Hitchhiker's Guide to Property Purchase in Singapore

CS5228 Project Report

Xiao Linfan
National University of Singapore
Team GhostBusters
linfan_xiao@u.nus.edu

Zheng Kexin
National University of Singapore
Team GhostBusters
k_zheng@u.nus.edu

Zhou Yemin
National University of Singapore
Team GhostBusters
e0729665@u.nus.edu

Zu Yuhan
National University of Singapore
Team GhostBusters
e0989359@u.nus.edu

***Abstract*—In this report, we present our work on three data mining tasks, namely prediction of property prices (task 1), property recommendation (task 2), and exploration of the relationship between subzone population and other factors related to the properties in the subzone as well as the subzone itself (task 3). The overarching goal of these tasks is to provide relevant insights to potential property buyers, sellers, as well as governmental and commercial parties that may have interest in the housing market of Singapore. To this end, we leverage a variety of data mining techniques, namely exploratory data analysis (EDA), data pre-processing, regression models such as linear model and tree ensembles, as well as content-based recommendation methods. We accomplish our goal by analyzing important features, evaluating various models for different tasks, as well as identifying use cases for property recommendation and designing solutions accordingly.**

***Index Terms*—Data mining, EDA, Tree ensembles, Content-based Recommendation**

I. Motivation

Property prices and property recommender systems play important roles in commercial activities in the housing market. Accurate predictions of property prices and appropriate property recommendations can help sellers maximize profit and help buyers secure ideal purchases. Therefore, our first and foremost tasks would be to build a robust regressor to predict property prices and a recommender system to recommend property listings to users based on information about their activities.

We start by performing EDA to identify important features in the given data as well as outliers, noise, and missing values that need to be handled. In the pre-processing stage, we proceed to clean the data and apply appropriate transformations to prepare the data for modelling. Next, for the price prediction task, we apply various regression techniques and examine their performance, including linear model, KNN, tree ensembles, neural network, and support vector regression. For the recommendation task, we identify two use cases, namely recommending based on the item currently being viewed, and recommending based on view history. We then design approaches for each scenario by leveraging content-based methods, such as pairwise item-item similarity and user-item similarity.

In addition to providing property price predictions and recommendations for potential buyers and sellers, we also seek to provide insights to other interested parties such as the government and real-estate developers. In particular, we explore the relationship between subzone population and factors related to properties in the subzone as well as to the subzone itself, including the number of various property types, property prices and sizes, and the subzone area. The results of our explorations are potentially useful for government planning, real-estate development, and other parties interested in the socio-economic aspects of Singapore's housing market.

II. Exploratory Data Analysis & Pre-processing

The raw train data consists of 20254 rows and 21 columns. It contains 9 numerical attributes and 12 categorical attributes. The details are shown in Table I.

Type	Attributes
Nominal	listing_id, title, address, property_name, property_type,subzone, planning_area, available_unit_types, property_details_url
Ordinal	tenure, floor_level, furnishing,
Interval	built_year, lat, lng, elevation
Ratio	size_sqft, num_beds, num_baths, total_num_units, price

TABLE I: Four types of attributes in the original dataset

A. Dependent Variable

The dependent variables are targets of the prediction. For task 1, we explore two dependent variables: `price` and price per square foot, or `per_price`, which is `price` divided by `size_sqft`.

From initial exploration with statistical evaluation, we find that there are two types of outliers in price: zeroes and extremely high values. We first drop records with zero price values. Then, we filter out extremely high-value outliers by leveraging three different standards: three sigma rules, inter-quartile range, and percentile ($25\% \sim 75\%$). This approach allows for more conservative outlier removal that avoids removing valid data with relatively high values. Moreover, we clean high-valued outliers records with HDB as property type, because the variation of prices among HDB listings should be relatively small, considering the fact that it is regulated by the government. The cleaned price serves as a reference for further EDA and pre-processing.

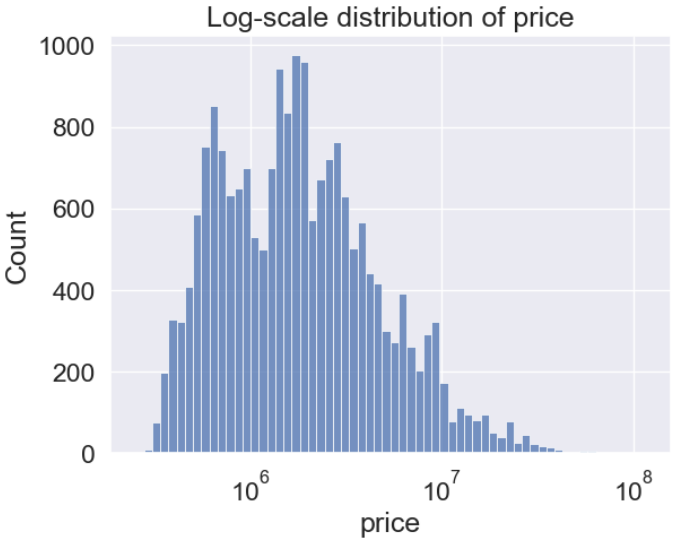
Since price depends on size_sqft, we expect that the per_price attribute is more consistent within the same complex than price. As shown in the log-scale distribution histograms of price and per_price in Figure 1, the distribution of per_price is less skewed than that of price. Therefore, we envision that per_price could serve as a better target for prediction. We add the per_price attribute after cleaning both price and size_sqft by dividing the former with the latter.

B. Nominal Features

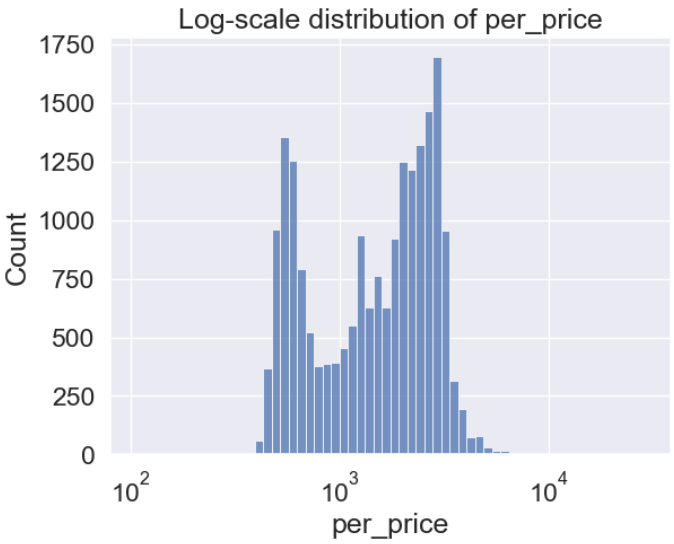
We examine the nominal features subzone and property_type, which we expect to be important factors in determining property pricing.

Plotting the grouped per_price attribute with latitude and longitude shown in Figure 4 shows that location is an important factor related to house pricing. Among nominal attributes, both subzone and planning_area embody geographical information, and it is technically possible to choose either of them to represent geographical information. We decide to use subzone because the amount of details it embodies is an appropriate balance between the numerical lat, lng (too detailed), and the nominal planning_area (too generalized). To handle missing values in subzone, we manually fill in Google Map results using the cleaned latitude and longitude information. As for encoding - there are 244 unique subzones, rendering one-hot encoding unsuitable - so we perform target encoding with per_price on subzone instead.

For property_type, we first unify its format by transforming all text into lower-case. Then, we replace all property types containing "hdb" (with various numbers of rooms) with the single category "hdb", because the number of rooms is already reflected in the num_beds attribute. Furthermore, by analyzing the mean and median price of each property type and boxplot result(Figure 2), we find a certain order in terms of prices among property types, suggesting that ordinal encoding may be appropriate. Therefore, we use this order as a reference to perform ordinal encoding on property_type. The resulting ordinal property type attribute consists of 9 categories that generalize property types by combining property types with similar average prices.



(a) Price.



(b) Price per suqare foot.

Fig. 1: Log-scale distribution of dependent variables.

C. Ordinal Features

More than 80% of the train data have missing values in floor_level, and more than 70% contain missing values or "unspecified" in furnishing. Since the majority records have missing values, it would not be meaningful to try data imputation or encoding the missing values as categorical variable, and so we decide to drop these two attributes.

Around 8% of the training data have missing values in tenure for which there exists no other listing in the same complex that can be used to directly fill in. Therefore, we use the tenure options on the 99.co website [1] as reference and generalize tenure into 4 categories, namely "NaN", "around 99-year", "around 999-year" and "freehold". We further apply ordinal

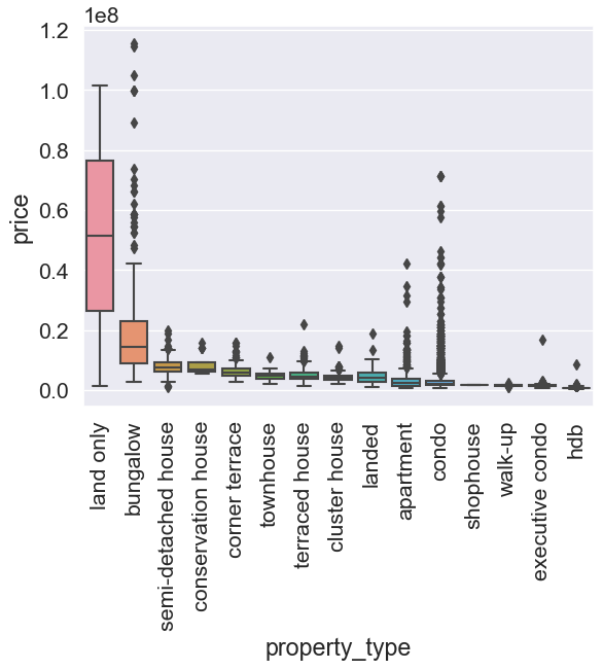


Fig. 2: Distribution of `price` follows `property_type`.

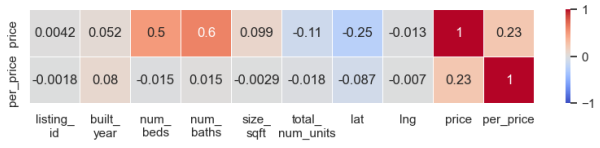


Fig. 3: Correlation heat map of different numerical attributes and target variables.

encoding to the four categories following the length of the tenure.

D. Interval attributes

`elevation` has the same value for all records, so we drop this attribute.

For attributes `lat` and `lng`, we find some outliers that are outside the territory of Singapore. We manually correct them by locating the property on Google Map.

Around 4% of the data have missing values in `built_year`. We have observed that `built_year` exhibits different distributions for different property types. Therefore, we decide to fill in the missing values with the mean `built_year` values aggregated over the general property types `"hdb"`, `"condo"`, and `"landed"`.

E. Ratio Attributes

We find two types of outliers in `size_sqft`. The first type uses square meter as unit of measure instead of square feet, and the other type is high-valued outliers. We identify entries with wrong unit of measure by comparing the value with a minimum threshold of 400, which corresponds to 4305 square feet. We convert these values back to square feet measures and

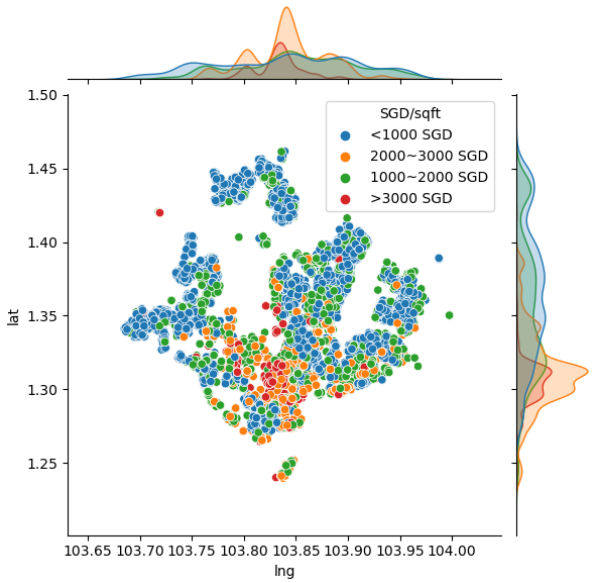


Fig. 4: Distribution of grouped `per_price` following geographical location.

validate correctness by finding records in the same complex with the same floor plan (i.e., `num_beds` and `num_baths`). We drop the record if the conversion cannot be validated or its value exceeds the maximum threshold of the property type determined from Figure 5. We perform the data cleaning separately for different general property types because we observe different distributions of sizes for different property types, as shown in Figure 5 .

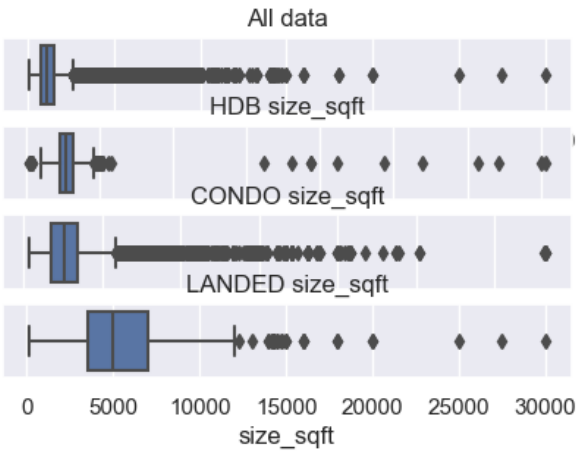


Fig. 5: Distribution of `size_sqft` for different property types.

We observe noticeable correlations between `size_sqft`, `num_beds`, `num_baths` from the EDA results, which we further exploit fill missing values. There are fewer than 100 records in the training data that have missing values in `num_beds`. We employ three approaches to fill in the missing values based on information obtained from EDA. Firstly,

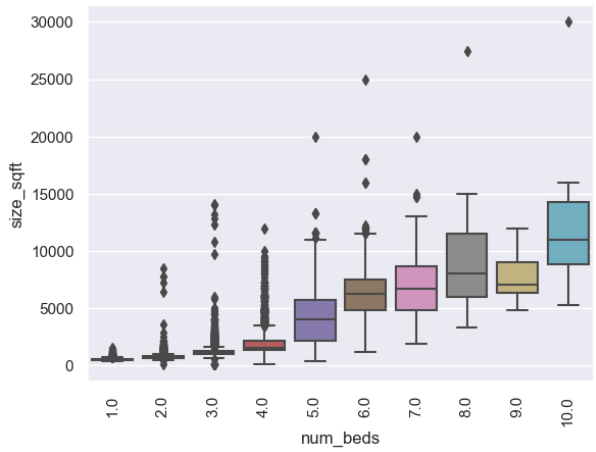


Fig. 6: Distribution of `size_sqft` follows `num_beds`

missing `num_beds` for Studios can be filled with 1 because by definition, studio has only one room. Secondly, we can use the information of other records at the same complex with the same `size_sqft` as reference. Lastly, the `num_beds` for HDBs can be estimated from `size_sqft`, because in comparison with other property types, the floor plans of HDBs tend be more restricted. For missing values in `num_baths`, if there exist other records in the same complex with the same `num_beds`, we fill the missing values using the `num_baths` of these records; if not, we drop the entries.

We further clean the outliers in `num_beds` and `num_baths` by looking for noises in the ratio of number of bedrooms to number of bathrooms (`b2b ratio`). More than 90% of the data have a `b2b ratio` larger than 0.3 (i.e., 1 bedroom, 3 bathrooms), and more than 99% of the data have a `b2b ratio` smaller than 3 (3 bedrooms, 1 bathroom). We identify outliers as records whose `b2b ratio` lies outside $[0.3, 3]$. If there exist other listings from the same complex with the same `size_sqft`, we correct the outliers using information from these records; if not, we drop the outliers.

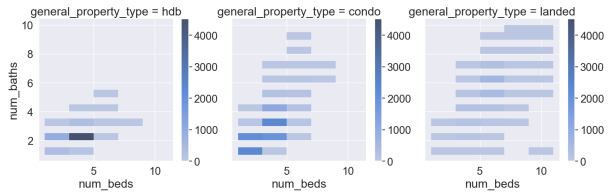


Fig. 7: Distribution of `num_beds` and `num_baths` for different general property types.

We drop the attribute `total_num_units` because it does not seem to have evident relationship with the dependent variable.

F. Auxiliary Data

For public facilities such as shopping malls and schools, we create new features representing the number of various facilities within a certain radius from the property, with the radius being 0.3km for shopping malls and 1km for schools.

Inspired by 99.co website [1], we decide to incorporate some notion of the distance to the nearest MRT station into our data. We choose 50 important MRT stations based on their degree centralities, and create a new feature representing the distance to the nearest one.

For commercial centres, we first find the nearest commercial centre of various types for each property, and further investigate their effects on the dependent variables. We observe a correlation between price and commercial centre types IHL, BN, CR. Therefore, we perform ordinal encoding on these three types of commercial centres in a manner similar to how `property_type` is encoded.

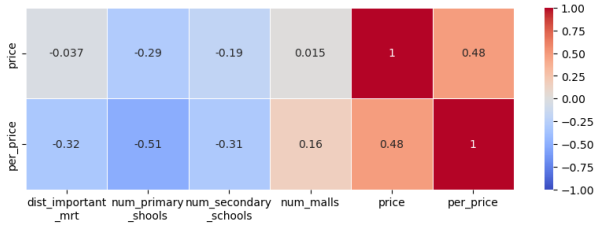


Fig. 8: Correlation heat map of auxiliary attributes and dependent attributes.

G. Task 2

We further generate pseudo user view history for user-based recommendation. The user data is created under a scenario where they are searching with different criteria on the real estate website. From our observation of the 99.co website's search criteria, we select 11 attributes as potential search filters that users can adjust based on their preference.

Type	Criteria
Mandatory criteria	num_beds, price property_type
Location criteria	subzone, planning_area
Optional criteria	nearest_mrt, num_baths,num_ beds, tenure, built_year, size_sqft, per_price

TABLE II: Three types of features as search criteria.

Each pseudo user is assigned one search criteria set that contains more than 1 mandatory criterion, at most one location criterion, and optional criteria. 15 properties that fit the search criteria are randomly collected to form the view history. An additional viewing time for each chosen property is randomly generated as a ranking standard to further indicate the user's preference on selected properties.

H. Task 3

1) Data creation: In task 3, we plan to explore factors in the main data and auxiliary data that may affect the total population in each subzone. The features we consider include the number of HDB sales in each subzone, the number of condo sales in each subzone, the number of landed sales in each subzone, the average size of all sales in each subzone, the average size of all sales in the planning area which each

subzone belongs to, the area size of each subzone, average price per square foot of each subzone, and average price per square foot of the planning area which each subzone belongs to. We obtain these data by aggregating the relevant features by subzone. The target we consider is the population of each subzone.

2) EDA: First of all, we remove all records that contain missing values. Then, by the correlation heat map of different numerical attributes and target variable `population`, we find that `hdb_count` has a good linear relationship with `population`.

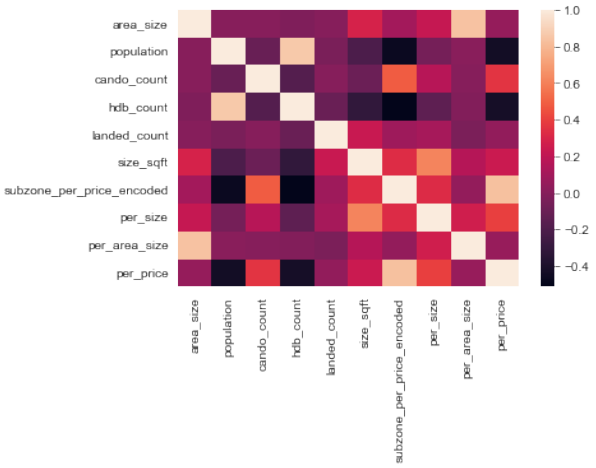


Fig. 9: Correlation heat map of different numerical attributes and target variables.

III. Section

IV. Data Mining Methods

A. Task 1: Property Price Prediction

1) Linear (Baseline): We use the linear model as our baseline to perform property price prediction. The assumption required is that there exists linear relationship between our selected features and the target variable. Although this is not necessarily the case for our data, we construct the linear model as a baseline for assessing the performance of other models.

TABLE III: Results for the linear model.

Model	30% test set RMSE
Linear	2988026

As shown in Table III, the performance of the linear model is sub-par, suggesting that the relationship between our selected features and target variable is non-linear. Therefore, we proceed to consider non-linear models.

2) KNN: Since the relationship between our selected features and target variable may be non-linear, KNN for regression could be a good choice. k is an important hyper-parameter that affects model performance. A k that is too small would make the predictions sensitive to noise, resulting in uneven decision boundaries. A k that is too large would diminish the model's capacity to capture local patterns. So we choose k from 2 to 20 and run the model of each integer k on the training dataset. Plotting MSE against values of k shows that MSE decreases rapidly for $k < 5$ and decreases slightly for $k > 5$, so we choose $k = 5$.

TABLE IV: Results for the KNN model.

Model	parameters	30% test set RMSE

Linear	`n_neighbors=5`
	`weights="distance"`
	`algorithm="kd_tree"`
	`p=2`
	`metric="minkowski"`
		1768119

As shown in Table IV, the performance of the KNN model on the 30% test dataset is better than that of linear model. The assumption of KNN is that data points that are close are similar, which is appropriate for our dataset since properties with similar features are expected to have similar prices. So KNN is a reasonable candidate for price prediction.

3) Tree ensembles: Tree-based models are another family of models that allow capturing non-linear relationships between the independent and dependent variables and thus are suitable for our prediction task. In particular, we consider tree ensemble models. For each model, we employ `GridSearchCV` from sklearn with 10-fold cross-validation to tune a selected set of hyperparameters. To restrict the search space, we limit the number of hyperparameters tuned to no more than 3 and keep the number of candidates for each hyperparameter to 4 via appropriate spacing.

TABLE V: Results for tree ensemble models.

Model	Grid search results	R^2	30% test set RMSE
AdaBoost	`learning_rate=0.01` `n_estimators=100`	0.75	2509656

Random Forest	`max_depth=50`
	`max_features=8`
	`n_estimators=100`
		0.87	1325211

Gradient Boosting	`learning_rate=0.5`
	`max_depth=4`
	`n_estimators=400`
		0.88	1353585

As shown in Table V, Random Forest and Gradient Boosting have much better performance than AdaBoost. We use different ranges for `max_depth` and `n_estimators` when tuning hyperparameters for Random Forest and Gradient Boosting, because the former uses relatively small number of full grown trees (strong learners), and the latter uses relatively large number of shallow trees (weak learners). Table V shows that Random Forest has slightly lower R^2 on training/validation set but slightly better RMSE on the 30% test set than Gradient Boosting, suggest that Random Forest may be less prone to over-fitting. But overall, Gradient Boosting and Random Forest have similar performance, making both of them reasonable candidates for the task (as well as stacking model, as will be discussed in a later section).

4) SVR: SVR is yet another family of non-linear models. We employ `GridSearchCV` with 5-fold cross-validation to tune a selected set of hyperparameters. To restrict the search

space, we limit the number of hyperparameters tuned to no more than 3 and keep the number of candidates for each hyperparameter to 3 via appropriate spacing.

TABLE VI: Results for the SVR model.

Model	parameters	30% test set RMSE
SVR	C=1 gamma=0.0667 kernel='sigmoid'	2797636

As shown in Table VI, the performance of SVR is not as good as tree ensembles.

5) Neural networks: Motivated by recent advancements in Neural Networks, we decide to also try Multi-Layer Perceptron (MLP) for the price prediction task. We expect MLP to be able to learn latent relationships between different real estate features and make good predictions based on them.

We have tried different model structures and hyperparameters and find that the model with simple linear layers and dropout layers works relatively well on the price prediction task.

Our final model is an MLP with 4 linear layers, the first 3 layers are activated by ReLU activations, followed by a dropout layer to avoid overfitting. The model is trained with a learning rate of 1e−3 for 1500 epochs. We use the Adam optimizer with a weight decay for regularization. We build the model using the PyTorch library. We choose a batch size of 64 to balance the trade-off between training time and accuracy. We split the train data into a training set and a validation set with a ratio of $8:1$, where the validation set servers to evaluate the generalizability of the trained model on unseen data. The best model with minimal MSE loss on the validation set is saved during training.

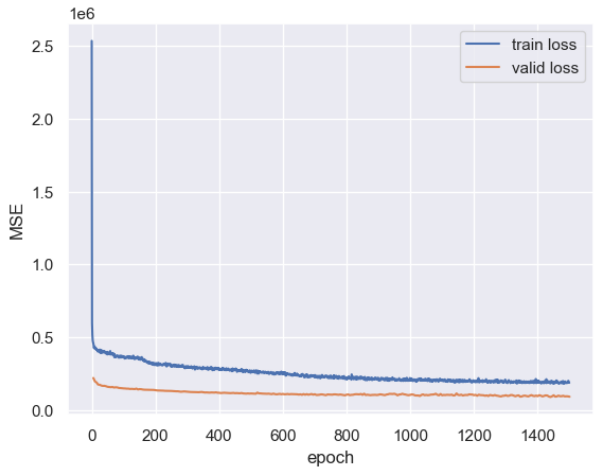


Fig. 10: MLP train loss and validation loss

6) Weighted Stacking Models: Given that different models have their own drawbacks and advantages, we decide to obtain a stacked prediction by combining prediction results of different models. We pick three models with the best performance on the 30% test data, namely Random Forest, Gradient Boosting, and MLP.

TABLE VII: Results for the MLP model.

Model	parameters	30% test set RMSE
MLP	dropout=0.2 learning_rate = 1e-3 epoch = 1500 weight_decay = 1e-4	1636327

The weights for different models are calculated given their RMSE score on training data. We give larger weights to models with smaller RMSEs. The RMSE on 30% test data is shown in Table VIII. From our observation, the stacking model has outperformed all 3 selected models on the 30% test set data.

TABLE VIII: Results for stacking models.

Model	30% test set RMSE
Gradient Boosting & Random Forest	1301756
Gradient Boosting & Random Forest& MLP	1307524

7) Model Selection: For the final submission, we select 5 models with the best performance on 30% test data, the selected models include Random Forest, Gradient Boosting, and the two aforementioned Stacking models.

B. Task 2: Property Recommendation

We consider two scenarios: 1) The user has just viewed or is currently viewing a property listing he/she is interested in, and wants to request other listings similar to that particular listing, and 2) the user has been browsing for some time, and would like to see some other suggestions based on his/her view history and the current search criteria.

Based on the above scenarios, we make the assumption that we have access to the users' view history (e.g., last 15 items) as well as search criteria. We do not consider ratings because a property is unlike a tourist attraction or movie that people can consume without competition, and so it would be difficult to justify the meaning of ratings on a property sales website. Therefore, we adopt content-based methods instead of collaborative filtering, and we choose to focus only on view history and search criteria as inputs.

The features we use for property recommendation include property type, number of bedrooms and bathrooms, latitude and longitude (so that it is easier to look for properties in the same community), size, distance to nearest MRT, price, built year, and tenure, which are all important factors to consider when searching for a house. In order to use cosine similarity, we apply one-hot encoding to the categorical variables. All features are normalized using sklearn's `MinMaxScaler` before computation.

We design our approaches as follows using content-based methods. For the first scenario, we propose an approach based on pairwise item-item similarity. For the second scenario,

we propose two approaches, one using pairwise item-item similarity, the other using user-item similarity.

1) Recommend based on item last viewed/currently being viewed: This scenario can be translated into a pairwise item-item similarity problem with one reference item. In this case, we only take the given item as input. We compute the cosine similarity between the given item and all other items and select the top k items with the highest similarities for recommendation.

2) Recommend based on view history: There are two interpretations of this scenario: 1) As a pairwise item-item similarity problem with multiple reference items, and 2) as a user-item similarity problem. In this scenario, we leverage the user's search criteria and filter the items accordingly before further computation.

In the first interpretation, we treat view history as multiple reference items. We compute the pairwise cosine similarity matrix between the reference items and all other items. Thus, for each candidate item, there would be one similarity score for each of the reference items. We combine the similarity scores by taking the average across reference items and select the top k items with the highest average similarity for recommendation.

In the second interpretation, we first construct a user profile by combining features of items in the view history using a weighted sum. Our intuition is that the later the view time and the more frequently an item is viewed, the more important that item should be in (current version of) the user profile. Thus, we compute the weighted sum by first dividing the view time by the latest view time and then aggregating by property listing id, so that items viewed later or viewed many times would have higher weight in the user profile. Then, we compute the pairwise cosine similarity matrix between the user profile and all other items and select the top k items with the highest similarities for recommendation.

C. Task 3

To find the relationship between subzone population and other factors related to property sales in the subzone and the subzone itself, we consider subzone population as the target variable, and select the factors shown in the data creation of task 3 as features.

1) SVR: For each model, we use the corresponding regressor from sklearn and employ `GridSearchCV` with 5-fold cross validation to tune a selected set of hyperparameters.

Because we cannot see the selling price of test data. We only consider the selling data in the training data of task 3. In order to compare different models, we split the data into training data and testing data with the same random state when using the method `train_test_split`.

2) tree ensembles: In this task, we consider the tree ensemble models: gradient boosting and random forest. For reach model, we use the corresponding regressor from sklearn and employ `GridSearchCV` with 10-fold cross validation to tune a selected set of hyperparameters.

Because we cannot see the selling price of test data. We only consider the selling data in the training data of task 3. In order to compare different models, we split the data into training data and testing data with the same random state when using the method `train_test_split` . The outcomes are in the table IX.

TABLE IX: Results for models in task3

Model	Grid search results	20% train set MSE

SVR	`C=1`
	`gamma=0.125`
	`kernel='poly'`
		316809375

Random Forest	`max_depth=50`
	`max_features=8`
	`n_estimators=50`
		70876798

Gradient Boosting	`learning_rate=0.1`
	`max_depth=4`
	`n_estimators=400`
		63760493

From the table IX, we can see the model of random forest and gradient boosting could get smaller MSE. So random forest and gradient boosting could be used in this task better than SVR.

V. Evaluation & Interpretation

A. Task 1: Property Price Prediction

1) Evaluation: We employed a variety of models in the data mining methods section, and each has certain advantages and disadvantages. Linear models are fast to train and use in prediction, and are known for good interpretability. However, it is not applicable to cases where the data does not exhibit a linear relationship, and its prediction accuracy can be sub-par. KNN is also simple and intuitive to use, and yet the distance computation when finding nearest neighbours during prediction can be slow, and all training data need to be stored, making storage a potential bottleneck for large datasets.

Tree ensembles such as Random Forest and Gradient Boosting provide state-of-the-art performance. Random Forest models can be parallelized to save training time, whereas Gradient Boosting is serial and may not be suitable for real-time prediction. Tree-based models are in general robust against outliers, however, they are also subject to limitations when being extrapolated to new data that are of a different range than the training data. For instance, the tree-based models we trained on the given data may not be suitable for price prediction in a luxury property market.

Neural Network methods such as MLP allows learning features from raw data without requiring extensive data pre-processing, and such features can be reused on other datasets or tasks, enabling them to benefit from knowledge gained previously. This means that Neural Network methods can be used in complex non-linear problems. However, Neural Networks in general require large amount of training data to achieve good performance, and are not robust to uninformative features [2]; that is, if an MLP is trained with many weak predictors, the effect would be sub-par.

SVR has the advantage that its computational complexity does not depend on the dimensionality of the input space.

It also has good generalization capability with high prediction accuracy. However, this model is not suitable for large datasets like our current target.

2) Evaluation on different price per square foot segments: In addition, we assess the performance of our models under different scenarios by splitting the target variable price per square foot into roughly even ranges, constituting 6 segments, namely below S$600, between S$600 and S$1200, between S$1200 and S$1800, between S$1800 and S$2400, between S$2400 and S$3000, as well as above S$3000. We then compute and compare the RMSE of all the above models on each segment to see whether they tend to perform well on certain segments but not the others. From Figure11, we can see that our models perform poorly on properties with a high price of over S$3000 per square foot. This is expected because the amount of data in high-price range is fewer than other segments, which only contains 10% of the train data. The lack of sufficient high-price data affects the ability of the model to learn useful features for good prediction. We also observe that the stack model out performed both Random forest and Gradient boosting model in the first 5 segments, which implies that the stack model benefited from the advantages of both model. Another observation is that the KNN model performs very well on different segments, while its performance on test data is not promising. This possibly because the model overfits on the train data, which is used for the price segment evaluation.

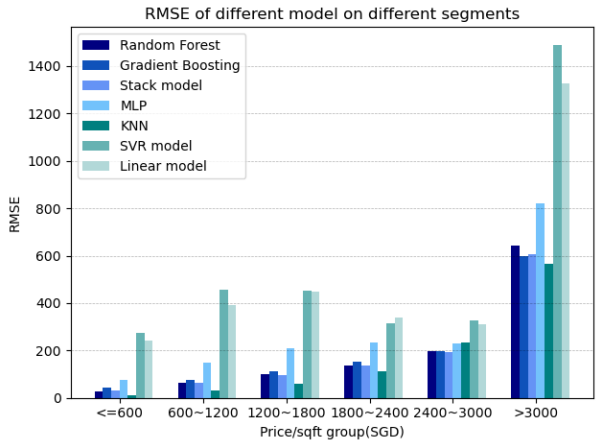


Fig. 11: RMSE of different model on `per_price` segments.

One possible improvement would be to train a separate model for segments in this range. We observe that the majority of the properties in this range are landed, and we conjecture that for potential buyers of this type of property, features such as distance to the nearest MRT, number of nearby shopping malls, are not as important as they are for buyers of properties in the other segments. Therefore, it would be sensible to train a separate model for these properties. Another possible reason for our models' poor performance on properties that are above S$3000 per square foot could be the lack of training data. A solution for this problem would be to collect more data on properties whose prices are on the higher end.

3) Interpretation: Figure 12 shows the importance of each feature in our Gradient Boosting and Random Forest models, respectively. We see that property type, built year, size, latitude, longitude, number of bathrooms and bedrooms all play important roles in predicting property prices. In addition, it is worth noting is that some of the auxiliary attributes we incorporate have higher importance than features related to the property itself. For instance, `name_of_nearest_CR_ordinal` and `name_of_nearest_BN_ordinal` are features derived from different types of commercial centers, and as shown in Figure 13b, they have higher importance than `num_beds` as well as the number of nearby facilities such as schools and shopping malls. We conjecture that this is because nearby commercial centres embody certain geographical information which more accurately reflects the condition of the surrounding community than information about the house itself as well as the number of nearby facilities.

This is actually a recurring theme we have seen during EDA. We found that the distance to the nearest MRT station does not have evident relationship with property prices, but the distance to the nearest 50 important MRT stations does (in terms of betweenness). MRT stations with high betweenness are usually traffic hubs, which are inherently located in prosperous regions. We hypothesize that distance to the nearest important MRT stations and name of the nearest commercial center embody similar geographical information that represents, to some extent, the level of prosperity of the property's surroundings. And this type of information has profound influence on the property's prices.

B. Task 2: Property Recommendation

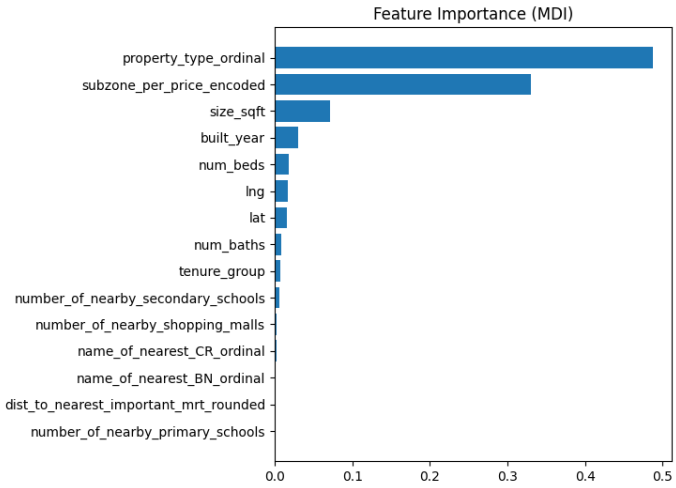
1) Recommend based on item last viewed/currently being viewed: We showcase our results using two examples for $k = 3$. In table X, the top row is the given item, and the remaining rows are our recommendations. Only a subset of the features are shown.

TABLE X: Recommendation results based on given item.

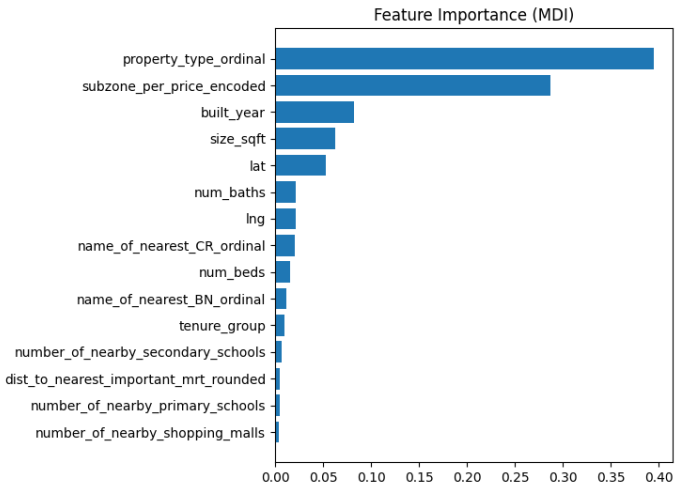
id	type	#beds	#baths	···	size	price	year
305042	condo	4	3	···	1324	2079000	2023
664945	condo	4	3	···	1324	2475900	2023
152766	condo	4	3	···	1324	2532000	2023
396631	condo	4	3	···	1324	2659600	2023

id	type	#beds	#baths	···	size	price	year
524387	condo	4	4	···	2077	6300000	2014
184136	condo	4	4	···	2476	7875000	2014
660576	condo	4	4	···	2077	6300000	2014
851641	condo	3	2	···	947	1933000	2026

We can see that the performance is reasonable in general. Yet we also note that due to the nature of cosine similarity, the recommendation engine sometimes counter-intuitively finds items that have higher values than the given item in some attributes but lower values in some other attributes. That is, given $v_1 = (1, 2, -1)$, $v_2 = (1, 3, -2)$ has higher similarity



(a) Gradient Boosting.



(b) Random Forest.

Fig. 12: Feature importance of tree ensemble models.

with it than $v_3 = (1, 2, -2)$, though intuitively we would find v_3 more similar to v_1. The last row in the second table above is one such example. A possible improvement would be to consider a combination of cosine similarity with some other metric.

2) Recommend based on view history: Again, we showcase our results using an example for $k = 3$. However, the evaluation of this scenario is arguably not as straightforward as the previous case. In Table XI, the first part is the given view history, and the remaining two parts are recommendations based on pairwise item-item similarity and user-item similarity, respectively.

In this scenario, the similarities are aggregated and the user view history contains multiple items. So it becomes less clear how to determine what constitutes a "good" recommendation, since doing so requires interpreting and evaluating the relationship between the view history and the recommendations. That said, we do see some reasonable level of agreement between

TABLE XI: Recommendation results based on view history.

View history							
id	**type**	**#beds**	**#baths**	...	**size**	**price**	**year**
682504	condo	4	2	···	1163	2422600	2025
450511	condo	3	2	···	915	2068500	2025
425951	condo	3	2	···	797	1827000	2025
496786	condo	4	3	···	1302	3299100	2024
901635	condo	3	3	···	1528	2520000	2013
...	...	...	...	...	...	...	...

Recommendations based on pairwise item-item similarity							
id	**type**	**#beds**	**#baths**	...	**size**	**price**	**year**
536624	condo	2	2	···	484	1711500	2026
618506	condo	4	3	···	1184	1520400	2017
272193	condo	2	2	···	883	1037400	1977

Recommendations based on user-item similarity							
id	**type**	**#beds**	**#baths**	...	**size**	**price**	**year**
536624	condo	2	2	···	484	1711500	2026
905005	hdb	3	2	···	1001	651000	2016
618506	condo	4	3	···	1184	1520400	2017

the pairwise item-item and user-item approaches. As shown in Table XI, listing 536624 is on top of both approaches' recommendations.

A possible improvement would be to restrict the important features (perhaps leveraging user input) to a finer subset and incorporate this information into the weights computation. We expect that doing so would improve both the quality and interpretability of the recommendations.

C. Task 3

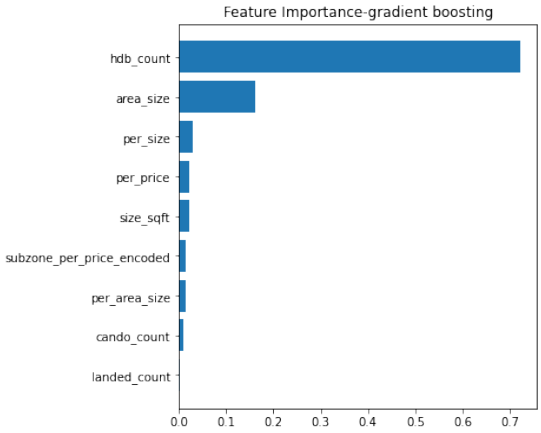
As discussed in the data mining methods for task 3, we employ random forest and gradient boosting for this task. In order to evaluate and interpret the factors used in the models, we compute the feature importance for each model.

Figure 13 shows the feature importances for our random forest and gradient boosting models, respectively. The figures shows certain interesting agreement between the two models, namely that both models consider the number of HDB sales in the subzone and the subzone area sizes as the two most important features in determining the subzone population. This is in line with expectation since the construction of HDB is a prominent part of Singapore government's policy to accommodate the majority of residents. Therefore, it is reasonable to see that these two features have profound impact on the population of each subzone.

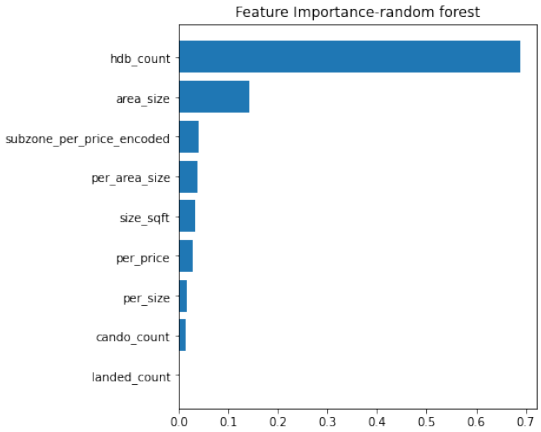
VI. Conclusion

In order to provide useful insights to potential buyers and sellers of the Singapore housing market as well as other interested parties, we utilize data from 99.co and adopt different data mining methods to perform price prediction, property recommendation, as well as real-estate related population feature exploration.

In particular, we demonstrate how we perform exploratory data analysis and pre-processing, how we design approaches to different data mining tasks, how we evaluate different data



(a) Gradient Boosting.



(b) Random Forest.

Fig. 13: Feature importance of tree ensemble models in task 3.

mining methods based on the task goals and identify their limitations. Our evaluation shows that our approaches are valid and feasible. During the process, we have seen significant improvement on performance benefiting from further data exploration and model selections. We envision that we can further improve our results via better feature engineering and additional model exploration.

REFERENCES

[1] Singapore's smarter property search. [Online]. Available: https://www.99.co/

[2] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on tabular data?" 2022. [Online]. Available: https://arxiv.org/abs/2207.08815