Untitled4

September 25, 2020

[311]:
```
import os
import math
import warnings
import seaborn as sns
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
from pandas.plotting import lag_plot
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import statsmodels.stats as sms
import statsmodels.api as sm
from scipy.stats import norm
from numpy.random import normal, seed
from statsmodels.tsa.arima_model import ARMA
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima_process import ArmaProcess
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
```

[312]:
```
def read_data(csv_file):
    try:
        return pd.read_csv(csv_file, index_col='Date', parse_dates=['Date'])
    except:
        print("The file is not found")
        return None

stock_data_set = read_data("C:/Users/omri1/PycharmProjects/untitled2/prices.
 ↪csv")
```

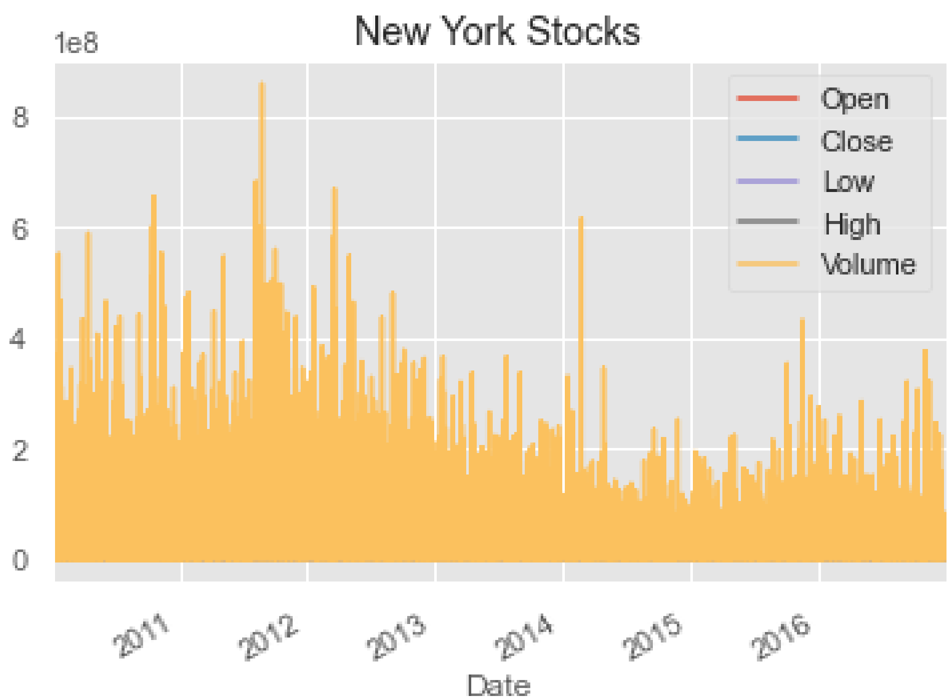
[313]:
```
stock_data_set.head()
```

[313]:
```
           Symbol       Open      Close        Low       High   Volume
Date
2010-04-01      A  31.389999  31.300001  31.130000  31.630001  3815500
```

2010-04-01	AAL	4.840000	4.770000	4.660000	4.940000	9837300
2010-04-01	AAP	40.700001	40.380001	40.360001	41.040001	1701700
2010-04-01	AAPL	213.429998	214.009998	212.380001	214.499996	123432400
2010-04-01	ABC	26.290001	26.629999	26.139999	26.690001	2455900

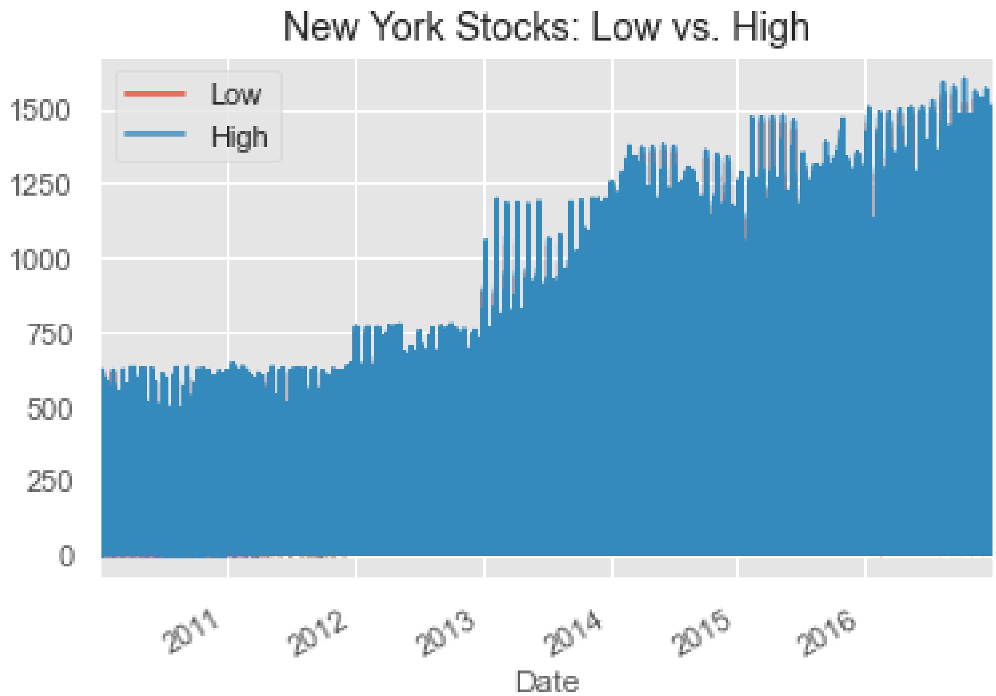
[314]:
```
stock_data_set.plot(title="New York Stocks")
```

[314]: <AxesSubplot:title={'center':'New York Stocks'}, xlabel='Date'>

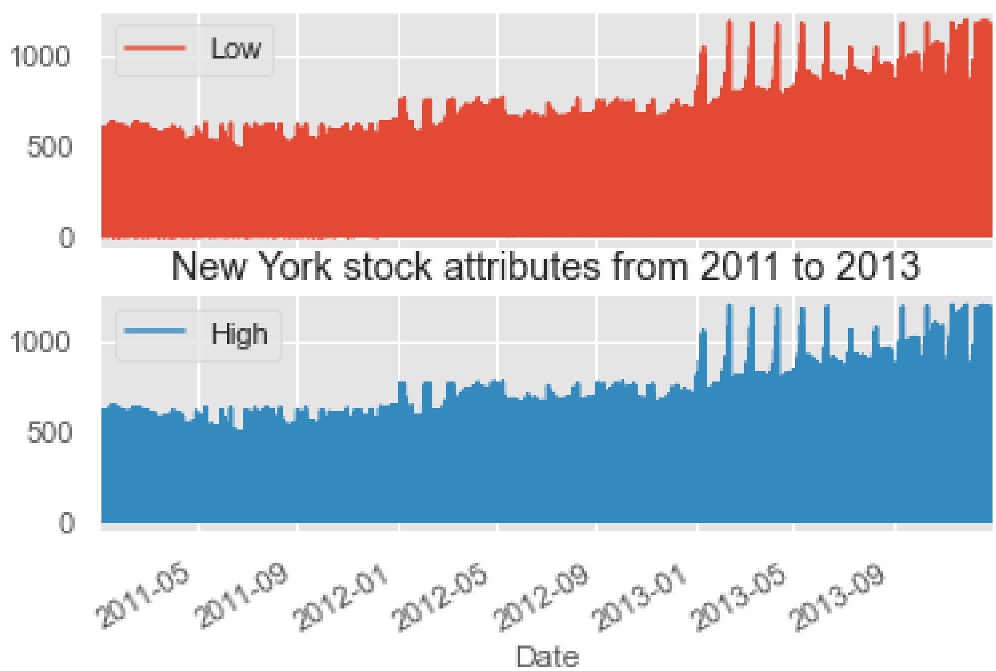


[315]:
```
stock_data_set[["Low", "High"]].plot(title="New York Stocks: Low vs. High")
```

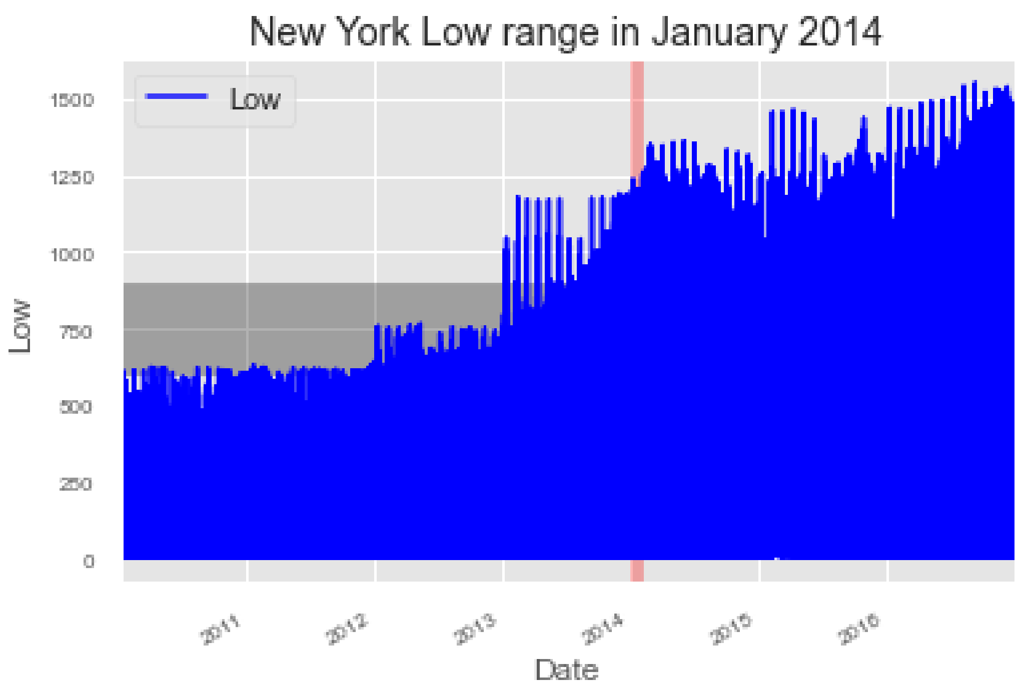
[315]: <AxesSubplot:title={'center':'New York Stocks: Low vs. High'}, xlabel='Date'>




```
[316]: stock_data_set['2011':'2013'][["Low", "High"]].plot(subplots=True) # split the␣
        ↪columns to different plots
plt.title('New York stock attributes from 2011 to 2013')
plt.show()
```

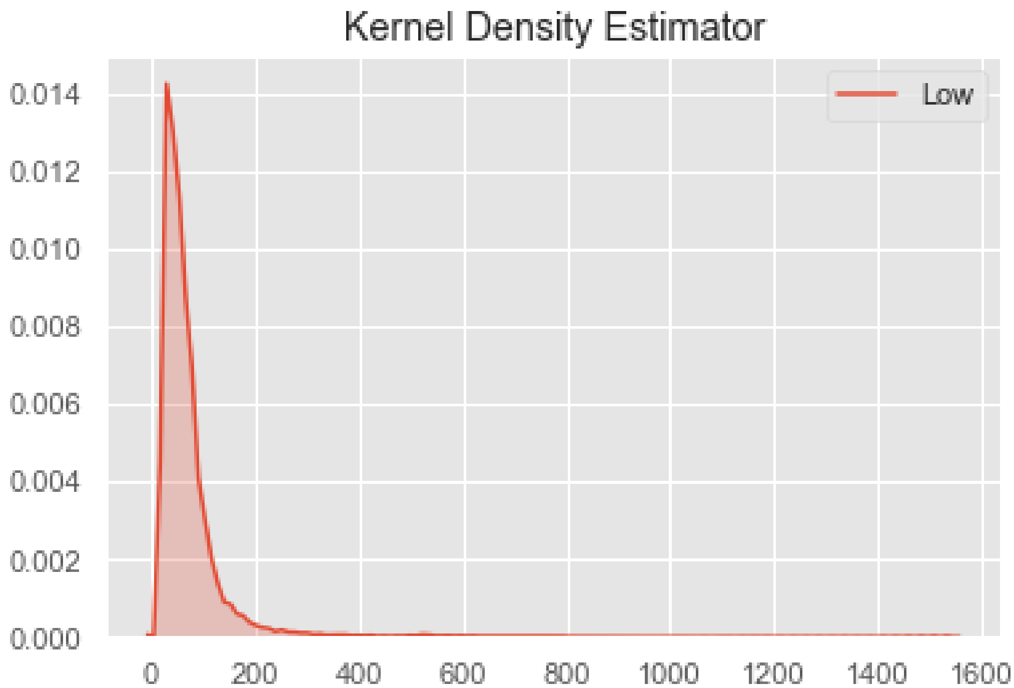


[317]:
```
ax = stock_data_set[["Low"]].plot(color='blue',fontsize=8)
ax.set_xlabel('Date')
ax.set_ylabel('Low')
# add markers
ax.axvspan('2014-01-01','2014-01-31', color='red', alpha=0.3)
ax.axhspan(600, 900, color='black',alpha=0.3)
plt.title("New York Low range in January 2014")
plt.show()
```



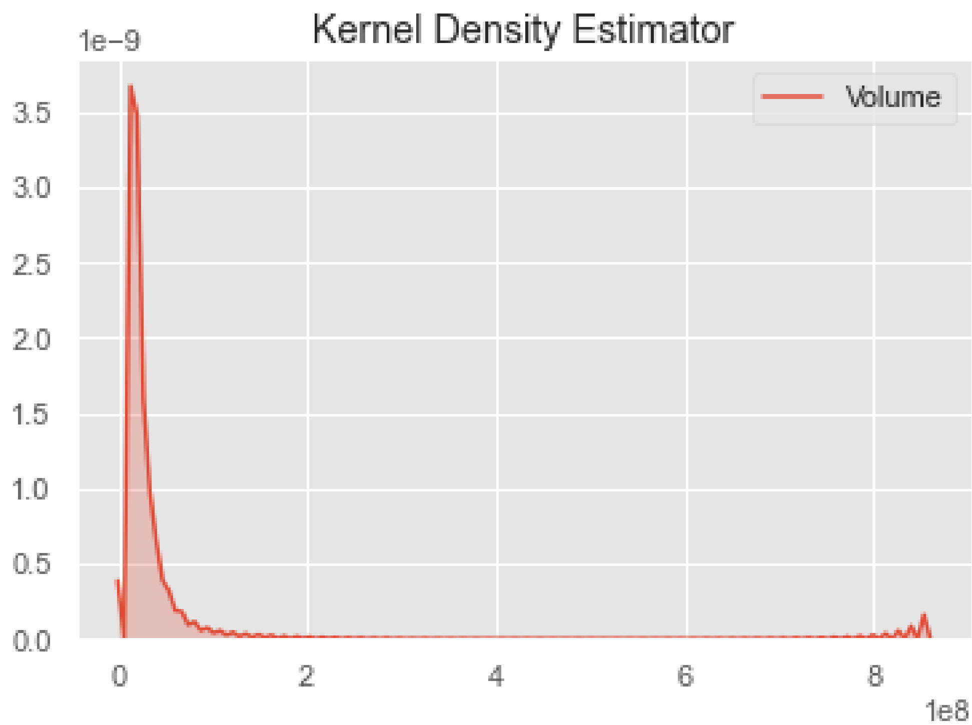
[318]:
```
sns.kdeplot(stock_data_set['Low'], shade=True)
plt.title("Kernel Density Estimator")
```

[318]: Text(0.5, 1.0, 'Kernel Density Estimator')

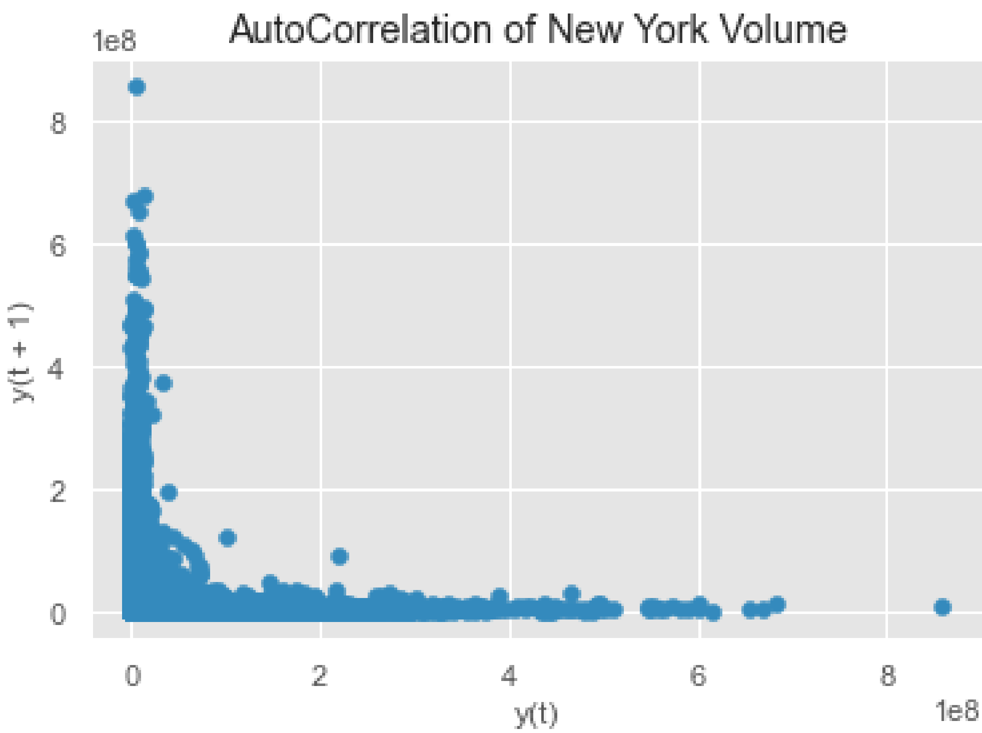


[319]:
```
sns.kdeplot(stock_data_set['Volume'], shade=True)
plt.title("Kernel Density Estimator")
```

[319]: Text(0.5, 1.0, 'Kernel Density Estimator')

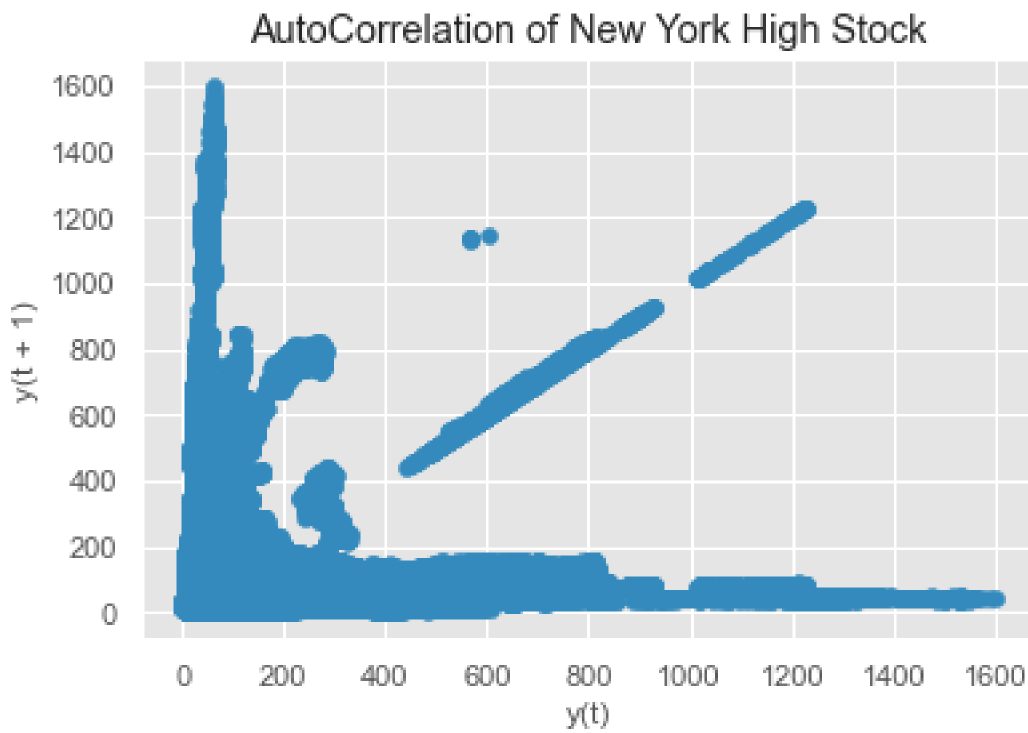


[320]:
```
lag_plot(stock_data_set["Volume"]) # lag plot is the dependency of Y(t+1) in␣
 ↪Y(t)
plt.title("AutoCorrelation of New York Volume")
plt.show()
```



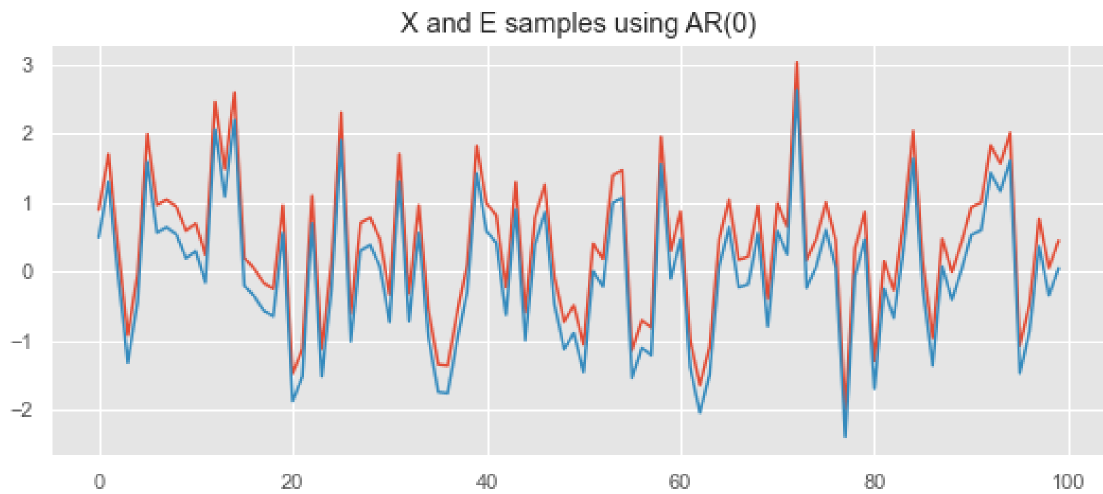
[321]:

```
lag_plot(stock_data_set["High"])
plt.title("AutoCorrelation of New York High Stock")
plt.show()
```



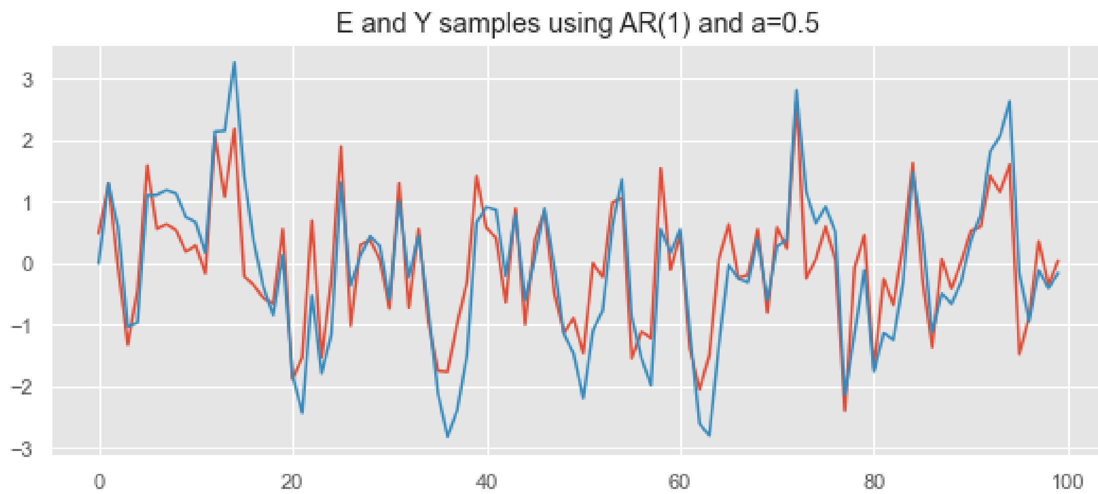
[322]:
```
# Examples for different autoregressive values, That is linearly dependtion on␣
↪its own previous values.

SAMPLES = 100

def ar_0(size, constant, noise):
    x = np.zeros(size)
    for i in range(size):
        x[i] = constant + noise[i]
    return x

e = np.random.randn(SAMPLES)
x = ar_0(SAMPLES, 0.4, e)

plt.figure(figsize=(10, 4))
plt.plot(range(SAMPLES), x, label="x")
plt.plot(range(SAMPLES), e, label="e")
plt.title("X and E samples using AR(0)")
```

[322]:
```
Text(0.5, 1.0, 'X and E samples using AR(0)')
```



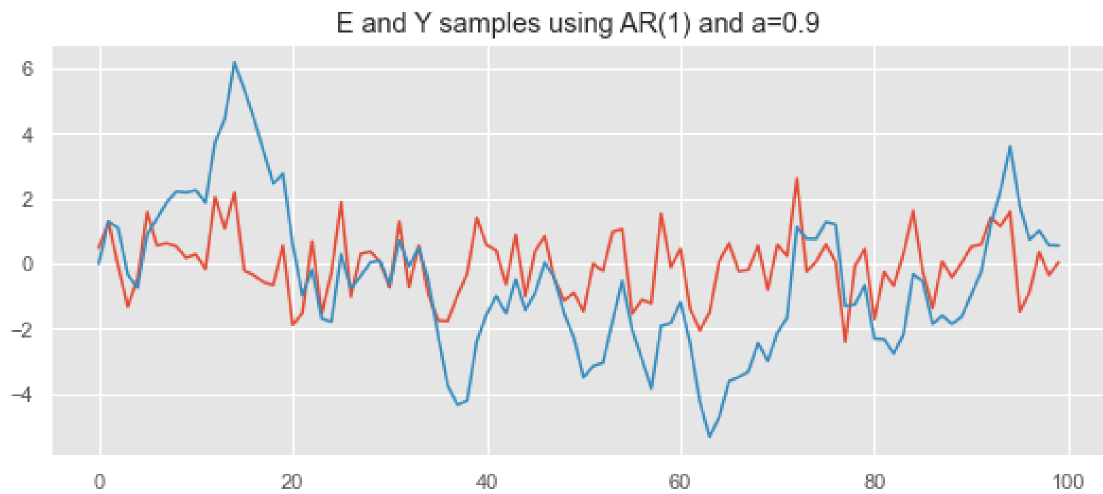
[323]:
```
def ar_1(size, p, constant, noise):
    x = np.zeros(size)
    for i in range(p, SAMPLES):
        x[i] = constant[0] * x[i-1] + e[i]
    return x

a = [0.5]
p = len(a)
y = ar_1(SAMPLES, len(a), a, e)

plt.figure(figsize=(10, 4))
plt.plot(range(SAMPLES), e, label="e")
plt.plot(range(SAMPLES), y, label="y")
plt.title("E and Y samples using AR(1) and a=0.5")
```

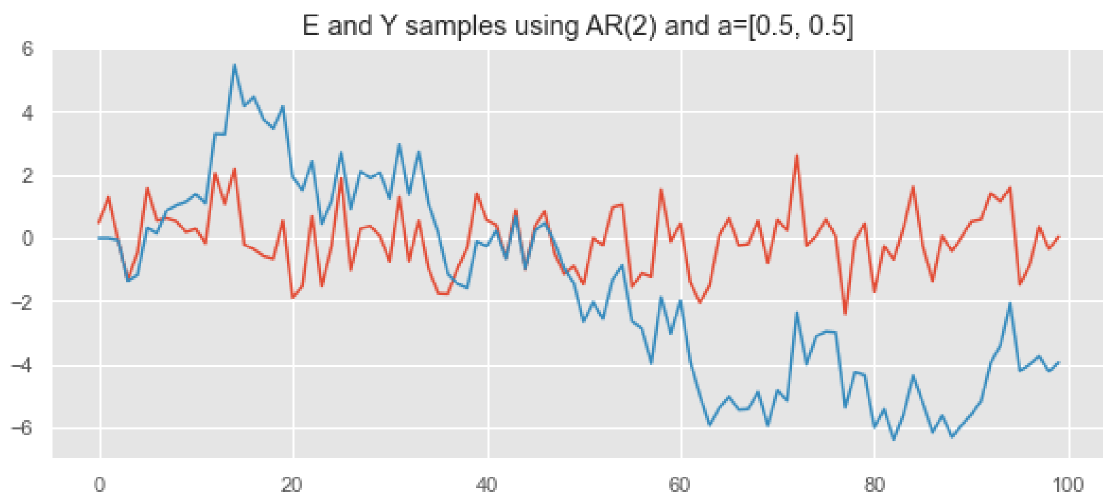
[323]: Text(0.5, 1.0, 'E and Y samples using AR(1) and a=0.5')




```
[324]: a = [0.9]
       y = ar_1(SAMPLES, len(a), a, e)
       plt.figure(figsize=(10, 4))
       plt.plot(range(SAMPLES), e, label="e")
       plt.plot(range(SAMPLES), y, label="y")
       plt.title("E and Y samples using AR(1) and a=0.9")

       def ar_2(size, p, constant, noise):
           x = np.zeros(size)
           for i in range(p, SAMPLES):
               x[i] = constant[0]*x[i-2] + constant[1]*x[i-1] + e[i]
           return x

       a = [0.5, 0.5]
       y = ar_2(SAMPLES, len(a), a, e)
       plt.figure(figsize=(10, 4))
       plt.plot(range(SAMPLES), e, label="e")
       plt.plot(range(SAMPLES), y, label="y")
       plt.title("E and Y samples using AR(2) and a=[0.5, 0.5]")
```

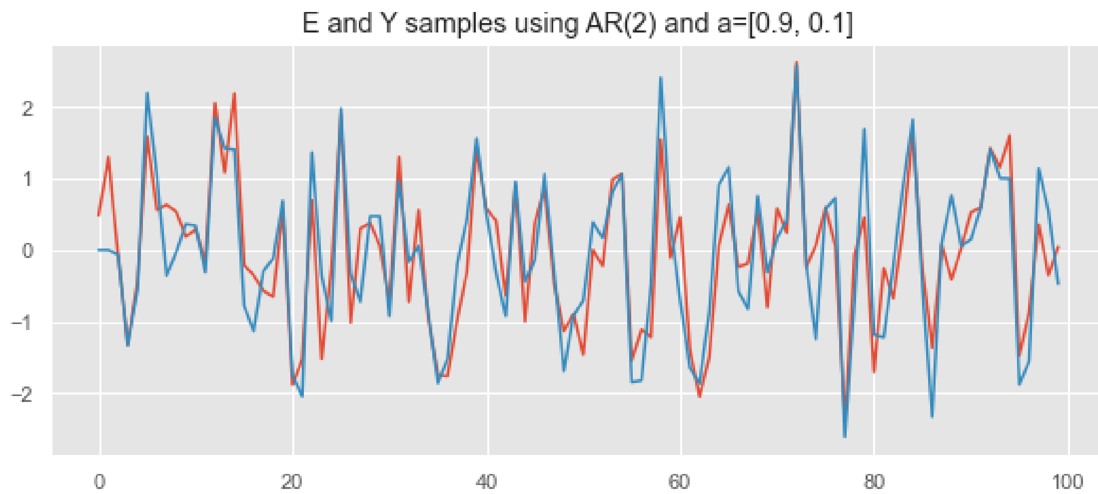
```
[324]: Text(0.5, 1.0, 'E and Y samples using AR(2) and a=[0.5, 0.5]')
```





[325]:
```
a = [-0.5, 0.1]
y = ar_2(SAMPLES, len(a), a, e)
plt.figure(figsize=(10, 4))
plt.plot(range(SAMPLES), e, label="e")
plt.plot(range(SAMPLES), y, label="y")
plt.title("E and Y samples using AR(2) and a=[0.9, 0.1]")
```

[325]:
```
Text(0.5, 1.0, 'E and Y samples using AR(2) and a=[0.9, 0.1]')
```



[326]:
```
def moving_average(numbers, N):
    i = 0
    moving_averages = []
    while i < len(numbers) - N + 1: # the chunk of last N observations
        N_tag = numbers[i : i + N]
        window_average = sum(N_tag) / N
        moving_averages.append(window_average)
        i += 1
    return moving_averages

moving_average([1, 2, 4, 5, 7, 9], 3)
```

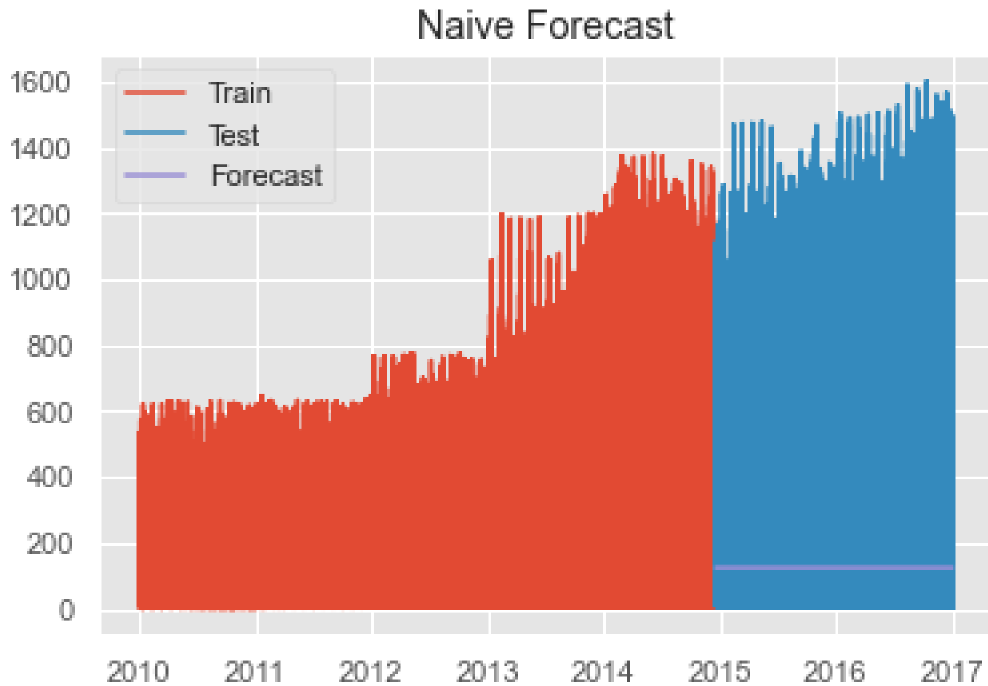
[326]: [2.3333333333333335, 3.6666666666666665, 5.333333333333333, 7.0]

[327]:
```
moving_average([1, 1, 1, 1, 1, 1], 3)
```

[327]: [1.0, 1.0, 1.0, 1.0]

[328]:
```
# The Naive Algorithm

X = stock_data_set["High"]
splitter = int(len(X) * 0.7)
train, test = X[:splitter], X[splitter:]

g_high = train.to_numpy()
plt.plot(train.index, train, label='Train')
plt.plot(test.index, test, label='Test')
plt.plot(test.index, [train[len(train)-1]] * len(test), label="Forecast")
plt.legend(loc='best')
plt.title("Naive Forecast")
```

```
plt.show()
```



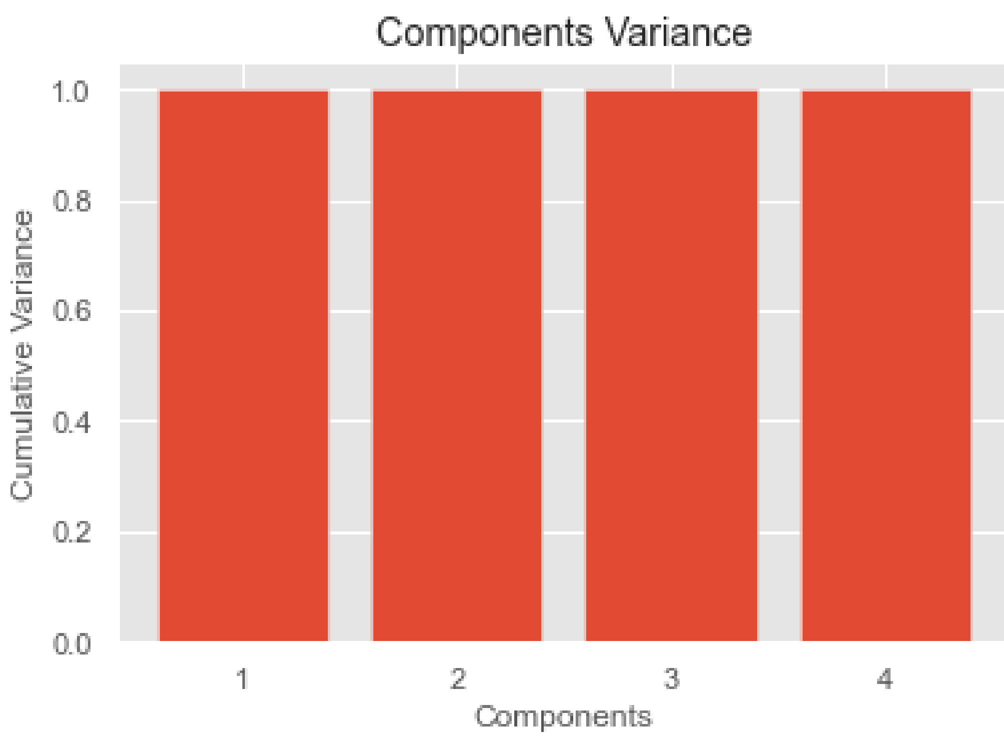
[335]:
```
# PCA
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

stock_data_set
```

[335]:
```
           Symbol        Open       Close         Low        High
Date
2010-04-01      A   31.389999   31.300001   31.130000   31.630001
2010-04-01    AAL    4.840000    4.770000    4.660000    4.940000
2010-04-01    AAP   40.700001   40.380001   40.360001   41.040001
2010-04-01   AAPL  213.429998  214.009998  212.380001  214.499996
2010-04-01    ABC   26.290001   26.629999   26.139999   26.690001
…             …          …          …          …          …
2016-12-30    ZBH  103.309998  103.199997  102.849998  103.930000
2016-12-30   ZION   43.070000   43.040001   42.689999   43.310001
2016-12-30    ZTS   53.639999   53.529999   53.270000   53.740002
2016-12-30    AIV   44.730000   45.450001   44.410000   45.590000
2016-12-30    FTV   54.200001   53.630001   53.389999   54.480000

[851013 rows x 5 columns]
```

[340]:
```
stock_data_set.drop(columns=["Symbol"], inplace=True)

sc = StandardScaler()
normalized_data = sc.fit_transform(stock_data_set)
pca = PCA()
pca_data = pca.fit_transform(normalized_data)
```

[341]:
```
plt.bar(range(1, len(pca.explained_variance_ratio_)+1), np.cumsum(pca.
 ↪explained_variance_ratio_))
plt.xlabel('Components')
plt.ylabel('Cumulative Variance')
plt.xticks(range(1, len(pca.explained_variance_ratio_)+1))
plt.title("Components Variance")
plt.plot()
```

[341]: []



[342]:
```
pd.DataFrame({
"Variance": pca.explained_variance_ratio_
}, index=range(1, len(pca.explained_variance_ratio_) + 1))
```

[342]:

	Variance
1	0.999930

```
2  0.000038
3  0.000027
4  0.000005
```

[343]:

```
# using components = 1
pca = PCA(n_components=1)
pca_data = pca.fit_transform(normalized_data)
components = pd.DataFrame(pca.components_, columns = stock_data_set.columns)
components
```

[343]:

	Open	Close	Low	High
0	0.499997	0.499998	0.500002	0.500004

[344]:

```
pd.DataFrame(pca_data).plot(title="1D New York Stock", figsize=(10,5))
```

[344]:

```
<AxesSubplot:title={'center':'1D New York Stock'}>
```