IN-Season Price Optimization Model Handbook

This model forecast sales qty at SKU or STYLE level on a select day with MARKDOWN for each product as a given knowledge.

Global parameters

- `pass_days:` 提前多少天预测目标日期的销量
- `fcst_date:` 预测的目标日期，没有可填写一个数据没有的日期，或""
- `rm_dates:` retail moment dates，在是否预测retail moments和计算历史销量变量是否考虑retail moments会用到
- `PE:` Product Engine(FTW)
- `use_platform:` TMALL
- `paths:` 路径管理工具
  - paths.source: 原始从DB中取到的数
  - paths.step_data: 对source的数处理后的数据
  - paths.style_model/paths.sku_model: 模型数据

> global parameters中的pass_days/rm_dates/paths在config.py中定义，这些参数会影响到后面每个code file的运行，务必在运行前确认或修改，并且在运行各代码文件时保持一致；另外重要的原始数据文件的文件名也在config.py中定义

Running Procedures

Step1. 从DB取transaction+attribute+inventory的数据，合并

- **Code File:** 1_md_sensitivity_model_data_init_include_sql.py
- **Key Parameters:**
  - `data_end_date:` 取数的截止日期
  - `pass_days:` global parameter
  - `rm_dates:` global parameter
  - `use_platform:` TMALL

> 1. 取数的逻辑是先从存量（之前处理好的）数据中读取数据最后的日期，作为data start date, 从数据库取data start date至data end date之间的数据，拼接好后，续到存量数据后，再存于本地**（如果只有训练数据可以直接覆盖原存量数据文件）**
> 2. 如果是预测数据，需要将预测日期的产品，折扣，MSRP，是否inseason等信息，事先存到一张表，merge 完product attribute等信息之后，append到之前的存量数据后，存到一个新的文件，不要直接覆盖原存量数据，因为预测日期的md并非真实，sales_qty，sales_amt也不存在，**存量数据用于存放真实已发生的数据**

- **Data files:**
  - **存量模型训练数据（输入）：** step_data/tmall_sku_daily_data_master_fcst.csv
  - **MPM product attributrs**(输入)
    每个season一张表，需要及时找Will Sun要最新season的
  - **更新日期后的模型训练数据（输出）：** step_data/tmall_sku_daily_data_master_fcst.csv
    如果只需要训练模型，不需要下面两步，这一步的结果可以直接作为后面程序的输入
  - **预测当天的产品数据（输入）：** step_data/plan_99_products.csv 仅包含99的产品
  - **模型预测的输入数据（输出）：** step_data/fcst_20200720.csv 存量训练数据+merge了product attribute和inventory的预测产品数据，如果需要做预测，这一步的结果是后面程序文件的输入
  - **本地库存数据（输出）：** inventory_data.csv 也会随着日期更新被更新
  - **retail moment销售数据（输出）：** 会作为3_zx_rm_features.py的输入，计算历史大促相关的特征

Step2. Feature Engineering

> 特征工程Part分为三块代码, 需要依次运行
> **2_Sophie_feature_engineering_style(sku).py**
> **3_zx_rm_features.py** (can skip if forecast non-retail moment)
> **4_md_sensitivity_style_99_doc.py**

2.1 Sophie features:

分为style和sku两个版本

- **Code File:**
  - 2_Sophie_feature_engineering_style(sku).py
- **Input Data:**
  - `data_master_include_forecast_day`(fcst_20200720.csv) :
    训练/预测数据，Step1的结果
  - `sophie_features_src` / `sophie_features_sku_src`
    (20200711df_style.csv/20200714df_sku.csv):
    Sophie features的存量feature数据
- **Key Parameters:**
  - `code_start_date:` 需要重新计算feature的起始日期
  - `code_end_date:` 需要重新计算feature的终止日期
  - `excl_rm:` 是否需要在计算feature是把rm_dates去除，即计算历史销量的时候不考虑retail moments时的销量
  - `fcst_date:` global parameter
  - `PE:` product engine
- **Output Data:**
  - updated `sophie_features_src` / `sophie_features_sku_src`
    (20200711df_style.csv/20200714df_sku.csv)
    by sku/style by date by inseason_flag1 的features

> 如果本地存量数据中含有上次预测日的数据，则需要将上次预测日的特征重新计算，e.g.设置code_start_date小于上次预测日，即可覆盖上次预测日的特征

2.2 Retail Moment Features

Retail-moment related features for forecasting retail moments only
These features will be **NAN** for non-retail moment days

- **Code File**
  - 3_zx_rm_features.py
- **Input Data:**
  - `local_retail_moment_only_data_src:` retail moments transaction data
  - `data_master_include_forecast_day:` this data is only used for getting DIVISION(FTW/APP/EQP)
  - `rm_calendar:` retail moment calendar, this file need to be manually maintained when new retail moments are settled

sales_date	RM_name	number	day_number
2017-03-06	38女王节	1	1
2017-03-07	38女王节	1	2
2017-03-08	38女王节	1	3
2017-06-18	618大促	2	1
2017-06-19	618大促	2	2
2017-06-20	618大促	2	3
2017-09-09	99大促	3	1
2017-09-10	99大促	3	2
2017-11-11	11.11	4	1
2017-12-12	12.12	5	1
2018-03-07	38女王节	6	1
2018-03-08	38女王节	6	2
2018-03-09	38女王节	6	3
2018-06-01	618大促	7	1

- **Key Parameters**
  - `PE:` product engine
  - `use_platform:` TMALL
  - `level:` sku or style
- **Output Data**
  - `zx_rm_sku(style)_features:` retail moment related features

2.3 Get All Features Ready

- **Code File:**
  - 4_md_sensitivity_style_99_doc.py (Initial setup + Step 1)
- **Key Parameters:**
  - `PE:` product engine

  - `fcst_date:` global parameter
  - `fcst_level:` sku/style sku level的预测还是style level的预测
  - `season_rm_dict:` defined in config.py, 每个season retail moment的起始日期，用于计算距离retail moment的天数，如果这个season没有retail moment可不运行
  - `fcst_date_site_traffic:` 预测当天的traffic预测值，如果不需要或者traffic数据文件中已有，可设置成None
- **Key Functions:**
  - `prepare_modeling_data:` 用于计算新的features以及merge上两步的features
  - `cal_days_from_rm:` 计算距离retail moment的天数，如果这个season没有retail moment可不运行
- **Input Data:**
  - `data_master_include_forecast_day:` (fcst_20200720.csv) Step1的输出
  - `competing_styles/skus_existing_features:` 本地存量competing-product features
  - `sku_color_features_to_date:` 本地存量颜色变量
  - `md_sensitivity_model_traffic_file:` daily site traffic, 如果数据中包含未来要预测的数据，则需要保证traffic文件中**包含未来预测那天的traffic预测值或者把值给到变量fcst_date_site_traffic**
  - 前面两步的输出(Sophie's features and Zhaoxu's rm features if needed)
- **Output Data:**
  - `md_sensitivity_model_master_file:` feature engineering finished and ready for modeling or forecasting
  - 变量列表及解释：\feature engineering\20200526_FeatureEngineering_master.xlsx

> 1. 如果pass_days不变，可以只计算增量日期的features，但是增量时间范围包含的sku/style的历史数据必须也加起来算增量日期的特征
> 2. 如果pass_days改变了，那么存量数据亦需要重新计算特征

Step3. Model Training

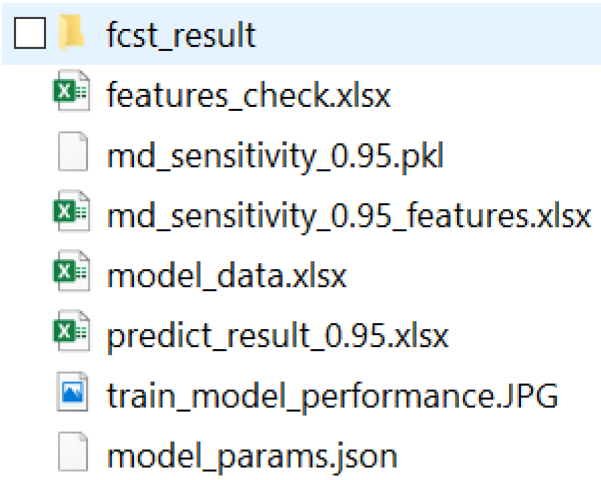
- **Code File:** 4_md_sensitivity_style_99_doc.py (Initial setup + Step 2)

3.1 Filter modeling target

- **Key Parameters:**
  - `train_seasons:` SP/SU/FA/HO select season for training model
  - `rm_remove:` False if forecast retail moment days, otherwise True
  - `off_season_remove:` True if only forecast in-season products, otherwise False
  - `full_price_remove:` True if only forecast discounted products(md>=0.05), otherwise False
  - `exclude_date_since:` if use only FA2018, FA2019 to train model, but with data longer than that, then set this date to '2020-01-01' can exclude 2020 data when modeling
- **Key Function:** `filter_modeling_data`
- **Input Data:** `md_sensitivity_model_master_file` (Step2 output)

3.2 Run model and check result

- **Key Parameters:**
    - `y_denom:` This model forecast sales boost, which is the sales qty of a target day over previous sales performance, so y_denom defines denominator of sales boost. Currently, past 3 days median sales qty is used as denominator
    - `y_numer:` sales qty of the forecast target day
    - `model_params:` This model is a tree-based model. This param defines learning rate, n_estimators, depth, number of leaves
    - `train_test_ratio:` train dataset size/available dataset size
- **Key Function:** `run_model`
- **Key Output:**
    - Performance Indices:
        - Sales qty MAPE of test data per day per product
        - Sales qty percentage of MAPE within 30%
        - Frequency percentage of MAPE with 30%
        - Sales boost accuracy matrix
    - Numerical features distribution
    - Finalized Model for next-level forecasting
    - Used features in Model
    - Predict result of test dataset
    - Feature importance(in predict result file)



Step4. Forecasting

- **Key Parameters:**
    - `is_forecast:` True
    - `fcst_date:` global parameter
    - `train_test_ratio:` for select model file only, model is named with train_test_ratio
- **Key Function:**
    - `filter_modeling_data:` for select in-season/off-season status, whether contains full price products
    - `ModelData:` prepare modeling data
    - `MDModel:` load model and make forecast
- **Input Data:** `md_sensitivity_model_master_file` (Step2 output)
- **Output Data:** forecast result with modeling attributes

> **如果模型早已训练好，只想加工预测数据的特征**，Sophie的feature部分依然需要把所有产品的历史数据都准备好作为code的输入，但是通过修改code_start_date, code_end_date都等于 fcst_date, 可以只计算预测数据的特征； Ruofei的feature部分可以只把预测当天所涉及到的产品的历史数据先取出，再run `prepare_modeling_data` ，然后只取sales_date=fcst_date的部分，就是预测数据的特征