# Big-Data & FinTech: Stock Market in Python

This machine learning project is about clustering similar companies with K-means clustering algorithm for trading purposes. The similarity is based on daily stock movements.

The necessary packages are imported.

#Install Package

!pip install yfinance

!pip install yahoofinancials

import yfinance as yf

from yahoofinancials import YahooFinancials

from datetime import datetime

import matplotlib.pyplot as plt

import pandas as pd

import datetime

import numpy as np

import plotly.graph\_objects as go

A dictionary ‘companies\_dict’ is defined where ‘key’ is company’s name and ‘value’ is company’s stock code. 28 companies are considered, 3 are not listed currently.

## Import the Data

companies\_dict = {

    'Amazon':'AMZN',

    'Apple':'AAPL',

    'Walgreen':'WBA',

    'Northrop Grumman':'NOC',

    'Boeing':'BA',

    'Lockheed Martin':'LMT',

    'McDonalds':'MCD',

    'Intel':'INTC',

    'Navistar':'NAV',

    'IBM':'IBM',

    'Texas Instruments':'TXN',

    'MasterCard':'MA',

    'Microsoft':'MSFT',

    'General Electrics':'GE',

    'Symantec':'SYMC',

    'American Express':'AXP',

    'Pepsi':'PEP',

    'Coca Cola':'KO',

    'Johnson & Johnson':'JNJ',

    'Toyota':'TM',

    'Honda':'HMC',

    'Mistubishi':'MSBHY',

    'Sony':'SNE',

    'Exxon':'XOM',

    'Chevron':'CVX',

    'Valero Energy':'VLO',

    'Ford':'F',

    'Bank of America':'BAC'}

Stock market data is extracted from yahoo finance. The time period considered is from ‘2021–01–01’ to today. The stock movement of companies would be compared based on these 6 parameters :- ‘High’, ‘Low’, ‘Open’, ‘Close’, ‘Volume’, ‘Adj Close’.

‘High’ :- Highest price during the day.  
‘Low’ :- Lowest price during the day.  
‘Open’ :- Opening price of the day.  
‘Close’ :- Closing price of the day.  
‘Volume’ :- Total number of shares of stock traded during the day.  
‘Adj Close’ :- The closing price is amended to account for any corporate actions to give the ‘Adjusted closing’ price.

from datetime import datetime

# Define a start date and End Date

start = '2021-01-01'

#setting today date as End Date

end = datetime.today().strftime('%Y-%m-%d')

data\_source ='yahoo' # Source of data is yahoo finance.

p =  yf.download(list(companies\_dict.values()), start, end)

# Numpy arrays are more convenient. So ‘stock\_open’ and ‘stock\_close’ are defined as follows :-

stock\_open = np.array(p['Open']).T

# stock\_open is numpy array of transpose of p['Open']

stock\_close = np.array(p['Close']).T # stock\_close is numpy array of transpose of p['Close']

## Calculate Movements

# Movement’ is defined as difference of opening and closing prices of a particular day. Positive movement suggests to go long on stock(buy) and negative movement suggests to short the stock(sell).

movements = stock\_close - stock\_open

# ‘sum\_of\_movement’ of a company is defined as sum of differences of closing and opening prices of all days. The company and its ‘sum\_of\_movement’ is printed.

sum\_of\_movement = np.sum(movements,1)

for i in range(len(sum\_of\_movement)):

 print('company:{}, Change:{}'.format(p['High'].columns[i],sum\_of\_movement[i]))

company:AAPL, Change:38.919952392578125

company:AMZN, Change:-747.260986328125

company:AXP, Change:4.769935607910156

company:BA, Change:-82.080078125

company:BAC, Change:-1.7500019073486328

company:CVX, Change:74.63007354736328

company:F, Change:-0.8200006484985352

company:GE, Change:-14.599990844726562

company:HMC, Change:-1.5599994659423828

company:IBM, Change:16.077835083007812

company:INTC, Change:-3.57000732421875

company:JNJ, Change:26.140029907226562

company:KO, Change:2.7100143432617188

company:LMT, Change:84.84014892578125

company:MA, Change:-51.46978759765625

company:MCD, Change:-15.229949951171875

company:MSBHY, Change:nan

company:MSFT, Change:32.500030517578125

company:NAV, Change:nan

company:NOC, Change:136.260009765625

company:PEP, Change:11.850006103515625

company:SNE, Change:nan

company:SYMC, Change:nan

company:TM, Change:-15.90997314453125

company:TXN, Change:2.4601287841796875

company:VLO, Change:11.689998626708984

company:WBA, Change:0.2999992370605469

company:XOM, Change:29.15997314453125

**If we have positive ‘sum\_of\_movement’. Hence it is advisable to go long(buy) on these stocks.**

**If we‘have negative ‘sum\_of\_movement’. Hence it is advisable to short(sell) the stocks.**

## **VISUALIZING THE DATA**

The variation of opening prices of 2 companies(Amazon and Apple) have been plotted below.

plt.figure(figsize = (20,10))

plt.subplot(1,2,1)

plt.title('Company:Amazon',fontsize = 20)

plt.xticks(fontsize = 10)

plt.yticks(fontsize = 20)

plt.xlabel('Date',fontsize = 15)

plt.ylabel('Opening price',fontsize = 15)

plt.plot(p['Open']['AMZN'])

plt.subplot(1,2,2)

plt.title('Company:Apple',fontsize = 20)

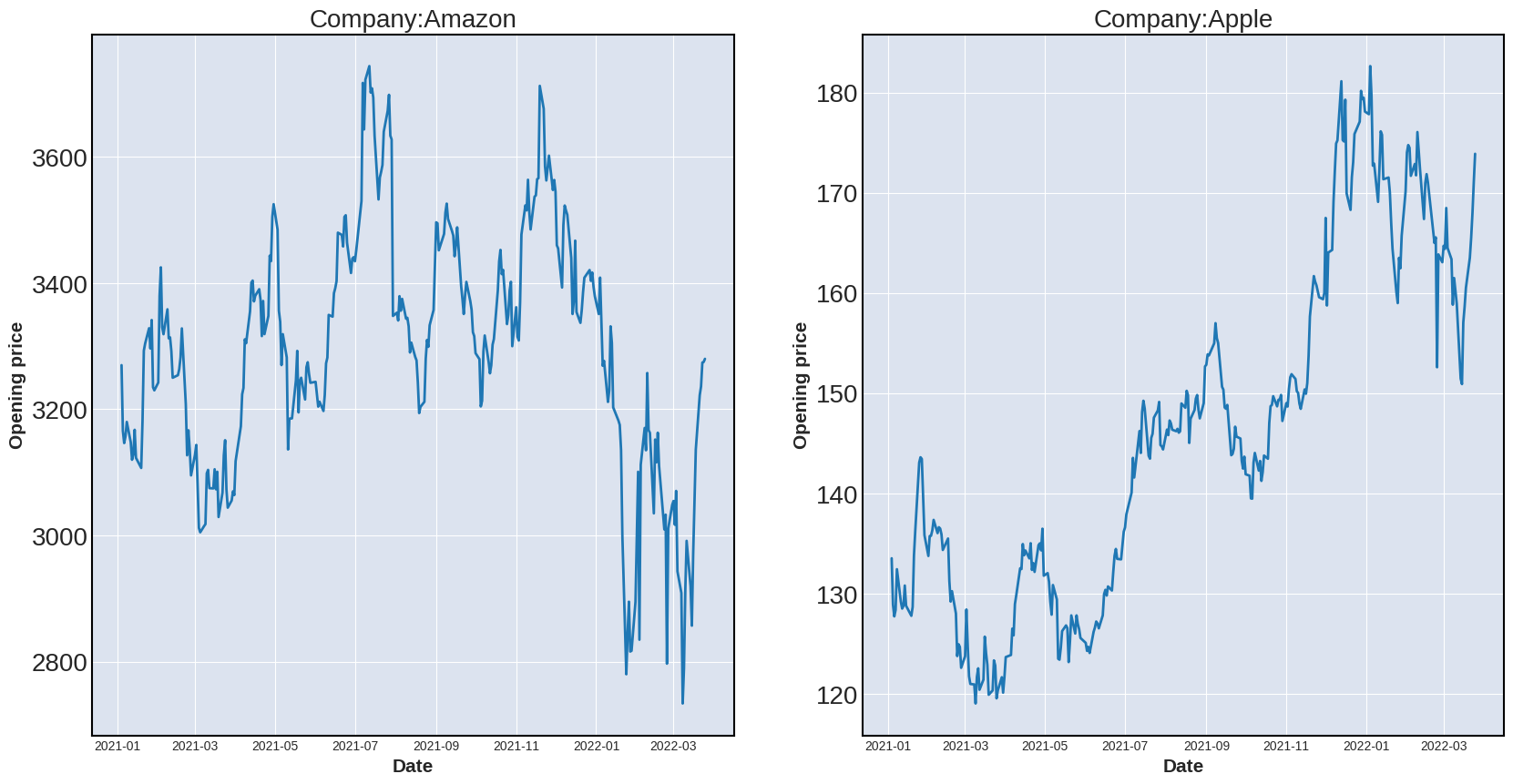
plt.xticks(fontsize = 10)

plt.yticks(fontsize = 20)

plt.xlabel('Date',fontsize = 15)

plt.ylabel('Opening price',fontsize = 15)

plt.plot(p['Open']['AAPL'])



Apple has better growth than Amazon in the given time period.

Let us plot the opening and closing prices of Amazon in the time period from ‘2015–01–02’ to ‘2015–01–23’.

plt.figure(figsize = (20,10)) # Adjusting figure size

plt.title('Company:Amazon',fontsize = 20)

plt.xticks(fontsize = 10)

plt.yticks(fontsize = 20)

plt.xlabel('Date',fontsize = 20)

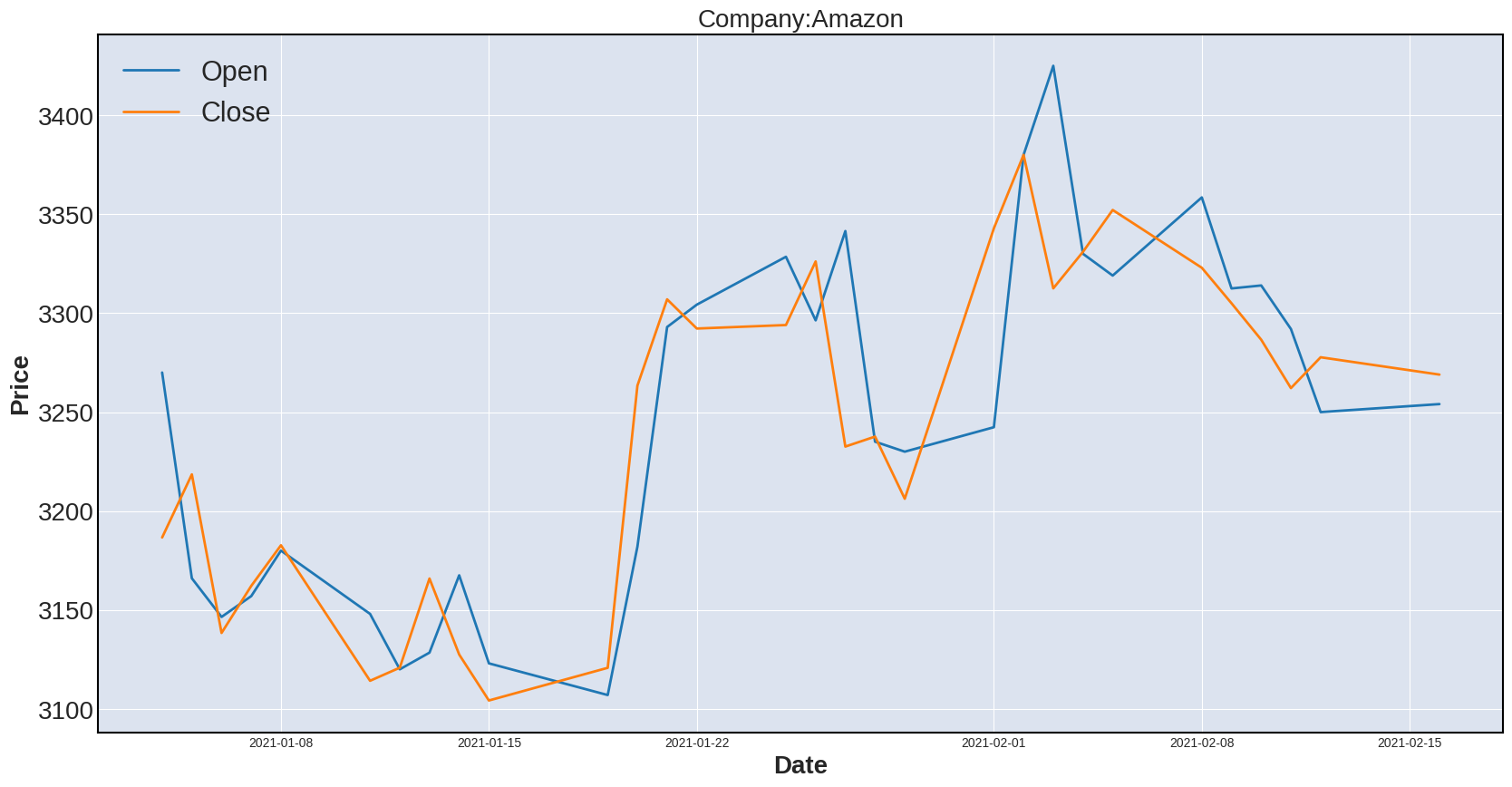
plt.ylabel('Price',fontsize = 20)

plt.plot(p.iloc[0:30]['Open']['AMZN'],label = 'Open') # Opening prices of first 30 days are plotted against date

plt.plot(p.iloc[0:30]['Close']['AMZN'],label = 'Close') # Closing prices of first 30 days are plotted against date

plt.legend(loc='upper left', frameon=False,framealpha=1,prop={'size': 22}) # Properties of legend box

There is an overall increase in ‘Open’ and ‘Close’ prices over the 30 day time period which shows the company in a positive light.

‘Movement’ as defined earlier is difference of closing and opening prices of a particular day. The variation of ‘movement’ of amazon in the time period is plotted below.

plt.figure(figsize = (20,8))

plt.title('Company:Amazon',fontsize = 20)

plt.xticks(fontsize = 18)

plt.yticks(fontsize = 20)

plt.xlabel('Date',fontsize = 20)

plt.ylabel('Movement',fontsize = 20)

plt.plot(movements[0][0:30])

Movement


Positive ‘movement’ is desirable which suggests the price has increased during the day.

Another important parameter is ‘Volume’.

plt.figure(figsize = (20,10))

plt.title('Company:Amazon',fontsize = 20)

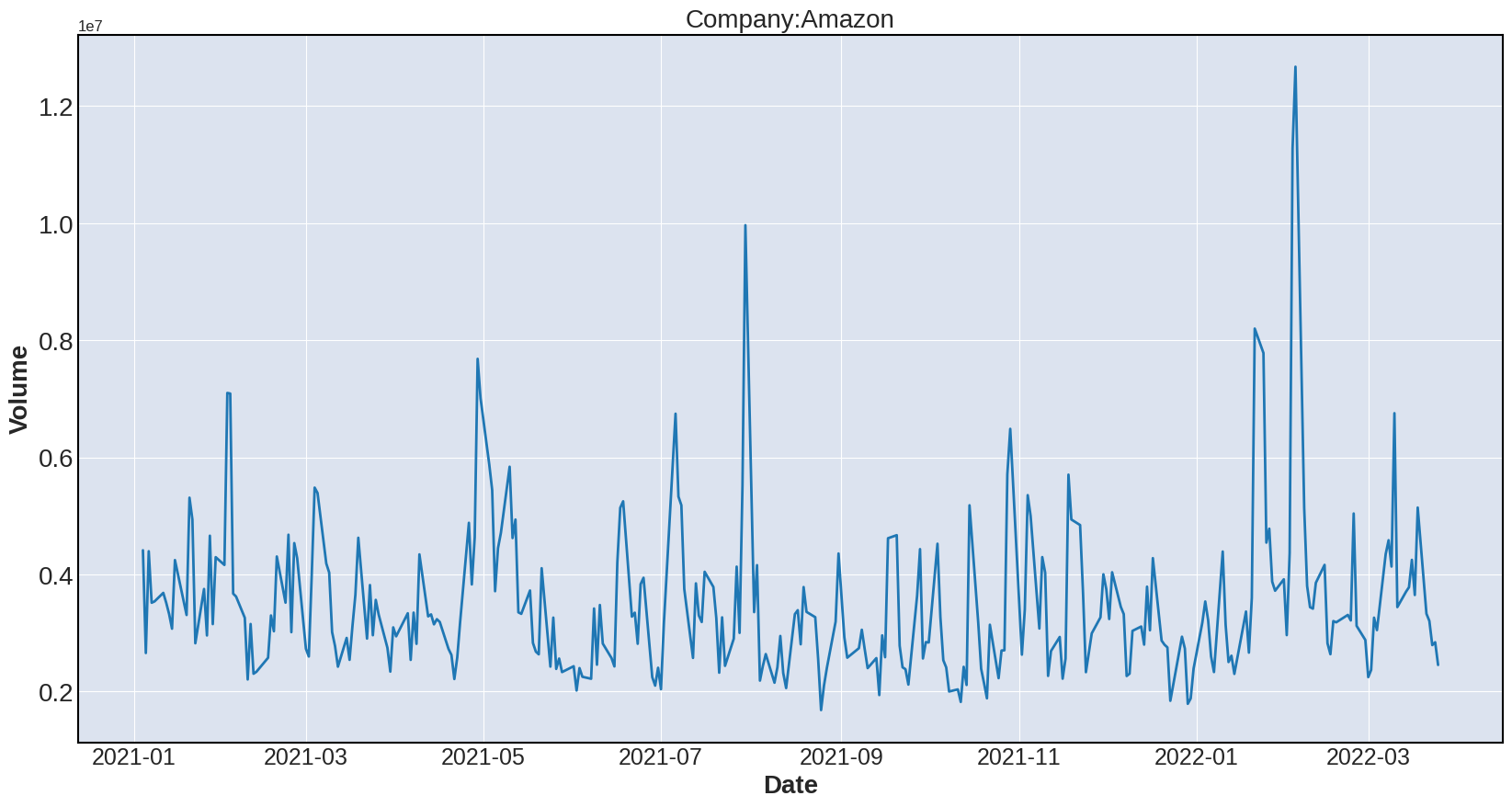
plt.xticks(fontsize = 18)

plt.yticks(fontsize = 20)

plt.xlabel('Date',fontsize = 20)

plt.ylabel('Volume',fontsize = 20)

plt.plot(p['Volume']['AMZN'],label = 'Open')



The peaks suggest there are high volumes of stocks traded on certain days. This may be due to restructuring of company’s management, change of ownership, yearly announcements, new ventures and other factors which affect the image of company in the minds of market participants.

The candlestick chart for the first 60 days of Amazon has been plotted below.

Candlestick pattern is an important tool of technical analysis of stocks to predict particular market movements. A **green** candlestick means that the opening price on that day was lower than the closing price that day (i.e. the price moved up during the day); a **red** candlestick means that the opening price was higher than the closing price that day (i.e. the price moved down during the day).

fig = go.Figure(data=[go.Candlestick(x=p.index,

 open=p.iloc[0:60]['Open']['AMZN'],

 high=p.iloc[0:60]['High']['AMZN'],

 low=p.iloc[0:60]['Low']['AMZN'],

 close=p.iloc[0:60]['Close']['AMZN'])])

fig.show()

A picture containing scatter chart /candlestick


## **NEED FOR NORMALIZATION**

Let us plot the variation of ‘movement’ of Amazon and Apple.

Stock prices of Amazon and Apple have different scales. Thus x units of change in stock price of Amazon is not the same as x units of change in stock price of Apple. Some companies are worth a lot more than other companies. Hence the data has to be normalized.

The goal of Normalization is to change the value of numeric columns in the dataset to a common scale without distorting differences in the range of values. It is required only when features have different ranges like in the present scenario.

**‘norm\_movements’ is defined as normalized version of ‘movements’. The Normalizer() rescales each row of ‘movements’ independently.First remove the nan or null values**

movements

new\_movements = movements[~np.isnan(movements)]

new\_movements

from sklearn.preprocessing import Normalizer

normalizer = Normalizer() # Define a Normalizer

norm\_movements = normalizer.fit\_transform(new\_movements.reshape(-1,1))

# Fit and transform

**The minimum, maximum and mean value of ‘norm\_movements’ is printed.**

print(norm\_movements.min())

print(norm\_movements.max())

print(norm\_movements.mean())

plt.figure(figsize = (20,8))

ax1 = plt.subplot(1,2,1)

plt.title('Company:Amazon',fontsize = 20)

plt.xticks(fontsize = 18)

plt.yticks(fontsize = 20)

plt.xlabel('Date',fontsize = 20)

plt.ylabel('Movement',fontsize = 20)

plt.plot(movements[0])

plt.subplot(1,2,2,sharey = ax1)

plt.title('Company:Apple',fontsize = 20)

plt.xticks(fontsize = 18)

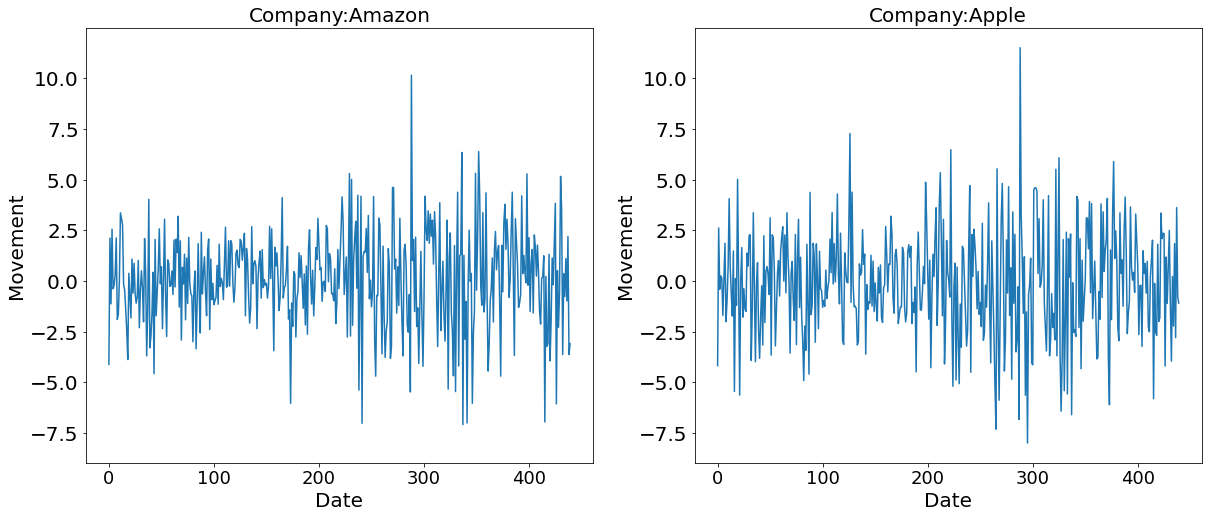
plt.yticks(fontsize = 20)

plt.xlabel('Date',fontsize = 20)

plt.ylabel('Movement',fontsize = 20)

plt.plot(movements[1])

The normalized movements of Amazon and Apple are in similar range.



## 