Hybrid Recommender Systems for Personalized Travel Destinations

Yasaman Ensafi

Table of Contents

Context

Nowadays, online reservations have become very popular and travellers find it much easier to book hotels of their choice online.

However, people have a difficult time choosing the hotel that they want to book. Therefore, the recommender system comes into play.

The goals of this project are to:

- Build a hybrid recommender system using state-of-the-art models such as Wide and Deep, DeepFM, and XDeepFM
- Identify similar hotel clusters for each of them.

The Expedia dataset

Source

This data is anonymized and it is from the Kaggle Expedia Hotel Recommendations competition.

train.csv

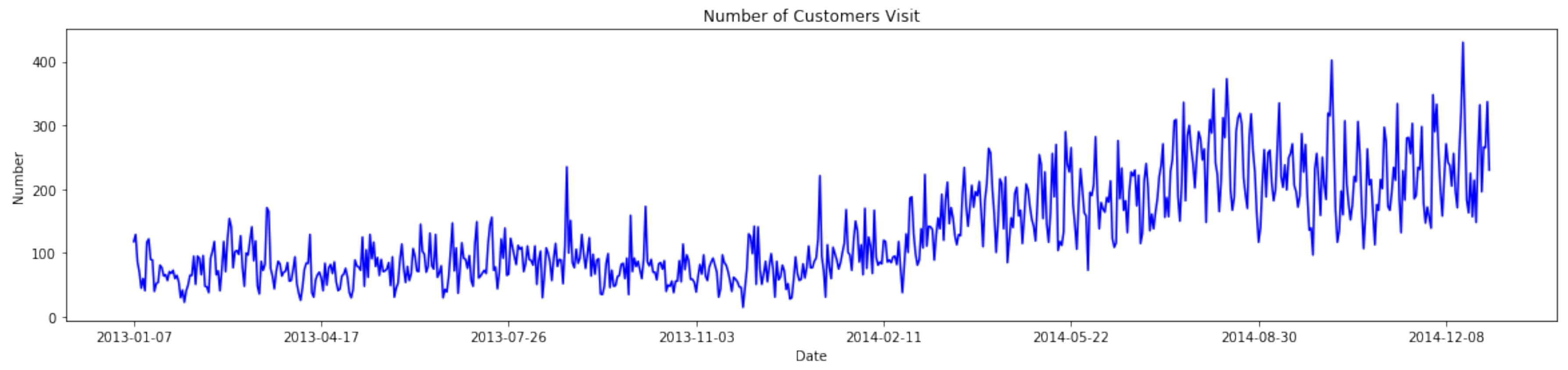
Train set consists of 37,670,293 rows from 2013 to 2014 and 24 features.

destination.csv

Consists of 150 latent features for each of the destinations recorded and the top 10 most correlated features were chosen to be used.

Data pre-processing and EDA

First of all, exploratory data analysis was performed to extract some useful insights.



The above figure represents the number of customers search between the years 2013 to 2014. We can observe the increasing trend.

The next step was to clean and pre-process the data.

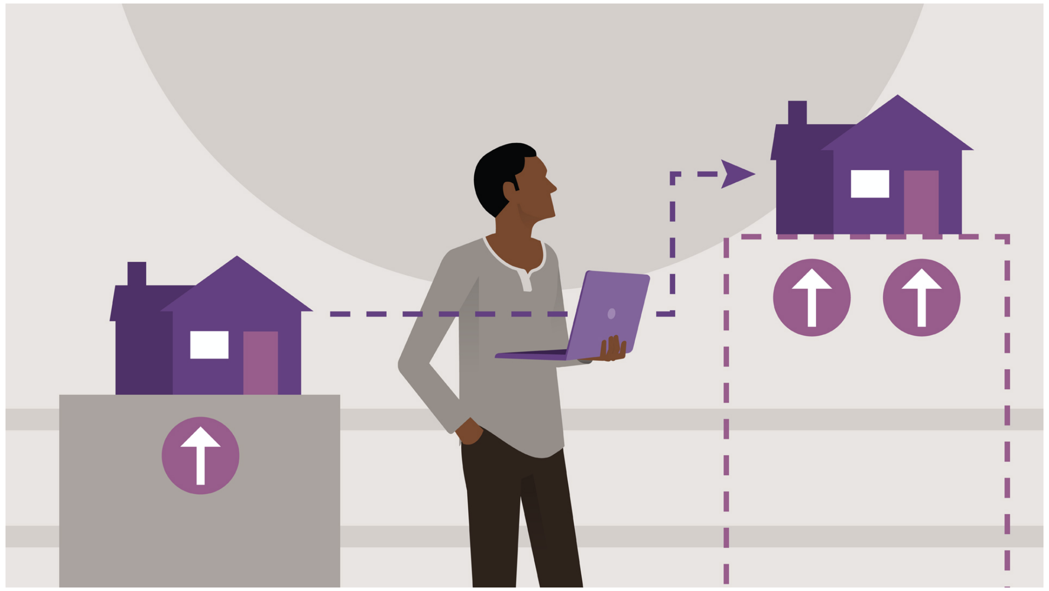
- For the users who ranked an item twice, only the maximum rating was kept.
- Log transformation was applied to the orig_destination_distance to make it more interpretable for the models.

Feature Engineering

The date time, check-in date and check-out date columns can not be used directly. Therefore, some features such as month, week, number of searches on the past two weeks, trip duration and North America holidays, were extracted from them.

Moreover, the family status extracted from the srch_adults_cnt and srch_children_cnt columns which represent the number of adults and the number of children specified in the hotel room.

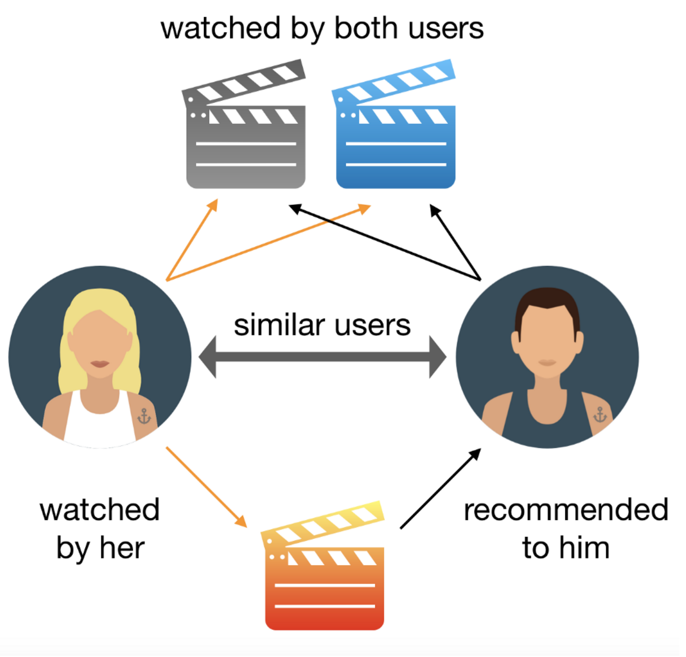
Also, the KMeans clustering algorithm was used on some features such as user_location_region and user_location_city to convert each of them to 2 clusters.

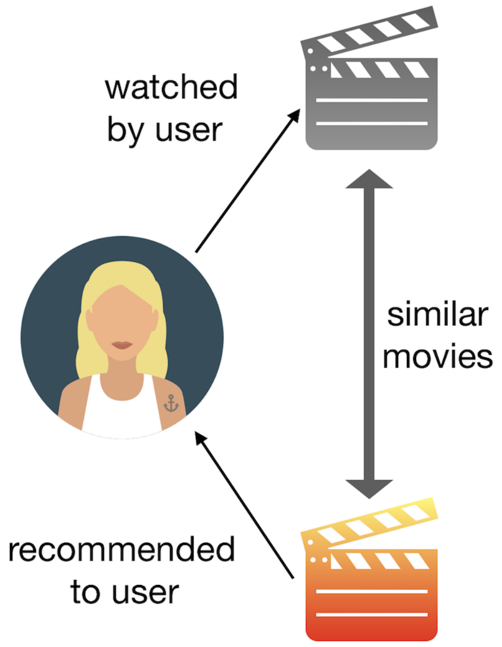


Collaborative-Filtering (Baseline)

The two basic architectures for a recommendation system are:

Collaborative-Filtering models **Content-Based models**





Content-Based models recommend items based on how similar this item is to the other items. On the other hand, Collaborative-Filtering models are entirely based on past behaviours and focus on the relationship between users and items.

We used Collaborative-Filtering model as the baseline algorithm.

First of all, user-item matrices for train and test set were created which can be used to calculate the similarity between users and items.

There are two types of Collaborative-Filtering mode:

	Item 1	Item 2	Item 3	Item 4	· · ·	Item M
User 1	X				X	
User 2			X			
User 3				X		X
User 4					X	
User 5	X			X		
User 6			X			
User 7	X	X				X
⋮		X				
User N			X			

For train set

	Item 1	Item 2	Item 3	Item 4	· · ·	Item M
User 1						X
User 2		X				
User 3					X	
User 4		X				
User 5						X
User 6				X		
User 7			X			
⋮	X					
User N						X

For test set

- Memory-Based CF by computing cosine similarity:

The Memory-Based model uses either user-user similarity or item-item similarity in order to make recommendations from the ratings matrix.

- Model-Based CF by using singular value decomposition (SVD) and ALS method:

These types of CF models are developed using ML algorithms to predict a user's rating of unrated items. To name a few of these algorithms, we can mention KNN, Singular Value Decomposition (SVD), Matrix Factorization (ALS) and etc.

Why Hybrid Recommender systems?

The problem with our baseline model (collaborative-filtering recommender system) was that we couldn't use more features of our large dataset.

Therefore, we decided to build hybrid recommender systems to be able to use not only user_id, item_id, and rating features, but also 20 more features that would give us more insights and better recommendations.

This models will also solve the problems of

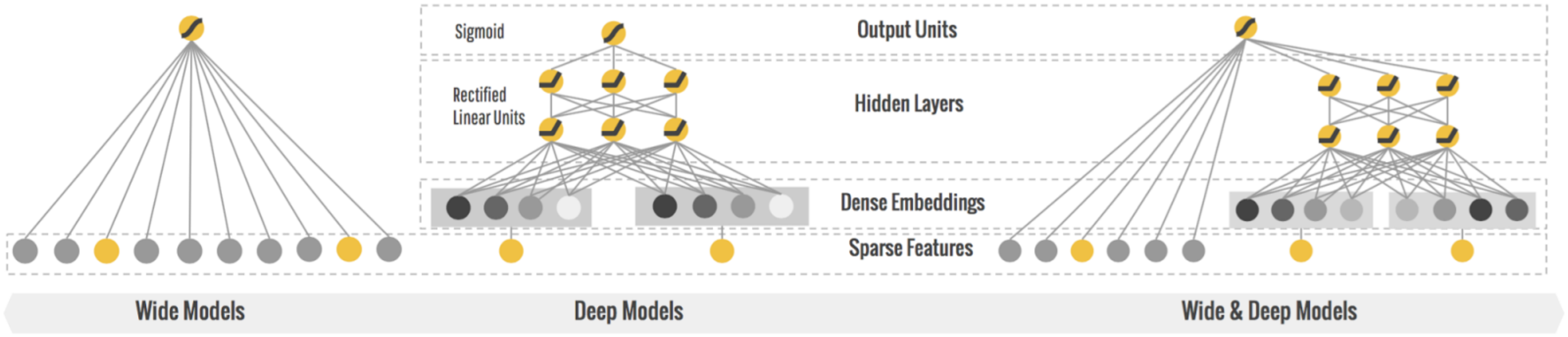
- Memorisation : Learning the co-occurrence items
- Generalization.: Exploring the new feature combinations that have never or rarely occurred in the past.

Wide & Deep

Wide & Deep is a hybrid model that joins trained wide linear models and deep neural networks to combine the benefits of memorization and generalization for recommender systems.

In this project wide and deep was implemented using DeepCTR package.

After the features were divided into sparse (categorical) and dense (continuous) features, Label Encoding for sparse features, and normalization for dense numerical features were applied



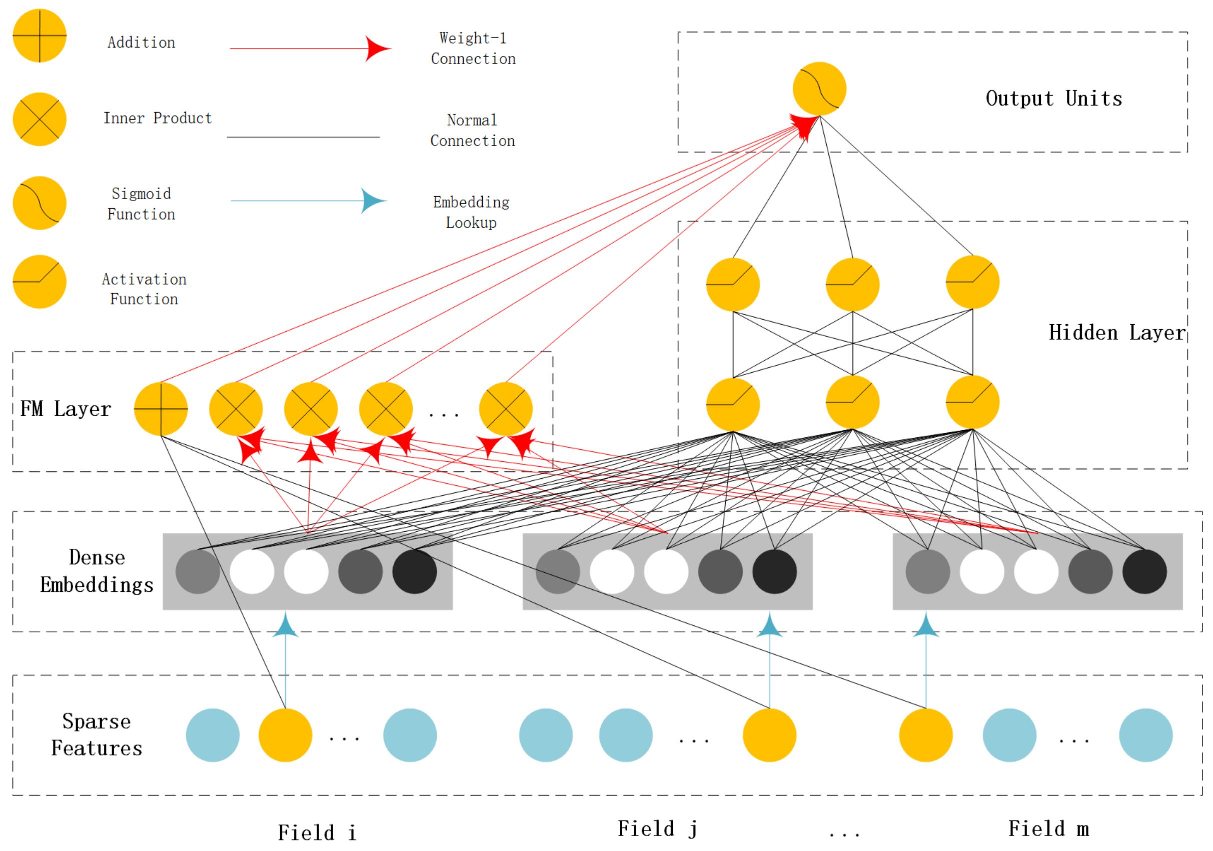
As the left side of the above figure shows, the wide network is a single layer feed-forward network which assigns weights to each feature and adds bias to them to model the matrix factorization method. The deep model is a feed forward neural network as shown in the above figure.

The combined wide and deep model takes the weighted sum of the outputs from both wide model and deep model as the prediction value.

DeepFM

Compared to the latest Wide & Deep model from Google, DeepFM has a shared raw feature input to both its “wide” and “deep” components, with no need for feature engineering besides raw features.

The wide component of DeepFM is an FM layer. The Deep Component of DeepFM can be any neural network.

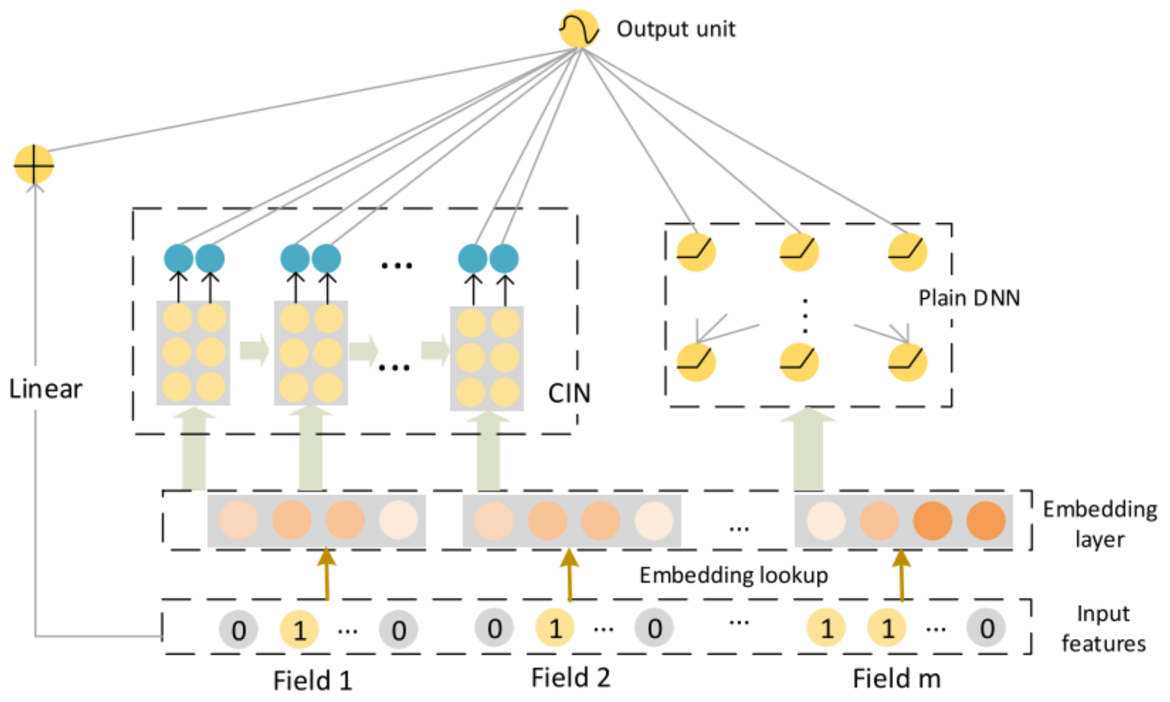


XDeepFM

XDeepFM (eXtreme Deep Factorization Machine) uses a Compressed Interaction Network (CIN) to learn both low and high order feature interaction explicitly, and uses a classical DNN (MLP) to learn feature interaction implicitly.

Same as the wide and deep and DeepFM:

- Features were divided to sparse and dense
- The model was defined and trained
- Hyper-parameter tuning was applied



Evaluation

Results

Model	RMSE	MAE	MSE	AUC
Wide and Deep	0.303	0.197	0.092	0.771
Deep FM	0.286	0.149	0.082	0.832
XDeep FM	0.285	0.285	0.081	0.832

- Mean Absolute Error (MAE)

The mean of the absolute value of the errors

- Mean Squared Error (MSE)

The mean of the squared errors:

- Root Mean Squared Error (RMSE)

The square root of the mean of the squared errors

- Area under the ROC Curve (AUC)

AUC measures the entire two-dimensional area underneath the entire ROC curve

Identify similar clusters (hotel types)

The Expedia hotel dataset is an anonymized dataset and the true values of some features such as item_id (or previously hotel_cluster) are hidden behind integer codes.

We tried to identify the location of some of the hotels with the help of user location country, region, city and most importantly, orig_destination_distance which represents the distance between the hotel and the user.

After finding the location of some of the hotel_markets (New York. Las Vegas, Cancún, London, Miami, ...) and combing the results with the most common Accommodation Types in that city (that extracted from Expedia website), we can guess the hotel type of that cluster.

- Hawaii - Hotel_market ID : 212
- The most common hotel_cluster = 0

$\rightarrow$ Assign "beach resort" to hotel_cluster = 0

Considering the information that we gained from Expedia website, we We took this finding further and after finding the top most similar clusters to 0, we assigned beach resorts to them too.

We applied this method to a few more clusters, came up with 10 categories such as apartment, business_hotels, bed_n_breakfast, and etc. and covered more than 60 item_ids out of 100.

Generating Output

- After finding the similar clusters based on the results of DeepFM model, the results were compared to the self-defined and confirmed their similarity.
- In the end, as an output, a csv file was generated which consists of item_id of the hotels and the three most similar clusters to them.
- It should be mentioned that for each item_id the recommended clusters that are in the same cluster category were excluded.

Item_id	Hotel type	Similar Hotel 1	Similar Hotel 2	Similar Hotel 3
0	beach_resort	hotel_resort	hotel_resort	private_vacation_homes
4	private_vacation_homes	apartment	apartment	casino_hotel
12	hosetel	bed_n_breakfast	condo	condo
25	motel	apartment	apartment	condo
52	hotel_resort	beach_resort	beach_resort	beach_resort

Output sample

Generating Recommendation

- Finally, 5 hotel clusters were recommended to each user.

user_id = 800	
Item_id (hotel_cluster)	**Hotel Type**
39	bed_n_breakfast
65	hotel_resort
77	private_vacation_homes
26	beach_resort
51	bed_n_breakfast

Recommendation for user_id 800

Thank you!