LinkedIn GreyKite Forecast - Staten Island 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_statenisland.xls_prepped_bare.csv', usec
```

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	315000	2020-10-02
1	450000	2020-06-24
2	525000	2020-07-02
3	455000	2021-01-21
4	720000	2020-10-15
...	...	...
4510	500000	2020-06-18
4511	537000	2020-09-25
4512	525000	2020-10-09
4513	500000	2020-06-02
4514	500000	2020-06-02

4515 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 [8]:
```
df
```

Out[8]:

ts	y
2020-10-02	315000
2020-06-24	450000
2020-07-02	525000
2021-01-21	455000
2020-10-15	720000
...	...
2020-06-18	500000
2020-09-25	537000
2020-10-09	525000
2020-06-02	500000
2020-06-02	500000

4515 rows × 1 columns

In [9]:
```
df = df.resample('D').mean()
```

In [10]:
```
df = df.reset_index()
```

In [11]:
```
df.fillna(0)
```

Out[11]:

	ts	y
0	2020-04-01	577500.000000
1	2020-04-02	650666.666667
2	2020-04-03	519414.285714
3	2020-04-04	0.000000
4	2020-04-05	0.000000
...	...	...
358	2021-03-25	0.000000
359	2021-03-26	0.000000
360	2021-03-27	0.000000
361	2021-03-28	0.000000
362	2021-03-29	435000.000000

363 rows × 2 columns

In [12]:
```
df['ts']= pd.to_datetime(df['ts'])
```

In [13]:

```
# 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:   12.0s finished
```

Out[13]:

```
ForecastResult(timeseries=<greykite.framework.input.univariate_time_series.Univ
ariateTimeSeries object at 0x000002267659F700>, 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', 'QUARTERLY_SEASONALITY', 'MONTHLY_SEA
SONALITY', 'YEARLY_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 0x000002267694B700>, forecast=<g
reykite.framework.output.univariate_forecast.UnivariateForecast object at 0x000
002267CE4E8B0>)
```

In [14]:

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

In [15]:
```
ts = result.timeseries
fig = ts.plot()
plotly.io.show(fig)
```

In [16]:
```
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 [17]:
```
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 [18]:
```
forecast = result.forecast
fig = forecast.plot()
iplot(fig)
```

In [19]:
```
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 [20]:
```
summary = result.model[-1].summary()  # -1 retrieves the estimator from the pipe
print(summary)
```

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

Number of observations: 363,   Number of features: 88
Method: Ridge regression
Number of nonzero features: 88
Regularization parameter: 1.000e+05

Residuals:
         Min           1Q       Median          3Q         Max
  -5.422e+05   -7.856e+04   -3.075e+04    2.286e+04    4.392e+06

             Pred_col    Estimate   Std. Err Pr(>)_boot sig. code
95%CI
            Intercept   6.056e+05  1.828e+04     <2e-16       ***  (5.740e+05,
6.436e+05)
  events_C...New Year     -0.1276      0.289      0.416                  (-0.8
89, 0.301)
  events_C...w Year-1     -0.2338     0.3531      0.602                (-1.22
5, 0.08686)
  events_C...w Year-2      -0.625      0.717      0.522                     (-
2.373, 0.)
  events_C...w Year+1      -0.166     0.3226      0.684                 (-1.15
9, 0.1662)
  events_C...w Year+2     -0.2074     0.3338      0.250                  (-1.0
27, 0.109)
 events_Christmas Day       0.391     0.4423      0.190
(0., 1.498)
  events_C...as Day-1      0.7182     0.7936      0.558
(0., 2.714)
  events_C...as Day-2      0.2267     0.3382      0.258                 (-0.21
27, 1.099)
  events_C...as Day+1     0.06869     0.2519      0.488                 (-0.42
91, 0.687)
  events_C...as Day+2     -0.2566     0.3135      0.234                (-1.06
8, 0.03435)
  events_E...Ireland]     0.06068     0.2484      0.506                (-0.491
6, 0.6566)
  events_E...eland]-1    -0.04682     0.2702      0.512                   (-0.
77, 0.427)
  events_E...eland]-2     -0.1542     0.3303      0.706                 (-1.00
7, 0.1652)
  events_E...eland]+1     -0.2329      0.338      0.584                 (-1.17
5, 0.0859)
  events_E...eland]+2      -1.088      1.115      0.514                     (-
3.745, 0.)
    events_Good Friday    -0.2642     0.3731      0.206                 (-1.26
9, 0.0389)
  events_Good Friday-1    -0.6828     0.6944      0.170                     (-
2.342, 0.)
  events_Good Friday-2      1.767      1.807      0.528
(0., 5.944)
  events_Good Friday+1    -0.1542     0.3303      0.706                 (-1.00
7, 0.1652)
  events_Good Friday+2   -0.04682     0.2702      0.512                   (-0.
77, 0.427)
  events_I...ence Day       1.357      1.188      0.266
(0., 4.348)
  events_I...ce Day-1       3.252      2.514      0.100
(0., 8.96)
  events_I...ce Day-2       10.61      12.04      0.590                  (-4.1
94, 36.32)
  events_I...ce Day+1     -0.5441     0.8114      0.432                 (-2.36
2, 0.6875)
  events_I...ce Day+2      -2.445      2.261      0.188                 (-7.34
4, 0.1349)
     events_Labor Day  -0.0001173     0.5037      1.000                  (-1.1
07, 1.072)
    events_Labor Day-1     -1.113     0.9413      0.140                     (-
3.373, 0.)
    events_Labor Day-2    -0.5205     0.7624      0.428                 (-2.24
```

```
6, 0.6396)
   events_Labor Day+1      0.1395      0.9067      0.936                  (-1.6
52, 2.136)
   events_Labor Day+2      -2.084       1.568      0.106                     (-
5.417, 0.)
  events_Memorial Day      0.1345        0.31      0.744                 (-0.256
5, 0.9456)
  events_M...al Day-1     -0.3575      0.4631      0.530                     (-
1.686, 0.)
  events_M...al Day-2     -0.8493      0.8981      0.166                     (-
2.873, 0.)
  events_M...al Day+1      0.6274      0.6338      0.150
(0., 2.296)
  events_M...al Day+2     -0.7284      0.7652      0.508                     (-
2.674, 0.)
 events_New Years Day      -1.199       1.161      0.496                     (-
3.858, 0.)
  events_N...rs Day-1     -0.6248       0.662      0.520
(-2.27, 0.)
  events_N...rs Day-2       43.92       44.59      0.140
(0., 131.5)
  events_N...rs Day+1      -1.767       1.858      0.502                     (-
6.662, 0.)
  events_N...rs Day+2     -0.7825      0.8113      0.174                     (-
2.708, 0.)
         events_Other       37.49       39.84      0.376                   (-1
1.6, 125.9)
       events_Other-1       13.11       16.03      0.396                  (-16.
79, 46.63)
       events_Other-2       31.65       39.78      0.506                  (-21.
62, 120.0)
       events_Other+1      -9.571       12.19      0.424                  (-33.
25, 14.32)
       events_Other+2       25.74       39.93      0.604                  (-25.
59, 113.7)
  events_Thanksgiving      -1.232       1.247      0.174                     (-
4.221, 0.)
  events_T...giving-1     0.07556       0.275      0.484                 (-0.418
4, 0.7861)
  events_T...giving-2      0.8792      0.9364      0.534
(0., 3.146)
  events_T...giving+1      -2.545       2.457      0.178                     (-
7.954, 0.)
  events_T...giving+2      -1.687       1.596      0.178
(-5.23, 0.)
  events_Veterans Day     -0.4824      0.6013      0.556                     (-
2.112, 0.)
  events_V...ns Day-1     -0.5289      0.5968      0.512                     (-
1.988, 0.)
  events_V...ns Day-2     -0.6877      0.6735      0.130                     (-
2.243, 0.)
  events_V...ns Day+1      0.3978      0.4646      0.172
(0., 1.639)
  events_V...ns Day+2     -0.4655      0.4947      0.496                     (-
1.567, 0.)
        str_dow_2-Tue      -9.469       10.96      0.394                  (-30.
34, 9.062)
        str_dow_3-Wed       37.73       39.72      0.348                   (-1
6.9, 128.8)
        str_dow_4-Thu      0.8309       16.73      0.970                  (-29.
83, 35.73)
        str_dow_5-Fri        19.8       29.23      0.510                  (-23.
24, 85.99)
        str_dow_6-Sat      -19.18       23.04      0.370                   (-6
1.37, 32.0)
        str_dow_7-Sun      -17.01       16.82      0.314                   (-4
6.89, 17.2)
                  ct1       12.23       16.89      0.462                  (-16.
11, 46.51)
       is_weekend:ct1      -12.12       18.68      0.486                  (-47.
34, 26.75)
    str_dow_2-Tue:ct1      -7.552       6.476      0.228                   (-2
1.63, 3.48)
    str_dow_3-Wed:ct1       26.97       30.66      0.452                  (-13.
```

```
37, 92.84)
    str_dow_4-Thu:ct1      0.2639       9.25      0.968                      (-18.
15, 19.06)
    str_dow_5-Fri:ct1       13.93      21.32      0.544                      (-16.
25, 62.55)
    str_dow_6-Sat:ct1      -5.349      15.99      0.728                       (-3
1.96, 29.0)
    str_dow_7-Sun:ct1      -6.775      11.42      0.538                      (-27.
17, 18.16)
      sin1_tow_weekly       53.15      52.99      0.282                      (-38.
67, 171.8)
      cos1_tow_weekly      -51.92      37.39      0.164                      (-12
8.4, 15.21)
      sin2_tow_weekly      -2.519      41.39      0.956                      (-87.
05, 79.44)
      cos2_tow_weekly      -10.66       53.6      0.860                      (-12
5.7, 82.56)
      sin3_tow_weekly      -59.72      55.67      0.264                       (-18
0.0, 39.2)
      cos3_tow_weekly       18.13      37.48      0.600                       (-4
6.8, 102.5)
      sin4_tow_weekly       59.72      55.67      0.264                       (-3
9.2, 180.0)
      cos4_tow_weekly       18.13      37.48      0.600                       (-4
6.8, 102.5)
   sin1_toq_quarterly      -35.42      38.11      0.364                      (-11
8.6, 33.58)
   cos1_toq_quarterly       96.69      53.35      0.056        .              (10.
14, 213.8)
   sin2_toq_quarterly      -18.61      43.56      0.688                      (-10
9.4, 62.87)
   cos2_toq_quarterly       34.58      47.51      0.462                      (-40.
03, 145.9)
   sin3_toq_quarterly      -23.77      30.22      0.386                      (-94.
68, 29.02)
   cos3_toq_quarterly      -21.77      56.25      0.678                      (-12
5.4, 99.35)
   sin4_toq_quarterly       19.69       35.6      0.570                       (-5
0.0, 91.21)
   cos4_toq_quarterly       15.37      52.42      0.786                       (-7
7.6, 121.3)
   sin5_toq_quarterly       22.76      50.47      0.652                      (-77.
41, 115.7)
   cos5_toq_quarterly       41.54      40.93      0.296                      (-23.
96, 135.6)
Signif. Code: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Multiple R-squared: 0.000175,   Adjusted R-squared: 0.002807
F-statistic: -7.8377e-05 on 0 and 362 DF,   p-value: nan
Model AIC: 11407.0,   model BIC: 11407.0

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 stabe around $600,000.

Data might be skewed. Will have to analyze further with different parameters. Lower amount of data for Staten Island

Trend shows upward. Will have to analyze that futher.