Data Mining Project Report

Recommendation system for Rentals on Airbnb

Group 14 : Arkapriya Paul , Abhinav Singh

Project Objective :

To come up with an Airbnb listings recommendation system based on users' preference for amenities and within their announced budget.

Data Fetching and Preprocessing:

We got .csv files giving the details of airbnb listings from insideairbnb.com. We chose 2 cities from the USA - Los Angeles and Seattle. There were several missing values in the data, which we dealt with. There were a large number of attributes, from which we chose those relevant to our project - listing id, address, amenities, price.

EDA:

For the exploratory data analysis, we considered two cities - Seattle, a business hub, and Los Angeles, a tourist destination, and compared the nature of listings in these cities. Some observations are listed below:

- We made a bar graph of the number of listings per neighborhood group in Los Angeles and Seattle, and checked this against other attributes. For Seattle, majority of the house types are entire apartments, and a significant number of private rooms. Shared rooms are negligible except in a few neighbourhoods . For Los Angeles, majority of the listing types are entire apartments or homes, with the next being private rooms and shared rooms in that order.
- We made a price vs room-type box plot for Los Angeles and Seattle. In Seattle, most private rooms and entire homes have minimum night stays of about a week. There are outliers with a month as minimum nights. Shared rooms have minimum stay of less than 5 days. In Los Angeles, most private rooms and entire apartments have minimum nights stay of less 30 days. There are a significant number of outliers with > 50 days, some even going upto >300 days. For shared rooms, most listings have less than 10 nights as minimum days and nearly all within 30 days.
- A price vs minimum nights boxplot was made for Seattle and Los Angeles. For Seattle, most of the listings had a minimum nights stay of 1-8. And a smaller proportion of listings with nights greater than 10. The listings with minimum night stay of 2-6 had the highest mean and upper quartile prices. Listings with 1-3 minimum nights stay had significant number of outliers with higher prices. For Los Angeles, prices are higher and much more varying than for listings than in Seattle.

The mean price for listings is relatively low and uniform across the minimum number of nights. But there are a large number of outliers with very high prices, some even going till 10000. Most of the outliers lie below the price range of 5000 and before the minimum night stay of a week.

- Did Price variation analysis on randomly picked houses at various locations of Seattle and Los Angeles. In Seattle, prices increase from january reaching a peak in July, and then go down. The minimum is in october after which they, rise again towards the December. In Los Angeles, prices remain relatively low from May to November. Then there is a rise from December to April.

Recommendation System

Based on a given user query, we had to suggest him a list of places (Airbnb listings) that he could rent in the city of his choice.
We decided to go with facilities offered in each listing as the selecting criterion, and then filter these according to the user's budget. Later, we incorporated the user's priority for different facilities. A K- Nearest Neighbours based algorithm was chosen so that we could compute which places had facilities closest to user's preferences.

Data transformation -

The facility set attribute given in the detailed listings file was split and a binary data matrix was formed, based on existence of the facility in the listing. This could be used to compute distance of each listing from the user query, and give best matches for availability of facilities.

Function mtabulate of qdaptools package was used to do this - it returns a data frame with columns equal to number of unique elements and the number of rows equal to the the original length of the used data frame. The facilities column for each tuple was split and each tuple now had n columns with 0 or 1 - n being the number of unique values/facilities across the data frame.

The user query was simulated by generating an n-tuple of random values ranging from 0-1 for the non-priority recommendation, and 0-10 for the priority recommendation.

Non- Priority Based Recommendation Algorithm

To simulate the user query we first generate an n-tuple of random values ranging from 0-1 - 0 meaning no preference for that particular facility and 1 meaning the user prefers that facility to be present in a listing.

Two different KNN algorithms were made, using two different distance metrics - Cosine similarity and Euclidean distance.

Cosine Similarity is useful when we are dealing with binary data frame where all features are Assigned equal weightage. It is defined as follows:-

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2}$$

Here A:-features,B:- query of same dimension say n and the denominator consists of the product norm of A and B.

Another metric is the Euclidean distance metric which between a vector p and q is defined as:-

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

If we want the unity coefficient along every squared term can be changed to different value, thereby introducing the concept of weighted distance

Priority Based Recommendation Algorithm

To simulate the user query we first generate an n-tuple of random values ranging from 0-1. Then, the facilities with priority 1 are allotted random values from 1 to 10 - 1 being the least priority and 10 being the highest.

Cosine similarity does not fare well when a priority/weightage are assigned to the features.
Due this reason our further work was only based on Euclidean distance metric which accounted for absence of weighted features rather accurately

The first round of KNN takes into consideration all of these priorities. A set of R1 recommendations are generated, based on least Euclidean distance between the query and the attribute set.

Next, we ask the user for his budget range and filter these recommendations to fit into his price range. If the recommended listings do not fit into his price range, we move to the next round of KNN. Alternatively, the user can also choose to see the next set of recommendations.

For the second round of KNN, we consider only those facilities which have priority greater than or equal to the threshold (here, we take the average of max and min priority assigned to the queries). We take the R1 recommendations and compute closeness to the user's high priority preferences. On this basis, we generate R2 recommendations.

Problem Faced and Rectification:

Initially for implementing the priority concept, we multiplied the weights/priority to entire column of a feature. However it turned out for this configuration for any query, the house which had no feature will be recommended, even though it gave least euclidean distance, so it was discarded.

Results

For the non-priority based KNN algorithm, one set of recommendations is generated.

For the priority based KNN algorithm, two sets of recommendations are generated. These sets also consider the budget of the user.

Limitations

We cannot test the recommendation system as we do not have the data for acceptance and ratings of the lhe recommended listings by real time users.

We had considered taking facilities like nearby markets, transport systems, into consideration for recommending listings in locations convenient to the user. This would

require us to reverse geocode the latitude and longitude coordinates of each listing to its address. But we could not get nearby landmarks, which would serve as the marking point for finding other nearby facilities like markets and subway stations.

Learning Outcomes

One of main learning outcomes was that we understood the principles of data mining in more detail, and how they may be applied to solve real world problems.

We got to learn how to extract relevant data from large datasets and use them to solve the problem. We figured out additional features we could add to our recommendation algorithm (problem statement) - to make it more user friendly and to include more of his wants while choosing a listing, and learnt about the limitations one may face.

Finally, we learnt how to go about a project - from framing a proper problem statement, getting data, extracting information from it, to choosing the proper algorithm and streamlining the code.