LinkedIn GreyKite Forecast - Bronx 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_bronx.xls_prepped_bare.csv', usecols=['SA
```

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	600000	2021-01-15
1	475000	2020-07-23
2	289000	2020-08-25
3	526000	2020-09-22
4	734000	2020-04-22
...	...	...
3977	290809	2021-03-04
3978	129000	2020-08-31
3979	210000	2021-01-23
3980	305803	2020-12-18
3981	178000	2020-11-05

3982 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
2021-01-15	600000
2020-07-23	475000
2020-08-25	289000
2020-09-22	526000
2020-04-22	734000
...	...
2021-03-04	290809
2020-08-31	129000
2021-01-23	210000
2020-12-18	305803
2020-11-05	178000

3982 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	1.234333e+06
1	2020-04-02	5.502250e+05
2	2020-04-03	6.185000e+05
3	2020-04-04	0.000000e+00
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	4.119000e+05
363	2021-03-30	5.161438e+05
364	2021-03-31	5.053775e+05

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:   14.0s finished
```

Out[12]:

```
ForecastResult(timeseries=<greykite.framework.input.univariate_time_series.Univ
ariateTimeSeries object at 0x0000022CFF6F8310>, 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={'WEEKLY_SEASONALITY', 'MONTHLY_SEASONALITY', 'YEARLY_SEASON
ALITY', 'QUARTERLY_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 0x0000022C979E6070>, forecast=<g
reykite.framework.output.univariate_forecast.UnivariateForecast object at 0x000
0022C979E4970>)
```

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: 1.000e+05

Residuals:
         Min           1Q       Median           3Q          Max
  -9.500e+05   -4.399e+05   -3.448e+05   -9.317e+04    2.568e+07

             Pred_col   Estimate   Std. Err Pr(>)_boot sig. code
95%CI
            Intercept  9.538e+05  8.436e+04     <2e-16       ***  (8.127e+05,
1.136e+06)
  events_C...New Year     -1.284      1.727      0.174                 (-6.005,
0.09491)
  events_C...w Year-1     -4.526      4.462      0.156                      (-1
4.88, 0.)
  events_C...w Year-2     -1.872      2.146      0.146                      (-
7.995, 0.)
  events_C...w Year+1     -7.523      7.902      0.546                      (-2
5.47, 0.)
  events_C...w Year+2     -6.789      6.703      0.148                      (-2
3.85, 0.)
 events_Christmas Day     -3.916      3.606      0.132                      (-1
1.98, 0.)
  events_C...as Day-1     -3.628      3.694      0.144                      (-1
2.22, 0.)
  events_C...as Day-2     -3.282      3.287      0.162                      (-1
0.44, 0.)
  events_C...as Day+1     -4.208      4.404      0.166                      (-1
4.11, 0.)
  events_C...as Day+2     -4.505      4.591      0.168                      (-1
5.65, 0.)
  events_E...Ireland]     -4.375      4.245      0.156                      (-1
4.03, 0.)
  events_E...eland]-1     -4.039      3.931      0.152                      (-1
3.37, 0.)
  events_E...eland]-2     -3.706      3.808      0.138                      (-1
4.27, 0.)
  events_E...eland]+1     -3.983      4.033      0.166                      (-1
3.02, 0.)
  events_E...eland]+2      1.353      1.937      0.626                    (-0.3
187, 6.4)
    events_Good Friday    -3.378      3.536      0.142                      (-1
1.68, 0.)
 events_Good Friday-1     -3.306       3.29      0.158                      (-1
1.42, 0.)
 events_Good Friday-2     -8.679      6.171      0.082         .            (-2
2.51, 0.)
 events_Good Friday+1     -3.706      3.808      0.138                      (-1
4.27, 0.)
 events_Good Friday+2     -4.039      3.931      0.152                      (-1
3.37, 0.)
  events_I...ence Day      12.42      17.78      0.450                    (-15.0
1, 51.24)
  events_I...ce Day-1     -6.376       5.01      0.108                      (-1
7.51, 0.)
  events_I...ce Day-2      48.26      44.59      0.144
(0., 143.5)
  events_I...ce Day+1       31.2      38.36      0.570                    (-19.5
1, 115.8)
  events_I...ce Day+2      56.03      59.72      0.540                     (-11.
3, 182.7)
      events_Labor Day     -7.33      5.469      0.102                      (-1
9.55, 0.)
    events_Labor Day-1     -4.67       3.63      0.098         .            (-1
3.96, 0.)
    events_Labor Day-2    -6.719      4.812      0.096         .            (-1
```

```
7.49, 0.)
   events_Labor Day+1    -7.497      5.631      0.108                   (-1
9.96, 0.)
   events_Labor Day+2    -8.476      6.037      0.096         .         (-2
0.65, 0.)
  events_Memorial Day    -3.506      4.013      0.512                   (-1
3.84, 0.)
  events_M...al Day-1    -4.066      3.994      0.148                    (-
13.4, 0.)
  events_M...al Day-2    -4.628       4.57      0.138                   (-1
6.25, 0.)
  events_M...al Day+1    -2.953       3.12      0.170                   (-1
1.12, 0.)
  events_M...al Day+2    -3.996      4.166      0.514                   (-1
4.31, 0.)
 events_New Years Day     27.66      27.68      0.576
(0., 84.66)
  events_N...rs Day-1     37.08      37.22      0.538
(0., 113.5)
  events_N...rs Day-2    -1.441      1.887      0.160                    (-
6.669, 0.)
  events_N...rs Day+1     18.23      19.71      0.512
(0., 69.82)
  events_N...rs Day+2     8.798      9.453      0.560
(0., 32.04)
         events_Other    -31.94      53.83      0.552                (-127.
3, 80.04)
       events_Other-1     116.8      135.3      0.412                (-94.7
3, 405.8)
       events_Other-2     223.8      149.7      0.114                (-17.3
4, 546.7)
       events_Other+1     82.67      79.22      0.264                (-56.6
3, 251.6)
       events_Other+2    -38.85      60.06      0.506                (-145.
0, 93.37)
  events_Thanksgiving     2.997      3.292      0.534
(0., 11.02)
  events_T...giving-1     8.223      8.428      0.184
(0., 27.17)
  events_T...giving-2     256.8      228.1      0.216
(0., 766.3)
  events_T...giving+1    -2.229      2.425      0.134                    (-
8.496, 0.)
  events_T...giving+2    -9.266      9.263      0.500                   (-3
1.31, 0.)
  events_Veterans Day    -4.789      4.561      0.138                   (-1
5.49, 0.)
  events_V...ns Day-1   -0.9835      1.581      0.230               (-5.04
1, 0.5143)
  events_V...ns Day-2    -0.597       1.39      0.370                 (-4.1
15, 1.25)
  events_V...ns Day+1    -3.623      3.705      0.144                   (-1
2.76, 0.)
  events_V...ns Day+2    -3.867      4.107      0.146                   (-1
5.51, 0.)
        str_dow_2-Tue     159.8      201.3      0.518                (-119.
5, 615.5)
        str_dow_3-Wed     145.9      134.3      0.262                (-54.8
1, 470.2)
        str_dow_4-Thu     43.02      77.22      0.578                 (-98.
1, 196.4)
        str_dow_5-Fri    -57.48      70.78      0.386                (-187.
9, 84.23)
        str_dow_6-Sat    -134.6      53.48      0.016         *     (-240.
1, -33.38)
        str_dow_7-Sun    -111.6      59.74      0.060         .      (-221.
9, 5.355)
                  ct1     24.12      47.13      0.600                 (-65.
0, 122.9)
       is_weekend:ct1    -138.0      48.59      0.008        **     (-238.
9, -47.36)
    str_dow_2-Tue:ct1     121.8      134.7      0.470                (-60.8
2, 429.8)
    str_dow_3-Wed:ct1     59.84      66.39      0.350                (-41.4
```

```
2, 210.6)
    str_dow_4-Thu:ct1      26.59      43.02      0.518                 (-46.8
3, 113.6)
    str_dow_5-Fri:ct1     -14.94      41.31      0.718                 (-88.1
1, 74.91)
    str_dow_6-Sat:ct1     -72.59      29.78      0.020         *      (-131.
7, -14.54)
    str_dow_7-Sun:ct1     -65.43      28.35      0.018         *       (-123.
4, -14.5)
  ct1:sin1_tow_weekly      293.5      136.2      0.034         *        (54.2
8, 596.4)
  ct1:cos1_tow_weekly     -3.638      116.1      0.984                 (-226.
9, 224.1)
  ct1:sin2_tow_weekly      92.57      160.4      0.570                 (-166.
4, 427.7)
  ct1:cos2_tow_weekly      -24.9       90.6      0.762                 (-204.
3, 150.8)
      sin1_tow_weekly      529.3      230.7      0.026         *       (122.
8, 1003.0)
      cos1_tow_weekly     -4.394      195.1      0.984                 (-367.
7, 377.8)
      sin2_tow_weekly      64.32      257.5      0.800                 (-352.
3, 608.8)
      cos2_tow_weekly      -74.9      174.3      0.644                 (-451.
1, 230.1)
      sin3_tow_weekly     -3.564      183.0      0.990                 (-401.
3, 347.5)
      cos3_tow_weekly     -78.33      252.1      0.762                 (-581.
2, 374.3)
      sin4_tow_weekly      3.564      183.0      0.990                 (-347.
5, 401.3)
      cos4_tow_weekly     -78.33      252.1      0.762                 (-581.
2, 374.3)
   sin1_toq_quarterly     -173.2      153.0      0.240                 (-488.
3, 113.1)
   cos1_toq_quarterly      125.3      270.7      0.646                 (-361.
9, 640.6)
   sin2_toq_quarterly      234.5      227.8      0.318                 (-166.
6, 718.4)
   cos2_toq_quarterly      422.1      209.8      0.046         *        (18.8
8, 848.3)
   sin3_toq_quarterly     -203.0      243.3      0.432                 (-713.
4, 221.8)
   cos3_toq_quarterly      390.4      189.7      0.030         *        (19.1
3, 731.2)
   sin4_toq_quarterly      368.3      226.8      0.096         .       (-24.0
2, 867.6)
   cos4_toq_quarterly     -11.83      212.4      0.956                 (-399.
6, 409.6)
   sin5_toq_quarterly      25.25      152.7      0.876                 (-261.
5, 338.7)
   cos5_toq_quarterly      332.8      266.1      0.188                 (-151.
7, 884.8)
Signif. Code: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Multiple R-squared: 0.0002528,   Adjusted R-squared: 0.002863
F-statistic: -0.00010588 on 0 and 364 DF,   p-value: nan
Model AIC: 12644.0,   model BIC: 12644.0

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, property prices will be stable around mid $900,000-1$ million for Bronx