In [1]:
```
import datetime as dt
import os
import gc

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import lightgbm as lgb
from sklearn.model_selection import KFold

pd.set_option('display.max_rows', 1000)
pd.set_option('display.max_columns', 100)

INPUT_FOLDER = './data/input'
OUTPUT_FOLDER = './data/output'

os.listdir(INPUT_FOLDER)
dateparser = lambda x: dt.datetime.strptime(x, '%Y-%m-%d')
```

Зчитаємо дані

Заповнимо пропущені значення, обмежемо тренувальну вибірку.

In [2]:
```
df_train = pd.read_csv(os.path.join(INPUT_FOLDER, 'train_final.csv'),
    index_col='ID',
    parse_dates = ['date'],
    date_parser=dateparser)

df_train['sales'].fillna(0, inplace=True)
df_train.sort_values('date', inplace=True)
df_train['price'] = df_train.groupby(['geoCluster', 'SKU'], sort=False)['price'].apply(

# df_train['price_change'] = df_train.sort_values('date').groupby(['geoCluster', 'SKU']
df_train = df_train[df_train['date'] >= '2021-05-01']
df_train.head()
```

Out[2]:

ID	geoCluster	SKU	date	price	sales
RR41796038	2653	808038	2021-05-01	34.19	0.0
RR53812295	2239	259516	2021-05-01	19.99	0.0
RR32235156	2006	756640	2021-05-01	120.79	0.0
RR44125338	2086	709122	2021-05-01	18.49	0.0
RR53810272	2239	221793	2021-05-01	11.39	0.0

Порахуємо зміну ціни.

In [3]:
```
df_train['price_change'] = df_train.sort_values('date').groupby(['geoCluster', 'SKU'],
```

In [4]:
```
df_train[(df_train['price_change'] != 0) & (df_train['price_change'].notna()) & (df_tra
```

Out[4]:
```
0
```

Якщо ми все заповнимо нулями, то `%MAE=1` .

Ми точно знаємо, що коли ціна змінюється, продажі відбуваються (1)
Коли ціна не змінюється, продажі не обов'язково відбуваються (2)

Тому для мінімізації метрики, можемо заповнити таблицю 0 у випадку (2) - це найбільш "безпечний варіант". Прогноз для випадку (1) Таким чином ми знаємо,що ми точно не зробили помилку `%MAE>1` у випадку (2), а для випадку (1) ми підійшли творчо - спробували заповнити різними статистиками для продажів з тренувального датасету. Насправді модель теж пробували будувати для підвибірки, де змінюється ціна, але це не дало результату кращого ніж медіана.

In [5]:
```
df_test = pd.read_csv(
    os.path.join(INPUT_FOLDER, 'test_data.csv'),
    index_col='ID',
    parse_dates = ['date'],
    date_parser=dateparser
)


df_test['price_change'] = df_test.groupby(['geoCluster', 'SKU'], sort=False)['price'].a
df_test.reset_index(drop=True, inplace=True)

print(df_test.shape)
df_test.head()
```

```
(1666028, 5)
```

Out[5]:

	geoCluster	SKU	date	price	price_change
0	21	32485	2021-07-20	66.69	NaN
1	21	32485	2021-07-21	66.69	0.0
2	21	32485	2021-07-22	66.69	0.0
3	21	32485	2021-07-23	66.69	0.0
4	21	32485	2021-07-24	66.69	0.0

In [6]:
```
submission = pd.read_csv(os.path.join(INPUT_FOLDER, 'sample_final.csv'), index_col='ID'
submission.head()
```

Out[6]:

	sales
ID	
RR1666030	0
RR1666031	0
RR1666032	0

	sales
ID	
RR1666033	0
RR1666034	0

Прозноузування

Тестові дані заповнюємо відповідно.

In [7]:
```
df_test = df_test.merge(
    df_train[df_train['sales'] > 0].groupby(['geoCluster', 'SKU'])['sales'].median().re
df_test.sample(10)
```

Out[7]:

	geoCluster	SKU	date	price	price_change	sales
195875	2017	17	2021-07-21	19.99	0.0	NaN
776122	2158	419952	2021-07-24	374.99	0.0	NaN
831838	2183	540390	2021-07-20	41.29	NaN	NaN
1081854	2258	607635	2021-07-24	16.39	0.0	3.0
1615220	2984	819150	2021-08-01	51.19	0.0	2.0
491855	2061	837329	2021-07-27	25.59	0.0	NaN
643144	2117	868167	2021-08-01	24.09	0.0	NaN
1287839	2406	810184	2021-07-27	71.09	0.0	NaN
58052	1935	838007	2021-07-28	248.49	0.0	NaN
1535683	2807	559301	2021-07-29	326.59	0.0	NaN

In [8]:
```
submission['sales'] = 0
submission['sales'] = np.where(
    (np.abs(df_test['price_change']) > 0),
    df_test['sales'],
    submission['sales']
)

submission['sales'].fillna(0, inplace=True)
```

In [9]:
```
ts = dt.datetime.now().strftime('%Y%m%d_%H_%M_%S')
submission.to_csv(
    os.path.join(
        OUTPUT_FOLDER,
        f'{ts}.csv'
    )
)
```