Model Documentation

Competition Name: Predict Future Sales

Name: Xin Jie Lee

Public Leaderboard: 0.922124

Private Leaderboard: 0.932341

Summary

- The most important training method used is XGBoost.
- The most important features are lagged values of mean encodings. Two types of mean encodings are used:
    1. Mean encoding without regularization
    2. Mean encoding using the expanding mean scheme
- The models are trained on a laptop with an Intel Core i7 4700MQ (Quad-core 2.4GHZ) and 8GB RAM. No GPUs were used during training. I increased my laptop's paging file size to 40GB in order to train the models.
- The final model is an ensemble of 3 models. Each of these model takes around 2 to 6 hours to train. The ensemble model is a simple stacking model using a ridge regression model. The training of the stacking model takes less than 10 mins.
- Tools used for this project are: Anaconda (Numpy, Pandas, Scikit-Learn), XGBoost, pickle.

Exploratory Data Analysis

- The EDA can be found in EDA.ipynb.
- There are two outliers in the data with regards to the number of items sold in a day by a shop.
    - In month 24, shop 'id 12' sold 1000 of item 'id 20949'. It is likely that this item refers to plastic bags.
    - In month 33, shop 'id 12' sold 2169 of item 'id 11373'. It is likely that this item refers to a delivery service.
- There is an item, 'item id 6066', that costs more than \$300,000. The 99th percentile of item prices in the train set is \$5,999.
- Every shop in the test set contains the same number and type of items. This is not the case in the train set.
- There appears to be some seasonality trend in the sales of items.
- There are 102,796 'shop id', 'item id' pairs in the test dataset that is not in train dataset.

Feature Engineering and Selection

- Main features that were engineered were mean encodings and lagged values of these mean encodings.

- In model #1, expanding mean encodings of target (item_cnt_day), item_price and revenue grouped-by shop_id and item_id were generated. The 1-,2-,3-,4-,5-,6-,7-,8-,9-,10-,11- and 12-months lagged values of the expanding mean encodings were also generated.
- In model #2, the lagged mean encodings of target (item_cnt_day), item_price and revenue grouped-by (item_id, date_block_num), (shop_id, date_block_num), (item_category_id, date_block_num), (item_id, shop_id, date_block_num), (shop_id, item_category_id, date_block_num) were generated. The 1-,3-,6-,9- and 12-months lagged values of these mean encodings were generated as well.
- In model #3, the same mean encodings as model #2 were generated as well. The difference is that the 2-,4-,5-,7- and 8-months lagged values of the mean encodings were generated instead.
- Since the target shows some seasonality, I added a feature that tracked the month of the year (0 to 11).
- Since not all test shop-item pairs are found in the train set, we can also add all possible shop-item pairs to the training data and assign targets of 0 for shop-item pairs with no sale. However, this procedure would consume a lot of memory since we will be increasing the size of the training data tremendously. Hence, I opted not to do so for this project since I am training the models on my laptop.
- Did not perform feature selection.

Training Methods

- There are 3 models used for the final ensemble model:
    1. Model #1: XGBoost model with expanding mean encodings of target (item_cnt_day), item_price and revenue grouped-by shop_id and item_id. The 1-,2-,3-,4-,5-,6-,7-,8-,9-,10-,11- and 12-months lagged values of the expanding mean will also be generated. Code for this model is found in XGB_Expanding_Mean.ipynb.
    2. Model #2: XGBoost model with lagged mean encodings of target (item_cnt_day), item_price and revenue grouped-by (item_id, date_block_num), (shop_id, date_block_num), (item_category_id, date_block_num), (item_id, shop_id, date_block_num), (shop_id, item_category_id, date_block_num). No regularization was used for this mean encoding. The 1-,3-,6-,9- and 12-months lagged values of these mean encodings were generated. Code for this model is found in XGB_date.ipynb.
    3. Model #3: Same as model #2 but the 2-,4-,5-,7- and 8-months lagged values of the mean encodings will be generated instead. Code for this model is found in XGB_date.ipynb.
- Used 1 month of validation (month 33) for all three models. This month was used to manually cross-validate the model. There is the possibility that using more months as validation months could lead to better generalization of the model. I did not explore this due to time constraint.
- Performed manual parameter tuning rather than use a grid search to save time. There is the possibility that models' performance could be improved with more extensive parameter tuning.
- Tried other models such as Random Forest, but they did not perform as well as XGBoost models.
- I optimized the models with RMSE since the competition is evaluating the results based on RMSE.
- I did not eliminate the outliers when training the models. It is possible that doing so might improve the results.

Model Execute Time

- Model #1: About 2-3 hours
- Model #2 and Model #3: About 4-6 hours each
- Ensemble: About 10 minutes
- All of these models were trained on a laptop with an Intel Core i7 4700MQ (Quad-core 2.4GHZ) with no GPU.