1. Background:

Competition: Predict Future Sales
Team: none, just me
Name: Xin Lu Tan

2. Model Summary

The following types of features are generated:

(a) Lagged features: These features are generated as lagged version of monthly sold item counts. I considered aggregation at the level of (shop_id, item_id), shop_id, item_id, and item_category_id.
(b) Mean encoded features: I used expanding mean encoding to encode the categorical features (shop_id, item_id), shop_id, item_id, and item_category_id.
(c) Time: this is the month of year.
(d) Text features: Constructed from shop_name using Tfidf, but they turned out to to be very useful.

The solution is a convex combination of predictions from a light GBM model and a linear model.

To reproduce the evolution of the entire submission, follow the Jupyter notebooks in the following order: prepare_data.ipynb, train.ipynb, predict.ipynb. These notebooks are available in the Code Folder. The notebook explore_tune_model.ipynb is not needed to reproduce the final submission, but is included because it shows the development process of the final models in train.py.

(1) prepare_data.ipynb: This notebook loads raw data from the Data folder, performs some exploratory analysis, data preprocessing and cleaning. The cleaned data ready for training, validation, and testing are stored in the CleanData folder.

(2) explore_tune_model.ipynb: This notebooks loads the cleaned training and validation data from the CleanData folder, and trains and tunes light GBM and linear models. Models are trained on the training data and evaluated on the validation data. A simple ensembling scheme based on simple convex combination of light GBM and linear model predictions are considered at the end of the notebook. The best convex combination is determined using light GBM and linear model predictions on the validation data only.

(3) train.ipynb: This notebook retrains light GBM and linear models using all the data available for training (the aggregation of both training and validation data used in explore_tune_model.ipynb). The models are stored in the Model folder.

(4) predict.ipynb: This notebook loads the trained light GBM and linear models from the Model folder, and also the cleaned test data from the CleanData folder. It then generates test predictions from both models and combines them using the alpha coefficient determined in explore_tune_model.ipynb. The resulting predictions are then stored in the Submission folder, ready for submission. The public and private LB scores I received are: 0.967965 and 0.967967.

In addition, some utility functions used in explore_tune_model.ipynb are included in Utility.py.

The main library versions are:
numpy 1.18.1
pandas 1.0.1
sklearn 0.22.1
lightgbm 2.3.1

3. Features Engineering

I added item_category_id to the sales data for analysis of its predictive power.

I began by first removing some outliers in item_price and item_cnt_day. Then I performed aggregation over each month to obtain the monthly item counts, since the goal is to predict item counts in the following month. The groupings of item counts (target variable) is performed in conjunction with the following keys:

- (shop_id, item_id)
- shop_id
- item_id
- item_category_id

I also considered the average item_price for each (shop_id, item_id) each month.

After the aggregation above, I generated lagged features from these aggregated features. I considered lags 1, 2, 3, 4, 5, 6, 12. This part is performed for the concatenated training and test data. The training data is obtained from the monthly aggregation, while the test data is the raw data from test.csv.

Furthermore, I considered mean encoding for the categorical features (shop_id, item_id), shop_id, item_id, and item_category_id. To construct encoded features for training data, I first sorted data by month and considered expanding mean encoding with prior regularization. To construct encoded features for test data, I computed category means over the entire training data and further regularized by prior.

Finally, I added the month of year as a feature.

I also tried generating text features from shop_name using TfidfVectorizer, but they don't seem very informative.

4. Model Training

After the features engineering step, I discarded the first 12 months of data (date_block_num 0-11), as we have lagged features up to lag 12 and these are not available for the first 12 months of data. Discarding the features in the first 12 months also helps ensure that the mean encoded features are more stable, since the feature values for initial rows will have higher variability when using expanding mean encoding.

Next, I split data from remaining months into two parts: train_df which covers date_block_num 12-27, and valid_df which covers date_block_num 28-33. During model development process (explore_tune_model.ipynb), I trained model on train_df and evaluated its performance on valid_df. I also used valid_df to help tune model.

I considered training a lightGBM model on all features. I trained the model on train_df and used valid_df for early stopping. I then plotted the progress of model training over the number of iterations and examined feature importance. Model predictions are generated for valid_df.

I also trained an elastic net model with some heuristic hyperparameter values, and decided to fine tune the hyper parameters using ElasticNetCV on train_df. The resulting model helped select a subset of features, which I then re-fitted on train_df using a linear model (without regularization). The predictive accuracy on valid_df does not differ much from the one with heuristic hyper parameter values. Similar as before, model predictions are generated for valid_df.

Finally, I considered a convex combination of the light GBM and linear model valid_df predictions. The best combination is determined using RMSE on valid_df.

Now that the light GBM and linear models are tuned and a convex combination is determined, I retrained the light GBM and linear models on the concatenation of train_df and valid_df, using the tuned hyperparameter values. Predictions are then generated for the test data. Lastly, I combined the two set of predictions using the convex combination determined before. This constitutes the final predictions ready for submission.

5. Future Work

Due to time constraint, I haven't tried exploring other advanced features engineering technique and exploiting data leakage. I also have limited computing resources and so did not try models like neural network or k-nearest neighbor. However, I think the approach I have explored indicated that with proper features generation and simple modeling and ensembling technique, one can already obtain reasonable predictive performance on this dataset.