have a floor area between 82 sqm and 112 sqm, with a mean of 97.34 sqm. This suggests that most flats are medium-sized.

Resale Price:

The resale prices vary significantly, with a mean price of around $498,733.70. The prices are skewed towards higher values, as indicated by the maximum price of $1,588,000.

Lease Remaining:

The lease remaining for most flats is quite high, with a median of 74.5 years. This indicates that the majority of the flats have a substantial amount of lease remaining, which is a positive factor for potential buyers.

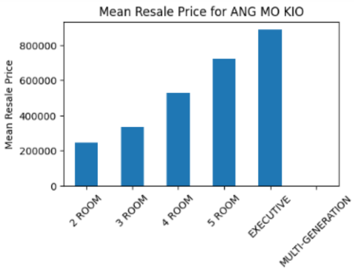
- **Visualization**

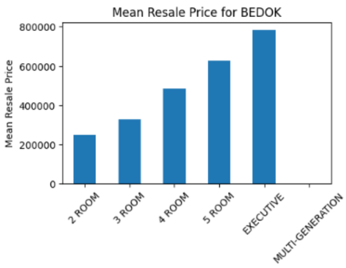
Graphical visualizations are powerful tools in data analysis, enabling the identification of trends and patterns that support informed decision-making. In this project, however, the primary purpose of the graphs is to validate the data. For instance, we expect that the price will increase as the floor size increases across all towns, as shown below.

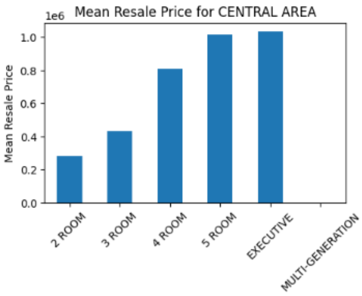
```
•[33]: # barchart comparing mean prices of 'flat_type' by 'town'

mean_resale_price = hdb_df.groupby(['town', 'flat_type'])['resale_price'].mean().unstack()

towns = mean_resale_price.index

for town in towns:
    plt.figure(figsize=(5, 3))
    mean_resale_price.loc[town].plot(kind='bar')
    plt.title(f'Mean Resale Price for {town}')
    plt.xlabel('Flat Type')
    plt.ylabel('Mean Resale Price')
    plt.xticks(rotation=45)
    plt.show()
```

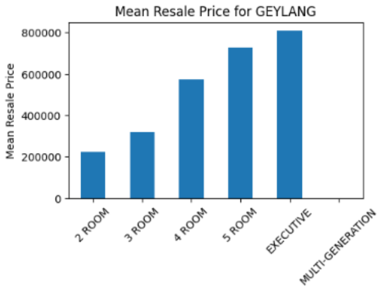
Click below to see code shown on the left.

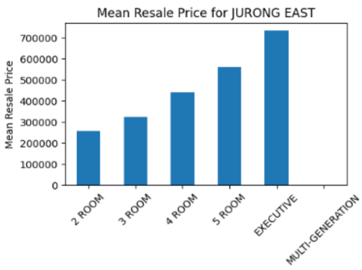
hdb_dataset_statistics_visualization.ipynb

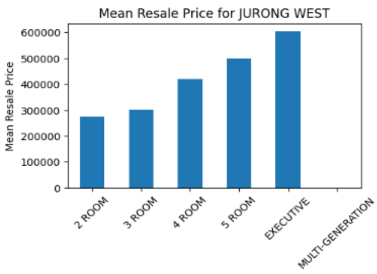










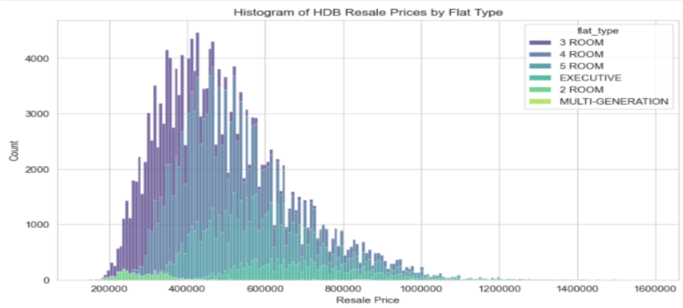


In the next analysis, the histogram (refer to figure below) provides a clear visual representation of the distribution of HDB resale prices across different flat types.

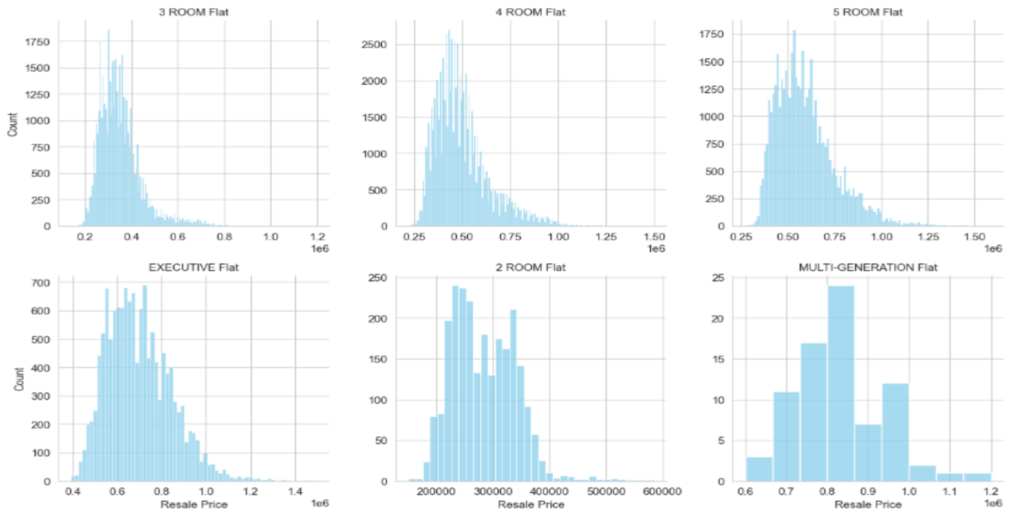
Click below to see code shown on the left.

hdb_resale_prices_by_towns.ipynb

```
[24]: # Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Create a histogram
plt.figure(figsize=(10, 6))
sns.histplot(data=hdb_df, x='resale_price', hue='flat_type', multiple='stack', palette='viridis')

# Remove scientific notation from the x-axis
plt.ticklabel_format(style='plain', axis='x')

# Add titles and labels
plt.title('Histogram of HDB Resale Prices by Flat Type')
plt.xlabel('Resale Price')
plt.ylabel('Count')

# Show the plot
plt.show()
```




```
[25]: # Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Create a FacetGrid
g = sns.FacetGrid(hdb_df, col='flat_type', col_wrap=3, height=4, sharex=False, sharey=False)

# Map the histogram to the grid
g.map(sns.histplot, 'resale_price', kde=False, color='skyblue')

# Add titles and labels
g.set_titles("{col_name} Flat")
g.set_axis_labels('Resale Price', 'Count')

# Show the plot
plt.show()
```



Here are the observations:

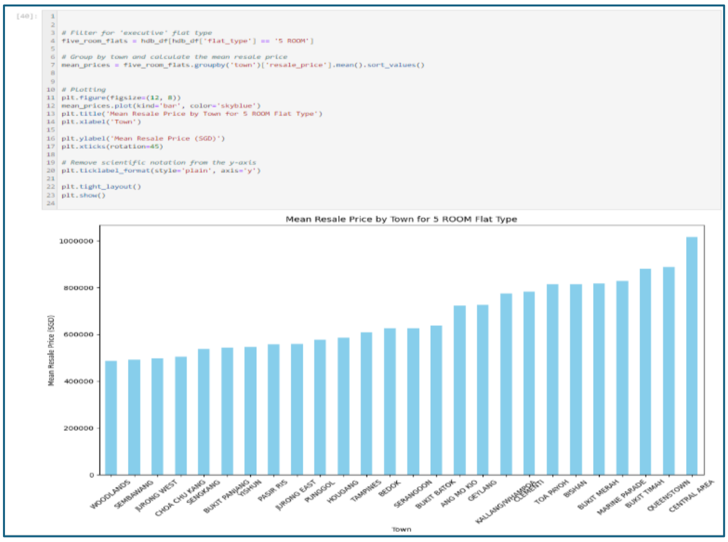
Price Distribution
The resale prices for 3 ROOM and 4 ROOM flats are more concentrated in the lower price ranges, while EXECUTIVE and MULTI-GENERATIONAL flats have a wider spread extending into higher price range.

Market Trends
The higher count of 4 ROOM flats in the mid-price range suggests that they are quite popular, possibly to a balance of size and affordability.

Investment Insights
For potential investors or buyers, understanding these distributions can help make informed decisions based on budget and flat type preferences.

Policy Implications
This data could be useful for policymakers to understand housing affordability and demand trends, potentially guiding future housing policies.

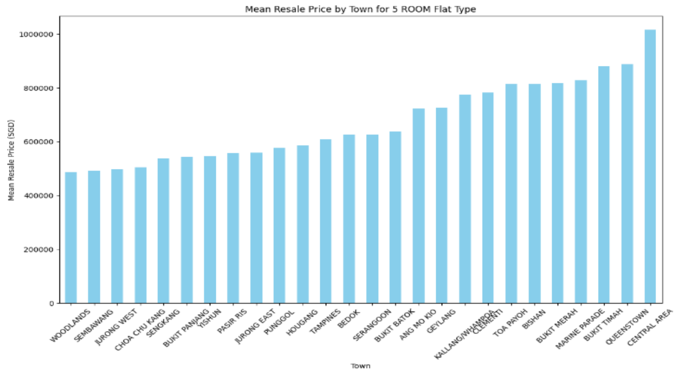
From another perspective, when comparing the mean resale prices across various towns by flat types, it becomes evident that certain areas, such as Queenstown, the Central Areas, Bishan, and Ang Mo Kio, consistently have higher prices compared to other towns. This may explain the right-tail distribution of the mean prices across all flat types as shown in the previous graphs. This trend for the mean resale prices across various towns is illustrated in the charts below.



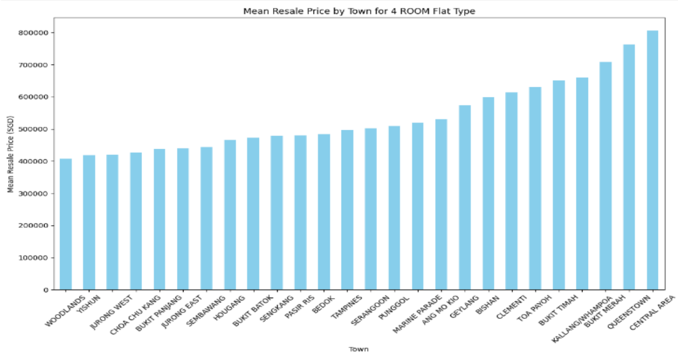
Click below to see code shown on the left.

hdb_stats_meanpricebyflats_across_towns.ipynb

```python
[40]:


# Filter for 'executive' flat type
five_room_flats = hdb_df[hdb_df['flat_type'] == '5 ROOM']

# Group by town and calculate the mean resale price
mean_prices = five_room_flats.groupby('town')['resale_price'].mean().sort_values()


# Plotting
plt.figure(figsize=(12, 8))
mean_prices.plot(kind='bar', color='skyblue')
plt.title('Mean Resale Price by Town for 5 ROOM Flat Type')
plt.xlabel('Town')

plt.ylabel('Mean Resale Price (SGD)')
plt.xticks(rotation=45)

# Remove scientific notation from the y-axis
plt.ticklabel_format(style='plain', axis='y')

plt.tight_layout()
plt.show()
```




```python
[41]:

# Filter for 'executive' flat type
four_room_flats = hdb_df[hdb_df['flat_type'] == '4 ROOM']

# Group by town and calculate the mean resale price
mean_prices = four_room_flats.groupby('town')['resale_price'].mean().sort_values()


# Plotting
plt.figure(figsize=(12, 8))
mean_prices.plot(kind='bar', color='skyblue')
plt.title('Mean Resale Price by Town for 4 ROOM Flat Type')
plt.xlabel('Town')

plt.ylabel('Mean Resale Price (SGD)')
plt.xticks(rotation=45)

# Remove scientific notation from the y-axis
plt.ticklabel_format(style='plain', axis='y')

plt.tight_layout()
plt.show()
```



Several factors could contribute to the higher resale prices in towns like Queenstown, the Central Areas, Bishan, and Ang Mo Kio, and within generally accepted expectations.

<u>Location and Accessibility</u>

Proximity to the City Centre: Areas closer to the Central Business District (CBD) and city centre, like Queenstown and the Central Areas, tend to have higher property values due to their prime location.

<u>Transportation Links</u>
These towns often have excellent public transport links, including MRT stations and bus services, making them highly accessible.

<u>Amenities and Facilities</u>
Educational Institutions: Presence of reputable schools and educational institutions can drive up property prices as families seek to live near good schools.

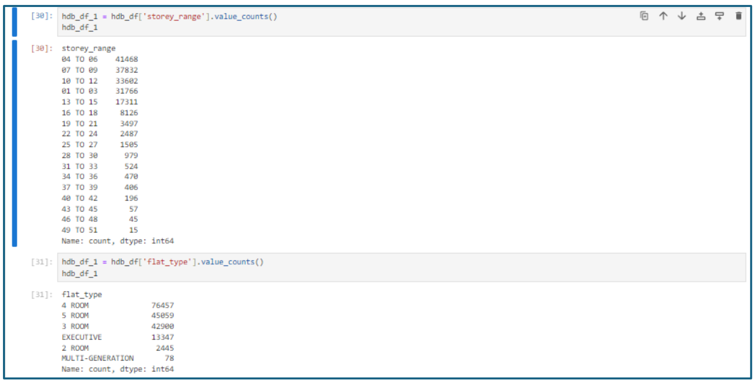
<u>Healthcare Facilities</u>
Proximity to hospitals and clinics adds to the desirability of these areas.
Shopping and Entertainment: Availability of shopping malls, restaurants, and entertainment options enhances the attractiveness of these towns.

<u>Market Demand</u>
High Demand: These towns may have a higher demand due to their desirable attributes, leading to increased competition and higher prices.

<u>Limited Supply</u>
Limited availability of flats in these prime locations tends to drive prices up.

- **Correlation Analysis**

To prepare for the correlation analysis, the remaining categorical features ('town', 'flat_type', and 'storey_range') are converted into numerical data types (refer to the figure below). The 'town' feature, which has a larger number of categories (see the second figure below), will additionally be one-hot encoded later in preparation for machine learning. The third and fourth figures below display the converted categorical features.



```
[30]: hdb_df_1 = hdb_df['storey_range'].value_counts()
      hdb_df_1

[30]: storey_range
      04 TO 06    41468
      07 TO 09    37832
      10 TO 12    33602
      01 TO 03    31766
      13 TO 15    17311
      16 TO 18     8126
      19 TO 21     3497
      22 TO 24     2487
      25 TO 27     1505
      28 TO 30      979
      31 TO 33      524
      34 TO 36      470
      37 TO 39      406
      40 TO 42      196
      43 TO 45       57
      46 TO 48       45
      49 TO 51       15
      Name: count, dtype: int64

[31]: hdb_df_1 = hdb_df['flat_type'].value_counts()
      hdb_df_1

[31]: flat_type
      4 ROOM              76457
      5 ROOM              45059
      3 ROOM              42900
      EXECUTIVE           13347
      2 ROOM               2445
      MULTI-GENERATION       78
      Name: count, dtype: int64
```

Click below to see code shown on the left.

hdb_dataset_statistics_numerical_features.ipynb

```
[32]: hdb_df_1 = hdb_df['town'].value_counts()
      hdb_df_1
```

```
[32]: town
SENGKANG           14963
PUNGGOL            13392
WOODLANDS          12722
YISHUN             12370
TAMPINES           12139
JURONG WEST        11967
BEDOK               9608
HOUGANG             9134
CHOA CHU KANG       8291
ANG MO KIO          7358
BUKIT BATOK         6906
BUKIT MERAH         6829
BUKIT PANJANG       6664
TOA PAYOH           5528
KALLANG/WHAMPOA     5423
PASIR RIS           5387
QUEENSTOWN          4996
SEMBAWANG           4914
GEYLANG             4365
CLEMENTI            4061
JURONG EAST         3676
SERANGOON           3293
BISHAN              3271
CENTRAL AREA        1461
MARINE PARADE       1112
BUKIT TIMAH          456
Name: count, dtype: int64
```

```
[ ]:
```

```
[30]: from sklearn.preprocessing import LabelEncoder

      # Initialize the LabelEncoder
      le = LabelEncoder()

      # Fit and transform the categorical variable
      hdb_df['flat_type'] = le.fit_transform(hdb_df['flat_type'])
      hdb_df
```

[30]:

	town	flat_type	storey_range	floor_area_sqm	resale_price	lease_remaining
1	ANG MO KIO	1	01 TO 03	67.0	250000.0	727
2	ANG MO KIO	1	01 TO 03	67.0	262000.0	749
3	ANG MO KIO	1	04 TO 06	68.0	265000.0	745
4	ANG MO KIO	1	01 TO 03	67.0	265000.0	749
5	ANG MO KIO	1	01 TO 03	68.0	275000.0	756
...	...	...	...	...	...	...
181257	YISHUN	3	10 TO 12	112.0	738000.0	1112
181258	YISHUN	3	07 TO 09	122.0	680000.0	754
181259	YISHUN	4	10 TO 12	146.0	790000.0	732
181260	YISHUN	4	10 TO 12	146.0	1000000.0	752
181261	YISHUN	4	04 TO 06	146.0	1000000.0	752

180286 rows × 6 columns

```
[31]: hdb_df['storey_range'] = le.fit_transform(hdb_df['storey_range'])
      hdb_df
```

[31]:

	town	flat_type	storey_range	floor_area_sqm	resale_price	lease_remaining
1	ANG MO KIO	1	0	67.0	250000.0	727
2	ANG MO KIO	1	0	67.0	262000.0	749
3	ANG MO KIO	1	1	68.0	265000.0	745
4	ANG MO KIO	1	0	67.0	265000.0	749
5	ANG MO KIO	1	0	68.0	275000.0	756
...	...	...	...	...	...	...
181257	YISHUN	3	3	112.0	738000.0	1112
181258	YISHUN	3	2	122.0	680000.0	754
181259	YISHUN	4	3	146.0	790000.0	732
181260	YISHUN	4	3	146.0	1000000.0	752
181261	YISHUN	4	1	146.0	1000000.0	752

180286 rows × 6 columns

```
[32]: hdb_df['town'] = le.fit_transform(hdb_df['town'])
      hdb_df
```

[32]:

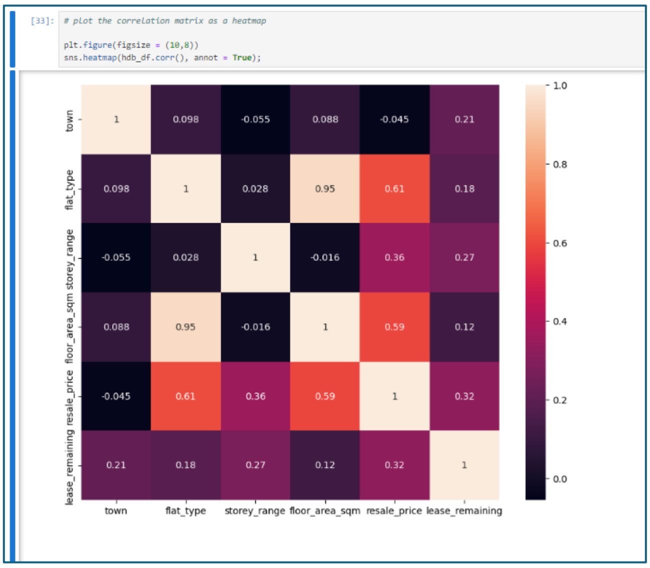
	town	flat_type	storey_range	floor_area_sqm	resale_price	lease_remaining
1	0	1	0	67.0	250000.0	727
2	0	1	0	67.0	262000.0	749
3	0	1	1	68.0	265000.0	745
4	0	1	0	67.0	265000.0	749
5	0	1	0	68.0	275000.0	756
...	...	...	...	...	...	...
181257	25	3	3	112.0	738000.0	1112
181258	25	3	2	122.0	680000.0	754
181259	25	4	3	146.0	790000.0	732
181260	25	4	3	146.0	1000000.0	752
181261	25	4	1	146.0	1000000.0	752

180286 rows × 6 columns

To identify features that are correlated with the resale price, a heat map is generated. The features ‘flat_type’ and ‘floor_area_sqm’ show strong positive correlations with ‘resale_price’, while ‘lease_remaining’ and ‘storey_range’ exhibit moderate positive correlations. Interestingly, the ‘town’ feature has a weak negative correlation index of –0.045.

This is unexpected as graphs of mean resale prices across towns indicate that some towns command a premium. One possible explanation for this could be outliers or anomalies in the data that affect the correlation coefficient. A few towns with extremely high or low prices might distort the overall correlation.

Despite it, this feature will be retained for further analysis. I plan to make adjustments at a later stage to improve the low correlation index observed in the data (refer to 'Model Adjustments' under the 'Feature Importance and Interpretation' section).

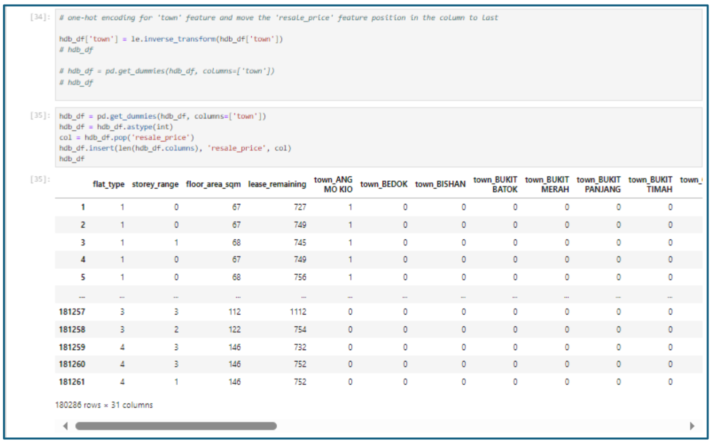


Click below to see code shown on the left.

hdb_dataset_statistics_corr.ipynb

- **One-hot encoding**

The steps below (refer to the figure and code) demonstrate the one-hot encoding of the 'town' feature and the repositioning of the 'resale_price' column to the end. The dataset now contains thirty-one columns, incorporating all twenty-six towns. The dataset is now prepared for the machine learning phase.



```
[34]: # one-hot encoding for 'town' feature and move the 'resale_price' feature position in the column to last

hdb_df['town'] = le.inverse_transform(hdb_df['town'])
# hdb_df

# hdb_df = pd.get_dummies(hdb_df, columns=['town'])
# hdb_df

[35]: hdb_df = pd.get_dummies(hdb_df, columns=['town'])
hdb_df = hdb_df.astype(int)
col = hdb_df.pop('resale_price')
hdb_df.insert(len(hdb_df.columns), 'resale_price', col)
hdb_df
```

[35]:

	flat_type	storey_range	floor_area_sqm	lease_remaining	town_ANG MO KIO	town_BEDOK	town_BISHAN	town_BUKIT BATOK	town_BUKIT MERAH	town_BUKIT PANJANG	town_BUKIT TIMAH	town_
1	1	0	67	727	1	0	0	0	0	0	0	
2	1	0	67	749	1	0	0	0	0	0	0	
3	1	1	68	745	1	0	0	0	0	0	0	
4	1	0	67	749	1	0	0	0	0	0	0	
5	1	0	68	756	1	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
181257	3	3	112	1112	0	0	0	0	0	0	0	
181258	3	2	122	754	0	0	0	0	0	0	0	
181259	4	3	146	732	0	0	0	0	0	0	0	
181260	4	3	146	752	0	0	0	0	0	0	0	
181261	4	1	146	752	0	0	0	0	0	0	0	

180286 rows × 31 columns

Click below to see code shown on the left.

hdb_dataset_statistics_OHE.ipynb

5.3 Model Development

- **Algorithm Selection**

In this section, I will explore various machine learning algorithms, in particular the Linear Regression, Decision Trees, and Random Forest, to predict resale prices. The reasons for selecting these algorithms are as follows:

<u>Linear Regression</u>

- Simplicity: It's a straightforward algorithm that is easy to implement and interpret.
- Efficiency: Works well with smaller datasets and provides a good baseline for comparison with more complex models.
- Interpretability: The coefficients can give insights into the relationship between features and the target variable.

<u>Decision Trees</u>

- Non-linearity: Can capture non-linear relationships between features and the target variable.
- Interpretability: The tree structure is easy to visualize and understand, making it clear how decisions are made.
- Feature Importance: Can provide insights into the importance of different features in predicting the target variable.

<u>Random Forest</u>

- Robustness: Combines multiple decision trees to reduce overfitting and improve generalization.
- Accuracy: Often provides better predictive performance compared to individual decision trees.
- Feature Importance: Aggregates feature importance from multiple trees, giving a more reliable measure of feature significance.

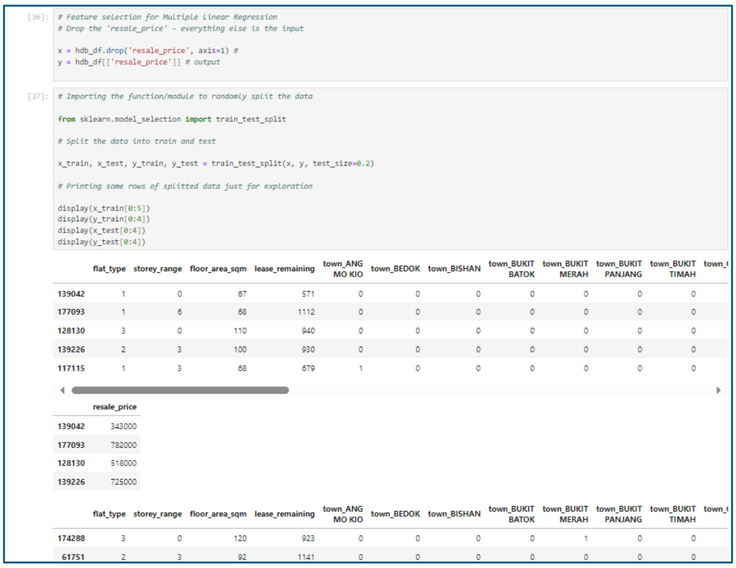
These algorithms offer a good mix of simplicity, interpretability, and predictive power, making them suitable for initial exploration and comparison in this machine learning project.

- **Preparation for Model Training**

In this process of feature selection for multiple linear regression, the target variable `resale_price` is isolated from the dataset to serve as the dependent variable, while all other columns are designated as independent variables for the model.

The independent variables, which represent various factors influencing resale prices, are stored in a new DataFrame (`x`) by dropping the `resale_price` column. Meanwhile, `resale_price` is set as the output variable (`y`) for the regression analysis.

This setup allows the model to use the input features to predict resale prices in a housing dataset, facilitating the next steps of splitting the data for training and testing, fitting the model, and making predictions (refer to figure below).



```
[36]: # Feature selection for Multiple Linear Regression
      # Drop the 'resale_price' - everything else is the input

      x = hdb_df.drop('resale_price', axis=1) #
      y = hdb_df[['resale_price']] # output
```

```
[37]: # Importing the function/module to randomly split the data

      from sklearn.model_selection import train_test_split

      # Split the data into train and test

      x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

      # Printing some rows of splitted data just for exploration

      display(x_train[0:5])
      display(y_train[0:4])
      display(x_test[0:4])
      display(y_test[0:4])
```

	flat_type	storey_range	floor_area_sqm	lease_remaining	town_ANG MO KIO	town_BEDOK	town_BISHAN	town_BUKIT BATOK	town_BUKIT MERAH	town_BUKIT PANJANG	town_BUKIT TIMAH	town_
139042	1	0	67	571	0	0	0	0	0	0	0	
177093	1	6	68	1112	0	0	0	0	0	0	0	
128130	3	0	110	940	0	0	0	0	0	0	0	
139226	2	3	100	930	0	0	0	0	0	0	0	
117115	1	3	68	679	1	0	0	0	0	0	0	

	resale_price
139042	343000
177093	782000
128130	518000
139226	725000

	flat_type	storey_range	floor_area_sqm	lease_remaining	town_ANG MO KIO	town_BEDOK	town_BISHAN	town_BUKIT BATOK	town_BUKIT MERAH	town_BUKIT PANJANG	town_BUKIT TIMAH	town_
174288	3	0	120	923	0	0	0	0	1	0	0	
61751	2	3	92	1141	0	0	0	0	0	0	0	

Click below to see code shown on the left.

hdb_dataset_split.ipynb

- **Model Training**

The code below demonstrates the steps involved in training the model. It starts by importing the required libraries and evaluation metrics, then proceeds to instantiate the model using Linear Regression. The model is then fitted to the training data, and finally, the results are generated.

```
[39]: # Training the Model

import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn import metrics

# Instantiate the linear regression model
lin_reg = LinearRegression()

# Fit the model to your training data (X_train and Y_train)
lin_reg.fit(x_train, y_train)

# Make predictions on the test data
linPrediction = lin_reg.predict(x_test)

# Calculate the Mean Absolute Error, R-squared score and Root Mean Squared Error
mae = metrics.mean_absolute_error(y_test, linPrediction)
r2 = metrics.r2_score(y_test, linPrediction)
rmse =  np.sqrt(metrics.mean_squared_error(y_test, linPrediction))

# Output the results
print("Mean Absolute Error:", "$", round(mae, 2))
print("R-squared score:",round(r2*100), "%")
print("Root Mean Squared Error score:", "$", round(rmse, 2) )

Mean Absolute Error: $ 73883.89
R-squared score: 72 %
Root Mean Squared Error score: $ 91456.3

[ ]:
```

Click below to see code shown on the left.

hdb_dataset_trained.ipynb

Note: Value may vary with each iteration

5.4 Model Evaluation

- **Performance Metrics**

In this section, the performance of this regression model is evaluated to determine its efficacy in predicting the target variable accurately and reliably. To assess the quality of this model, I used three key metrics: R-squared (R^2), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). By analysing these metrics, we can gauge the accuracy of the predictions, identify potential overfitting, and understand the model's strengths and weaknesses in capturing the underlying data patterns.

<u>Key results</u>

Mean Absolute Error (MAE): $73,883.89

This metric indicates the average magnitude of errors in predictions. In this case, the average difference between the predicted and actual values is around $73,883.89.

R-squared Score (R^2): 72%

The R-squared score indicates how well the model captures the variability of the target variable. A score of 72% means that 72% of the total variance in resale price can be explained by the feature variables, suggesting that the model provides a reasonably good fit to the data.

Root Mean Squared Error (RMSE): $91,456.30

RMSE represents the square root of the average of the squared differences between the predicted and actual values. An RMSE of $91,456.30, which is larger than the MAE, indicates that some predictions are significantly farther from the actual values. This suggests that the dataset may have considerable variability, or improvements might be needed, such as exploring different model types to reduce prediction errors.

- **Model Comparison**

Additionally, I trained the model using Decision Tree Regression and Random Forest algorithms. The goal is to compare and evaluate which model performs best for predicting HDB resale prices.

The models are evaluated using Mean Absolute Error (MAE), R-squared (R^2), and Root Mean Squared Error (RMSE) metrics to determine which performs best on the same test set (refer to figures below).

```
In [39]:
# Using the DecisionTree Model

import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics

# Instantiate the DecisionTree model
decision_tree = DecisionTreeRegressor(random_state=42)

# Fit the model to your training data (X_train and Y_train)
decision_tree.fit(x_train,y_train)

# Make predictions on the test data
treePrediction = decision_tree.predict(x_test)

# Calculate the Mean Absolute Error, R-squared score and Root Mean Squared Error
mae = metrics.mean_absolute_error(y_test, treePrediction)
r2 = metrics.r2_score(y_test, treePrediction)
rmse =  np.sqrt(metrics.mean_squared_error(y_test, treePrediction))

# Output the results
print("Mean Absolute Error:", "$", round(mae, 2))
print("R-squared score:",round(r2*100), "%")
print("Root Mean Squared Error score:", "$", round(rmse, 2) )

Mean Absolute Error: $ 47925.77
R-squared score: 83 %
Root Mean Squared Error score: $ 70348.48
```

Click below to see code shown on the left.

hdb_dataset_results_all_metrics.ipynb

Note: Value may vary with each iteration

```
In [40]:
# Using the Random Forest Model

import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics

# Instantiate the Random Forest model
random_forest = RandomForestRegressor(n_estimators=100, random_state=42)

# Fit the model to your training data (X_train and Y_train)
random_forest.fit(x_train,y_train)

# Make predictions on the test data
y_Pred = random_forest.predict(x_test)

# Calculate the Mean Absolute Error, R-squared score and Root Mean Squared Error
mae = metrics.mean_absolute_error(y_test, y_Pred)
r2 = metrics.r2_score(y_test, y_Pred)
rmse =  np.sqrt(metrics.mean_squared_error(y_test, y_Pred))

# Output the results
print("Mean Absolute Error:", "$", round(mae, 2))
print("R-squared score:",round(r2*100), "%")
print("Root Mean Squared Error score:", "$", round(rmse, 2) )
```

```
C:\Users\rosla\AppData\Local\Temp\ipykernel_11284\1052648872.py:11: DataConversionWarning: A column-vector
a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  random_forest.fit(x_train,y_train)
Mean Absolute Error: $ 40532.22
R-squared score: 89 %
Root Mean Squared Error score: $ 57606.42
```

The results from the three iterations show a clear progression in model performance:

<u>Linear Regression Model</u>

The Mean Absolute Error (MAE) is $73,883.89, with an R-squared score of 72% and a Root Mean Squared Error (RMSE) of $91,456.30. This indicates that the model's predictions are relatively far from the actual values, and the R-squared score suggests that about 72% of the variance in the resale prices is explained by the model

.

<u>Decision Tree Regressor Model</u>

There's a noticeable improvement, with the MAE decreasing to $47,925.77 and the R-squared score rising to 83%. The RMSE also improves to $70,348.48. This suggests that the model is better at capturing the underlying patterns in the data, resulting in more accurate predictions.

<u>Random Forest Regressor Model</u>

The results continue to improve, with the MAE further reduced to $40,532.22 and the R-squared score increasing to 89%. The RMSE decreases to $57,606.42, indicating that the model has become significantly more accurate and is able to explain a larger portion of the variance in resale prices.

Overall, the consistent decline in MAE and RMSE, along with the increase in R-squared scores, demonstrates effective model enhancement across the tests, suggesting that the techniques or parameters being adjusted are yielding positive results.

5.5 Feature Importance and Interpretation

- **Feature Importance**

Based on the correlation analysis, ‘flat_type’ and ‘floor_area_sqm’ appear to be the most important feature for predicting resale prices due to its strong positive correlation.

The features ‘storey_range’ and ‘lease_remaining’ also contribute to predicting resale prices but to a lesser extent.

- **Implications for Model Building**

When building predictive models, priorities should be given to the ‘flat_type’ and ‘floor_area_sqm’ because to the strong correlation with the target variable, ‘resale_price’. Features with weak correlations that might exhibit less predictive power could be considered for exclusion.

5.6 Model Adjustments

Given the unexpectedly low correlation between ‘town’ and ‘resale_price’ shown in the heatmap, which contrasts with findings that towns closer to amenities tend to have higher resale prices, I will now attempt to remove all the outliers that were retained in the earlier analysis.

<u>Analysing Outliers by Flat Types</u>

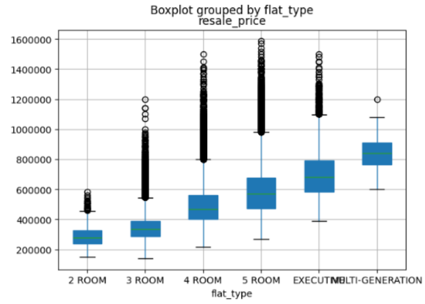
This analysis aims to identify the onset of outliers within the dataset categorized by ‘flat_type’. Once identified, the records containing these outliers will be removed. This adjustment is intended to evaluate whether the model’s performance can be enhanced by excluding these extreme values.

Click below to see code shown on the left.

hdb_dataset_adjusted_outliers_removed.ipynb

[30]:
```
import matplotlib.pyplot as plt
import matplotlib.ticker as mticker

# ploting the boxplot
boxplot_dict = hdb_df.boxplot(column='resale_price', by='flat_type', patch_artist=True, return_type='dict')

# Extract the first outlier for each flat type
outliers = {}
for flat_type, data in boxplot_dict.items():
    outliers[flat_type] = [flier.get_ydata()[0] for flier in data['fliers'] if len(flier.get_ydata()) > 0]

# Print the first outlier for each flat type
for flat_type, outlier_value in outliers.items():
    print(f"First outlier for {flat_type}: {outlier_value}")

# Remove scientific notation on the y-axis
plt.gca().yaxis.set_major_formatter(mticker.ScalarFormatter())
plt.gca().ticklabel_format(style='plain', axis='y')

plt.show()
```

```
First outlier for resale_price: [508000.0, 580000.0, 808000.0, 1108000.0, 1160000.0, 1200000.0]
```



[31]:
```
# Define a function to calculate the upper boundary
def calculate_upper_boundary(df, column):
    Q1 = df[column].quantile(0.25)  # First quartile (25th percentile)
    Q3 = df[column].quantile(0.75)  # Third quartile (75th percentile)
    IQR = Q3 - Q1                   # Interquartile Range
    upper_boundary = Q3 + 1.5 * IQR  # Upper boundary
    return upper_boundary

# List of flat types to calculate upper boundaries for
flat_types = hdb_df['flat_type'].unique()

# Create a copy of the original dataset to store the filtered data
filtered_hdb_df = pd.DataFrame()

# Loop through each flat type to remove outliers
for flat_type in flat_types:
    # Filter the DataFrame by flat type
    flat_data = hdb_df[hdb_df['flat_type'] == flat_type]

    # Calculate the upper boundary for resale_price
    upper_boundary = calculate_upper_boundary(flat_data, 'resale_price')

    # Print the upper boundary for the current flat type
    print(f"Upper boundary for {flat_type}: {upper_boundary:.2f}")

    # Filter out outliers above the upper boundary
    filtered_data = flat_data[flat_data['resale_price'] <= upper_boundary]

    # Append the filtered data (without outliers) to the new DataFrame
    filtered_hdb_df = pd.concat([filtered_hdb_df, filtered_data])

# Check the filtered dataset (without outliers)
filtered_hdb_df
```

```
Upper boundary for 3 ROOM: 546168.00
Upper boundary for 4 ROOM: 800000.00
Upper boundary for 5 ROOM: 980000.00
Upper boundary for EXECUTIVE: 1097500.00
Upper boundary for 2 ROOM: 455500.00
Upper boundary for MULTI-GENERATION: 1124000.00
```

[31]:

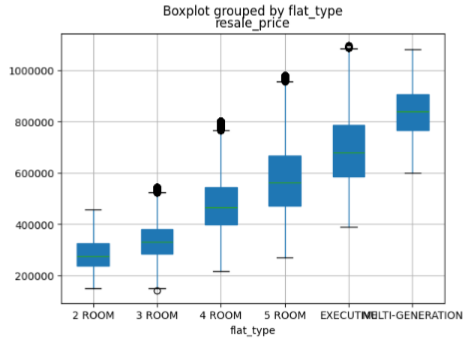
	town	flat_type	storey_range	floor_area_sqm	resale_price	lease_remaining
1	ANG MO KIO	3 ROOM	01 TO 03	67.0	250000.0	727
2	ANG MO KIO	3 ROOM	01 TO 03	67.0	262000.0	749
3	ANG MO KIO	3 ROOM	04 TO 06	68.0	265000.0	745
4	ANG MO KIO	3 ROOM	01 TO 03	67.0	265000.0	749
5	ANG MO KIO	3 ROOM	01 TO 03	68.0	275000.0	756
...	...	...	...	...	...	...
156335	YISHUN	MULTI-GENERATION	04 TO 06	159.0	935000.0	762
169151	YISHUN	MULTI-GENERATION	01 TO 03	179.0	975000.0	757
171961	BISHAN	MULTI-GENERATION	07 TO 09	134.0	980000.0	749
173897	YISHUN	MULTI-GENERATION	04 TO 06	164.0	998000.0	755
175619	TAMPINES	MULTI-GENERATION	10 TO 12	166.0	1060000.0	750

173851 rows × 6 columns

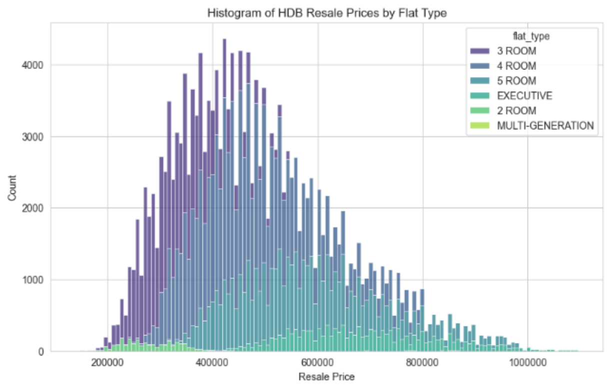
The boxplot figures above illustrate the outliers for each flat type, marking the initial data points identified for removal. The subsequent figure displays the filtered dataset, with all records containing outliers excluded. As a result, the dataset now comprises 173,851 rows, down from the original 180,286 rows. The figures below show the boxplot and the reconstructed histogram with the removed outliers.

```
[32]: # import matplotlib.ticker as mticker

# ploting the boxplot
boxplot_dict = filtered_hdb_df.boxplot(column='resale_price', by='flat_type', patch_artist=True, return_type='dict')

# Extract the first outlier for each flat type
outliers = {}
for flat_type, data in boxplot_dict.items():
    outliers[flat_type] = [flier.get_ydata()[0] for flier in data['fliers'] if len(flier.get_ydata()) > 0]

# Print the first outlier for each flat type
for flat_type, outlier_value in outliers.items():
    print(f"First outlier for {flat_type}: {outlier_value}")

# Remove scientific notation on the y-axis
plt.gca().yaxis.set_major_formatter(mticker.ScalarFormatter())
plt.gca().ticklabel_format(style='plain', axis='y')

plt.show()
```

```
First outlier for resale_price: [140000.0, 790000.0, 980000.0, 1088000.0]
```




```
[33]: # Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Create a histogram
plt.figure(figsize=(10, 6))
sns.histplot(data=filtered_hdb_df, x='resale_price', hue='flat_type', multiple='stack', palette='viridis')

# Remove scientific notation from the x-axis
plt.ticklabel_format(style='plain', axis='x')

# Add titles and labels
plt.title('Histogram of HDB Resale Prices by Flat Type')
plt.xlabel('Resale Price')
plt.ylabel('Count')

# Show the plot
plt.show()
```



Click below to see code shown on the left.

hdb_dataset_adjucted_boxplot_histrogram.ipynb

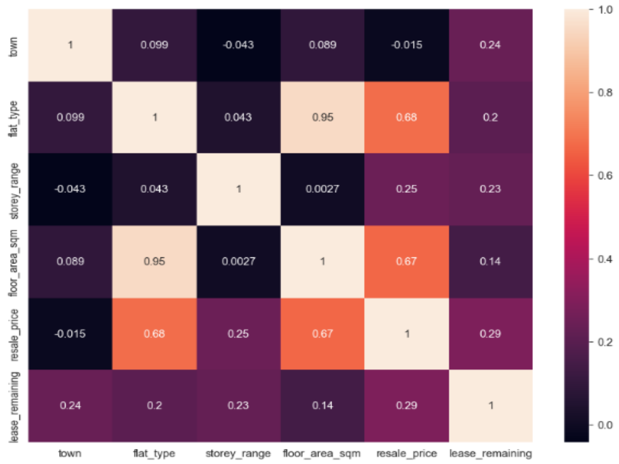
A quick comparison of the filtered versus unfiltered data (see figures below) reveals that a significant amount of information has been removed in the filtered dataset. Notably, floor levels from the 43rd storey onwards are missing. This omission does not accurately represent the real situation.

```
[35]: filtered_hdb_df['storey_range'].value_counts()

[35]: storey_range
      04 TO 06    41013
      07 TO 09    37288
      10 TO 12    32956
      01 TO 03    31418
      13 TO 15    16739
      16 TO 18     7560
      19 TO 21     2929
      22 TO 24     1986
      25 TO 27     1024
      28 TO 30      497
      31 TO 33      175
      34 TO 36      140
      37 TO 39      102
      40 TO 42       24
      Name: count, dtype: int64

[36]: filtered_hdb_df['flat_type'].value_counts()

[36]: flat_type
      4 ROOM              73044
      5 ROOM              44000
      3 ROOM              41098
      EXECUTIVE           13208
      2 ROOM               2424
      MULTI-GENERATION       77
      Name: count, dtype: int64
```

```
[27]: hdb_df['storey_range'].value_counts()

[27]: storey_range
      04 TO 06    41468
      07 TO 09    37832
      10 TO 12    33602
      01 TO 03    31766
      13 TO 15    17311
      16 TO 18     8126
      19 TO 21     3497
      22 TO 24     2487
      25 TO 27     1505
      28 TO 30      979
      31 TO 33      524
      34 TO 36      470
      37 TO 39      406
      40 TO 42      196
      43 TO 45       57
      46 TO 48       45
      49 TO 51       15
      Name: count, dtype: int64

[28]: hdb_df['flat_type'].value_counts()

[28]: flat_type
      4 ROOM              76457
      5 ROOM              45059
      3 ROOM              42900
      EXECUTIVE           13347
      2 ROOM               2445
      MULTI-GENERATION       78
      Name: count, dtype: int64
```

The figure below illustrates the correlation after adjusting for outliers. The correlation indices between flat_type, floor_area_sqm, storey_range, and lease_remaining in relation to resale_price have all increased slightly. However, the correlation between town and resale_price has weakened. While this aligns with expectations, removing the town feature is not realistic, as this feature should impact resale prices.

```
[41]: # plot the correlation matrix as a heatmap
      plt.figure(figsize = (10,8))
      sns.heatmap(filtered_hdb_df.corr(), annot = True);
```



Click below to see code shown on the left.

hdb_dataset_adjusted_corr_matrix.ipynb

The categorical data in the adjusted dataset was subsequently converted to numerical data types, and the model retrained.

Here are the results (refer to figures below) of the MAE, R^2, and RMSE scores after iterating with Linear Regression, Decision Tree Regression, and Random Forest models:

- **Linear Regression**:
  MAE: $ 70448.52
  R^2: 70%
  RMSE: $ 85477.40

- **Decision Tree Regression**:
  MAE: $ 46565.89
  R^2: 81%
  RMSE: $ 67573.67

- **Random Forest**:
  MAE: $ 39570.58
  R^2: 87%
  RMSE: $ 55667.86

```
In [46]:
# Training the Model

import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn import metrics

# Instantiate the linear regression model
lin_reg = LinearRegression()

# Fit the model to your training data (X_train and Y_train)
lin_reg.fit(x_train, y_train)

# Make predictions on the test data
linPrediction = lin_reg.predict(x_test)

# Calculate the Mean Absolute Error, R-squared score and Root Mean Squared Error
mae = metrics.mean_absolute_error(y_test, linPrediction)
r2 = metrics.r2_score(y_test, linPrediction)
rmse =  np.sqrt(metrics.mean_squared_error(y_test, linPrediction))

# Output the results
print("Mean Absolute Error:", "$", round(mae, 2))
print("R-squared score:",round(r2*100), "%")
print("Root Mean Squared Error score:", "$", round(rmse, 2) )

Mean Absolute Error: $ 70448.52
R-squared score: 70 %
Root Mean Squared Error score: $ 85477.4
```

Click below to see code shown on the left.

hdb_dataset_adjusted_results_all_metrics.ipynb

Note: Value may vary with each iteration

```
In [47]:
# Using the DecisionTree Model

import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics

# Instantiate the DecisionTree model
decision_tree = DecisionTreeRegressor(random_state=42)

# Fit the model to your training data (X_train and Y_train)
decision_tree.fit(x_train,y_train)

# Make predictions on the test data
treePrediction = decision_tree.predict(x_test)

# Calculate the Mean Absolute Error, R-squared score and Root Mean Squared Error
mae = metrics.mean_absolute_error(y_test, treePrediction)
r2 = metrics.r2_score(y_test, treePrediction)
rmse =  np.sqrt(metrics.mean_squared_error(y_test, treePrediction))

# Output the results
print("Mean Absolute Error:", "$", round(mae, 2))
print("R-squared score:",round(r2*100), "%")
print("Root Mean Squared Error score:", "$", round(rmse, 2) )

Mean Absolute Error: $ 46565.89
R-squared score: 81 %
Root Mean Squared Error score: $ 67571.67
```

```
In [48]:
# Using the Random Forest Model

import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics

# Instantiate the Random Forest model
random_forest = RandomForestRegressor(n_estimators=100, random_state=42)

# Fit the model to your training data (X_train and Y_train)
random_forest.fit(x_train,y_train)

# Make predictions on the test data
y_Pred = random_forest.predict(x_test)

# Calculate the Mean Absolute Error, R-squared score and Root Mean Squared Error
mae = metrics.mean_absolute_error(y_test, y_Pred)
r2 = metrics.r2_score(y_test, y_Pred)
rmse =  np.sqrt(metrics.mean_squared_error(y_test, y_Pred))

# Output the results
print("Mean Absolute Error:", "$", round(mae, 2))
print("R-squared score:", round(r2*100), "%")
print("Root Mean Squared Error score:", "$", round(rmse, 2) )

C:\Users\rosla\AppData\Local\Temp\ipykernel_21948\2503519406.py:11: DataConversionWarning: A column-v
en a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  random_forest.fit(x_train,y_train)
Mean Absolute Error: $ 39570.58
R-squared score: 87 %
Root Mean Squared Error score: $ 55667.86
```

When comparing the results before and after removing outliers (refer to figures below), improvements in the MAE and RMSE scores are evident. However, the R^2 scores have decreased slightly by 1 to 2%. The decision to retain or remove outliers is debatable, but I recommend keeping the outliers to ensure the results remain realistic and to capture the real situation in the market.

Before Adjustment	After Adjustment
▪ **Linear Regression**: MAE: $ 73883.89 R^2: 72% RMSE: $ 91456.30 ▪ **Decision Tree Regression**: MAE: $ 47925.77 R^2: 83% RMSE: $ 70348.48 ▪ **Random Forest**: MAE: $ 40532.22 R^2: 89% RMSE: $ 57606.42	▪ **Linear Regression**: MAE: $ 70448.52 R^2: 70% RMSE: $ 85477.40 ▪ **Decision Tree Regression**: MAE: $ 46565.89 R^2: 81% RMSE: $ 67573.67 ▪ **Random Forest**: MAE: $ 39570.58 R^2: 87% RMSE: $ 55667.86

6. Stakeholders

- **Homebuyers and Sellers**

  Individuals looking to buy or sell HDB flats can use the model to estimate fair prices.

- **Real Estate Agents**

  Professionals can leverage the insights to provide better advice to their clients.

- **Policymakers**

  Government bodies can use the analysis to understand market trends and implement effective housing policies.

- **Financial Institutions**

  Banks and lenders can assess the value of properties more accurately for mortgage purposes.

- **Researchers and Academics**

  Scholars can use the findings for further research in real estate economics and urban planning.

7. Benefits

- **Transparency**

  Enhances transparency in the HDB resale market by providing data-driven price estimates.

- **Informed Decisions**

  Empowers stakeholders with accurate information to make better decisions.

- **Market Stability**

  Helps in stabilizing the market by reducing price speculation and volatility.

- **Policy Formulation**

  Assists policymakers in crafting data-driven housing policies to address market needs.

8. Use Cases

The predictive model for HDB resale prices can be adapted for various applications in the real estate and urban planning sectors. Here are some potential applications:

- **Rental Price Prediction**

  Predict rental prices for HDB flats based on similar features used for resale price prediction. This may help tenants and landlords set fair rental prices and understand market trends.

- **Property Valuation for Insurance**

  Provide accurate property valuations for insurance purposes. It ensures that properties are adequately insured, reflecting their true market value.

- **Urban Planning and Development**

  Assist urban planners in identifying areas with high growth potential and planning infrastructure development accordingly. This will support strategic development and efficient allocation of resources.

- **Mortgage Risk Assessment**

  Help financial institutions assess the risk associated with mortgage lending by predicting future property values. It reduces the risk of loan defaults and improves the accuracy of loan-to-value ratios.

- **Investment Analysis**

  Aid real estate investors in identifying lucrative investment opportunities by predicting future price trends. It enhances investment decision-making and portfolio management.

- **Policy Formulation**

  Provide data-driven insights to policymakers for crafting housing policies and regulations. It ensures policies are aligned with market realities and address housing affordability and availability.

- **Community and Social Services Planning**

  Assist in planning community services and amenities based on predicted population growth and housing demand. It may help improve the quality of life by ensuring adequate provision of services and facilities.

- **Environmental Impact Assessment**

  Evaluate the environmental impact of new housing developments by predicting changes in property values and population density. It helps support sustainable development and environmental conservation efforts.

- **Real Estate Market Analysis**

  Provide comprehensive market analysis reports for real estate agencies and developers. It helps enhance market understanding and strategic planning for new projects.

- **Customized Real Estate Solutions**

  Develop personalized real estate solutions for clients based on their preferences and predicted market trends. It improves customer satisfaction and service quality.

9. Ethics and Governance

Ensuring ethical practices and robust governance is essential when creating and implementing predictive models, particularly in sensitive domains such as housing.

9.1 Ethical Considerations

- **Bias and Fairness**

  Historical data may contain biases that can be perpetuated by the model, leading to unfair predictions. Implementing fairness-aware algorithms and regularly audit the model for biases would ensure diverse and representative data to mitigate bias.

- **Transparency and Explainability**

  Complex models can be difficult to interpret, making it hard for stakeholders to understand how predictions are made. Using interpretable models or techniques to explain model predictions would provide clear documentation and visualizations to enhance transparency.

- **Privacy and Data Protection**

  Handling personal and sensitive data can lead to privacy concerns. Anonymizing data and comply with data protection regulations like the Personal Data Protection Act (PDPA) in Singapore would ensure secure data storage and access controls.

- **Informed Consent**

  Using data without the knowledge or consent of individuals can be unethical. Obtaining informed consent from data subjects and be transparent about how their data could be used.

- **Impact on Stakeholders**

  Predictions can influence market behaviour and individual decisions, potentially leading to unintended consequences. Conducting impact assessments to understand the potential effects on different stakeholders and implement measures could mitigate negative impacts.

9.2 Governance

- **Regulatory Compliance**

  Ensure the project complies with relevant laws and regulations, such as housing policies and data protection laws. This can be done with regular reviews of legal requirements and consultation with legal experts to ensure compliance.

- **Ethical Guidelines**

  Develop and adhere to a set of ethical guidelines for the project. This can be done by establishing a code of ethics that outlines principles such as fairness, transparency, and accountability. Also to ensure all team members are aware of and follow these guidelines.

- **Stakeholder Engagement**

  Engage with stakeholders throughout the project to understand their needs and concerns. This can be done by conducting regular consultations and feedback sessions with stakeholders, including homebuyers, sellers, policymakers, and community representatives.

- **Accountability and Oversight**

  Establish mechanisms for accountability and oversight to ensure ethical conduct. This can be done by forming an ethics review board or committee to oversee the project and address any ethical issues that arise. Implement regular audits and reviews of the model and its outcomes.

- **Continuous Monitoring and Evaluation**

  Continuously monitor and evaluate the model to ensure it remains fair and effective. Implement ongoing monitoring processes to track the model's performance and impact. Adjust as needed to address any issues.

Addressing these ethical considerations and governance aspects would ensure that the project is conducted responsibly and ethically, benefiting all stakeholders involved.

10. Conclusion

The HDB Resale Prediction Project has successfully demonstrated the potential of machine learning (ML) and artificial intelligence (AI) in accurately forecasting HDB resale prices. By leveraging a diverse set of features, including location, flat type, floor area, and transaction history, the model achieved a relatively high degree of predictive accuracy.

10.1 Key Findings

- **Predictive Performance and Accuracy**

  The model's performance metrics indicate a robust ability to predict resale prices, with a mean absolute error (MAE) of $ 40828.95, R-squared (R^2) of 89% and a root mean square error (RMSE) of $ 57943.24.

- **Feature Importance**

  Flat type and floor area emerged as the most significant predictors, highlighting the importance of flat type and size in determining resale values.

- **Biases**

  Although not examined in this project, the dataset may contain historical biases, such as location preferences or economic conditions that have evolved over time.

  Also, some machine learning algorithms might inherently favour certain features over others, leading to biased predictions. This is shown in the model iteration exercise earlier.

10.2 Implications

- **Policy Making**

  The insights gained from this project can inform policymakers in designing housing policies that are data-driven and equitable.

- **Market Transparency**

  Enhanced price prediction models can increase market transparency, helping buyers and sellers make more informed decisions.

- **Ethical Considerations**

  It's essential to address ethical concerns such as data privacy, bias, and transparency in applications.

  Data privacy is a critical ethical issue because it involves safeguarding individuals' personal information from unauthorized access and misuse. Protecting data privacy helps maintain trust, respects individuals' rights, and prevents potential harm from data breaches or misuse.

  Implementing strategies like anonymizing data by removing identifiable personal information, using aggregated data instead of detailed individual data, and conducting regular audits can significantly enhance the privacy, fairness, and transparency of the dataset while still allowing for valuable insights and analysis.

10.3 Future Directions

- **Model Improvement**

  Future work could focus on incorporating additional features, such as economic indicators and environmental factors, to further enhance predictive accuracy.

- **Ethical Frameworks**

  Developing and implementing robust ethical frameworks to govern the use of AI in real estate is essential to ensure fairness and accountability.

- **Stakeholder Engagement**

  Engaging with a broader range of stakeholders, including residents, policymakers, and ethicists, will be crucial in refining the model and its applications.

In conclusion, the HDB Resale Prediction Project has successfully demonstrated the potential of machine learning (ML) and artificial intelligence (AI) in accurately forecasting HDB resale prices. By leveraging a diverse set of features, including location, flat type, floor area, and transaction history, the model achieved a relatively high degree of predictive accuracy.

To effectively address ethical and governance challenges, it is crucial to maintain ongoing efforts. By continuously improving both the model and its ethical framework, we can ensure that AI-driven insights have a positive impact on the housing market and the user community.