IE4102 INDEPENDENT STUDY COURSE

Unsupervised and supervised learning on Singapore housing dataset

Submitted by

Low Eeron A0216716E

DEPARTMENT OF INDUSTRIAL SYSTEMS ENGINEERING AND MANAGEMENT

NATIONAL UNIVERSITY OF SINGAPORE

Semester 1 2023

Summary

Introduction

The dynamic nature of the housing market necessitates the use of robust data analysis tools to aid potential buyers and sellers in making informed decisions. This report details the application of unsupervised and supervised machine learning techniques to the housing resale price dataset with a dual objective: developing a recommendation system and predicting housing resale prices.

Problem Definition and purpose

Housing decisions are often based on historical data, which sometimes may be incomplete or not representative of niche housing parameters. Buyers face the uncertainty of determining a fair price for a property, while sellers grapple with listing their homes at prices that reflect their true value. This project aims to address these challenges through:

a. Unsupervised Learning: This approach is designed to help users input their desired housing parameters and receive recommendations based on houses that have sold in the past with similar attributes.

b. Supervised Learning: This aspect of the project focuses on predicting a house's resale price based on varying parameters. It offers an estimation tool for users, providing insights into potential property values.

Key results

The culmination of this project resulted in the creation of two applications tailored to meet the needs of the housing market:

a. A recommendation system built upon unsupervised learning techniques that offer insights into properties sold in the past based on user-specified parameters.

b. A resale price prediction tool developed using supervised learning that provides users with a ballpark estimate, ensuring informed decision-making during transactions.

These applications ensure that both buyers and sellers have an upper hand in understanding and negotiating prices, bridging the existing information gap.

Conclusion

This project successfully marries unsupervised and supervised learning techniques to offer users a comprehensive perspective of the housing market. The recommendation system and the price prediction tool, when used in tandem, promise a more transparent and data-driven approach to housing transactions, benefiting all stakeholders involved.

Recommendation

For effective housing exploration, start with the recommendation system to find properties and make informed decisions. Then, use the resale price prediction tool for insights on transaction values. Keep datasets updated with quarterly resale prices and retrain models for precision. Stakeholders should prioritize regular tool updates. Improve prediction accuracy by adding diverse data, such as macroeconomic indicators and socio-demographic trends, for a comprehensive market view.

Acknowledgements

I express my sincere gratitude to Senior Lecturer Dr. Li Haobin for his invaluable guidance and unwavering support throughout the preparation of this report. Professor Li's insightful feedback and regular progress updates have been instrumental in shaping the quality of my work. I am truly grateful for the opportunity to benefit from his expertise.

Furthermore, I would like to extend my appreciation to the ISE department for granting me the opportunity to contribute to this project. The academic environment and resources provided have been crucial in the successful completion of this research endeavor.

A special acknowledgment goes to Teyang Lau, whose work served as a source of inspiration for my research in the supervised prediction housing resale part. Building upon Teyang Lau's approach, I chose to integrate the housing resale dataset with information on six key amenities to address the existing limitations in the depth of information available in the current housing resale price dataset. While our analyses share a common foundation, it's crucial to recognize the unique aspects of my work. The range of datasets used differs significantly; Teyang Lau utilized data spanning from 1990 to 2020, whereas I focused on Q2 of 2023 for my analysis. Additionally, I employed a combination of 1km and 2km radii in my predictive housing resale price analysis, and the parameters used for predicting housing resale prices were distinct, contributing to the distinctive nature of my research. [1]

Table of Contents

Contents

List of Figures

1.0 Introduction

1.1 Problem definition

The modern housing market in Singapore is characterized by its inherent fluidity, inundating potential homeowners and renters with vast amounts of information. Platforms like PropertyGuru offer basic filtering mechanisms, such as property type, price, number of bedrooms, and even by MRT stations, districts, and HDB estates[2].Yet, these often fall short of discerning buyers' comprehensive needs. While some prioritize proximity to essential amenities, others may seek a more concentrated presence of specific amenities. This underscores a gap in meeting the diverse needs of a varied user base.

Compounding the issue are observed price discrepancies on platforms like PropertyGuru, where similar properties in the same area may have varying listings[2].This inconsistency complicates decision-making and amplifies the need for better data transparency.

Beyond property discovery lies the challenge of interpreting abundant data. With a plethora of listings and information, stakeholders face the risk of overpricing or undervaluing properties. Determining a property's true market value in a dynamic environment like Singapore becomes an intricate endeavor.

Given this backdrop, two primary challenges emerge:

1. The complex task of navigating vast listings, with limited filtering options that may not fully address the unique needs of many buyers.

2. The essential need for accurate interpretation of data to ensure optimal property pricing in a fast-evolving market.

1.2 Scope

Central to this research is the application of machine learning, a powerful tool that can uncover patterns and insights from vast datasets, effectively addressing the aforementioned challenges. The goals are twofold:

Data Structuring for Tailored Insights: Aimed at simplifying the property search process, this facet of the research focuses on structuring and categorizing housing data, ensuring stakeholders can efficiently identify properties meeting their criteria.

Predictive Insights on Housing Resale Values: To address transaction pricing challenges, the research emphasizes accurate market value forecasting. In a market influenced by countless variables, predictive tools become vital.

By converging these objectives, this research offers a comprehensive solution: streamlining property searches and offering foresight into potential market values.

2.0 Literature Review

As the intricacies of housing market analysis are delved into, it is reminded that every discovery and insight stands on the shoulders of preceding work. A spotlight is cast on the foundational knowledge and tools that have shaped the field of geospatial and housing market analytics by the subsequent literature review. These established techniques, borne out of rigorous academic inquiry and practical application, have paved the way for refined approaches to data analysis. Once a comprehensive understanding of the literature is attained, an in-depth exploration will follow, focusing on the specific methodologies tailored for our study. Each methodology has been chosen for its precision, relevance, and efficacy in deciphering complex datasets like ours.

2.1 Distance Calculation Using distVincentySphere

In the realm of geospatial analysis, accurately computing the distance between two points on the Earth's surface becomes paramount. The Earth's shape, deviating from a perfect sphere to resemble an oblate spheroid, complicates such calculations. Nevertheless, for many applications, where ultra-high precision isn't the end goal or over distances where the Earth's subtle flattening is less impactful, approximating Earth as a sphere proves reasonable. This leads to the use of the distVincentySphere, a derivative of Vincenty's formula that assumes a spherical Earth. This method finds favor for its commendable balance between precision and computational simplicity, especially in scenarios prioritizing performance [3].

In the context of the Singapore housing market, geospatial analysis is considered indispensable. The value of a property can be significantly influenced by its proximity to amenities, transportation hubs, and other urban landmarks. The distVincentySphere is leveraged so that distances between properties and these critical points of interest can be accurately calculated, ensuring that spatial attributes of properties are correctly captured and represented. Furthermore, it is recognized that, given Singapore's compact urban landscape, notable value differences can be led to by even minor distance variations, highlighting the essential need for accurate distance measurements.

2.2. Unsupervised Learning Techniques

Unsupervised learning excels in situations where the goal is to identify inherent patterns or structures in data without explicit labels [4]. Given the multifaceted attributes of the housing dataset, these techniques can be instrumental in uncovering hidden relationships or clusters.

Clustering aids in categorizing data into groups based on similarity measures. Such categorization can be pivotal for datasets like the one at hand, which encompasses various attributes ranging from flat type to proximity to amenities [5].

2.2.1 K-means

Segmenting the housing dataset into 'K' clusters based on similarity, K-means can discern common housing patterns or categories. The algorithm minimizes the intra-cluster variance, ensuring that data points in the same cluster are as similar as possible [6].

Within the housing dataset, K-means can be instrumental in segmenting properties into distinct clusters based on attributes like size, location, and type. This can offer potential buyers a more tailored list of properties that fit within a specific category of interest

2.2.2 Hierarchical Clustering

This method provides a tree-like structure of clusters, making it suitable for understanding data at various granularities. Hierarchical clustering is especially pertinent when there's potential nested structure in the data, such as housing categories and subcategories [7].

Given the multi-layered nature of housing attributes, hierarchical clustering can aid in understanding how properties can be categorized at multiple levels. For instance, the first level might cluster based on property type, while subsequent levels could focus on price brackets or proximity to amenities

2.2.3 DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Unlike K-means, which assumes clusters to be spherical, DBSCAN can detect clusters of arbitrary shapes. This is particularly useful for complex datasets where clusters may not be clearly defined. Moreover, its ability to distinguish noise or outliers ensures that anomalous data points don't influence the clustering unduly [8].

For a dataset as varied as housing, where properties might not always fit neatly into spherical clusters, DBSCAN provides flexibility. It can highlight unique or niche property groupings that other algorithms might miss, ensuring a more comprehensive analysis

2.3 Supervised Learning Techniques

In scenarios with labeled data, such as housing datasets with known resale prices, supervised learning techniques become indispensable. These techniques model the relationship between input attributes and the target variable. In the context of our study, supervised learning is employed to predict housing resale prices based on various features of the data [9].

2.3.1 Random Forest

An ensemble method, Random Forest constructs multiple decision trees, enhancing prediction accuracy and robustness against overfitting. This technique can handle a wide range of data attributes without extensive preprocessing [10].

In predicting housing prices, the Random Forest algorithm can accommodate the myriad of features associated with properties, from basic attributes like size and age to more nuanced ones like view quality or architectural significance

2.3.2 XGBoost (Extreme Gradient Boosting)

Celebrated for its computational efficiency and prediction accuracy, XGBoost is a scalable take on gradient boosting. It builds trees in a sequential manner, where each tree corrects the errors of its predecessor [11].

With its ability to iteratively enhance prediction accuracy, XGBoost can refine housing price predictions, ensuring they resonate with the dynamic Singapore housing market trends

2.3.3 Support Vector Machines (SVM)

SVMs are adept at both classification and regression tasks. They work by identifying an optimal hyperplane that best separates data into classes. The kernel trick in SVM allows it to handle non-linear data, providing flexibility in modeling complex relationships [12].

SVM's flexibility in handling non-linear relationships makes it a strong contender for modeling the intricate variables influencing housing prices

2.3.4 Gradient Boosting Machine (GBM)

GBM, like XGBoost, builds trees iteratively. Each tree aims to correct the residuals or errors from the preceding trees, ensuring that prediction accuracy improves with each iteration [13].

Much like XGBoost, GBM's iterative approach can be instrumental in fine-tuning predictions, ensuring that as the model learns, it gets progressively closer to actual market values

2.3.5 Lasso and Ridge Regression

Both are regularization techniques for linear regression. Lasso performs feature selection, ensuring irrelevant features don't influence the model, while Ridge provides robustness against multicollinearity, ensuring stable coefficient estimates [14,15].

With the multitude of potential influencing factors in a housing dataset, Lasso can ensure that only the most relevant features impact the model. Simultaneously, Ridge Regression provides a safeguard against any inter-correlation between features, ensuring a more stable and reliable prediction

2.4 Evaluation Metrics in Supervised Learning

Evaluation metrics ensure that the supervised models' predictions align well with the actual outcomes, providing a quantitative measure of performance.

2.4.1 Root Mean Square Error (RMSE)

RMSE represents the square root of the average squared differences between the predicted and actual values. In simpler terms, it measures the average magnitude of errors made by a predictive model [16].

In the context of predicting housing resale prices, RMSE provides a clear indication of the model's overall prediction accuracy in terms of the actual monetary difference. A lower RMSE indicates that the model's predictions are closer to the actual resale prices, making it a vital metric for gauging the model's effectiveness. Given the substantial financial implications of housing transactions, even small prediction errors can translate to significant monetary discrepancies. Hence, minimizing RMSE becomes paramount to ensure stakeholders make informed decisions based on the model's outputs.

2.4.2 Mean Absolute Percentage Error (MAPE)

MAPE calculates the average of the absolute percentage differences between the predicted and actual values. It provides a relative measure of error, expressing prediction inaccuracies as a percentage [17].

While RMSE provides an absolute measure of error in terms of monetary units, MAPE offers a relative perspective, making it easier for stakeholders to contextualize the model's prediction accuracy. For instance, a MAPE of 5% suggests that, on average, the model's predictions deviate from actual prices by 5%. This percentage representation can be especially intuitive for stakeholders, allowing them to gauge the model's reliability in a more comprehensible format. In the dynamic housing market of Singapore, where prices can vary widely across districts and property types, having a relative measure like MAPE can be particularly insightful for potential buyers, sellers, and investors.

3.0 Methodology

An in-depth exploration into Singapore's housing resale market was undertaken, blending data science, spatial analytics, and machine learning techniques. This section elucidates the intricate methodologies employed.

3.1 Geospatial Data Handling and Proximity Analysis: An Overview

To provide a brief overview of our research approach: Datasets encompassing housing resale data and various amenities were primarily acquired from Data.Gov.sg and Kaggle. Missing spatial data were supplemented using geocoding, converting addresses or postal codes to latitude and longitude details. Subsequent spatial analyses determined property proximity to amenities and the density of amenities within specific radii. Following this geospatial integration, a recommendation system was developed, and predictive models for housing resale prices were formulated. Figure 3.1 offers a visual summary of this workflow.

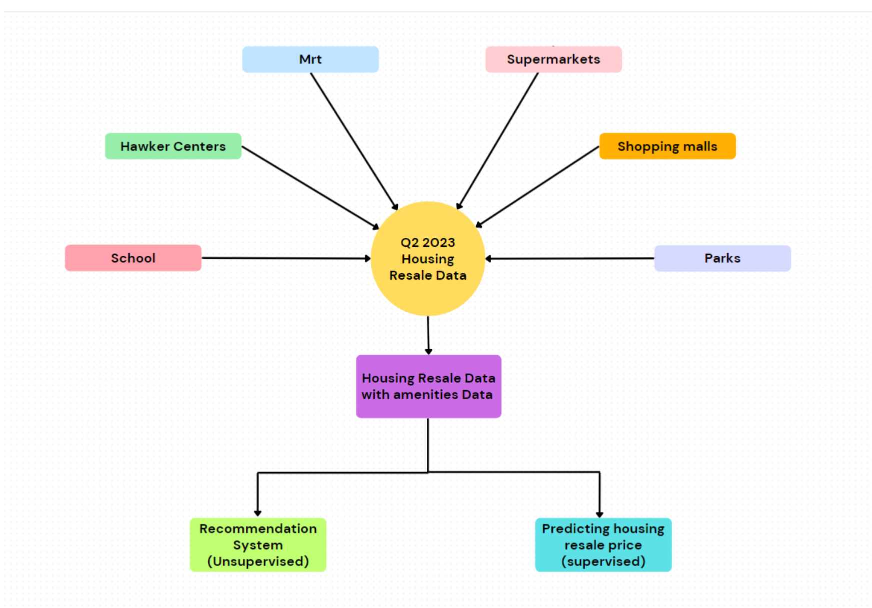


Figure 3.1: Overview of Data analysis

With this visual and descriptive guide, the subsequent sections will delve into the detailed methodologies and their nuances.

3.2 Data Acquisition and Refinement

3.2.1 Dataset Compilation and Selection

Data for the study was sourced from Data.Gov.sg [18], a recognized data repository. The Q2 2023 dataset was chosen to offer a contemporary perspective on housing resale trends. The research incorporated datasets including: Housing Resale Data [18] , School Data [19], Hawker Centers [20],MRT Stations [21], Supermarkets [22], Shopping Malls [23], Parks [24]

3.2.2 Geospatial Augmentation of the Base Dataset

The base dataset was initially devoid of latitude and longitude details. To address this, the ggmap package was utilized in tandem with the Google Maps API to geocode addresses, thereby equipping each property with accurate spatial coordinates.

3.2.3 Extraction of Spatial Coordinates for Amenities

For parks and hawker centers, KML files were processed to retrieve their spatial coordinates. Using the st_read function, KML data was read, and coordinates were subsequently extracted.For schools and supermarkets, where latitude and longitude were absent, a dedicated function leveraging the ggmap package and Google Maps API was crafted to geocode postal codes. This function transformed postal codes into accurate spatial markers.

3.2.4 Spatial Proximity Analysis

Once all datasets were enriched with spatial coordinates (longitude and latitude), a comprehensive spatial analysis was conducted. The proximity of each property to various amenities was determined:

For all amenities, including hawker centers, MRT stations, parks, shopping malls, supermarkets, and even schools, two primary metrics were derived: distance to the closest facility and number of facilities within a specific radius. However, while a 1km vicinity was considered for most amenities, schools were evaluated within a broader 2km boundary due to their distinct importance in housing decisions. The decision to consider a 2km radius for schools was informed by the understanding that a larger catchment area increases the likelihood of children gaining admission. This expanded radius is deemed beneficial for families with school-going children, as proximity to preferred educational institutions often plays a pivotal role in housing decisions [25].

3.3 Data Cleaning and Enhancement

3.3.1 Data Integrity and Quality

For coherent analyses, columns with mixed temporal data, like 'remaining_lease', were split into separate 'years' and 'months' columns. Further, categorical variables, including 'town', 'flat_type', 'storey_range', and 'flat_model', were transformed via one-hot encoding, making them amenable to machine learning techniques.

3.3.2 Handling Missing Data

In cases where certain amenities lacked spatial coordinates, innovative geocoding methods were employed. Postal codes available for these amenities were transformed into latitude and longitude markers using the aforementioned geocoding function.

3.3.3 Data Validation

Post data acquisition, a validation phase was incorporated. The constructed price prediction model was tested against the latest Q3 2023 data, and the predictions were found to be within Q1 and Q3 of the predicted range, affirming the model's reliability.

3.3.4 Data Refinement and Preliminary Analysis

Upon validation, our study embarked on thorough data cleaning, eliminating non-compliant entries and superfluous columns, enhancing data reliability. This refined data was cataloged in a 'final' CSV file, earmarked for ensuing inquiries. In tandem, insightful data visualizations, encompassing bar charts were crafted, offering an instant, lucid comprehension of the geographical spread of amenities. This phase also embraced a focused descriptive statistical analysis, involving key summary statistics — mean, median, and others — to capture the data's essence. These processes not only affirmed the data's robustness but also set the stage for in-depth, successive analyses.

3.4 Feature Engineering

3.4.1 Temporal Decomposition

The 'remaining_lease' column, teeming with combined temporal data of years and sometimes months, was elegantly decomposed to facilitate a more granulated temporal scrutiny.

3.4.2 Encoding and Spatial Feature Creation

Ensuring the dataset's compatibility with machine learning models, categorical variables underwent a transformation via one-hot encoding. Moreover, the spatial coordinates obtained through geocoding were instrumental in devising spatial features. These features encompassed proximity metrics and density measurements, which added depth to the dataset's spatial narrative.

3.5 Model Development, Validation, and Interpretation

3.5.1 Model Assumptions

Several assumptions were made during the modeling phase. It was assumed that external monetary conditions had no notable effect on the resale price fluctuations. Furthermore, popular amenities like renowned schools, shopping malls, or MRT stations were assumed not to exert a significant impact on the resale prices. Given that the dataset combines data from different years, it was assumed that such combinations did not introduce crucial disparities in the results or the model's performance.

3.5.2 Model Validation and Overfitting

To fathom the performance of models beyond their training data, the dataset was bifurcated into training and test subsets. Embracing cross-validation techniques fortified the models against overfitting, ensuring their robustness. Regularization techniques, including LASSO and Ridge regression, were integrated, placing constraints on coefficient magnitudes and ensuring model generalizability. For hyperparameter tuning, a conscious decision was made not to overfit the model. The Mean Absolute Percentage Error (MAPE) was observed to be less than 5% for both Random Forest and XGBoost, indicating satisfactory model performance.

3.5.3 Model Interpretation

The Random Forest model, an ensemble methodology, was trained, unveiling feature importance and providing a spectrum of expected resale prices. Concurrently, the XGBoost model was leveraged to capture intricate non-linear relationships, highlighting influential variables. The robustness of models was further validated by computing the Root Mean Square Error (RMSE) for an array of models, ensuring a comprehensive and rounded analysis.

3.6 Model Comparison

An array of models was trained, and their performance was rigorously compared using RMSE and MAPE. The Random Forest and XGBoost models were ultimately chosen due to their superior performance metrics.

3.7 Comprehensive Data Management

For the sanctity of consistency and future replication potential, an unwavering data management protocol was adopted. Every significant analytical stride and its results were meticulously documented and archived in a structured CSV format, laying the groundwork for subsequent analyses or visualizations.

3.8 Crafting the Shiny App

At the heart of the digital solution is a Shiny app, a product of meticulous design and tailored functionalities, aimed specifically at potential homeowners in Singapore.

3.8.1 Recommendation System (Unsupervised Learning)

Dynamic User Interface: The Shiny app's recommendation system offers a dynamic and interactive interface where users can specify their preferences, from proximity to amenities to types of neighborhoods. This interface is not just about input; it's about experience. It offers users a chance to navigate through various clustering methods, allowing for customization at every step. The culmination of this is a detailed map, visually representing recommended housing areas based on user-defined criteria.

Behind the user interface, a sophisticated server logic operates, processing user inputs, filtering data accordingly, and applying the appropriate clustering techniques. Once clusters have been established, specific clusters are highlighted based on predefined parameters and clustering criteria, offering users a curated list of recommendations.

3.8.2 Predictive Models (Supervised Learning)

Data Preparation: Before any predictive modeling could commence, the dataset underwent thorough preprocessing. Essential transformations were applied to ensure that the data was in the optimal format for modeling. It was then divided into training and test subsets, setting the stage for the subsequent model training phase.

Model Training and Evaluation: An array of regression models was trained on the dataset. Each model's performance was rigorously evaluated using established metrics. This comparative analysis allows for the identification of the most accurate models for price prediction.

In addition, a Manual Prediction Shiny App was introduced. Here, users can input specific housing attributes. These inputs are then processed and used by the trained models to predict resale prices. A buffer is added to account for uncertainties, ensuring users receive a prediction range, offering a more realistic and practical forecast.

3.9 Limitations and Challenges

One notable limitation of this study was the dataset's recency. Although efforts were made to incorporate the most recent data, the dataset for amenities wasn't the absolute latest available. Another challenge was the amalgamation of data across different years, which brought its own set of complexities.

4.0 Results

4.1 Preliminary Analysis of the Data

An initial exploration of the dataset was conducted, providing insights into the spatial distribution of amenities in relation to housing units across different towns in Singapore. The findings, illustrated by the provided visualizations and summary table, offer a nuanced perspective on the housing landscape in Singapore.

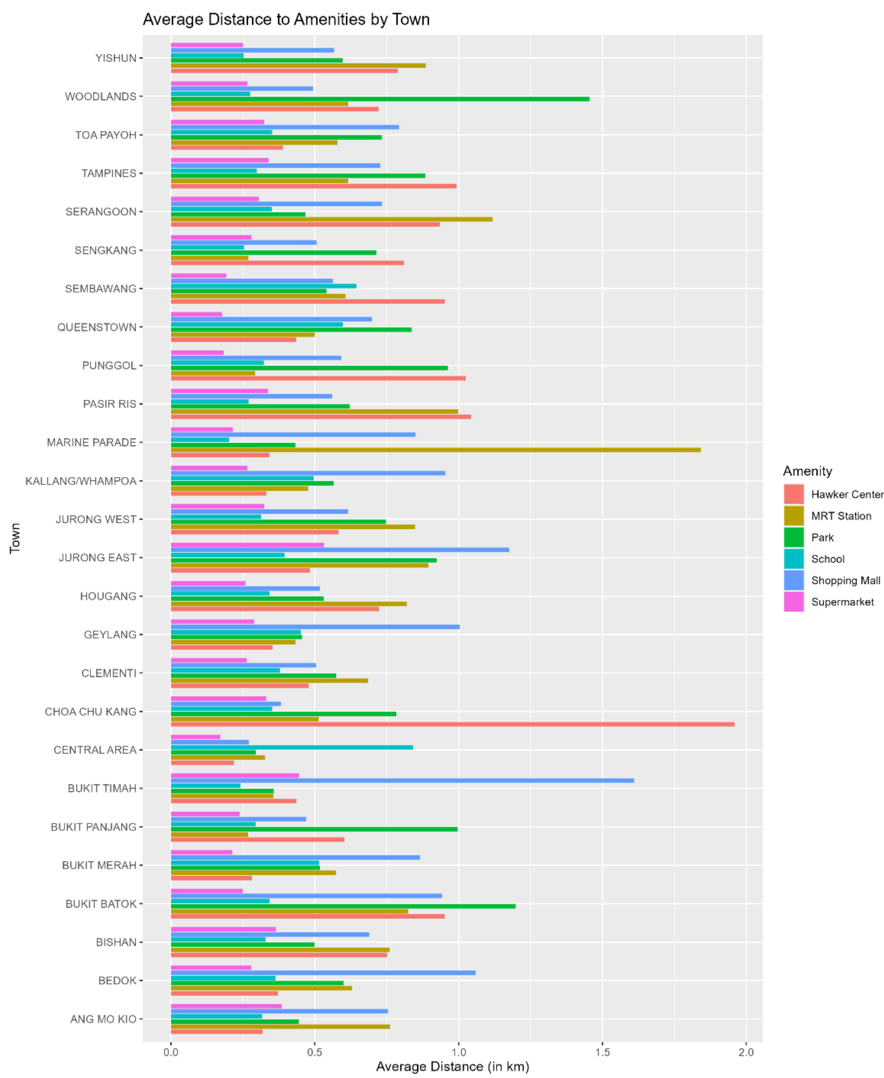


Figure 4.1: Average distances in km arrange by town

Figure 4.1 delineates the average distances from housing units to various amenities across different towns. Predominantly, it can be observed that the majority of the towns have amenities situated within an average distance of 1km. However, there are specific exceptions that stand out: Woodlands residents, on average, find themselves approximately 1.5km away from the nearest park. In Marine Parade, the average distance to the closest MRT station is about 1.75km.Choa Chu Kang exhibits a slightly extended average distance of around 2km to the nearest hawker center. For those residing in Bukit Timah, the average distance to a shopping mall is about 1.6km.

These variations emphasize the spatial heterogeneity in amenity accessibility across different towns and underscore the importance of geospatial analysis in understanding urban dynamics.

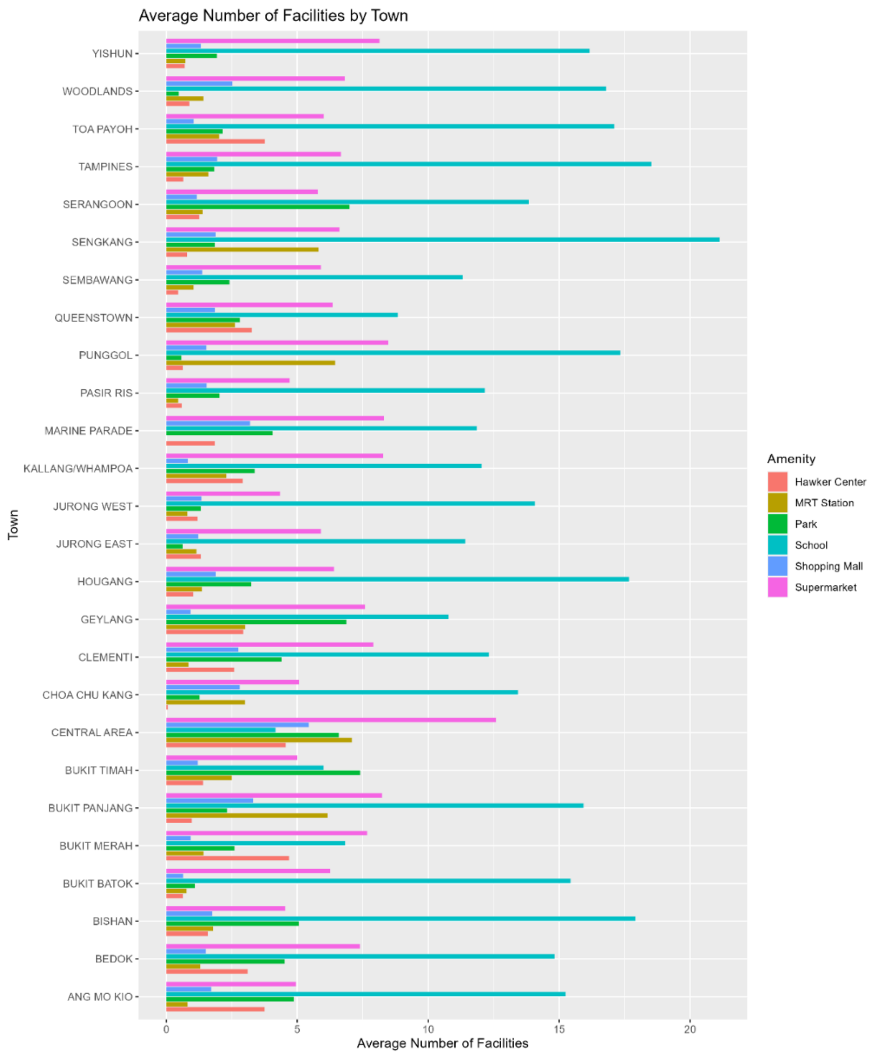


Figure 4.2: Average number of facilities arrange by town

Figure 4.2 illustrates the density of various amenities across different towns. Overall, towns seem to exhibit a consistent number of facilities, indicating a balanced distribution of resources. Specific observations that stand out are:

Central Area displays a relatively lower density of school facilities compared to other regions, suggesting potential urban planning strategies that prioritize other amenities. Sengkang stands out with the highest density of schools, reflecting the area's potential appeal to families with school-going children. When it comes to supermarkets, the Central

Area leads with the highest density, catering to the urban population's daily needs. On the contrary, Jurong West appears to have a slightly lower density of supermarkets.

These observations underscore the unique amenity profiles of each town, with each catering to distinct resident needs and urban planning priorities.

Descriptive Statistics of Distances and Counts:

	Min	Median	Mean	Max	X1st.Qu..25.	X3rd.Qu..75.
hawker_closest_distance	0.033691061138632	0.640130826777672	0.761725473886178	2.80398433388695	0.371057398578678	1.02172033451958
num_hawker_1km	0	1	1.43161343161343	9	0	2
mrt_closest_dist	0.0264104881460214	0.552227521337943	0.617313454533644	3.41317620024466	0.336933801516211	0.825955311936475
num_mrt_1km	0	1	2.38886158886159	14	1	3
park_closest_dist	0.0681389534230855	0.686109097951199	0.78765783301917	2.40322990458056	0.46813407138607	0.995607483107017
num_park_1km	0	1	2.28058968058968	15	1	3
sch_closest_dist	0.00812176866117962	0.293563337370964	0.345709935582443	3.29578401975786	0.195406361779354	0.426793142889845
num_sch_2km	0	16	15.3772317772318	29	12	18
shoppingmall_closest_dist	0.00505698397679216	0.595793485996857	0.66374383359772	3.11757249627717	0.391263429374416	0.873920675335062
num_shoppingmall_1km	0	2	1.6972972972973	15	1	2
supermarket_closest_dist	0	0.251493000806214	0.276753460509932	3.09937827460337	0.161356218160154	0.367957784724172
num_supermarket_1km	0	6	6.69041769041769	18	5	9

Figure 4.3: Summary statistics of the amenities

Figure 4.3 above provides a detailed perspective on the dataset's metrics, offering insights into the central tendencies and spread of distances to amenities and their respective counts. Specific observations from the table include:

Supermarkets stand out with the lowest average distance from housing units, signifying their widespread presence and accessibility in Singapore. This is further emphasized by the fact that all amenities, on average, are located within a 1km radius of housing units, underscoring the well-planned urban infrastructure of Singapore. Contrarily, parks exhibit the highest average distance, suggesting that while they are integral to Singaporean neighborhoods, they might be spaced out a bit more compared to other amenities.

When considering the density of amenities within a 1km radius, supermarkets again lead with the highest average count, emphasizing their prevalence in residential areas. On the other hand, hawker centers, despite being iconic to Singapore's culture, have the lowest average count within the same radius.

These metrics elucidate the intricate balance of amenities in Singapore, highlighting the emphasis on both accessibility and distribution in urban planning. In light of these findings, it was concluded that each town in Singapore offers a unique blend of accessibility and amenity density. These preliminary insights pave the way for more in-depth analyses in the subsequent sections of the report.

4.2 Model Selection Based on Performance Metrics

Based on the performance metrics, particularly the Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE), Random Forest and XGBoost were chosen as the primary models for predicting resale prices of housing units.

The selection of these models is evident when observing the RMSE and MAPE figures. Both metrics provide insight into the accuracy of predictions made by the models. Lower RMSE values indicate better fit to the data, while lower MAPE values suggest the model's predictions are, on average, closer in percentage terms to the actual values.

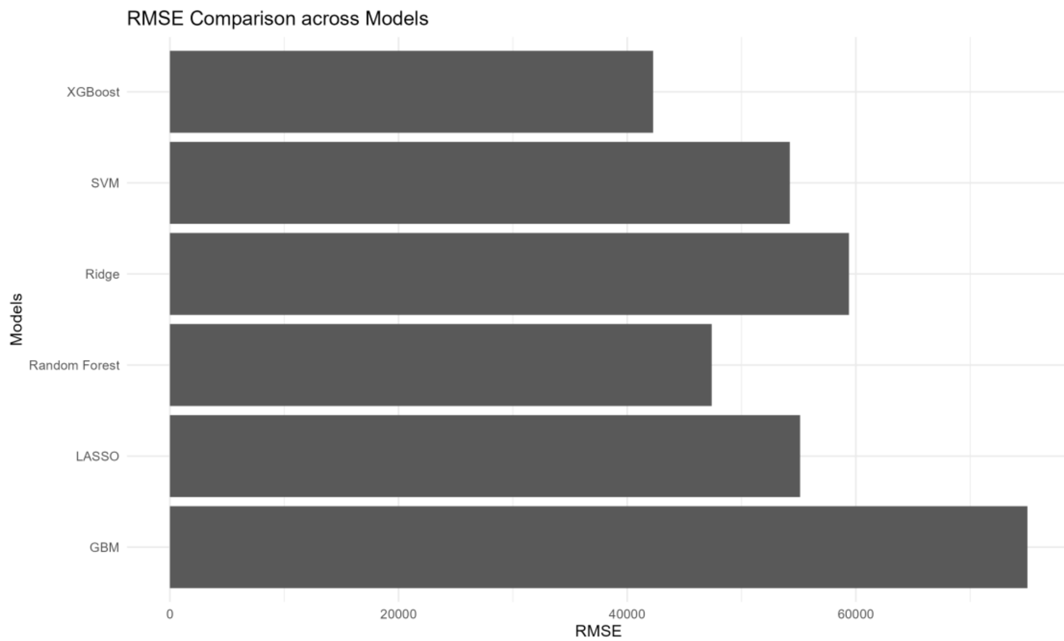


Figure 4.4: Comparison of RMSE across Different Models

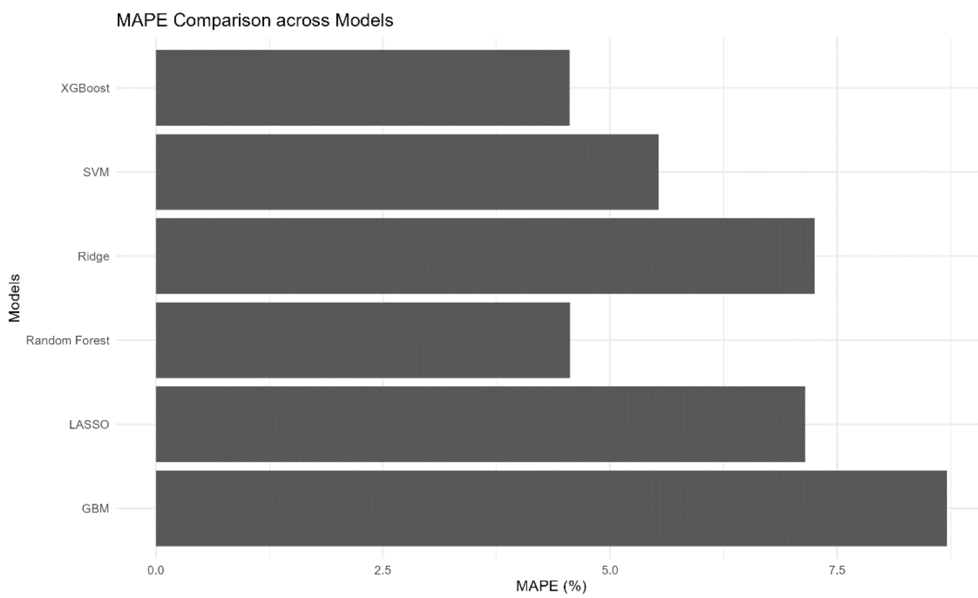


Figure 4.5: Comparison of MAPE across Different Models

From Figures 4.4 and 4.5, it's evident that both Random Forest and XGBoost outperformed the other models in terms of these metrics, making them the ideal choice for this dataset and the problem at hand.

4.3 Feature importance on random forest and xgboost

The significance of features, as determined by machine learning models, can provide profound insights into the factors that most influence the resale prices of housing units.

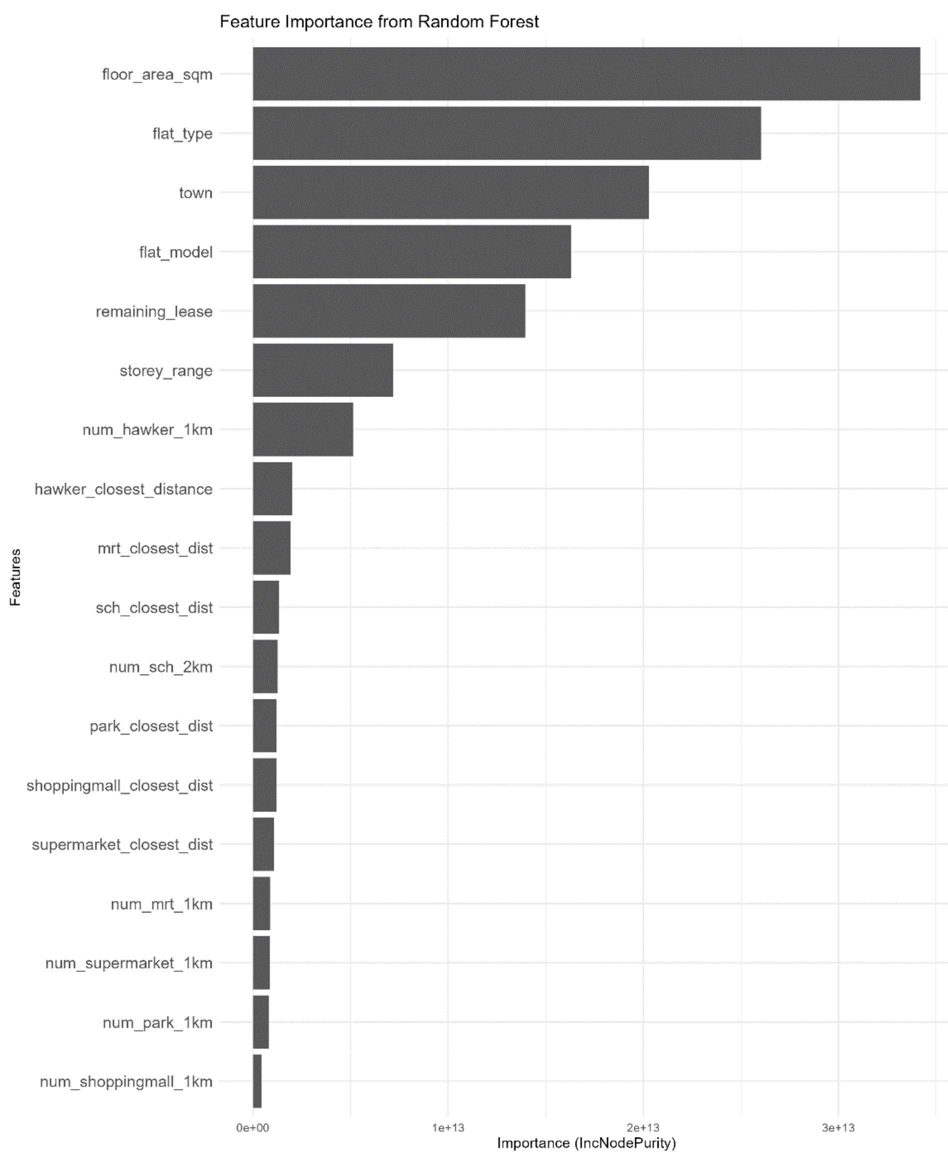


Figure 4.6: Random forest feature importance

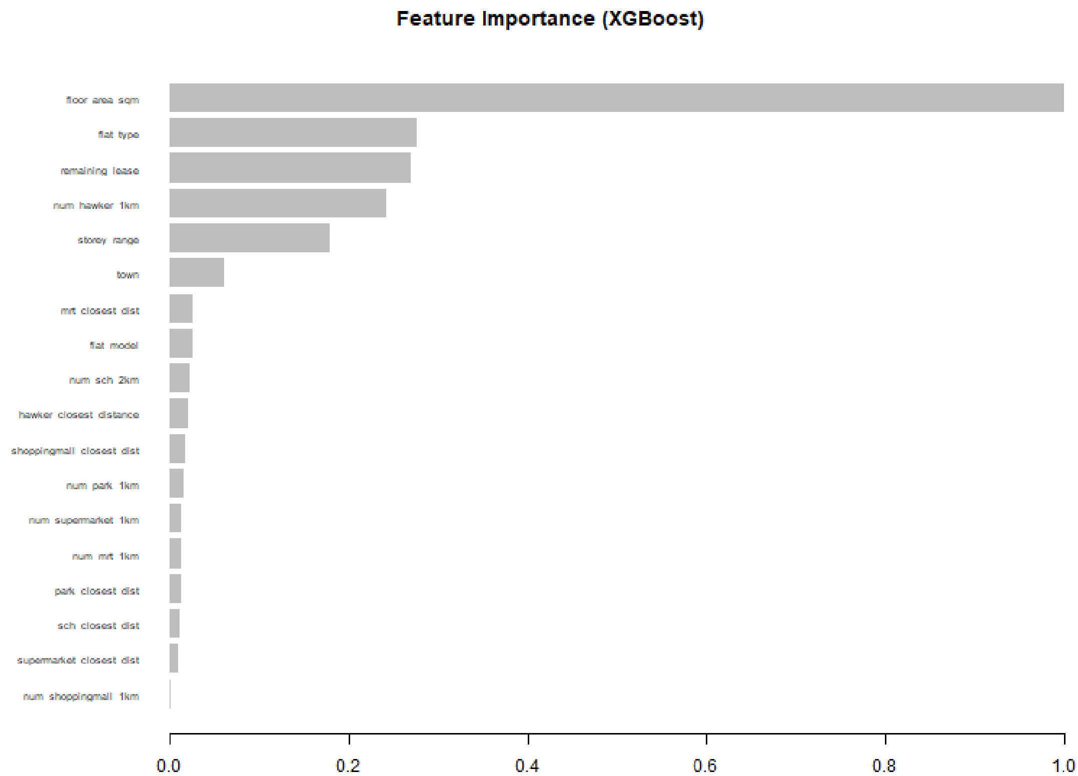


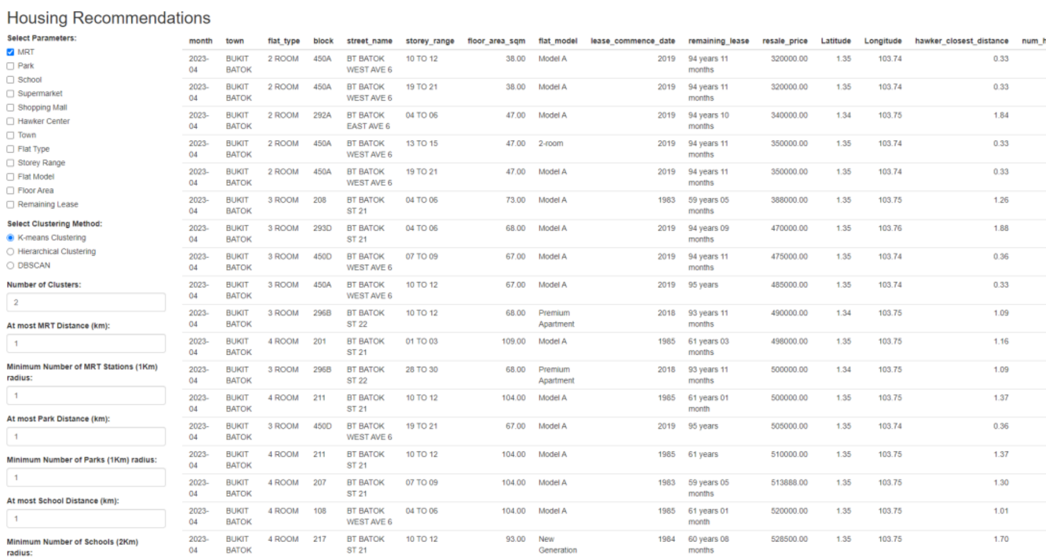
Figure 4.7: XGBoost feature importance

Figure 4.6 Random Forest reveals the model's priority on 'floor area', 'flat type', and 'town' in influencing resale prices, highlighting the critical roles of unit size, type, and location. Conversely, Figure 4.7 shows XGBoost's emphasis on 'floor area', 'flat type', and the unique 'remaining lease', pointing to the lease duration's unexpected significance. Despite these differences, both models concur on the paramountcy of 'floor area' and 'flat type'. This consistency across two distinct models solidifies the status of these features as reliable, universal predictors of resale value, regardless of the analytical method.

4.4 Shiny app

4.4.1 Unsupervised Recommendation System

The Shiny application, designed as a comprehensive tool for housing in Singapore, encompasses both an unsupervised recommendation system and a supervised predictive model for housing resale prices. The unsupervised recommendation system offers users a dynamic interface, allowing them to tailor their housing searches based on specific criteria, and subsequently receive a curated list of recommendations.



month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	remaining_lease	resale_price	Latitude	Longitude	hawker_closest_distance	num_h
2023-04	BUKIT BATOK	2 ROOM	450A	BT BATOK WEST AVE 6	10 TO 12	38.00	Model A	2019	94 years 11 months	320000.00	1.35	103.74	0.33	
2023-04	BUKIT BATOK	2 ROOM	450A	BT BATOK WEST AVE 6	19 TO 21	38.00	Model A	2019	94 years 11 months	320000.00	1.35	103.74	0.33	
2023-04	BUKIT BATOK	2 ROOM	292A	BT BATOK EAST AVE 6	04 TO 06	47.00	Model A	2019	94 years 10 months	340000.00	1.34	103.75	1.84	
2023-04	BUKIT BATOK	2 ROOM	450A	BT BATOK WEST AVE 6	13 TO 15	47.00	2-room	2019	94 years 11 months	350000.00	1.35	103.74	0.33	
2023-04	BUKIT BATOK	2 ROOM	450A	BT BATOK WEST AVE 6	19 TO 21	47.00	Model A	2019	94 years 11 months	350000.00	1.35	103.74	0.33	
2023-04	BUKIT BATOK	3 ROOM	208	BT BATOK ST 21	04 TO 06	73.00	Model A	1983	59 years 05 months	388000.00	1.35	103.75	1.26	
2023-04	BUKIT BATOK	3 ROOM	293D	BT BATOK ST 21	04 TO 06	68.00	Model A	2019	94 years 09 months	470000.00	1.35	103.76	1.88	
2023-04	BUKIT BATOK	3 ROOM	450D	BT BATOK WEST AVE 6	07 TO 09	67.00	Model A	2019	94 years 11 months	475000.00	1.35	103.74	0.36	
2023-04	BUKIT BATOK	3 ROOM	450A	BT BATOK WEST AVE 6	10 TO 12	67.00	Model A	2019	95 years	485000.00	1.35	103.74	0.33	
2023-04	BUKIT BATOK	3 ROOM	296B	BT BATOK ST 22	10 TO 12	68.00	Premium Apartment	2018	93 years 11 months	490000.00	1.34	103.75	1.09	
2023-04	BUKIT BATOK	4 ROOM	201	BT BATOK ST 21	01 TO 03	109.00	Model A	1985	61 years 03 months	498000.00	1.35	103.75	1.16	
2023-04	BUKIT BATOK	3 ROOM	296B	BT BATOK ST 22	28 TO 30	68.00	Premium Apartment	2018	93 years 11 months	500000.00	1.34	103.75	1.09	
2023-04	BUKIT BATOK	4 ROOM	211	BT BATOK ST 21	10 TO 12	104.00	Model A	1985	61 years 01 month	500000.00	1.35	103.75	1.37	
2023-04	BUKIT BATOK	3 ROOM	450D	BT BATOK WEST AVE 6	19 TO 21	67.00	Model A	2019	95 years	505000.00	1.35	103.74	0.36	
2023-04	BUKIT BATOK	4 ROOM	211	BT BATOK ST 21	10 TO 12	104.00	Model A	1985	61 years	510000.00	1.35	103.75	1.37	
2023-04	BUKIT BATOK	4 ROOM	207	BT BATOK ST 21	07 TO 09	104.00	Model A	1983	59 years 05 months	513888.00	1.35	103.75	1.30	
2023-04	BUKIT BATOK	4 ROOM	108	BT BATOK WEST AVE 6	04 TO 06	104.00	Model A	1985	61 years 01 month	520000.00	1.35	103.75	1.01	
2023-04	BUKIT BATOK	4 ROOM	217	BT BATOK ST 21	10 TO 12	93.00	New Generation	1984	60 years 08 months	528500.00	1.35	103.75	1.70	

Figure 4.8: Housing recommendation part 1

The development of an interactive recommendation system, as depicted in Figure 4.8, provides users with a sophisticated and user-friendly interface to customize their housing search based on specific criteria.

The system offers flexibility, allowing users to modify various parameters. They can fine-tune the clustering methods, determine the number of clusters, and even select the basis for clustering criteria.

Post customization, the interface presents a curated list of the top 45 housing recommendations on the right side of the shiny app. This list is tailored based on the user's selected parameters, ensuring that the recommendations align closely with their preferences.

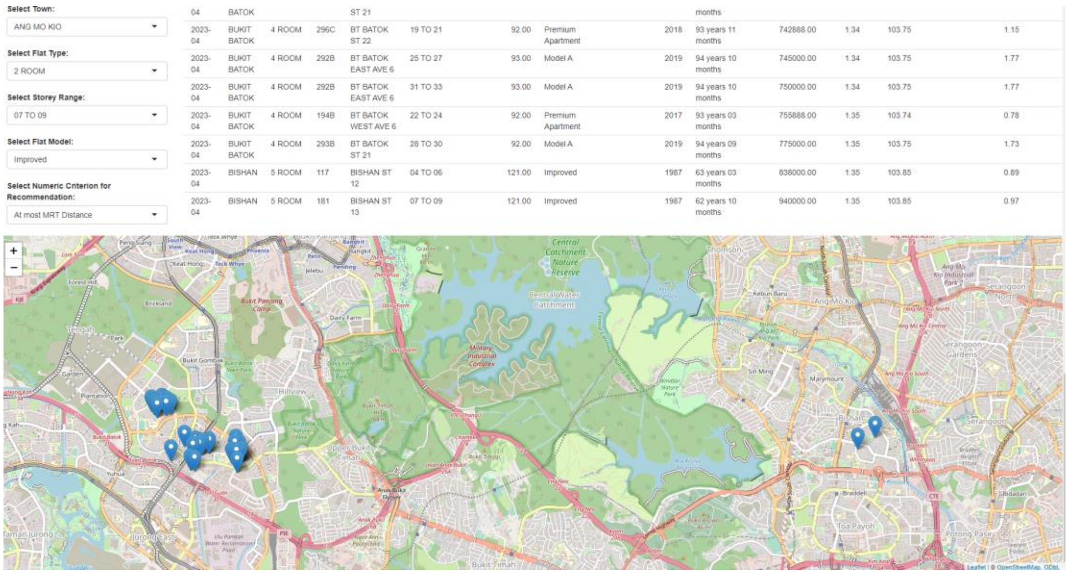


Figure 4.9: Housing recommendation part 2

Figure 4.9 displays an integrated map of Singapore at the bottom of the shiny app to enhance user experience and spatial understanding. This visual aid allows users to geographically situate the recommended housing units, aiding in decision-making by providing a spatial context.

This interactive recommendation system not only streamlines the housing search process but also empowers users by offering them control over the search criteria, ensuring a more personalized and efficient search experience.

4.4.2 Supervised Predictive Housing resale price

On the other hand, the supervised predictive model provides insights into potential housing resale prices based on various influential factors. Together, these components present users with a holistic view, assisting them in making informed housing decisions in the Singaporean market.

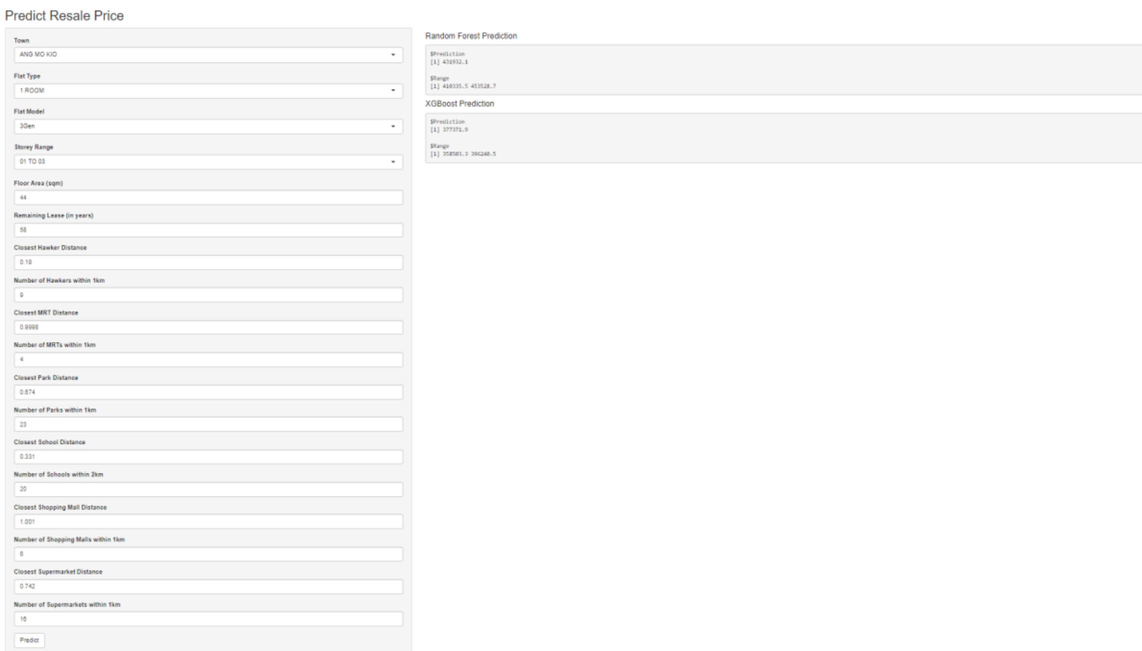


Figure 4.10: Predicting housing resale price

Within the Shiny application, users have the flexibility to modify parameters via the interface on the left side. Upon selecting their desired criteria and clicking the "Predict" option, the application promptly displays predictions from both the Random Forest and XGBoost models. Additionally, to offer a broader perspective on the predicted housing resale prices, the application provides the interquartile range, showcasing the Q1 to Q3 range of the predictions, as illustrated in Figure 4.10.

5.0 Discussion

The analysis presented in the results section underscores the nuanced and multifaceted nature of the housing landscape in Singapore. This discussion seeks to delve deeper into these findings, drawing connections, implications, and potential future directions.

5.1 Synthesizing Model Outcomes

The performance metrics, chiefly the RMSE and MAPE, have validated the selection of Random Forest and XGBoost as pivotal models for this analysis. Their superiority in predicting resale prices can be attributed to their inherent capabilities in handling non-linearities and intricate interactions between variables.

However, beyond mere prediction, the feature importance derived from both models offers strategic insights. The consistent emphasis on features like 'floor area' and 'flat type' across both models underscores their universal significance in influencing resale prices. This congruence across models reinforces the reliability and validity of these findings.

5.2 Spatial Distribution and Urban Dynamics

The preliminary analysis highlighted stark contrasts in amenity accessibility across towns. Such disparities might be rooted in historical urban planning decisions, demographic shifts, or evolving residential preferences. Understanding these spatial dynamics is pivotal for urban planners and policymakers, aiding them in future infrastructure developments or resource allocations.

5.3 Integrating Analysis with User Experience

The Shiny application, as presented, epitomizes the amalgamation of rigorous data analysis with user-centric design. While the unsupervised recommendation system offers users personalized housing suggestions, the supervised predictive model empowers them with resale price estimations. This dual approach not only facilitates informed decision-making but also democratizes data insights, making them accessible and actionable for the general populace.

5.4 Implications and Recommendations

The emphasis on 'floor area', 'flat type' suggests potential areas of focus for real estate developers or investors. For instance, properties with larger floor areas in desirable towns could command premium prices.

The Shiny application, in its current iteration, offers substantial utility to users. However, incorporating real-time data updates or expanding the range of amenities could enhance its predictive accuracy and recommendation relevance.

5.5 Limitations and Future Directions

While the models and analyses are robust, they are not devoid of limitations. Factors like historical price trends, imminent urban development projects, or socio-economic shifts were not incorporated into the current models but could influence resale prices.

Furthermore, evolving technologies, such as deep learning or neural networks, could be explored in future iterations to enhance predictive capabilities.

6.0 Conclusion

Characterized by a unique blend of spatial distribution and urban dynamics, Singapore's housing landscape offers a compelling field of study. Significant insights into the factors that influence the resale prices of housing units in the city-state have been gained through rigorous data analysis and modeling.

The adeptness of certain models, particularly Random Forest and XGBoost, in predicting resale prices has been proven. Not only have these models showcased impressive predictive capabilities, but key features, such as 'floor area' and 'flat type', have also been illuminated as being instrumental in determining resale prices.

Understanding has been further enriched by geospatial analysis, revealing the intricate balance of amenities across towns and emphasizing the importance of spatial context in urban studies.

A comprehensive platform to make informed housing decisions has been provided to users by offering both a recommendation system and a predictive tool.

It's crucial to acknowledge that, while many aspects of Singapore's housing market have been illuminated by our research, the urban fabric is ever-evolving. The factors influencing its housing market will continue to grow and change as the city does. For now, however, a significant step forward in understanding the current dynamics at play has been taken.

Looking ahead, immense potential exists to further refine our models, incorporate more variables, and leverage emerging technologies. Commitment to harnessing data for the benefit of its residents remains as the journey in understanding Singapore's urban landscape continues.

7.0 Recommendations

In light of the comprehensive analysis conducted and the tools developed, a structured approach is recommended for users and stakeholders to derive the most benefit. A detailed guide is provided here to ensure the utility of our findings and applications is maximized:

7.1 Structured Housing Exploration

7.1.1 Initial Exploration with Recommendation System

Begin your housing search by utilizing the recommendation system. This system, informed by historical data, offers a curated list of properties that align with user preferences. By starting here, users can gain insights into properties that have historically matched similar criteria and understand the sales patterns of such properties.

7.1.2 Informed Decision Making through the Resale Price Prediction Tool

After narrowing down potential housing options, transition to the resale price prediction tool. This tool provides a granular understanding of potential transaction values based on various influential factors. By leveraging this, users can make informed decisions, ensuring they get optimal value for their investments.

7.2 Continuous Data Updates and Training

The housing market is inherently fluid, with prices shaped by an ever-evolving array of factors. Given this dynamism, it is imperative to maintain an up-to-date dataset. By consistently integrating the most recent quarterly housing resale prices and retraining the models on this fresh data, predictions can be fine-tuned to anticipate the upcoming quarter's price trends more accurately. Stakeholders and developers of the application must recognize the significance of these periodic updates to ensure that the tool remains both relevant and reflective of the prevailing market conditions.

7.3 Diversify Data Sources

To further enhance the accuracy and relevance of predictions, consider incorporating additional data sources. Integrating macroeconomic indicators, upcoming infrastructural projects, or even socio-demographic trends can offer a more holistic view of the housing market and refine predictions.

In conclusion, the tools and insights provided aim to empower users in their housing decisions. By adhering to these recommendations, users can ensure they navigate Singapore's housing market with clarity, confidence, and a data-driven edge.

List of References

[1] T. Lau, "Singapore HDB Flat Resale Prices (1990-2020)," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/teyang/singapore-hdb-flat-resale-prices-19902020

[2] PropertyGuru. [Online]. Available: https://www.propertyguru.com.sg/

[3] T. Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations," Survey Review, vol. 23, no. 176, pp. 88-93, 1975.

[4] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[5] A. K. Jain, "Data clustering: 50 years beyond K-means," Pattern Recognition Letters, vol. 31, no. 8, pp. 651-666, 2010.

[6] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., vol. 1, no. 14, pp. 281-297, 1967.

[7] R. Sibson, "SLINK: an optimally efficient algorithm for the single-link cluster method," The Computer Journal, vol. 16, no. 1, pp. 30-34, 1973.

[8] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Proc. Kdd, vol. 96, pp. 226-231, 1996.

[9] C. M. Bishop, Pattern Recognition and Machine Learning. Springer, 2006.

[10] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

[11] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 785-794, 2016.

[12] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273-297, 1995.

[13] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," Annals of Statistics, pp. 1189-1232, 2001.

[14] R. Tibshirani, "Regression shrinkage and selection via the lasso," Journal of the Royal Statistical Society: Series B (Methodological), vol. 58, no. 1, pp. 267-288, 1996.

[15] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," Technometrics, vol. 12, no. 1, pp. 55-67, 1970.

[16] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," International Journal of Forecasting, vol. 22, no. 4, pp. 679-688, 2006.

[17] S. Makridakis, "Accuracy measures: theoretical and practical concerns," International Journal of Forecasting, vol. 9, no. 4, pp. 527-529, 1993.

[18] Data.Gov.sg, "Housing Resale Data," 2023. [Online]. Available: https://beta.data.gov.sg/datasets/189/resources/d_8b84c4ee58e3cfc0ece0d773c8ca6abc/view

[19] Data.Gov.sg, "School Data," 2022. [Online]. Available: https://beta.data.gov.sg/datasets/457/view

[20] Data.Gov.sg, "Hawker Centers," 2022. [Online]. Available: https://beta.data.gov.sg/datasets/1445/view

[21] Hxchua, "Train Stations in Singapore," Data.world, 2020. [Online]. Available: https://data.world/hxchua/train-stations-in-singapore

[22] Data.Gov.sg, "Supermarkets," 2020. [Online]. Available: https://beta.data.gov.sg/datasets/1565/view

[23] K. Gangula, "Shopping Mall Coordinates," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/karthikgangula/shopping-mall-coordinates

[24] Data.Gov.sg, "Parks," 2022. [Online]. Available: https://beta.data.gov.sg/datasets/1491/resources/d_0542d48f0991541706b58059381a6eca/view

[25] Ministry of Education, Singapore, "Primary One Registration: Distance," 2023. [Online]. Available: https://www.moe.gov.sg/primary/p1-registration/distance

Appendix A : Combining housing dataset with the 6 amenities

Figure A.1 illustrates the process of acquiring longitudes and latitudes for the original housing dataset. Subsequently, Figures A.2, A.6, and A.9 depict the codes used to obtain longitudes and latitudes for their respective amenities datasets. Finally, the rest of the figures present the codes implemented to calculate the number of amenities and identify the closest ones.

````
```{r adding longitude and latitude to original q2 2023}
library(ggmap)

# Set your Google Maps API key
register_google(key = "AIzaSyCmIgB23RKKNuz79bEBPVTodeMl7AK6OE0")

# Read addresses from the CSV file
address_data <- read.csv("data2023.csv")

# Convert the 'month' column to character
address_data$month <- as.character(address_data$month)

# Filter data for months 4, 5, and 6
filtered_data <- address_data[address_data$month >= "2023-04" & address_data$month <= "2023-06", ]

# Initialize empty vectors to store latitudes and longitudes
latitudes <- numeric()
longitudes <- numeric()

# Iterate through each row and geocode the address
for (i in seq_len(nrow(filtered_data))) {
  block <- filtered_data$block[i]
  street <- filtered_data$street[i]
  address <- paste(block, street)

  result <- geocode(address)
  if (!is.na(result$lat) && !is.na(result$lon)) {
    latitudes <- c(latitudes, result$lat)
    longitudes <- c(longitudes, result$lon)
  } else {
    latitudes <- c(latitudes, NA)
    longitudes <- c(longitudes, NA)
  }
}

# Add latitudes and longitudes to the filtered data
filtered_data$Latitude <- latitudes
filtered_data$Longitude <- longitudes

# Save the filtered data with latitudes and longitudes
write.csv(filtered_data, file = "filtered_geocoded_data.csv", row.names = FALSE)

cat("Geocoding and filtering completed. Filtered data saved to filtered_geocoded_data.csv.\n")


```
52:2  Chunk 2: adding longitude and latitude to original q2 2023  R Markdow
````

Figure A.1: Adding longitude and latitude to original Q2 2023

```
```{r getting parks}

library(sf)

# Read KML data
kml_data <- st_read("Parks.kml")

# Extract coordinates
coordinates <- st_coordinates(kml_data)

# Optional: Save coordinates to a CSV file
write.csv(coordinates, file = "parks.csv", row.names = FALSE)

cat("Coordinates extracted and saved to coordinates.csv.\n")


```


```{r getting hawker}
library(sf)

# Read KML data
kml_data <- st_read("HawkerCentresKML.kml")

# Extract coordinates
coordinates <- st_coordinates(kml_data)

# Optional: Save coordinates to a CSV file
write.csv(coordinates, file = "Hawker.csv", row.names = FALSE)

cat("Coordinates extracted and saved to coordinates.csv.\n")
```
```

Figure A.2: Getting parks and hawkers longitude and latitudes

```
```{r adding hawker}
# Load necessary libraries
library(dplyr)
library(geosphere)

# Read the CSV data
data <- read.csv("filtered_geocoded_data.csv")  # Replace 'your_data.csv' with your actual data file name
hawker_data <- read.csv("Hawker.csv")  # Replace 'hawker_data.csv' with the Hawker data file name

# Remove rows with negative longitude and latitude
data <- data %>%
  filter(Longitude >= 0, Latitude >= 0)

# Extract Hawker coordinates
hawker_coords <- hawker_data[, c("X", "Y")]

# Create a new column for closest distance
data <- data %>%
  mutate(hawker_closest_distance = 0.0,
         num_hawker_1km = 0)

for (i in 1:nrow(data)) {
  flat_coords <- c(data[i, "Longitude"], data[i, "Latitude"])
  distances <- distVincentySphere(flat_coords, hawker_coords)
  min_distance <- min(distances) / 1000  # Convert meters to kilometers
  data[i, "hawker_closest_distance"] <- min_distance

  # Count Hawker locations within 1km
  num_hawker_within_1km <- sum(distances <= 1000)  # Check distances <= 1000 meters (1km)
  data[i, "num_hawker_1km"] <- num_hawker_within_1km
}

# Save the updated data to a new CSV file
write.csv(data, "updated_data.csv", row.names = FALSE)
```
```

Figure A.3: Adding number of hawkers within 1km radius , and the closest hawker distance

```
```{r adding mrt}
# Load necessary libraries
library(dplyr)
library(geosphere)

# Read the CSV data
data <- read.csv("updated_data.csv")  # Replace with the actual updated data file name
mrt_data <- read.csv("mrt.csv")  # Replace with the MRT data file name

# Extract MRT coordinates
mrt_coords <- mrt_data[, c("Longitude", "Latitude")]

# Create new columns for closest MRT distance and number of MRT locations within 2km
data <- data %>%
  mutate(mrt_closest_dist = 0.0,
         num_mrt_1km = 0)  # Initialize the new columns

# Calculate closest MRT distance and count MRT locations within 1km
for (i in 1:nrow(data)) {
  flat_coords <- c(data[i, "Longitude"], data[i, "Latitude"])
  distances <- distVincentySphere(flat_coords, mrt_coords)
  min_distance <- min(distances) / 1000  # Convert meters to kilometers
  data[i, "mrt_closest_dist"] <- min_distance

  # Count MRT locations within 2km
  num_mrt_within_1km <- sum(distances <= 1000)  # Check distances <= 1000 meters (1km)
  data[i, "num_mrt_1km"] <- num_mrt_within_1km
}

# Save the updated data to a new CSV file
write.csv(data, "updated_data.csv", row.names = FALSE)

```
```

Figure A.4: Adding number of mrt within 1km radius , and the closest mrt distance

```
```{r adding park}
# Load necessary libraries
library(dplyr)
library(geosphere)

# Read the CSV data
data <- read.csv("updated_data.csv")  # Replace with the actual updated data file name
park_data <- read.csv("park.csv")  # Replace with the park data file name

# Extract park coordinates
park_coords <- park_data[, c("X", "Y")]

# Create new columns for closest park distance and number of park locations within 2km
data <- data %>%
  mutate(park_closest_dist = 0.0,
         num_park_1km = 0)  # Initialize the new columns

# Calculate closest park distance and count park locations within 1km
for (i in 1:nrow(data)) {
  flat_coords <- c(data[i, "Longitude"], data[i, "Latitude"])
  distances <- distVincentySphere(flat_coords, park_coords)
  min_distance <- min(distances) / 1000  # Convert meters to kilometers
  data[i, "park_closest_dist"] <- min_distance

  # Count park locations within 2km
  num_park_within_1km <- sum(distances <= 1000)  # Check distances <= 1000 meters (1km)
  data[i, "num_park_1km"] <- num_park_within_1km
}

# Save the updated data to a new CSV file
write.csv(data, "updated_data.csv", row.names = FALSE)

```
```

Figure A.5: Adding number of park within 1km radius , and the closest park distance

````r
```{r adding coordinates to school}
# Load necessary libraries
library(ggmap)

# Set your Google Maps API key
register_google(key = "AIzaSyCmIgB23RKKNuz79bEBPVTodeMl7AK6OE0")

# Read the CSV data
school_data <- read.csv("school.csv")

# Define a function to geocode postal codes to latitude and longitude using ggmap
geocode_postal <- function(postal_code) {
  geo_data <- geocode(postal_code)
  return(geo_data)
}

# Create new columns for latitude and longitude
school_data$Latitude <- NA
school_data$Longitude <- NA

# Iterate through each row and geocode the postal code
for (i in 1:nrow(school_data)) {
  postal_code <- as.character(school_data[i, "postal_code"])
  if (!is.na(postal_code)) {
    geo_data <- geocode_postal(postal_code)
    school_data[i, "Latitude"] <- geo_data$lat
    school_data[i, "Longitude"] <- geo_data$lon
  }
}

# Save the geocoded data to a new CSV file
write.csv(school_data, "school_with_coords.csv", row.names = FALSE)

cat("Geocoding completed. Geocoded data saved to school_with_coords.csv.\n")

```
````

Figure A.6: Adding longitude and latitude in the school dataset from the postal code

````r
```{r adding school}
# Load necessary libraries
library(dplyr)
library(geosphere)

# Read the CSV data
data <- read.csv("updated_data.csv")  # Replace with the actual updated data file name
school_data <- read.csv("school_with_coords.csv")  # Replace with the school data file name

# Extract school coordinates
school_coords <- na.omit(school_data[, c("Longitude", "Latitude")])

# Create new columns for closest school distance and number of school locations within 2km
data <- data %>%
  mutate(sch_closest_dist = 0.0,
         num_sch_2km = 0)  # Initialize the new columns

# Calculate closest school distance and count school locations within 2km
for (i in 1:nrow(data)) {
  flat_coords <- c(data[i, "Longitude"], data[i, "Latitude"])
  distances <- distVincentySphere(flat_coords, school_coords)
  min_distance <- min(distances) / 1000  # Convert meters to kilometers
  data[i, "sch_closest_dist"] <- min_distance

  # Count school locations within 2km
  num_sch_within_2km <- sum(distances <= 2000)  # Check distances <= 2000 meters (2km)
  data[i, "num_sch_2km"] <- num_sch_within_2km
}

# Save the updated data to a new CSV file
write.csv(data, "updated_data.csv", row.names = FALSE)

```
````

Figure A.7: Adding number of school within 2km radius , and the closest school distance

```
```{r adding shoppingmall}
# Load necessary libraries
library(dplyr)
library(geosphere)

# Read the CSV data
data <- read.csv("updated_data.csv")  # Replace with the actual updated data file name
shoppingmall_data <- read.csv("shoppingmall.csv")  # Replace with the shopping mall data file name

# Extract shopping mall coordinates and drop rows with NA values
shoppingmall_coords <- na.omit(shoppingmall_data[, c("LONGITUDE", "LATITUDE")])

# Create new columns for closest shopping mall distance and number of shopping malls within 2km
data <- data %>%
  mutate(shoppingmall_closest_dist = 0.0,
         num_shoppingmall_1km = 0)  # Initialize the new columns

# Calculate closest shopping mall distance and count shopping malls within 1km
for (i in 1:nrow(data)) {
  flat_coords <- c(data[i, "Longitude"], data[i, "Latitude"])
  distances <- distVincentySphere(flat_coords, shoppingmall_coords)
  min_distance <- min(distances) / 1000  # Convert meters to kilometers
  data[i, "shoppingmall_closest_dist"] <- min_distance

  # Count shopping malls within 2km
  num_shoppingmall_within_1km <- sum(distances <= 1000)  # Check distances <= 1000 meters (2km)
  data[i, "num_shoppingmall_1km"] <- num_shoppingmall_within_1km
}

# Save the updated data to a new CSV file
write.csv(data, "updated_data.csv", row.names = FALSE)

cat("Processing completed. Updated data saved to updated_data.csv.\n")

```
```

Figure A.8: Adding number of shopping mall within 1km radius , and the closest shopping mall distance

```
```{r adding coordinates to supermarket}
# Read the CSV data
supermarket_data <- read.csv("supermarket.csv")  # Replace with the actual supermarket data file name

# Extract postal codes using regular expressions
postal_codes <- gsub(".*S\\((\\d+)\\).*", "\\1", supermarket_data$premise_address)

# Create a new column for postal codes in the supermarket data
supermarket_data$PostalCode <- postal_codes

# Set your Google Maps API key
register_google(key = "AIzaSyCmIgB23RKKNuz79bEBPVTodeMl7AK6OEO")

# Define a function to geocode postal codes to latitude and longitude using ggmap
geocode_postal <- function(postal_code) {
  geo_data <- geocode(postal_code)
  return(geo_data)
}

# Create new columns for latitude and longitude
supermarket_data$Latitude <- NA
supermarket_data$Longitude <- NA

# Iterate through each row and geocode the postal code
for (i in 1:nrow(supermarket_data)) {
  postal_code <- as.character(supermarket_data[i, "PostalCode"])
  if (!is.na(postal_code)) {
    geo_data <- geocode_postal(postal_code)
    supermarket_data[i, "Latitude"] <- geo_data$lat
    supermarket_data[i, "Longitude"] <- geo_data$lon
  }
}

# Save the updated supermarket data to a new CSV file
write.csv(supermarket_data, "supermarket_with_coords.csv", row.names = FALSE)

cat("Processing completed. Supermarket data with postal codes saved to supermarket_with_postal.csv.\n")

```
```

Figure A.9: Adding longitude and latitude in the supermarket dataset from the postal code

```r
```{r adding supermarket}
# Load necessary libraries
library(dplyr)
library(geosphere)

# Read the CSV data
data <- read.csv("updated_data.csv")  # Replace with the actual updated data file name
supermarket_data <- read.csv("supermarket_with_coords.csv")  # Replace with the supermarket data file name

# Extract supermarket coordinates and drop rows with NA values
supermarket_coords <- na.omit(supermarket_data[, c("Longitude", "Latitude")])

# Create new columns for closest supermarket distance and number of supermarkets within 2km
data <- data %>%
  mutate(supermarket_closest_dist = 0.0,
         num_supermarket_1km = 0)  # Initialize the new columns

# Calculate closest supermarket distance and count supermarkets within 2km
for (i in 1:nrow(data)) {
  flat_coords <- c(data[i, "Longitude"], data[i, "Latitude"])
  distances <- distVincentySphere(flat_coords, supermarket_coords)
  min_distance <- min(distances) / 1000  # Convert meters to kilometers
  data[i, "supermarket_closest_dist"] <- min_distance

  # Count supermarkets within 1km
  num_supermarket_within_1km <- sum(distances <= 1000)  # Check distances <= 1000 meters (1km)
  data[i, "num_supermarket_1km"] <- num_supermarket_within_1km
}

# Save the updated data to a new CSV file
write.csv(data, "updated_data.csv", row.names = FALSE)

cat("Processing completed. Updated data saved to updated_data.csv.\n")

```
```

Figure A.10: Adding number of supermarket within 1km radius , and closest supermarket distance

Appendix B: Preliminary analysis of the combined dataset

Figure B.1 pertains to the data cleaning process, focusing on the elimination of outliers and the selection of meaningful columns for subsequent use. Figures B.2 and B.3 are dedicated to performing statistical analysis and generating visual plots, respectively, analyzing the housing dataset grouped by town.

````r
```{r data cleaning}
# Read data
library(tidyverse)
data <- read_csv('updated_data.csv',show_col_types = FALSE)
# Data cleaning:
# 1. Drop specified columns
# 2. Remove rows where flat_model is "2-room"
# 3. Remove rows where distances are greater than 4km for all towns
cleaned_data <- data %>%
  select(-block, -street_name, -month, -lease_commence_date, -Latitude, -Longitude) %>%
  filter(flat_model != "2-room") %>%
  filter(hawker_closest_distance <= 4 &
         mrt_closest_dist <= 4 &
         park_closest_dist <= 4 &
         sch_closest_dist <= 4 &
         shoppingmall_closest_dist <= 4 &
         supermarket_closest_dist <= 4)

# Save the cleaned data to a new CSV file
write.csv(cleaned_data, "final.csv", row.names = FALSE)
```
````

Figure B.1: Data cleaning to include the relevant columns for future data analysis

```r
```{r preliminary analysis}
library(dplyr)
library(tidyr)
library(ggplot2)
library(grid)
library(gridExtra)

# Read data
data <- read.csv("final.csv")

# Create a function to obtain the summary statistics
summary_stats <- function(column) {
  c(Min = min(column, na.rm = TRUE),
    Median = median(column, na.rm = TRUE),
    Mean = mean(column, na.rm = TRUE),
    Max = max(column, na.rm = TRUE),
    '1st Qu.' = quantile(column, 0.25, na.rm = TRUE),
    '3rd Qu.' = quantile(column, 0.75, na.rm = TRUE))
}

# Apply the function to each column of interest
summaries <- data.frame(
  t(sapply(data[, c("hawker_closest_distance",
                   "num_hawker_1km",
                   "mrt_closest_dist",
                   "num_mrt_1km",
                   "park_closest_dist",
                   "num_park_1km",
                   "sch_closest_dist",
                   "num_sch_2km",
                   "shoppingmall_closest_dist",
                   "num_shoppingmall_1km",
                   "supermarket_closest_dist",
                   "num_supermarket_1km")], summary_stats))
)

# View the summary table
print(summaries)

# Convert the data frame to a grid table
grid_table <- tableGrob(summaries)

# Save the table as an image
ggsave(filename = "summary_table.png", plot = grid_table, width = 20, height = 8)

```

Figure B.2: Preliminary analysis part 1

```r

# calculate the average distances by town in meters for all amenities
avg_dist_by_town <- data %>%
  group_by(town) %>%
  summarise(
    avg_dist_hawker = mean(hawker_closest_distance, na.rm = TRUE),
    avg_dist_mrt = mean(mrt_closest_dist, na.rm = TRUE),
    avg_dist_park = mean(park_closest_dist, na.rm = TRUE),
    avg_dist_sch = mean(sch_closest_dist, na.rm = TRUE),
    avg_dist_shoppingmall = mean(shoppingmall_closest_dist, na.rm = TRUE),
    avg_dist_supermarket = mean(supermarket_closest_dist, na.rm = TRUE)
  )

# convert data to long format for plotting
avg_dist_long <- avg_dist_by_town %>%
  pivot_longer(cols = starts_with("avg_dist"),
               names_to = "amenity",
               values_to = "avg_distance") %>%
  mutate(amenity = recode(amenity,
                          avg_dist_hawker = "Hawker Center",
                          avg_dist_mrt = "MRT Station",
                          avg_dist_park = "Park",
                          avg_dist_sch = "School",
                          avg_dist_shoppingmall = "Shopping Mall",
                          avg_dist_supermarket = "Supermarket"))

g <- ggplot(avg_dist_long, aes(x = town, y = avg_distance, fill = amenity)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.7) +  # Adjust bar width and dodge width
  labs(
    title = "Average Distance to Amenities by Town",
    y = "Average Distance (in km)",
    x = "Town",
    fill = "Amenity"
  ) +
  coord_flip() +
  theme(axis.text.x = element_text(angle = 0))

ggsave("avg_distance_plot_spaced_amenities.png", plot = g, width = 10, height = 12)  # Adjusted height to accommodate spacing

# calculate the average number of facilities by town for all amenities
avg_num_by_town <- data %>%
  group_by(town) %>%
  summarise(
    avg_num_hawker = mean(num_hawker_1km, na.rm = TRUE),
    avg_num_mrt = mean(num_mrt_1km, na.rm = TRUE),
    avg_num_park = mean(num_park_1km, na.rm = TRUE),
    avg_num_sch = mean(num_sch_2km, na.rm = TRUE),  # Note that schools use a 2km radius
    avg_num_shoppingmall = mean(num_shoppingmall_1km, na.rm = TRUE),
    avg_num_supermarket = mean(num_supermarket_1km, na.rm = TRUE)
  )

# convert data to long format for plotting
avg_num_long <- avg_num_by_town %>%
  pivot_longer(cols = starts_with("avg_num"),
               names_to = "amenity",
               values_to = "avg_number") %>%
  mutate(amenity = recode(amenity,
                          avg_num_hawker = "Hawker Center",
                          avg_num_mrt = "MRT Station",
                          avg_num_park = "Park",
                          avg_num_sch = "School",
                          avg_num_shoppingmall = "Shopping Mall",
                          avg_num_supermarket = "Supermarket"))

# create the plot with increased space between bars
g <- ggplot(avg_num_long, aes(x = town, y = avg_number, fill = amenity)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.7) +
  labs(
    title = "Average Number of Facilities by Town",
    y = "Average Number of Facilities",
    x = "Town",
    fill = "Amenity"
  ) +
  coord_flip() +
  theme(axis.text.x = element_text(angle = 0))

ggsave("avg_number_plot_spaced_amenities.png", plot = g, width = 10, height = 12)  # Adjusted height for spacing

```
```

Figure B.3: Preliminary analysis part 2

Appendix C: unsupervised and supervised analysis of the combined dataset

Figures C.1 to C.6 depict the code for unsupervised machine learning in the shiny app, while Figures C.7 to C.11 represent supervised machine learning code for the app.

```
```{r with map }
library(shiny)
library(caret)  # For one-hot encoding
library(tidyverse)
library(fpc)
library(leaflet)  # Added for map integration

# Read data
data <- read.csv("updated_data.csv")

# Extract years
data$years <- as.numeric(gsub(" years.*", "", data$remaining_lease))

# Extract months
data$months <- 0
data$months[grep("month", data$remaining_lease)] <- as.numeric(gsub(".*years ", "", gsub(" month.*", "", data$remaining_lease[grep("month", data$remaining_lease)])))


ui <- fluidPage(
  titlePanel("Housing Recommendations"),

  # Row for Selection and Map
  fluidRow(
    # Left Panel for Selection
    column(2,
      checkboxGroupInput("selected_parameters", "Select Parameters:",
                         choices = c("MRT", "Park", "School", "Supermarket", "Shopping Mall", "Hawker Center", "Town", "Flat Type", "Storey Range", "Flat Model", "Floor
Area", "Remaining Lease"),
                         selected = "MRT"),

      radioButtons("clustering_method", "Select Clustering Method:",
                   choices = c("K-means Clustering", "Hierarchical Clustering", "DBSCAN"),
                   selected = "K-means Clustering"),

    numericInput("k_clusters", "Number of Clusters:", value = 2, min = 1, max = 10000, step = 1),

      conditionalPanel(
        condition = "input.clustering_method == 'DBSCAN'",
        numericInput("dbscan_eps", "EPS (Max Distance):", value = 0.5, min = 0, max = 10, step = 0.1),
        numericInput("dbscan_minPts", "Min Points:", value = 5, min = 1, max = 100, step = 1)
      ),
      # Numeric inputs for distances
      conditionalPanel(
        condition = "'MRT' %in% input.selected_parameters",
        numericInput("mrt_distance", "At most MRT Distance (km):", value = 1, min = 0, max = 10, step = 0.1),
        numericInput("min_num_mrts", "Minimum Number of MRT Stations (1Km) radius:", value = 1, min = 0, max = 10, step = 1)
      ),
      conditionalPanel(
        condition = "'Park' %in% input.selected_parameters",
        numericInput("park_distance", "At most Park Distance (km):", value = 1, min = 0, max = 10, step = 0.1),
        numericInput("min_num_parks", "Minimum Number of Parks (1Km) radius:", value = 1, min = 0, max = 10, step = 1)
      ),
      conditionalPanel(
        condition = "'School' %in% input.selected_parameters",
        numericInput("school_distance", "At most School Distance (km):", value = 1, min = 0, max = 10, step = 0.1),
        numericInput("min_num_schools", "Minimum Number of Schools (2Km) radius:", value = 1, min = 0, max = 10, step = 1)
      ),
      conditionalPanel(
        condition = "'Supermarket' %in% input.selected_parameters",
        numericInput("supermarket_distance", "At most Supermarket Distance (km):", value = 1, min = 0, max = 10, step = 0.1),
        numericInput("min_num_supermarkets", "Minimum Number of Supermarkets (1Km) radius:", value = 1, min = 0, max = 10, step = 1)
      ),
```

Figure C.1: unsupervised recommendation system shiny app part 1
```

```r
    ),
    conditionalPanel(
      condition = "'Shopping Mall' %in% input.selected_parameters",
      numericInput("shoppingmall_distance", "At most Shopping Mall Distance (km):", value = 1, min = 0, max = 10, step = 0.1),
      numericInput("min_num_shoppingmalls", "Minimum Number of Shopping Malls (1Km) radius:", value = 1, min = 0, max = 10, step = 1)
    ),
    conditionalPanel(
      condition = "'Hawker Center' %in% input.selected_parameters",
      numericInput("hawker_distance", "At most Hawker Center Distance (km):", value = 1, min = 0, max = 10, step = 0.1),
      numericInput("min_num_hawkers", "Minimum Number of Hawker Centers (1Km) radius:", value = 1, min = 0, max = 10, step = 1)
    ),

    # Price filter
    checkboxInput("apply_price_filter", "Apply Price Filter", value = FALSE),
    conditionalPanel(
      condition = "input.apply_price_filter == true",
      numericInput("min_price", "Minimum Price:", value = 200000, min = 0, max = 2000000, step = 10000),
      numericInput("max_price", "Maximum Price:", value = 1000000, min = 0, max = 2000000, step = 10000)
    ),

 # Conditional input for floor_area_sqm based on checkbox
    conditionalPanel(
      condition = "'Floor Area' %in% input.selected_parameters",
      sliderInput("floor_area_sqm_range", "Floor Area (sqm):", min = min(data$floor_area_sqm), max = max(data$floor_area_sqm), value = c(min(data$floor_area_sqm),
max(data$floor_area_sqm)))
    ),

    # Conditional inputs for remaining_lease based on checkbox
    conditionalPanel(
      condition = "'Remaining Lease' %in% input.selected_parameters",
      numericInput("remaining_lease_years", "Remaining Lease (years):", value = max(data$years), min = min(data$years), max = max(data$years)),
      numericInput("remaining_lease_months", "Remaining Lease (months):", value = 0, min = 0, max = 11)
    ),

    # Dropdowns for categorical features
    conditionalPanel(
      condition = "'Town' %in% input.selected_parameters",
      selectInput("town", "Select Town:", choices = unique(data$town))
    ),
    conditionalPanel(
      condition = "'Flat Type' %in% input.selected_parameters",
      selectInput("flat_type", "Select Flat Type:", choices = unique(data$flat_type))
    ),
    conditionalPanel(
      condition = "'Storey Range' %in% input.selected_parameters",
      selectInput("storey_range", "Select Storey Range:", choices = unique(data$storey_range))
    ),
    conditionalPanel(
      condition = "'Flat Model' %in% input.selected_parameters",
      selectInput("flat_model", "Select Flat Model:", choices = unique(data$flat_model))
    ),

  condition = "'MRT' %in% input.selected_parameters || 'Park' %in% input.selected_parameters || 'School' %in% input.selected_parameters || 'Supermarket' %in%
input.selected_parameters || 'Shopping Mall' %in% input.selected_parameters || 'Hawker Center' %in% input.selected_parameters",
selectInput("numeric_criterion", "Select Numeric Criterion for Recommendation:",
            choices = c("At most MRT Distance" = "mrt_closest_dist",
                        "Minimum Number of MRT Stations" = "num_mrt_1km",
                        "At most Park Distance" = "park_closest_dist",
                        "Minimum Number of Parks" = "num_park_1km",
                        "At most School Distance" = "sch_closest_dist",
                        "Minimum Number of Schools" = "num_sch_2km",
                        "At most Supermarket Distance" = "supermarket_closest_dist",
```

Figure C.2: unsupervised recommendation system shiny app part 2

```r
                              "Minimum Number of Parks" = "num_park_1km",
                              "At most School Distance" = "sch_closest_dist",
                              "Minimum Number of Schools" = "num_sch_2km",
                              "At most Supermarket Distance" = "supermarket_closest_dist",
                              "Minimum Number of Supermarkets" = "num_supermarket_1km",
                              "At most Shopping Mall Distance" = "shoppingmall_closest_dist",
                              "Minimum Number of Shopping Malls" = "num_shoppingmall_1km",
                              "At most Hawker Center Distance" = "hawker_closest_distance",
                              "Minimum Number of Hawker Centers" = "num_hawker_1km",
                              "Minimum Mean Resale Price" = "resale_price_mean"),
                  selected = "mrt_closest_dist")
    ),

      # Right Panel for Map
      column(10,
        tableOutput("filtered_data")
      ),

    # Below Panel for Recommendations
    fluidRow(
      column(12,
        leafletOutput("map", height = "600px")
      )
    )
  )
  )

server <- function(input, output, session) {
  recommendation_data <- reactive({
    filtered_data <- data

    # Filter based on selected parameters and their corresponding inputs
    if ("MRT" %in% input$selected_parameters) {
      filtered_data <- filtered_data[filtered_data$mrt_closest_dist <= input$mrt_distance, ]
      if (nrow(filtered_data) == 0) return("No data after MRT filter")
    }

    if ("Park" %in% input$selected_parameters) {
      filtered_data <- filtered_data[filtered_data$park_closest_dist <= input$park_distance, ]
      if (nrow(filtered_data) == 0) return("No data after Park filter")
    }

    if ("School" %in% input$selected_parameters) {
      filtered_data <- filtered_data[filtered_data$sch_closest_dist <= input$school_distance, ]
      if (nrow(filtered_data) == 0) return("No data after School filter")
    }

    if ("Supermarket" %in% input$selected_parameters) {
      filtered_data <- filtered_data[filtered_data$supermarket_closest_dist <= input$supermarket_distance, ]
      if (nrow(filtered_data) == 0) return("No data after Supermarket filter")
    }

    if ("Shopping Mall" %in% input$selected_parameters) {
      filtered_data <- filtered_data[filtered_data$shoppingmall_closest_dist <= input$shoppingmall_distance, ]
      if (nrow(filtered_data) == 0) return("No data after Shopping Mall filter")
    }

    if ("Hawker Center" %in% input$selected_parameters) {
      filtered_data <- filtered_data[filtered_data$hawker_closest_dist <= input$hawker_distance, ]
      if (nrow(filtered_data) == 0) return("No data after Hawker Center filter")
    }

    # Filter by price range ONLY if "apply_price_filter" is TRUE
    if (input$apply_price_filter) {
```

Figure C.3: unsupervised recommendation system shiny app part 3

```r
if ("Shopping Mall" %in% input$selected_parameters && !is.null(input$min_num_shoppingmalls)) {
  filtered_data <- filtered_data[filtered_data$num_shoppingmall_1km >= input$min_num_shoppingmalls, ]
  if (nrow(filtered_data) == 0) return("No data after minimum shopping malls filter")
}

# Similarly add conditions for other amenities like schools
if ("School" %in% input$selected_parameters && !is.null(input$min_num_schools)) {
  filtered_data <- filtered_data[filtered_data$num_sch_2km >= input$min_num_schools, ]
  if (nrow(filtered_data) == 0) return("No data after minimum schools filter")
}

# ... Add conditions for additional amenities as needed

   # Define the function to determine if the criterion is based on minimizing or maximizing
    should_pick_min <- function(criterion) {
      min_based_criteria <- c("mrt_closest_dist", "park_closest_dist", "sch_closest_dist",
                              "supermarket_closest_dist", "shoppingmall_closest_dist",
                              "hawker_closest_distance", "resale_price_mean")
      return(criterion %in% min_based_criteria)
    }


# Mapping between user input and the actual dataset variable name
user_to_data_map <- list("Town" = "town",
                         "Flat Type" = "flat_type",
                         "Storey Range" = "storey_range",
                         "Flat Model" = "flat_model")

selected_features <- c("town", "flat_type", "storey_range", "flat_model")  # default features

# Check each feature; if it has only one unique value, exclude it from encoding
for (feature in selected_features) {
  if (length(unique(filtered_data[[feature]])) == 1) {
    selected_features <- setdiff(selected_features, feature)
  }
}

# Convert the vector of features to a formula
formula <- as.formula(paste("~", paste(selected_features, collapse = " + "), "- 1"))

# Encode the features
encoded_features <- model.matrix(formula, filtered_data)

    numeric_features <- filtered_data %>%
      dplyr::select(
        floor_area_sqm,
        resale_price,
        years,
        num_hawker_1km,
        mrt_closest_dist,
        hawker_closest_distance,
        shoppingmall_closest_dist,
        park_closest_dist,
        sch_closest_dist,
        supermarket_closest_dist,
        num_supermarket_1km,
        num_mrt_1km,
        num_sch_2km,
        num_park_1km,
        num_shoppingmall_1km,
        months
      )
    scaled_numeric_features <- scale(numeric_features)
    combined_features <- cbind(encoded_features, scaled_numeric_features)

```

Figure C.4: unsupervised recommendation system shiny app part 4

```r
    # Check for NA, NaN, or Inf values and remove corresponding rows
    combined_features <- combined_features[!apply(is.na(combined_features) | is.infinite(combined_features), 1, any),]

# Apply the clustering methods
    if(input$clustering_method == "K-means Clustering") {
      set.seed(123)
      clustering_model <- kmeans(combined_features, centers = input$k_clusters)

      # Add additional logic for the mean resale price
      if(input$numeric_criterion == "resale_price_mean") {
        cluster_means <- tapply(filtered_data$resale_price, clustering_model$cluster, mean)
      } else {
        cluster_means <- clustering_model$centers[, input$numeric_criterion]
      }

      # Decide the recommended cluster
      if (should_pick_min(input$numeric_criterion)) {
        recommended_cluster <- which.min(cluster_means)
      } else {
        recommended_cluster <- which.max(cluster_means)
      }

      recommendation <- filtered_data[clustering_model$cluster == recommended_cluster, ]
          # Limit the recommendation to the top 45 rows
    recommendation <- head(recommendation, 45)

    } else if(input$clustering_method == "Hierarchical Clustering") {
      hc <- hclust(dist(combined_features))
      cluster_assignments <- cutree(hc, k = input$k_clusters)

      cluster_means <- tapply(filtered_data[[input$numeric_criterion]], cluster_assignments, mean)

      if (should_pick_min(input$numeric_criterion)) {
        recommended_cluster <- which.min(cluster_means)
      } else {
        recommended_cluster <- which.max(cluster_means)
      }

      recommendation <- filtered_data[cluster_assignments == recommended_cluster, ]

          # Limit the recommendation to the top 45 rows
    recommendation <- head(recommendation, 45)

    } else {
      clustering_model <- dbscan(combined_features, eps = input$dbscan_eps, MinPts = input$dbscan_minPts)
      cluster_means <- tapply(filtered_data[[input$numeric_criterion]][clustering_model$cluster != -1],
                              clustering_model$cluster[clustering_model$cluster != -1],
                              mean)

      if (should_pick_min(input$numeric_criterion)) {
        recommended_cluster <- which.min(cluster_means)
      } else {
        recommended_cluster <- which.max(cluster_means)
      }

      recommendation <- filtered_data[clustering_model$cluster == recommended_cluster, ]
          # Limit the recommendation to the top 45 rows
    recommendation <- head(recommendation, 45)
    }

    recommendation <- recommendation[order(recommendation$resale_price), ]
    # Limit the recommendation to the top 45 rows
    recommendation <- head(recommendation, 45)

    if (nrow(recommendation) == 0) return("No recommendation available")
```

Figure C.5: unsupervised recommendation system shiny app part 5

```r
  })
output$filtered_data <- renderTable({
    recommendation <- recommendation_data()

    if (nrow(recommendation) == 0) {
        return("No recommendation available")
    }

    # Add a 'row_number' column to the start of your dataframe
    recommendation <- tibble::rowid_to_column(recommendation, "Row Number")

    return(recommendation)
})

  output$map <- renderLeaflet({
    recommendation <- recommendation_data()

    if (nrow(recommendation) == 0) {
      return(leaflet() %>% setView(lat = 1.3521, lng = 103.8198, zoom = 12))  # Default view of Singapore
    }

    m <- leaflet() %>%
      addTiles() %>%  # This adds the base map
      setView(lat = 1.3521, lng = 103.8198, zoom = 12)  # Centered on Singapore

    # Add a new column to 'recommendation' that represents the row number
    recommendation$row_number <- seq_len(nrow(recommendation))

    # Use the 'popup' argument to display the row number along with any other info you want
    popups <- sprintf(
      "<strong>Row number:</strong> %s<br/><strong>Resale Price:</strong> %s",
      recommendation$row_number,
      recommendation$resale_price  # Add other details as needed
    )

    m <- addMarkers(
      map = m,
      data = recommendation,
      lat = ~Latitude,
      lng = ~Longitude,
      popup = popups  # this argument accepts HTML content
    )

    return(m)
  })
}

shinyApp(ui, server)
```
```

Figure C.6: unsupervised recommendation system shiny app part 6
```

````
```{r training and testing model}
# Install and load necessary libraries
library(tidyverse)
library(lubridate)
library(caret)
library(randomForest)
library(xgboost)
library(e1071)
library(ggplot2)
library(gbm)
library(glmnet)
library(grid)

# Load the dataset
data <- read_csv('final.csv',show_col_types = FALSE)

# Convert categorical variables
data <- data %>%
  mutate_at(vars(town, flat_type, flat_model, storey_range), as.factor)

# Convert 'remaining_lease' into numeric (in terms of years)
data$remaining_lease <- sapply(strsplit(as.character(data$remaining_lease), " "), function(x) {
  years <- as.numeric(x[1])
  if(length(x) > 2) {
    months <- as.numeric(gsub("month.*", "", x[3])) / 12
    return(years + months)
  } else {
    return(years)
  }
})

# Split data into training and test sets
set.seed(123)
trainIndex <- createDataPartition(data$resale_price, p = 0.8, list = FALSE)
train_data <- data[trainIndex,]
test_data <- data[-trainIndex,]

# Train Random Forest
set.seed(123)
rf_model <- randomForest(resale_price ~ ., data=train_data, ntree=50)

# Train XGBoost
xgb_data <- train_data %>%
  mutate_if(is.character, as.numeric) %>%
  mutate_if(is.factor, as.numeric)
xgb_train <- xgb.DMatrix(data = as.matrix(xgb_data %>% select(-resale_price)), label = xgb_data$resale_price)
set.seed(123)
xgb_model <- xgboost(data = xgb_train, nrounds=100, objective="reg:squarederror")

# Evaluate models on test set
rf_predictions <- predict(rf_model, newdata = test_data)
xgb_predictions <- predict(xgb_model, newdata = as.matrix(test_data %>% select(-resale_price) %>% mutate_if(is.character, as.numeric) %>% mutate_if(is.factor, as.numeric)))

rf_rmse <- sqrt(mean((rf_predictions - test_data$resale_price)^2))
xgb_rmse <- sqrt(mean((xgb_predictions - test_data$resale_price)^2))

# SVM for regression
set.seed(123)
svm_model <- svm(resale_price ~ ., data=train_data, kernel="radial")
svm_predictions <- predict(svm_model, newdata = test_data)
svm_rmse <- sqrt(mean((svm_predictions - test_data$resale_price)^2))

# 1. Gradient Boosting Machine (GBM)
set.seed(123)
````

575:13 Chunk 5: training and testing model R Markdown

Figure C.7: Training and testing model part 1
````

```r
set.seed(123)
gbm_model <- gbm(resale_price ~ ., data=train_data, distribution="gaussian", n.trees=100)
gbm_predictions <- predict(gbm_model, newdata=test_data, n.trees=100)
gbm_rmse <- sqrt(mean((gbm_predictions - test_data$resale_price)^2))

# 2. LASSO Regression
# First, we need to create a model matrix
x_train <- model.matrix(resale_price ~ ., train_data)[,-1]
y_train <- train_data$resale_price
x_test <- model.matrix(resale_price ~ ., test_data)[,-1]
lasso_model <- cv.glmnet(x_train, y_train, alpha=1)
lasso_predictions <- predict(lasso_model, s=lasso_model$lambda.min, newx=x_test)
lasso_rmse <- sqrt(mean((lasso_predictions - test_data$resale_price)^2))

# 3. Ridge Regression
ridge_model <- cv.glmnet(x_train, y_train, alpha=0)
ridge_predictions <- predict(ridge_model, s=ridge_model$lambda.min, newx=x_test)
ridge_rmse <- sqrt(mean((ridge_predictions - test_data$resale_price)^2))

# Function to compute MAPE
calculate_mape <- function(actual, predicted) {
  return(mean(abs((actual - predicted) / actual)) * 100)
}

# Calculate MAPE for each model
rf_mape <- calculate_mape(test_data$resale_price, rf_predictions)
xgb_mape <- calculate_mape(test_data$resale_price, xgb_predictions)
svm_mape <- calculate_mape(test_data$resale_price, svm_predictions)
gbm_mape <- calculate_mape(test_data$resale_price, gbm_predictions)
lasso_mape <- calculate_mape(test_data$resale_price, lasso_predictions)
ridge_mape <- calculate_mape(test_data$resale_price, ridge_predictions)

# Add MAPE to the comparison dataframe
comparison_df <- tibble(
  Model = c("Random Forest", "XGBoost", "SVM", "GBM", "LASSO", "Ridge"),
  RMSE = c(rf_rmse, xgb_rmse, svm_rmse, gbm_rmse, lasso_rmse, ridge_rmse),
  MAPE = c(rf_mape, xgb_mape, svm_mape, gbm_mape, lasso_mape, ridge_mape)
)

print(comparison_df)
# Find the model with the lowest RMSE
best_model <- comparison_df[which.min(comparison_df$RMSE), ]

print(best_model)

# Plotting RMSE comparison
rmse_plot <- ggplot(comparison_df, aes(x = Model, y = RMSE)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  labs(title = "RMSE Comparison across Models", x = "Models", y = "RMSE") +
  theme_minimal() +
  theme(legend.position = "none")

# Save the RMSE plot
ggsave(filename = "rmse_comparison_plot.png", plot = rmse_plot, width = 10, height = 6, bg = "white")

# Plotting MAPE comparison
mape_plot <- ggplot(comparison_df, aes(x = Model, y = MAPE)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  labs(title = "MAPE Comparison across Models", x = "Models", y = "MAPE (%)") +
  theme_minimal() +
  theme(legend.position = "none")

# Save the MAPE plot
ggsave(filename = "mape_comparison_plot.png", plot = mape_plot, width = 10, height = 6, bg = "white")
```

655:66 Chunk 5: training and testing model R Markdown

Figure C.8: Training and testing model part 2

```r

# Extract feature importance
rf_importance_matrix <- importance(rf_model)

# Create a data frame for feature importance using IncNodePurity
rf_importance <- data.frame(Feature = rownames(rf_importance_matrix), Importance = rf_importance_matrix[, "IncNodePurity"])

# Sort the importance data in descending order for better visualization
rf_importance <- rf_importance %>%
  arrange(desc(Importance))

rf_plot <- ggplot(rf_importance, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  labs(title = "Feature Importance from Random Forest", x = "Features", y = "Importance (IncNodePurity)") +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 12))  # Increase text size

# Save the plot with a white background
ggsave("rf_feature_importance.png", plot = rf_plot, width = 10, height = 12, bg = "white")


# Extract feature importance
xgb_importance <- xgb.importance(model = xgb_model)

# Saving the XGBoost feature importance plot
png(filename = "xgb_feature_importance.png", width = 800, height = 600)
xgb.plot.importance(xgb_importance, rel_to_first = TRUE)
title("Feature Importance (XGBoost)")
dev.off()

```
```

Figure C.9: Training and testing model part 3
```

~~~~
```{r mannually predicting}

predict_resale_price <- function(town, flat_type, flat_model, storey_range,
                                 floor_area_sqm, remaining_lease,
                                 hawker_closest_distance, num_hawker_1km,
                                 mrt_closest_dist, num_mrt_1km,
                                 park_closest_dist, num_park_1km,
                                 sch_closest_dist, num_sch_2km,
                                 shoppingmall_closest_dist, num_shoppingmall_1km,
                                 supermarket_closest_dist, num_supermarket_1km) {
  # convert input parameters to a data frame
  input_data <- as.data.frame(list(town = town, flat_type = flat_type,
                                   flat_model = flat_model, storey_range = storey_range,
                                   floor_area_sqm = floor_area_sqm, remaining_lease = remaining_lease,
                                   hawker_closest_distance = hawker_closest_distance,
                                   num_hawker_1km = num_hawker_1km,
                                   mrt_closest_dist = mrt_closest_dist, num_mrt_1km = num_mrt_1km,
                                   park_closest_dist = park_closest_dist, num_park_1km = num_park_1km,
                                   sch_closest_dist = sch_closest_dist, num_sch_2km = num_sch_2km,
                                   shoppingmall_closest_dist = shoppingmall_closest_dist,
                                   num_shoppingmall_1km = num_shoppingmall_1km,
                                   supermarket_closest_dist = supermarket_closest_dist,
                                   num_supermarket_1km = num_supermarket_1km))

  # convert categorical variables to factor
  input_data <- input_data %>%
    mutate_at(vars(town, flat_type, flat_model, storey_range), as.factor)

  # Adjust the levels of the factors in input_data to match those in the original data
input_data$town <- factor(input_data$town, levels = levels(data$town))
input_data$flat_type <- factor(input_data$flat_type, levels = levels(data$flat_type))
input_data$flat_model <- factor(input_data$flat_model, levels = levels(data$flat_model))
input_data$storey_range <- factor(input_data$storey_range, levels = levels(data$storey_range))


# convert the newdata for XGBoost prediction
xgb_input_data <- input_data %>%
  mutate_if(is.character, as.numeric) %>%
  mutate_if(is.factor, as.numeric)

# ensure that the column order matches the training data for XGBoost
xgb_input_data <- xgb_input_data[, colnames(xgb_data %>% select(-resale_price))]

 # Make predictions
  rf_pred <- unname(predict(rf_model, newdata = input_data)) # unname the prediction
  xgb_pred <- predict(xgb_model, newdata = as.matrix(xgb_input_data))

  # calculate range
  range_percentage <- 0.05
  rf_range <- unname(c(rf_pred * (1 - range_percentage), rf_pred * (1 + range_percentage))) # unname the range
  xgb_range <- c(xgb_pred * (1 - range_percentage), xgb_pred * (1 + range_percentage))

  return(list(
    rf_prediction = rf_pred,
    rf_range = rf_range,
    xgb_prediction = xgb_pred,
    xgb_range = xgb_range
  ))
}


result <- predict_resale_price(
    town = "ANG MO KIO",
    flat_type = "2 ROOM",
    flat_model = "Improved",
    storey_range = "07 TO 09",
    floor_area_sqm = 44,
    remaining_lease = 58 + (1/12),  # 58 years and 1 month
    hawker_closest_distance = 0.190276573,
    num_hawker_1km = 9,
    mrt_closest_dist = 0.999806365,
    num_mrt_1km = 4,
    park_closest_dist = 0.674068637,
    num_park_1km = 23,
    sch_closest_dist = 0.331308119,
    num_sch_2km = 20,
    shoppingmall_closest_dist = 1.001278194,
    num_shoppingmall_1km = 6,
    supermarket_closest_dist = 0.742326582,
    num_supermarket_1km = 16
)

print(result)


```
~~~~

Figure C.10: Manually predicting housing resale price based on the parameters

~~~~
```{r shinny app}
library(shiny)
library(shinyWidgets)

# Define UI for application
ui <- fluidPage(

    # Application title
    titlePanel("Predict Resale Price"),

    # Sidebar with input elements
    sidebarLayout(
        sidebarPanel(
            selectInput("town", "Town", choices = levels(data$town)),
            selectInput("flat_type", "Flat Type", choices = levels(data$flat_type)),
            selectInput("flat_model", "Flat Model", choices = levels(data$flat_model)),
            selectInput("storey_range", "Storey Range", choices = levels(data$storey_range)),
            numericInput("floor_area_sqm", "Floor Area (SQM)", 44),
            numericInput("remaining_lease", "Remaining Lease (in years)", 58),
            numericInput("hawker_closest_distance", "Closest Hawker Distance", 0.19),
            numericInput("num_hawker_1km", "Number of Hawkers within 1km", 9),
            numericInput("mrt_closest_dist", "Closest MRT Distance", 0.9998),
            numericInput("num_mrt_1km", "Number of MRTs within 1km", 4),
            numericInput("park_closest_dist", "Closest Park Distance", 0.674),
            numericInput("num_park_1km", "Number of Parks within 1km", 23),
            numericInput("sch_closest_dist", "Closest School Distance", 0.331),
            numericInput("num_sch_2km", "Number of Schools within 2km", 20),
            numericInput("shoppingmall_closest_dist", "Closest Shopping Mall Distance", 1.001),
            numericInput("num_shoppingmall_1km", "Number of Shopping Malls within 1km", 6),
            numericInput("supermarket_closest_dist", "Closest Supermarket Distance", 0.742),
            numericInput("num_supermarket_1km", "Number of Supermarkets within 1km", 16),
            actionButton("predict", "Predict")
        ),

        # Main panel for displaying predictions
        mainPanel(
            h4("Random Forest Prediction"),
            verbatimTextOutput("rf_prediction"),
            h4("XGBoost Prediction"),
            verbatimTextOutput("xgb_prediction")
        )
    )
)

# Define server logic
server <- function(input, output) {

    observeEvent(input$predict, {
        result <- predict_resale_price(
            town = input$town,
            flat_type = input$flat_type,
            flat_model = input$flat_model,
            storey_range = input$storey_range,
            floor_area_sqm = input$floor_area_sqm,
            remaining_lease = input$remaining_lease,
            hawker_closest_distance = input$hawker_closest_distance,
            num_hawker_1km = input$num_hawker_1km,
            mrt_closest_dist = input$mrt_closest_dist,
            num_mrt_1km = input$num_mrt_1km,
            park_closest_dist = input$park_closest_dist,
            num_park_1km = input$num_park_1km,
            sch_closest_dist = input$sch_closest_dist,
            num_sch_2km = input$num_sch_2km,
            shoppingmall_closest_dist = input$shoppingmall_closest_dist,
            num_shoppingmall_1km = input$num_shoppingmall_1km,
            supermarket_closest_dist = input$supermarket_closest_dist,
            num_supermarket_1km = input$num_supermarket_1km
        )

        output$rf_prediction <- renderPrint({
            list(Prediction = result$rf_prediction, Range = result$rf_range)
        })

        output$xgb_prediction <- renderPrint({
            list(Prediction = result$xgb_prediction, Range = result$xgb_range)
        })
    })
}

# Run the application
shinyApp(ui = ui, server = server)

```
~~~~

Figure C.11: Supervised predicting housing resale price shiny app