DBA4761 Seminars in Analytics

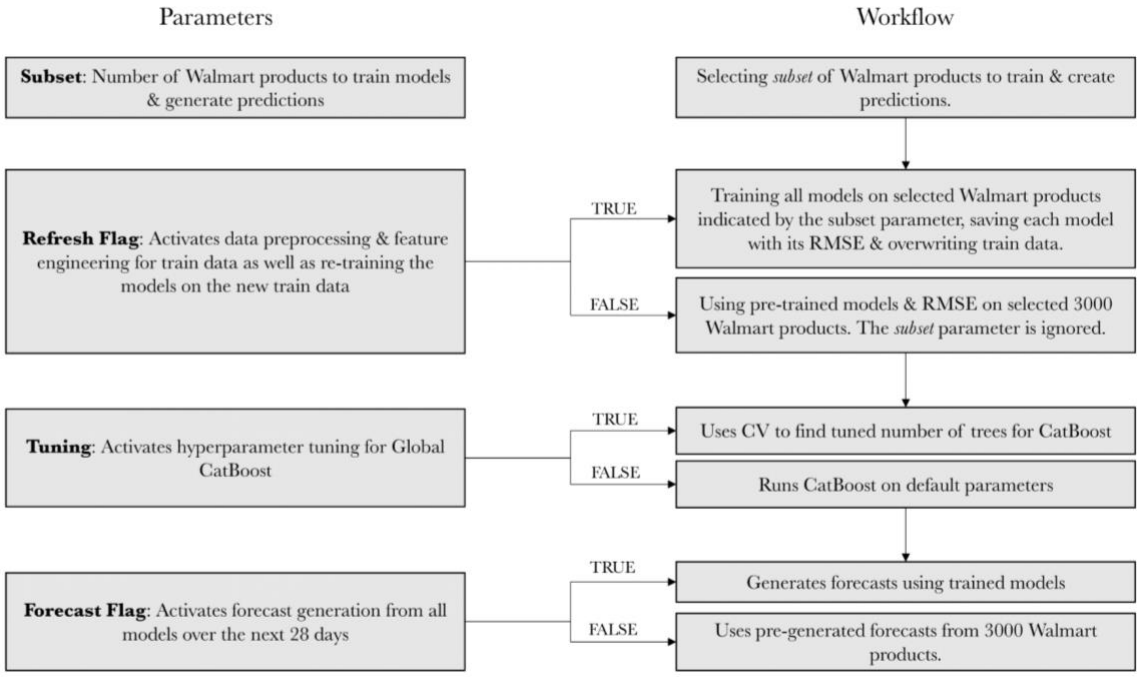
Prof Rafael Nicolas Fermin Cota

---

1. General

Specs: 16 Gb Ram

Models trained on Kaggle (Multiple Notebooks)



Refer to the above graph for the usage of parameters in our scripts

2. Usage

1) Set the parameter subset to the number of IDs you want to run
   - For demo purposes, use 5 (and set Refresh_Flag = TRUE)
   - Doing this overwrites the training data and all models will be retrained
   - The output files will not be overwritten if the Forecast_Flag is set to FALSE for all models
   - For best performance, use max (Not recommended, may take very long to train as all IDs will be used)
   - If Refresh_Flag = FALSE, saving functions will be disabled and pretrained models will be loaded instead
   - The full training data will be used and all models will be retrained but the associated weights and data will not be saved/overwritten

2) To generate the submission files

- Refresh_Flag = FALSE to use pretrained models (Recommended)
- Forecast_Flag will allow the user to control which model to rerun the test set predictions when the script 5_generate_forecast.R is activated
- If Forecast_Flag is set to FALSE for a particular model, inference will not be triggered and the original output will be loaded instead
- Run 1_load_pkgs.R, 5_generate_forecast, 6_combine_output
- Subset parameter will not affect these 3 files if Refresh_Flag = FALSE
- 3000 time series will be produced
- Re-generation of the submission files may take a very long time due to the large number of models

Output Generation Time

DeepEnsemble	1 Hour
GlobalEnsemble	2 Hours
LightGBM Global	3 Hours
LightGBM Nested	10 Hours

**3_ids_select.R will pick the top 3000 IDs to train. If subset parameter < 3000, the IDs selected = subset.*

**If the user would like to re-train a specific model, he can comment out the rest of the training scripts in main.R.*

**If the user would like to overwrite all files, set Refresh_Flag = TRUE, subset = 'max', Forecast_Flag = TRUE for all models (Beware that massive amount of ram maybe needed and the time taken will be very long) (The training of our models and test set inference are done on multiple laptops and kaggle notebooks)*

3. Data Preparation

1) Loading Data
   - We first load in the calendar.csv, sales_train_validation.csv and sell_price.csv
   - These 3 files are then combined to form one dataframe and stored as M5_sales.rds
   - The subset parameter will control the number of IDs saved in this file

2) ID Selection
   - Load in M5_sales.rds
   - Split data into train-val
   - Fit GP Forecaster on the train set (IDs, Date, Demand)
   - Evaluate on the validation set
   - Pick the top 3000 IDs with the lowest RMSE scores

3) Feature Engineering
   - Event names are converted to isEvent (0, 1)
   - Convert event types to dummies

- Calculate weekly price momentum (Current sell price - sell price 7 days ago)
- Add date features - year, month, day, day of week, moonphase and weekend. Note that Sunday is denoted by 1.
- Create lags and rolling aggregates (mean and standard deviation) for the intervals of 7, 14, 30, 60 and 180 days.
- Split data into training and testing set
- Normalise sell price

4. Modelling Approach

1) Deep Ensemble
   - We first split the data into train-val
   - Fit DeepAR, NbeatsEnsemble, DeepState and GP Forecaster on the train set
   - Evaluate on the val set and save the RMSE scores for each model
   - Combine all 4 models into a weighted ensemble with loadings set to 1/RMSE scores
   - Refit Ensemble on full data

2) Global Ensemble
   - Catboost and Theta (Autoregressive) are trained
   - We stack these two models together to form an ensemble using XGBoost
   - Recursive function is added to the stacked ensemble so that lag features can be calculated during inference

3) Global Light GBM
   - We first enter the default parameters and use CV to find best number of trees for all 3000 time series
   - We fit a global model using all 3000 time series using tuned parameters
   - A manual recursion was done to generate predictions day-by-day and incorporating lagged predicted values in future prediction

4) Nested Light GBM
   - We first enter the default parameters and use CV to find best number of trees for each time series
   - We fit a nested model for each item using tuned parameters
   - A manual recursion was done to generate predictions day-by-day and incorporating lagged predicted values in future predictions

5. Submission Generation

1) Test set predictions from each model will be saved in the output folder (rds format)
2) 6_combine_output.R must be triggered
3) These predictions will be combined together using a softmax weighted ensemble approach
4) Submission file will be generated (csv format) with timestamp embedded in the file name
5) Our submission file for the project is named final_submission.csv

6. Appendix

This folder contains other modelling approaches which are not selected due to undesirable performance / excessive training and inference time

1) Magic Multiplier = y_true / y_pred for the whole train set, grouped by id
    - This multiplier will be used to scale our prediction for test set

2) Nested ML
    - We fit a nested model for each item using XGBoost, Catboost and Multilayer Perceptron
    - We then create an additional weighted ensemble model, with 50%, 33%, 17% weightage for the 1st, 2nd, 3rd best model respectively
    - The best model among these 4 models (XGB, Cat, MLP, Weighted Ensemble) are selected and used to forecast the demand

3) Nested AR
    - We fit a nested model for each item using Exponential Smoothing, Croston, Theta, Thief and Nnetar
    - The best model among these 5 models are selected and used to forecast the demand

**Nested AR and Nested ML are in the appendix folder as we are unable to obtain the test set predictions before the submission date (inference time is extremely long). These models, however, have already been trained and their weights are stored in the appendix/nested_models folder.*