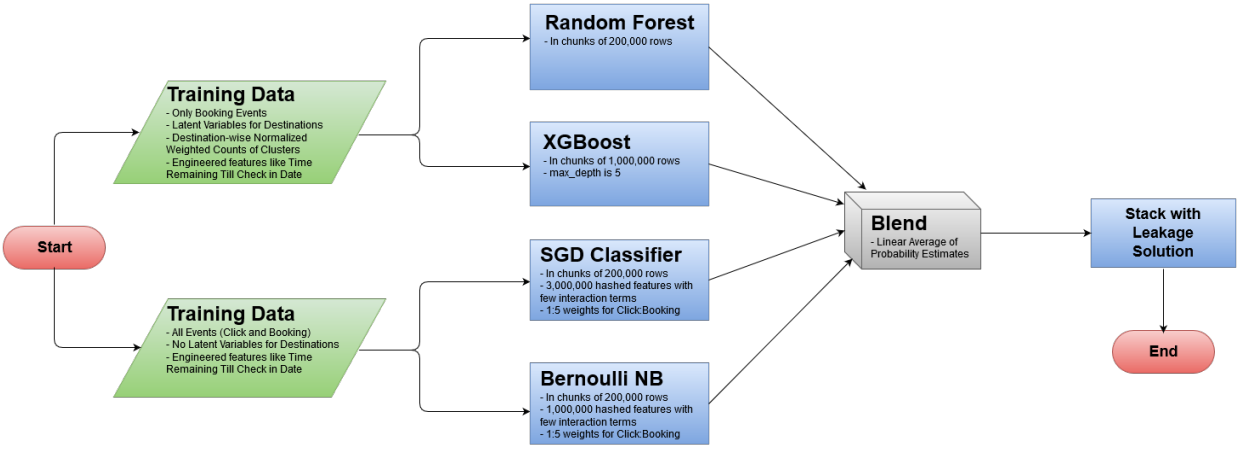
Kaggle: Expedia 酒店推荐

比赛链接: https://www.kaggle.com/c/expedia-hotel-recommendations

模型描述

整理by @寒小阳

整个解法方案的流程图如下：



还原代码结果的系统要求:

- Windows/Linux
- 4 cores
- 8 GB RAM
- Python 3
- 10 GB 硬盘空间用于中间结果数据存储

代码细节:

- 0_leakage.py : 数据泄露进行一部分预估
- 1_random_forest.py : 随机森林
- 2_xgboost.py : XGBoost
- 3_sgd_classifier.py : SGDClassifier（随机梯度下降分类器用于比较大的数据）
- 4_naive_bayes.py : 朴素贝叶斯
- blend_models.py : 对前4个模型预测的类别概率结果进行加权平均
- combine_results.py : 和数据泄露的结果合并生成最终的结果
- run_loop.cmd : 命令行工具文件，对数据跑若干轮。

解法讲解

① 问题描述

Predict most likely hotels (rather hotel clusters) to be booked by a user in a certain destination. Evaluation metric is MAP@5, so that for each row predicting top 5 clusters would be sufficient. In total there are 100 hotel clusters.

② 数据描述

Expedia has provided sample hotel booking data from 2013 to 2015. Data from 2013 and 2014 are used for training, while that from 2015 is the test data. The data contain **actual bookings and even just clicks, check-in and check-out dates, user location details, destination details and origin-destination distance**. Besides, there are **149 latent variables** for each destination. The training data contain around 37 million rows, whereas the test data has 2.5 million rows. All the user id's in the test set are there in the train set.

③ 关于数据泄露

A few days into the competition a leakage was discovered in the data. Just by matching user location city, origination distance and few other variables almost 1/3rd of the test data could be accurately predicted. However, for the remaining rows of test data, there is a scope for machine learning.

④ 机器学习策略

4.1 如何建模

For machine learning, the problem can either be interpreted as a **100 class classification problem**, where the classes are the hotel clusters; or a binary classification problem where for a given user-destination combination we calculate whether it was a booking or just click for each of the 100 clusters. I did little experimentation with a small training set and based on the results I chose the former, i.e. 100 class classification.

4.2 选择合适的训练集

Yet another question is, should we model on the whole data or **only the booking data, excluding the "just click" data**. As the test data contains only booking events, it makes sense to choose only booking data for training purposes. Another reason is that it reduces the data size significantly, given that I am working on a system with average specifications. On the other hand, giving some weight to the click events improves the model.

For Random Forest and XGBoost I chose only booking data for training, but **I did not omit the click information completely**. **I included the probability-like outputs from a destination based weighted click-counting model as features**. That would be 100 destination based features corresponding to the 100 clusters which sum up to 1.

For online learning models like SGDClassifier and BernoulliNB, I trained on all rows by giving more weight to the booking events than the click events.

4.3 特征工程

Next, we need to see which features can be significantly useful in the prediction. Some engineered features may make the models more efficient. Here I chose the **time remaining from booking date till check-in date**, **stay duration** as my new variables. And not to mention the obvious, I broke down each date column into **year**, **month** and **day** columns, especially the month column may be important as a **seasonality indicator**.

4.4 验证集选择

For validation, I chose a **random 80-20 split**. A time based split might have been a good option here, but given that we have only 2 years of data, we have to sacrifice either seasonality or half of the data to train on.

⑤ 结果融合与组合策略

Since it is clear that for known combinations of user location cities, origin-destination distances and search destinations the hotel cluster is almost certain, machine learning would be of little or no help there. A simple counting will do a great job there. But this affects only 1/3rd of the data. Machine learning can be useful on the rest of the data.

So my strategy was to **append the machine learning predictions on top of the leakage based predictions**. I used 4 models for that: **Random Forest, XGBoost, SGDClassifier and BernoulliNB**. A weighted average of probability estimates from these models were used in the final solution for appending with the leakage.