LinkedIn GreyKite Forecast - Brooklyn Rolling Data

Here I will do the following:

1. Import necessary modules
2. Import the data
3. Prepare data to work with greykite
4. Forecast 30 days ahead with greykite
5. Note observation of what prices the model predicts

In [1]:
```
import sys
sys.path.append('../greykite')
```

In [2]:
```
import pandas as pd
import greykite
from greykite.framework.templates.autogen.forecast_config import ForecastConfig
from greykite.framework.templates.autogen.forecast_config import MetadataParam
from greykite.framework.templates.forecaster import Forecaster
from greykite.framework.templates.model_templates import ModelTemplateEnum
import datetime
import plotly

#Supress default INFO logging
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
import logging
logger = logging.getLogger()
logger.setLevel(logging.CRITICAL)
import logging, sys
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [3]:
```
df = pd.read_csv('datasets/rollingsales_brooklyn.xls_prepped_bare.csv', usecols=
```

In [4]:
```
df['SALE DATE']= pd.to_datetime(df['SALE DATE'])
```

In [5]:
```
df.dropna(inplace=True)
df.reset_index(drop=True)
```

Out[5]:

	SALE PRICE	SALE DATE
0	1300000	2020-04-28
1	75000	2020-11-30
2	830000	2020-06-26
3	1188000	2020-07-20
4	990000	2021-02-22
...	...	...
11619	470000	2020-12-01
11620	15000	2021-02-18
11621	600000	2021-02-12
11622	900000	2020-11-19
11623	960000	2020-12-21

11624 rows × 2 columns

In [6]:
```
df = df.rename(columns={'SALE DATE':'ts', 'SALE PRICE': 'y'})
df.columns = df.columns.astype(str)
df = df.set_index(['ts'], drop=True)
df.index= pd.to_datetime(df.index)
```

In [7]:
```
df
```

Out[7]:

ts	y
2020-04-28	1300000
2020-11-30	75000
2020-06-26	830000
2020-07-20	1188000
2021-02-22	990000
...	...
2020-12-01	470000
2021-02-18	15000
2021-02-12	600000
2020-11-19	900000
2020-12-21	960000

11624 rows × 1 columns

In [8]:
```
df = df.resample('D').mean()
```

In [9]:
```
df = df.reset_index()
```

In [10]:
```
df.fillna(0)
```

Out[10]:

	ts	y
0	2020-04-01	3.977437e+06
1	2020-04-02	8.185471e+05
2	2020-04-03	1.815030e+06
3	2020-04-04	2.333627e+05
4	2020-04-05	0.000000e+00
...	...	...
360	2021-03-27	0.000000e+00
361	2021-03-28	0.000000e+00
362	2021-03-29	1.002984e+06
363	2021-03-30	1.058857e+06
364	2021-03-31	1.126519e+06

365 rows × 2 columns

In [11]:
```
df['ts']= pd.to_datetime(df['ts'])
```

In [12]:

```
# df = ...  # your input timeseries!
df=df
metadata = MetadataParam(
    time_col= 'ts',      # time column in `df`
    value_col='y'      # value in `df`
)
forecaster = Forecaster()  # creates forecasts and stores the result
result = forecaster.run_forecast_config(
     df=df,
     config=ForecastConfig(
         # uses the SILVERKITE model template parameters
         model_template=ModelTemplateEnum.SILVERKITE.name,
         forecast_horizon=30,  # forecasts 30 steps ahead
         coverage=0.95,         # 95% prediction intervals
         metadata_param=metadata
     )
 )
# Access the result
forecaster.forecast_result
```

```
Fitting 3 folds for each of 1 candidates, totalling 3 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worker
s.
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:   13.9s finished
```

Out[12]:

```
ForecastResult(timeseries=<greykite.framework.input.univariate_time_series.Univ
ariateTimeSeries object at 0x000001DE5FFCB8E0>, grid_search=RandomizedSearchCV
(cv=RollingTimeSeriesSplit(expanding_window=True, forecast_horizon=30,
            max_splits=3, min_train_periods=60, periods_between_splits=30,
            periods_between_train_test=0, use_most_recent_splits=False),
                   estimator=Pipeline(steps=[('input',
                                              PandasFeatureUnion(transformer_li
st=[('date',

Pipeline(steps=[('select_date',

ColumnSelector(column_names=['ts'...
                            'OutsideTolerance3p': make_scorer(score_func_finit
e),
                            'OutsideTolerance4p': make_scorer(score_func_finit
e),
                            'OutsideTolerance5p': make_scorer(score_func_finit
e),
                            'Q80': make_scorer(score_func_finite),
                            'Q95': make_scorer(score_func_finite),
                            'Q99': make_scorer(score_func_finite),
                            'R2': make_scorer(score_func_finite),
                            'RMSE': make_scorer(score_func_finite),
                            'sMAPE': make_scorer(score_func_finite)},
                   verbose=1), model=Pipeline(steps=[('input',
                 PandasFeatureUnion(transformer_list=[('date',
                                                       Pipeline(steps=[('select
_date',
                                                                        ColumnS
elector(column_names=['ts']))])),
                                                      ('response',
                                                       Pipeline(steps=[('select
_val',
                                                                        ColumnS
elector(column_names=['y'])),
                                                                       ('outlie
r',
                                                                        ZscoreO
utlierTransformer()),
                                                                       ('null',
                                                                        NullTra
nsformer(impute_algorithm='interpolate',

impute_params={'axis': 0,

'limit_direct...
```

```
                                                            'simple_freq': <Sim
pleTimeFrequencyEnum.DAY: Frequency(default_horizon=30, seconds_per_observation
=86400, valid_seas={'QUARTERLY_SEASONALITY', 'WEEKLY_SEASONALITY', 'YEARLY_SEAS
ONALITY', 'MONTHLY_SEASONALITY'})>,
                                                            'start_year': 202
0},
                                         uncertainty_dict={'params': {'condit
ional_cols': ['dow_hr'],
                                                                        'quanti
le_estimation_method': 'normal_fit',
                                                                        'quanti
les': [0.025000000000000022,

0.975],
                                                                        'sample
_size_thresh': 5,
                                                                        'small_
sample_size_method': 'std_quantiles',
                                                                        'small_
sample_size_quantile': 0.98},
                                                             'uncertainty_metho
d': 'simple_conditional_residuals'}))]), backtest=<greykite.framework.output.un
ivariate_forecast.UnivariateForecast object at 0x000001DE66647340>, forecast=<g
reykite.framework.output.univariate_forecast.UnivariateForecast object at 0x000
001DE600B0F70>)
```

In [13]:

```
ts = result.timeseries
fig = ts.plot()
plotly.io.show(fig)
```

In [14]:
```
ts = result.timeseries
fig = ts.plot()
plotly.io.show(fig)
```

In [15]:
```
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)   # for generating offline graphs within Jupy

backtest = result.backtest
fig = backtest.plot()
iplot(fig)
```

In [16]:
```
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)

fig = backtest.plot_components()
iplot(fig)  # fig.show() if you are using "PROPHET" template
```

In [17]:
```
forecast = result.forecast
fig = forecast.plot()
iplot(fig)
```

In [18]:
```
from plotly.offline import init_notebook_mode, iplot
from greykite.common.evaluation import EvaluationMetricEnum

init_notebook_mode(connected=True)   # for generating offline graphs within Jupy

# MAPE by day of week
fig = backtest.plot_grouping_evaluation(
    score_func=EvaluationMetricEnum.MeanAbsolutePercentError.get_metric_func(),
    score_func_name=EvaluationMetricEnum.MeanAbsolutePercentError.get_metric_nam
    which="train", # "train" or "test" set
    groupby_time_feature="dow",  # day of week
    groupby_sliding_window_size=None,
    groupby_custom_column=None)
iplot(fig)

# RMSE over time
fig = backtest.plot_grouping_evaluation(
    score_func=EvaluationMetricEnum.RootMeanSquaredError.get_metric_func(),
    score_func_name=EvaluationMetricEnum.RootMeanSquaredError.get_metric_name(),
    which="test", # "train" or "test" set
    groupby_time_feature=None,
    groupby_sliding_window_size=7,  # weekly aggregation of daily data
    groupby_custom_column=None)
iplot(fig)
```

In [19]:
```
summary = result.model[-1].summary()  # -1 retrieves the estimator from the pip
print(summary)
```

```
================================ Model Summary ===============================
=

Number of observations: 365,   Number of features: 92
Method: Ridge regression
Number of nonzero features: 92
Regularization parameter: 475.1

Residuals:
         Min           1Q       Median           3Q          Max
  -1.185e+06   -3.785e+05   -1.946e+05    6.041e+04    1.128e+07

            Pred_col    Estimate   Std. Err Pr(>)_boot sig. code
95%CI
           Intercept   1.301e+06  5.848e+04     <2e-16       ***    (1.197e+0
6, 1.415e+06)
  events_C...New Year       481.2      511.3      0.504
(0., 1760.0)
  events_C...w Year-1      -730.3      738.9      0.150
(-2469.0, 0.)
  events_C...w Year-2      -486.1      607.3      0.144
(-2211.0, 0.)
  events_C...w Year+1      -212.7      267.0      0.560
(-853.6, 0.)
  events_C...w Year+2     -1604.0     1648.0      0.498
(-6137.0, 0.)
 events_Christmas Day       226.8      441.8      0.690                    (-39
5.1, 1251.0)
  events_C...as Day-1       202.3      492.6      0.438                    (-71
4.3, 1390.0)
  events_C...as Day-2      1762.0     1653.0      0.174
(0., 5509.0)
  events_C...as Day+1       645.9      627.5      0.154
(0., 1963.0)
  events_C...as Day+2       413.2      497.2      0.540
(0., 1760.0)
  events_E...Ireland]      -933.1      888.7      0.142
(-3081.0, 0.)
  events_E...eland]-1      -632.4      653.7      0.164
(-2331.0, 0.)
  events_E...eland]-2      -200.3      295.2      0.244                     (-9
08.3, 105.1)
  events_E...eland]+1      3799.0     3457.0      0.164
(0., 1.135e+04)
  events_E...eland]+2      -804.5      867.9      0.534
(-2858.0, 0.)
   events_Good Friday      -380.7      446.7      0.154
(-1472.0, 0.)
 events_Good Friday-1      -214.2      425.9      0.378                    (-11
63.0, 411.0)
 events_Good Friday-2     -1125.0      923.8      0.116
(-3123.0, 0.)
 events_Good Friday+1      -200.3      295.2      0.244                     (-9
08.3, 105.1)
 events_Good Friday+2      -632.4      653.7      0.164
(-2331.0, 0.)
  events_I...ence Day     -3204.0     2275.0      0.188
(-8575.0, 0.)
  events_I...ce Day-1      1734.0     3620.0      0.596                (-4488.
0, 1.006e+04)
  events_I...ce Day-2     -1464.0     1201.0      0.118
(-4538.0, 0.)
  events_I...ce Day+1     -2215.0     1507.0      0.090         .
(-5352.0, 0.)
  events_I...ce Day+2     -1121.0      954.7      0.146                    (-30
20.0, 23.91)
     events_Labor Day      -477.2      477.1      0.232                    (-16
25.0, 162.3)
   events_Labor Day-1     -1416.0     1340.0      0.170                    (-46
82.0, 57.48)
   events_Labor Day-2     -2834.0     1987.0      0.090         .
```

```
(-6850.0, 0.)
   events_Labor Day+1     -2524.0     1878.0      0.216
(-7159.0, 0.)
   events_Labor Day+2     -1593.0     1107.0      0.094          .
(-3962.0, 0.)
  events_Memorial Day      -895.4      882.1      0.144
(-2912.0, 0.)
  events_M...al Day-1     -1348.0     1360.0      0.498
(-4473.0, 0.)
  events_M...al Day-2     -1685.0     1621.0      0.166
(-5319.0, 0.)
  events_M...al Day+1      -791.0      794.4      0.182
(-2671.0, 0.)
  events_M...al Day+2      2875.0     2723.0      0.168
(0., 8943.0)
 events_New Years Day      -1.965      214.1      0.996                        (-4
53.5, 434.8)
  events_N...rs Day-1       34.26      433.4      0.962                       (-85
0.3, 1031.0)
  events_N...rs Day-2       481.9      602.8      0.546
(0., 2157.0)
  events_N...rs Day+1       354.3      407.5      0.526
(0., 1343.0)
  events_N...rs Day+2       34.37      206.1      0.540                        (-3
75.6, 482.9)
         events_Other       53.55  1.896e+04      1.000              (-2.733e+0
4, 3.949e+04)
       events_Other-1  -1.581e+04     6684.0      0.014          *     (-2.844e+
04, -1778.0)
       events_Other-2  -1.406e+04     6508.0      0.032          *     (-2.749e+
04, -2707.0)
       events_Other+1  -1.257e+04     6456.0      0.048          *      (-2.494e
+04, -22.63)
       events_Other+2     -1904.0     7418.0      0.794              (-1.619e+0
4, 1.238e+04)
  events_Thanksgiving      -547.4      624.1      0.532
(-2110.0, 0.)
  events_T...giving-1       18.92      244.9      0.950                        (-4
48.1, 641.8)
  events_T...giving-2       259.0      297.0      0.530
(0., 963.4)
  events_T...giving+1      -859.1      877.1      0.496
(-2903.0, 0.)
  events_T...giving+2      -191.9      260.8      0.566
(-818.7, 0.)
  events_Veterans Day     -2001.0     1946.0      0.148
(-6723.0, 0.)
  events_V...ns Day-1      -108.5      177.9      0.306                        (-5
86.8, 83.93)
  events_V...ns Day-2      -691.9      687.9      0.166
(-2210.0, 0.)
  events_V...ns Day+1      -269.5      426.4      0.632                       (-15
11.0, 103.3)
  events_V...ns Day+2      2699.0     2671.0      0.452
(0., 8655.0)
        str_dow_2-Tue     -2465.0     6292.0      0.680              (-1.350e+0
4, 1.013e+04)
        str_dow_3-Wed   2.071e+04  1.039e+04      0.034          *       (2736.
0, 4.222e+04)
        str_dow_4-Thu   3.246e+04  1.614e+04      0.048          *       (3483.
0, 6.661e+04)
        str_dow_5-Fri   1.111e+04     5491.0      0.050          .        (48.1
3, 2.199e+04)
        str_dow_6-Sat  -2.887e+04     6358.0     <2e-16        ***  (-4.102e+04,
-1.637e+04)
        str_dow_7-Sun  -1.954e+04     5456.0     <2e-16        ***     (-2.954e+
04, -8301.0)
                  ct1  -1.435e+04  1.148e+04      0.208                 (-3.814e
+04, 6373.0)
       is_weekend:ct1  -1.727e+04     4730.0     <2e-16        ***     (-2.686e+
04, -8619.0)
    str_dow_2-Tue:ct1     -3233.0     3731.0      0.358                 (-1.026e
+04, 5288.0)
    str_dow_3-Wed:ct1      7751.0     5228.0      0.142                   (-1928.
```

```
0, 1.844e+04)
    str_dow_4-Thu:ct1      -148.5     3941.0      0.976                 (-931
5.0, 6144.0)
    str_dow_5-Fri:ct1      2736.0     2680.0      0.282                 (-215
8.0, 8209.0)
    str_dow_6-Sat:ct1  -1.008e+04     3226.0      0.002       **     (-1.654e+
04, -3860.0)
    str_dow_7-Sun:ct1     -7195.0     2861.0      0.012        *     (-1.356e+
04, -1747.0)
  ct1:sin1_tow_weekly   1.923e+04     7298.0      0.008       **       (6590.
0, 3.410e+04)
  ct1:cos1_tow_weekly  -1.249e+04     5908.0      0.036        *      (-2.437e
+04, -958.7)
  ct1:sin2_tow_weekly     -1618.0     5741.0      0.776               (-1.203e
+04, 9876.0)
  ct1:cos2_tow_weekly      1855.0     7362.0      0.806             (-1.330e+0
4, 1.461e+04)
      sin1_tow_weekly   7.095e+04  1.650e+04     <2e-16      ***    (4.138e+0
4, 1.025e+05)
      cos1_tow_weekly  -6.458e+04  2.058e+04      0.002       **  (-1.083e+05,
-2.461e+04)
      sin2_tow_weekly  -2.156e+04  1.787e+04      0.224             (-5.712e+0
4, 1.332e+04)
      cos2_tow_weekly   2.599e+04  1.886e+04      0.190                (-9987.
0, 6.150e+04)
      sin3_tow_weekly  -1.053e+04  1.964e+04      0.576             (-4.855e+0
4, 2.798e+04)
      cos3_tow_weekly     -8372.0  1.089e+04      0.436             (-2.861e+0
4, 1.369e+04)
      sin4_tow_weekly   1.053e+04  1.964e+04      0.576             (-2.798e+0
4, 4.855e+04)
      cos4_tow_weekly     -8372.0  1.089e+04      0.436             (-2.861e+0
4, 1.369e+04)
   sin1_toq_quarterly   1.671e+04  2.054e+04      0.450             (-2.393e+0
4, 5.629e+04)
   cos1_toq_quarterly   5.776e+04  2.205e+04      0.012        *     (1.874e+0
4, 1.010e+05)
   sin2_toq_quarterly  -3.225e+04  2.255e+04      0.158               (-7.378e
+04, 8607.0)
   cos2_toq_quarterly     -4085.0  2.009e+04      0.850             (-4.320e+0
4, 3.426e+04)
   sin3_toq_quarterly  -1.691e+04  2.304e+04      0.466             (-6.165e+0
4, 2.627e+04)
   cos3_toq_quarterly      2180.0  2.022e+04      0.902             (-4.074e+0
4, 3.706e+04)
   sin4_toq_quarterly  -1.294e+04  2.630e+04      0.598             (-6.752e+0
4, 3.621e+04)
   cos4_toq_quarterly  -2.195e+04  1.533e+04      0.154               (-5.086e
+04, 5805.0)
   sin5_toq_quarterly  -3.552e+04  2.329e+04      0.128             (-8.010e+0
4, 1.111e+04)
   cos5_toq_quarterly  -1.933e+04  1.904e+04      0.302             (-5.659e+0
4, 1.874e+04)
Signif. Code: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Multiple R-squared: 0.06071,   Adjusted R-squared: 0.04643
F-statistic: 0.86709 on 5 and 358 DF,   p-value: 0.5108
Model AIC: 12303.0,   model BIC: 12328.0

WARNING: the F-ratio and its p-value on regularized methods might be misleadin
g, they are provided only for reference purposes.
```

Observation:

Per the model in Linkedin Greykite, properties prices will fluctuate around a range in the low $1.1 - 1.3 million dollar mark for Brooklyn