M5-forecasting, CatBoost.

In [1]:

```
1 !pip install catboost
```

```
Collecting catboost
  Downloading https://files.pythonhosted.org/packages/b1/61/2b8106c887
0601671d99ca94d8b8d180f2b740b7cdb95c930147508abcf9/catboost-0.23-cp36-
none-manylinux1_x86_64.whl (https://files.pythonhosted.org/packages/b
1/61/2b8106c8870601671d99ca94d8b8d180f2b740b7cdb95c930147508abcf9/catb
oost-0.23-cp36-none-manylinux1_x86_64.whl) (64.7MB)
     |████████████████████████████████| 64.8MB 45kB/s
Requirement already satisfied: graphviz in /usr/local/lib/python3.6/di
st-packages (from catboost) (0.10.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/
dist-packages (from catboost) (3.2.1)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python
3.6/dist-packages (from catboost) (1.0.3)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-pa
ckages (from catboost) (1.12.0)
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist
-packages (from catboost) (4.4.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python
3.6/dist-packages (from catboost) (1.18.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-
packages (from catboost) (1.4.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.
1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->catboos
t) (2.4.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.
6/dist-packages (from matplotlib->catboost) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/
python3.6/dist-packages (from matplotlib->catboost) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/pyt
hon3.6/dist-packages (from matplotlib->catboost) (1.2.0)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.
6/dist-packages (from pandas>=0.24.0->catboost) (2018.9)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/pytho
n3.6/dist-packages (from plotly->catboost) (1.3.3)
Installing collected packages: catboost
Successfully installed catboost-0.23
```

In [0]:

```
 1 import gc
 2 import matplotlib.pyplot as plt
 3 import numpy as np
 4 import os
 5 import pandas as pd
 6
 7 from catboost import Pool, CatBoostRegressor
 8 from catboost.utils import get_gpu_device_count
 9 from datetime import datetime, timedelta
10 from tqdm.notebook import tqdm
```

In [18]:

```
print('GPU devices CatBoost:', get_gpu_device_count())
```

```
GPU devices CatBoost: 1
```

In [3]:

```
from google.colab import drive
drive.mount('/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/aut
h?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleu
sercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&respon
se_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fd
ocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%
2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2
fwww.googleapis.com%2fauth%2fpeopleapi.readonly (https://accounts.goog
le.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc
0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoaut
h%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googl
eapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2faut
h%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.rea
donly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly)

Enter your authorization code:
··········
Mounted at /gdrive
```

In [19]:

```
DATA_DIR = '/gdrive/My Drive/M5-forecasting'
MODEL_VER = 'v0'
BACKWARD_LAGS = 60
END_D = 1913
CUT_D = END_D - int(365 * 1.2)
END_DATE = '2016-04-24'
print(datetime.strptime(END_DATE, '%Y-%m-%d'))
np.random.seed(0)
```

```
2016-04-24 00:00:00
```

Загрузка данных

In [0]:

```
CALENDAR_DTYPES = {
    'date':             'str',
    'wm_yr_wk':         'int16',
    'weekday':          'object',
    'wday':             'int16',
    'month':            'int16',
    'year':             'int16',
    'd':                'object',
    'event_name_1':     'object',
    'event_type_1':     'object',
    'event_name_2':     'object',
    'event_type_2':     'object',
    'snap_CA':          'int16',
    'snap_TX':          'int16',
    'snap_WI':          'int16'
}
PARSE_DATES = ['date']
SPRICES_DTYPES = {
    'store_id':    'object',
    'item_id':     'object',
    'wm_yr_wk':    'int16',
    'sell_price':  'float32'
}
```

In [0]:

```
def get_df(is_train:bool=True,
           backward_lags:int=None):
    strain = pd.read_csv('{}/sales_train_validation.csv'.format(DATA_DIR))
    print('sales_train_validation.csv:', strain.shape)
    cat_cols = ['id', 'item_id', 'dept_id','store_id', 'cat_id', 'state_id']
    last_day = int(strain.columns[-1].replace('d_', ''))
    print('First day:', CUT_D)
    print('Last day:', last_day)
    if not is_train:
        for day in range(last_day + 1, last_day + 28 + 28 + 1):
            strain['d_{}'.format(day)] = np.nan
        value_vars = [col for col in strain.columns
                      if (col.startswith('d_') and (int(col.replace('d_', ''))
    else:
        value_vars = [col for col in strain.columns
                      if (col.startswith('d_') and (int(col.replace('d_', ''))
    strain = pd.melt(
        strain,
        id_vars = cat_cols,
        value_vars = value_vars,
        var_name = 'd',
        value_name = 'sales'
    )
    print('Melted sales:', strain.shape)
    calendar = pd.read_csv('{}/calendar.csv'.format(DATA_DIR), dtype=CALENDAR_
    print('calendar.csv:', calendar.shape)
    strain = strain.merge(calendar, on='d', copy=False)
    del calendar
    gc.collect()
    print('Merged done')
    sprices = pd.read_csv('{}/sell_prices.csv'.format(DATA_DIR), dtype=SPRICES
    print('read prices:', sprices.shape)
    strain = strain.merge(
        sprices,
        on=['store_id', 'item_id', 'wm_yr_wk'],
        copy=False
    )
    del sprices
    gc.collect()

    if not is_train:
        strain = strain.loc[
            strain['date'] >= (datetime.strptime(END_DATE, '%Y-%m-%d') - timed
        ]
    return strain
```

In [0]:

```
def make_features(strain):
    print('make features dataframe:', strain.shape)
    lags = [7, 28]
    windows= [7, 28]
    wnd_feats = ['id', 'item_id']
    lag_cols = ['lag_{}'.format(lag) for lag in lags ]
    for lag, lag_col in zip(lags, lag_cols):
        strain[lag_col] = strain[['id', 'sales']].groupby('id')['sales'].shift
    for wnd_feat in wnd_feats:
        for wnd in windows:
            for lag_col in lag_cols:
                wnd_col = '{}_{}_rmean_{}'.format(lag_col, wnd_feat, wnd)
                strain[wnd_col] = strain[[wnd_feat, lag_col]].groupby(wnd_feat
                    lambda x: x.rolling(wnd).mean()
                )
    date_features = {
        'week_num': 'weekofyear',
        'quarter': 'quarter',
        'mday': 'day'
    }
    for date_feat_name, date_feat_func in date_features.items():
        strain[date_feat_name] = getattr(strain['date'].dt, date_feat_func).as
    strain['d'] = strain['d'].apply(lambda x: int(x.replace('d_', '')))
    return strain
```

In [ ]:

```
%%time
strain = get_df(is_train=True, backward_lags=None)
strain = make_features(strain)
```

In [25]:

```
strain.head(3)
```

Out[25]:

	id	item_id	dept_id	store_id	cat_id	state_id
0	HOBBIES_1_001_CA_1_validation	HOBBIES_1_001	HOBBIES_1	CA_1	HOBBIES	C
1	HOBBIES_1_001_CA_1_validation	HOBBIES_1_001	HOBBIES_1	CA_1	HOBBIES	C
2	HOBBIES_1_001_CA_1_validation	HOBBIES_1_001	HOBBIES_1	CA_1	HOBBIES	C

In [0]:

```
drop_cols = ['id', 'sales', 'date', 'wm_yr_wk', 'weekday']
train_cols = strain.columns[~strain.columns.isin(drop_cols)]
cat_cols = [
    'item_id', 'dept_id', 'store_id', 'cat_id', 'state_id',
    'event_name_1', 'event_type_1', 'event_name_2', 'event_type_2'
]
strain[cat_cols] = strain[cat_cols].fillna(0)
```

CatBoost

In [10]:

```
%%time
val_size = int(strain.shape[0] * .15)
val_idxs = np.random.choice(strain.index.values, val_size, replace=False)
train_idxs = np.setdiff1d(strain.index.values, val_idxs)
train_pool = Pool(
    strain.loc[train_idxs][train_cols],
    strain.loc[train_idxs]['sales'],
    cat_features=cat_cols
)
val_pool = Pool(
    strain.loc[val_idxs][train_cols],
    strain.loc[val_idxs]['sales'],
    cat_features=cat_cols
)
del strain
gc.collect()
```

```
CPU times: user 42.1 s, sys: 1.85 s, total: 44 s
Wall time: 41.1 s
```

In [11]:

```
model = CatBoostRegressor(
    iterations=1000,
    task_type='GPU',
    verbose=200,
    loss_function='RMSE',
    boosting_type='Plain',
    depth=8,
    gpu_cat_features_storage='CpuPinnedMemory',
    #max_ctr_complexity=2
model.fit(
    train_pool,
    eval_set = val_pool
    #plot=True
)
del train_pool, val_pool
gc.collect()
```

```
Learning rate set to 0.35204
0:      learn: 2.9428610        test: 2.9549983 best: 2.9549983 (0)
total: 1.32s    remaining: 21m 59s
200:    learn: 1.9990122        test: 2.0761268 best: 2.0761084 (199)
total: 3m 36s   remaining: 14m 22s
400:    learn: 1.9394297        test: 2.0586270 best: 2.0586270 (400)
total: 7m 10s   remaining: 10m 42s
600:    learn: 1.9001072        test: 2.0478947 best: 2.0478416 (598)
total: 10m 50s  remaining: 7m 11s
800:    learn: 1.8674147        test: 2.0435845 best: 2.0432455 (789)
total: 14m 27s  remaining: 3m 35s
999:    learn: 1.8398260        test: 2.0388751 best: 2.0387949 (998)
total: 18m 2s   remaining: 0us
bestTest = 2.038794938
bestIteration = 998
Shrink model to first 999 iterations.
```

Out[11]:

```
0
```

In [0]:

```
model.save_model('model_{}.cbm'.format(MODEL_VER))
```

Prediction Loop

In [ ]:

```
%%time
spred = get_df(is_train=False, backward_lags=BACKWARD_LAGS)
for pred_day in tqdm(range(1, 28 + 28 + 1)):
    pred_date = datetime.strptime(END_DATE, '%Y-%m-%d') + timedelta(days=pred_
    pred_date_back = pred_date - timedelta(days=BACKWARD_LAGS + 1)
    print('-' * 70)
    print('forecast day forward:', pred_day, '| forecast date:', pred_date)
    spred_data = spred[(spred['date'] >= pred_date_back) & (spred['date'] <= p
    spred_data = make_features(spred_data)
    spred_data = spred_data.loc[spred['date'] == pred_date, train_cols]
    spred_data[cat_cols] = spred_data[cat_cols].fillna(0)
    spred.loc[spred['date'] == pred_date, 'sales'] = model.predict(spred_data)
del spred_data
gc.collect()
```

Submission

In [14]:

```
spred_subm = spred.loc[spred['date'] > END_DATE, ['id', 'd', 'sales']].copy()
last_d = int(spred.loc[spred['date'] == END_DATE, 'd'].unique()[0].replace('d_
print('last d num:', last_d)
spred_subm['d'] = spred_subm['d'].apply(lambda x: 'F{}'.format(int(x.replace(
spred_subm.loc[spred_subm['sales'] < 0, 'sales'] = 0
```

```
last d num: 1913
```

In [0]:

```
f_cols = ['F{}'.format(x) for x in range(1, 28 + 28 + 1)]
spred_subm = spred_subm.set_index(['id', 'd']).unstack()['sales'][f_cols].rese
spred_subm.fillna(0, inplace=True)
spred_subm.sort_values('id', inplace=True)
spred_subm.reset_index(drop=True, inplace=True)
```

In [16]:

```
f_cols_val = ['F{}'.format(x) for x in range(1, 28 + 1)]
f_cols_eval = ['F{}'.format(x) for x in range(28 + 1, 28 + 28 + 1)]
spred_subm_eval = spred_subm.copy()
spred_subm.drop(columns=f_cols_eval, inplace=True)
spred_subm_eval.drop(columns=f_cols_val, inplace=True)
spred_subm_eval.columns = spred_subm.columns
spred_subm_eval['id'] = spred_subm_eval['id'].str.replace('validation', 'evalu
spred_subm = pd.concat([spred_subm, spred_subm_eval], axis=0, sort=False)
spred_subm.reset_index(drop=True, inplace=True)
spred_subm.to_csv('submission.csv', index=False)
print('submission saved:', spred_subm.shape)
```

```
submission saved: (60980, 29)
```