MIM_Model 包使用说明

【Note：黄底字为可复制代码】

Step 1: prompt执行

- 下载 "…/Model_File/packages/_init_.py" 中的所有功能包
  - pandas
  - numpy
  - matplotlib
  - seaborn
  - joblib
  - lightgbm
  - datetime
  - sklearn
  - shap

- 在终端中下载：【代码】
  1. Upgrade python:
     1. 检查版本： python -V
     2. 更新至 3.9：

        conda create -n py39 python=3.9
        conda activate py39
  2. Upgrade pip: 如果提示pip版本不够，可用此代码更新

     python -m pip install --upgrade pip --user
  3. Install packages:

     conda install pandas
     conda install numpy
     conda install matplotlib
     conda install seaborn
     conda install joblib
     pip install lightgbm==3.3.1
     pip install datetime
     pip install scikit-learn
     pip install shap
  4. 如遇到"TimeoutError(HTTPSConnectionPool)"错误, 可添加以下镜像源
     1. pip 下载时, 在上述每个指令后添加

        -i http://pypi.douban.com/simple --trusted-host pypi.douban.com --trusted-host pypi.douban.com
     2. conda 下载时， 在所有指令前添加

        conda config --show-sources
        conda config --add channels http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
  5. 如遇到"OSError拒绝访问"，可在指令后添加以下代码解决

     --user
  6. shap包可能需要提前下载ipython: pip install ipython

- 执行指令：
  1. 前期自动生成数据configure参数表格，并保存至“Input/Input_Data”, 方便填写：

     -- formate: 【格式】

     python build_config.py ：fixed

     data filename : string

     -- example:

     python build_config.py raw_input

  2. 自动生成模型congifure表格，并保存至”Input/Input_Config”, 方便设置MCMC模型参数：

     -- formate: 【格式】

     python build_Mconfig.py ：fixed

     data filename : string

     num of points: response curve 中的点数

     data size: 指定训练数据的时间范围（行数）

     -- example:

     python build_Mconfig.py raw_input 10 36

  3. 最终运行模型

     -- format: 【格式】

     python main.py ： fixed

     data filename ： string

     configure filename ： string

     carry_over method: -i OR -o OR -n

     model type: -L OR -M

     -- example:

     python main.py raw_input -i -M

     -- note:

     -i inner power and linear effect

     -o outer power and linear effect

     -n no power, only linear effect

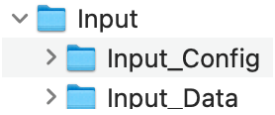
     -L Light Gradient Boosting Model

     -M Markov Chain Monte Carlo Model

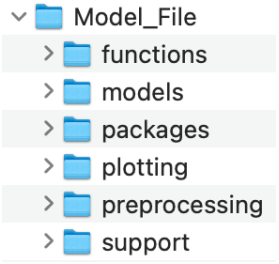
Step 2: 文件格式. MIM_Model

- 文件内部结构：

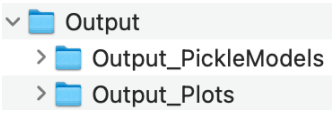
- **Input**: 数据集 & 参数值



Model_File: 运用所用到的所有脚本



- **Output**: 训练好的模型 & 输出的画图和信息



- **build_config.py**: 自动生成相应的数据参数文件

build_Mconfig.py: 自动生成相应的MCMC模型参数文件

- **main.py**: 主要运行的脚本

- **辅助脚本.ipynb**: 在修改脚本时用来辅助，其中包括转换原excel格式到标准csv格式等

- **Readme.txt**：终端指令输入的格式提醒。 该版本脚本的修改信息，以及待提升的地方。

- **README**：该文件

To 使用者：文件固定格式【必看】

Step 1：使用标准格式的数据集文件：

- 所有的数据文件需为csv，Columns: 【Date; Feature1; Feature2; Feature3 ….. 】，格式如图：

Date	OMOTotal-SalesVolKG	OMOTotal-AVGPriceIDX	LIBYTotal-AVGPriceIDX	BlueMoonTotal-AVGPriceIDX	TIDETotal-AVGPriceIDX	OMORT-Total-TTS-CPP	OMOR
44214	11139.5909	91.363209	93.651776	131.236712	93.935266	39573086.72	30540
44245	8846.686366	93.403221	96.575262	133.755355	97.315248	16218620.73	12286
44273	11507.1092	90.918507	94.378191	127.239768	93.949941	29584630.73	89043
44304	9575.318982	92.373546	94.379108	128.448529	96.352959	31864297.54	41699
44334	10731.15318	91.696355	94.214919	133.060735	96.186731	18122024.8	66383
44365	10022.94902	92.514388	95.363521	131.589425	95.58209	30524306.27	57244
44395	10947.53748	93.769697	94.268207	131.170893	95.611924	27818927.78	52610
44426	9392.950421	95.066902	97.023134	134.457111	96.032851	18739374.22	55460

- Note：联合利华原始数据，多数为Excel的行格式。可参考本文件的最后一页笔记，使用辅助脚本进行格式转换。

Step 2: prompt终端中建立参数文件：

- 举例说，我们用数据集 **"raw_input.csv"** 来进行测试和训练，在终端里输入：
  `python build_config.py raw_input`， **"Input / Input_Config"** 文件夹中会自动生成相应的数据参数文件，命名：【数据集文件名称_config.csv】，参数文件如图：

Feature	Constraints	Target	Media	CR	pw
OMOTotal-SalesVolKG					
OMOTotal-AVGPriceIDX					
LIBYTotal-AVGPriceIDX					
BlueMoonTotal-AVGPriceIDX					

- 当参数文件生成后，需要手动填写相应的参数值，【第一行位置为默认的预测变量，不要填写任何值；"**Contraints**": 单调性用**1** & **-1**区分； 媒体变量后需要将"**Media**"设置为 **1**；**"CR"**和"**PW**"分别为媒体变量carry-over effect指数, 为0至1之间数值，而当"**Media**"不为 **1** 时该处有值也不会进行考虑。 如图：

Feature	Target	Constraints	Media	pw	CR
OMOTotal-SalesVolKG	0	0	0	0	0
OMOTotal-AVGPriceIDX	1	-1	0	0	0
LIBYTotal-AVGPriceIDX	1	1	0	0	0
BlueMoonTotal-AVGPriceIDX	1	1	0	0	0
TIDETotal-AVGPriceIDX	1	1	0	0	0

- 当使用**MCMC**模型时，除了数据参数文件之外还需要设置模型参数文件：
  `python build_Mconfig.py raw_input 10 36`， **"Input / Input_Config"** 文件夹中会自动生成相应的模型参数文件，命名：【数据集文件名称_MCMC_config.csv】，参数文件如图：

Obj	ParameterName	ParameterValue	Explanation
TimeSeriesTrans	n_points	10	ResponseCurve中的描点个数的一半
MCMC_Model	model_name	MCMC_V1	自定义的待保存模型pickle文件的名称
MCMC_Model	stan_name	MLR_V1.stan	pystan库StanModel所需的一个配置模型参数的.stan文件的名称
MCMC_Model	data_size	36	指定训练数据的时间范围（行数）
MCMC_Model	target_path	Target_Feature.csv	进一步需要结合原数据观察贡献度的目标特征

- Note: 输入的格式可以参考本文件第二页 "执行指令"。

Step 3: 训练和运行模型：

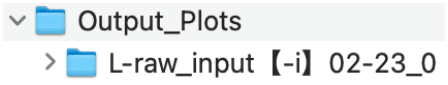
- 输入需要测试的数据集，且该数据集已有设置完成的参数文件。

Step4: LGBM 返回结果 `python main.py raw_input -i -L`

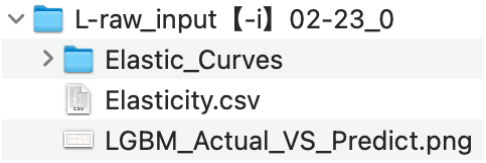
- 运行完后，终端会返回相对应的信息，并将需要的信息自动储存在“**Output**”文件夹里。

```
********** Training Ended **********

>>>>> After tuning parameters:
 {'num_leaves': 22, 'n_estimators': 170, 'min_data_in_leaf': 8,
 'max_depth': 14, 'max_bin': 330, 'learning_rate': 0.09}

********** Model Saved **********

>>>>> Model R-squared: 0.941899887373763
**********Done Plotting Fittness **********
**********Done Plotting Response Curve **********

>>>>> The Light Gradient Boosting Model is DONE !!! <<<<<
```

- 运行后生成的画图以及弹性系数等信息，会保存在”**Output / Output_Plots**” 中的一个新生成的文件里。命名：【模型类型 - 数据集文件名称【mode】训练时的日期 _ 后缀序列号】， 如图：



- 每个生成的文件夹里，包含了【模型表现图；弹性系数等信息表；所有变量的Response Curves等】

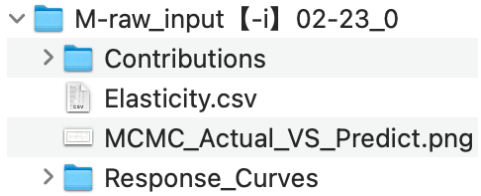


Step4: MCMC 检查返回结果 `python main.py raw_input -i -M`

- 运行完后，终端会返回相对应的信息，并将需要的信息自动储存在“**Output / Output_Plots**”文件夹里。

```
E-BFMI indicated no pathological behavior
**********Building Ended**********
**********Model Saved**********
**********Done Plotting Fittness **********
>>>>>Model R-squared: 0.5293442859716962
**********Done Scatter Plotting **********
**********Done Plotting Response Curve **********

>>>>> The Markov chain Monte Carlo is DONE !!! <<<<<
```

- 同上，每个生成的文件夹里包含了【模型表现图；弹性系数等信息表; Response Curves; Scatter plot等】



To 编译者：调试脚本【更多信息请见注释】

- **所有需要运行的脚本都在“Model_File”文件夹中，并以分类存放。**

Model_File 文件夹中

functions
__init__.py

- 自定义的计算函数
- 可以添加、删除或修改
- 在其它脚本里调用时：
  - from Model_File.functions import *

models
LGBM.py
MCMC.py

- 训练模型
- 返回R-Squared值
- 【自动保存模型表现图】
- 在其它脚本里调用时：
  - from Model_File.models import LGBM， MCMC
- 使用模型时：example
  - Model, folder_path = LGBM.do(x, y, config, file_name)

packages
__init__.py

- 所需要的全部功能包，要提前根据step1安装
- 可以添加、删除或修改
- 在其它脚本里调用时：
  - from Model_File.packages import *

plotting
__init__.py

- 自定义的画图函数
- 可以添加、删除或修改
- 【自动保存和生成文件夹】
- 在其它脚本里调用时：
  - from Model_File.plotting import *

preprocessing
split_data.py
read_file.py
data_trans.py

split_data.py

- 从原始数据集里根据constraint文件，提取所需的预测和特征值
- 使用时：
  - x, y = split_data.do(data, config)

read_file.py

- 读取prompt输入的名称
- 返回：result = [raw, file_name, constraints, media, mode]
- 使用时：
  - raw, file_name, config, mode = read_file.result

data_trans.py

- 进行carry-over转换
- 三种模式：
  - - n: 单纯线性转换
  - - o: 线性转换 + 整列指数转换
  - - i: 线性转换 + 指数转换
- 文件名称：原文件名称_转换方法.csv
- 【自动保存新数据集】
- 使用时：
  - new_data_path, new_data = data_trans.do(raw, file_name, config, mode)

- **当需要调用某脚本时：**
  - 比如从”Model_File / preprocessing”中调用“data_trans”、”read_file”、“split_data”函数，如图第一行：
  - 如需要调用某脚本中所有的函数，如图最后一行：

```
from Model_File.preprocessing import data_trans, read_file, split_data
from Model_File.models import LGBM_test1
from Model_File.plotting import *
```

- **脚本思路：以main.py主脚本中 LGBM为例**
  - 首先从终端中读取指令， 识别使用模型的类型：

```
#Given which model
model_type = sys.argv[-1]
```

  - 识别需要的使用的数据集、参数文件、模型文件、转换数据的方法

```
# read input from system terminal
raw, file_name, config, mode = read_file.result
```

  - 预处理数据，使用指定的carry-over method，并保存新数据：

```
# apply relative carry-over effect
new_data_path, new_data = data_trans.do(raw, file_name, config, mode)
new_data.to_csv(new_data_path, index=0)
```

  - 根据参数文件，区分训练特征和预测特征：

```
# splitting new dataset into x and y
x, y = split_data.do(new_data, config)
```

  - 训练模型，并返回生成的文件路径：

```
# do the modelling part
file_name_mode = f'{file_name}【{mode}】' #add mode into the filename
model, folder_path = LGBM.do(x, y, config, file_name_mode)
```

  - 用训练好的模型进行画图等分析操作，并保存到之前返回的路径中：

```
# calculate elasticity and plot all response curves
plot_allRC(x, y, raw, model, folder_path)
```

辅助脚本思路：辅助脚本.ipynb

- 当数据为以下格式「excel」时：

Date	Jun-20	Jul-20	Aug-20	Sep-20	Oct-20	Nov-20
Dove Dakota Body Scrub EC Volume	138822.016	46400.03	65722.219	47772.694	39270.308	182804.908
Dove PW MSP TV GRP	38.1351731	93.9921101	0	46.9204195	62.4197457	57.0094254
Dove Dakota Body Scrub EC Average Price	175.223856	219.587534	207.452752	210.054338	219.602922	171.311263
Dove Dakota BS EC CPP Promotion	522012.382	40.8495224	760.94224	32340.1032	302815.372	9913.33668
Dove PW BB-EC promotion	0	0	0	0	33627.96	0
Dove PW ECP Promotion	3081997.47	2792025.19	2385536.58	7802506.87	5186944.54	3951131.73
Dove PW CDS Digital	750000	0	0	0	2000000	800000
Dove PW CDS Social	6793339.64	1576171.41	2423395.3	4264551.7	5634426.65	5010944.19

- 用如图部分进行转换到标准格式：

转换至标准文件格式：

```
import pandas as pd
filename = 'Dove'
sheet = input()
data = pd.read_excel(f'Input/Input_Data/{filename}.xlsx', sheet_name = sheet)
data = data.transpose()
data = data.reset_index()
data.columns = data.loc[0]
data = data.drop(0)
data = data.reset_index(drop='index')
data.to_csv(f'Input/Input_Data/{sheet}.csv', index = 0)
```

```
Dove1
```

- 标准格式为「csv」：

Date	Dove Bar Offline Volume	Dove Bar Offline Average Price	Safeguard PW TV GRP	Dove Bar Offline TDP Distribution	Dove PW MSP TV GRP
Jun-20	46166.304	66.83805574	446.8773368	59	38.13517315
Jul-20	83284.928	55.21204869	0	60	93.9921101
Aug-20	51111.12	66.51520745	0	56	0
Sep-20	50231.812	65.65956151	0	51	46.92041947
Oct-20	41890.844	70.09672567	0	50	62.41974571
Nov-20	39394.628	68.00084519	0	47	57.00942536
Dec-20	36385.184	68.27463756	0	41	0
Jan-21	78855.544	49.64228641	552.563825	53	65.52775415
Feb-21	40503.232	65.38865219	397.1014673	46	0