```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt, rcParams, style
style.use('seaborn-darkgrid')
import seaborn as sns
sns.set_style('darkgrid')
from plotly import express as px, graph_objects as go

from sklearn.preprocessing import RobustScaler, StandardScaler, Normalizer, MinM
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.ensemble import RandomForestRegressor, ExtraTreesRegressor, Bagging

import gc
gc.enable()
from warnings import filterwarnings, simplefilter
filterwarnings('ignore')
simplefilter('ignore')
```

```
train = pd.read_csv('train.csv',
                    parse_dates = ['date'], infer_datetime_format = True,
                    dtype = {'store_nbr' : 'category',
                             'family' : 'category'},
                    usecols = ['date', 'store_nbr', 'family', 'sales'])
train['date'] = train.date.dt.to_period('D')
train = train.set_index(['date', 'store_nbr', 'family']).sort_index()
train
```

```
                                                           sales
      date  store_nbr                          family
2013-01-01          1                      AUTOMOTIVE      0.000
                                            BABY CARE      0.000
                                               BEAUTY      0.000
                                            BEVERAGES      0.000
                                                BOOKS      0.000
       ...        ...                             ...        ...
2017-08-15          9                         POULTRY    438.133
                                       PREPARED FOODS    154.553
                                              PRODUCE   2419.729
                           SCHOOL AND OFFICE SUPPLIES    121.000
                                              SEAFOOD     16.000

3000888 rows × 1 columns
```

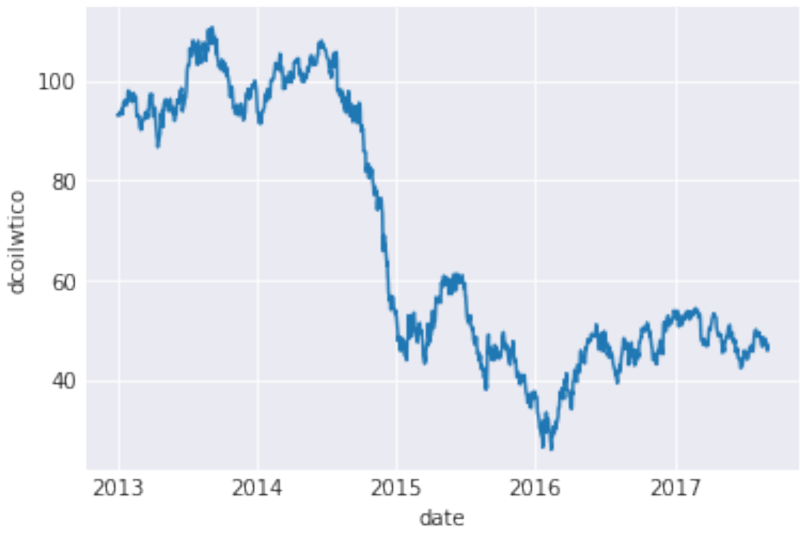
```
test = pd.read_csv('test.csv',
                   parse_dates = ['date'], infer_datetime_format = True)
test['date'] = test.date.dt.to_period('D')
test = test.set_index(['date', 'store_nbr', 'family']).sort_values('id')
test
```

			id	onpromotion
date	**store_nbr**	**family**		
2017-08-16	**1**	**AUTOMOTIVE**	3000888	0
		BABY CARE	3000889	0
		BEAUTY	3000890	2
		BEVERAGES	3000891	20
		BOOKS	3000892	0
...	**...**	**...**	...	...
2017-08-31	**9**	**POULTRY**	3029395	1
		PREPARED FOODS	3029396	0
		PRODUCE	3029397	1
		SCHOOL AND OFFICE SUPPLIES	3029398	9
		SEAFOOD	3029399	0

28512 rows × 2 columns

```
calendar = pd.DataFrame(index = pd.date_range('2013-01-01', '2017-08-31')).to_pe
oil = pd.read_csv('oil.csv',
                  parse_dates = ['date'], infer_datetime_format = True,
                  index_col = 'date').to_period('D')
oil['avg_oil'] = oil['dcoilwtico'].rolling(7).mean()
calendar = calendar.join(oil.avg_oil)
calendar['avg_oil'].fillna(method = 'ffill', inplace = True)
calendar.dropna(inplace = True)
```

```
# Plotting oil price
_ = sns.lineplot(data = oil.dcoilwtico.to_timestamp())
```

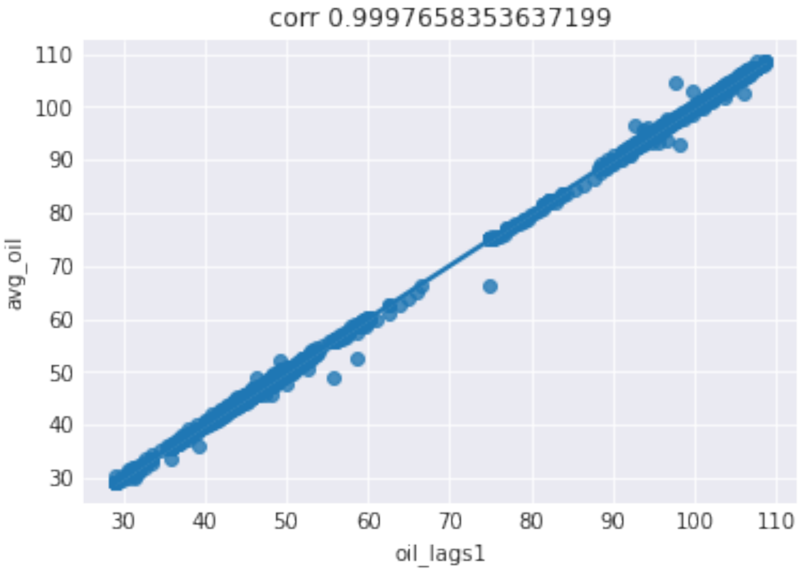


```
n_lags = 3
for l in range(1, n_lags + 1) :
    calendar[f'oil_lags{l}'] = calendar.avg_oil.shift(l)
calendar.dropna(inplace = True)
calendar
```

	avg_oil	oil_lags1	oil_lags2	oil_lags3
2013-01-13	93.284286	93.284286	93.284286	93.218571
2013-01-14	93.470000	93.284286	93.284286	93.284286
2013-01-15	93.490000	93.470000	93.284286	93.284286
2013-01-16	93.644286	93.490000	93.470000	93.284286
2013-01-17	93.970000	93.644286	93.490000	93.470000
...	...	...	...	...
2017-08-27	47.720000	47.720000	47.720000	47.598571
2017-08-28	47.624286	47.720000	47.720000	47.720000
2017-08-29	47.320000	47.624286	47.720000	47.720000
2017-08-30	47.115714	47.320000	47.624286	47.720000
2017-08-31	47.060000	47.115714	47.320000	47.624286

1692 rows × 4 columns

```
lag = 'oil_lags1'
plt.figure()
sns.regplot(x = calendar[lag], y = calendar.avg_oil)
plt.title(f'corr {calendar.avg_oil.corr(calendar[lag])}')
plt.show()
```



```
hol = pd.read_csv('holidays_events.csv',
                  parse_dates = ['date'], infer_datetime_format = True,
                  index_col = 'date').to_period('D')
hol = hol[hol.locale == 'National'] # I'm only taking National holiday so there'
hol = hol.groupby(hol.index).first() # Removing duplicated holiday at the same d
hol
```

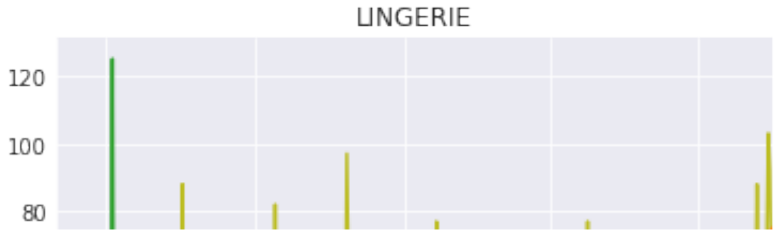
	type	locale	locale_name	description	transferred
date					
2012-08-10	Holiday	National	Ecuador	Primer Grito de Independencia	False
2012-10-09	Holiday	National	Ecuador	Independencia de Guayaquil	True
2012-10-12	Transfer	National	Ecuador	Traslado Independencia de Guayaquil	False
2012-11-02	Holiday	National	Ecuador	Dia de Difuntos	False
2012-11-03	Holiday	National	Ecuador	Independencia de Cuenca	False
...	...	...	...	...	...
2017-12-22	Additional	National	Ecuador	Navidad-3	False
2017-12-23	Additional	National	Ecuador	Navidad-2	False

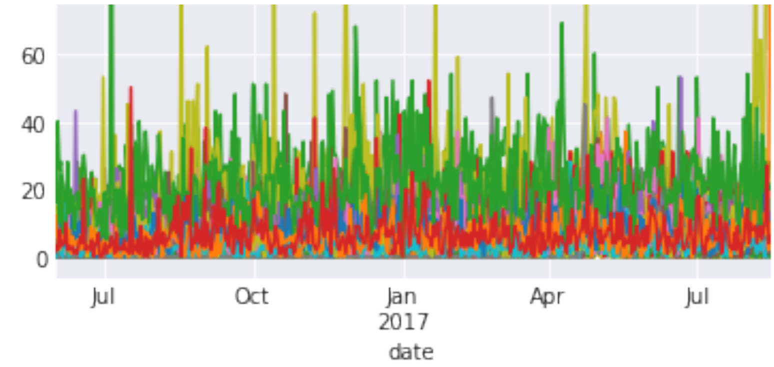
```
calendar = calendar.join(hol) # Joining calendar with holiday dataset
calendar['dofw'] = calendar.index.dayofweek # Weekly day
calendar['wd'] = 1
calendar.loc[calendar.dofw > 4, 'wd'] = 0 # If it's saturday or sunday then it's
calendar.loc[calendar.type == 'Work Day', 'wd'] = 1 # If it's Work Day event the
calendar.loc[calendar.type == 'Transfer', 'wd'] = 0 # If it's Transfer event the
calendar.loc[calendar.type == 'Bridge', 'wd'] = 0 # If it's Bridge event then it
calendar.loc[(calendar.type == 'Holiday') & (calendar.transferred == False), 'wd
calendar.loc[(calendar.type == 'Holiday') & (calendar.transferred == True), 'wd'
calendar = pd.get_dummies(calendar, columns = ['dofw'], drop_first = True) # One
calendar = pd.get_dummies(calendar, columns = ['type']) # One-hot encoding for t
calendar.drop(['locale', 'locale_name', 'description', 'transferred'], axis = 1,
calendar
```

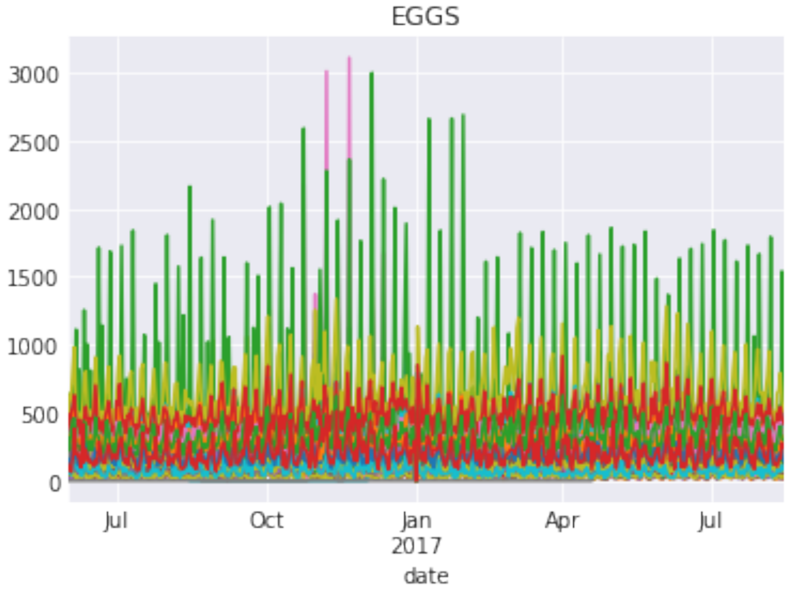
	avg_oil	oil_lags1	oil_lags2	oil_lags3	wd	dofw_1	dofw_2	dofw_3
2013-01-13	93.284286	93.284286	93.284286	93.218571	0	0	0	0
2013-01-14	93.470000	93.284286	93.284286	93.284286	1	0	0	0
2013-01-15	93.490000	93.470000	93.284286	93.284286	1	1	0	0
2013-01-16	93.644286	93.490000	93.470000	93.284286	1	0	1	0
2013-01-17	93.970000	93.644286	93.490000	93.470000	1	0	0	1
...	...	...	...	...	...	...	...	...
2017-08-27	47.720000	47.720000	47.720000	47.598571	0	0	0	0
2017-08-28	47.624286	47.720000	47.720000	47.720000	1	0	0	0
2017-08-29	47.320000	47.624286	47.720000	47.720000	1	1	0	0
2017-08-30	47.115714	47.320000	47.624286	47.720000	1	0	1	0
2017-08-31	47.060000	47.115714	47.320000	47.624286	1	0	0	1

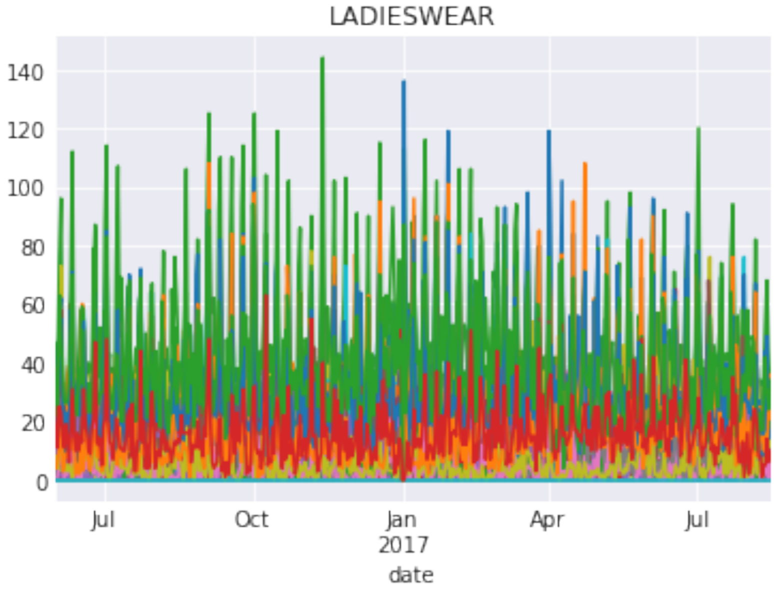
1692 rows × 17 columns

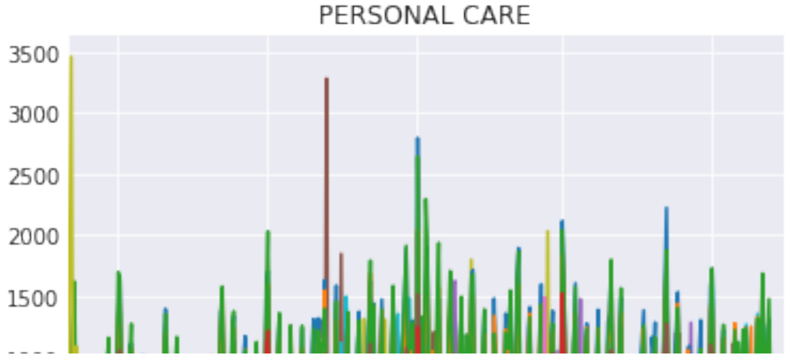
```
y = train.unstack(['store_nbr', 'family']).loc['2016-06':'2017']
family = {c[2] for c in train.index}
for f in family :
    ax = y.loc(axis = 1)['sales', :, f].plot(legend = None)
    ax.set_title(f)
```

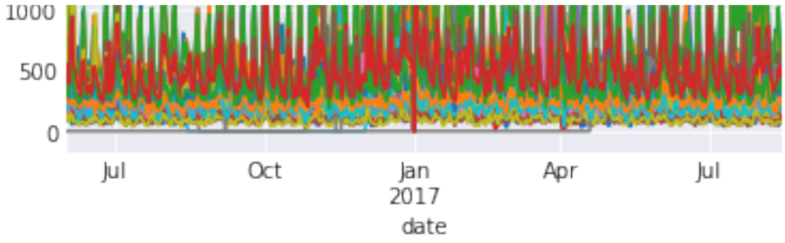



60
40
20
0
Jul
Oct
Jan
2017
Apr
Jul
date

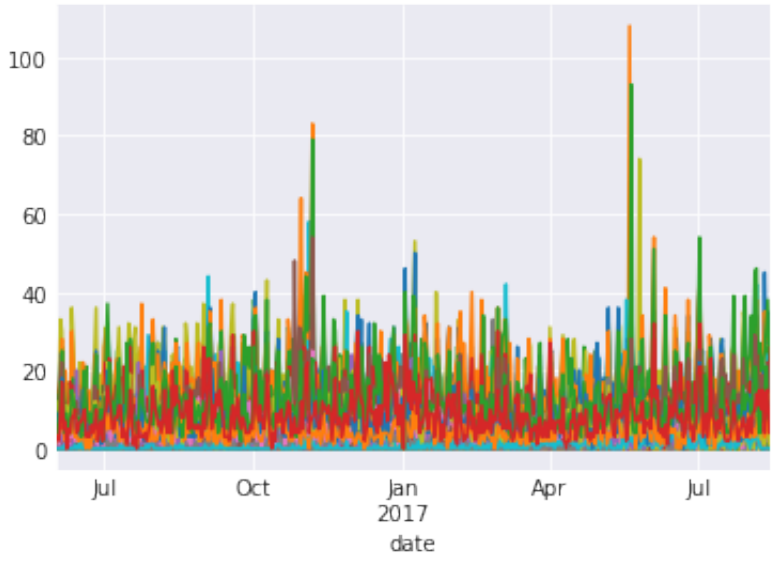

EGGS
3000
2500
2000
1500
1000
500
0
Jul
Oct
Jan
2017
Apr
Jul
date


LADIESWEAR
140
120
100
80
60
40
20
0
Jul
Oct
Jan
2017
Apr
Jul
date

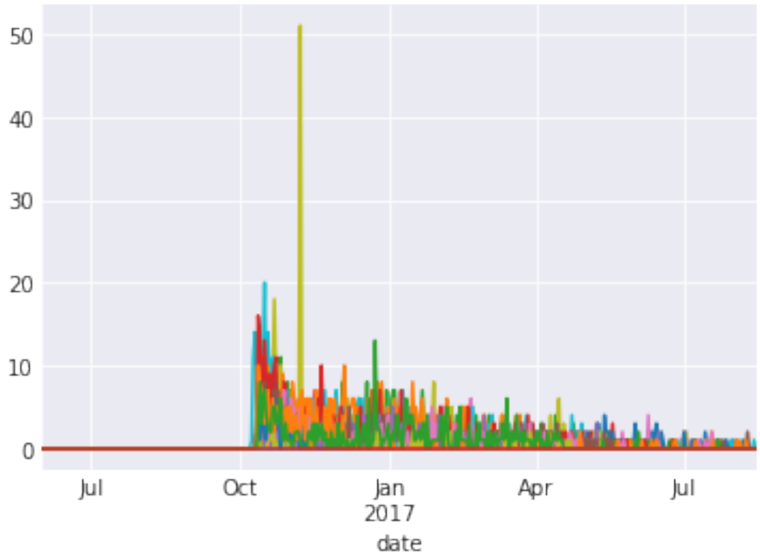

PERSONAL CARE
3500
3000
2500
2000
1500



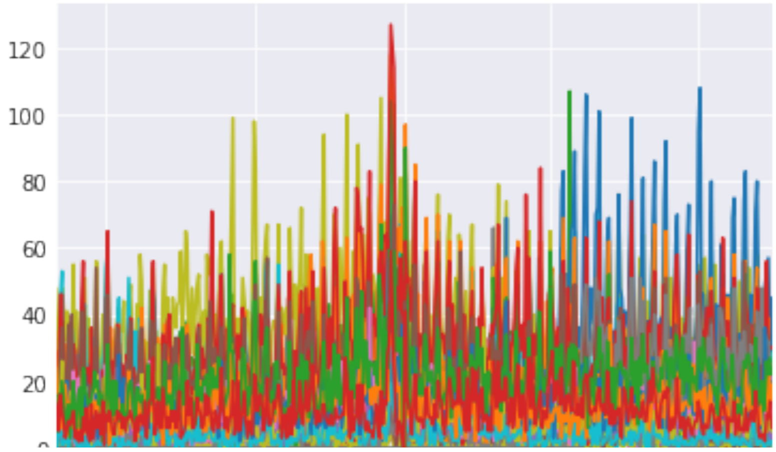
BEAUTY



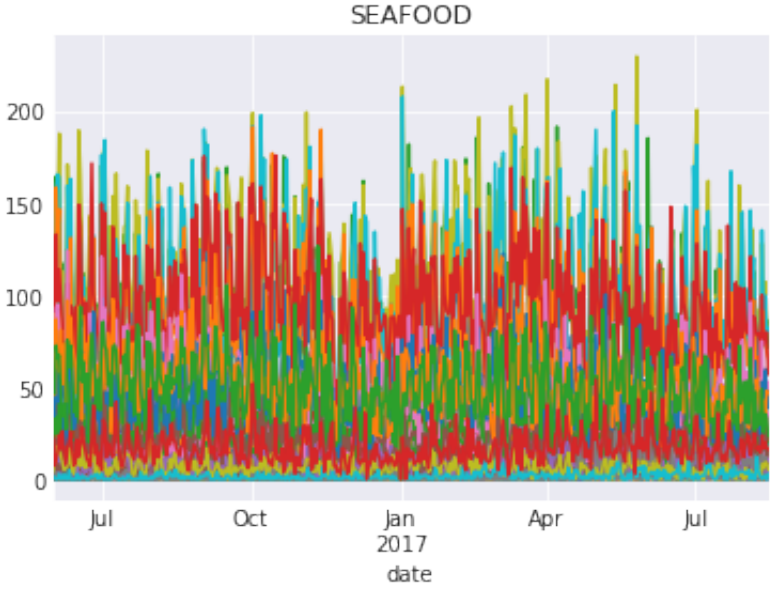
BOOKS

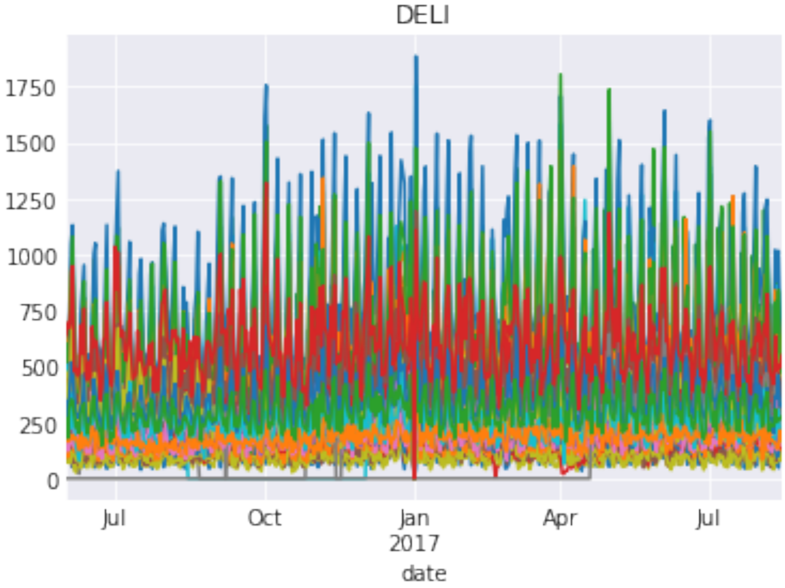


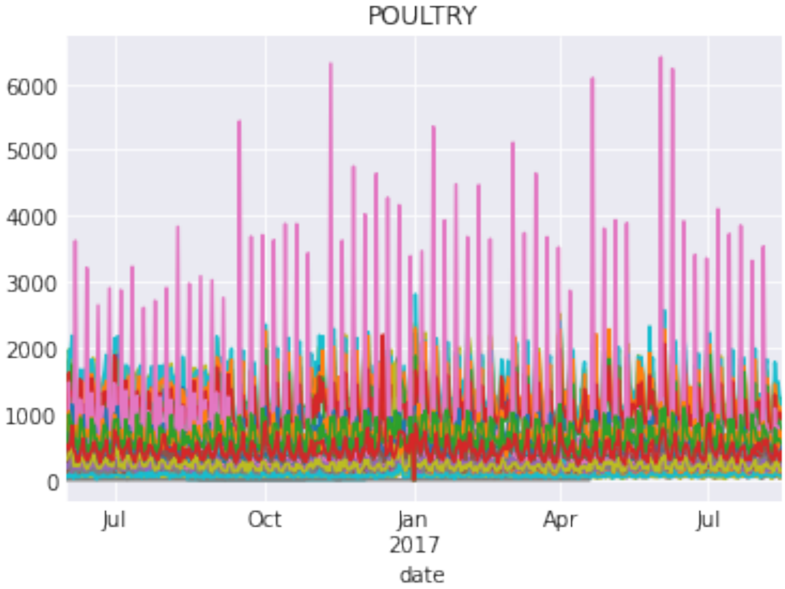
PLAYERS AND ELECTRONICS

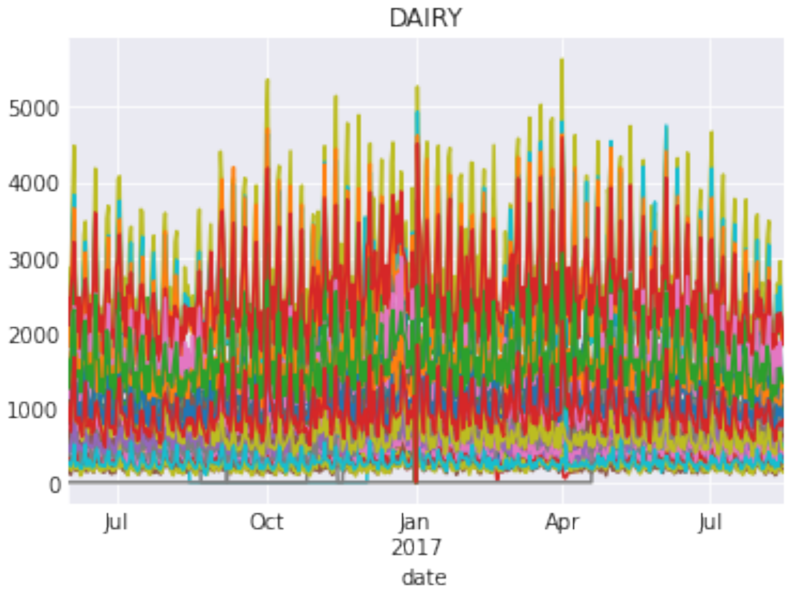


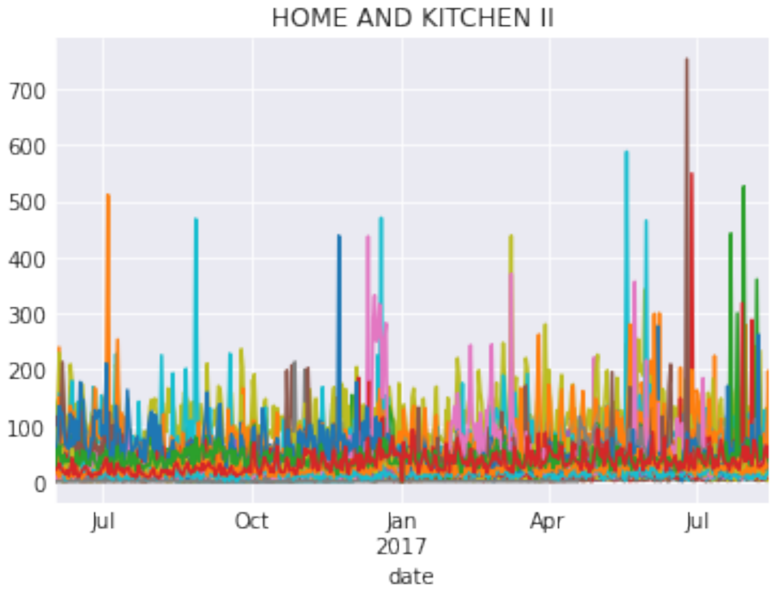
0
Jul
Oct
Jan
2017
Apr
Jul
date

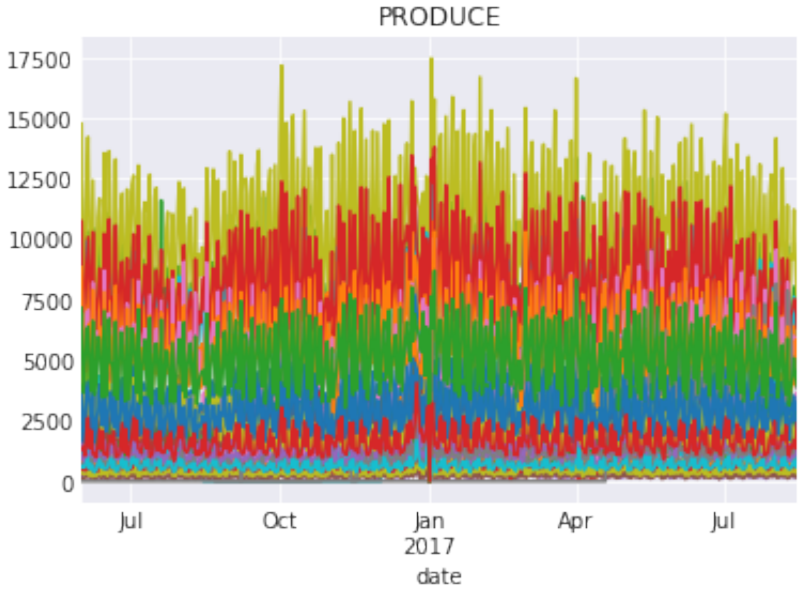

SEAFOOD
200
150
100
50
0
Jul
Oct
Jan
2017
Apr
Jul
date

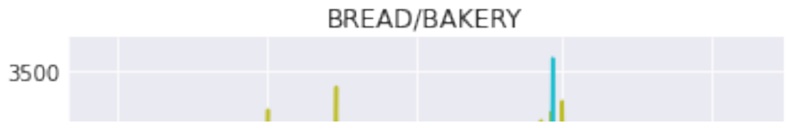

DELI
1750
1500
1250
1000
750
500
250
0
Jul
Oct
Jan
2017
Apr
Jul
date

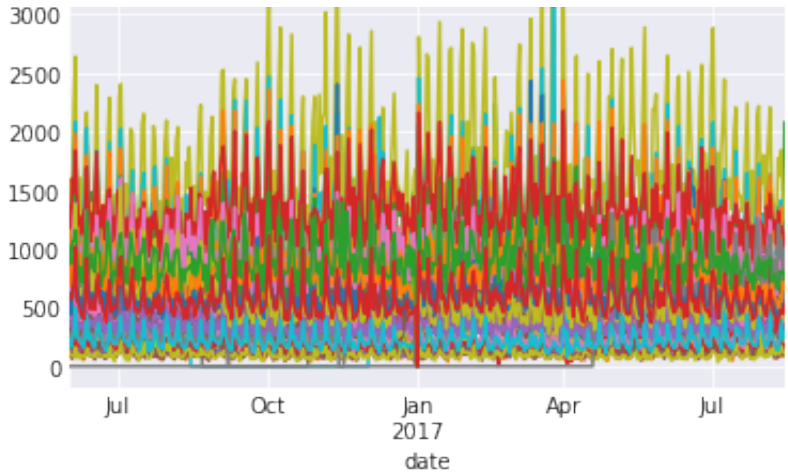

POULTRY
6000
5000
4000
3000
2000
1000
0
Jul
Oct
Jan
2017
Apr
Jul
date


DAIRY
5000
4000
3000
2000
1000
0
Jul
Oct
Jan
2017
Apr
Jul
date

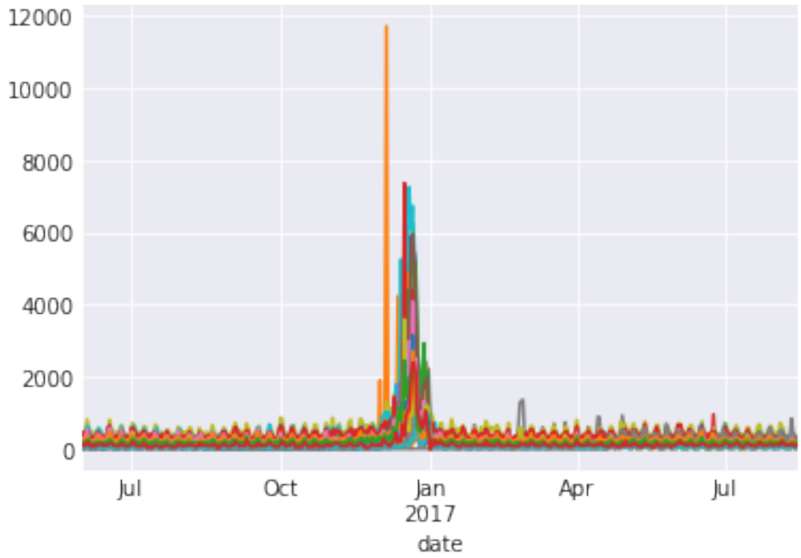

HOME AND KITCHEN II
700
600
500
400
300
200
100
0
Jul
Oct
Jan
2017
Apr
Jul
date


PRODUCE
17500
15000
12500
10000
7500
5000
2500
0
Jul
Oct
Jan
2017
Apr
Jul
date

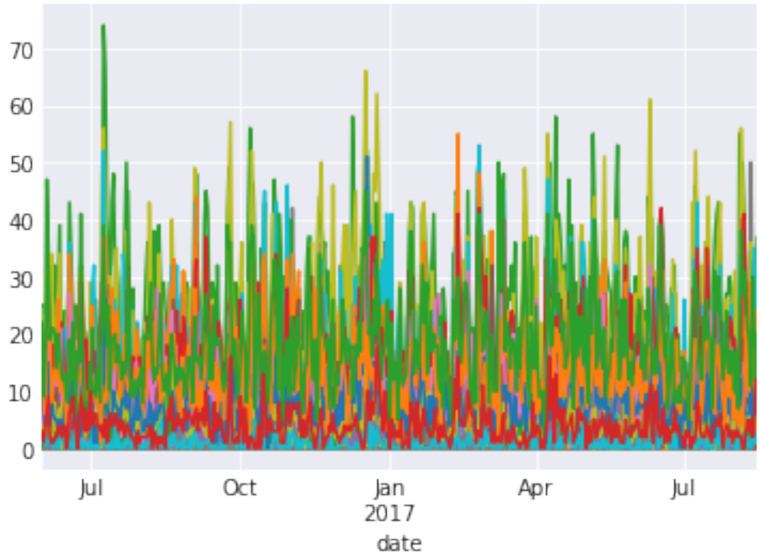

BREAD/BAKERY
3500



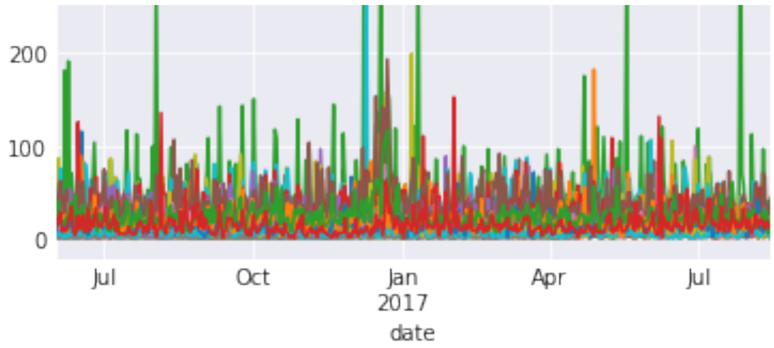
FROZEN FOODS



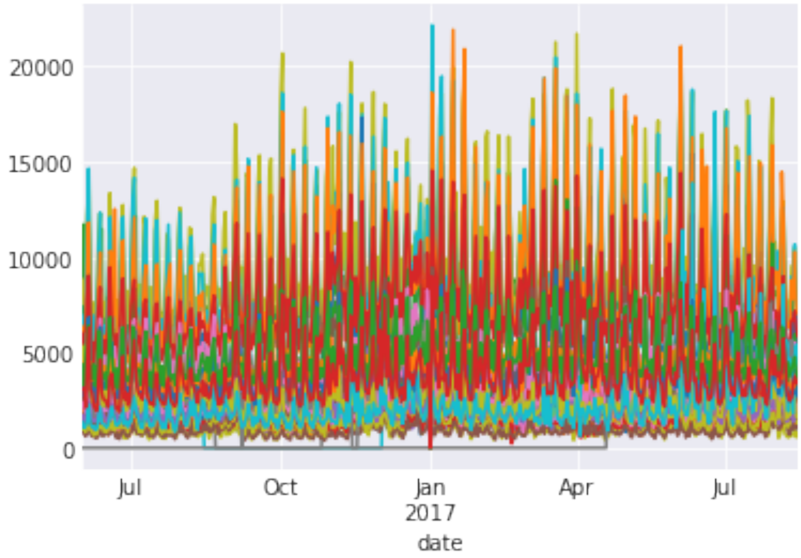
MAGAZINES



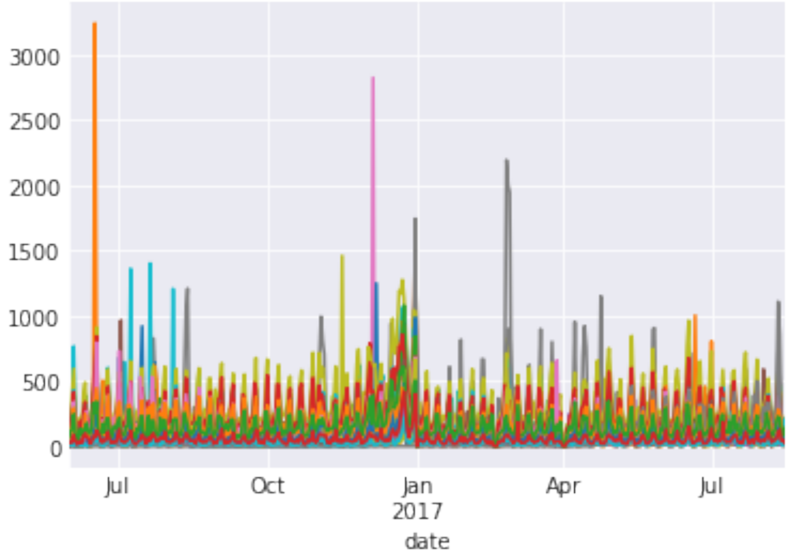
CELEBRATION



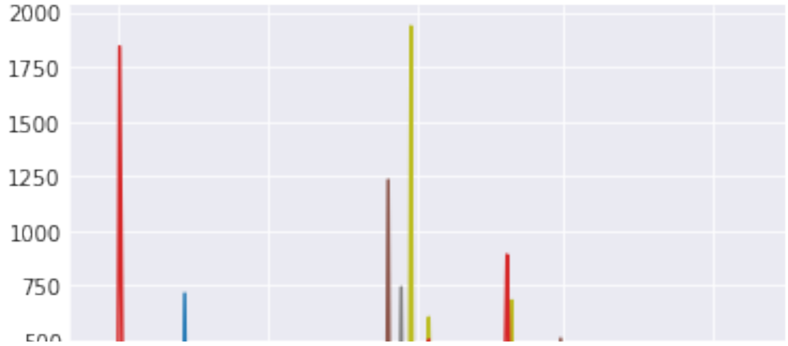
BEVERAGES

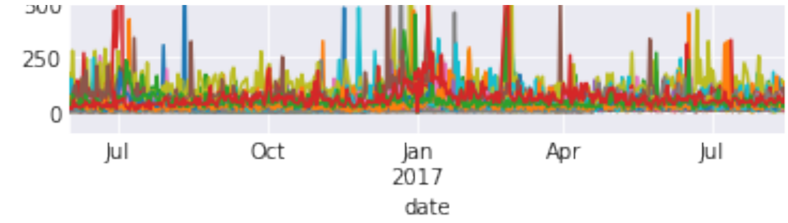


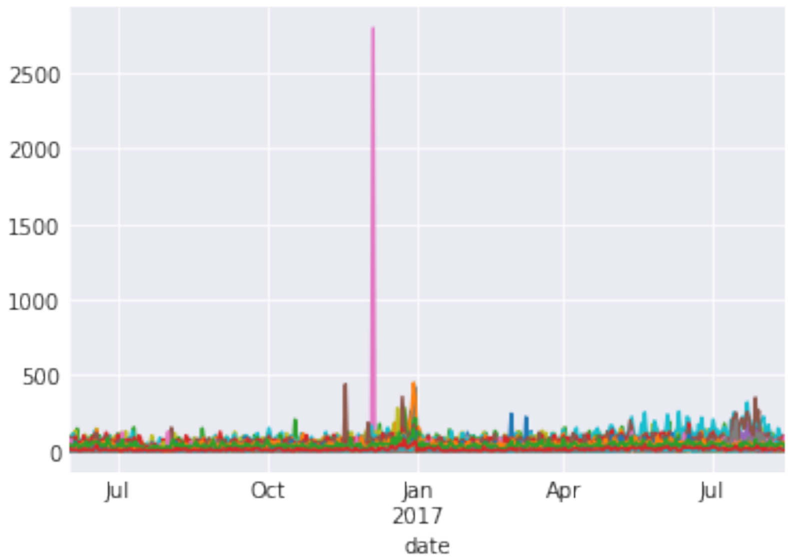
LIQUOR,WINE,BEER

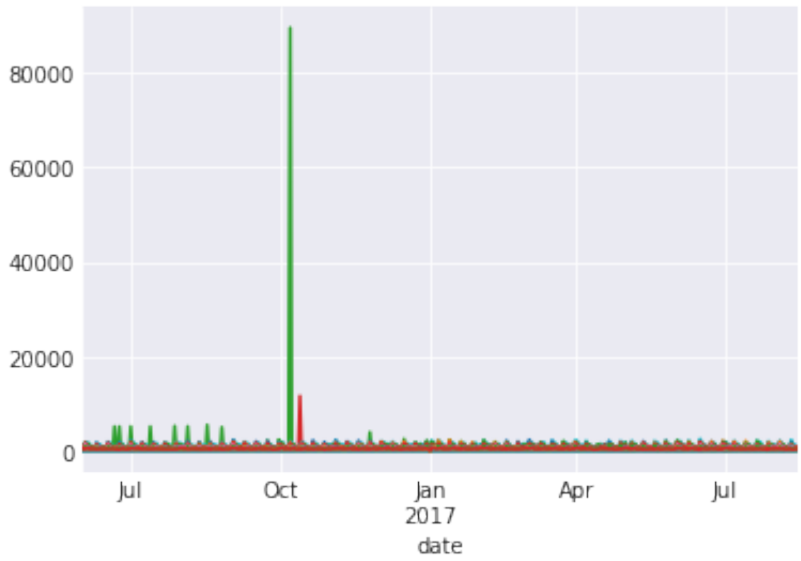


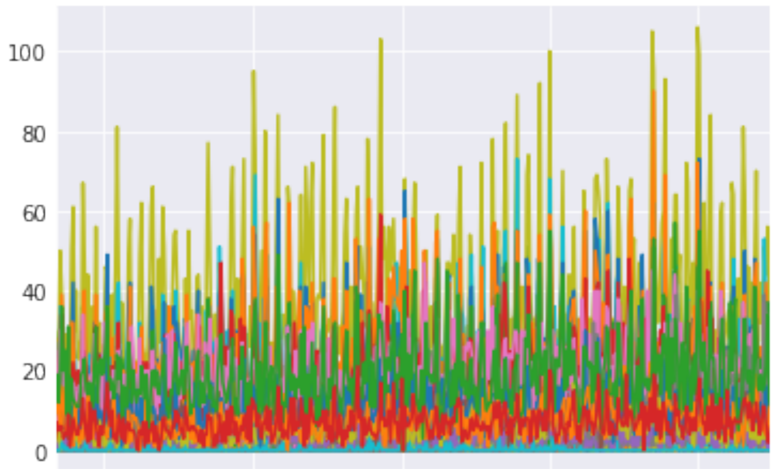
HOME AND KITCHEN I



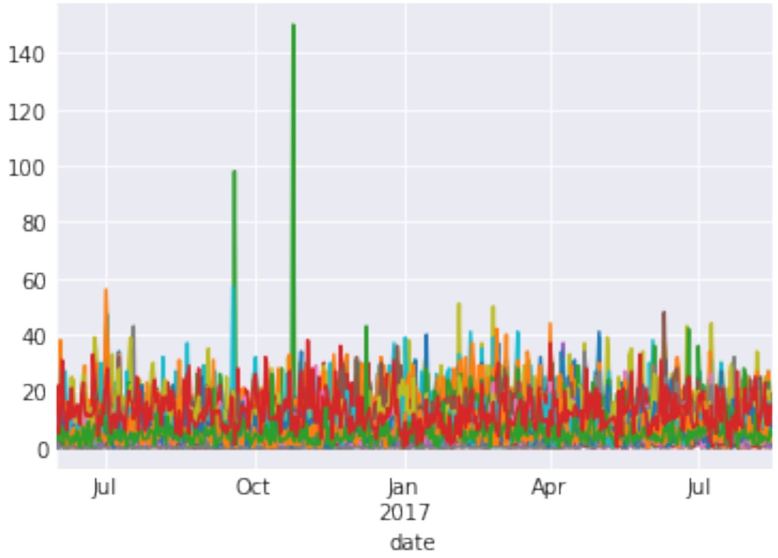



GROCERY II

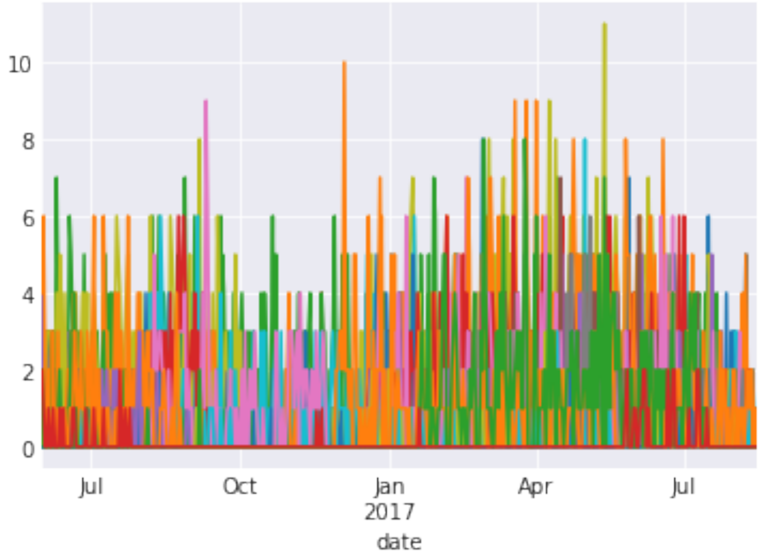

MEATS


PET SUPPLIES

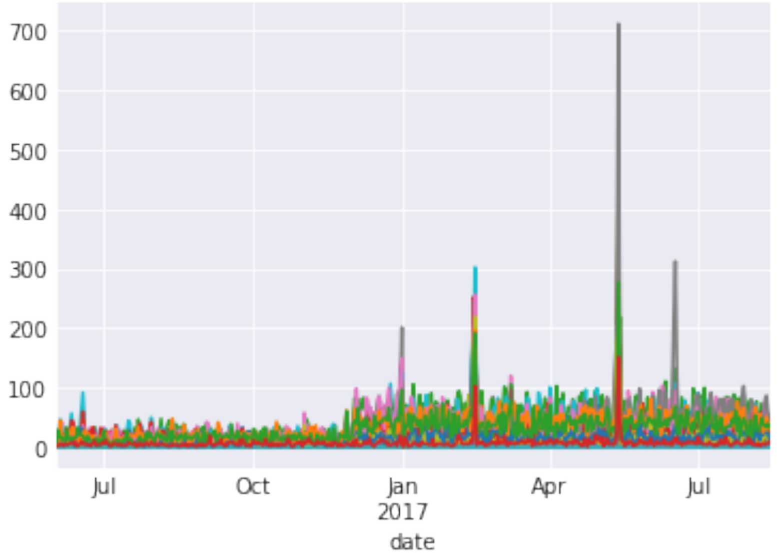
AUTOMOTIVE



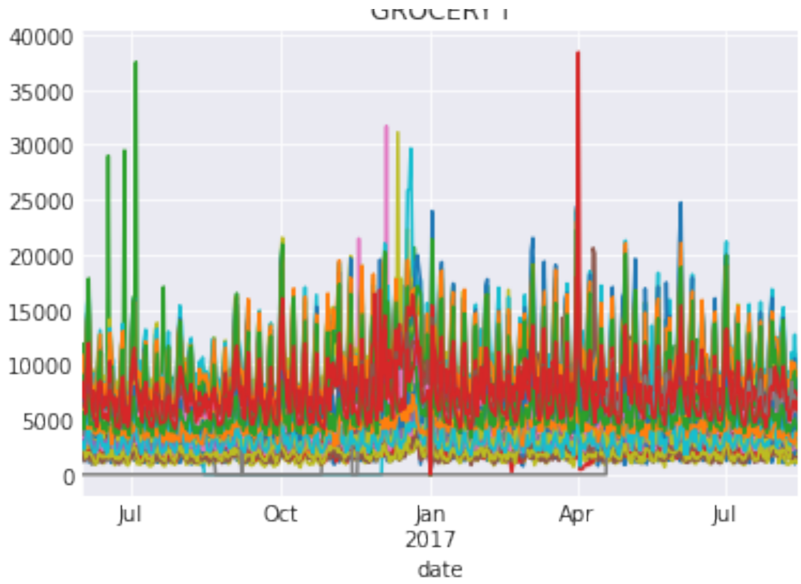
HOME APPLIANCES

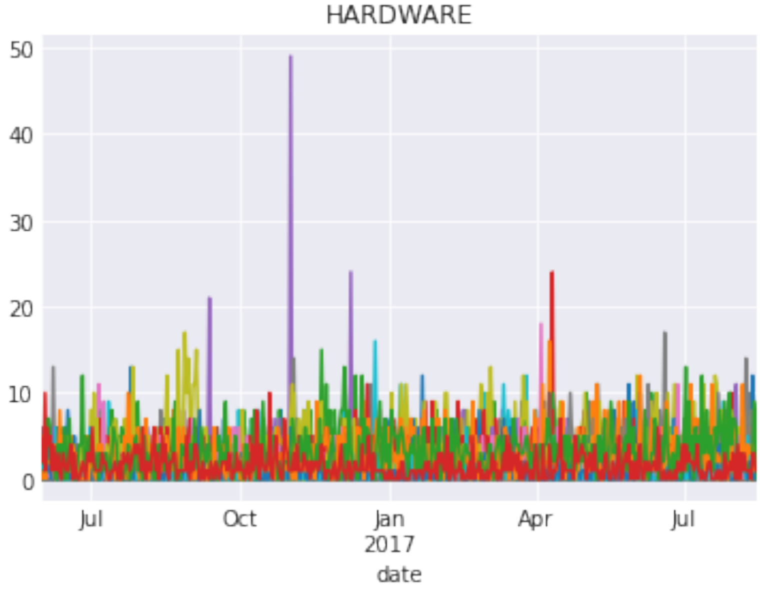


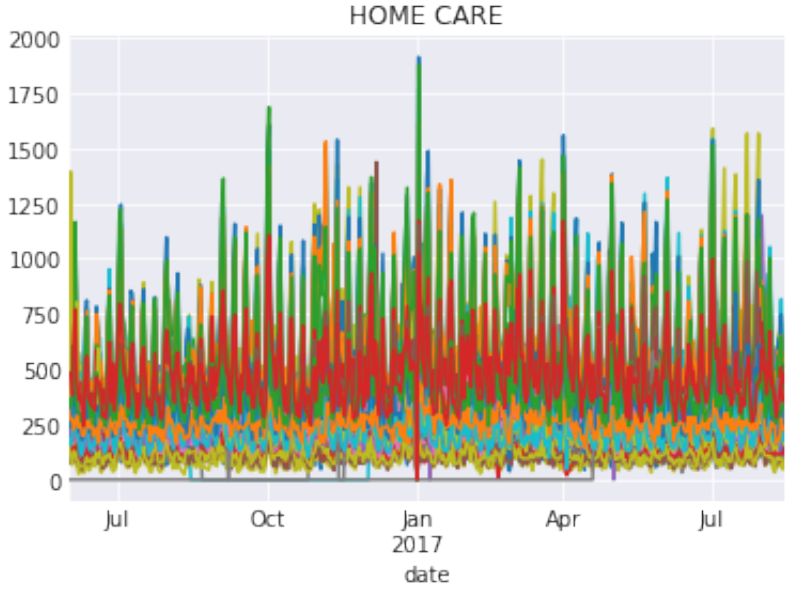
LAWN AND GARDEN

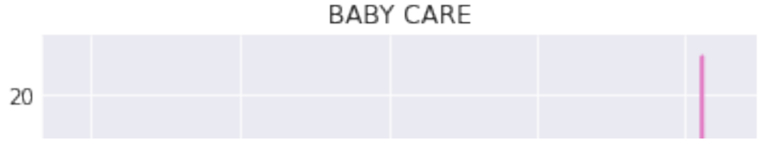


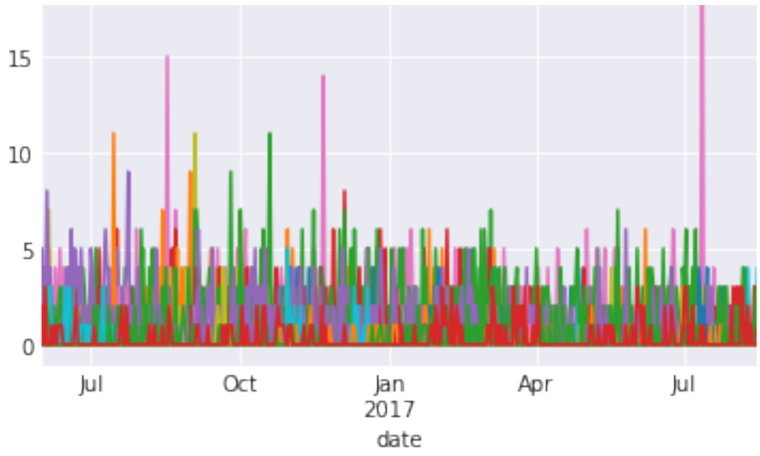
GROCERY I

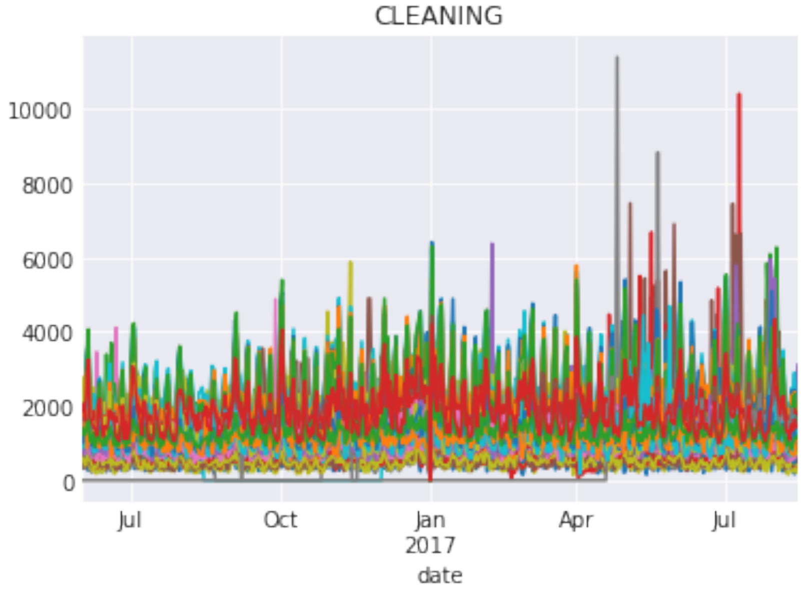

GROCERY I
40000
35000
30000
25000
20000
15000
10000
5000
0
Jul
Oct
Jan
2017
Apr
Jul
date

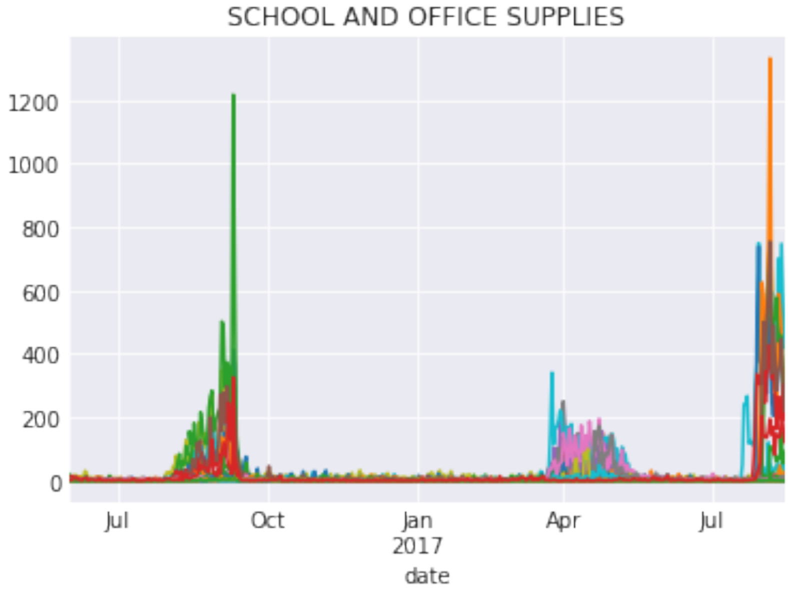

HARDWARE
50
40
30
20
10
0
Jul
Oct
Jan
2017
Apr
Jul
date

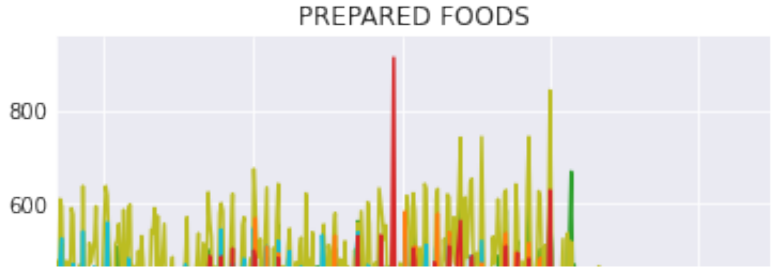

HOME CARE
2000
1750
1500
1250
1000
750
500
250
0
Jul
Oct
Jan
2017
Apr
Jul
date

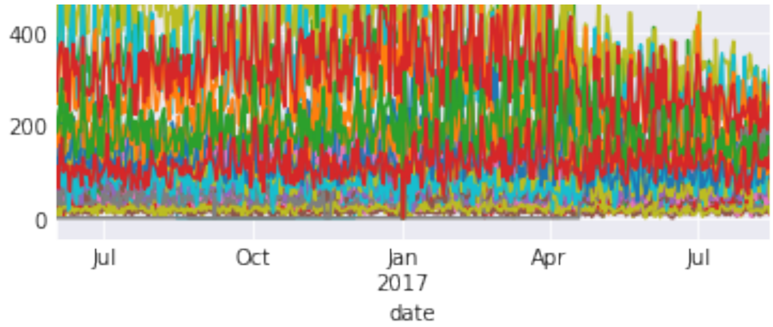

BABY CARE
20


15
10
5
0
Jul
Oct
Jan
2017
Apr
Jul
date


CLEANING
10000
8000
6000
4000
2000
0
Jul
Oct
Jan
2017
Apr
Jul
date


SCHOOL AND OFFICE SUPPLIES
1200
1000
800
600
400
200
0
Jul
Oct
Jan
2017
Apr
Jul
date


PREPARED FOODS
800
600



```
sdate = '2017-04-30' # Start and end of training date
edate = '2017-08-15'
```

```
school_season = [] # Feature for school fluctuations
for i, r in calendar.iterrows() :
    if i.month in [4, 5, 8, 9] :
        school_season.append(1)
    else :
        school_season.append(0)
calendar['school_season'] = school_season
calendar
```

	avg_oil	oil_lags1	oil_lags2	oil_lags3	wd	dofw_1	dofw_2	dofw_3
2013-01-13	93.284286	93.284286	93.284286	93.218571	0	0	0	0
2013-01-14	93.470000	93.284286	93.284286	93.284286	1	0	0	0
2013-01-15	93.490000	93.470000	93.284286	93.284286	1	1	0	0
2013-01-16	93.644286	93.490000	93.470000	93.284286	1	0	1	0
2013-01-17	93.970000	93.644286	93.490000	93.470000	1	0	0	1
...	...	...	...	...	...	...	...	...
2017-08-27	47.720000	47.720000	47.720000	47.598571	0	0	0	0
2017-08-28	47.624286	47.720000	47.720000	47.720000	1	0	0	0
2017-08-29	47.320000	47.624286	47.720000	47.720000	1	1	0	0
2017-08-30	47.115714	47.320000	47.624286	47.720000	1	0	1	0
2017-08-31	47.060000	47.115714	47.320000	47.624286	1	0	0	1

1692 rows × 18 columns

```
y = train.unstack(['store_nbr', 'family']).loc[sdate:edate]
fourier = CalendarFourier(freq = 'W', order = 4)
```

```
dp = DeterministicProcess(index = y.index,
                          order = 1,
                          seasonal = False,
                          constant = False,
                          additional_terms = [fourier],
                          drop = True)
x = dp.in_sample()
x = x.join(calendar)
x
```

	trend	sin(1,freq=W-SUN)	cos(1,freq=W-SUN)	sin(2,freq=W-SUN)	cos(2,freq=W-SUN)	sin
date						
2017-04-30	1.0	-0.781831	0.623490	-0.974928	-0.222521	
2017-05-01	2.0	0.000000	1.000000	0.000000	1.000000	
2017-05-02	3.0	0.781831	0.623490	0.974928	-0.222521	
2017-05-03	4.0	0.974928	-0.222521	-0.433884	-0.900969	
2017-05-04	5.0	0.433884	-0.900969	-0.781831	0.623490	
...	...	...	...	...	...	
2017-08-11	104.0	-0.433884	-0.900969	0.781831	0.623490	
2017-08-12	105.0	-0.974928	-0.222521	0.433884	-0.900969	
2017-08-13	106.0	-0.781831	0.623490	-0.974928	-0.222521	
2017-08-14	107.0	0.000000	1.000000	0.000000	1.000000	
2017-08-15	108.0	0.781831	0.623490	0.974928	-0.222521	

108 rows × 25 columns

```
from statsmodels.tsa.deterministic import DeterministicProcess, CalendarFourier
from statsmodels.graphics.tsaplots import plot_pacf
```

```
!pip install statsmodels
```

```
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.7/dis
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist
Requirement already satisfied: scipy>=0.18 in /usr/local/lib/python3.7/dist
Requirement already satisfied: patsy>=0.4.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/pyt
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-package
```

```
!pip install --upgrade --no-deps statsmodels
```

```
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist
Collecting statsmodels
  Downloading statsmodels-0.13.2-cp37-cp37m-manylinux_2_17_x86_64.manylinux
     |████████████████████████████████| 9.8 MB 4.5 MB/s
Installing collected packages: statsmodels
  Attempting uninstall: statsmodels
    Found existing installation: statsmodels 0.10.2
    Uninstalling statsmodels-0.10.2:
      Successfully uninstalled statsmodels-0.10.2
Successfully installed statsmodels-0.13.2
WARNING: The following packages were previously imported in this runtime:
  [statsmodels]
You must restart the runtime in order to use newly installed versions.
```

RESTART RUNTIME

```
!pip3 uninstall statsmodels
```

```
Found existing installation: statsmodels 0.13.2
Uninstalling statsmodels-0.13.2:
  Would remove:
    /usr/local/lib/python3.7/dist-packages/statsmodels-0.13.2.dist-info/*
    /usr/local/lib/python3.7/dist-packages/statsmodels/*
Proceed (y/n)? y
  Successfully uninstalled statsmodels-0.13.2
```

```
!pip3 install numpy scipy patsy pandas
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: patsy in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/pyt
```

```
!pip3 install statsmodels
```

```
Collecting statsmodels
  Using cached statsmodels-0.13.2-cp37-cp37m-manylinux_2_17_x86_64.manylinu
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.7/dis
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.7/
Requirement already satisfied: pandas>=0.25 in /usr/local/lib/python3.7/dis
Requirement already satisfied: scipy>=1.3 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/p
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-package
Installing collected packages: statsmodels
Successfully installed statsmodels-0.13.2
WARNING: The following packages were previously imported in this runtime:
  [statsmodels]
You must restart the runtime in order to use newly installed versions.
```

RESTART RUNTIME

```
print(y.isna().sum().sum())
display(y)
```

```
0
```

	sales						
store_nbr	1						
family	AUTOMOTIVE	BABY CARE	BEAUTY	BEVERAGES	BOOKS	BREAD/BAKERY	CELEBRAT
date							
2017-04-30	3.0	0.0	0.0	995.0	1.0	139.50700	
2017-05-01	0.0	0.0	2.0	825.0	0.0	116.33900	
2017-05-02	2.0	0.0	2.0	3179.0	0.0	447.23800	
2017-05-03	5.0	0.0	6.0	2479.0	1.0	434.02900	
2017-05-04	3.0	0.0	1.0	2454.0	0.0	438.21400	
...	...	...	...	...	...	...	
2017-08-11	1.0	0.0	1.0	1006.0	0.0	145.60700	
2017-08-12	6.0	0.0	3.0	1659.0	0.0	243.22000	
2017-08-13	1.0	0.0	1.0	803.0	0.0	136.67900	
2017-08-14	1.0	0.0	6.0	2201.0	0.0	346.03800	
2017-08-15	4.0	0.0	4.0	1942.0	0.0	329.54102	

108 rows × 1782 columns

```
xtest = dp.out_of_sample(steps = 16) # 16 because we are predicting next 16 days
xtest = xtest.join(calendar)
xtest
```

	trend	sin(1,freq=W-SUN)	cos(1,freq=W-SUN)	sin(2,freq=W-SUN)	cos(2,freq=W-SUN)	sin
2017-08-16	109.0	0.974928	-0.222521	-0.433884	-0.900969	
2017-						

08-17	110.0	0.433884	-0.900969	-0.781831	0.623490
2017-08-18	111.0	-0.433884	-0.900969	0.781831	0.623490
2017-08-19	112.0	-0.974928	-0.222521	0.433884	-0.900969
2017-08-20	113.0	-0.781831	0.623490	-0.974928	-0.222521
2017-08-21	114.0	0.000000	1.000000	0.000000	1.000000
2017-08-22	115.0	0.781831	0.623490	0.974928	-0.222521
2017-08-23	116.0	0.974928	-0.222521	-0.433884	-0.900969
2017-08-24	117.0	0.433884	-0.900969	-0.781831	0.623490
2017-08-25	118.0	-0.433884	-0.900969	0.781831	0.623490
2017-08-26	119.0	-0.974928	-0.222521	0.433884	-0.900969
2017-08-27	120.0	-0.781831	0.623490	-0.974928	-0.222521
2017-08-28	121.0	0.000000	1.000000	0.000000	1.000000
2017-08-29	122.0	0.781831	0.623490	0.974928	-0.222521
2017-08-30	123.0	0.974928	-0.222521	-0.433884	-0.900969
2017-08-31	124.0	0.433884	-0.900969	-0.781831	0.623490

16 rows × 25 columns

```
def make_lags(x, lags = 1) : #Fungsi untuk membuat fitur lags
    lags = lags
    x_ = x.copy()
    for i in range(lags) :
        lag = x_.shift(i + 1)
        x = pd.concat([x, lag], axis = 1)
    return x


from joblib import Parallel, delayed
from tqdm.auto import tqdm
from sklearn.metrics import mean_squared_log_error as msle
from sklearn.model_selection import TimeSeriesSplit
from sklearn.svm import SVR
from sklearn.multioutput import MultiOutputRegressor

lnr = LinearRegression(fit_intercept = True, n_jobs = -1, normalize = True)
lnr.fit(x, y)

yfit_lnr = pd.DataFrame(lnr.predict(x), index = x.index, columns = y.columns).cl
ypred_lnr = pd.DataFrame(lnr.predict(xtest), index = xtest.index, columns = y.co

svr = MultiOutputRegressor(SVR(C = 0.2, kernel = 'rbf'), n_jobs = -1)
svr.fit(x, y)

yfit_svr = pd.DataFrame(svr.predict(x), index = x.index, columns = y.columns).cl
ypred_svr = pd.DataFrame(svr.predict(xtest), index = xtest.index, columns = y.co

yfit_mean = pd.DataFrame(np.mean([yfit_svr.values, yfit_lnr.values], axis = 0),
ypred_mean = pd.DataFrame(np.mean([ypred_lnr.values, ypred_svr.values], axis = 0

y_ = y.stack(['store_nbr', 'family'])
y_['lnr'] = yfit_lnr.stack(['store_nbr', 'family'])['sales']
y_['svr'] = yfit_svr.stack(['store_nbr', 'family'])['sales']
y_['mean'] = yfit_mean.stack(['store_nbr', 'family'])['sales']

print('='*70, 'Linear Regression', '='*70)
print(y_.groupby('family').apply(lambda r : np.sqrt(msle(r['sales'], r['lnr'])))
print('LNR RMSLE :', np.sqrt(msle(y, yfit_lnr)))
print('='*70, 'SVR', '='*70)
print(y_.groupby('family').apply(lambda r : np.sqrt(msle(r['sales'], r['svr'])))
print('SVR RMSLE :', np.sqrt(msle(y, yfit_svr)))
print('='*70, 'Mean', '='*70)
print(y_.groupby('family').apply(lambda r : np.sqrt(msle(r['sales'], r['mean']))
print('Mean RMSLE :', np.sqrt(msle(y, yfit_mean)))
```

```
    ====================================================================== Li
    family
    AUTOMOTIVE                    0.470853
```

```
AUTOMOTIVE                    0.470953
BABY CARE                     0.253725
BEAUTY                        0.470013
BEVERAGES                     0.165701
BOOKS                         0.124115
BREAD/BAKERY                  0.133141
CELEBRATION                   0.512125
CLEANING                      0.325909
DAIRY                         0.122254
DELI                          0.152218
EGGS                          0.281778
FROZEN FOODS                  0.243512
GROCERY I                     0.147955
GROCERY II                    0.548617
HARDWARE                      0.491726
HOME AND KITCHEN I            0.457245
HOME AND KITCHEN II           0.449896
HOME APPLIANCES               0.363741
HOME CARE                     0.192405
LADIESWEAR                    0.447912
LAWN AND GARDEN               0.398217
LINGERIE                      0.587564
LIQUOR,WINE,BEER              0.538526
MAGAZINES                     0.461284
MEATS                         0.165268
PERSONAL CARE                 0.209257
PET SUPPLIES                  0.423930
PLAYERS AND ELECTRONICS       0.423179
POULTRY                       0.171110
PREPARED FOODS                0.245395
PRODUCE                       0.107894
SCHOOL AND OFFICE SUPPLIES    0.981280
SEAFOOD                       0.459579
dtype: float64
LNR RMSLE : 0.3958433835879891
====================================================================== SV
family
AUTOMOTIVE                    0.537979
BABY CARE                     0.276018
BEAUTY                        0.548399
BEVERAGES                     0.287354
BOOKS                         0.148110
BREAD/BAKERY                  0.247432
CELEBRATION                   0.583759
CLEANING                      0.342808
DAIRY                         0.256998
DELI                          0.259028
EGGS                          0.404389
FROZEN FOODS                  0.397948
GROCERY I                     0.249870
GROCERY II                    0.594346
HARDWARE                      0.537422
HOME AND KITCHEN I            0.512461
HOME AND KITCHEN II           0.468299
HOME APPLIANCES               0.407741
```

```
    HOME APPLIANCES              0.407741
    HOME CARE                    0.306196
    LADIESWEAR                   0.542852
```

```
from sklearn.metrics import mean_absolute_error as mae

print('='*70, 'Linear Regression', '='*70)
print(y_.groupby('family').apply(lambda r : mae(r['sales'], r['lnr'])))
print('LNR RMSLE :', mae(y, yfit_lnr))
print('='*70, 'SVR', '='*70)
print(y_.groupby('family').apply(lambda r : mae(r['sales'], r['svr'])))
print('SVR RMSLE :', mae(y, yfit_svr))
print('='*70, 'Mean', '='*70)
print(y_.groupby('family').apply(lambda r : mae(r['sales'], r['mean'])))
print('Mean RMSLE :', mae(y, yfit_mean))
```

```
    ====================================================================== Li
    family
    AUTOMOTIVE                     2.462666
    BABY CARE                      0.218935
    BEAUTY                         1.993658
    BEVERAGES                    393.223348
    BOOKS                          0.062452
    BREAD/BAKERY                  48.987934
    CELEBRATION                    4.508831
    CLEANING                     234.958095
    DAIRY                         77.517478
    DELI                          35.185488
    EGGS                          29.019481
    FROZEN FOODS                  19.628004
    GROCERY I                    488.737700
    GROCERY II                     8.638179
    HARDWARE                       0.944746
    HOME AND KITCHEN I            10.298440
    HOME AND KITCHEN II            9.755713
    HOME APPLIANCES                0.467302
    HOME CARE                     39.605771
    LADIESWEAR                     3.409966
    LAWN AND GARDEN                4.229637
    LINGERIE                       2.976668
    LIQUOR,WINE,BEER              22.118200
    MAGAZINES                      2.093751
    MEATS                         40.686022
    PERSONAL CARE                 47.811009
    PET SUPPLIES                   2.440073
    PLAYERS AND ELECTRONICS        2.952325
    POULTRY                       42.445535
    PREPARED FOODS                12.464928
    PRODUCE                      164.030993
    SCHOOL AND OFFICE SUPPLIES     7.653268
    SEAFOOD                        4.159589
    dtype: float64
```

```
LNR RMSLE : 53.50564200304283
===================================================================== SV
family
AUTOMOTIVE                      3.038670
BABY CARE                       0.213618
BEAUTY                          2.589378
BEVERAGES                     807.456838
BOOKS                           0.078344
BREAD/BAKERY                  101.144459
CELEBRATION                     5.584227
CLEANING                      319.913149
DAIRY                         180.690911
DELI                           68.286139
EGGS                           56.841999
FROZEN FOODS                   38.263943
GROCERY I                     920.832156
GROCERY II                     10.857102
HARDWARE                        1.031637
HOME AND KITCHEN I             12.069000
HOME AND KITCHEN II            10.981285
HOME APPLIANCES                 0.495080
HOME CARE                      72.677757
LADIESWEAR                      5.104274
```

```
true_low = [2]
pred_low = [4]

print('RMSLE for low value :', np.sqrt(msle(true_low, pred_low)))
print('MAE for low value :', mae(true_low, pred_low))

true_high = [255]
pred_high = [269]

print('RMSLE for high value :', np.sqrt(msle(true_high, pred_high)))
print('MAE for high value :', mae(true_high, pred_high))
```

```
RMSLE for low value : 0.5108256237659907
MAE for low value : 2.0
RMSLE for high value : 0.053244514518812736
MAE for high value : 14.0
```

```
display(x, xtest)
```

	trend	sin(1,freq=W-SUN)	cos(1,freq=W-SUN)	sin(2,freq=W-SUN)	cos(2,freq=W-SUN)	sin
date						
2017-04-30	1.0	-0.781831	0.623490	-0.974928	-0.222521	

2017-05-01	2.0	0.000000	1.000000	0.000000	1.000000
2017-05-02	3.0	0.781831	0.623490	0.974928	-0.222521
2017-05-03	4.0	0.974928	-0.222521	-0.433884	-0.900969
2017-05-04	5.0	0.433884	-0.900969	-0.781831	0.623490
...	...	...	...	...	...
2017-08-11	104.0	-0.433884	-0.900969	0.781831	0.623490
2017-08-12	105.0	-0.974928	-0.222521	0.433884	-0.900969
2017-08-13	106.0	-0.781831	0.623490	-0.974928	-0.222521
2017-08-14	107.0	0.000000	1.000000	0.000000	1.000000
2017-08-15	108.0	0.781831	0.623490	0.974928	-0.222521

108 rows × 25 columns

	trend	sin(1,freq=W-SUN)	cos(1,freq=W-SUN)	sin(2,freq=W-SUN)	cos(2,freq=W-SUN)	sin
2017-08-16	109.0	0.974928	-0.222521	-0.433884	-0.900969	
2017-08-17	110.0	0.433884	-0.900969	-0.781831	0.623490	
2017-08-18	111.0	-0.433884	-0.900969	0.781831	0.623490	
2017-08-19	112.0	-0.974928	-0.222521	0.433884	-0.900969	
2017-08-20	113.0	-0.781831	0.623490	-0.974928	-0.222521	
2017-08-21	114.0	0.000000	1.000000	0.000000	1.000000	
2017-08-22	115.0	0.781831	0.623490	0.974928	-0.222521	

2017-08-23	116.0	0.974928	-0.222521	-0.433884	-0.900969
2017-08-24	117.0	0.433884	-0.900969	-0.781831	0.623490
2017-08-25	118.0	-0.433884	-0.900969	0.781831	0.623490
2017-08-26	119.0	-0.974928	-0.222521	0.433884	-0.900969
2017-08-27	120.0	-0.781831	0.623490	-0.974928	-0.222521
2017-08-28	121.0	0.000000	1.000000	0.000000	1.000000
2017-08-29	122.0	0.781831	0.623490	0.974928	-0.222521
2017-08-30	123.0	0.974928	-0.222521	-0.433884	-0.900969
2017-08-31	124.0	0.433884	-0.900969	-0.781831	0.623490

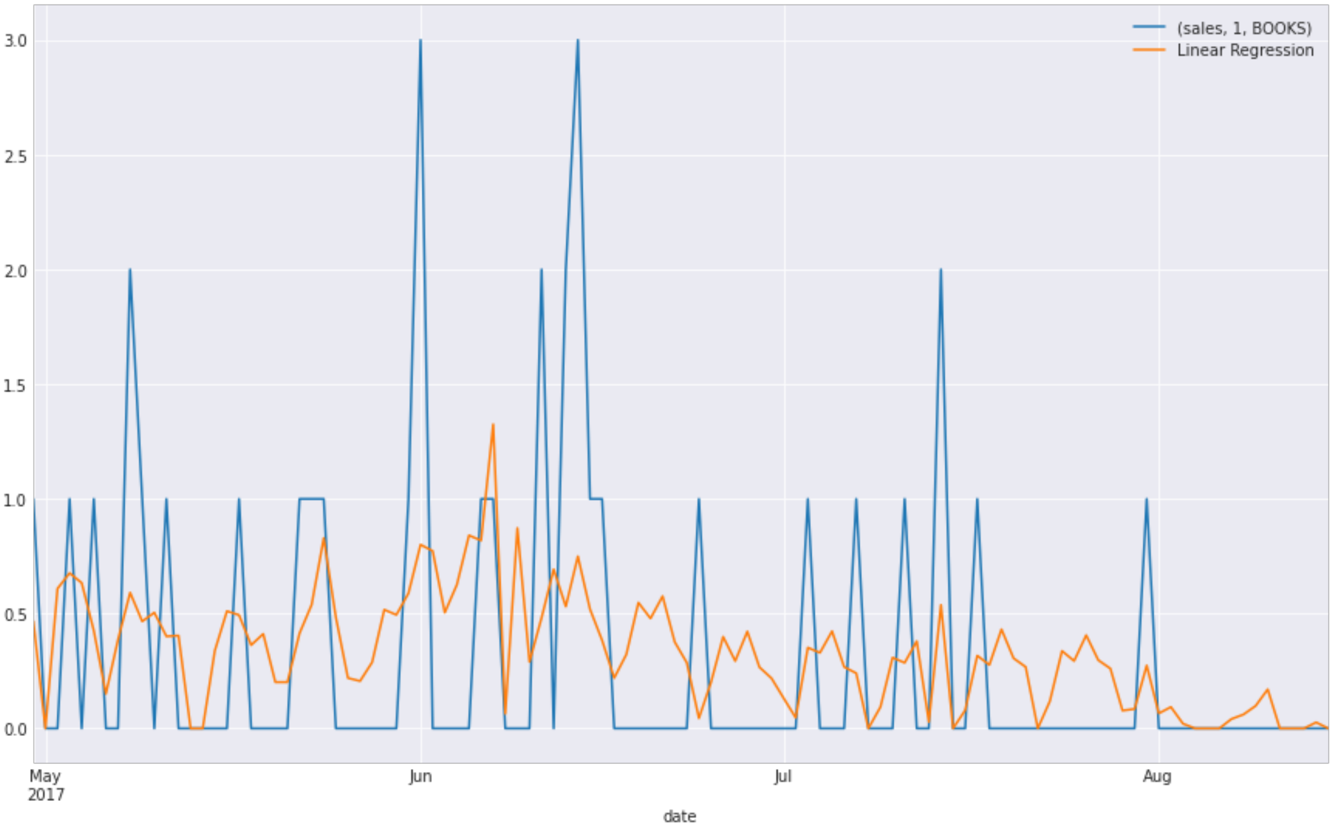
16 rows × 25 columns

```
ypred_svr
```

	sales						
store_nbr	1						
family	**AUTOMOTIVE**	**BABY CARE**	**BEAUTY**	**BEVERAGES**	**BOOKS**	**BREAD/BAKERY**	**CELE**
2017-08-16	4.168328	0.0	3.106363	2325.245879	0.100142	373.672883	1
2017-08-17	4.171205	0.0	3.108065	2325.234816	0.100236	373.657439	1
2017-08-18	4.174432	0.0	3.108846	2325.228624	0.100270	373.645969	1
2017-08-19	4.176663	0.0	3.110402	2325.218258	0.100128	373.632541	1
2017-08-20	4.176519	0.0	3.109982	2325.207597	0.100150	373.618532	1
2017-08-21	4.179100	0.0	3.112900	2325.201505	0.100278	373.609427	1
2017-08-22	4.180521	0.0	3.113378	2325.193626	0.100241	373.597693	1
2017-08-23	4.181644	0.0	3.113919	2325.185312	0.100157	373.588922	1
2017-08-24	4.182878	0.0	3.114176	2325.175266	0.100292	373.576802	1
2017-08-25	4.184311	0.0	3.113415	2325.169632	0.100337	373.568393	1
2017-08-26	4.184841	0.0	3.113694	2325.160160	0.100217	373.558223	1
2017-08-27	4.183350	0.0	3.112574	2325.150750	0.100236	373.547657	1
2017-08-28	4.184480	0.0	3.114721	2325.145678	0.100366	373.541627	1
2017-08-29	4.184687	0.0	3.114931	2325.139185	0.100322	373.533325	1
2017-08-30	4.184661	0.0	3.115486	2325.132428	0.100213	373.527987	1
2017-08-31	4.184795	0.0	3.115784	2325.124179	0.100352	373.519514	1

16 rows × 1782 columns

```
fam = 'BOOKS'
nbr = '1'
plt.rcParams['figure.figsize'] = (15, 9)
plt.figure()
y.loc(axis = 1)['sales', nbr, fam].plot()
yfit_lnr.loc(axis = 1)['sales', nbr, fam].plot(label = 'Linear Regression')
#yfit_svr.loc(axis = 1)['sales', nbr, fam].plot(label = 'SVR')
#yfit_mean.loc(axis = 1)['sales', nbr, fam].plot(label = 'Mean')
#y.mean(axis = 1).plot()
#yfit_lnr.median(axis = 1).plot(label = 'Linear Regression')
#yfit_svr.median(axis = 1).plot(label = 'SVR')
#yfit_mean.mean(axis = 1).plot(label = 'Mean')
plt.legend()
plt.show()
```



```
ymean = yfit_lnr.append(ypred_lnr)
school = ymean.loc(axis = 1)['sales', :, 'SCHOOL AND OFFICE SUPPLIES']
ymean = ymean.join(school.shift(1), rsuffix = 'lag1') # I'm also adding school l
x = x.loc['2017-05-01':]
```

```
ymean.loc['2017-08-16':]
```

	sales						
store_nbr	**1**						
family	**AUTOMOTIVE**	**BABY CARE**	**BEAUTY**	**BEVERAGES**	**BOOKS**	**BREAD/BAKERY**	**CELE**
2017-08-16	3.555439	0.0	5.867603	2177.101514	0.051118	375.696460	1
2017-08-17	3.157970	0.0	5.351209	1820.884426	0.000000	325.401119	1
2017-08-18	7.196938	0.0	4.625604	2226.318076	0.000000	360.358263	2
2017-08-19	5.081869	0.0	5.161570	2110.332182	0.000000	324.401823	
2017-08-20	1.890016	0.0	3.552465	791.784055	0.000000	122.069647	
2017-08-21	3.861995	0.0	5.501442	2086.850066	0.000000	350.405306	1
2017-08-22	3.870789	0.0	5.098935	2142.494108	0.000000	331.975768	1
2017-08-23	3.805114	0.0	5.927498	2152.189495	0.000000	374.242688	1
2017-08-24	3.201296	0.0	5.033964	1831.757276	0.000000	320.689802	1
2017-08-25	7.808126	0.0	4.222571	2217.743153	0.000000	353.374110	2
2017-08-26	5.231181	0.0	4.950071	2107.504993	0.000000	317.896659	
2017-08-27	2.180226	0.0	3.556282	766.607195	0.000000	116.138004	
2017-08-28	3.932675	0.0	5.609625	2067.059825	0.000000	346.750075	1
2017-08-29	3.741585	0.0	5.253178	2125.002831	0.000000	327.227568	1
2017-08-30	3.644864	0.0	6.388955	2108.349389	0.000000	369.009468	1
2017-08-31	3.123013	0.0	5.688508	1774.781559	0.000000	319.225734	1

16 rows × 1836 columns

```
x = x.join(ymean) # Concating linear result
xtest = xtest.join(ymean)
display(x, xtest)
```

	trend	sin(1,freq=W-SUN)	cos(1,freq=W-SUN)	sin(2,freq=W-SUN)	cos(2,freq=W-SUN)	sin
date						
2017-05-01	2.0	0.000000	1.000000	0.000000	1.000000	
2017-05-02	3.0	0.781831	0.623490	0.974928	-0.222521	
2017-05-03	4.0	0.974928	-0.222521	-0.433884	-0.900969	
2017-05-04	5.0	0.433884	-0.900969	-0.781831	0.623490	
2017-05-05	6.0	-0.433884	-0.900969	0.781831	0.623490	
...	...	...	...	...	...	
2017-08-11	104.0	-0.433884	-0.900969	0.781831	0.623490	
2017-08-12	105.0	-0.974928	-0.222521	0.433884	-0.900969	
2017-08-13	106.0	-0.781831	0.623490	-0.974928	-0.222521	
2017-08-14	107.0	0.000000	1.000000	0.000000	1.000000	
2017-08-15	108.0	0.781831	0.623490	0.974928	-0.222521	

107 rows × 1861 columns

	trend	sin(1,freq=W-SUN)	cos(1,freq=W-SUN)	sin(2,freq=W-SUN)	cos(2,freq=W-SUN)	sin
2017-08-16	109.0	0.974928	-0.222521	-0.433884	-0.900969	

```
2017-
08-17   110.0    0.433884   -0.900969   -0.781831    0.623490

2017-
08-18   111.0   -0.433884   -0.900969    0.781831    0.623490

2017-
08-19   112.0   -0.974928   -0.222521    0.433884   -0.900969

2017-
08-20   113.0   -0.781831    0.623490   -0.974928   -0.222521

2017-
08-21   114.0    0.000000    1.000000    0.000000    1.000000

2017-
08-22   115.0    0.781831    0.623490    0.974928   -0.222521

2017-
08-23   116.0    0.974928   -0.222521   -0.433884   -0.900969

2017-
08-24   117.0    0.433884   -0.900969   -0.781831    0.623490

2017-
08-25   118.0   -0.433884   -0.900969    0.781831    0.623490

2017-
08-26   119.0   -0.974928   -0.222521    0.433884   -0.900969

2017-
08-27   120.0   -0.781831    0.623490   -0.974928   -0.222521

2017-
08-28   121.0    0.000000    1.000000    0.000000    1.000000

2017-
08-29   122.0    0.781831    0.623490    0.974928   -0.222521

2017-
08-30   123.0    0.974928   -0.222521   -0.433884   -0.900969

2017-
08-31   124.0    0.433884   -0.900969   -0.781831    0.623490

16 rows × 1861 columns
```

```
y = y.loc['2017-05-01':]
y
```

	sales						
store_nbr	**1**						
family	**AUTOMOTIVE**	**BABY CARE**	**BEAUTY**	**BEVERAGES**	**BOOKS**	**BREAD/BAKERY**	**CELEBRAT**
date							
2017-05-01	0.0	0.0	2.0	825.0	0.0	116.33900	
2017-05-02	2.0	0.0	2.0	3179.0	0.0	447.23800	
2017-05-03	5.0	0.0	6.0	2479.0	1.0	434.02900	
2017-05-04	3.0	0.0	1.0	2454.0	0.0	438.21400	
2017-05-05	12.0	0.0	0.0	2243.0	1.0	398.96500	
...	...	...	...	...	...	...	
2017-08-11	1.0	0.0	1.0	1006.0	0.0	145.60700	
2017-08-12	6.0	0.0	3.0	1659.0	0.0	243.22000	
2017-08-13	1.0	0.0	1.0	803.0	0.0	136.67900	
2017-08-14	1.0	0.0	6.0	2201.0	0.0	346.03800	
2017-08-15	4.0	0.0	4.0	1942.0	0.0	329.54102	

107 rows × 1782 columns

```
print(y.isna().sum().sum())
```

```
0
```

```
display(x, xtest)
```

	trend	sin(1,freq=W-SUN)	cos(1,freq=W-SUN)	sin(2,freq=W-SUN)	cos(2,freq=W-SUN)	sin

date					
2017-05-01	2.0	0.000000	1.000000	0.000000	1.000000
2017-05-02	3.0	0.781831	0.623490	0.974928	-0.222521
2017-05-03	4.0	0.974928	-0.222521	-0.433884	-0.900969
2017-05-04	5.0	0.433884	-0.900969	-0.781831	0.623490
2017-05-05	6.0	-0.433884	-0.900969	0.781831	0.623490
...	...	...	...	...	...
2017-08-11	104.0	-0.433884	-0.900969	0.781831	0.623490
2017-08-12	105.0	-0.974928	-0.222521	0.433884	-0.900969
2017-08-13	106.0	-0.781831	0.623490	-0.974928	-0.222521
2017-08-14	107.0	0.000000	1.000000	0.000000	1.000000
2017-08-15	108.0	0.781831	0.623490	0.974928	-0.222521

107 rows × 1861 columns

	trend	sin(1,freq=W-SUN)	cos(1,freq=W-SUN)	sin(2,freq=W-SUN)	cos(2,freq=W-SUN)	sin
2017-08-16	109.0	0.974928	-0.222521	-0.433884	-0.900969	
2017-08-17	110.0	0.433884	-0.900969	-0.781831	0.623490	
2017-08-18	111.0	-0.433884	-0.900969	0.781831	0.623490	
2017-08-19	112.0	-0.974928	-0.222521	0.433884	-0.900969	
2017-	113.0	-0.781831	0.623490	-0.974928	-0.222521	

08-20					
2017-08-21	114.0	0.000000	1.000000	0.000000	1.000000
2017-08-22	115.0	0.781831	0.623490	0.974928	-0.222521
2017-08-23	116.0	0.974928	-0.222521	-0.433884	-0.900969
2017-08-24	117.0	0.433884	-0.900969	-0.781831	0.623490
2017-08-25	118.0	-0.433884	-0.900969	0.781831	0.623490
2017-08-26	119.0	-0.974928	-0.222521	0.433884	-0.900969
2017-08-27	120.0	-0.781831	0.623490	-0.974928	-0.222521
2017-08-28	121.0	0.000000	1.000000	0.000000	1.000000
2017-08-29	122.0	0.781831	0.623490	0.974928	-0.222521
2017-08-30	123.0	0.974928	-0.222521	-0.433884	-0.900969
2017-08-31	124.0	0.433884	-0.900969	-0.781831	0.623490

16 rows × 1861 columns

```
from joblib import Parallel, delayed
import warnings

# Import necessary library
from sklearn.linear_model import Ridge, LinearRegression, ElasticNet
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import VotingRegressor

# SEED for reproducible result
SEED = 5

class CustomRegressor():
    def __init__(self, n_jobs=-1, verbose=0):

        self.n_jobs = n_jobs
        self.verbose = verbose

        self.estimators_ = None

    def _estimator_(self, X, y):

        warnings.simplefilter(action='ignore', category=FutureWarning)

        if y.name[2] == 'SCHOOL AND OFFICE SUPPLIES': # Because SCHOOL AND OFFIC
            r1 = ExtraTreesRegressor(n_estimators = 225, n_jobs=-1, random_state
            r2 = RandomForestRegressor(n_estimators = 225, n_jobs=-1, random_sta
            b1 = BaggingRegressor(base_estimator=r1,
                                  n_estimators=10,
                                  n_jobs=-1,
                                  random_state=SEED)
            b2 = BaggingRegressor(base_estimator=r2,
                                  n_estimators=10,
                                  n_jobs=-1,
                                  random_state=SEED)
            model = VotingRegressor([('et', b1), ('rf', b2)]) # Averaging the re
        else:
            ridge = Ridge(fit_intercept=True, solver='auto', alpha=0.75, normali
            svr = SVR(C = 0.2, kernel = 'rbf')

            model = VotingRegressor([('ridge', ridge), ('svr', svr)]) # Averagin
        model.fit(X, y)

        return model

    def fit(self, X, y):
        from tqdm.auto import tqdm


        if self.verbose == 0 :
            self.estimators_ = Parallel(n_jobs=self.n_jobs,
                                  verbose=0,
                                  )(delayed(self._estimator_)(X, y.iloc[:, i]) f
        else :
            print('Fit Progress')
            self.estimators_ = Parallel(n_jobs=self.n_jobs,
                                  verbose=0,
                                  )(delayed(self._estimator_)(X, y.iloc[:, i]) f
        return
    def predict(self, X):
        from tqdm.auto import tqdm
        if self.verbose == 0 :
            y_pred = Parallel(n_jobs=self.n_jobs,
                              verbose=0)(delayed(e.predict)(X) for e in self.est
        else :
            print('Predict Progress')
            y_pred = Parallel(n_jobs=self.n_jobs,
                              verbose=0)(delayed(e.predict)(X) for e in tqdm(sel

        return np.stack(y_pred, axis=1)
```

```
%%time

model = CustomRegressor(n_jobs=-1, verbose=1)
model.fit(x, y)
y_pred = pd.DataFrame(model.predict(x), index=x.index, columns=y.columns)
```

```
Fit Progress
100%                                        1782/1782 [50:08<00:00, 6.46it/s]
Predict Progress
100%                                        1782/1782 [03:35<00:00, 13.32it/s]
CPU times: user 3min 2s, sys: 15.6 s, total: 3min 18s
Wall time: 55min 40s
```

```
display(y_pred)
print(y_pred.isna().sum().sum())
```

	sales						
store_nbr	1						
family	AUTOMOTIVE	BABY CARE	BEAUTY	BEVERAGES	BOOKS	BREAD/BAKERY	CEL
date							
2017-05-01	1.698524	0.0	2.173227	1579.802698	0.054388	245.079779	
2017-05-02	3.004028	0.0	2.616627	2680.868116	0.177364	409.074859	
2017-05-03	3.521669	0.0	3.814092	2386.076368	0.404251	407.459876	
2017-05-04	3.257565	0.0	3.335985	2214.912187	0.331597	382.893081	
2017-05-05	5.520494	0.0	3.073227	2386.152081	0.267680	397.429907	
...	...	...	...	...	...	...	
2017-08-11	2.418747	0.0	2.013310	1636.205361	-0.022421	260.527651	
2017-08-12	4.411518	0.0	3.135655	2173.003408	-0.047607	331.894410	
2017-08-13	3.088893	0.0	2.752154	1598.121771	-0.034620	251.354349	
2017-08-14	4.142985	0.0	4.142910	2239.278581	0.101574	364.541466	
2017-08-15	4.359934	0.0	3.894137	2237.649700	0.063734	354.972784	

107 rows × 1782 columns

```
0
```

```
from sklearn.metrics import mean_squared_log_error
y_pred = y_pred.stack(['store_nbr', 'family']).clip(0.)
y_ = y.stack(['store_nbr', 'family']).clip(0.)

y_['pred'] = y_pred.values
print(y_.groupby('family').apply(lambda r : np.sqrt(np.sqrt(mean_squared_log_err
# Looking at error
print('RMSLE : ', np.sqrt(np.sqrt(msle(y_['sales'], y_['pred']))))
```

```
family
AUTOMOTIVE                    0.682456
BABY CARE                     0.485725
BEAUTY                        0.678844
BEVERAGES                     0.429658
BOOKS                         0.345595
BREAD/BAKERY                  0.398289
CELEBRATION                   0.706130
CLEANING                      0.508470
DAIRY                         0.393521
DELI                          0.418994
EGGS                          0.532926
FROZEN FOODS                  0.517150
GROCERY I                     0.407054
GROCERY II                    0.707978
HARDWARE                      0.685809
HOME AND KITCHEN I            0.657512
HOME AND KITCHEN II           0.630601
HOME APPLIANCES               0.589133
HOME CARE                     0.453047
LADIESWEAR                    0.656678
LAWN AND GARDEN               0.636974
LINGERIE                      0.750291
LIQUOR,WINE,BEER              0.688461
MAGAZINES                     0.673514
MEATS                         0.445274
PERSONAL CARE                 0.474278
PET SUPPLIES                  0.647409
PLAYERS AND ELECTRONICS       0.648474
POULTRY                       0.446971
PREPARED FOODS                0.502954
PRODUCE                       0.400223
SCHOOL AND OFFICE SUPPLIES    0.659191
SEAFOOD                       0.674311
dtype: float64
RMSLE :  0.5956418338066279
```

```
y_pred.isna().sum()
```

```
sales    0
dtype: int64
```

```
ypred = pd.DataFrame(model.predict(xtest), index = xtest.index, columns = y.colu
ypred
```

```
Predict Progress
100%                                                    1782/1782 [03:05<00:00, 17.47it/s]
```

	sales						
store_nbr	**1**						
family	**AUTOMOTIVE**	**BABY CARE**	**BEAUTY**	**BEVERAGES**	**BOOKS**	**BREAD/BAKERY**	**CELE**
2017-08-16	4.101210	0.0	4.555177	2262.699439	0.083691	376.819611	1
2017-08-17	3.999122	0.0	4.505269	2088.280422	0.000000	353.284558	1
2017-08-18	5.929922	0.0	3.980679	2288.157058	0.000000	369.008065	1
2017-08-19	4.965905	0.0	3.877032	2209.862598	0.000000	348.557214	1
2017-08-20	3.073618	0.0	3.006838	1565.133852	0.000000	250.150492	
2017-08-21	4.058973	0.0	4.425900	2213.437938	0.057167	363.289727	1
2017-08-22	4.371059	0.0	4.122630	2217.087207	0.005848	353.989573	1
2017-08-23	4.234922	0.0	4.576363	2254.981841	0.011189	377.320787	1
2017-08-24	3.971611	0.0	4.438793	2104.928285	0.000000	350.620445	1
2017-08-25	6.203956	0.0	3.812185	2290.446241	0.000000	366.933996	1
2017-08-26	4.867389	0.0	3.832786	2216.427732	0.000000	344.879051	1
2017-08-27	3.105635	0.0	3.019963	1550.103784	0.000000	246.607038	
2017-08-28	4.106690	0.0	4.521539	2202.474459	0.032032	361.053611	1
2017-08-29	4.366303	0.0	4.228360	2210.058813	0.004319	352.237739	1
2017-08-30	4.235968	0.0	4.766981	2225.605043	0.000000	374.495130	1
2017-08-31	4.115426	0.0	4.796192	2072.149925	0.000000	351.192205	1

16 rows × 1782 columns

```
ypred = ypred.stack(['store_nbr', 'family'])
ypred
```

```
                                                           sales
            store_nbr               family
2017-08-16      1                 AUTOMOTIVE            4.101210
                                  BABY CARE             0.000000
                                  BEAUTY                4.555177
                                  BEVERAGES          2262.699439
                                  BOOKS                 0.083691
   ...         ...                    ...                    ...
2017-08-31      9                 POULTRY             367.115985
                               PREPARED FOODS         109.489971
                                  PRODUCE            1336.562983
                       SCHOOL AND OFFICE SUPPLIES     136.896222
                                  SEAFOOD              14.133084

28512 rows × 1 columns
```

```
sub = pd.read_csv('sample_submission.csv')
sub['sales'] = ypred.values
sub.to_csv('submission.csv', index = False) # Submit
sub
```

	id	sales
0	3000888	4.101210
1	3000889	0.000000
2	3000890	4.555177
3	3000891	2262.699439
4	3000892	0.083691
...	...	...
28507	3029395	367.115985
28508	3029396	109.489971
28509	3029397	1336.562983
28510	3029398	136.896222
28511	3029399	14.133084

28512 rows × 2 columns

```
sub
```

	id	sales
0	3000888	4.101210
1	3000889	0.000000
2	3000890	4.555177
3	3000891	2262.699439
4	3000892	0.083691
...	...	...
28507	3029395	367.115985
28508	3029396	109.489971
28509	3029397	1336.562983
28510	3029398	136.896222
28511	3029399	14.133084

28512 rows × 2 columns

```
sub.to_csv('submission.csv', index = False)
```

✓ 0초 오후 5:49에 완료됨 ● ✕