lab6

June 7, 2022

[ ]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

[ ]:
```
coffee = pd.read_csv('coffee.csv', header = 0, index_col = 0, parse_dates =␣
↪True, squeeze = True)
```

[ ]:
```
coffee.head()
```

[ ]:

Date	Open	High	Low	Close	Volume	Currency
2000-01-03	122.25	124.00	116.10	116.50	6640	USD
2000-01-04	116.25	120.50	115.75	116.25	5492	USD
2000-01-05	115.00	121.00	115.00	118.60	6165	USD
2000-01-06	119.00	121.40	116.50	116.85	5094	USD
2000-01-07	117.25	117.75	113.80	114.15	6855	USD

[ ]:
```
coffee.shape
```

[ ]: (5683, 6)

[ ]:
```
ts_coffee = pd.read_csv('coffee.csv', header = 0, index_col = 0, parse_dates =␣
↪True)
```

[ ]:
```
ts_coffee.head()
```

[ ]:

Date	Open	High	Low	Close	Volume	Currency
2000-01-03	122.25	124.00	116.10	116.50	6640	USD
2000-01-04	116.25	120.50	115.75	116.25	5492	USD
2000-01-05	115.00	121.00	115.00	118.60	6165	USD
2000-01-06	119.00	121.40	116.50	116.85	5094	USD
2000-01-07	117.25	117.75	113.80	114.15	6855	USD

[ ]:
```
ts_coffee.shape
```

[ ]:
```
(5683, 6)
```

[ ]:
```
drop_columns = ['High','Low', 'Close', 'Volume', 'Currency']
```

[ ]:
```
ts_coffee=ts_coffee.drop(drop_columns,axis=1)
```

[ ]:
```
ts_coffee.head()
```

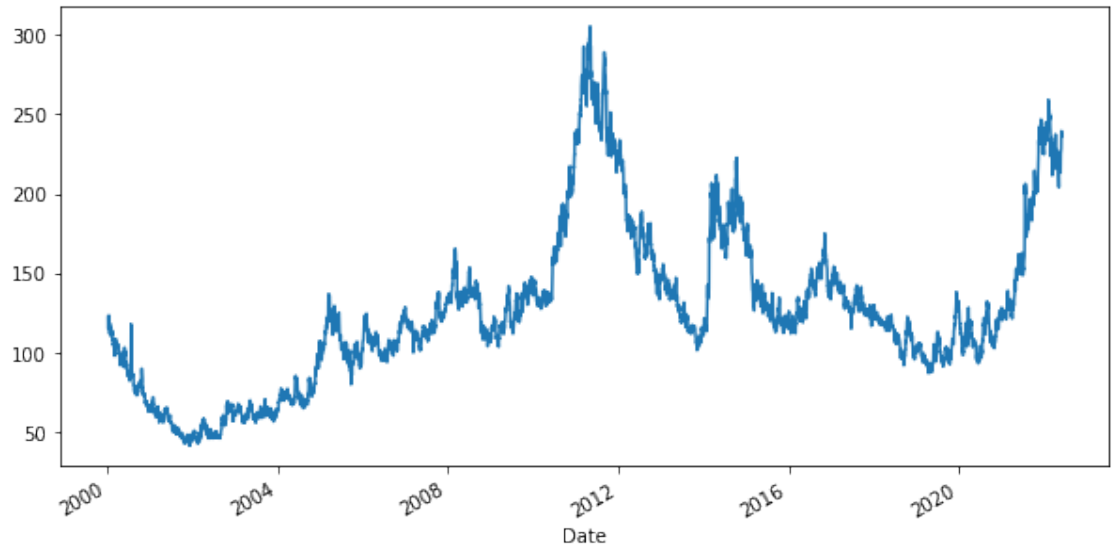
[ ]:
```
              Open
Date
2000-01-03  122.25
2000-01-04  116.25
2000-01-05  115.00
2000-01-06  119.00
2000-01-07  117.25
```

[ ]:
```
ts_coffee['Open']['2000-01']
```

[ ]:
```
Date
2000-01-03    122.25
2000-01-04    116.25
2000-01-05    115.00
2000-01-06    119.00
2000-01-07    117.25
2000-01-10    123.50
2000-01-11    115.50
2000-01-12    117.80
2000-01-13    119.25
2000-01-14    117.75
2000-01-18    111.75
2000-01-19    116.50
2000-01-20    118.25
2000-01-21    112.00
2000-01-24    110.95
2000-01-25    111.60
2000-01-26    112.50
2000-01-27    114.75
2000-01-28    115.10
2000-01-31    113.75
Name: Open, dtype: float64
```

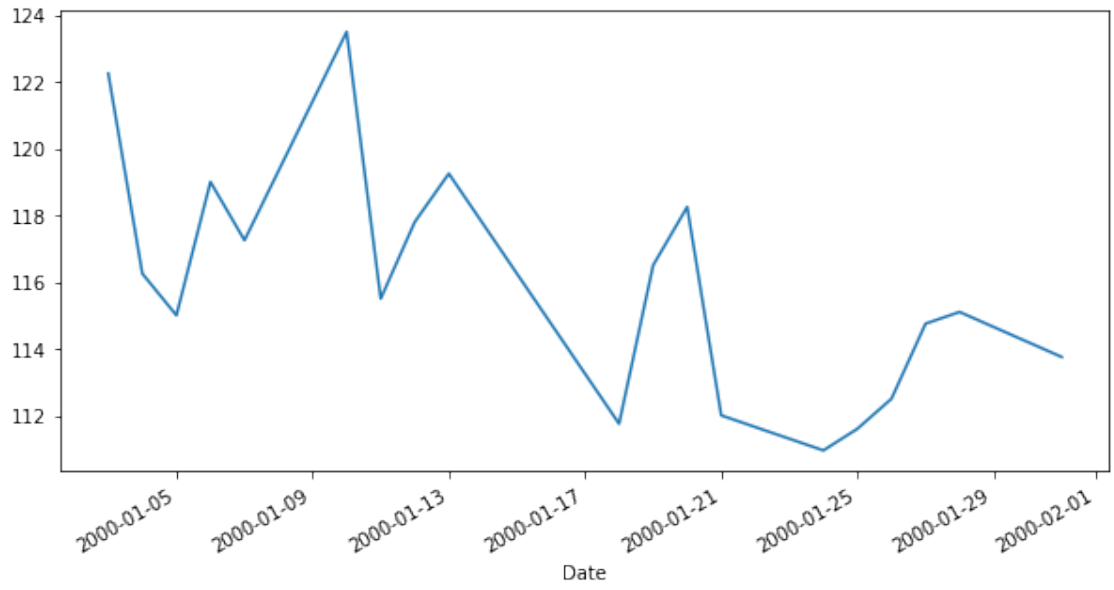
[ ]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('                  ')
ts_coffee.plot(ax=ax, legend=False)
pyplot.show()
```

Временной ряд в виде графика

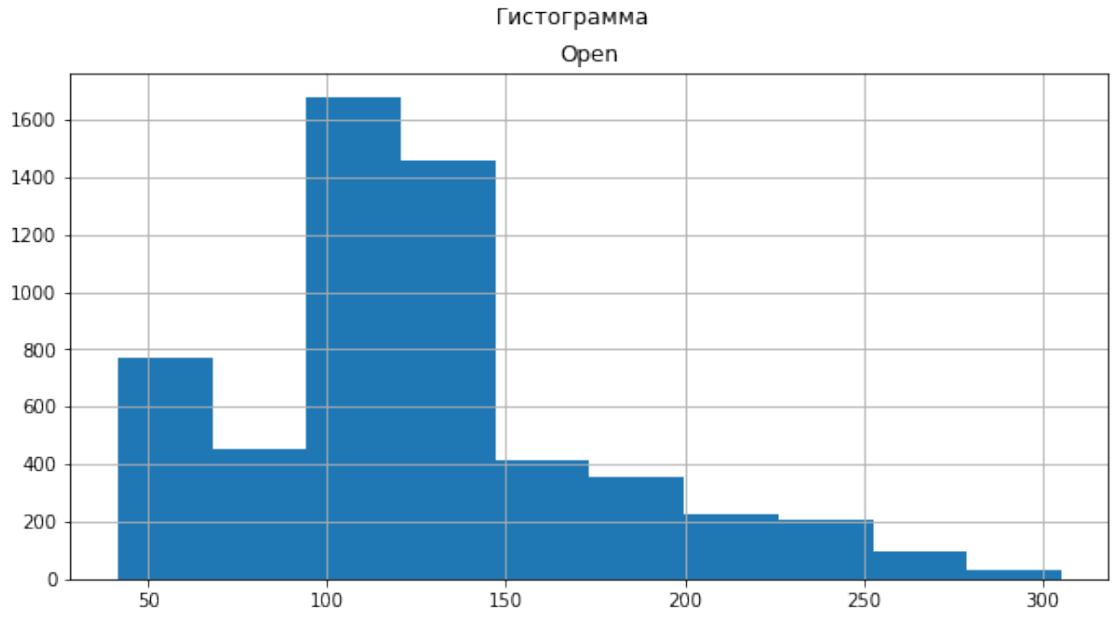



```
[ ]: fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
     fig.suptitle('    20       ')
     ts_coffee[:20].plot(ax=ax, legend=False)
     pyplot.show()
```

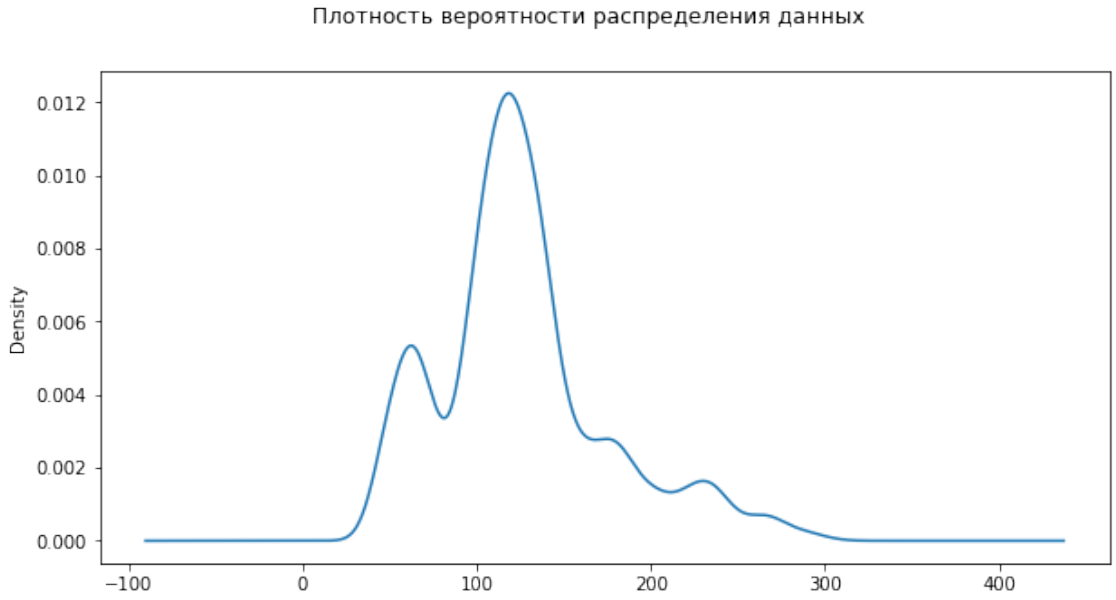
Первые 20 точек ряда



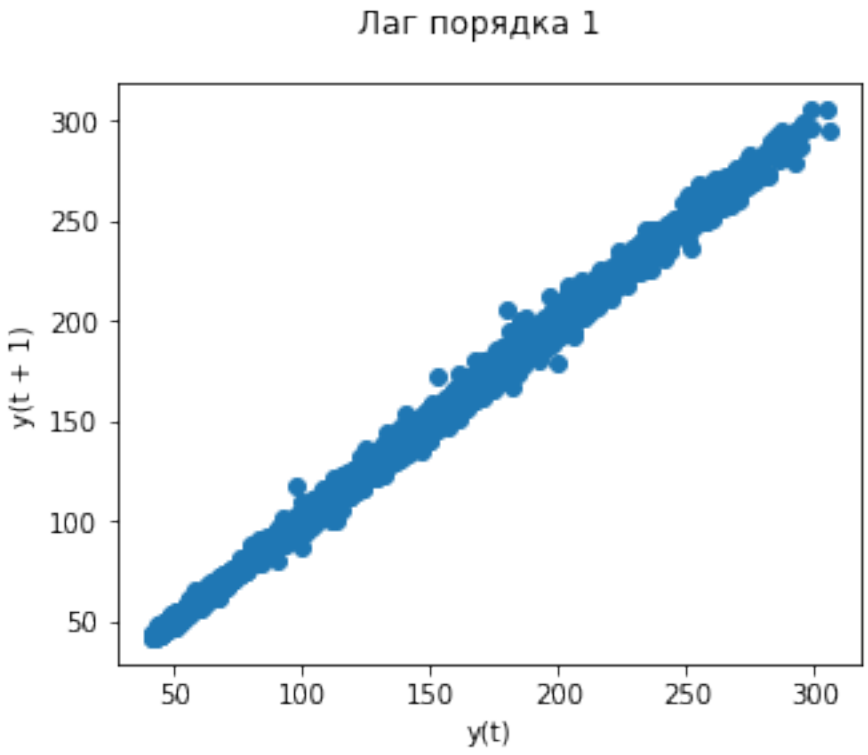
[ ]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('      ')
ts_coffee.hist(ax=ax, legend=False)
pyplot.show()
```



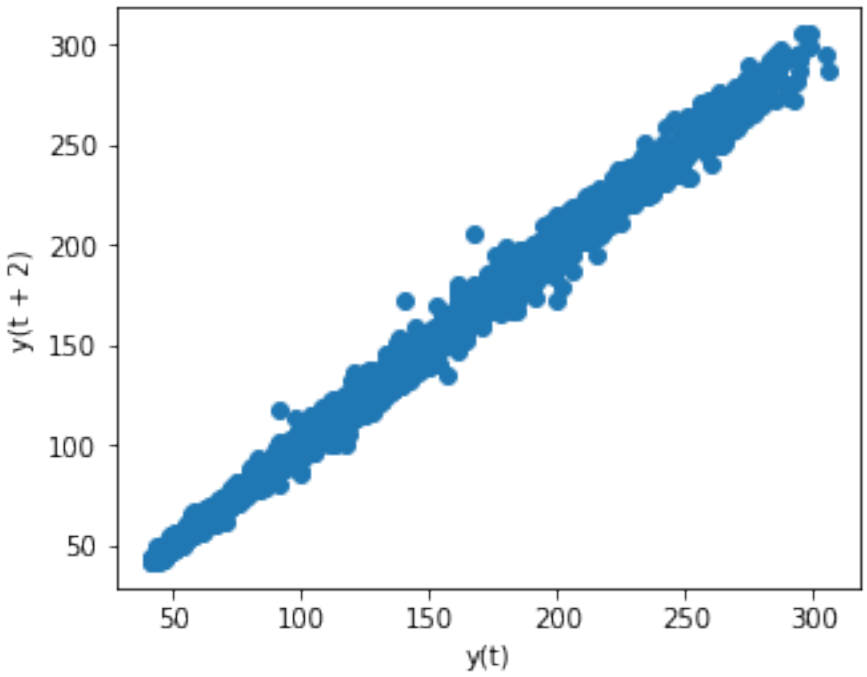
[ ]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('                          ')
ts_coffee.plot(ax=ax, kind='kde', legend=False)
pyplot.show()
```



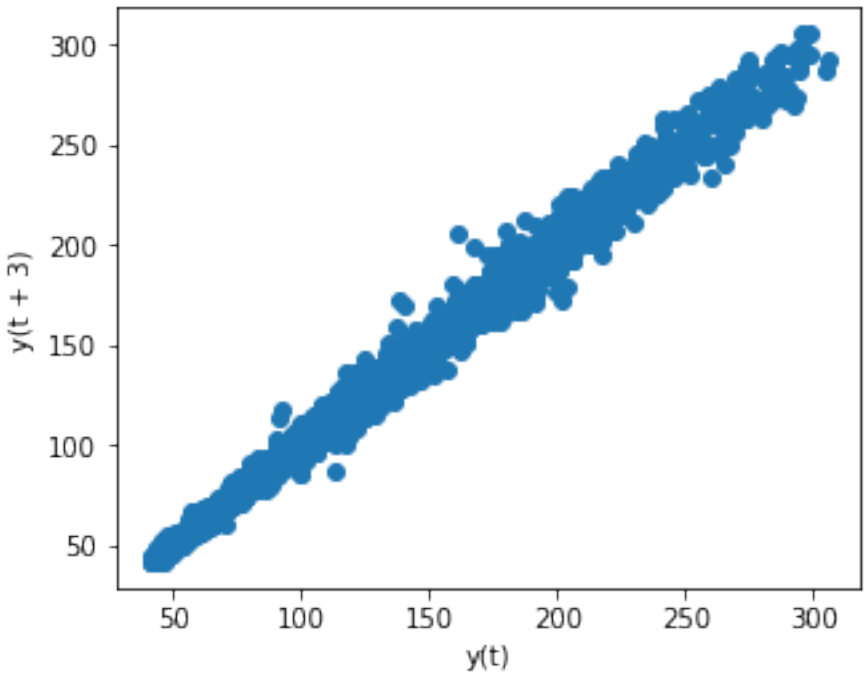
[ ]:
```
for i in range(1, 5):
    fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(5,4))
    fig.suptitle(f'        {i}')
    pd.plotting.lag_plot(ts_coffee, lag=i, ax=ax)
    pyplot.show()
```

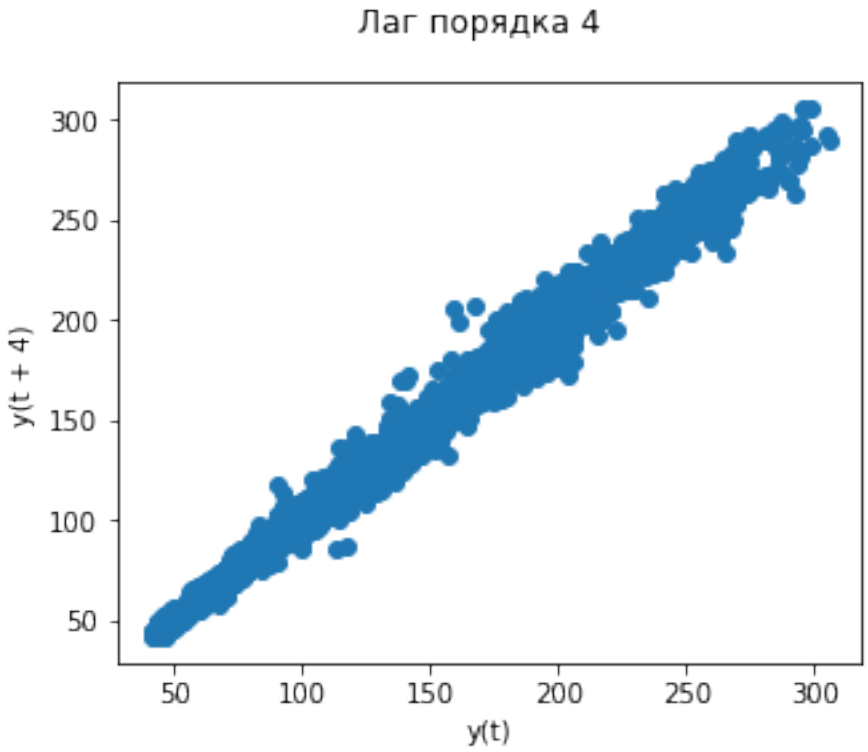


Лаг порядка 2

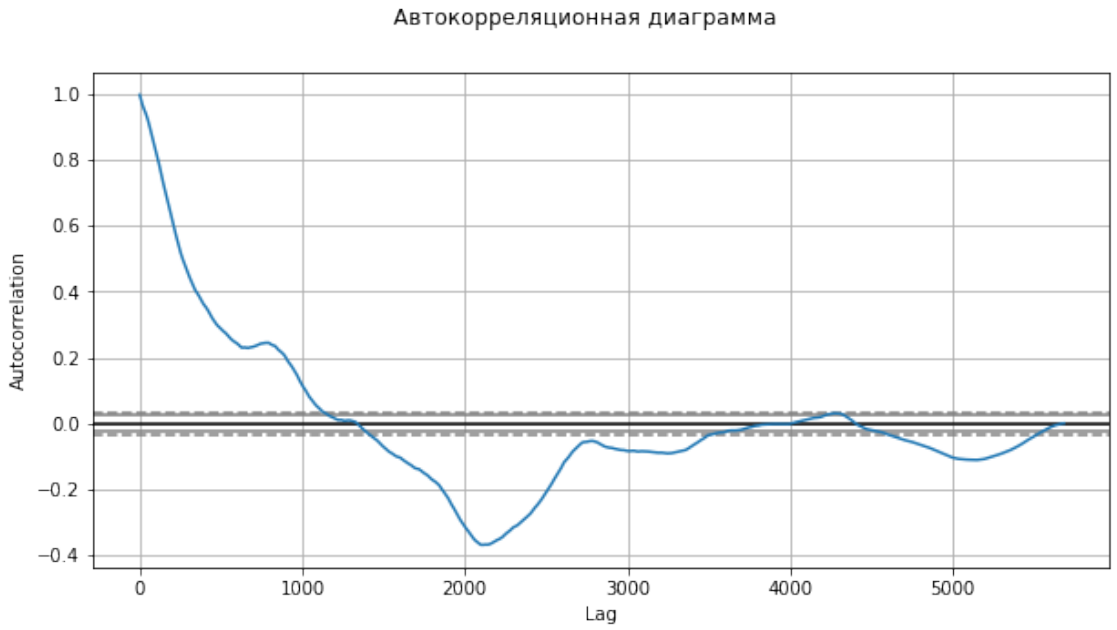


Лаг порядка 3

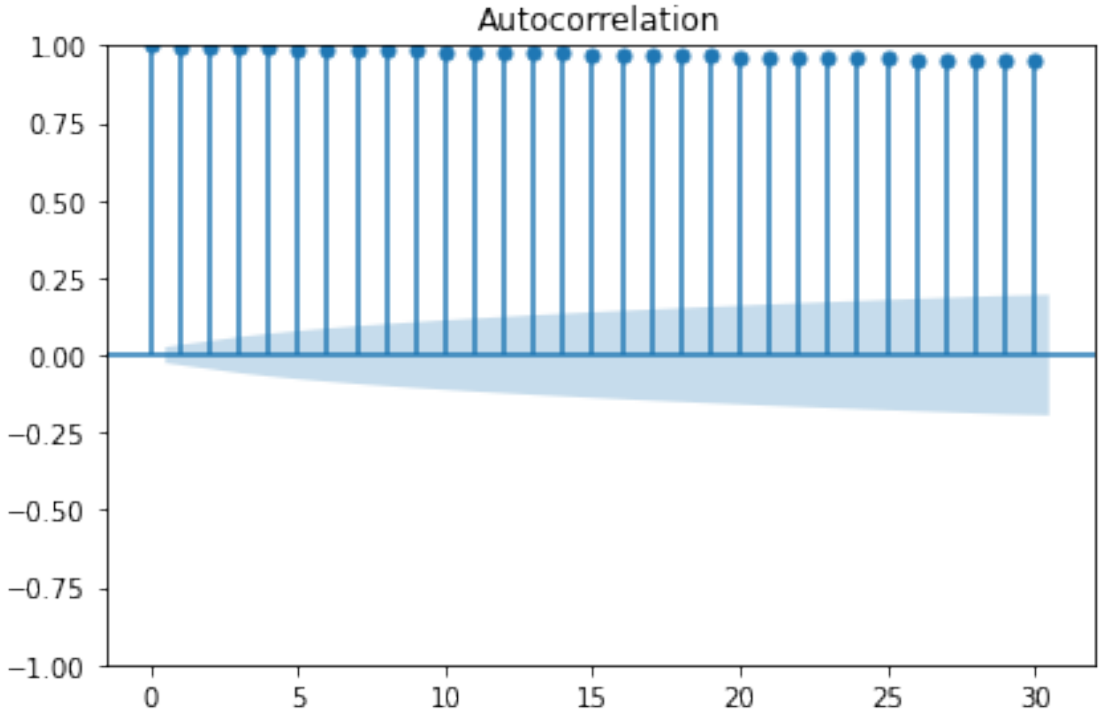




[ ]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('                ')
pd.plotting.autocorrelation_plot(ts_coffee, ax=ax)
pyplot.show()
```



[ ]:
```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(ts_coffee, lags=30)
plt.tight_layout()
```

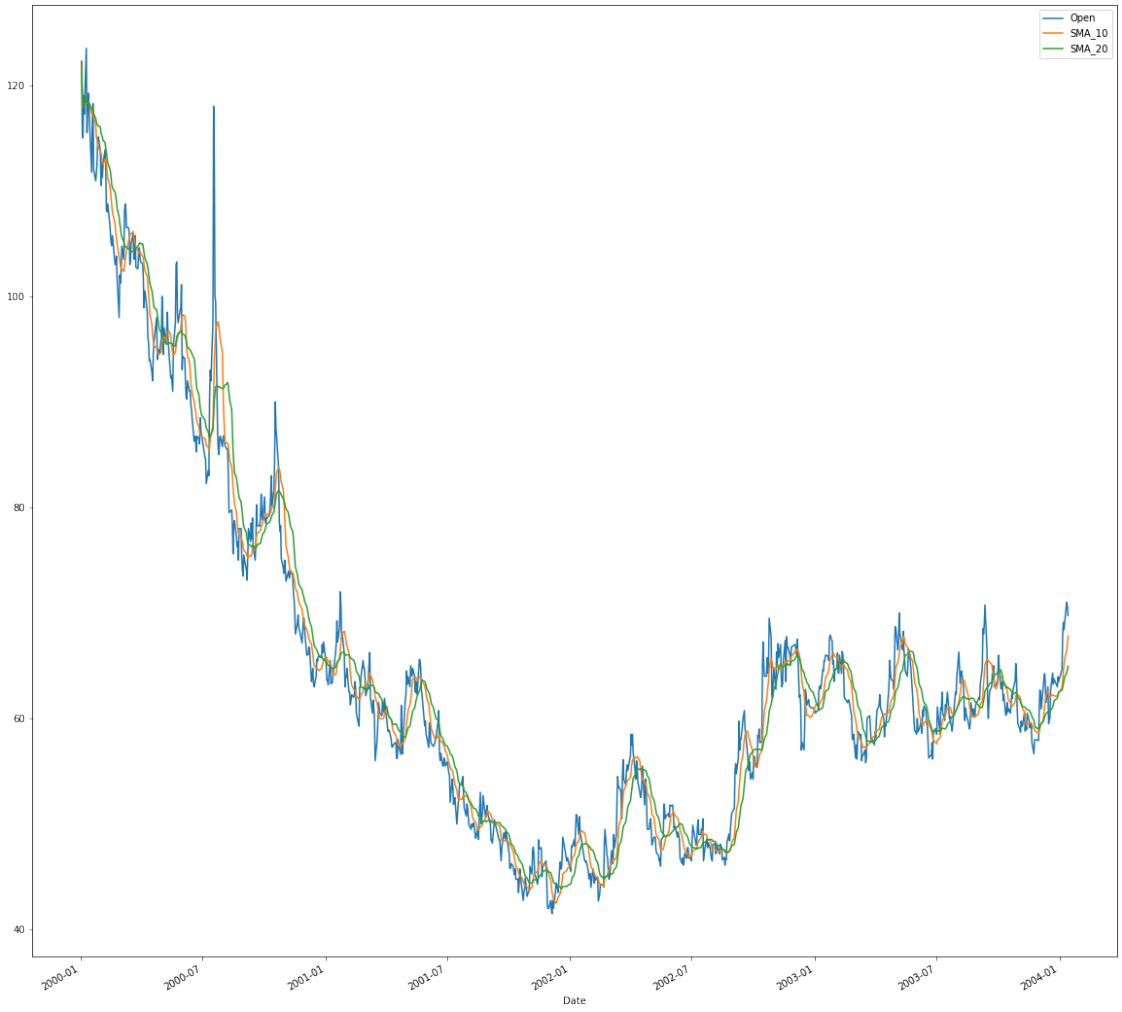


[ ]:
```
ts_coffee2 = ts_coffee.copy()
```

[ ]:
```
ts_coffee2['SMA_10'] = ts_coffee2['Open'].rolling(10, min_periods=1).mean()
ts_coffee2['SMA_20'] = ts_coffee2['Open'].rolling(20, min_periods=1).mean()
```

[ ]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(20,20))
fig.suptitle('                        ')
ts_coffee2[:1000].plot(ax=ax, legend=True)
pyplot.show()
```

Временной ряд со скользящими средними



[ ]:
```
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

[ ]:
```
xnum = list(range(ts_coffee2.shape[0]))
#
Y = ts_coffee2['Open'].values
train_size = int(len(Y) * 0.7)
xnum_train, xnum_test = xnum[0:train_size], xnum[train_size:]
train, test = Y[0:train_size], Y[train_size:]
history_arima = [x for x in train]
history_es = [x for x in train]
```

[ ]:
```
#           (p,d,q)
arima_order = (6,1,0)
#
predictions_arima = list()
for t in range(len(test)):
    model_arima = ARIMA(history_arima, order=arima_order)
    model_arima_fit = model_arima.fit()
    yhat_arima = model_arima_fit.forecast()[0]
    predictions_arima.append(yhat_arima)
    history_arima.append(test[t])
#            RMSE
error_arima = mean_squared_error(test, predictions_arima, squared=False)
```

[ ]:
```
#
predictions_es = list()
for t in range(len(test)):
    model_es = ExponentialSmoothing(history_es)
    model_es_fit = model_es.fit()
    yhat_es = model_es_fit.forecast()[0]
    predictions_es.append(yhat_es)
    history_es.append(test[t])
#            RMSE
error_es = mean_squared_error(test, predictions_es, squared=False)
```

[ ]:
```
np.mean(Y), error_arima, error_es
```

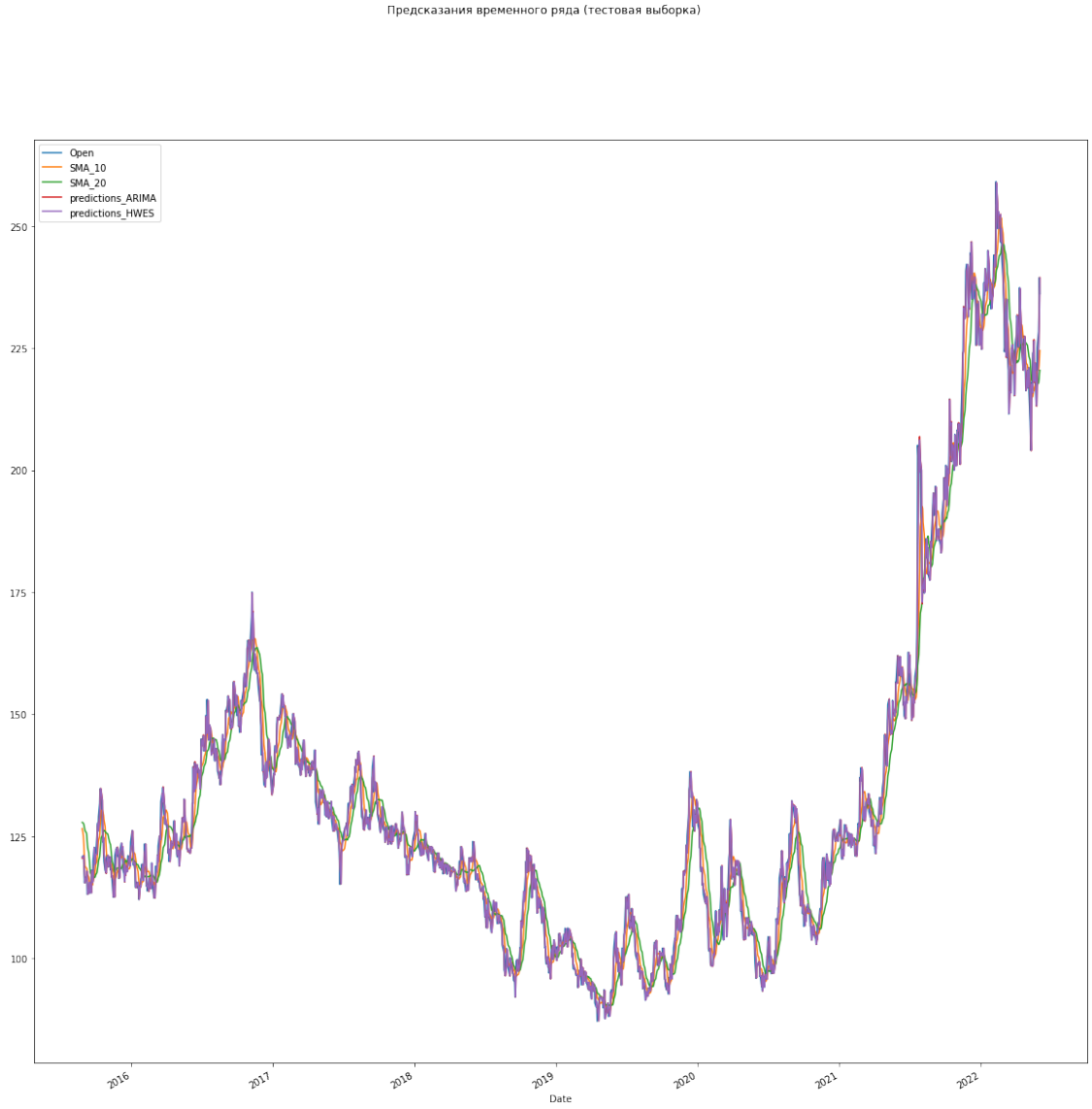
[ ]:
```
(126.20606545838467, 2.772484753663894, 2.7684518794710455)
```

[ ]:
```
ts_coffee2['predictions_ARIMA'] = (train_size * [np.NAN]) +␣
 ↪list(predictions_arima)
ts_coffee2['predictions_HWES'] = (train_size * [np.NAN]) + list(predictions_es)
```

[ ]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(20,20))
fig.suptitle('                ')
```

```
ts_coffee2[5600:].plot(ax=ax, legend=True)
pyplot.show()
```

[ ]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(20,20))
fig.suptitle('                    (          )')
ts_coffee2[train_size:].plot(ax=ax, legend=True)
pyplot.show()
```



[ ]:
```
from gplearn.genetic import SymbolicRegressor
```

[ ]:
```
function_set = ['add', 'sub', 'mul', 'div', 'sin']
est_gp = SymbolicRegressor(population_size=500, metric='mse',
                           generations=70, stopping_criteria=0.01,
                           init_depth=(4, 10), verbose=1,␣
 ↪function_set=function_set,
                           const_range=(-100, 100), random_state=0)
```

[ ]:
```
est_gp.fit(np.array(xnum_train).reshape(-1, 1), train.reshape(-1, 1))
```

```
    |   Population Average    |             Best Individual              |
```

```
---- ------------------------ ----------------------------------------
----------
 Gen   Length          Fitness   Length          Fitness      OOB Fitness  Time
Left
   0   263.65      1.83344e+82       26          7096.47              N/A
3.87m
   1   143.20      4.52488e+18       37          6423.63              N/A
1.97m
   2   125.76      3.17714e+28       36          2981.26              N/A
1.68m
   3    37.19      1.00884e+15       35          1357.48              N/A
46.25s
   4    33.24      3.00561e+11       36          1348.53              N/A
43.14s
   5    36.24       3.6604e+12       29          1347.86              N/A
42.24s
   6    33.99      5.00527e+11       53          1340.74              N/A
49.27s
   7    32.79      1.00247e+11       32          1337.52              N/A
38.62s
   8    39.85      3.02114e+13       67          1333.78              N/A
46.22s
   9    41.60      2.04272e+11       80          1330.71              N/A
41.00s
  10    51.43       3.0042e+11       87           1327.7              N/A
46.35s
  11    67.91      1.17323e+10       56          1321.53              N/A
55.39s
  12    84.75      3.00497e+11       74          1315.42              N/A
1.02m
  13   117.59      4.38221e+11      175          1313.81              N/A
1.36m
  14   134.62      4.42353e+11       78          1306.47              N/A
1.59m
  15    93.23      6.40904e+16       70          1305.42              N/A
1.08m
  16    93.54      4.18069e+11       82           1292.6              N/A
1.14m
  17    83.26      2.36324e+11      105          1292.17              N/A
1.10m
  18    95.85       3.0474e+11       97          1285.02              N/A
1.04m
  19    97.38      2.59852e+10       65           1283.7              N/A
1.37m
  20    97.62      2.37122e+11       92          1280.18              N/A
1.03m
  21    97.70      4.16481e+11      104           1277.5              N/A
1.31m
```

```
  22   108.00      4.15384e+11      108          1270.33              N/A
1.30m
  23   119.44      5.75953e+11      107          1270.26              N/A
1.40m
  24   122.91      7.85085e+11      189           1265.6              N/A
1.10m
  25   117.03      2.06137e+11      289           1262.8              N/A
1.13m
  26   161.66       3.8515e+13      125           1258.4              N/A
1.30m
  27   208.47      3.54604e+10      179          1240.85              N/A
1.76m
  28   214.05      2.08075e+11      169          1240.86              N/A
1.54m
  29   193.55      1.16327e+11      187          1240.58              N/A
1.95m
  30   282.15       1.4358e+11      171          1237.85              N/A
1.92m
  31   202.84      4.27824e+11      169          1236.93              N/A
1.65m
  32   193.22      1.34157e+11      205          1235.69              N/A
1.15m
  33   186.32         2.08e+11      165          1227.65              N/A
1.49m
  34   195.75      3.61611e+11      273          1223.82              N/A
1.63m
  35   207.24      4.16854e+11      182          1215.98              N/A
1.48m
  36   200.67      1.48128e+11      187          1215.08              N/A
1.24m
  37   217.53      2.35494e+10      188          1213.41              N/A
1.24m
  38   188.83      3.12841e+10      285          1210.87              N/A
1.01m
  39   194.81      1.30725e+17      194          1208.82              N/A
1.28m
  40   239.22       1.7514e+10      301          1208.81              N/A
1.15m
  41   264.01      1.52692e+11      273          1207.24              N/A
1.26m
  42   220.34       1.9401e+11      234           1206.3              N/A
1.30m
  43   248.83      3.12576e+10      277          1203.89              N/A
1.02m
  44   265.90      1.23314e+11      277          1203.89              N/A
1.12m
  45   254.21      1.24033e+11      263          1202.06              N/A
58.05s
```

```
  46   259.05      2.33082e+11      246          1202.06              N/A
1.00m
  47   274.01      3.51011e+14      269          1200.61              N/A
54.18s
  48   238.53      1.37946e+11      283          1198.32              N/A
47.20s
  49   254.42      1.19972e+11      282          1198.32              N/A
1.10m
  50   250.23      6.59532e+09      289          1196.49              N/A
45.62s
  51   268.52      1.13479e+09      289          1196.16              N/A
46.43s
  52   327.28       1.1708e+11      385          1193.54              N/A
48.66s
  53   320.06       1.8292e+10      385          1193.54              N/A
51.41s
  54   357.94      1.63158e+10      652          1193.53              N/A
54.40s
  55   453.47      1.00109e+11      408          1191.62              N/A
52.91s
  56   485.86      9.71437e+09      420          1191.53              N/A
53.40s
  57   401.32      1.08283e+11      429          1191.14              N/A
45.21s
  58   420.28      5.76415e+09      431          1191.14              N/A
39.19s
  59   414.51      2.07525e+10      406          1190.02              N/A
37.46s
  60   421.65      4.04333e+11      406          1190.02              N/A
32.86s
  61   425.74      3.40622e+09      504          1189.98              N/A
28.43s
  62   444.41        3.729e+10      509          1189.98              N/A
27.50s
  63   435.44      7.55312e+09      418          1189.94              N/A
23.63s
  64   406.27      1.25908e+11      393          1188.65              N/A
17.92s
  65   406.30      1.10143e+11      431          1189.85              N/A
14.61s
  66   406.42      6.81453e+09      420          1188.81              N/A
10.32s
  67   398.40      1.04182e+11      424          1188.67              N/A
7.10s
  68   411.13      4.36312e+11      418          1188.26              N/A
3.56s
  69   435.89      1.07835e+11      440          1187.87              N/A
0.00s
```

```
[ ]: SymbolicRegressor(const_range=(-100, 100),
                       function_set=['add', 'sub', 'mul', 'div', 'sin'],
                       generations=70, init_depth=(4, 10), metric='mse',
                       population_size=500, random_state=0, stopping_criteria=0.01,
                       verbose=1)
```

```
[ ]: print(est_gp._program)
```

```
add(sub(sub(div(X0, 28.307), div(sub(div(add(div(div(mul(X0, X0), sub(37.158,
-19.620)), add(sin(-38.730), add(sin(-38.730), sub(X0, X0)))), sin(-38.730)),
div(X0, 28.307)), sub(37.158, -19.620)), X0)), sub(add(sin(-38.730),
add(sin(-38.730), sub(X0, X0))), sub(37.158, -19.620))), div(sub(mul(X0,
-11.576), sub(X0, X0)), add(div(div(mul(X0, X0), add(sub(sub(sub(sub(sub(div(X0,
28.307), div(X0, X0)), sub(div(X0, X0), sub(37.158, -19.620))), -19.620),
-19.620), -19.620), div(add(div(div(mul(X0, X0), add(sub(sub(div(X0, 28.307),
div(X0, X0)), sub(div(X0, X0), sub(37.158, -19.620))), div(sub(mul(X0, -11.576),
sub(37.158, -19.620)), add(div(div(mul(X0, X0), add(sub(37.158, -19.620),
div(add(div(div(mul(X0, X0), add(37.158, sub(div(X0, 28.307), sub(X0, X0)))),
add(sin(-38.730), add(sin(-38.730), sub(X0, X0)))), sub(sub(37.158, -19.620),
-19.620)), div(X0, 28.307)))), add(-19.620, add(sin(-38.730), sub(X0, X0)))),
div(sub(sin(div(sub(div(X0, X0), sub(37.158, -19.620)), X0)), sub(37.158,
add(div(div(mul(X0, X0), add(sub(sub(div(X0, 28.307), div(X0, X0)), sub(div(X0,
X0), sub(37.158, -19.620))), div(sub(mul(X0, -11.576), sub(sub(div(X0, 28.307),
div(add(div(div(mul(X0, X0), sub(37.158, -19.620)), sub(sub(div(X0, 28.307),
sub(sub(div(X0, 28.307), div(sub(div(X0, X0), sub(37.158, -19.620)), X0)),
sub(div(sub(X0, X0), X0), sub(37.158, -19.620)))), sub(div(X0, X0), sub(37.158,
-19.620)))), sin(-38.730)), div(X0, 28.307))), sub(div(X0, X0), sub(37.158,
-19.620)))), add(div(div(mul(X0, X0), add(sub(37.158, -19.620),
div(add(div(div(mul(X0, X0), sub(37.158, -19.620)), add(sin(-38.730),
add(sin(-38.730), sub(X0, X0)))), sin(-38.730)), div(X0, 28.307)))),
add(-19.620, add(sin(-38.730), sub(X0, X0)))), div(sub(sin(sub(X0, X0)),
sub(37.158, add(div(div(mul(X0, X0), add(sub(sub(div(X0, 28.307), div(X0, X0)),
sub(div(X0, X0), sub(37.158, -19.620))), div(sub(mul(X0, -11.576), X0),
add(div(div(mul(X0, X0), add(sub(37.158, -19.620), div(add(div(div(mul(X0, X0),
sub(37.158, -19.620)), add(sin(-38.730), add(sin(-38.730), sub(X0, X0)))),
sin(-38.730)), div(X0, 28.307)))), add(-19.620, add(sin(-38.730), sub(X0,
X0)))), div(sub(X0, X0), X0))))), add(sin(-38.730), sub(X0, X0))), X0))),
X0))))), add(sin(-38.730), sub(X0, X0))), X0))), X0))))), add(sin(-38.730),
sub(X0, X0))), X0), div(X0, 28.307)))), add(sin(-38.730), add(sin(-38.730),
sub(X0, X0)))), div(sub(sin(sub(X0, X0)), sub(37.158, -19.620)), X0))))
```

```
[ ]: y_gp = est_gp.predict(np.array(xnum_test).reshape(-1, 1))
     y_gp[:10]
```

```
[ ]: array([197.02008255, 197.05604924, 197.09201562, 197.1279817 ,
            197.16394748, 197.19991296, 197.23587813, 197.271843  ,
            197.30780756, 197.34377183])
```

[ ]:
```
ts_coffee2['predictions_GPLEARN'] = (train_size * [np.NAN]) + list(y_gp)
```

[ ]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(20,20))
fig.suptitle('                    (          )')
ts_coffee2[train_size:].plot(ax=ax, legend=True)
pyplot.show()
```