In [0]:

```python
from  datetime import datetime, timedelta
import gc
import numpy as np, pandas as pd
import lightgbm as lgb
```

In [0]:

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly (https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly)

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

In [0]:

```python
CAL_DTYPES={
    "event_name_1": "category",
    "event_name_2": "category",
    "event_type_1": "category",
    "event_type_2": "category",
    "weekday": "category",
    "wm_yr_wk": "int16",
    "wday": "int16",
    "month": "int16",
    "year": "int16",
    "snap_CA": "float32",
    "snap_TX": "float32",
    "snap_WI": "float32"
}
PRICE_DTYPES = {
    "store_id": "category",
     "item_id": "category",
     "wm_yr_wk": "int16",
     "sell_price":"float32"
}
```

In [0]:

```python
DATA_DIR = '/gdrive/My Drive/M5-forecasting/'
h = 28
BACKWARD_LAGS = 60
END_D = 1913
END_DATE = datetime(2016, 4, 25)
np.random.seed(0)
```

In [0]:

```python
def create_dt(is_train = True, nrows = None, first_day = 1500):
    prices = pd.read_csv(DATA_DIR+"sell_prices.csv", dtype = PRICE_DTYPES)
    for col, col_dtype in PRICE_DTYPES.items():
        if col_dtype == "category":
            prices[col] = prices[col].cat.codes.astype("int16")
            prices[col] -= prices[col].min()

    cal = pd.read_csv(DATA_DIR+"calendar.csv", dtype = CAL_DTYPES)
    cal["date"] = pd.to_datetime(cal["date"])

    for col, col_dtype in CAL_DTYPES.items():
        if col_dtype == "category":
            cal[col] = cal[col].cat.codes.astype("int16")
            cal[col] -= cal[col].min()

    start_day = max(1 if is_train  else END_D-BACKWARD_LAGS, first_day)
    numcols = [f"d_{day}" for day in range(start_day, END_D+1)]
    catcols = ['id', 'item_id', 'dept_id','store_id', 'cat_id', 'state_id']
    dtype = {numcol:"float32" for numcol in numcols}
    dtype.update({col: "category" for col in catcols if col != "id"})
    dt = pd.read_csv(DATA_DIR+"sales_train_validation.csv",
                     nrows = nrows, usecols = catcols + numcols, dtype = dtype

    for col in catcols:
        if col != "id":
            dt[col] = dt[col].cat.codes.astype("int16")
            dt[col] -= dt[col].min()

    if not is_train:
        for day in range(END_D +1, END_D + 28 +1):
            dt[f"d_{day}"] = np.nan

    dt = pd.melt(dt,
                 id_vars = catcols,
                 value_vars = [col for col in dt.columns if col.startswith("d
                 var_name = "d",
                 value_name = "sales")

    dt = dt.merge(cal, on= "d", copy = False)
    dt = dt.merge(prices, on = ["store_id", "item_id", "wm_yr_wk"], copy = Fal

    return dt
```

In [0]:

```python
def create_fea(dt):
    lags = [7, 28]
    lag_cols = [f"lag_{lag}" for lag in lags ]
    for lag, lag_col in zip(lags, lag_cols):
        dt[lag_col] = dt[["id","sales"]].groupby("id")["sales"].shift(lag)

    wins = [7, 28]
    for win in wins :
        for lag,lag_col in zip(lags, lag_cols):
            dt[f"rmean_{lag}_{win}"] = dt[["id", lag_col]].groupby("id")[lag_c



    date_features = {
        "wday": "weekday",
        "week": "weekofyear",
        "month": "month",
        "quarter": "quarter",
        "year": "year",
        "mday": "day",
#           "ime": "is_month_end",
#           "ims": "is_month_start",
    }

#     dt.drop(["d", "wm_yr_wk", "weekday"], axis=1, inplace = True)

    for date_feat_name, date_feat_func in date_features.items():
        if date_feat_name in dt.columns:
            dt[date_feat_name] = dt[date_feat_name].astype("int16")
        else:
            dt[date_feat_name] = getattr(dt["date"].dt, date_feat_func).astype
```

In [0]:

```python
FIRST_DAY = 1400
```

In [0]:

```python
%%time
df = create_dt(is_train=True, first_day= FIRST_DAY)
df.shape
```

```
CPU times: user 15.3 s, sys: 3.5 s, total: 18.8 s
Wall time: 22.3 s
```

In [0]:

```
df.head()
```

Out[12]:

	id	item_id	dept_id	store_id	cat_id	state_id	d
0	HOBBIES_1_001_CA_1_validation	0	0	0	0	0	d_14
1	HOBBIES_1_002_CA_1_validation	1	0	0	0	0	d_14
2	HOBBIES_1_003_CA_1_validation	2	0	0	0	0	d_14
3	HOBBIES_1_004_CA_1_validation	3	0	0	0	0	d_14
4	HOBBIES_1_005_CA_1_validation	4	0	0	0	0	d_14

In [0]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15537606 entries, 0 to 15537605
Data columns (total 22 columns):
 #   Column        Dtype
---  ------        -----
 0   id            object
 1   item_id       int16
 2   dept_id       int16
 3   store_id      int16
 4   cat_id        int16
 5   state_id      int16
 6   d             object
 7   sales         float32
 8   date          datetime64[ns]
 9   wm_yr_wk      int16
 10  weekday       int16
 11  wday          int16
 12  month         int16
 13  year          int16
 14  event_name_1  int16
 15  event_type_1  int16
 16  event_name_2  int16
 17  event_type_2  int16
 18  snap_CA       float32
 19  snap_TX       float32
 20  snap_WI       float32
 21  sell_price    float32
dtypes: datetime64[ns](1), float32(5), int16(14), object(2)
memory usage: 1.2+ GB
```

In [0]:

```
%%time
create_fea(df)
df.shape
```

```
CPU times: user 2min 3s, sys: 2.16 s, total: 2min 5s
Wall time: 2min 5s
```

In [0]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15537606 entries, 0 to 15537605
Data columns (total 31 columns):
 #   Column         Dtype         
---  ------         -----         
 0   id             object        
 1   item_id        int16         
 2   dept_id        int16         
 3   store_id       int16         
 4   cat_id         int16         
 5   state_id       int16         
 6   d              object        
 7   sales          float32       
 8   date           datetime64[ns]
 9   wm_yr_wk       int16         
 10  weekday        int16         
 11  wday           int16         
 12  month          int16         
 13  year           int16         
 14  event_name_1   int16         
 15  event_type_1   int16         
 16  event_name_2   int16         
 17  event_type_2   int16         
 18  snap_CA        float32       
 19  snap_TX        float32       
 20  snap_WI        float32       
 21  sell_price     float32       
 22  lag_7          float32       
 23  lag_28         float32       
 24  rmean_7_7      float32       
 25  rmean_28_7     float32       
 26  rmean_7_28     float32       
 27  rmean_28_28    float32       
 28  week           int16         
 29  quarter        int16         
 30  mday           int16         
dtypes: datetime64[ns](1), float32(11), int16(17), object(2)
memory usage: 1.6+ GB
```

In [0]:

```
df.head()
```

Out[16]:

	id	item_id	dept_id	store_id	cat_id	state_id	d
0	HOBBIES_1_001_CA_1_validation	0	0	0	0	0	d_14
1	HOBBIES_1_002_CA_1_validation	1	0	0	0	0	d_14
2	HOBBIES_1_003_CA_1_validation	2	0	0	0	0	d_14
3	HOBBIES_1_004_CA_1_validation	3	0	0	0	0	d_14
4	HOBBIES_1_005_CA_1_validation	4	0	0	0	0	d_14

In [0]:

```
df.dropna(inplace = True)
df.shape
```

Out[17]:

```
(13860656, 31)
```

In [0]:

```
cat_feats = ['item_id', 'dept_id','store_id', 'cat_id', 'state_id'] + ["event_
useless_cols = ["id", "date", "sales","d", "wm_yr_wk", "weekday"]
train_cols = df.columns[~df.columns.isin(useless_cols)]
X_train = df[train_cols]
y_train = df["sales"]
```

In [0]:

```
%%time

np.random.seed(777)

fake_valid_inds = np.random.choice(X_train.index.values, 2_000_000, replace =
train_inds = np.setdiff1d(X_train.index.values, fake_valid_inds)
train_data = lgb.Dataset(X_train.loc[train_inds] , label = y_train.loc[train_i
                         categorical_feature=cat_feats, free_raw_data=False)
fake_valid_data = lgb.Dataset(X_train.loc[fake_valid_inds], label = y_train.lo
                              categorical_feature=cat_feats,
                 free_raw_data=False)
```

```
CPU times: user 7.77 s, sys: 759 ms, total: 8.53 s
Wall time: 8.52 s
```

In [0]:

```
del df, X_train, y_train, fake_valid_inds,train_inds ; gc.collect()
```

Out[20]:

```
54
```

In [0]:

```
params = {
        "objective" : "poisson",
        "metric" :"rmse",
        "force_row_wise" : True,
        "learning_rate" : 0.075,
        "sub_row" : 0.75,
        "bagging_freq" : 1,
        "lambda_l2" : 0.1,
        "metric": ["rmse"],
    'verbosity': 1,
    'num_iterations' : 1200,
    'num_leaves': 128,
    "min_data_in_leaf": 100,
}
```

In [0]:

```

```

In [0]:

```
%%time

m_lgb = lgb.train(params,
                  train_data,
                  valid_sets = [fake_valid_data],
                  verbose_eval=20)
```

```
/usr/local/lib/python3.6/dist-packages/lightgbm/engine.py:118: UserWar
ning: Found `num_iterations` in params. Will use it instead of argumen
t
  warnings.warn("Found `{}` in params. Will use it instead of argumen
t".format(alias))
/usr/local/lib/python3.6/dist-packages/lightgbm/basic.py:1205: UserWar
ning: Using categorical_feature in Dataset.
  warnings.warn('Using categorical_feature in Dataset.')
/usr/local/lib/python3.6/dist-packages/lightgbm/basic.py:762: UserWarn
ing: categorical_feature in param dict is overridden.
  warnings.warn('categorical_feature in param dict is overridden.')

[20]	valid_0's rmse: 2.48775
[40]	valid_0's rmse: 2.2178
[60]	valid_0's rmse: 2.14983
[80]	valid_0's rmse: 2.13033
[100]	valid_0's rmse: 2.12059
[120]	valid_0's rmse: 2.11354
[140]	valid_0's rmse: 2.10844
[160]	valid_0's rmse: 2.10244
[180]	valid_0's rmse: 2.0966
[200]	valid_0's rmse: 2.09106
[220]	valid_0's rmse: 2.08791
[240]	valid_0's rmse: 2.08513
[260]	valid_0's rmse: 2.08301
[280]	valid_0's rmse: 2.07984
[300]	valid_0's rmse: 2.07798
[320]	valid_0's rmse: 2.07543
[340]	valid_0's rmse: 2.07371
[360]	valid_0's rmse: 2.07257
[380]	valid_0's rmse: 2.06983
[400]	valid_0's rmse: 2.06768
[420]	valid_0's rmse: 2.06607
[440]	valid_0's rmse: 2.06339
[460]	valid_0's rmse: 2.0616
[480]	valid_0's rmse: 2.06036
[500]	valid_0's rmse: 2.05916
[520]	valid_0's rmse: 2.05687
[540]	valid_0's rmse: 2.05527
[560]	valid_0's rmse: 2.05342
[580]	valid_0's rmse: 2.05213
[600]	valid_0's rmse: 2.05065
[620]	valid_0's rmse: 2.04926
[640]	valid_0's rmse: 2.04803
[660]	valid_0's rmse: 2.04658
[680]	valid_0's rmse: 2.04555
[700]	valid_0's rmse: 2.04381
[720]	valid_0's rmse: 2.0427
[740]	valid_0's rmse: 2.04128
[760]	valid_0's rmse: 2.03945
[780]	valid_0's rmse: 2.03806
[800]	valid_0's rmse: 2.03678
```

```
[820]	valid_0's rmse: 2.03555
[840]	valid_0's rmse: 2.03418
[860]	valid_0's rmse: 2.03366
[880]	valid_0's rmse: 2.03287
[900]	valid_0's rmse: 2.032
[920]	valid_0's rmse: 2.03138
[940]	valid_0's rmse: 2.03073
[960]	valid_0's rmse: 2.03009
[980]	valid_0's rmse: 2.0289
[1000]	valid_0's rmse: 2.02778
[1020]	valid_0's rmse: 2.02742
[1040]	valid_0's rmse: 2.02666
[1060]	valid_0's rmse: 2.0259
[1080]	valid_0's rmse: 2.02548
[1100]	valid_0's rmse: 2.02461
[1120]	valid_0's rmse: 2.02402
[1140]	valid_0's rmse: 2.02324
[1160]	valid_0's rmse: 2.02286
[1180]	valid_0's rmse: 2.0225
[1200]	valid_0's rmse: 2.02174
CPU times: user 2h 12min 40s, sys: 10.5 s, total: 2h 12min 50s
Wall time: 1h 8min 3s
```

In [0]:

```
m_lgb.save_model("model.lgb")
```

Out[27]:

```
<lightgbm.basic.Booster at 0x7fb60e6dcac8>
```

In [0]:

```python
%%time

alphas = [1.028, 1.023, 1.018]
weights = [1 / len(alphas)]*len(alphas)
sub = 0.

for icount, (alpha, weight) in enumerate(zip(alphas, weights)):

    te = create_dt(False)
    cols = [f"F{i}" for i in range(1,29)]

    for tdelta in range(0, 28):
        day = END_DATE + timedelta(days=tdelta)
        print(tdelta, day)
        tst = te[(te.date >= day - timedelta(days=BACKWARD_LAGS)) & (te.date <
        create_fea(tst)
        tst = tst.loc[tst.date == day , train_cols]
        te.loc[te.date == day, "sales"] = alpha * m_lgb.predict(tst) # magic m



    te_sub = te.loc[te.date >= END_DATE, ["id", "sales"]].copy()

    te_sub["F"] = [f"F{rank}" for rank in te_sub.groupby("id")["id"].cumcount(
    te_sub = te_sub.set_index(["id", "F" ]).unstack()["sales"][cols].reset_inc
    te_sub.fillna(0., inplace = True)
    te_sub.sort_values("id", inplace = True)
    te_sub.reset_index(drop=True, inplace = True)
    te_sub.to_csv(f"submission_{icount}.csv",index=False)
    if icount == 0 :
        sub = te_sub
        sub[cols] *= weight
    else:
        sub[cols] += te_sub[cols]*weight
    print(icount, alpha, weight)


sub2 = sub.copy()
sub2["id"] = sub2["id"].str.replace("validation$", "evaluation")
sub = pd.concat([sub, sub2], axis=0, sort=False)
sub.to_csv("submission.csv",index=False)
```

```
0 2016-04-25 00:00:00
1 2016-04-26 00:00:00
2 2016-04-27 00:00:00
3 2016-04-28 00:00:00
4 2016-04-29 00:00:00
5 2016-04-30 00:00:00
6 2016-05-01 00:00:00
7 2016-05-02 00:00:00
8 2016-05-03 00:00:00
9 2016-05-04 00:00:00
10 2016-05-05 00:00:00
11 2016-05-06 00:00:00
12 2016-05-07 00:00:00
13 2016-05-08 00:00:00
14 2016-05-09 00:00:00
15 2016-05-10 00:00:00
16 2016-05-11 00:00:00
```

```
17 2016-05-12 00:00:00
18 2016-05-13 00:00:00
19 2016-05-14 00:00:00
20 2016-05-15 00:00:00
21 2016-05-16 00:00:00
22 2016-05-17 00:00:00
23 2016-05-18 00:00:00
24 2016-05-19 00:00:00
25 2016-05-20 00:00:00
26 2016-05-21 00:00:00
27 2016-05-22 00:00:00
0 1.028 0.3333333333333333
0 2016-04-25 00:00:00
1 2016-04-26 00:00:00
2 2016-04-27 00:00:00
3 2016-04-28 00:00:00
4 2016-04-29 00:00:00
5 2016-04-30 00:00:00
6 2016-05-01 00:00:00
7 2016-05-02 00:00:00
8 2016-05-03 00:00:00
9 2016-05-04 00:00:00
10 2016-05-05 00:00:00
11 2016-05-06 00:00:00
12 2016-05-07 00:00:00
13 2016-05-08 00:00:00
14 2016-05-09 00:00:00
15 2016-05-10 00:00:00
16 2016-05-11 00:00:00
17 2016-05-12 00:00:00
18 2016-05-13 00:00:00
19 2016-05-14 00:00:00
20 2016-05-15 00:00:00
21 2016-05-16 00:00:00
22 2016-05-17 00:00:00
23 2016-05-18 00:00:00
24 2016-05-19 00:00:00
25 2016-05-20 00:00:00
26 2016-05-21 00:00:00
27 2016-05-22 00:00:00
1 1.023 0.3333333333333333
0 2016-04-25 00:00:00
1 2016-04-26 00:00:00
2 2016-04-27 00:00:00
3 2016-04-28 00:00:00
4 2016-04-29 00:00:00
5 2016-04-30 00:00:00
6 2016-05-01 00:00:00
7 2016-05-02 00:00:00
8 2016-05-03 00:00:00
9 2016-05-04 00:00:00
10 2016-05-05 00:00:00
11 2016-05-06 00:00:00
12 2016-05-07 00:00:00
13 2016-05-08 00:00:00
14 2016-05-09 00:00:00
15 2016-05-10 00:00:00
16 2016-05-11 00:00:00
17 2016-05-12 00:00:00
18 2016-05-13 00:00:00
19 2016-05-14 00:00:00
```

```
20 2016-05-15 00:00:00
21 2016-05-16 00:00:00
22 2016-05-17 00:00:00
23 2016-05-18 00:00:00
24 2016-05-19 00:00:00
25 2016-05-20 00:00:00
26 2016-05-21 00:00:00
27 2016-05-22 00:00:00
2 1.018 0.3333333333333333
CPU times: user 2h 22min 52s, sys: 37.4 s, total: 2h 23min 30s
Wall time: 2h 11min 44s
```

In [0]:

```
sub.head(10)
```

Out[26]:

F	id	F1	F2	F3	F4	F5	F6
0	FOODS_1_001_CA_1_validation	0.694917	0.752471	0.754582	0.753150	1.052376	1.186456
1	FOODS_1_001_CA_2_validation	0.764737	0.713269	0.658067	0.674257	0.887919	1.108696
2	FOODS_1_001_CA_3_validation	0.661003	0.613899	0.594123	0.665871	0.813425	1.324989
3	FOODS_1_001_CA_4_validation	0.388224	0.321583	0.348039	0.351394	0.370363	0.413981
4	FOODS_1_001_TX_1_validation	0.179126	0.181186	0.179805	0.181161	0.195914	0.212531
5	FOODS_1_001_TX_2_validation	0.473003	0.426398	0.435832	0.437532	0.480572	0.580027
6	FOODS_1_001_TX_3_validation	0.404118	0.389440	0.401637	0.471192	0.494556	0.471899
7	FOODS_1_001_WI_1_validation	0.348886	0.414848	0.382348	0.396833	0.500632	0.506067
8	FOODS_1_001_WI_2_validation	0.318699	0.317227	0.343764	0.346648	0.419075	0.444170
9	FOODS_1_001_WI_3_validation	0.236639	0.242035	0.232483	0.235570	0.261549	0.315759

In [0]:

```
sub.head(10)
```

Out[23]:

F	id	F1	F2	F3	F4	F5	F6
0	FOODS_1_001_CA_1_validation	0.910026	0.847982	0.850890	0.801789	1.071719	1.296671
1	FOODS_1_001_CA_2_validation	0.940942	0.954666	0.882539	1.273918	1.290628	1.354433
2	FOODS_1_001_CA_3_validation	1.091795	1.044925	0.949970	0.917520	0.990185	1.107758
3	FOODS_1_001_CA_4_validation	0.414818	0.361572	0.355099	0.349345	0.405143	0.451183
4	FOODS_1_001_TX_1_validation	0.180371	0.179159	0.170700	0.171773	0.172805	0.180151
5	FOODS_1_001_TX_2_validation	0.482585	0.436249	0.445677	0.395896	0.447061	0.496326
6	FOODS_1_001_TX_3_validation	0.398578	0.368744	0.376498	0.413743	0.451219	0.457103
7	FOODS_1_001_WI_1_validation	0.331981	0.372827	0.364709	0.363885	0.447759	0.680631
8	FOODS_1_001_WI_2_validation	0.310635	0.340609	0.353715	0.338573	0.414242	0.423919
9	FOODS_1_001_WI_3_validation	0.237874	0.235062	0.209730	0.220884	0.267701	0.319444

In [0]:

```
sub.id.nunique(), sub["id"].str.contains("validation$").sum()
```

Out[24]:

```
(60980, 30490)
```

In [0]:

```
sub.shape
```

Out[25]:

```
(60980, 29)
```

In [0]:

```

```

In [0]:

```

```