1.Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
import gc
from tqdm import tqdm_notebook as tqdm
import warnings
warnings.filterwarnings('ignore')
import lightgbm as lgb

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/features-new/__results__.html
/kaggle/input/features-new/data.h5
/kaggle/input/features-new/__notebook__.ipynb
/kaggle/input/features-new/custom.css
/kaggle/input/features-new/__output__.json
```

2.Reading the Data

- Reading the saved data from the New_featurization notebook.

In [2]:

```
data = pd.read_hdf('/kaggle/input/features-new/data.h5', 'data')
```

3. Splitting the data

In [3]:

```
test_data = data[(data['date']>'2016-05-22')]
train_data = data[(data['date']<='2016-04-24')]
val_data = data[(data['date']> '2016-04-24')& (data['date']<='2016-05-22')]
del data
```

4.Splitting the train and validation data

In [4]:

```
data_list=[]
def data_list_fn():
    for j in range(1,5):
        for i in range(10):
            data_list.append(f'data_s{i}_w{j}')
data_list_fn()
#print(data_list)
model_list = []
def model_list_fn():
    for j in range(1,5):
        for i in range(10):
            model_list.append(f'model_s{i}_w{j}')
model_list_fn()
#print(model_list)


train_data_parts = {}
val_data_parts={}
test_data_parts={}
week_start = [23,30,6,13]
```

```
week_end = [29,5,12,19]


def ramu_data(data_list,data,data_parts):
    m = 0
    for j,k in zip(week_start,week_end):
        for i in range(10):
            data_parts[data_list[m]]=data[(data['store_id']==i) & ((data['tm_d']>=j)&(data['tm_d']<
=k))]
            m+=1



def ramu_data2(data_list,data,data_parts):
    m = 10
    for i in range(10):
        data_parts[data_list[m]]=data[(data['store_id']==i) & ((data['tm_d']>=30) | (data['tm_d']<=
5))]
        m+=1



ramu_data(data_list,train_data,train_data_parts)
ramu_data(data_list,val_data,val_data_parts)
ramu_data2(data_list,train_data,train_data_parts)
ramu_data2(data_list,val_data,val_data_parts)
ramu_data(data_list,test_data,test_data_parts)
ramu_data2(data_list,test_data,test_data_parts)

del train_data,val_data,test_data
```

5.Features division

In [5]:

```
features_w1 = ['item_id', 'dept_id', 'cat_id', 'store_id', 'state_id', 'wm_yr_wk',
            'event_name_1', 'event_type_1','event_name_2', 'event_type_2','snap_CA', 'snap_TX',
            'snap_WI','sell_price', 'price_max', 'price_min', 'price_std', 'price_mean',
            'price_nunique', 'item_nunique', 'encoded_id', 'lag_d_7', 'lag_d_8','lag_d_9',
            'r_std_d7', 'r_std_d14','r_std_d30', 'r_mean_d7', 'r_mean_d14',
            'r_mean_d30', 'tm_d', 'tm_w','tm_m', 'tm_y', 'tm_wm', 'tm_dw', 'tm_w_end']

features_w2 = ['item_id', 'dept_id', 'cat_id', 'store_id', 'state_id', 'wm_yr_wk',
            'event_name_1', 'event_type_1','event_name_2', 'event_type_2','snap_CA', 'snap_TX',
            'snap_WI','sell_price', 'price_max', 'price_min', 'price_std', 'price_mean',
            'price_nunique', 'item_nunique', 'encoded_id','lag_d_14', 'lag_d_15', 'lag_d_16',
            'r_std_d7', 'r_std_d14','r_std_d30', 'r_mean_d7', 'r_mean_d14',
            'r_mean_d30', 'tm_d', 'tm_w','tm_m', 'tm_y', 'tm_wm', 'tm_dw', 'tm_w_end']

features_w3 = ['item_id', 'dept_id', 'cat_id', 'store_id', 'state_id', 'wm_yr_wk',
            'event_name_1', 'event_type_1','event_name_2', 'event_type_2','snap_CA', 'snap_TX',
            'snap_WI','sell_price', 'price_max', 'price_min', 'price_std', 'price_mean',
            'price_nunique', 'item_nunique', 'encoded_id','lag_d_21', 'lag_d_22','lag_d_23',
            'r_std_d7', 'r_std_d14','r_std_d30', 'r_mean_d7', 'r_mean_d14',
            'r_mean_d30', 'tm_d', 'tm_w','tm_m', 'tm_y', 'tm_wm', 'tm_dw', 'tm_w_end']

features_w4 = ['item_id', 'dept_id', 'cat_id', 'store_id', 'state_id', 'wm_yr_wk',
            'event_name_1', 'event_type_1','event_name_2', 'event_type_2','snap_CA', 'snap_TX',
            'snap_WI','sell_price', 'price_max', 'price_min', 'price_std', 'price_mean',
            'price_nunique', 'item_nunique', 'encoded_id', 'lag_d_28','lag_d_29', 'lag_d_30',
            'r_std_d7', 'r_std_d14','r_std_d30', 'r_mean_d7', 'r_mean_d14',
            'r_mean_d30', 'tm_d', 'tm_w','tm_m', 'tm_y', 'tm_wm', 'tm_dw', 'tm_w_end']
```

6. Modelling 40 models

In [6]:

```
categorical_features = ['item_id', 'dept_id', 'cat_id', 'store_id', 'state_id','event_name_1',
                        'event_name_2','snap_CA','snap_TX', 'snap_WI','tm_d','encoded_id']

def lgbm_model(x_train,y_train,x_val,y_val,categorical_features,lr):
```

```
    params =  {'boosting_type': 'gbdt','objective': 'tweedie','tweedie_variance_power': 1.1,
                       'metric': 'rmse','subsample': 0.6,'subsample_freq': 3,'bagging_fraction': 0.5
                       'learning_rate': lr,'num_leaves': 70,'min_data_in_leaf': 2**8-1,'max_depth':7
                       'max_bin': 100,'n_estimators': 1000,'sub_feature': 0.6,'boost_from_average':
else,
                       'seed': 42,'feature_fraction': 0.5,'lambda_l2': 0.02,
                }


    d_train = lgb.Dataset(x_train,label=y_train, categorical_feature=categorical_features)
    d_val   = lgb.Dataset(x_val,label=y_val, categorical_feature=categorical_features)
    watchlist = [d_train,d_val]

    model = lgb.train(params,train_set=d_train,valid_sets=watchlist,verbose_eval = 1000)

    return model
```

In [7]:

```
lr=[0.04,0.03,0.04,0.05]
features_list = [features_w1,features_w2,features_w3,features_w4]
model_data_parts={}
def model_training(lr,features_list,data_list):
    m=0
    for i,j in zip(lr,features_list):
        w=1
        for k in range(10):
            x_train = train_data_parts.get(data_list[m])[j]
            y_train = train_data_parts.get(data_list[m])[['unit_sales']]
            x_val   = val_data_parts.get(data_list[m])[j]
            y_val   = val_data_parts.get(data_list[m])[['unit_sales']]
            model   = lgbm_model(x_train,y_train,x_val,y_val,categorical_features,lr=i)
            #print('Model of store {} in week {} is done '.format(k,w))
            model_data_parts[model_list[m]]= model
            m+=1
        w+=1
model_training(lr,features_list,data_list)
```

```
[1000] training's rmse: 2.04947 valid_1's rmse: 1.89648
[1000] training's rmse: 1.63724 valid_1's rmse: 1.91266
[1000] training's rmse: 2.70293 valid_1's rmse: 2.4378
[1000] training's rmse: 1.30546 valid_1's rmse: 1.32009
[1000] training's rmse: 1.63604 valid_1's rmse: 1.46832
[1000] training's rmse: 1.94497 valid_1's rmse: 1.73726
[1000] training's rmse: 1.77331 valid_1's rmse: 1.86768
[1000] training's rmse: 1.49721 valid_1's rmse: 1.59549
[1000] training's rmse: 1.9119 valid_1's rmse: 2.14571
[1000] training's rmse: 1.60276 valid_1's rmse: 1.67612
[1000] training's rmse: 2.30978 valid_1's rmse: 2.2128
[1000] training's rmse: 1.73419 valid_1's rmse: 2.17388
[1000] training's rmse: 3.15868 valid_1's rmse: 2.74446
[1000] training's rmse: 1.38562 valid_1's rmse: 1.43782
[1000] training's rmse: 1.83554 valid_1's rmse: 1.75912
[1000] training's rmse: 2.10023 valid_1's rmse: 1.98075
[1000] training's rmse: 1.95602 valid_1's rmse: 1.97632
[1000] training's rmse: 1.64666 valid_1's rmse: 1.77472
[1000] training's rmse: 2.68624 valid_1's rmse: 3.26958
[1000] training's rmse: 1.97283 valid_1's rmse: 2.07035
[1000] training's rmse: 2.28074 valid_1's rmse: 2.1923
[1000] training's rmse: 1.71063 valid_1's rmse: 1.97579
[1000] training's rmse: 3.15146 valid_1's rmse: 2.54053
[1000] training's rmse: 1.36837 valid_1's rmse: 1.44142
[1000] training's rmse: 1.86867 valid_1's rmse: 1.71469
[1000] training's rmse: 2.15456 valid_1's rmse: 1.98143
[1000] training's rmse: 1.98336 valid_1's rmse: 2.18605
[1000] training's rmse: 1.60191 valid_1's rmse: 1.75605
[1000] training's rmse: 2.91741 valid_1's rmse: 3.76998
[1000] training's rmse: 1.99206 valid_1's rmse: 2.39666
[1000] training's rmse: 2.1727 valid_1's rmse: 2.37884
[1000] training's rmse: 1.71342 valid_1's rmse: 2.05219
[1000] training's rmse: 2.95408 valid_1's rmse: 2.79983
[1000] training's rmse: 1.35014 valid_1's rmse: 1.44742
[1000] training's rmse: 1.82033 valid_1's rmse: 1.98428
[1000] training's rmse: 2.18836 valid_1's rmse: 2.00809
[1000] training's rmse: 2.01754 valid_1's rmse: 2.01028
```

```
[1000] training's rmse: 1.60881 valid_1's rmse: 1.77979
[1000] training's rmse: 2.62817 valid_1's rmse: 3.16582
[1000] training's rmse: 1.92945 valid_1's rmse: 2.08182
```

7. Saving all models

In [8]:

```
import pickle
filename = 'models.sav'
pickle.dump(model_data_parts, open(filename, 'wb'))
```

Summary

3. Splitting the data

- Splitting the data into train,validation and test data based on time.
- Validation and test dat for 28 days.

4.Splitting the train and validation data

- Splitting the train data and validation data weekly for every store of total 10 stores.
- Making it 4 * 10 = 40 train data and validation data.
- Storing these in a dictionary for easy access.

5.Features division

- lag features are separated for each week separately.
- week1 - 'lag_d_7', 'lag_d_8','lag_d_9'
- week2 - 'lag_d_14', 'lag_d_15', 'lag_d_16'
- week3 - 'lag_d_21', 'lag_d_22','lag_d_23'
- week4 - 'lag_d_28','lag_d_29', 'lag_d_30'

6. Modelling 40 models

- Individually taken train and validation data from the 40 data frames from the dictionary.
- Train the respective model and stored the model to a dictionary based on its name.

- Finally saving all the models for testing.