1c company - predict future sales (Kaggle)

This is the solution for the ‘1C company, predict future sales’ competition, which serves as Final project for "How to win a data science competition" Coursera course.
I ranked 6th on the public leaderboard out of 388 participants.
https://www.kaggle.com/c/competitive-data-science-final-project/leaderboard

Project Description:

We were provided time-series dataset consisting of daily sales data of 1C Company, and were asked to predict total sales for every product and store in the next month.

Highlights:

- The best single model is an entity embedding neural network from studying the winning solution of Rossman Sales prediction competition (https://github.com/entron/entity-embedding-rossmann)
- Stacking using simple Linear Regression over 4 base models (xgboost, randomforest, linear regression, embedding neural networks) shows significant improvement over one single model.

EDA

- Target is heavily skewed, most of the values are zero. 0,20 range contain 99.9% of the data range. Therefore, clipping the target to (0,20)
- Category_target and shop_target shows strong decreasing trend and yearly seasonal pattern, therefore, should incorporate lag 12 features. Autocorrelation plot shows the previous 6 months often have positive correlation, therefore include lag 1 to 6 features.

Feature preprocessing and generation

- Remove outlier from sales_train data
- Calculate aggregation features for each month on shop_id and item_id, shop_id only , item_id only, category_id only. For each aggregation, calculate item_cnt_day sum , item_price median, and sales sum.
- Split the date column into month and year features
- Generate lag features for month 1,2,3,4,5,6,12
- For neural networks, numerical features are standardized before fitting into the model
- For tree based features, no scaling is performed since they do not affect the model performance.
- For linear regression, only numerical features are fed into the model.

Feature extraction from text

- Use TfidfVectorizer to transform item_name and category_name into vectors.
- Then use TruncatedSVD to reduce its dimensions to 10

Mean encodings

- Generated mean encoding for all categorical features using expanding mean
- Features encoded: item_id,shop_id,item_category_id,month,year
- Target used for encoding: target, shop_target, item_target, category_target

Validation

- Train test split is time based.
- Two ways to split for train and validation:
  1. use last two month as validation set
  2. Use date_block_num in {9,21,33} as validation set
- After comparing the validation RMSE score vs. leaderboard RMSE score, selected the second validation method.

Metrics optimization

- Regressors minimize mean squared error. Validation metric used RMSE, same as the evaluation metric of the project.

Hyperparameter tuning

- used early stopping to do parameter tuning for xgb and neural networks.

Ensembles

- Stacking five model: xgb, rfr, lr, simple nn, embedding nn
- Train meta-features are generated using scheme f) from the reading material of the course. T equal to month, M=28
- Add pairwise differences to the level2 meta features and fit using LinearRegression.

How to generate solutions

1. Generate the full dataframe and split it into training, validation, test
   Python run_all_data.py
2. Generate best xgboost regressor prediction, and generate feature importances
   Python model_xgboost.py
3. Generate best simple neural network model prediction, as well as a standardized features dictionary dumped to local for stacking.
   Python model_simple_neural_network.py

4. Generate best embedding neural network model prediction, as well as a processed feature dictionary dumped to local for stacking.
   Python model_embedding_neural_network.py
5. Generate an ensemble using stacking for XGBRegressor, RandomForestRegressor, LinearRegression, simple NeuralNetwork and embedding Neural Network.
   Python run_ensemble.py