LinkedIn GreyKite Forecast - Queens 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 [2]:
```
import sys
sys.path.append('../greykite')
```

In [3]:
```
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 [4]:
```
df = pd.read_csv('datasets/rollingsales_queens.xls_prepped_bare.csv', usecols=['S
```

In [5]:
```
df['SALE DATE']= pd.to_datetime(df['SALE DATE'])
```

In [6]:
```
df.dropna(inplace=True)
df.reset_index(drop=True)
```

Out[6]:

	SALE PRICE	SALE DATE
0	4121000	2020-07-16
1	584569	2020-08-28
2	800000	2021-01-11
3	300000	2020-12-16
4	360000	2020-06-23
...	...	...
13166	254563	2020-12-31
13167	254563	2020-12-31
13168	1000000	2020-07-10
13169	960000	2020-07-01
13170	718000	2020-11-03

13171 rows × 2 columns

In [7]:
```
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 [11]:
```
df
```

Out[11]:

ts	y
2020-07-16	4121000
2020-08-28	584569
2021-01-11	800000
2020-12-16	300000
2020-06-23	360000
...	...
2020-12-31	254563
2020-12-31	254563
2020-07-10	1000000
2020-07-01	960000
2020-11-03	718000

13171 rows × 1 columns

In [12]:
```
df = df.resample('D').mean()
```

In [13]:
```
df = df.reset_index()
```

In [14]:
```
df.fillna(0)
```

Out[14]:

	ts	y
0	2020-04-01	961150.000000
1	2020-04-02	753357.142857
2	2020-04-03	681724.206897
3	2020-04-04	0.000000
4	2020-04-05	0.000000
...	...	...
360	2021-03-27	0.000000
361	2021-03-28	0.000000
362	2021-03-29	694114.470588
363	2021-03-30	747610.935484
364	2021-03-31	602154.750000

365 rows × 2 columns

In [15]:
```
df['ts']= pd.to_datetime(df['ts'])
```

In [16]:

```
# 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:   12.2s finished
```

Out[16]:

```
ForecastResult(timeseries=<greykite.framework.input.univariate_time_series.Univ
ariateTimeSeries object at 0x00000153CA690A60>, 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={'YEARLY_SEASONALITY', 'MONTHLY_SEASONALITY', 'QUARTERLY_SEA
SONALITY', 'WEEKLY_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 0x00000153C6760730>, forecast=<g
reykite.framework.output.univariate_forecast.UnivariateForecast object at 0x000
00153C6753DC0>)
```

In [17]:

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

In [18]:
```
ts = result.timeseries
fig = ts.plot()
plotly.io.show(fig)
```

In [19]:
```
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 [20]:
```
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 [21]:
```
forecast = result.forecast
fig = forecast.plot()
iplot(fig)
```

In [22]:
```
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_name
    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 [23]:
```
summary = result.model[-1].summary()  # -1 retrieves the estimator from the pipe
print(summary)
```

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

Number of observations: 365,   Number of features: 92
Method: Ridge regression
Number of nonzero features: 92
Regularization parameter: 9770.0

Residuals:
         Min           1Q       Median           3Q          Max
  -8.924e+05   -2.735e+05   -1.967e+05   -8.940e+04    1.378e+07

             Pred_col   Estimate   Std. Err Pr(>)_boot sig. code
95%CI
            Intercept  8.882e+05  6.122e+04     <2e-16       ***  (7.821e+05,
1.007e+06)
  events_C...New Year     -7.897      11.63      0.222                 (-40.7
9, 1.979)
  events_C...w Year-1    -0.7624      9.193      0.584                 (-20.4
7, 20.81)
  events_C...w Year-2     -18.19      21.51      0.518                    (-7
6.82, 0.)
  events_C...w Year+1     -14.87      18.42      0.150                    (-6
3.06, 0.)
  events_C...w Year+2     -22.23      22.21      0.136                    (-7
6.25, 0.)
 events_Christmas Day      83.83      83.91      0.536
(0., 273.1)
  events_C...as Day-1      219.5      216.7      0.500
(0., 675.2)
  events_C...as Day-2     -8.717      13.75      0.232                 (-45.2
7, 3.039)
  events_C...as Day+1     -50.96      51.77      0.158                    (-1
81.9, 0.)
  events_C...as Day+2     -9.693      13.83      0.564                  (-49.
45, 1.33)
  events_E...Ireland]      31.57      31.67      0.526
(0., 109.7)
  events_E...eland]-1     -56.38      55.57      0.166                    (-1
92.6, 0.)
  events_E...eland]-2     -39.28      42.81      0.520                    (-1
50.9, 0.)
  events_E...eland]+1      0.492      8.065      0.944                 (-16.2
1, 18.75)
  events_E...eland]+2      22.57      26.06      0.510
(0., 91.91)
   events_Good Friday      -22.4      23.37      0.534                    (-8
1.44, 0.)
 events_Good Friday-1     -10.98      12.82      0.178                    (-4
0.95, 0.)
 events_Good Friday-2     -54.12      39.83      0.210                    (-1
44.7, 0.)
 events_Good Friday+1     -39.28      42.81      0.520                    (-1
50.9, 0.)
 events_Good Friday+2     -56.38      55.57      0.166                    (-1
92.6, 0.)
  events_I...ence Day     -52.81      48.28      0.274                    (-1
63.6, 0.)
  events_I...ce Day-1     -79.45      60.09      0.096         .          (-2
12.4, 0.)
  events_I...ce Day-2     -56.98       40.9      0.092         .          (-1
52.1, 0.)
  events_I...ce Day+1     -113.7      84.69      0.100                    (-3
14.1, 0.)
  events_I...ce Day+2     -49.63      37.15      0.208                    (-1
32.5, 0.)
     events_Labor Day     -104.2      95.03      0.268                    (-3
28.5, 0.)
   events_Labor Day-1     -71.63      51.81      0.108                    (-1
85.5, 0.)
   events_Labor Day-2      -22.2      23.85      0.244                 (-78.9
```

```
8, 5.749)
   events_Labor Day+1     -44.77      35.74      0.130                      (-1
20.4, 0.)
   events_Labor Day+2     1099.0     1116.0      0.398                  (-25.7
8, 3323.0)
  events_Memorial Day     -44.97      44.54      0.500                      (-1
38.2, 0.)
  events_M...al Day-1     -56.89      56.18      0.170                      (-1
83.2, 0.)
  events_M...al Day-2     -21.87      23.16      0.508                      (-8
1.36, 0.)
  events_M...al Day+1     -31.59      32.57      0.168                      (-1
04.7, 0.)
  events_M...al Day+2     -30.11      32.49      0.536                      (-1
10.0, 0.)
 events_New Years Day     -9.243      13.27      0.206                   (-47.1
7, 2.419)
  events_N...rs Day-1     0.6554      9.234      0.972                    (-17.
23, 22.3)
  events_N...rs Day-2      27.86      27.27      0.168
(0., 89.87)
  events_N...rs Day+1     -18.35      19.28      0.152                      (-6
4.98, 0.)
  events_N...rs Day+2     -27.83      29.38      0.172                      (-9
6.58, 0.)
         events_Other     -564.4      350.6      0.106                  (-1275.
0, 72.93)
       events_Other-1     1139.0     1264.0      0.406                  (-762.
4, 3788.0)
       events_Other-2      971.2     1241.0      0.516                  (-689.
0, 3677.0)
       events_Other+1      380.3     1021.0      0.728                  (-962.
7, 2698.0)
       events_Other+2     -353.0      362.0      0.310                  (-1042.
0, 353.8)
  events_Thanksgiving     -9.927      12.35      0.156                      (-4
0.52, 0.)
  events_T...giving-1     -16.13      17.43      0.138                      (-5
6.89, 0.)
  events_T...giving-2      -10.6      13.53      0.522                       (-
47.4, 0.)
  events_T...giving+1     -5.426      10.08      0.324                    (-34.
09, 6.38)
  events_T...giving+2     -34.71      33.54      0.152                      (-1
17.7, 0.)
  events_Veterans Day     -40.67       43.0      0.520                      (-1
56.7, 0.)
  events_V...ns Day-1     -14.17      15.85      0.512                      (-5
5.53, 0.)
  events_V...ns Day-2     -15.17      18.66      0.496                      (-6
6.84, 0.)
  events_V...ns Day+1     -18.53      18.86      0.500                       (-
59.2, 0.)
  events_V...ns Day+2      181.4      174.1      0.472
(0., 562.3)
        str_dow_2-Tue     -331.8      386.3      0.376                  (-1091.
0, 425.5)
        str_dow_3-Wed     1962.0     1498.0      0.230                  (-358.
7, 5201.0)
        str_dow_4-Thu      187.8      571.1      0.752                  (-836.
3, 1396.0)
        str_dow_5-Fri      157.0      449.8      0.696                  (-656.
5, 1181.0)
        str_dow_6-Sat     -833.5      480.3      0.094           .      (-1717.
0, 133.9)
        str_dow_7-Sun    -1195.0      472.7      0.006          **     (-2101.
0, -227.8)
                  ct1     -149.9      306.8      0.612                   (-794.
2, 403.6)
       is_weekend:ct1     -934.3      419.6      0.026           *     (-1710.
0, -98.99)
    str_dow_2-Tue:ct1     -156.6      217.6      0.448                   (-559.
3, 254.9)
    str_dow_3-Wed:ct1      850.7      703.9      0.290                  (-208.
```

```
0, 2388.0)
    str_dow_4-Thu:ct1     -15.94      221.2      0.926                 (-414.
1, 449.8)
    str_dow_5-Fri:ct1      89.71      229.9      0.694                 (-357.
7, 566.9)
    str_dow_6-Sat:ct1     -281.6      320.7      0.376                 (-833.
9, 416.0)
    str_dow_7-Sun:ct1     -652.7      245.8      0.004        **      (-1104.
0, -163.4)
  ct1:sin1_tow_weekly     1446.0      883.1      0.082         .       (-66.3
6, 3332.0)
  ct1:cos1_tow_weekly     -681.2      457.8      0.130                (-1572.
0, 229.7)
  ct1:sin2_tow_weekly      74.94      536.0      0.918               (-1021.
0, 1029.0)
  ct1:cos2_tow_weekly     -270.2      868.9      0.734               (-2025.
0, 1266.0)
      sin1_tow_weekly     3413.0     1792.0      0.050         .        (188.
4, 7326.0)
      cos1_tow_weekly    -1461.0     1017.0      0.132                (-3409.
0, 499.3)
      sin2_tow_weekly     -395.9     1108.0      0.724               (-2766.
0, 1629.0)
      cos2_tow_weekly     -409.5     1802.0      0.798               (-4210.
0, 2755.0)
      sin3_tow_weekly    -1781.0     1554.0      0.260               (-5086.
0, 1041.0)
      cos3_tow_weekly     2055.0     1362.0      0.128                (-473.
4, 4641.0)
      sin4_tow_weekly     1781.0     1554.0      0.260               (-1041.
0, 5086.0)
      cos4_tow_weekly     2055.0     1362.0      0.128                (-473.
4, 4641.0)
   sin1_toq_quarterly    -3685.0     1424.0      0.012         *     (-6822.0,
-1300.0)
   cos1_toq_quarterly     2179.0     1559.0      0.134                (-370.
7, 5397.0)
   sin2_toq_quarterly    -2640.0      818.2      0.002        **     (-4269.0,
-1131.0)
   cos2_toq_quarterly     -20.49     1913.0      0.996               (-3738.
0, 3677.0)
   sin3_toq_quarterly      661.6     1372.0      0.626               (-1762.
0, 3665.0)
   cos3_toq_quarterly      388.2     1578.0      0.794               (-2254.
0, 3701.0)
   sin4_toq_quarterly      410.0     1023.0      0.684               (-1458.
0, 2525.0)
   cos4_toq_quarterly     2667.0     1935.0      0.158                (-776.
8, 6645.0)
   sin5_toq_quarterly    -2185.0     1343.0      0.100                (-4798.
0, 207.9)
   cos5_toq_quarterly     2108.0     1617.0      0.188                (-581.
7, 5240.0)
Signif. Code: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Multiple R-squared: 0.0038,   Adjusted R-squared: 0.005245
F-statistic: -0.03447 on 0 and 364 DF,   p-value: nan
Model AIC: 12315.0,   model BIC: 12317.0

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, house prices will be stable around almost $900,000 for Queens