Hong Kong Baptist University
MATH 3836 Data Mining
Semester B (2022/23)

Recommendation System on Real Estate Market

Abstract:

The real estate market is a complex and dynamic industry that requires consistent monitoring and analysis in order to be able to make informed decisions. Real estate professionals find it more difficult to remain ahead of trends and make accurate forecasts due to an increase in the amount of data available. This is where a recommendation system can be useful. It is based on different machine learning techniques that analyses customer data and provides customised recommendations to users in real time. In the property market, it can help real estate agents make informed decisions by providing them with relevant and timely information, while end-users can also be benefited by the robustness of recommendation algorithms to find their preferred properties efficiently.

This project mainly focuses on ways in assisting real-estate professionals make informed decisions and end-users to find preferred properties efficiently. However, the recommendation task is substantially more challenging in the real estate domain due to the availability and quality of data. They are often fragmented and incomplete, making it difficult to analyse and interpret. Moreover, real-estate recommendation system usually faces the cold-start problem. For instance, there are no historical data for new users. Despite these challenges, we endeavour to build a complete and efficient recommendation system for both users and agents.

Name	Student ID
WU Chun Yip	22515313
MUNG Tsz	19230680

Table of Content

1. Introduction
- 1.1. Settings
- 1.2. Problem Formulation
- 1.3. Goals

2. Dataset
- 2.1. Modelling available data
  - 2.1.1. Exploratory Data Analysis
  - 2.1.2. Feature Engineering
- 2.2. “Hope to have” data
- 2.3. Futuristic data mining application

3. Implementation & Discussion
- 3.1. Linear Model
- 3.2. Decision Tree
- 3.3. Collaborative Filtering
- 3.4. K Nearest Neighbours

4. Conclusion

5. Appendix

6. Reference

1. Introduction

1.1. Settings

We are working in real estate agent company and our business is to provide accurate and efficient recommendation services to our clients. We want to combine the advanced machine learning technology to enhance our recommendation services, so that we can provide our clients with more accurate and data-driven insights into the property market. We are designing a property recommendation system which aims to provide suggestions for our clients when considering properties.

1.2. Problem Formulation

What is a good recommendation? In an ideal situation, we would like to know how real users react to recommendations, and adjust the following recommendations based on their feedback. Through a series of trial and error, the model should be able to give suggestions as close as the real users' choices, and there are some general statistical accuracy metrics to evaluate the recall rate of the model. People tend to use online platforms to look for open houses, as they can specify their best match criteria among many other irrelevant houses. Yet, the search results might not be the best fit as they are generalized by the search engine.

Client choices are another big topic that the system need to take into account, some essential elements such as their budget, preferred location, and desired property features. In order to provide the most accurate and personalized recommendations, we will need to collect and analyze a large amount of data from our clients. This data will include information on their previous property searches, budget constraints, and preferred locations. By carefully analyzing this data, we can gain insights into our clients' preferences and provide recommendations that are tailored specifically for them.

Cold start is a common challenge in recommendation system. This is a phenomenon that the system cannot draw any inferences for items or users due to the lack of historical logs for new or inactive users.

1.3. Goals

The purpose of our proposal is to build a complete and efficient recommendation system for both users and agents. The agents can receive real-time insights and statistics from our recommendation system, which helps them make wise decisions and increase their productivity. Moreover, we anticipate that the recommendation system will make it simple and quick for agents to get customized recommendations that are catered to each user's unique needs and interests. On the other hand, we aim to provide each user with personalized recommendations that are pertinent and helpful based on their preference and behaviors. Also, we are eager to provide a seamless and integrated user experience by leveraging data from multiple sources, such as CRM systems and user behavior data.

To achieve our goal, we construct a linear regression model and decision tree to model the house price. Furthermore, we combine collaborative filtering and KNN technique to build a recommendation system. The collaborative filtering is efficient at recommending the house based on the user preference, while the KNN is efficient at recommending the house based on their features, which can address the cold-start problem caused by the collaborative filtering. We hope our models are able to find out the user’s preference via their activities and recommend corresponding houses for them. It is hoped that our recommendation system improves the user’s experience so as to increases the transaction success rate.

2. Dataset

2.1. Modelling Available Data

In our report, we have selected the Ames Housing dataset from Kaggle, which provides a lot of information about the property that influences the price. There are 79 explanatory variables describing almost every aspect of residential homes in Ames, and the goal is to predict the final prices.

2.1.1. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a technique that involves visualizing data to gain a deeper understanding of the dataset and determine the best approach for analyzing it.

There are two files provided, trained and test in CSV format, and there is no SalePrice column in the test file as this is the target label. Pandas package was used to read and clean the data. Since train and test files have the same structure, the two files are concatenated into a merged dataset for later cleaning purposes. It will be splitted back to the original dataset with the same amount of data, and the train dataset will be further divided into training and validation set in order to evaluate the models.

It is observed that the merged dataset has 2919 entries, of which 38 are numerical and 43 are mixed with strings. Excluding the ID and SalePrice columns, there are 36 numerical columns. With such a high dimension of data, an EDA would be required to gain a deeper understanding of the dataset and determine the approach of feature engineering.

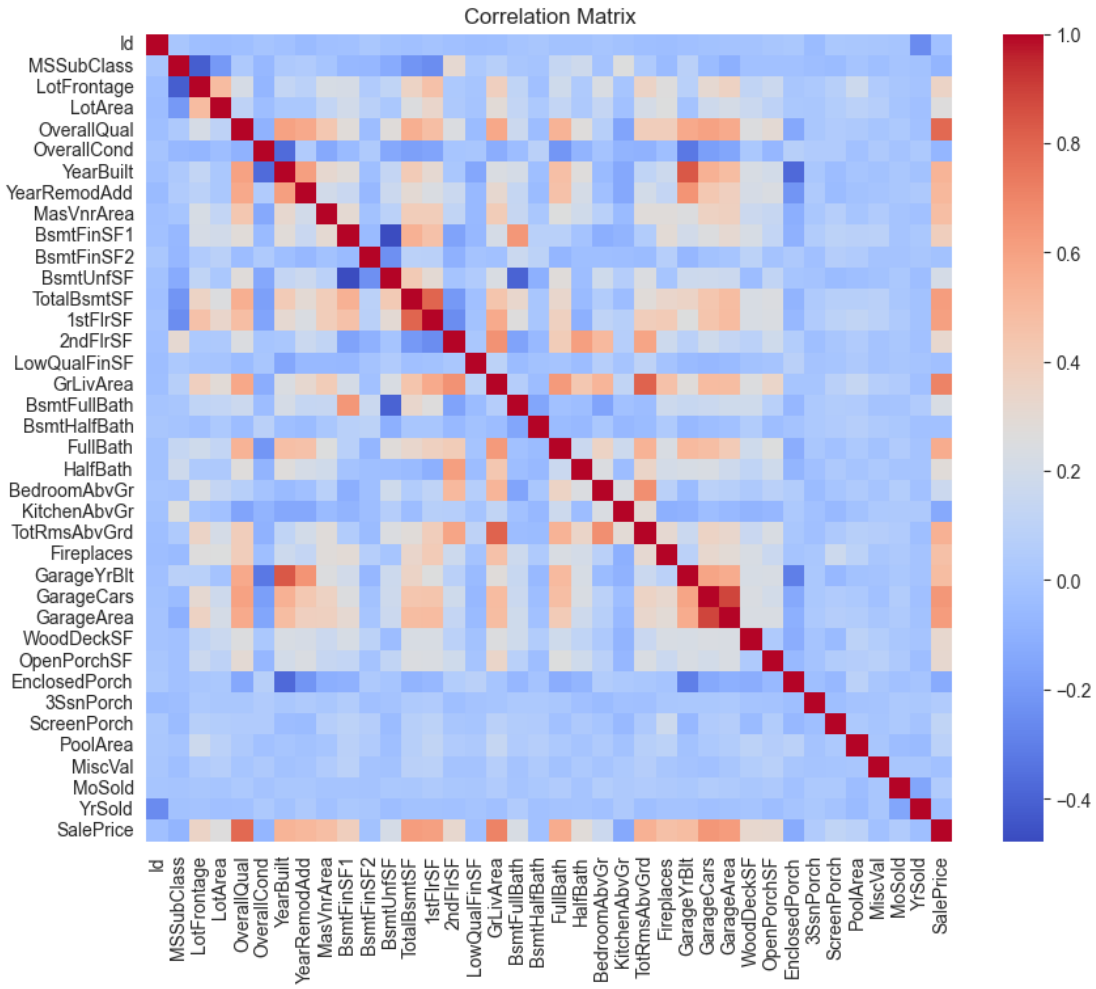


Fig. 1a: Correlation Matrix of the dataset (pre one-hot encoded)

Fig. 1a is the correlation matrix visualised in a heatmap which can illustrate the relationship between two variables. From the SalePrice column the heatmap shows that the size of different parts of house plays an important role in the property price, and it

makes sense that it is a quantitative measure of the property itself. The heatmap can also help in identifying highly correlated variables so that they could be dropped to reduce the dimension of data. For instance, LotFrontage and 1stFlrSF, TotRmsAbvGrd and GrLivArea, YearBuild and GarageYrBlt. The one with higher correlations with the saleprice was kept while the other one was dropped.

Since the above correlation matrix only shows the relationship of numerical variables, while the categorical variables are ignored, it is hoped that all variables could be included to have a big picture. Therefore, we tried to see the impact of one-hot encoding to the correlation matrix and visualise it in the same manner as above after one-hot encoding as shown in Fig. 1b. However, since the dimension of dataset was expanded to 290, it is impossible to interpret the correlation in such way, and we also realised the importance of feature engineering due to the curse of dimensionality.

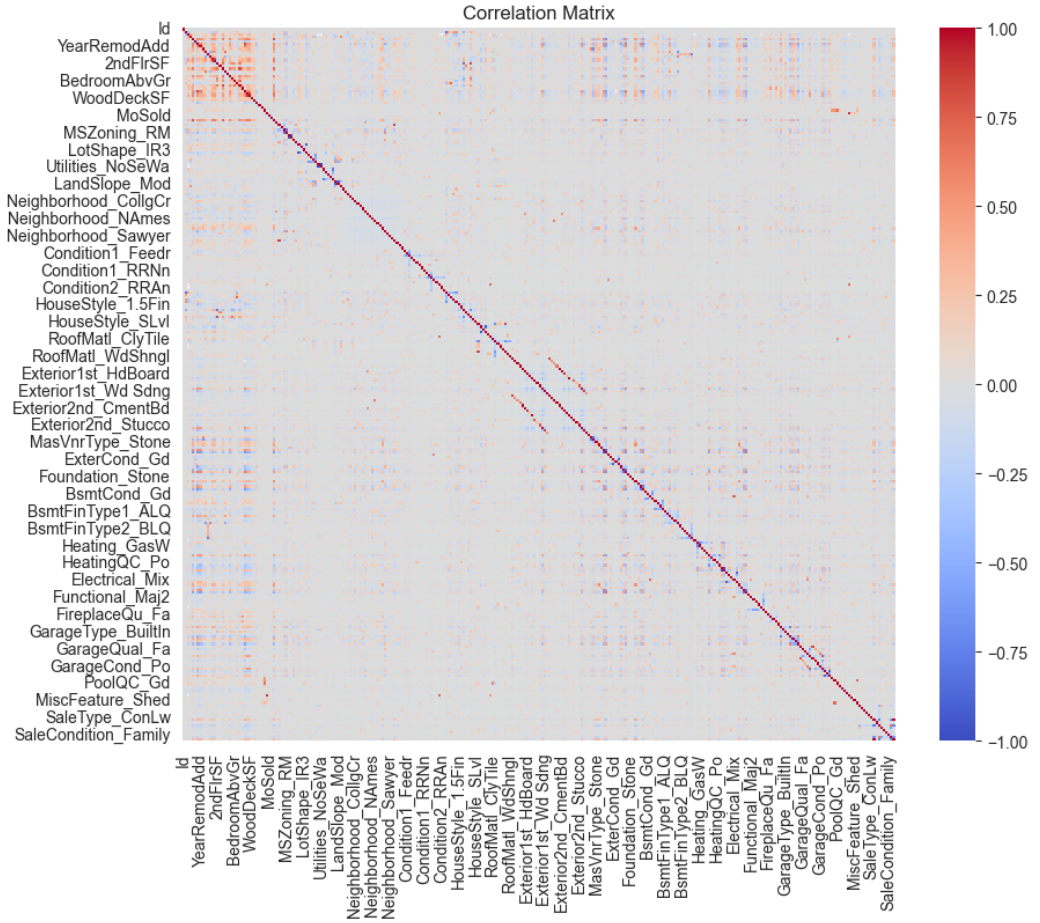


Fig. 1b: Correlation Matrix of the dataset (post one-hot encoded)

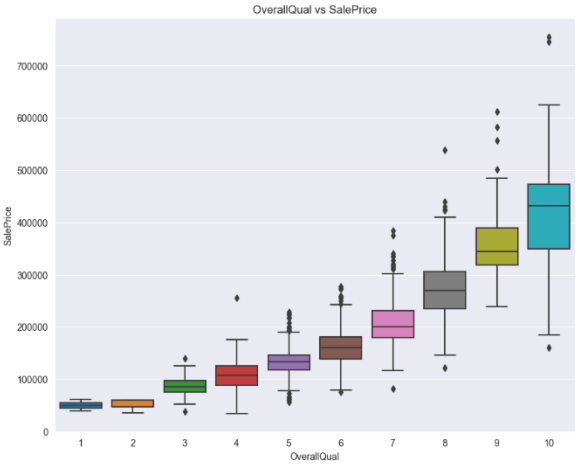


Fig. 2a: OverallQual vs SalePrice

Fig. 2a is a box plot of the overallqual against saleprice. The OverallQual is a rate of the overall material and finish of the house. There is an apparent relationship between two features, the price of the houses increases as the overall rates increases.

Fig. 2b: GrLivArea vs SalePrice

Fig. 2b is a scatter plot between GrLivArea and saleprice. There is an apparent relationship between two features, the price of the houses increases as the the size of living area increases. There are two outliers detected from the graph, which were marked and dropped in feature engineering.

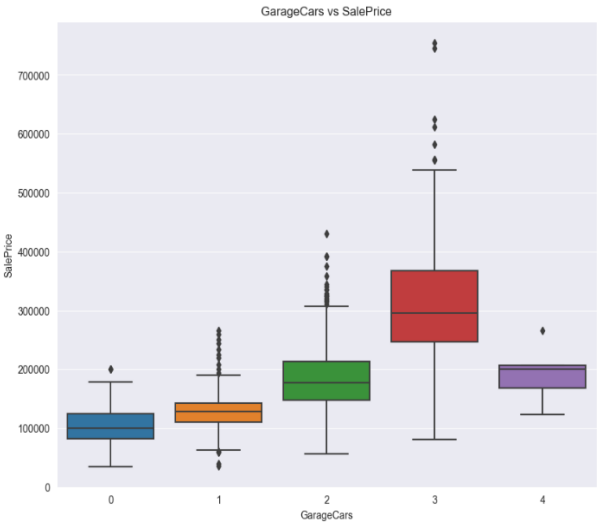


Fig. 2c is a box plot between GarageCars and saleprice. The graph shows the houses with 3 car capacity has the largest range of house price. Also, the GarageCars is highly correlated with GarageArea and it has a higher correlation with saleprice, so the GarageArea has been noted to drop.

Fig. 2c GarageCars vs SalePrice

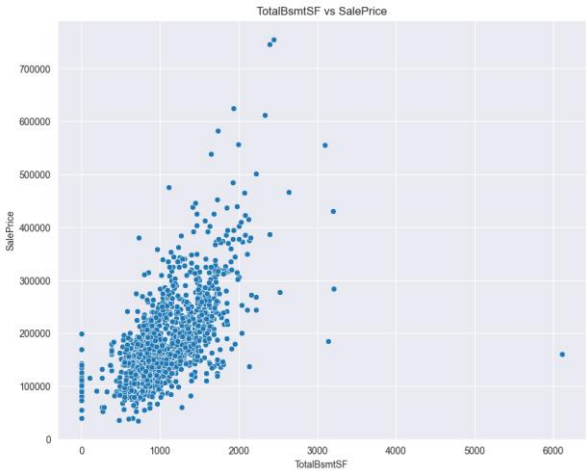


Fig 2d. ToalBsmtSF vs SalePrice

Fig. 2d is a scatter plot between TotalBsmtSF and saleprice. There is an apparent relationship between two features, the price of the houses increases as the the size of basement increases. There is one outlier detected from the graph, which was marked and dropped in feature engineering.

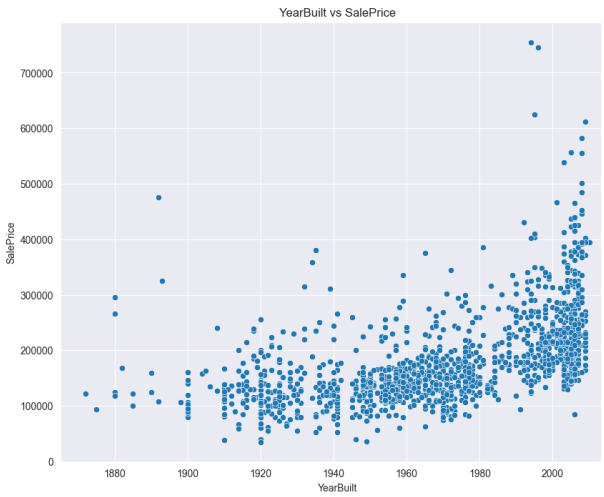


Fig. 2e: YearBuilt vs SalePrice

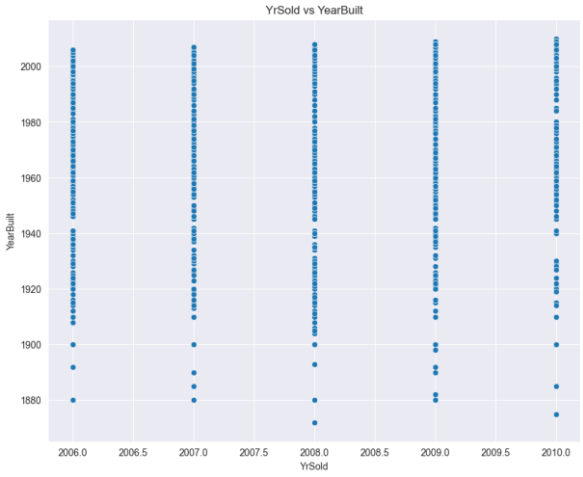


Fig. 2f: YrSold vs YearBuilt

Fig. 2e and 2f are scatter plot between YearBuild and saleprice, and YrSold with YearBuild respectively. New property tends to have a higher price, and the transactions of old houses are uniformly distributed.

2.1.2. Feature Engineering

Since this is a regression type of problem, we have to transform the non-numerical labels into numeric values. By using one-hot encoding a dummy-coded dataframe is returned. Moreover, there are null values in some columns that need to be taken care of. In high-dimensional datasets, the curse of dimensionality can occur especially after one-hot encoded, where the sparsity of data increases as the number of dimensions increases and thus causing difficulties in data analysis. Therefore, data preprocessing is crucial in

addressing this issue as it involves selecting or creating a subset of relevant features that capture the essence of the data while reducing the number of dimensions.

Below columns were dropped

1. More than 10% missing values
2. Correlation less than or equal to 30%
3. Highly correlated variables
4. Outliers that have been highlighted in EDA

There is only one missing value in Electrical column, so we decided to remove the observation and keep the variable. Missing values in MasVnrArea column are replaced with 0, as null value means there is no masonry veneer for these houses. Outliers that have been highlighted in EDA are also removed.

Noted that we will be building the multiple linear regression, it is essential that the features have to be normally distributed. In fig. 3 the original sale price is not normally distributed and is right-skewed (mean and median are greater than the mode). This implies there were more houses sold cheaper than the average price. Log transformation was applied to remove the skewness of the original data.

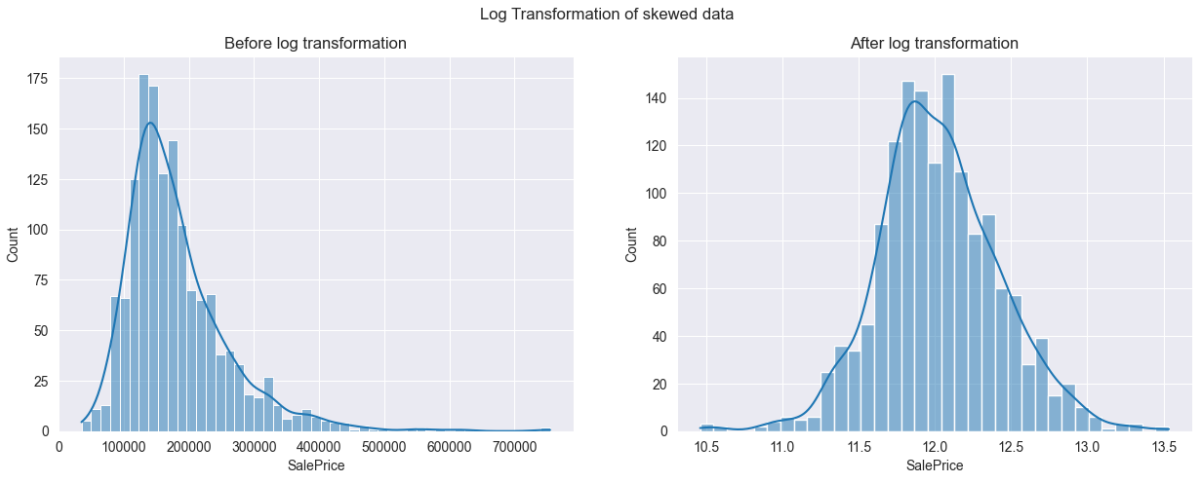


Fig. 3: Log Transformation of Skewed Data

Up to this stage it is close to the end, and we need to split the dataset into training and validation sets. As a failure analysis, our first try was to do the feature engineering separately on train and test files with the same procedures. However, the trained model failed to make predictions in the test data. This is because when we do the one-hot

encoding separately on train and test set, there are some unique responses on some columns that are splitted into a new column, but these columns do not exist on the other side. Therefore, we decided to start over by concatenating the two files into one merged dataset, and split it back to the original dataset after all the feature engineering. While we also divided 80% of the data in train file as training set, and remaining 20% of the data as validation set to evaluate the model.

2.2. "Hope to have" Data

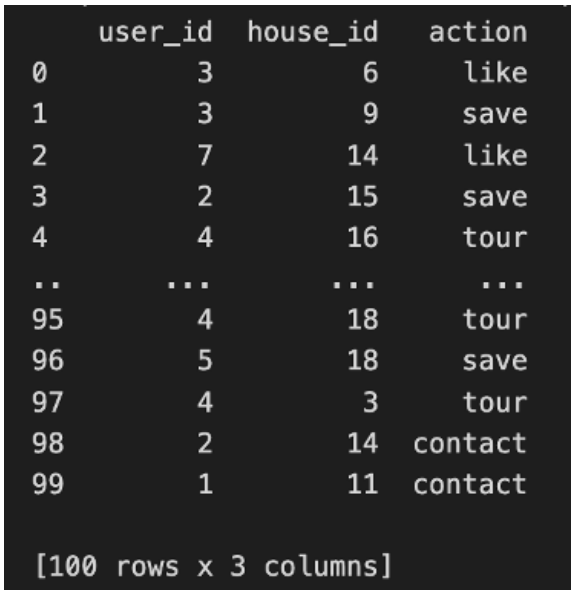
In order to make recommendation based on the user's preference, a user action log is required to analysis users' behaviours. Assume there are four buttons when viewing a house, including like, save, request house tour, contact agency. If the user only has one action to the house, for example gives a like to the house, it may consider he/she has little interest in it. If the user has more than one actions to the house, for example gives a like, requests house tour and contacts agency, it may consider he/she is very interested in it. In view of this, assume their action is showing their interest in the house. In order to measure their interest to the houses, the action list will be converted into rating list for further implementation of collaborative filtering. Therefore, the user's activity or action logs is the first set of information needed. A similar dataset from Kaggle has been found:

item_id	user_id	event_type	create_tim...
00062bc5-2535-4b1e-bbcb-228526c990b8	182aa519-83a8-848f-84a1-8697046d84c2	seen	2020-02-03 15:47:25.273977
00062bc5-2535-4b1e-bbcb-228526c990b8	189a081a-ae0f-499d-9092-01758d93fa7f	seen	2020-02-04 20:19:31.040304
00062bc5-2535-4b1e-bbcb-228526c990b8	189a081a-ae0f-499d-9092-01758d93fa7f	sent_catalog_link	2020-02-04 20:19:00.110416
00062bc5-2535-4b1e-bbcb-228526c990b8	189a081a-ae0f-499d-9092-01758d93fa7f	visit_request-canceled	2020-02-04 20:54:31.595305
00062bc5-2535-4b1e-bbcb-228526c990b8	189a081a-ae0f-499d-9092-01758d93fa7f	visit_request-new	2020-02-04 20:20:15.918646

Fig. 4: user activity dataset from Kaggle, retrieved from https://www.kaggle.com/datasets/arashnic/property-data?select=user_activity.csv

After getting the user's activity dataset like this, drop the irrelevant column but remaining item_id, user_id and event_type. The next step is to filter out all the 'seen' or

unrelated rows in even_type column because it is assumed that the 'seen' or other unrelated activity cannot show the user's interest in house. Then the action list will be formed, and rename the columns as house_id, user_id and action. Python is leveraged here to randomly generate user ID, house ID and action for creating a toy dataset:



	user_id	house_id	action
0	3	6	like
1	3	9	save
2	7	14	like
3	2	15	save
4	4	16	tour
..	...	...	...
95	4	18	tour
96	5	18	save
97	4	3	tour
98	2	14	contact
99	1	11	contact

[100 rows x 3 columns]

Fig. 5: Toy dataset: user action logs

Notice that there could be duplicated user_id because a user can have more than one actions towards the same listed house.

2.3. Futuristic data mining application

After getting the action logs like this, the next step is to convert it to a rating matrix. To generate rating matrix, it is assumed that each action represents one point, and then add up all the actions of a user to a house. Therefore, the rating is between 1 to 4 otherwise 0 if the user only seen but no actions were taken in this case. The toy data generated above was used to further stimulate the rating matrix. The row of the matrix represents the i^{th} user_id, while the column of the matrix represents j^{th} house.

	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013
0001	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0
0002	0.0	1.0	1.0	3.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0
0003	0.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0
0004	0.0	1.0	2.0	1.0	1.0	1.0	2.0	0.0	1.0	0.0	1.0	1.0	0.0
0005	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
0006	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
0007	0.0	1.0	1.0	2.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0
0008	0.0	1.0	1.0	0.0	0.0	2.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
0009	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
0010	0.0	1.0	0.0	2.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0

Fig. 6: Toy dataset: user rating matrix

The limitation of rating matrix is the weighting assigned to each action. For the simplicity in the implementation, the weight of each action is assigned equally. However, in the real world, this weighting is not ideal. For example, if user 01 gives a like to house 01, and contact agency to house 02 at the same time, it cannot tell user 01 show the same interest in both house 01 and 02. It is believed that many people think that contact agency will show more interest in the house than just giving a like.
As a result, the relationship between user's action and rating list is needed to be studied for finding out their weighting and make the rating more reasonable and credible.

In order to find out the relationship between users' action, the factor analysis is applied. Python is leveraging here to simply perform each step by randomly generating the matrix initially. First, create a matrix in which each row corresponds to a user and each column to a specific action. The number or frequency of times the associated action was performed by the corresponding user is contained in each cell of the matrix:

```
array([[[3, 2, 3, 1],
        [1, 3, 1, 3],
        [2, 1, 4, 2],
        ...,
        [2, 0, 0, 2],
        [3, 0, 0, 4],
        [0, 2, 4, 3]],
```

Fig. 7: user action matrix

The next step is to calculate the correlation matrix between the actions in the dataset to

better understand how they are related to each other. Then, apply Principal Component Analysis (PCA) to identify the underlying factors that explain the relationships between the actions, and extract the factors that explain the majority of the variation in the data. Kaiser's criterion (Braeken and Assen, 2017) is utilized to determine the number of factors to extract. According to this approach, only extract factors with eigenvalues higher than 1.0. The number of factors with eigenvalues greater than 1.0 can be determined by computing the eigenvalues of the correlation matrix or the covariance matrix.

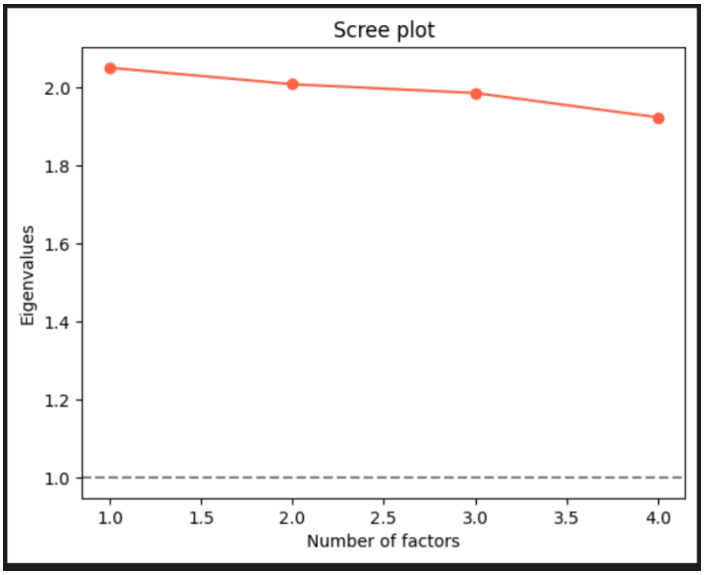


Fig. 8: example of number of factors being elected by Kaiser rule

Last but not least, look at the factor loadings to see how each action relates to the underlying variables and use factor loadings to adjust the weighting of each action.

```
[[-0.50580997 -0.07270783  0.76635889  0.38931209]
 [-0.27426493 -0.55699541  0.15636762 -0.76816927]
 [ 0.75782966  0.08398433  0.6131329  -0.20666129]
 [-0.30761865  0.82305304  0.11095613 -0.46437397]]
```

Fig. 9: example of factor loadings (each column represents one factor)

For simplicity, the weighting can be obtained by multiplying the original weights of each action by the weights obtained from the factor loadings for each principal component. The original weights should be determined before performing the PCA. For example, assign 0.2 to like, 0.2 to save, 0.3 to request tour and 0.3 to contact agency. Finally, normalize the adjusted weights add them up to get the final rating. The final part of determining the weighting of actions is not idea at all because it ignores the other factors that may contribute to the weighting, so further study for the weighting is needed once

the real factor loadings is obtained.

3. Implementation and Discussion

```
============================================================
Model Comparison on Training set
                              RMSE      R^2   Adj R^2
Random Forest             0.060547  0.976198  0.969105
Linear Regression baseline  0.092045  0.945092  0.932568
Decision Tree CV          0.094781  0.941672  0.924291
Ridge                     0.097552  0.938325  0.924257
Lasso                     0.098945  0.936552  0.922080
Decision Tree baseline    0.110024  0.921402  0.897980
Decision Tree pruned      0.148674  0.856747  0.824073
============================================================
Model Comparison on Validation set
                              RMSE      R^2   Adj R^2
Lasso                     0.112496  0.923196  0.702000
Ridge                     0.112738  0.922865  0.700715
Linear Regression baseline  0.114040  0.921073  0.693763
Random Forest             0.146562  0.872162 -0.611865
Decision Tree CV          0.185876  0.794382 -1.592578
Decision Tree baseline    0.200818  0.759995 -2.026152
Decision Tree pruned      0.212041  0.727130 -0.058734
```

Table 1: Model Comparison

3.1 Linear Model

A multivariate linear regression model is applied with all default settings as the baseline model. A coefficient of determination (r^2) is applied to see how well the model replicates the predictions, and Root Mean Squared Error (RMSE) is applied to evaluate the model accuracy at the same standard as the competition in Kaggle. From the model comparison table, the baseline linear regression model has an RMSE of 0.092 and an r^2

of 0.945 on training set, indicating that the average prediction error was relatively low and explained over 90% of the total variance of the data. However, the model's performance on the validation set was not as good as on the training set, with a higher RMSE of 0.114 and a lowered adjusted r^2 of 0.69 only, indicating that the model may not generalize well to unseen data.

Therefore, two regularization methods were introduced - Ridge and Lasso, trying to penalize the coefficients. From the table both models have similar performance on the training set. The Ridge model has an RMSE of 0.097, an r^2 of 0.938 and an adjusted r^2 of 0.924, while the Lasso model has an RMSE of 0.0989, an r^2 of 0.936 and an adjusted r^2 of 0.922. However, the Lasso model slightly outperforms the Ridge model on the validation set, having an RMSE of 0.1124, an r^2 of 0.923 and an adjusted r^2 of 0.702, while the Ridge model has 0.1127 in RMSE, 0.9228 in r^2 and 0.7007 in adjusted r^2. Overall speaking the Lasso model is slightly better than the Ridge Regression, this might be caused by the shrinkage of some coefficients to zero in Lasso Regression which further filtered out some unnecessary features for us.

3.2 Decision Tree

Next, a simple regression model was applied - decision tree to model the house price. One advantage is its high interpretability, it is easy to plot a flow-chart-like structure which is easy to follow and understand. There are several trees in the comparison table, the pruned tree is the first tree we have built, and the data is the same as the one used in linear model, that is considered as a pre-pruned tree. The pre-pruned tree model has the worst performance among all trees, having the highest RMSE in both training and validation set, and the lowest r^2 value as well. We think that the model might be underfitting due to the feature selection in the previous section, so we decided to increase the amount of features in the dataset to make it more generalized.

After reworking the feature selection, the dimension of the data increased from 216 to 268 features. The tree model was retrained with a few trials, and finally a maximum depth of 7 was picked as the baseline decision tree model. From the comparison table both RMSE and r^2 have been improved on training set compared with the pruned tree model, however a drastic drop in the performance on validation set is observed. This time the challenge becomes overfitting, which is a good news compared with the underfitting model. Two techniques were applied in order to optimize the hyperparameters.

Firstly, cross-validation was used to find the optimal hyperparameters for the decision tree model. This technique involves dividing the data into multiple subsets, training the model on each subset, and evaluating the performance on the remaining subset. 5 fold cross validation have been applied, and the best set of hyperparameters is basically the max value of the parameters provided. Therefore, the resulted best model generated here might be overfitting. From the comparison table, the CV model has a much lowered RMSE on training set compared with baseline and pruned decision tree models, but on the validation set the RMSE increased to 0.1858, which shows an improvement from baseline and pruned models, but it is still overfitting.

Secondly, random forest was applied to the dataset. This technique involves building multiple decision trees and averaging their predictions to reduce the variance of the model. By doing this, we can improve the performance of the model and reduce the risk of overfitting. From the comparison table, Random forest outperformed both linear regression and decision tree models on training set, but the overfitting problem still have not yet solved. It has the lowest RMSE of 0.06, the highest r^2 and adjusted r^2 of 0.976 and 0.969 on training set, but an RMSE of 0.1465, an r^2 of 0.872 and an adjusted r^2 of -0.611.

A comparison of the different models used in this project shows that linear regression with regularization had the best performance, with an RMSE of 0.112496 and an r^2 of 0.923 on the validation set. This indicates that the linear regression model has a good balance between bias and variance, and can better generalize to unseen data. The random forest had the best performance on training set but that does not mean the best model in predicting the house price due to the overfitting issue, with much higher RMSE and lower r^2 values on the validation set.

3.3 Collaborative Filtering

User-based collaborative filtering technique is applied to generate the house recommendation. The collaborative filtering uses interactions between users and houses to identify similarities between houses and make recommendations. The interaction mentioned here is the rating matrix, which is the data hope to have. As mentioned above, the rating matrix has been converted from the action list. The key advantage of presenting data as a rating matrix is to use matrix factorization techniques to identify latent factors that account for the observed user-item ratings. These latent components can be used to forecast missing ratings for users and products that were not included in the initial matrix.

The implementation of collaborative filtering is shown below:

STEP 1: Standardize the rating matrix so that each column represents a house and each row represents a user's ratings.

STEP 2: In the normalized rating matrix, determine the cosine similarity between each pair of rows. In the similarity matrix that results, each entry (i, j) stands for the cosine similarity between users i and j.

STEP 3: Utilizing the similarity matrix, determine the weighted average of the ratings for each house for a specific user. This entails averaging these ratings across all users by multiplying each rating by the associated cosine similarity.

STEP 4: Sort the properties according to their weighted ratings and then suggest the top 10 properties to the user.

The main challenge of this technique is the rating matrix which has discussed before. Another challenge is the sparsity problem. In real-world scenarios, a sparse user-item (i.e., rating) matrix results from a large number of users only rating a small portion of the available things. Since there might not be enough information to precisely estimate user preferences, this can cause accuracy problems. To address this problem, we are determined to use matrix factorization. By breaking the rating matrix into two or more latent factor matrix, matrix factorization can be used to estimate missing ratings.

3.4 K-nearest Neighbors

KNN is applied to generate house recommendation based on the similar house feature. The main challenge of the collaborative filtering technique is the cold-start problem, which occurs when a new user or house is added to the system and there is not enough data available to make accurate recommendations. More specifically, when a new user joins the system because there is no past information about their preferences available. As a result, the system can have trouble making recommendations that are helpful until the user has given the algorithm enough information to train it. The same problem happens in the new item. Therefore, the hybrid recommendation systems are applied to combine the collaborative filtering and KNN to address the cold start problem. The implementation of KNN is shown below:

STEP 1: Once the user clicks in a house and view it, the viewed house is considered as the input.

STEP 2: Prepare feature vectors and normalize them so that each feature has a similar range and scale. The step of feature selection has done and discussed in the previous section.

STEP 3: Calculate the distance between the feature vectors for each pair of houses. The distance metric will be discussed later.

STEP 4: Identify the k-nearest neighbors for each house. These are the houses that are most similar to the house in input, based on the defined features.

STEP 5: Generate the top 10 recommendations by selecting houses that are similar to the house in input, based on the k-nearest neighbors.

The main challenge of the KNN is the calculation of distance metric. For example, there are two houses, the only two main differences are the number of bedrooms and price. The difference in number of bedrooms is 1 while the difference in price is 1,000USD. From a numerical point of view, 1 is indeed smaller than 1000, but in a sense, the weight of 1 is very large here, but the calculation of KNN ignores this. To address this problem, the house feature data frame will be normalized to a comparable size.

Another challenge is the selection of distance metrics. The house feature vectors is mixed data types, including categorical and continuous data. However, Euclidean distance only considers continuous data. Although the feature vector has already become one-hot, a new problem arises is the sparsity. Therefore, a more powerful distance metric is utilized called Gower distance without normalizing the feature vectors. Gower's dissimilarity coefficient is used to manage various types of variables (Gower, 1971). By default, Gower's dissimilarity coefficient is extended using the work of Kaufman and Rousseeuw (1990). A weighted sum of the final differences for the ith and jth units is calculated for each variable:

$$d(i,j) = \frac{\sum_k \delta_{ijk} d_{ijk} w_k}{\sum_k \delta_{ijk} w_k}$$

.

d_{ijk} represents the distance between the *i*th and *j*th unit calculated considering the *k*th variable, while is the weight given to variable *k.* The logical columns are considered as asymmetric binary variables, for such case $d_{ijk} = 0$ if $x_{ik} = x_{jk} = TRUE,$ 1 otherwise; The factor or character columns are regarded as categorical nominal variables and $d_{ijk} = 0.$ If $x_{ik} = x_{jk},$ 1 otherwise; the numeric columns are considered as interval-scaled variables and

$$d_{ijk} = \frac{|x_{ik} - x_{jk}|}{R_k}$$

where R_k is the range of the kth variable; the ordered columns are considered as categorical ordinal variables and

$$z_{ik} = \frac{(r_{ik}-1)}{max(r_{ik})-1}$$

.

When it comes to the weight δ_{ijk} that $\delta_{ijk} = 0$ if $x_{ik} = NA$ or $x_{jk} = NA$; $\delta_{ijk} = 0$ if the variable is logical and $x_{ik} = x_{jk} = 0$ or $x_{ik} = x_{jk} = FALSE$; $\delta_{ijk} = 1$ otherwise. If there is NA or False, it does not contribute to distance computation.

The following shows the computation of the Gower distance between two houses. For numeric feature vector, the formula is the absolute difference divided by the range. For non-numeric variables, it equals to 0 if the elements of them are different otherwise 1. Here is an example of the calculation of Gower distance. Assume there are three data types, including categorical, continuous and ordinal. The dataset includes the number of bedrooms and price because they have very different scales. Despite this, the Gower distance still can handle this dataset without normalization. Suppose the feature vectors columns are:

House_id	House Type	Price ($)	Bedrooms
1	Condo	250,000	1
2	House	500,000	3
3	Townhouse	400,000	2
4	House	600,000	4
5	Condo	300,000	1
Range	NA	350,000	3

To calculate the Gower distance between the first two houses in this dataset, the first step is to calculate the absolute differences between the values of each variable:
House Type: (different type) = 1
Price($): |250,000 - 500,000| / 350,000 = 0.56

Bedrooms: |1 - 3|/3 = 0.67
Therefore, the Gower distance = (1 + 0.56 + 0.67) / 3 = 0.74. After iterating the calculation in all pairs, a distance matrix will be obtained. The distance matrix can be further used for KNN to generate house recommendation. Once house recommendation is generated by KNN and collaborative filtering, the duplicated recommendation will be removed and show the remaining recommendation to user.

4. Conclusion

To summarize, the implementation of machine learning algorithms has shown the potential for building an effective recommendation system for the real estate market. For modelling the house price, various techniques such as cross-validation and random forest managed to improve the performance of the models. More accurate predictions could be achieved by further finetuning the parameters or trying more advanced models. On the other hand, the combination of collaborative filtering and KKN techniques provide a personalised and diverse recommendation for users so as to improve the quality of the systems. It is hoped that the innovation of the recommendation system will benefit both users and agents in the future.

5. Appendix

Detailed code could be accessed on GitHub.
https://github.com/nod32ba1/MATH3836-data_mining

6. Reference

(1) Bock, T. (2022, September 13). Factor analysis and principal component analysis: A simple explanation. Displayr. https://www.displayr.com/factor-analysis-and-principal-component-analysis-a-simple-explanation/

(2) Braeken, J., & Van Assen, M. A. (2017). An empirical kaiser criterion. Psychological Methods, 22(3), 450-466. https://doi.org/10.1037/met0000074

(3) D'Orazio, M. (2022). Statistical Matching or Data Fusion. *gower.dist*, 3. https://github.com/marcellodo/StatMatch/tree/master/Tutorials_Vignette_OtherDocs

(4) Gharahighehi, A., Pliakos, K., & Vens, C. (2021). Recommender Systems in the Real Estate Market—A Survey. Applied Sciences, 11(16), 7502–. https://doi.org/10.3390/app11167502

(5) Gower, J. C. (1971), "A general coefficient of similarity and some of its properties". Biometrics, 27, 623–637.

(6) Jaadi Z. (2021, April 1). *A step-by-step explanation of principal component analysis (PCA)*. Built In. https://builtin.com/data-science/step-step-explanation-principal-component-analysis

(7) Kaufman, L. and Rousseeuw, P.J. (1990), Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York.

(8) Makwana, A. (2020, December 26). *Understanding recommendation system and KNN with project — Book recommendation system*. Medium. https://aman-makwana101932.medium.com/understanding-recommendation-system-and-knn-with-project-book-recommendation-system-c648e47ff4f6

(9) Miles, J. (2014). R Squared, Adjusted R Squared. In *Wiley StatsRef: Statistics Reference Online*. John Wiley & Sons, Ltd. https://doi.org/10.1002/9781118445112.stat06627

(10) Möbius. (n.d.). *Rental properties collaboration data*. Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com/datasets/arashnic/property-data?select=event_types.xlsx

(11) Montoya, A. (2016). House prices - advanced regression techniques. House Prices. Retrieved May 2, 2023, from https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques

(12) Turing. (2022, July 29). *How collaborative filtering works in recommender systems*. Hire the World's Most Deeply Vetted Remote Developers | Turing. https://www.turing.com/kb/collaborative-filtering-in-recommender-system

(13) Yuan, X., Lee, J.-H., Kim, S.-J., & Kim, Y.-H. (2013). Toward a user-oriented recommendation system for real estate websites. *Information Systems (Oxford)*, *38*(2), 231–243. https://doi.org/10.1016/j.is.2012.08.004