New York Stock

September 28, 2020

[1]:
```
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
```

[2]:
```
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")
```

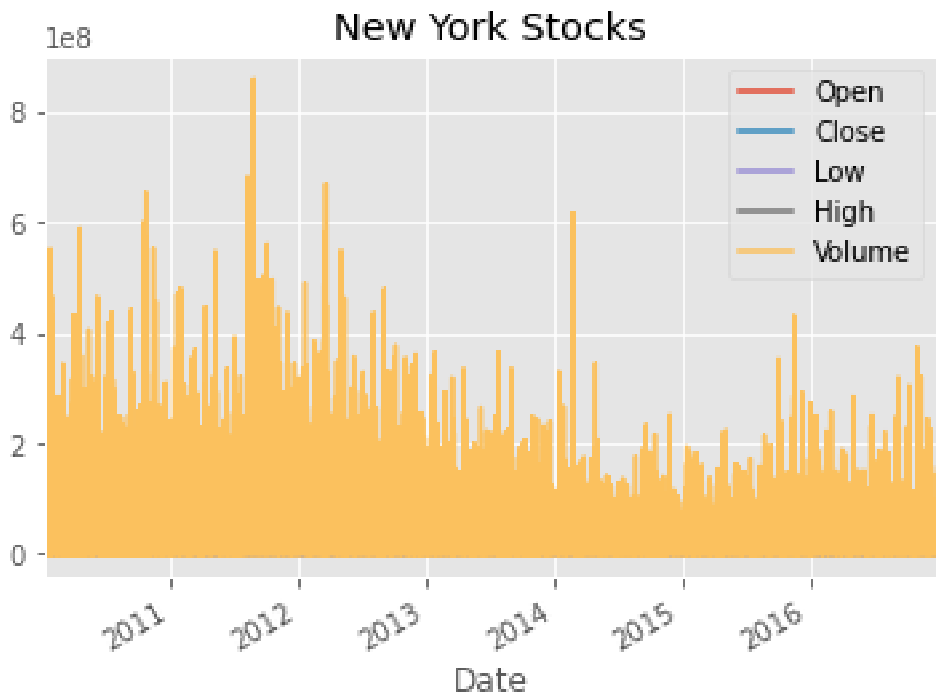
[3]:
```
stock_data_set.head()
```

[3]:
```
           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

[4]:
```
stock_data_set.plot(title="New York Stocks")
```

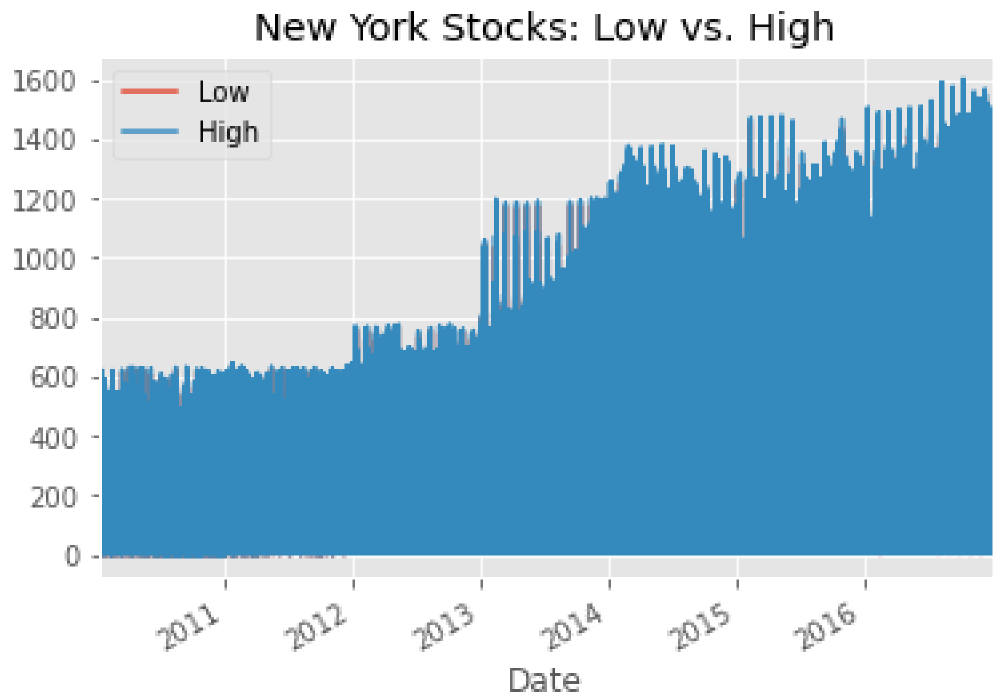
[4]: <AxesSubplot:title={'center':'New York Stocks'}, xlabel='Date'>



[ ]:
```
# The volume was pretty high between 2010 to 2013, with a peak on the end of␣
↪2011.
# After then the volume decreases and flats till the end of 2016.
```

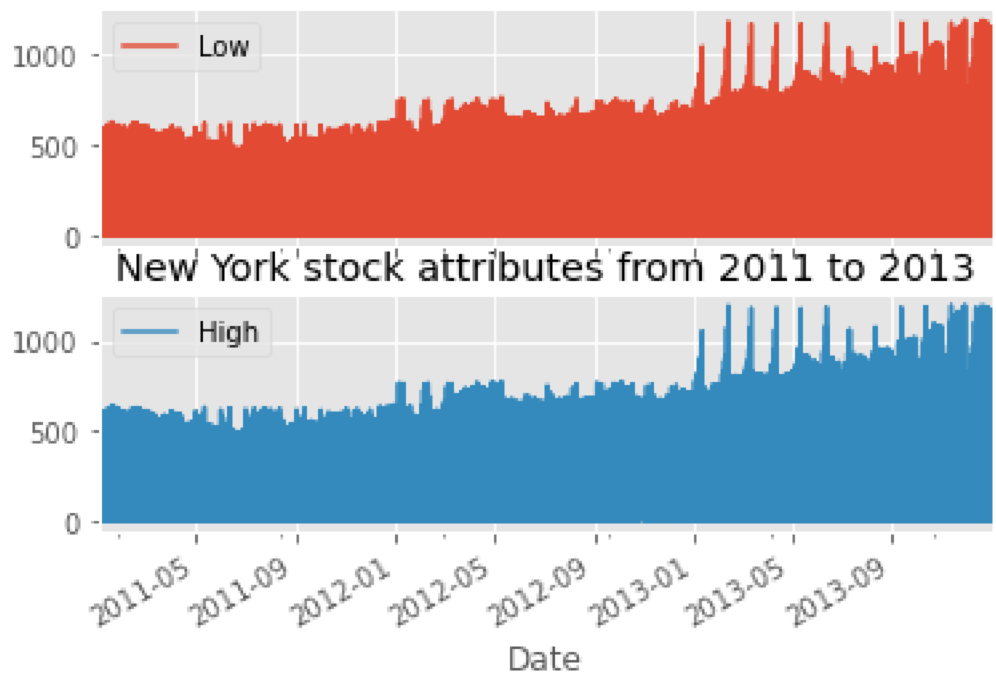
[5]:
```
stock_data_set[["Low", "High"]].plot(title="New York Stocks: Low vs. High")
```

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



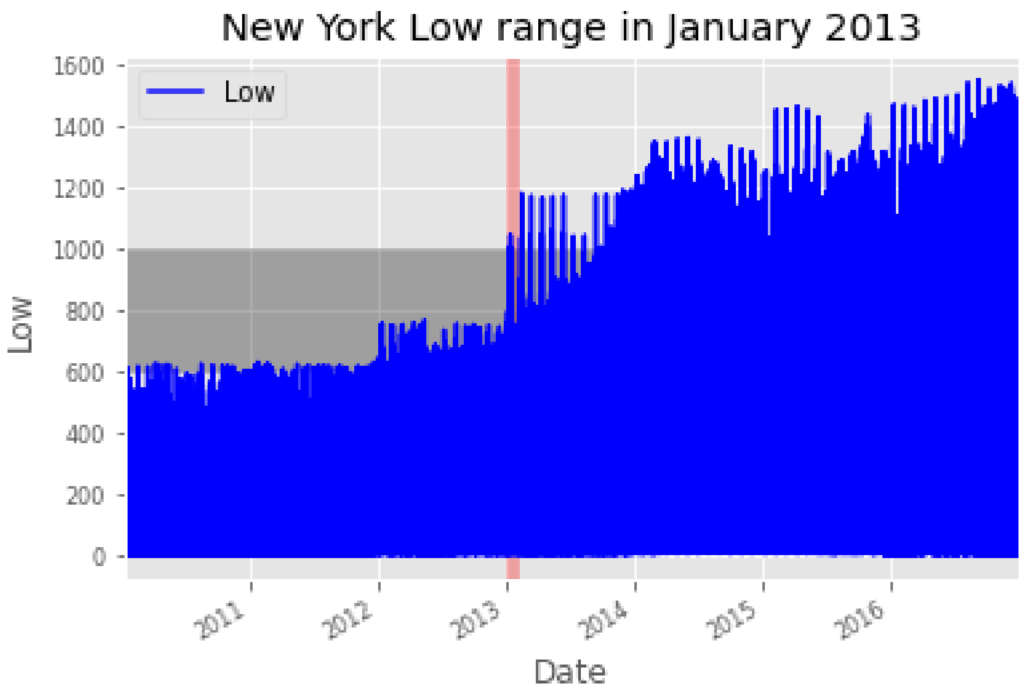

```
[ ]: # As we can see in the plot, all over the years the lowest price and the␣
     ↪highest price of the stock are very close.
     # According to this, we can tell the New York stock is very stable.
```

```
[6]: 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()
```



[ ]:
```
# Those plots are going hand to hand with the previous conclusion.
# For both plots have the same movements with the almost exact highness.
```

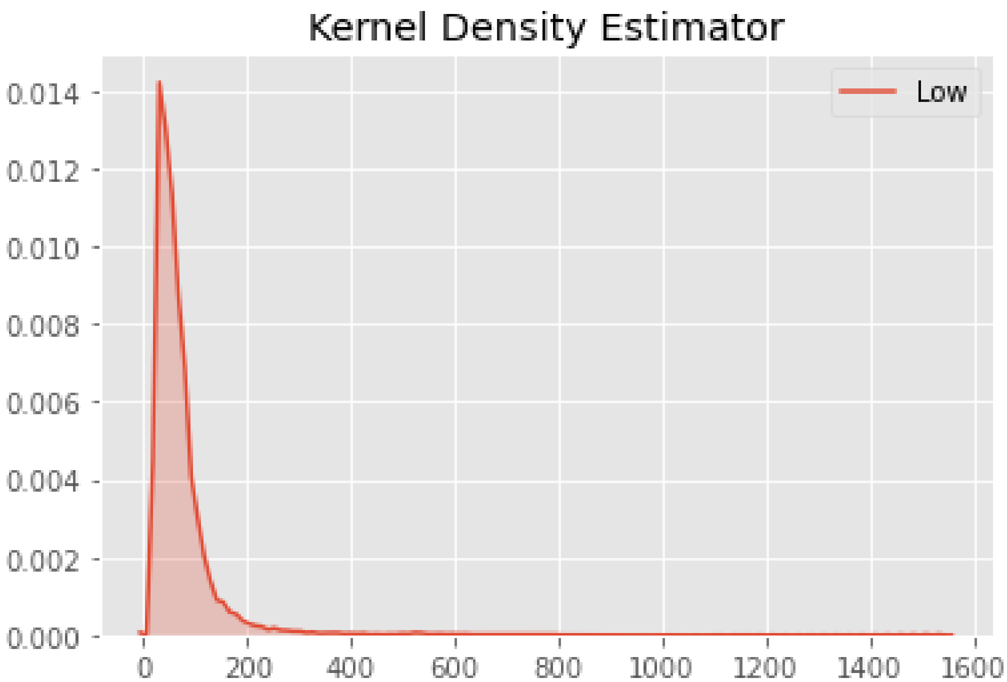
[26]:
```
ax = stock_data_set[["Low"]].plot(color='blue',fontsize=8)
ax.set_xlabel('Date')
ax.set_ylabel('Low')
# add markers
ax.axvspan('2013-01-01','2013-01-31', color='red', alpha=0.3)
ax.axhspan(600, 1000, color='black',alpha=0.3)
plt.title("New York Low range in January 2013")
plt.show()
```



[ ]:
```
# In January 2013, we can see the beginning of the acceleration in the New York␣
↪stock.
# On this year the stock price has jumped from 600 USD to 1000 USD.
```

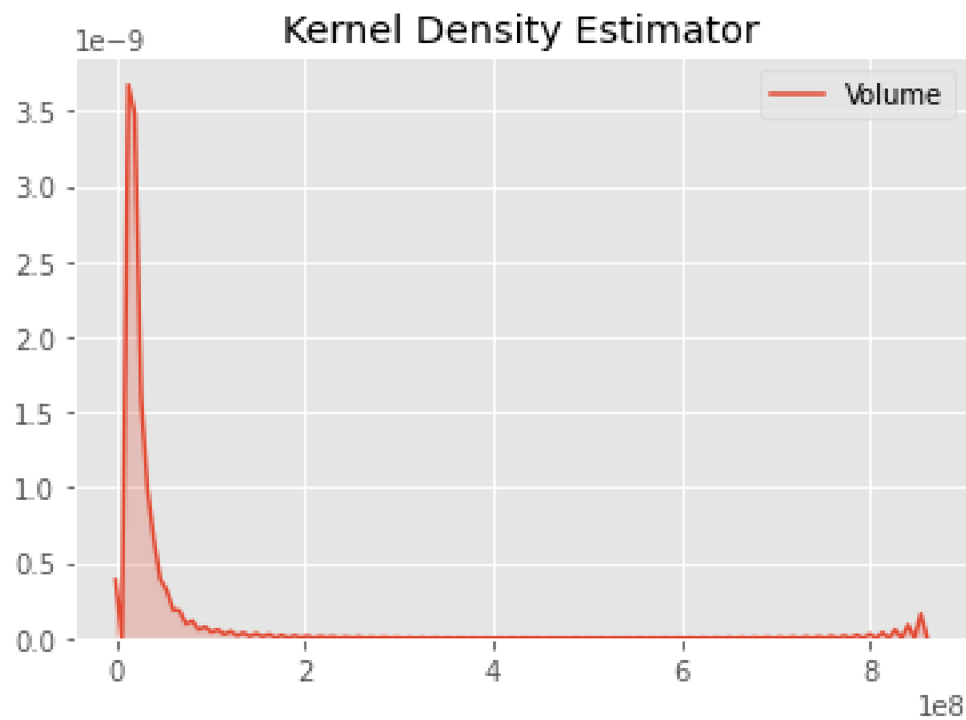
[8]:
```
sns.kdeplot(stock_data_set['Low'], shade=True)
plt.title("Kernel Density Estimator")
```

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



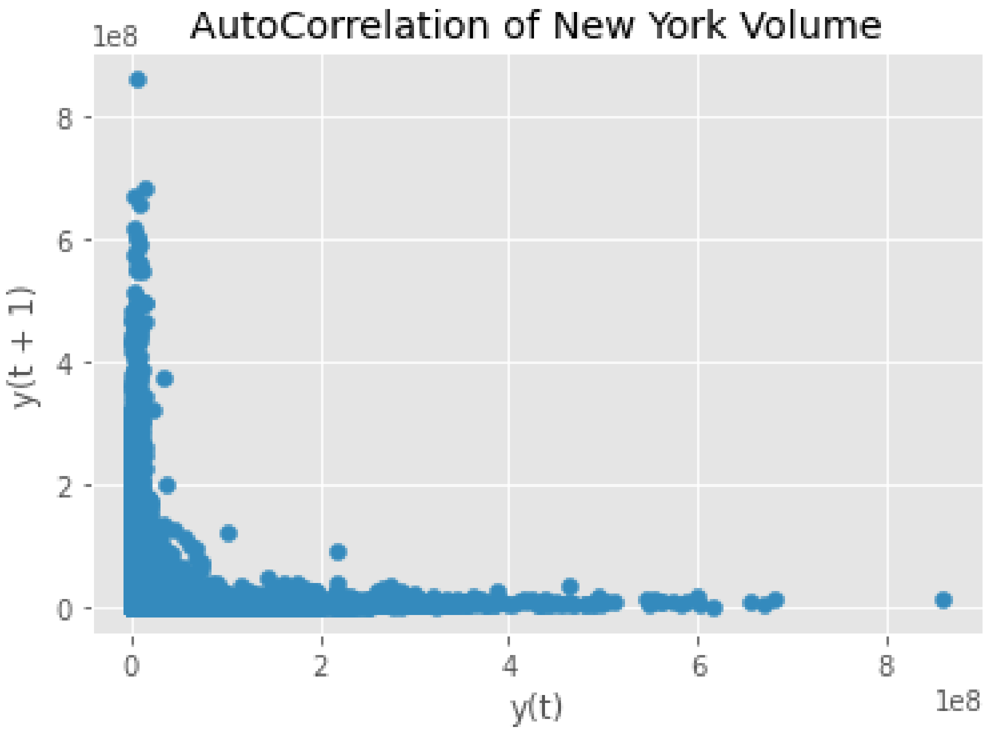
[9]:
```
sns.kdeplot(stock_data_set['Volume'], shade=True)
plt.title("Kernel Density Estimator")
```

[9]:
```
Text(0.5, 1.0, 'Kernel Density Estimator')
```

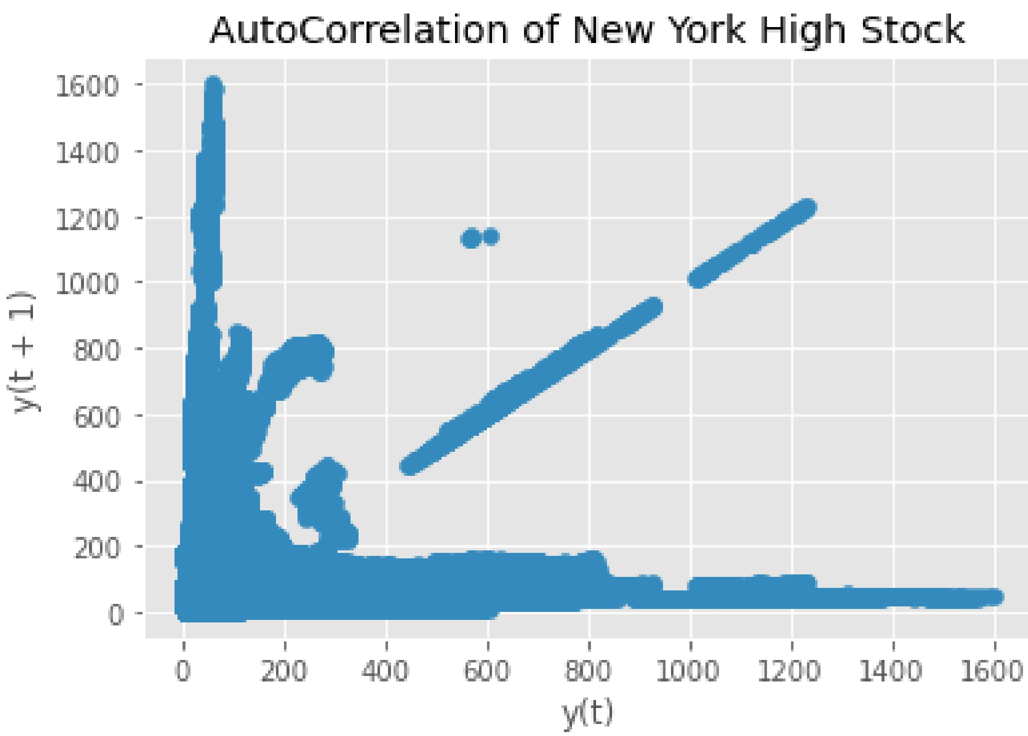



```
[ ]: # Those plots show where is the density point.
```

```
[10]: 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()
```



[11]:
```
lag_plot(stock_data_set["High"])
plt.title("AutoCorrelation of New York High Stock")
plt.show()
```

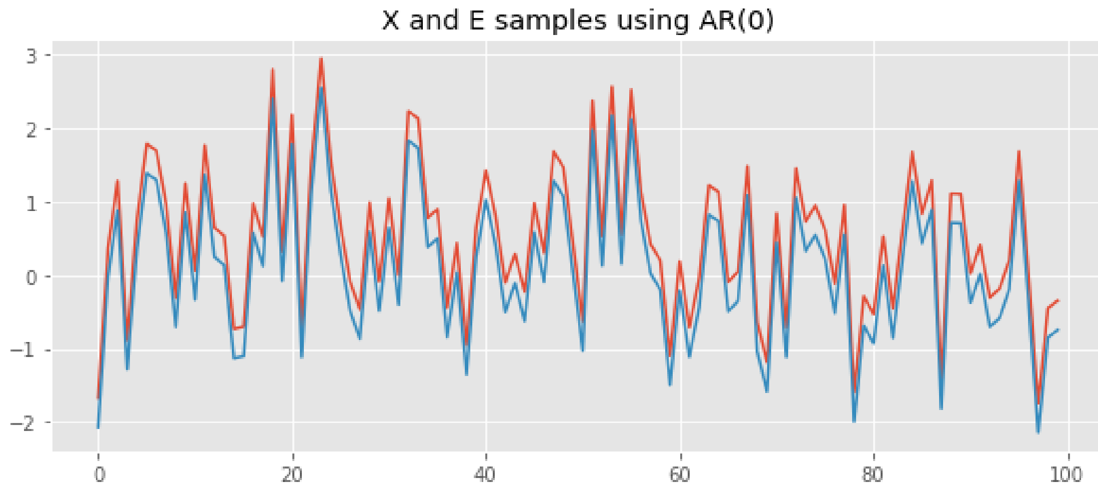


[ ]:
```
# The first Autocorellation plot is showing us that the volume of the stock is␣
↪not dependent on the previous day.
# From the second plot we can conclude, about the High price stock, that has a␣
↪kind of correlation between the previous day to the day come after.
# The strong correlation is begun when the price is about 400 USD.
```

[12]:
```
# 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")
```

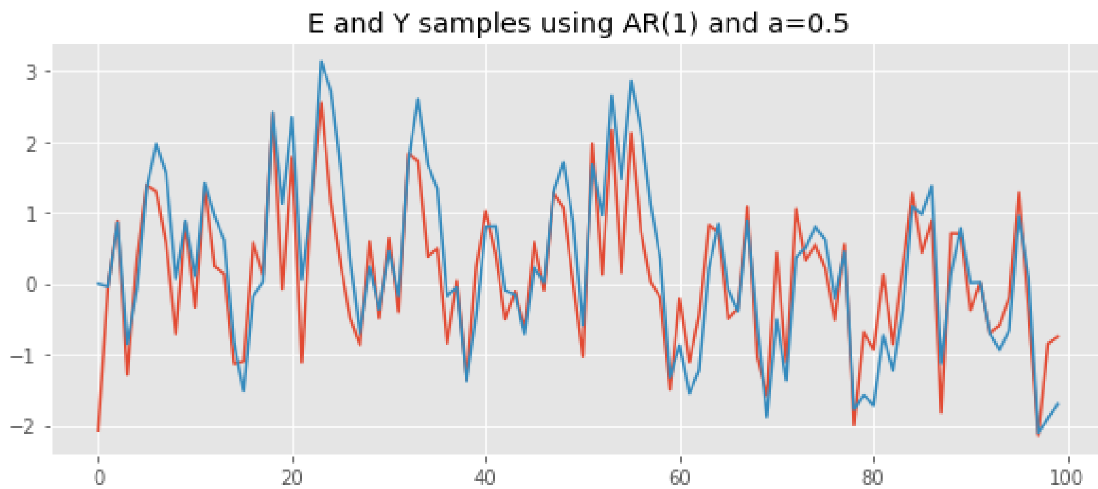
```
plt.title("X and E samples using AR(0)")
```

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



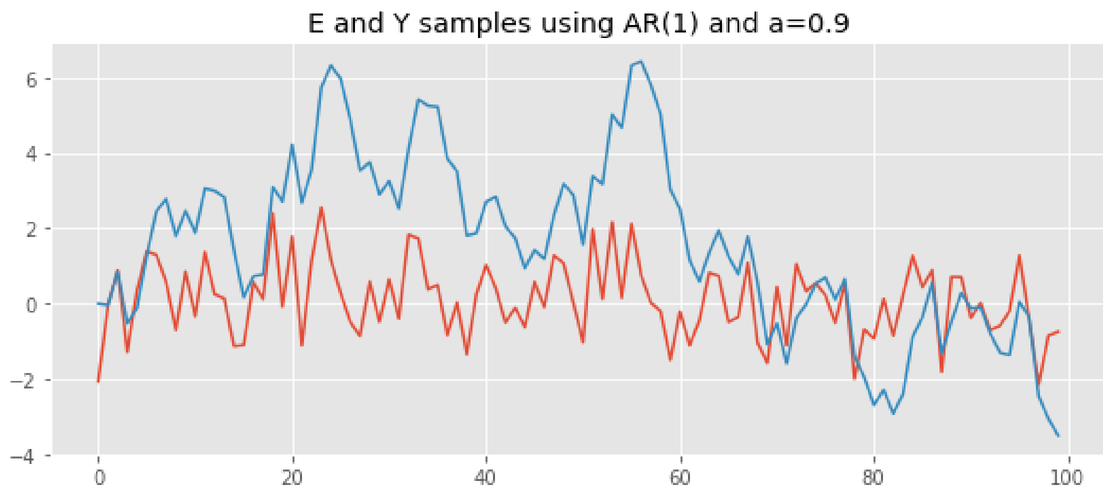
[13]:
```
def ar_1(size, p, constant, noise):
    x = np.zeros(size)
    for i in range(p, SAMPLES):
        x[i] = constant[0] * x[i-1] + e[i]
    return x

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")
```

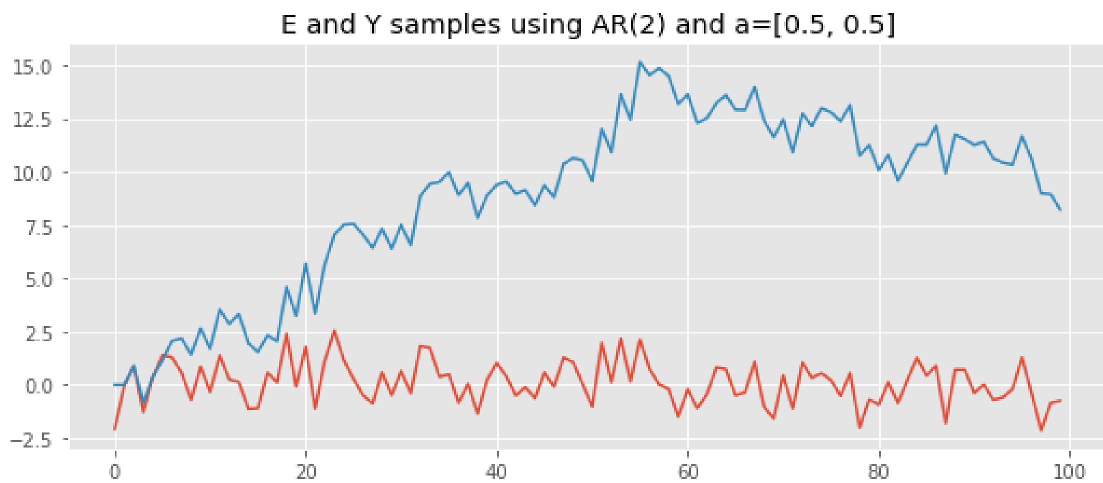
[13]: Text(0.5, 1.0, 'E and Y samples using AR(1) and a=0.5')



[14]:
```
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]")
```

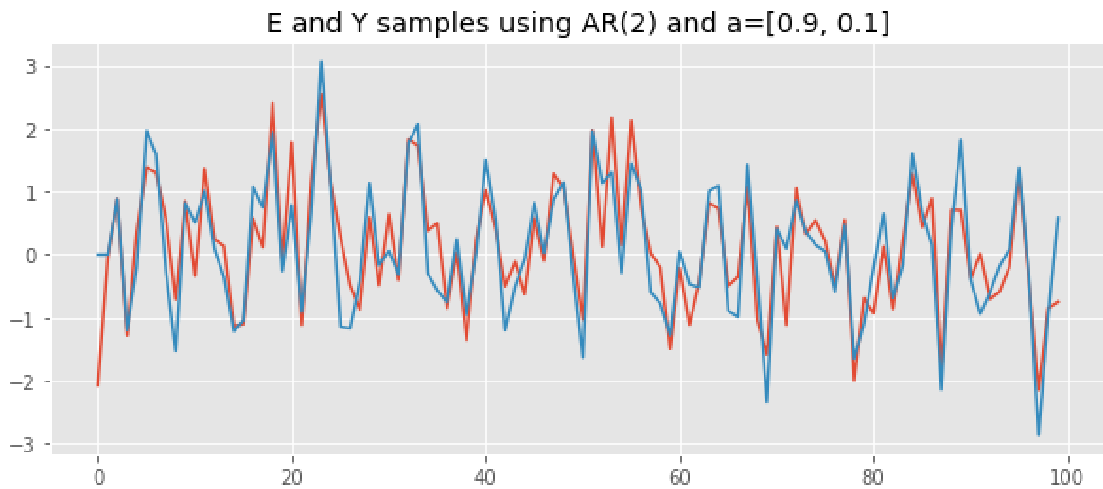
[14]:
```
Text(0.5, 1.0, 'E and Y samples using AR(2) and a=[0.5, 0.5]')
```





[15]:
```
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]")
```

[15]:
```
Text(0.5, 1.0, 'E and Y samples using AR(2) and a=[0.9, 0.1]')
```



[16]:
```
# Examples to forecast the future stock with the average of the n last numbers.

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)
```

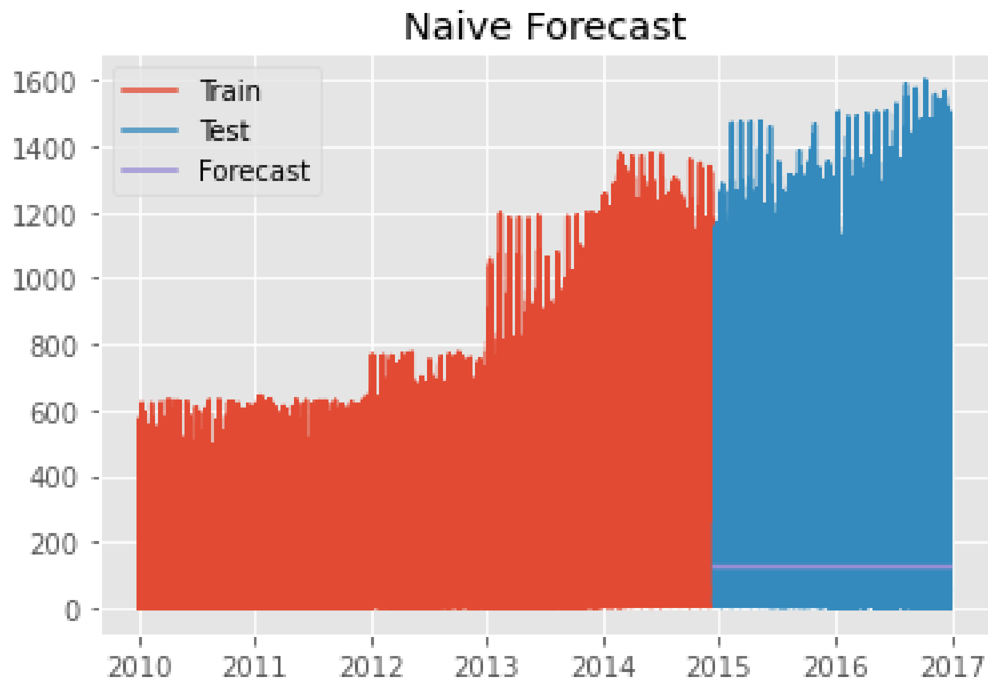
[16]:
```
[2.3333333333333335, 3.6666666666666665, 5.333333333333333, 7.0]
```

[17]:
```
moving_average([1, 1, 1, 1, 1, 1], 3)
```

[17]:
```
[1.0, 1.0, 1.0, 1.0]
```

[18]:
```
# 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()
```



[ ]:
```
# The result of the Naive forecast is showing the stock will go up consistently␣
↪from 2015 to 2017.
```

[19]:
```
# PCA
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

stock_data_set
```

[19]:

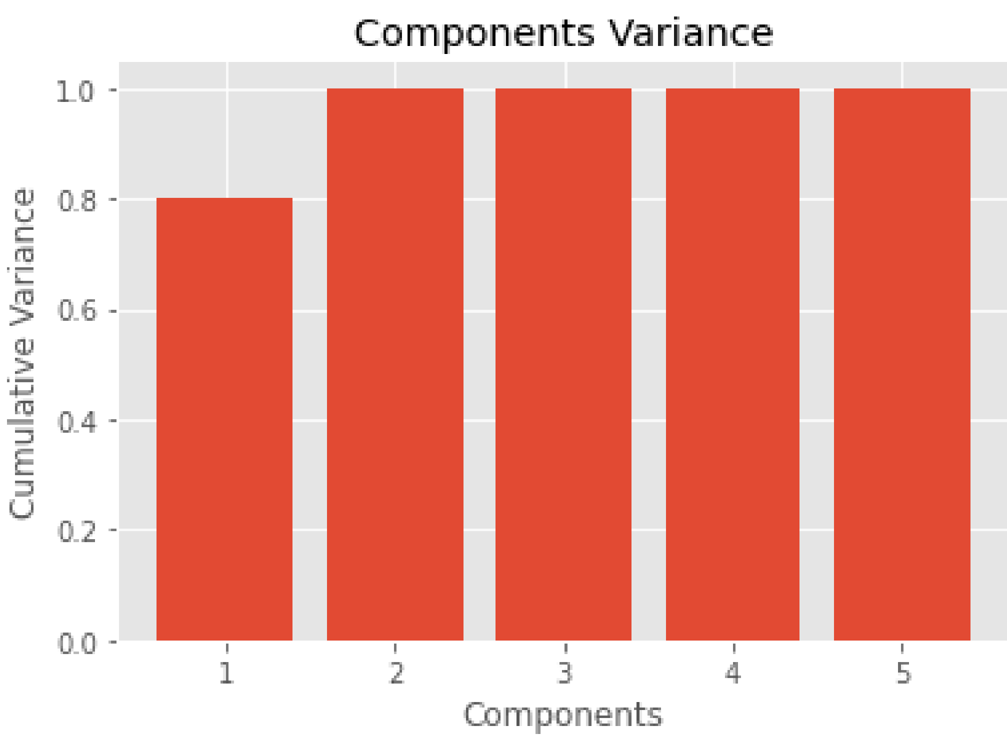
	Symbol	Open	Close	Low	High	Volume
Date						
2010-04-01	A	31.389999	31.300001	31.130000	31.630001	3815500
2010-04-01	AAL	4.840000	4.770000	4.660000	4.940000	9837300
2010-04-01	AAP	40.700001	40.380001	40.360001	41.040001	1701700
2010-04-01	AAPL	213.429998	214.009998	212.380001	214.499996	123432400
2010-04-01	ABC	26.290001	26.629999	26.139999	26.690001	2455900
…	…	…	…	…	…	…
2016-12-30	ZBH	103.309998	103.199997	102.849998	103.930000	973800
2016-12-30	ZION	43.070000	43.040001	42.689999	43.310001	1938100
2016-12-30	ZTS	53.639999	53.529999	53.270000	53.740002	1701200

```
2016-12-30    AIV   44.730000   45.450001   44.410000   45.590000   1380900
2016-12-30    FTV   54.200001   53.630001   53.389999   54.480000    705100

[851013 rows x 6 columns]
```

[20]:
```
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)
```

[21]:
```
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()
```

[21]: []



[22]:
```
pd.DataFrame({
"Variance": pca.explained_variance_ratio_
}, index=range(1, len(pca.explained_variance_ratio_) + 1))
```

[22]:

	Variance
1	0.800903
2	0.199041
3	0.000031
4	0.000021
5	0.000004

[ ]:
```
# Most of the data is included in the first component, so we can delete all the␣
↪four last components.
```

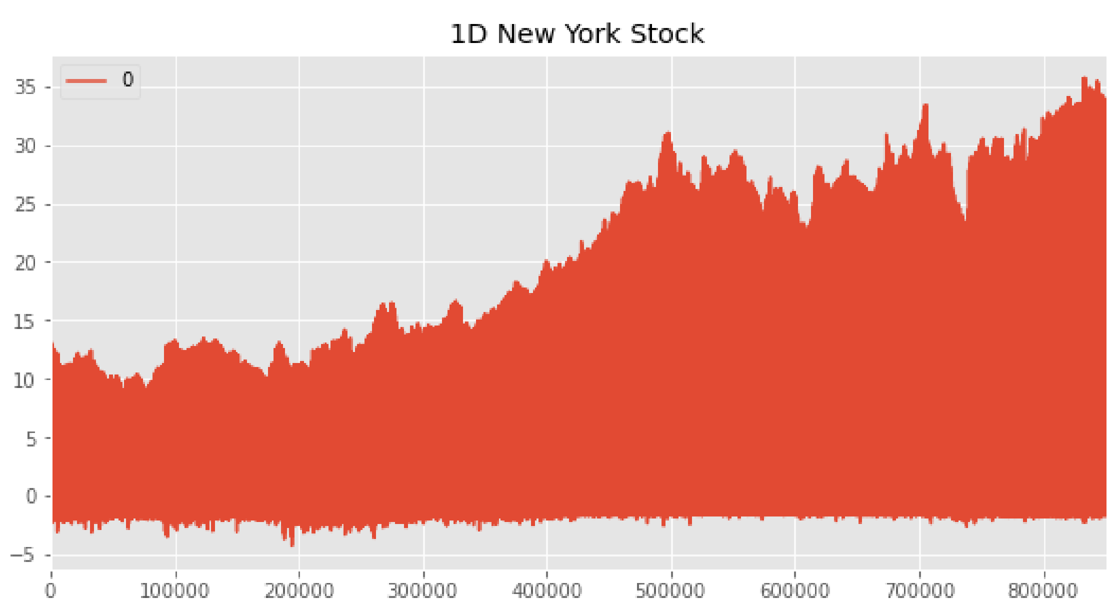
[29]:
```
# using components = 1
pca = PCA(n_components=1)
pca_data = pca.fit_transform(normalized_data)
components = pd.DataFrame(pca.components_, columns = stock_data_set.columns)
components
```

[29]:

	Open	Close	Low	High	Volume
0	0.499597	0.4996	0.49961	0.499599	-0.039919

[30]:
```
pd.DataFrame(pca_data).plot(title="1D New York Stock", figsize=(10,5))
```

[30]: <AxesSubplot:title={'center':'1D New York Stock'}>



```
[ ]:
```