SC1015 mini-project

HDB Resale Prices

Ryan Tang, Lin Rui Jun,
Sean Tongvanikkul

Our Project Outline

01

Problem Statement

02

Data Cleaning and Preparation

03

Exploratory Data Analysis

04

Machine Learning

05

Conclusion

Motivation

01 Relation to our field of study

02 Real-world Relevance

How can we predict resale prices of HDB flats?

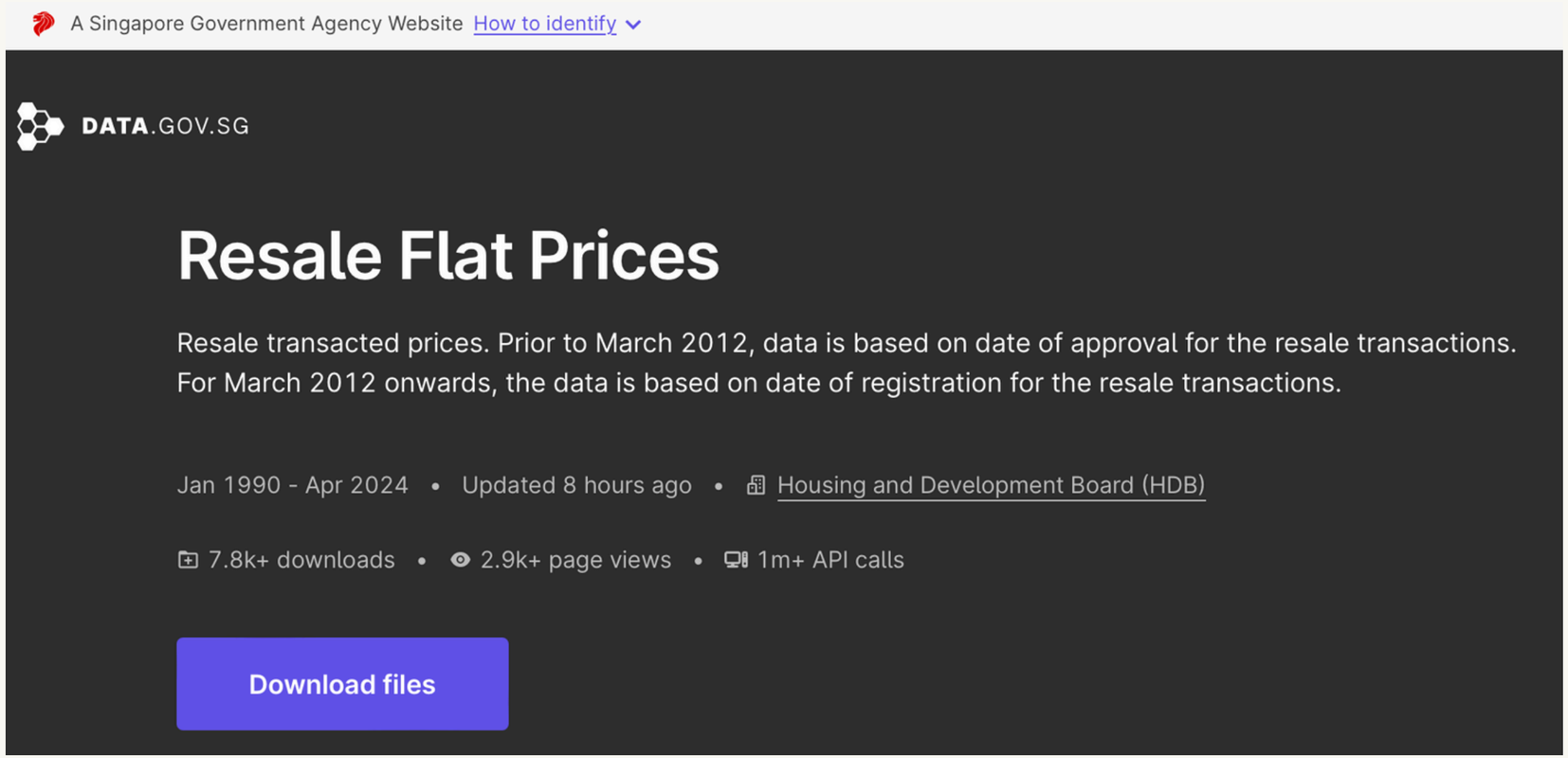
26,735

HDB resale applicants in 2023

from hub.gov.sg

Our Dataset

where did we get our data from?



Why?

Factors for choosing our source

Reliability

The source is from a governemnet agency, which ensures the data is accurate and usable.

Completeness

Good data sources should have minimal missing values and gaps. Incomplete data can skew results and limit the effectiveness of your analysis.

Timeliness

The data is up-to-date and relevant to the time period of your study. This is particularly important for dynamic fields such as economics, technology, and social sciences, where patterns can change significantly over time.

Data Cleaning & Preparation

01

Convert variables to workable format

02

Add coordinate data for addresses

03

Dropping NaNs, outliers and irrelevant columns

04

Creating new features

```python
df_resale["month"] = pd.to_datetime(df_resale["month"])
df_resale["lease_commence_date"] = pd.to_datetime(df_resale["lease_commence_date"])
# Define a function to convert the string into total number of months
def convert_to_years(s):
    try:
        # Split the string into years and months
        parts = s.split(' years ')
        if len(parts) == 2:  # Case: Years and months are both specified
            years_str, months_str = parts
            months_parts = months_str.split()
            # Extract the numeric part of months and convert it to an integer
            months = int(months_parts[0]) if months_parts else 0
            # Calculate the total number of years including the fraction of months
            decimal_years = int(years_str) + months / 12
            return round(decimal_years,2)
        elif len(parts) == 1:  # Case: Only years are specified
            years_parts = parts[0].split()
            if years_parts:
                return int(years_parts[0])
            else:
                return None
        else:
            # If the string format is unexpected, log a warning and return None
            print("Warning: Unexpected input format:", s)
            return None
    except ValueError as e:
        # Handle the case where the string doesn't have the expected format
        # Log the error for debugging purposes
        print("Error:", e)
        # Return None or another appropriate default value
        return None

# Specify the name of the column you want to convert
column_to_convert = 'remaining_lease'

# Apply the function to the column to convert it into numeric data
df_resale['remainginlease'] = df_resale[column_to_convert].apply(convert_to_years)

#create new column 'address' which combines blk and st
blk = 'block'
st = 'street_name'
df_resale['address'] = df_resale[blk] + ' ' + df_resale[st]
```

01 Conversion of values

We decided to convert the date and remaining lease values for easier maniupulation later on

07 TO 09	8
07 TO 09	8
07 TO 09	8
10 TO 12	11
01 TO 03	2
10 TO 12	11

01 Conversion of values

Storey range from categorical to numerical

2023-09-01 00:0...	81
2023-09-01 00:0...	81
2023-09-01 00:0...	81
2023-08-01 00:0...	80
2023-08-01 00:0...	80
2023-08-01 00:0...	80

01 Conversion of values

Converted data and time into a numerical variable starting with 2017 Jan = 1, 2018 Jan = 13 etc

02 Adding Coordinate Data

```
df_resale = pd.merge(df_resale, df_coordinate[['address', 'latitude', 'longitude']],
                     how='left', left_on='address', right_on='address')
```

Add Coordinate data

```
df_resale.head()
```

Visualize

remaining_lease o...	resale_price float64	remaininglease flo...	address object	latitude float64	longitude float64
61 years 04 mont...	232000	61.33	406 ANG MO KIO...	1.362004539	103.8538799
60 years 07 mont...	250000	60.58	108 ANG MO KIO ...	1.370966352	103.8382019
62 years 05 mont...	262000	62.42	602 ANG MO KIO...	1.38070883	103.8353682
62 years 01 month	265000	62.08	465 ANG MO KIO...	1.366201041	103.857201
62 years 05 mont...	265000	62.42	601 ANG MO KIO ...	1.381041355	103.8351317

```
1  total_na = df_resale.isna().sum().sum()
2  print(f"Total number of NA values in the DataFrame: {total_na}")
```

```
Total number of NA values in the DataFrame: 5880
```

```
1  filtered_df = df_resale.dropna(how = 'any')
2  filtered_df
```

03 Null Values

We first identified the number of null values present in the data set before dropping rows of data which had these null values as it might cause issues in our machine learning algorithms later on

03 Irrelevant Data

After identifying which columns gave us information that was not beneficial, we decided to drop those columns to declutter the data set

```
filtered_df.drop(columns = "street_name", inplace = True) ##maybe keep?
filtered_df.drop(columns = "block", inplace = True)
filtered_df.drop(columns = "remaining_lease", inplace = True)


filtered_df
```

03 Outliers

We decided to not remove any outliers as we wanted to understand the full range of resale prices in the housing market, removing outliers might remove any uniquely positioned flats or luxury units in prime locations.

04 New Features

Months from Base Period (Jan 2017)

Distance from Centre (km)

Distance to Nearest MRT (m)

Distance to Nearest Mall (m)

```python
from scipy.spatial import cKDTree

# Function to convert latitude and longitude to approximate meters
def latlon_to_meters(lat, lon):
    lat_m = lat * 110574  # Approximate length of a degree of latitude in meters
    lon_m = lon * 111320 * np.cos(np.radians(lat))  # Approximate length of a degree of longit
    return lat_m, lon_m

# Load data

mrt_stations = pd.read_csv('MRT Stations.csv')
mall = pd.read_csv('/work/shopping_mall_coordinates.csv')

# Prepare coordinates for KD-Tree in meter approximation
filtered_coords = np.array([latlon_to_meters(lat, lon) for lat, lon in zip(filtered_df['latitu
mrt_coords = np.array([latlon_to_meters(lat, lon) for lat, lon in zip(mrt_stations['Latitude']
mall_coords = np.array([latlon_to_meters(lat, lon) for lat, lon in zip(mall['LATITUDE'], mall[

# Create KD-Tree for MRT stations
mrt_tree = cKDTree(mrt_coords)
mall_tree = cKDTree(mall_coords)

# Query KD-Tree for nearest MRT station distance for each property
distances_mrt, _ = mrt_tree.query(filtered_coords, k=1)  # distances are in meters
distances_mall, _ = mall_tree.query(filtered_coords, k=1)  # distances are in meters


# Assign calculated distances to the DataFrame
filtered_df['distance_to_nearest_mrt'] = distances_mrt
filtered_df['distance_to_nearest_mall'] = distances_mall
```

Exploratory Data Analysis

A Quick Overview

```
1  filtered_df = pd.read_csv('filtered_df.csv')
2  filtered_df["month"] = pd.to_datetime(filtered_df["month"])
3  filtered_df.head()
```

Visualize

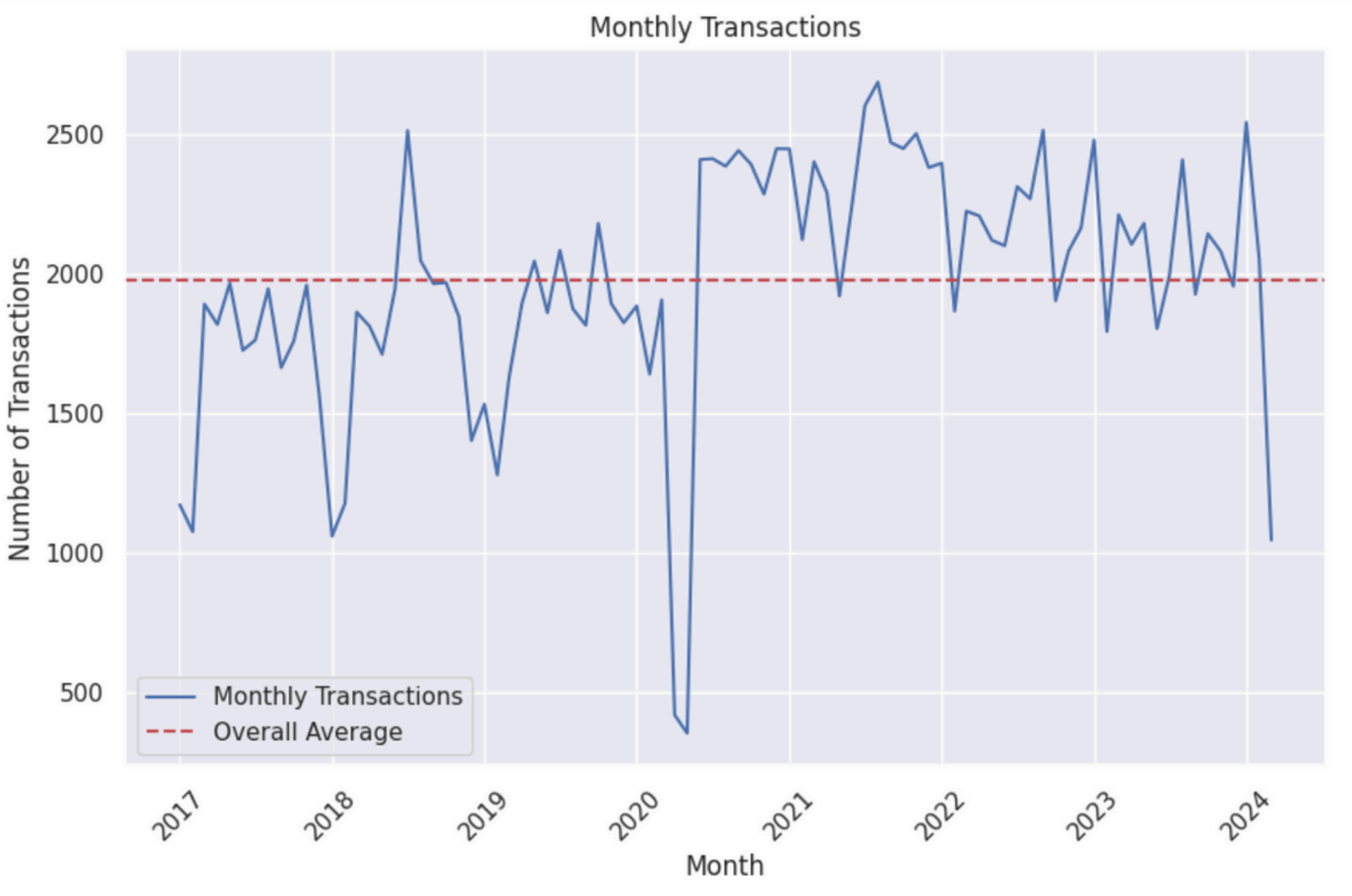
	month datetime64...	town object	flat_type object	storey_range object	floor_area_sqm flo...	flat_model object	lease_commence_...	r
0	2017-01-01 00:0...	ANG MO KIO	2 ROOM	10 TO 12	44	Improved	1970-01-01 00:00...	
1	2017-01-01 00:0...	ANG MO KIO	3 ROOM	01 TO 03	67	New Generation	1970-01-01 00:00...	
2	2017-01-01 00:0...	ANG MO KIO	3 ROOM	01 TO 03	67	New Generation	1970-01-01 00:00...	
3	2017-01-01 00:0...	ANG MO KIO	3 ROOM	04 TO 06	68	New Generation	1970-01-01 00:00...	
4	2017-01-01 00:0...	ANG MO KIO	3 ROOM	01 TO 03	67	New Generation	1970-01-01 00:00...	

Check the variables (and their types) in the dataset using the dtypes attribute.

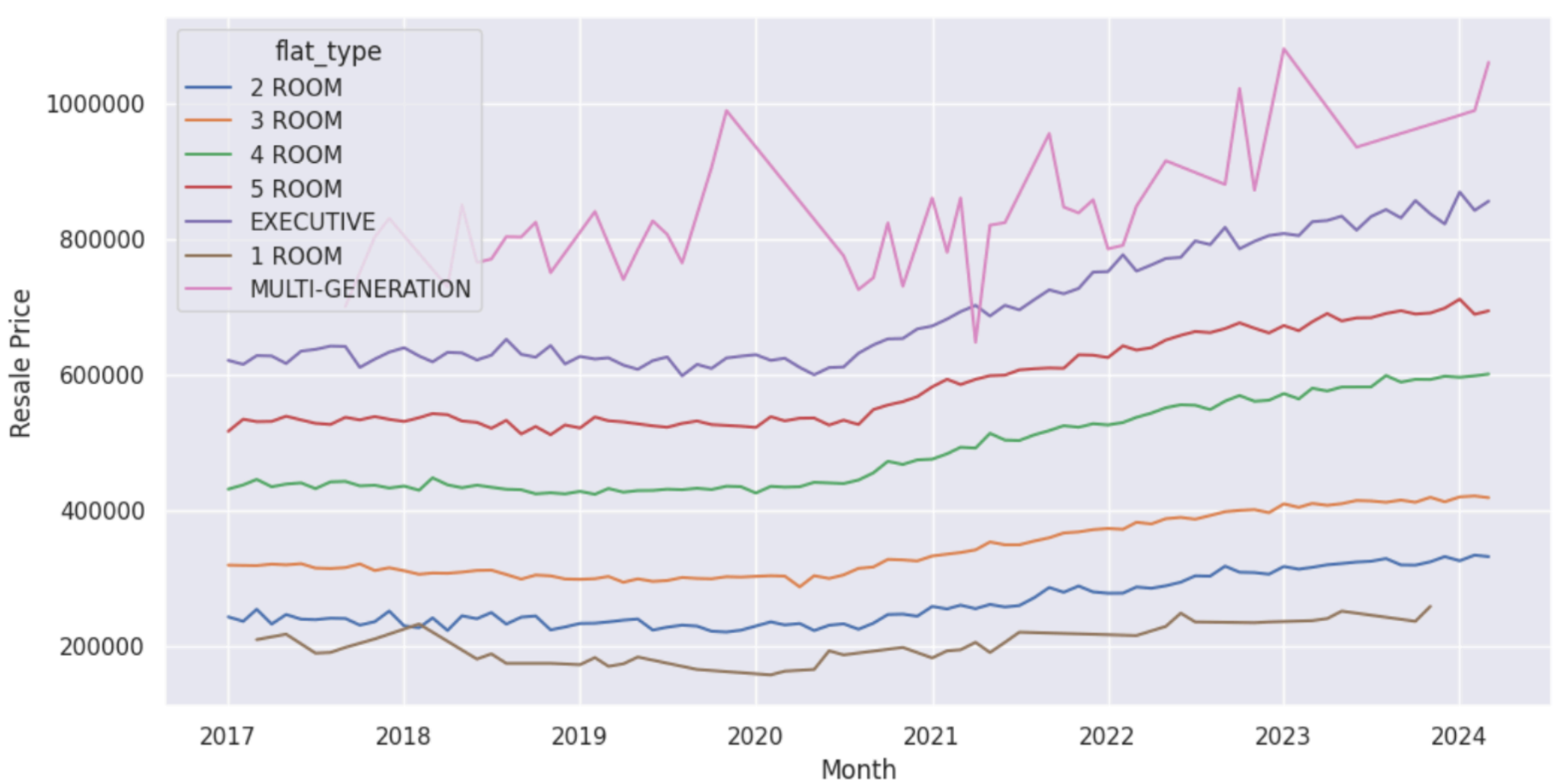
```
1  filtered_df.info()
2
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 172055 entries, 0 to 172054
Data columns (total 18 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   month                      172055 non-null  datetime64[ns]
 1   town                       172055 non-null  object
 2   flat_type                  172055 non-null  object
 3   storey_range               172055 non-null  object
 4   floor_area_sqm             172055 non-null  float64
 5   flat_model                 172055 non-null  object
 6   lease_commence_date        172055 non-null  object
 7   resale_price               172055 non-null  float64
 8   remaininglease             172055 non-null  float64
 9   address                    172055 non-null  object
 10  latitude                   172055 non-null  float64
 11  longitude                  172055 non-null  float64
 12  storey_midpoint            172055 non-null  float64
 13  numerical_month            172055 non-null  int64
 14  Distance_from_Centre (km)  172055 non-null  float64
 15  geometry                   172055 non-null  object
 16  distance_to_nearest_mrt    172055 non-null  float64
 17  distance_to_nearest_mall   172055 non-null  float64
dtypes: datetime64[ns](1), float64(9), int64(1), object(7)
```

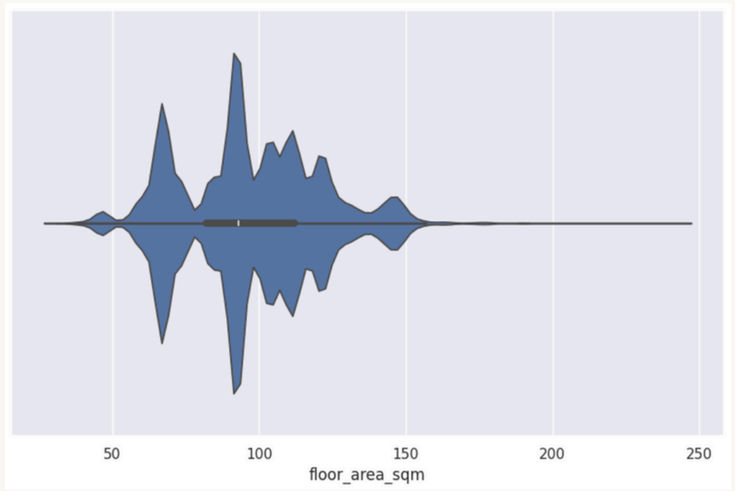
HDB Transactions over time

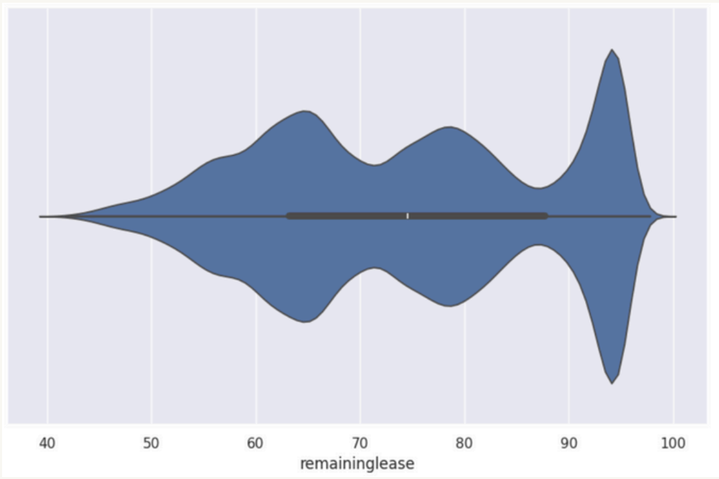


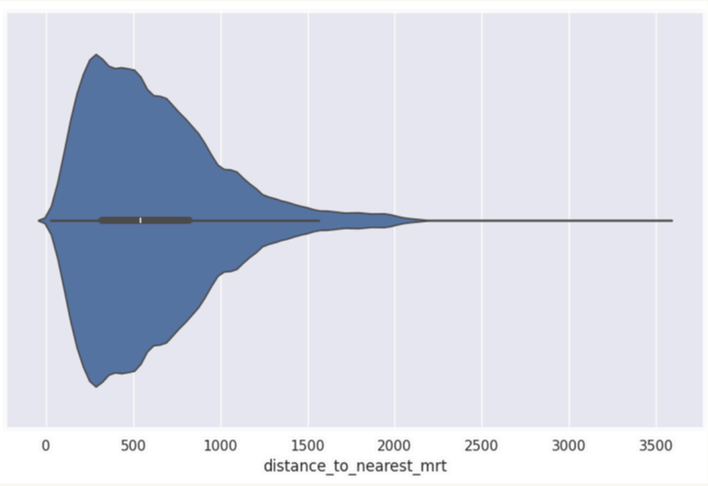
Transaction per room type over time

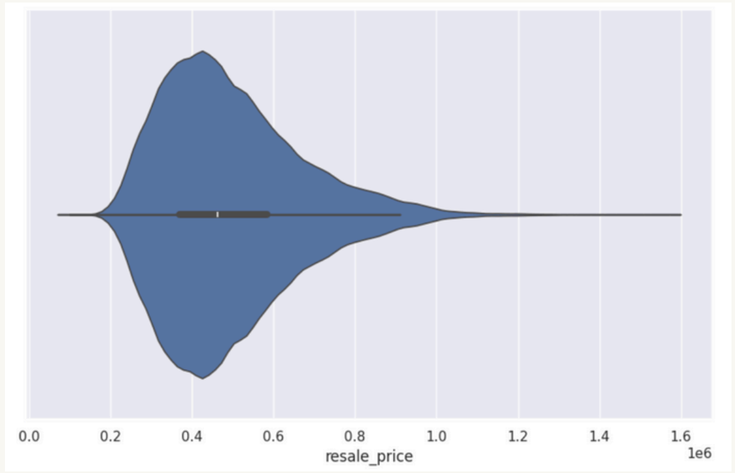


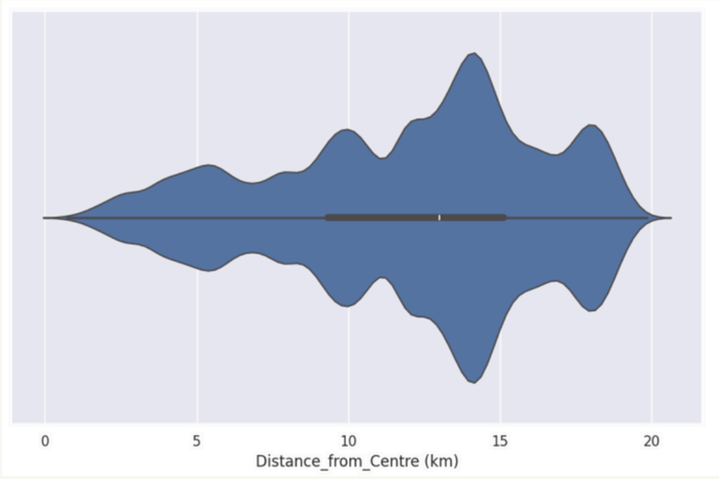
Distribution Of Data

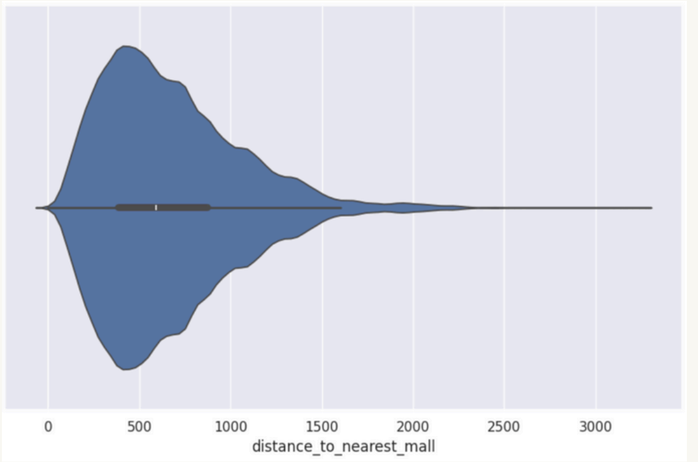




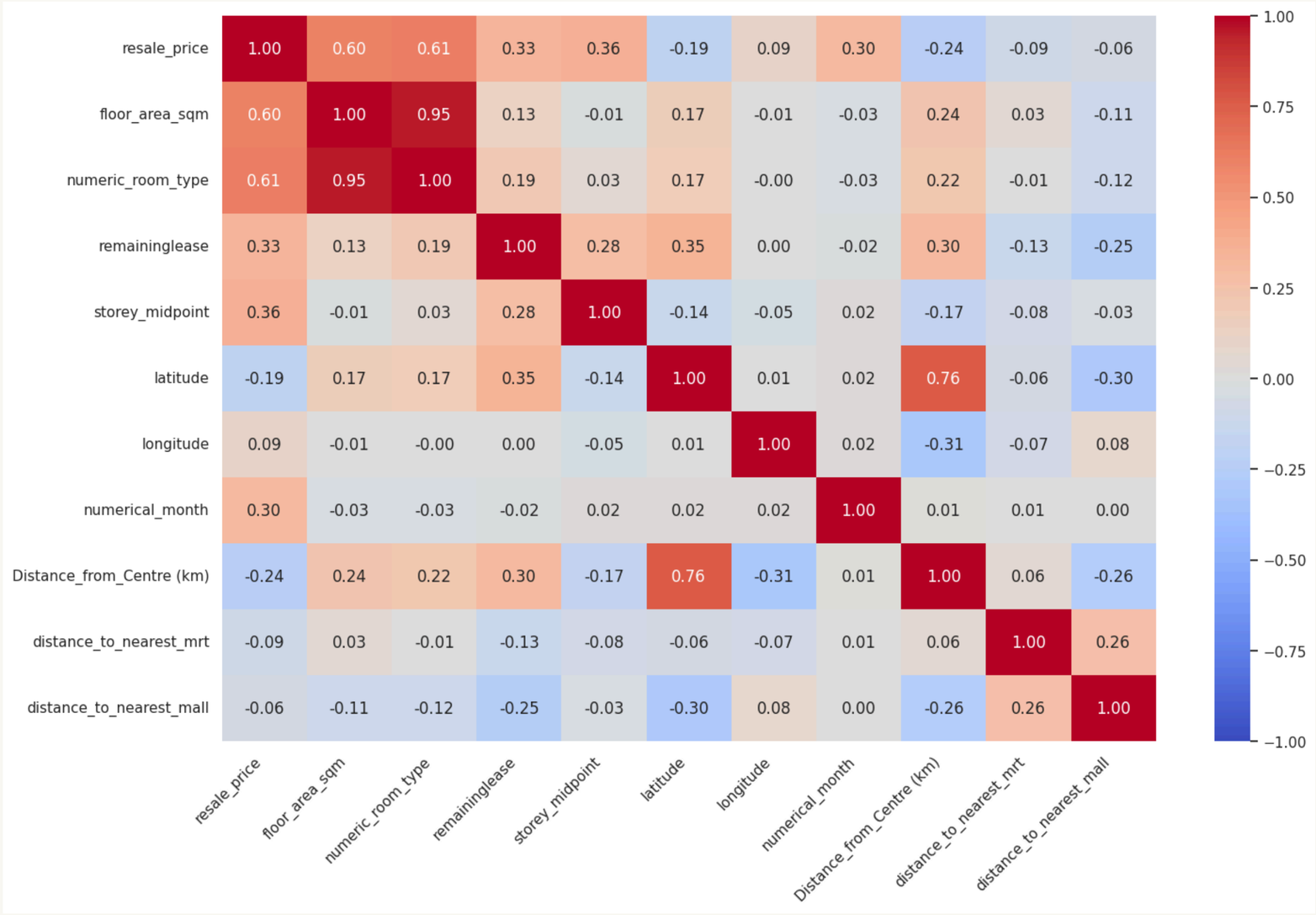




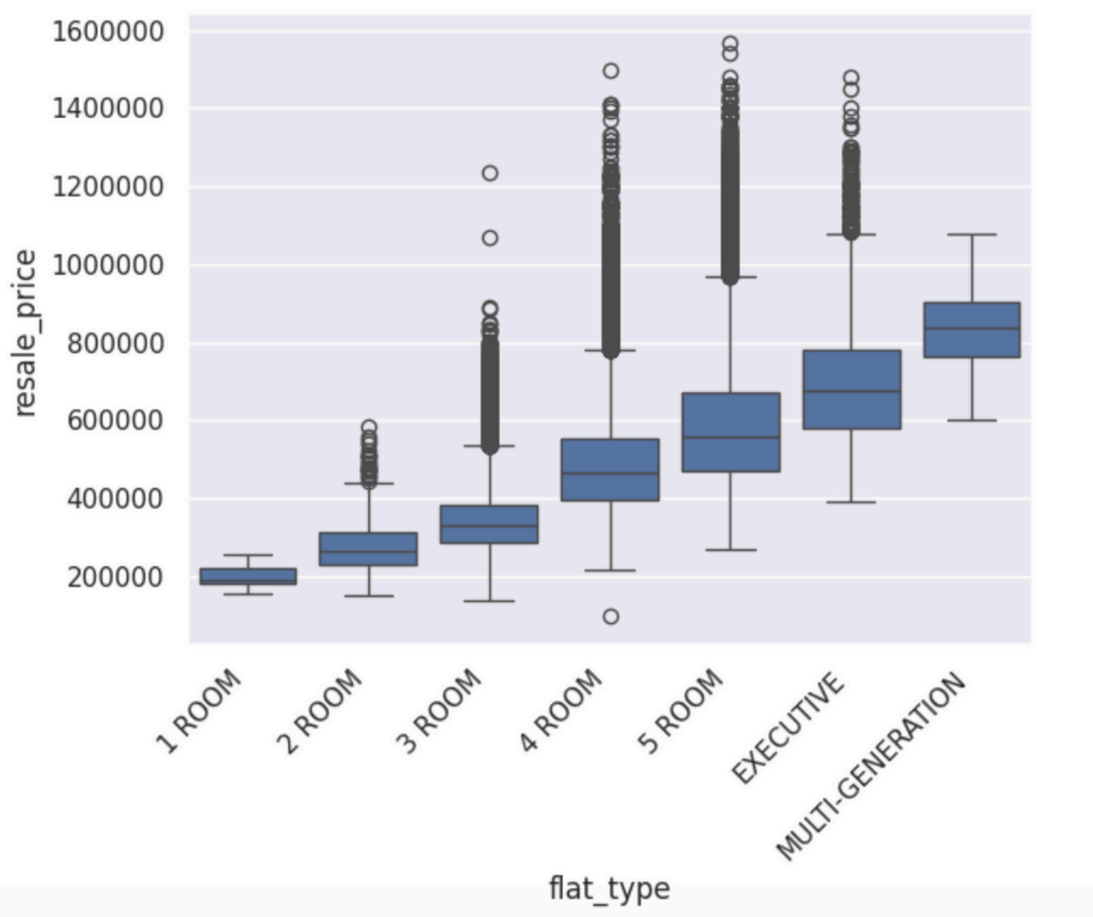




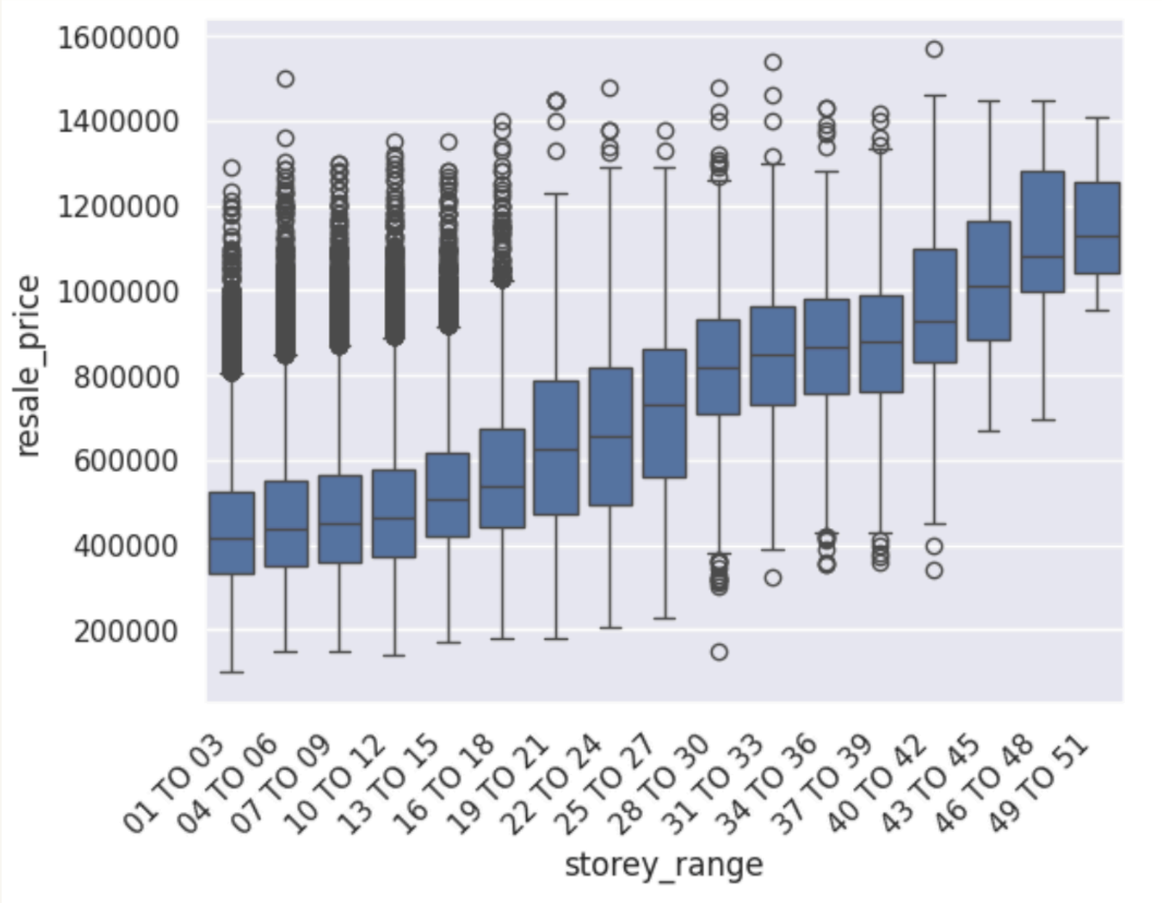
Correlation



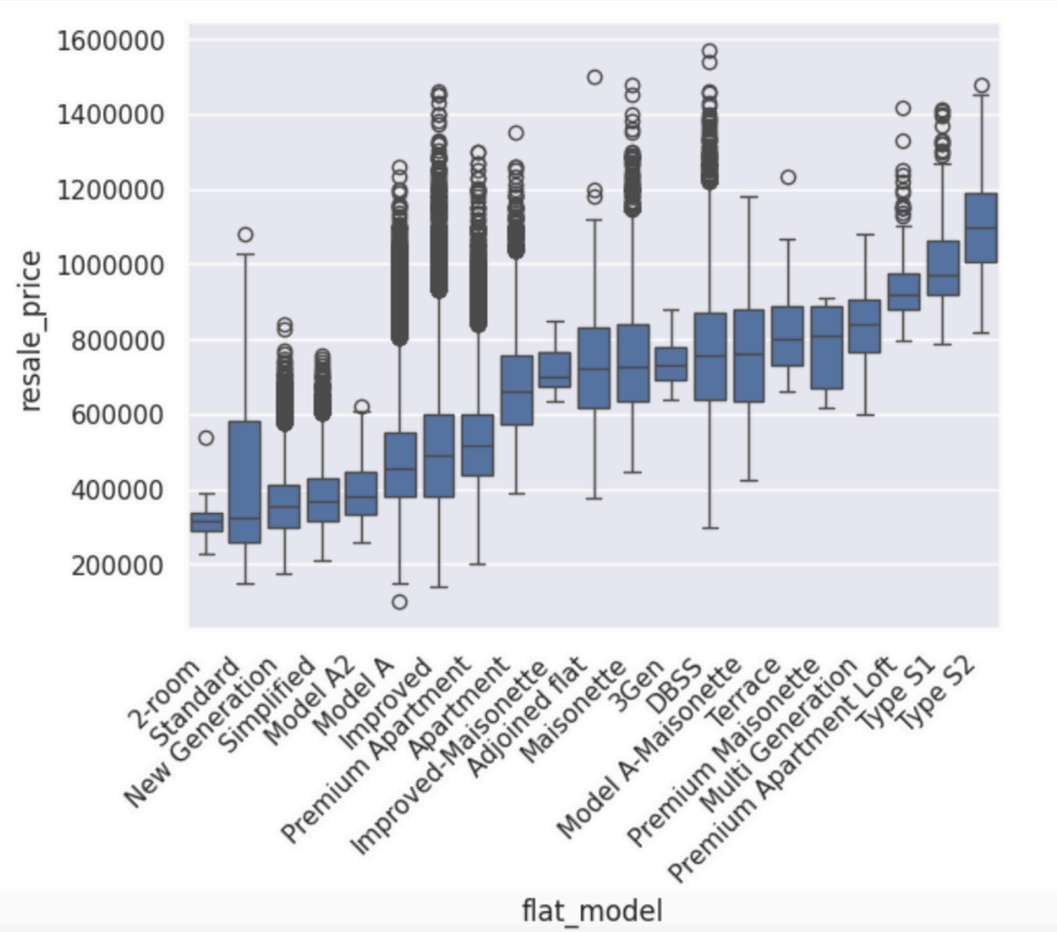
Flat Type vs Resale Price



Storey vs Resale Price



Flat Model vs Resale Price



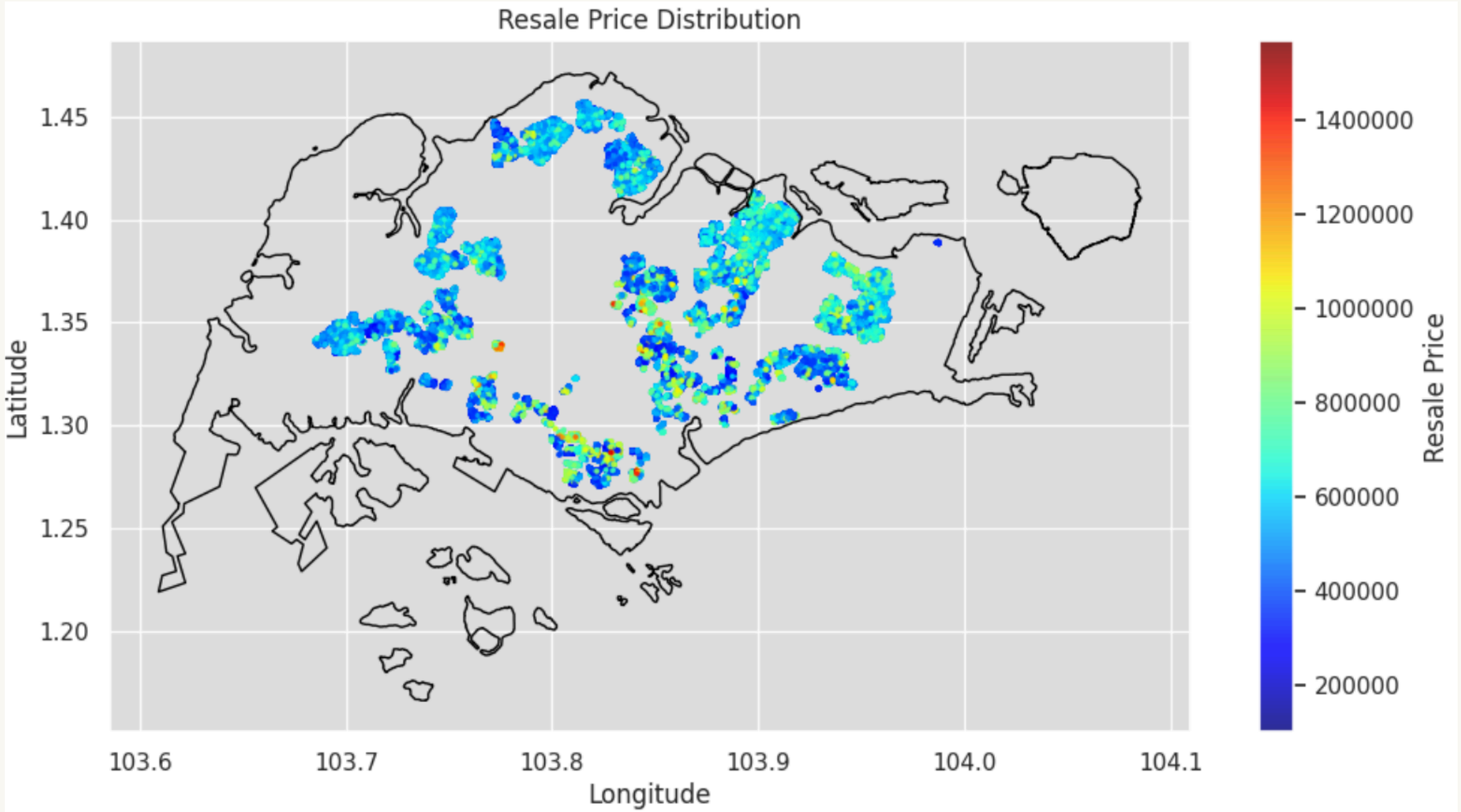
Flat types

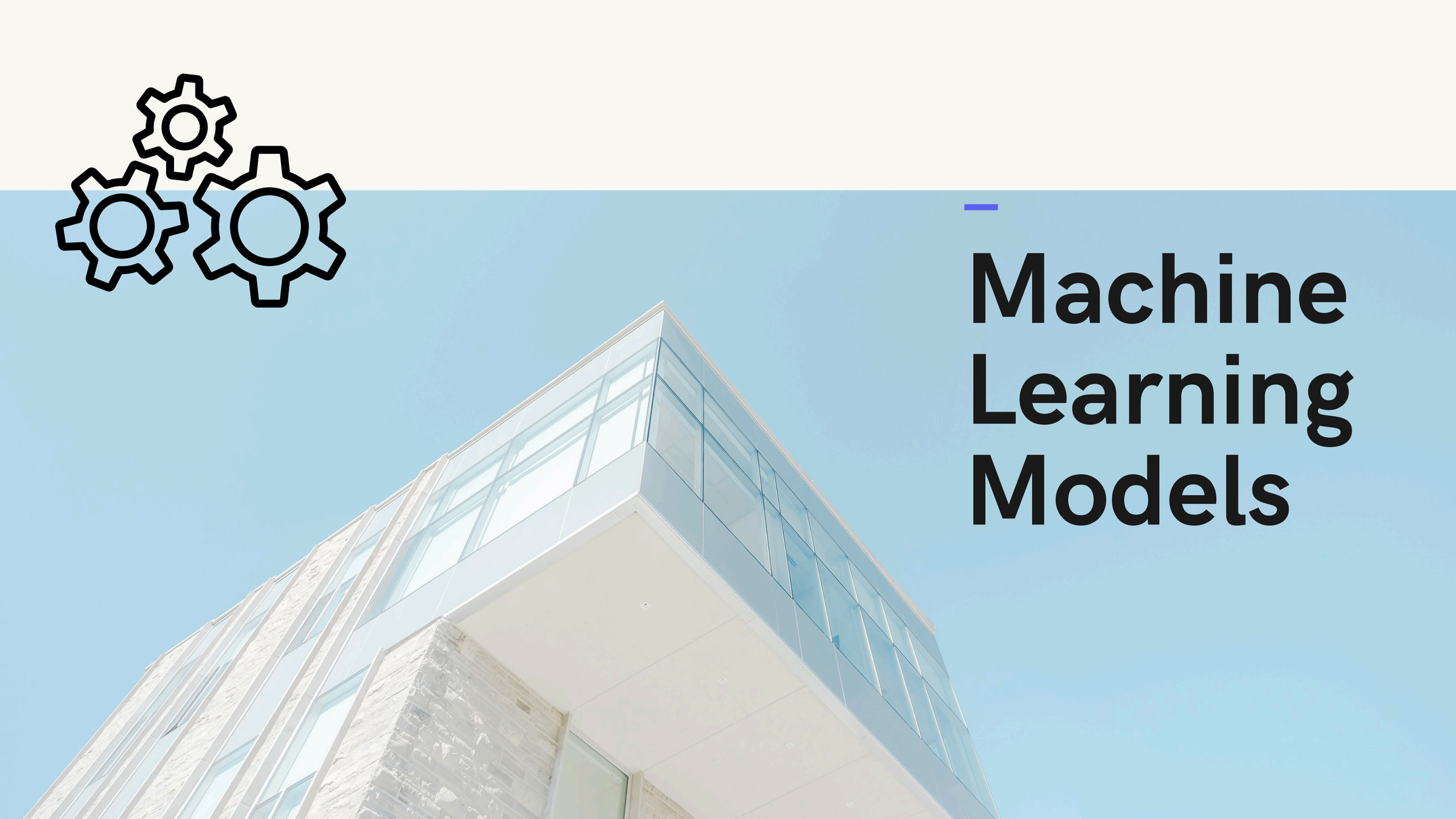
```
1  filtered_df["flat_type"].value_counts()
```

```
flat_type
4 ROOM                72617
5 ROOM                42907
3 ROOM                40844
EXECUTIVE             12511
2 ROOM                 3035
MULTI-GENERATION         77
1 ROOM                   64
Name: count, dtype: int64
```

4-Room flats have the highest volume of transactions

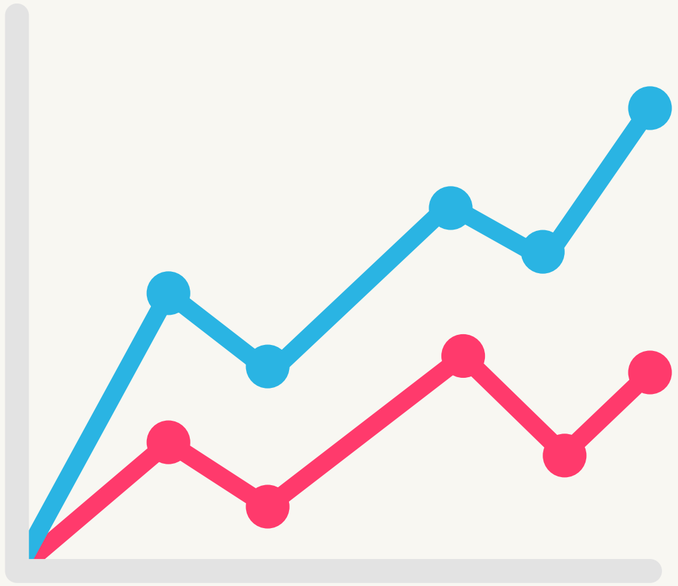
Resale prices in relation to geographic location





Machine Learning Models

Multivariate-linear Regression

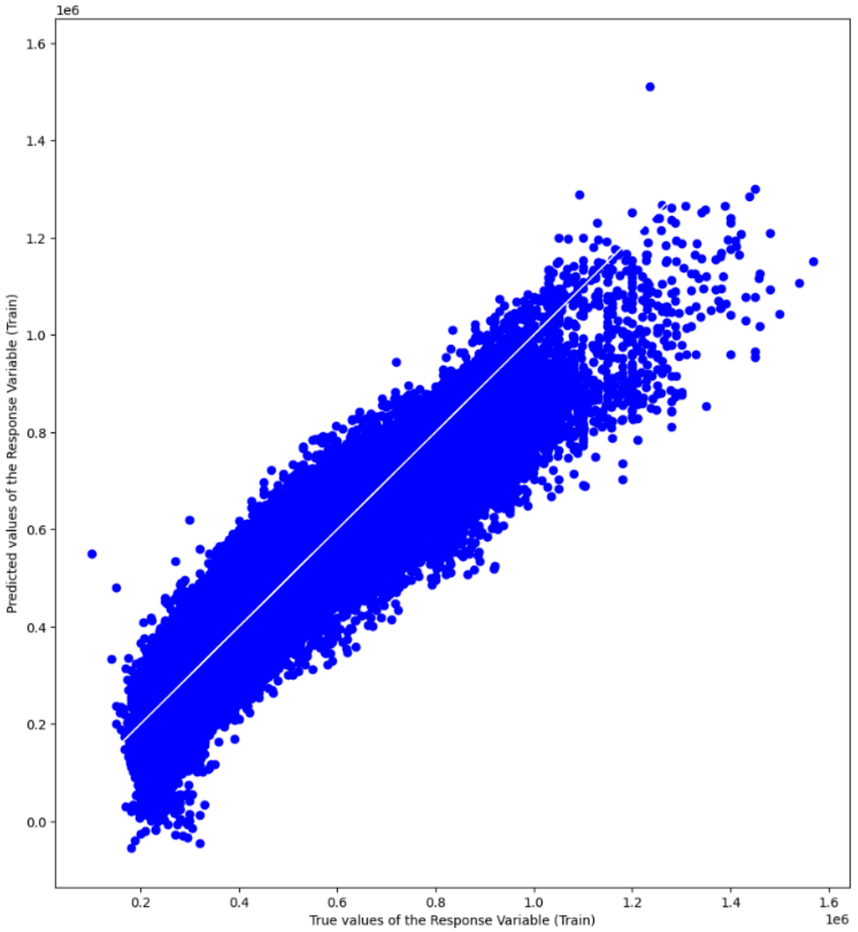


How does it work?

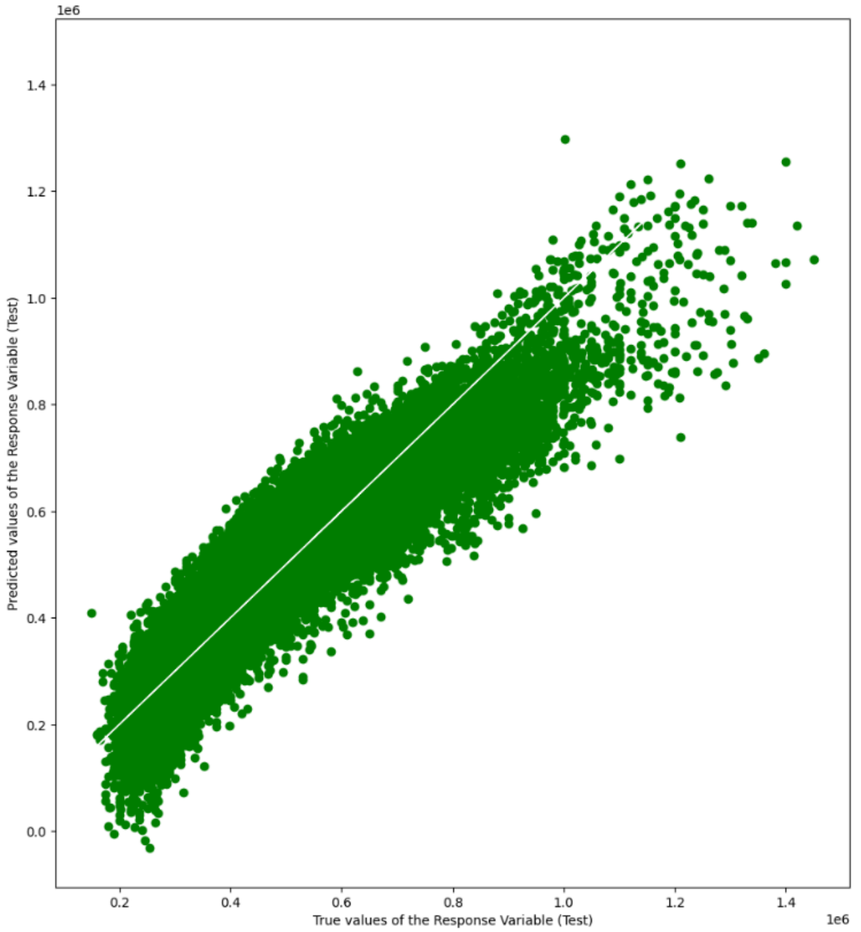
Predicted Price ≈ 5,900,000

+ 4300 * **Floor Area (sqm)**
+ 4200 * **Remaining Lease**
+ 5200 * **Floor Number**
+ 14000 * **Room Type (numeric)**
+ 2300 * **Months from Base Period (Jan 2017)**
- 460000 * **Latitude**
- 53000 * **Longitude**
- 16000 * **Distance from Centre (km)**
- 15 * **Distance to Nearest MRT (m)**
- 9.7 * **Distance to Nearest Mall (m)**

Train Data



Test Data



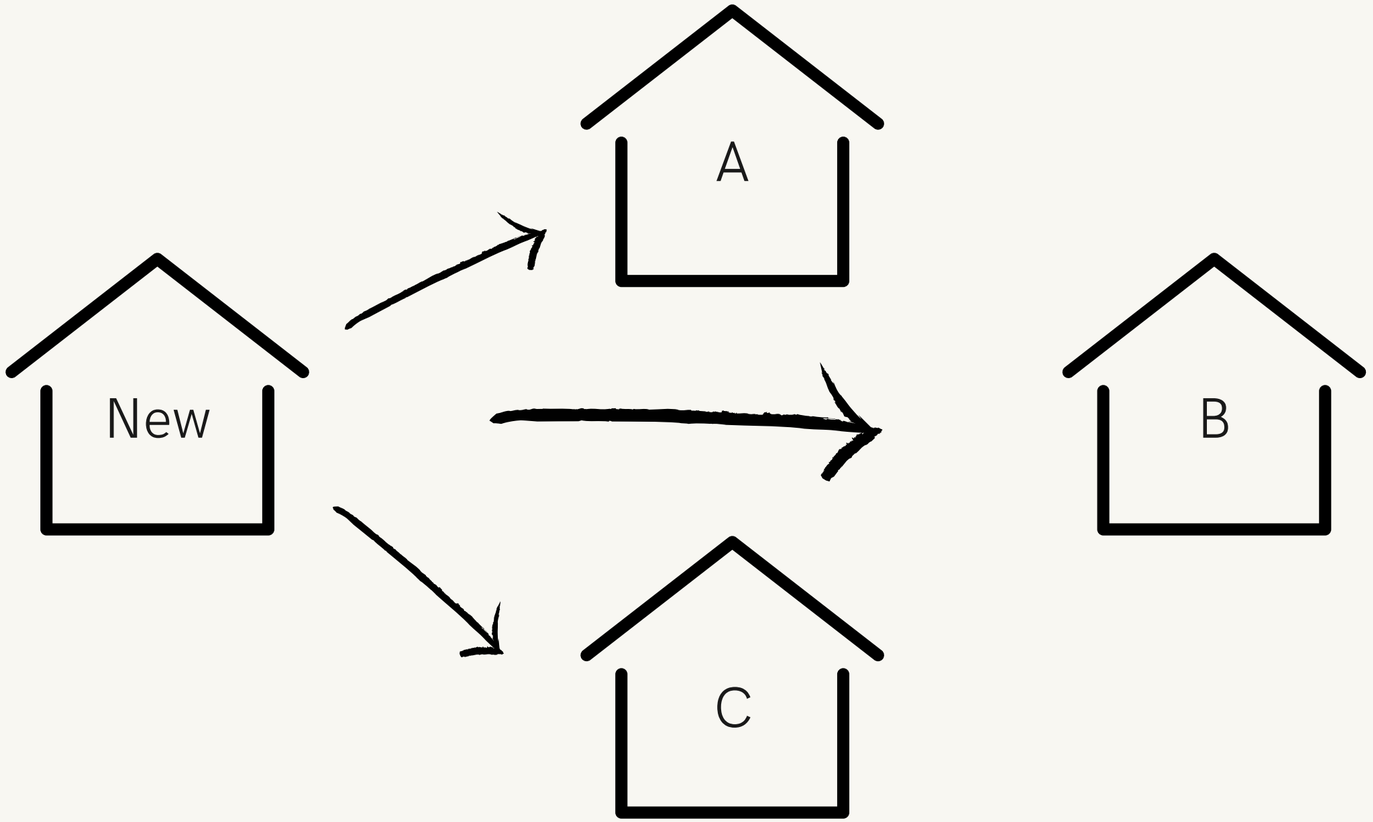
Evaluation of Multivariate-linear Regression

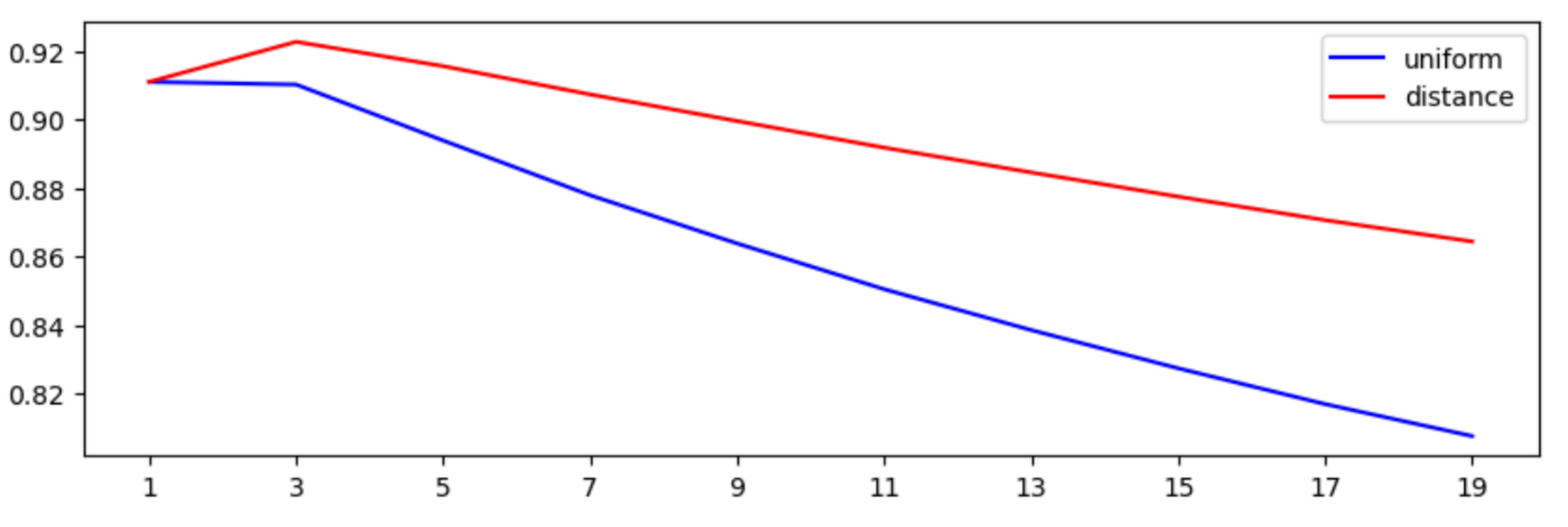
```
Goodness of Fit of Model        Train Dataset
Explained Variance (R^2)        : 0.808155937755407
Mean Squared Error (MSE)        : 5592256851.584773

Goodness of Fit of Model        Test Dataset
Explained Variance (R^2)        : 0.804086028431105
Mean Squared Error (MSE)        : 5735132362.21999
```

K-Nearest-Neighbour Regressor

How does it work?

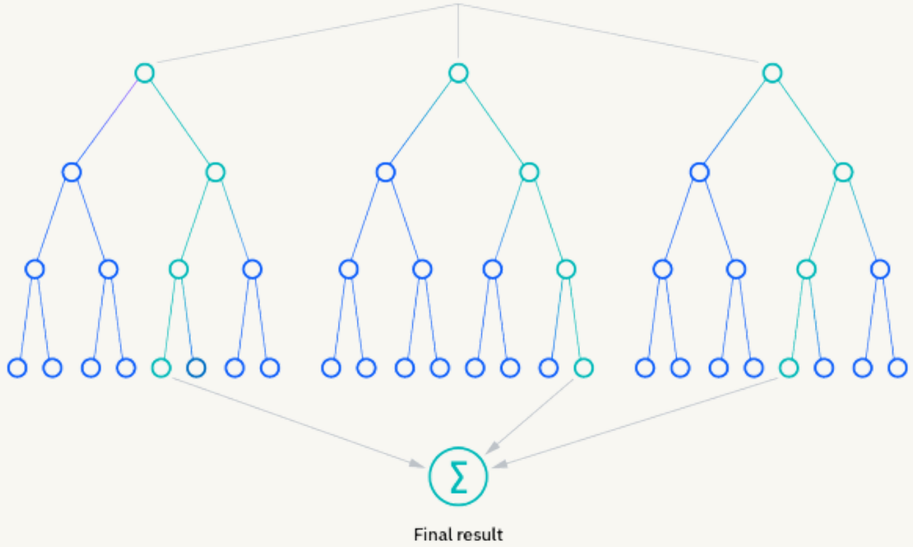



0.92
0.90
0.88
0.86
0.84
0.82
1
3
5
7
9
11
13
15
17
19
uniform
distance

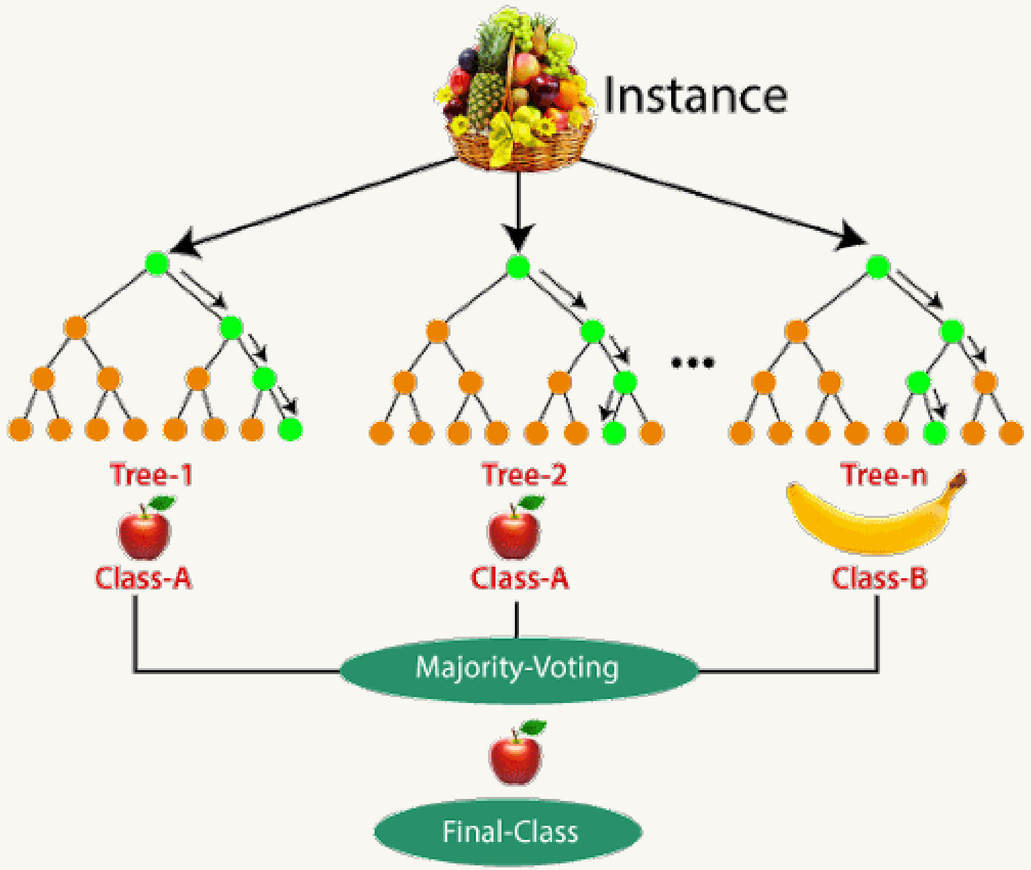
Evaluation of KNN

```
Explained Variance (R^2)          : 0.9226643141708415
Mean Squared Error (MSE)          : 53403248469.737236
```

Random Forest



How does it work?



Evaluation of Random Forest

```
Explained Variance (R^2)          : 0.9736618928030015
Mean Squared Error (MSE)          : 766385221.5424925
```

Evaluation of all 3 methods using R^2 and MSE

Model	R^2	MSE
Multivariate linear regression	0.804	5740000000
KNN	0.923	53400000000
Random Forest	0.974	766000000

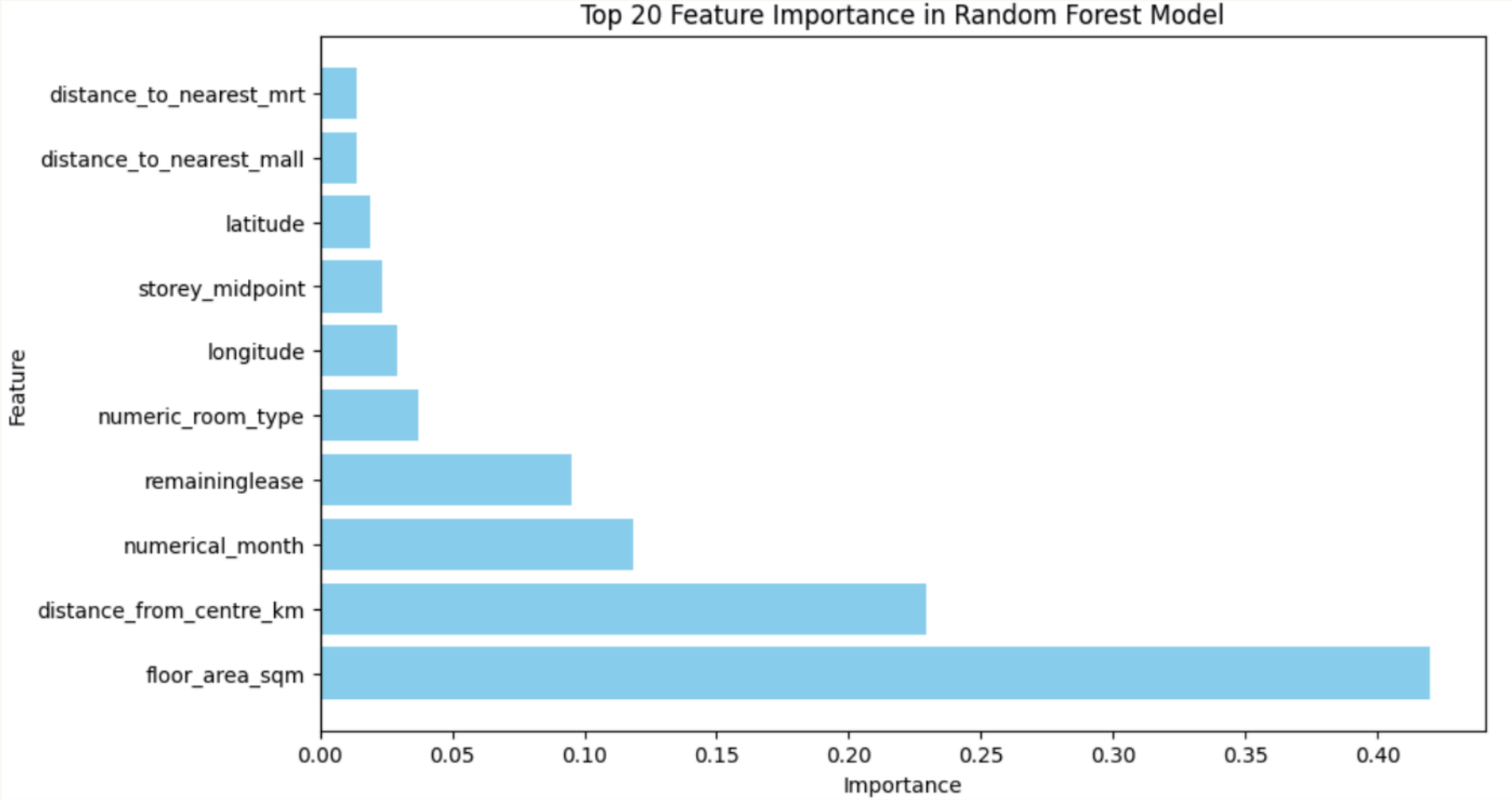
Tool to Calculate Resale Prices

Function that takes in user input to predict resale price

```
Please enter the property details.
Postal code: 380044
Latitude: 1.3163506
Longitude: 103.8773419
distance to nearest mrt is 618.9232856757742
distance to nearest mall is 1522.8483003483946
Floor area sqm: 88.0
Remaining lease: 50
Storey midpoint: 14
Year: 2024
Month: 3
Please input the corresponding number that matches to the flat type you are looking for.
1: 1 Room
2: 2 Room
3: 3 Room
4: 4 Room
5: 5 Room
6: Executive
7: Multi-gen
The predicted resale price is: $485062.76
```

Month	town	flat_type	storey_range	floor_area_sqm	flat_model
1/3/24 0:00	GEYLANG	4 ROOM	13 TO 15	88	Improved
lease_commence_date	resale_price	remaininglease	address	latitude	longitude
1/1/70	450000	50.5	44 SIMS DR	1.316235211	103.8768812
storey_midpoint	numerical_month	Distance_from_Centre (km)	distance_to_nearest_mrt	distance_to_nearest_mall	
14	87	4.204032467	669.949142	1497.737285	

Data Driven Insights



Thank you!