Report to Expedia hotel recommendations competition

Summary

Although this is a recommendations problem, I consider it as a classification problem and generate the top 5 hotel clusters based on the model confidence score to match the MAP@5 evaluation metrics. My best result 0.47138 rank the 25th in the public leaderboard so far. All the analysis, code and scripts can be found in https://github.com/parkerzf/kaggle-expedia.

Data understanding & Visual analysis

The goal is to obtain detailed insights into the data and the business instead of doing modeling blindly. Due to the huge size of the dataset, I import the data into the Postgresql database (link), in order to analyze the whole dataset more efficiently.

Time-based analysis (link): (i) do the daily stats query in the database (link); (ii) use ipython notebooks to analysis the daily/monthly stats visually.

Categorical fields analysis (link): (i) use plot.ly libary to visualize the histogram of categorical fields, which also includes the train/test breakdowns in order to compare their difference visually; (ii) use the python code (link) to query the database and automatically generate the html report.

Preprocessing & Feature Engineering

Baseline features (link): For baseline features, I apply direct feature engineering for hotel reservations but not for hotel clicks: (i) transform all the categorical fields into one hot encoding format, (ii) enrich date_time, srch_ci and srch_co fields (day of week, month, hour, booking window, length of stay, etc.), and (iii) fill the missing values in all the numerical fields with default values. In total, there are 86 baseline features.

Group-by/profile features (link): First, I group the hotels according to the following four categorical fields one at a time in order: orig_destination_distance (using data leak to compare with other solutions in the leaderboard), srch_destination_id, user_id and hotel_market, and select the top 5 hotel clusters. That is, first generate hotel clusters based on the categorical field of orig_destination_distance. If a reservation results in less than 5 clusters, then add srch_destination_id as the secondary categorical field. This procedure continues until the top 5 hotel clusters are generated. Then, I generate a hotel cluster ranking for each hotel reservation as the feature input of the classification models. Each reservation has 100 features due to the existence of 100 hotel clusters.

Modeling

Group-by Model (train, predict): Use the group-by/profile features only. If only orig_destination_distance is used to select the top 5 clusters, the resulting score is around 0.33. However, combining it with other groups by information results in a significant improvement to 0.47138.
Parameter tuning (link): Given that 2 hours on average are needed to train and predict a model, I limit my experiments by adjusting the input year and the click_weight. The conclusion is that all the data in 2013 and 2014 should be used, and click_weight should be set to 0.2 to get the best score.

Random Forest Model (train, predict): Use both baseline and group-by features. Only the booking data is used as training data. Due to the memory issue, the model is constrained with 30 trees. The result is 0.35069, not as good as the Group-by Model. The main reasons are three folded. First, Only the booking data is used as training data due to the memory constraint. Second, the ranking based features may not be the best way to integrate the group-by features. An alternative can be the weighted sum score, because the contribution of different group-by features are largely different. Third, only 30 trees are used and there is no execution time to run models with different parameters.

XGBoost Model (train, predict): The same setting as random forest model. I tried a XGBoost model with 50 iterations and the score is 0.39068. This model need to have more careful parameter tuning because XGBoost model is to overfitting.

To sum up, group-by model currently is the best comparing to the other two. However, with more model iterations and better features, I believe the other two can reach the same level as the group by model and be better than the group-by model finally.

Code Description

The code organization is in the readme file (link). To reproduce the best result, please follow the steps with python2.7 and Linux/Mac OS:

1. Clone the git repo, download the raw data and put them in here.
2. Configure the repo_path in utils.py.
3. Go to the base directory in git repo, run “make group”. It will install dependent python libraries, set parameters, train and predict with the group-by model. Make sure that you have the permission to use pip command. Virtualenv is recommended to initialize a new environment and reproduce the result without polluting your own python environment. The result file with Kaggle submission format will be generated in the models/results directory (link).

Conclusion & Future work

I very much enjoy the hackathon way of doing this interview with real data. The major challenge is the 8GB macbook environment that I use for this competition. I would like to continue the competition with the following directions in mind.

In terms of modeling:

1. The current solution does not make use of the latent descriptions, although I have already joined it with the main dataset. I plan to use the latent descriptions to enrich the features.
2. Try seasonal modeling. For instance, a model can be further decomposed into high season & low season models, or weekday & weekend models.
3. Split train set by year into train-2013 set and validation-2014 set, so that I can do parameter tuning, feature selection, model selection locally and less overfit to the test set. Since this is the time sensitive task, we cannot directly use cross validation but need to use the time-split cross validation.
4. Try the Field-aware factorization machine model and the Vowpal Wabbit model and blend them with the current solution.

In terms of engineering:

1. I will use Spark to re-implement the ETL, feature engineering, model training and prediction in order to speed up the model iterations.
2. I will test the src code based on tox and document the project based on Sphinx. Actually I have already included them in the git repo.

If you have any queries, please do not hesitate to contact me at feng.zhao@feedzai.com.