Predicting the Price of Stocks

Time Series Analysis by Corbin Couger

9/26/2023

Introduction

Having a strong background in statistics and data science you'd think I'd have some understanding of how the stock market works? And, you're sort of right, I understand how the market prices rise and fall, but I am still so clueless to many of the details and techniques that go into trading stocks. I've done extensive research in how money is made by day trading, long term trading, etc. and still find myself lost. This is the case for many people who are wanting to use their extra income to have their money work for them. The market can be daunting to try and get a start in, and in the world we live in currently, people want certainty and results! Now, this is an unrealistic nature of our society, but why not make things easier for people who are not wanting to do tons of research and make figuring out what stocks will make or lose money.

This is where this project comes in. I want (at the best of my ability) to solve the problem of predicting stock prices and answer the question: 'What will a stock be worth in the future?' Now, this is nothing new and has been done in various ways throughout time, but typically this process is still to complicated for any new user and most trading platforms do not provide much insight to this type of analysis. This is a problem I want to solve and make easier for users.

For this project I'm going to use an awesome API I found by Alpha Vantage. This API tracks real-time and historic stock prices and seems like the perfect API for this project. A link to this API source is here or at https://www.alphavantage.co/.

Methods

The goal for this project is to take a user input, preferably stock ticker/symbol, but I'm also thinking of allowing a company name input to make the experience easier. Using the input, I will call the API to collect the necessary data for that stock. I will code and develop a time-series model that creates accurate future price predictions of that specific stock, based on the data. The user will then get a list of furture prices as well as the respective model's accuracy to give some insight on how accurate that price is. Now, there are some concerns and considerations that I have with doing this projcet, so make sure to check them out below.

Ethical Considerations

Of course when it comes to people's personal finances and assets, it is important to be transparent and insightful about what the intentions of the project is. I want to make it clear, this project is simply for fun and could hypothetically be scalable for actual use. I am not a

financial expert, just a Data Scientist who enjoys creating predictive models. I will make sure to address the risk of using this in the real-world as well as output the accuracy of each model to let the user know how accurate the predictions are and to take it with a grain of salt and gamble at their own risk.

Challenges/Issues

I see some possible challengis or issues with this project. Run time could be an issue since I'm creating a new API call, model, and output everytime the program is run. This can take a lot of computer memory and I can see this being a challenge when testing my code. Another issue that could arise is creating consistent time-series models. I would like for each model to be as perfect as the last, but I can assume that not every model will be the same. This is something I will keep myself aware of, but knowing the data will be different for each stock, I will not worry to much about this.

The Data

API KEY: IHF7D726REA61QOT

Like mentioned in the Introduction, this API is from Alpha Vantage and tracks the stock market on many different scales. I will utilize this to capture historic and current stock prices for a user inputed stock symbol and capture the data like that. Here is an example of this:

In [190…
```
import pandas as pd
import numpy as np
import requests
import json
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from datetime import date
import datetime
from pmdarima.arima import auto_arima
from statsmodels.tsa.arima.model import ARIMA
from math import sqrt
from sklearn.metrics import mean_squared_error
```

In [2]:
```
apikey = 'IHF7D726REA61QOT'
symbol = 'IBM'
# replace the "demo" apikey below with your own key from https://www.alphavantage.co/s
url = f'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&outputsize=compac
r = requests.get(url)
data = r.json()
```

In [220…
```
print(data['Time Series (Daily)'])
```

{'2023-10-10': {'1. open': '142.6000', '2. high': '143.4150', '3. low': '141.7200', '4. close': '142.1100', '5. volume': '3015784'}, '2023-10-09': {'1. open': '142.3000', '2. high': '142.4000', '3. low': '140.6800', '4. close': '142.2000', '5. volume': '2354396'}, '2023-10-06': {'1. open': '141.4000', '2. high': '142.9400', '3. low': '140.1100', '4. close': '142.0300', '5. volume': '3511347'}, '2023-10-05': {'1. open': '140.9000', '2. high': '141.7000', '3. low': '140.1900', '4. close': '141.5200', '5. volume': '3223910'}, '2023-10-04': {'1. open': '140.3700', '2. high': '141.2004', '3. low': '139.9900', '4. close': '141.0700', '5. volume': '2637779'}, '2023-10-03': {'1. open': '140.8700', '2. high': '141.6400', '3. low': '140.0000', '4. close': '140.3900', '5. volume': '3284421'}, '2023-10-02': {'1. open': '140.0400', '2. high': '141.4500', '3. low': '139.8600', '4. close': '140.8000', '5. volume': '3275461'}, '2023-09-29': {'1. open': '142.0000', '2. high': '142.1300', '3. low': '139.6100', '4. close': '140.3000', '5. volume': '5703983'}, '2023-09-28': {'1. open': '142.1400', '2. high': '142.2820', '3. low': '140.2050', '4. close': '141.5800', '5. volume': '5783422'}, '2023-09-27': {'1. open': '143.6700', '2. high': '143.8200', '3. low': '141.7600', '4. close': '143.1700', '5. volume': '4439121'}, '2023-09-26': {'1. open': '145.5100', '2. high': '146.1700', '3. low': '143.0201', '4. close': '143.2400', '5. volume': '4824654'}, '2023-09-25': {'1. open': '146.5700', '2. high': '147.4300', '3. low': '146.2500', '4. close': '146.4800', '5. volume': '2694245'}, '2023-09-22': {'1. open': '147.4100', '2. high': '148.1000', '3. low': '146.8200', '4. close': '146.9100', '5. volume': '2562216'}, '2023-09-21': {'1. open': '149.0000', '2. high': '149.2500', '3. low': '147.3100', '4. close': '147.3800', '5. volume': '4944786'}, '2023-09-20': {'1. open': '148.3600', '2. high': '151.9299', '3. low': '148.1300', '4. close': '149.8300', '5. volume': '9636681'}, '2023-09-19': {'1. open': '145.0000', '2. high': '146.7200', '3. low': '144.6600', '4. close': '146.5200', '5. volume': '3945423'}, '2023-09-18': {'1. open': '145.7700', '2. high': '146.4800', '3. low': '145.0600', '4. close': '145.0900', '5. volume': '2508062'}, '2023-09-15': {'1. open': '147.1100', '2. high': '147.8500', '3. low': '145.5300', '4. close': '145.9900', '5. volume': '6234033'}, '2023-09-14': {'1. open': '147.3800', '2. high': '147.7300', '3. low': '146.4800', '4. close': '147.3500', '5. volume': '2723200'}, '2023-09-13': {'1. open': '145.9500', '2. high': '146.9800', '3. low': '145.9200', '4. close': '146.5500', '5. volume': '2627999'}, '2023-09-12': {'1. open': '147.9200', '2. high': '148.0000', '3. low': '145.8000', '4. close': '146.3000', '5. volume': '4457695'}, '2023-09-11': {'1. open': '148.5700', '2. high': '148.7800', '3. low': '147.5800', '4. close': '148.3800', '5. volume': '3273720'}, '2023-09-08': {'1. open': '147.3500', '2. high': '148.5900', '3. low': '147.2600', '4. close': '147.6800', '5. volume': '3722927'}, '2023-09-07': {'1. open': '148.1300', '2. high': '148.7800', '3. low': '147.4000', '4. close': '147.5200', '5. volume': '3333040'}, '2023-09-06': {'1. open': '147.6600', '2. high': '148.3300', '3. low': '147.1200', '4. close': '148.0600', '5. volume': '2932203'}, '2023-09-05': {'1. open': '147.9100', '2. high': '149.0000', '3. low': '147.5719', '4. close': '148.1300', '5. volume': '3731281'}, '2023-09-01': {'1. open': '147.2600', '2. high': '148.1000', '3. low': '146.9200', '4. close': '147.9400', '5. volume': '2727796'}, '2023-08-31': {'1. open': '146.9400', '2. high': '147.7275', '3. low': '146.5400', '4. close': '146.8300', '5. volume': '3885949'}, '2023-08-30': {'1. open': '146.4200', '2. high': '146.9200', '3. low': '145.7452', '4. close': '146.8600', '5. volume': '2245402'}, '2023-08-29': {'1. open': '146.3000', '2. high': '146.7300', '3. low': '145.6200', '4. close': '146.4500', '5. volume': '2778113'}, '2023-08-28': {'1. open': '145.4100', '2. high': '146.7400', '3. low': '145.2100', '4. close': '146.0200', '5. volume': '3561347'}, '2023-08-25': {'1. open': '144.1800', '2. high': '145.4700', '3. low': '143.5000', '4. close': '145.3500', '5. volume': '3660147'}, '2023-08-24': {'1. open': '143.5050', '2. high': '144.4700', '3. low': '143.2200', '4. close': '143.5500', '5. volume': '2900244'}, '2023-08-23': {'1. open': '141.7200', '2. high': '143.4750', '3. low': '141.5800', '4. close': '143.4100', '5. volume': '2559083'}, '2023-08-22': {'1. open': '142.6600', '2. high': '143.2250', '3. low': '141.3000', '4. close': '141.4900', '5. volume': '3557734'}, '2023-08-21': {'1. open': '141.4200', '2. high': '142.3900', '3. low': '141.1100', '4. close': '142.2800', '5. volume': '2937781'}, '2023-08-18': {'1. open': '140.0000', '2. high': '141.8300', '3. low': '139.7600', '4. close': '141.4100', '5. volume': '3915480'}, '2023-08-17': {'1. open': '141.0100', '2. high': '142.6600', '3. low': '140.6000', '4. close': '140.6600', '5. volume': '3742058'}, '2023-08-16': {'1. open': '141.7000', '2. high': '142.0900', '3. low': '140.

5600', '4. close': '140.6400', '5. volume': '3285347'}, '2023-08-15': {'1. open': '14
1.5000', '2. high': '142.3100', '3. low': '141.2000', '4. close': '141.8700', '5. vol
ume': '3656559'}, '2023-08-14': {'1. open': '143.0500', '2. high': '143.3650', '3. lo
w': '141.8020', '4. close': '141.9100', '5. volume': '4226563'}, '2023-08-11': {'1. o
pen': '143.1200', '2. high': '143.4500', '3. low': '142.2050', '4. close': '143.120
0', '5. volume': '2526433'}, '2023-08-10': {'1. open': '143.0400', '2. high': '144.58
00', '3. low': '142.6900', '4. close': '143.2500', '5. volume': '4735763'}, '2023-08-
09': {'1. open': '144.9400', '2. high': '144.9400', '3. low': '142.3000', '4. close':
'142.4900', '5. volume': '4073038'}, '2023-08-08': {'1. open': '145.7000', '2. high':
'146.1500', '3. low': '144.1100', '4. close': '145.9100', '5. volume': '4654582'}, '2
023-08-07': {'1. open': '145.0000', '2. high': '146.5000', '3. low': '144.9300', '4.
close': '146.1800', '5. volume': '3438654'}, '2023-08-04': {'1. open': '145.0900',
'2. high': '146.0900', '3. low': '143.9900', '4. close': '144.2400', '5. volume': '42
23204'}, '2023-08-03': {'1. open': '143.7800', '2. high': '145.2200', '3. low': '143.
3116', '4. close': '144.4500', '5. volume': '3952640'}, '2023-08-02': {'1. open': '14
2.7800', '2. high': '144.3000', '3. low': '142.3100', '4. close': '144.1700', '5. vol
ume': '4959381'}, '2023-08-01': {'1. open': '144.2500', '2. high': '144.4800', '3. lo
w': '142.1700', '4. close': '143.3300', '5. volume': '4798703'}, '2023-07-31': {'1. o
pen': '143.8100', '2. high': '144.6050', '3. low': '143.5300', '4. close': '144.180
0', '5. volume': '6138902'}, '2023-07-28': {'1. open': '143.4400', '2. high': '143.95
00', '3. low': '142.8500', '4. close': '143.4500', '5. volume': '6686627'}, '2023-07-
27': {'1. open': '142.3000', '2. high': '143.3800', '3. low': '141.9000', '4. close':
'142.9700', '5. volume': '6331563'}, '2023-07-26': {'1. open': '140.4400', '2. high':
'141.2500', '3. low': '139.8800', '4. close': '141.0700', '5. volume': '4046441'}, '2
023-07-25': {'1. open': '139.4200', '2. high': '140.4300', '3. low': '139.0403', '4.
close': '140.3300', '5. volume': '3770813'}, '2023-07-24': {'1. open': '139.3500',
'2. high': '140.1200', '3. low': '138.7788', '4. close': '139.5400', '5. volume': '34
75442'}, '2023-07-21': {'1. open': '138.2100', '2. high': '139.7799', '3. low': '137.
7600', '4. close': '138.9400', '5. volume': '5858741'}, '2023-07-20': {'1. open': '13
7.1900', '2. high': '140.3200', '3. low': '136.5600', '4. close': '138.3800', '5. vol
ume': '10896330'}, '2023-07-19': {'1. open': '135.5300', '2. high': '136.4500', '3. l
ow': '135.1900', '4. close': '135.4800', '5. volume': '5519992'}, '2023-07-18': {'1.
open': '134.7100', '2. high': '135.9500', '3. low': '134.2900', '4. close': '135.360
0', '5. volume': '3852058'}, '2023-07-17': {'1. open': '133.2600', '2. high': '134.61
00', '3. low': '133.1000', '4. close': '134.2400', '5. volume': '3168419'}, '2023-07-
14': {'1. open': '133.9100', '2. high': '133.9200', '3. low': '132.9400', '4. close':
'133.4000', '5. volume': '2861496'}, '2023-07-13': {'1. open': '133.5100', '2. high':
'135.0700', '3. low': '133.3600', '4. close': '133.9200', '5. volume': '3221422'}, '2
023-07-12': {'1. open': '135.0700', '2. high': '135.3300', '3. low': '132.5750', '4.
close': '132.8400', '5. volume': '3732189'}, '2023-07-11': {'1. open': '133.6600',
'2. high': '134.5600', '3. low': '133.2300', '4. close': '134.4400', '5. volume': '29
25238'}, '2023-07-10': {'1. open': '131.7600', '2. high': '133.0500', '3. low': '131.
6950', '4. close': '132.9000', '5. volume': '2369425'}, '2023-07-07': {'1. open': '13
1.7800', '2. high': '133.8550', '3. low': '131.7500', '4. close': '132.0800', '5. vol
ume': '2982738'}, '2023-07-06': {'1. open': '133.2350', '2. high': '133.9000', '3. lo
w': '131.5500', '4. close': '132.1600', '5. volume': '3508083'}, '2023-07-05': {'1. o
pen': '133.3200', '2. high': '134.3100', '3. low': '132.5900', '4. close': '134.240
0', '5. volume': '2955870'}, '2023-07-03': {'1. open': '133.4200', '2. high': '134.35
00', '3. low': '132.8700', '4. close': '133.6700', '5. volume': '1477149'}, '2023-06-
30': {'1. open': '134.6900', '2. high': '135.0300', '3. low': '133.4250', '4. close':
'133.8100', '5. volume': '4236677'}, '2023-06-29': {'1. open': '131.7500', '2. high':
'134.3500', '3. low': '131.6900', '4. close': '134.0600', '5. volume': '3639836'}, '2
023-06-28': {'1. open': '132.0600', '2. high': '132.1700', '3. low': '130.9100', '4.
close': '131.7600', '5. volume': '2753779'}, '2023-06-27': {'1. open': '131.3000',
'2. high': '132.9500', '3. low': '130.8300', '4. close': '132.3400', '5. volume': '32
19909'}, '2023-06-26': {'1. open': '129.3900', '2. high': '131.4100', '3. low': '129.
3100', '4. close': '131.3400', '5. volume': '4845649'}, '2023-06-23': {'1. open': '13
0.4000', '2. high': '130.6200', '3. low': '129.1800', '4. close': '129.4300', '5. vol
ume': '11324705'}, '2023-06-22': {'1. open': '131.6800', '2. high': '132.9600', '3. l
ow': '130.6800', '4. close': '131.1700', '5. volume': '6013021'}, '2023-06-21': {'1.

open': '135.1100', '2. high': '135.3900', '3. low': '133.2900', '4. close': '133.690
0', '5. volume': '5501272'}, '2023-06-20': {'1. open': '136.3600', '2. high': '137.23
00', '3. low': '135.8900', '4. close': '135.9600', '5. volume': '4272511'}, '2023-06-
16': {'1. open': '139.2300', '2. high': '139.4690', '3. low': '137.4700', '4. close':
'137.4800', '5. volume': '7473676'}, '2023-06-15': {'1. open': '137.2700', '2. high':
'138.8000', '3. low': '137.1750', '4. close': '138.4000', '5. volume': '3812582'}, '2
023-06-14': {'1. open': '137.8000', '2. high': '138.9300', '3. low': '136.9400', '4.
close': '137.2000', '5. volume': '4514888'}, '2023-06-13': {'1. open': '136.5100',
'2. high': '138.1700', '3. low': '136.0000', '4. close': '137.6000', '5. volume': '39
27331'}, '2023-06-12': {'1. open': '136.0000', '2. high': '136.6200', '3. low': '135.
8216', '4. close': '136.4200', '5. volume': '4500120'}, '2023-06-09': {'1. open': '13
4.3600', '2. high': '136.1000', '3. low': '134.1700', '4. close': '135.3000', '5. vol
ume': '3981748'}, '2023-06-08': {'1. open': '134.6900', '2. high': '135.9800', '3. lo
w': '134.0100', '4. close': '134.4100', '5. volume': '4128939'}, '2023-06-07': {'1. o
pen': '132.5000', '2. high': '134.4400', '3. low': '132.1900', '4. close': '134.380
0', '5. volume': '5772024'}, '2023-06-06': {'1. open': '132.4300', '2. high': '132.94
00', '3. low': '131.8800', '4. close': '132.6900', '5. volume': '3297951'}, '2023-06-
05': {'1. open': '133.1200', '2. high': '133.5800', '3. low': '132.2700', '4. close':
'132.6400', '5. volume': '3993516'}, '2023-06-02': {'1. open': '130.3800', '2. high':
'133.1200', '3. low': '130.1500', '4. close': '132.4200', '5. volume': '5375796'}, '2
023-06-01': {'1. open': '128.4400', '2. high': '130.1450', '3. low': '127.7800', '4.
close': '129.8200', '5. volume': '4136086'}, '2023-05-31': {'1. open': '128.5100',
'2. high': '129.4400', '3. low': '127.4600', '4. close': '128.5900', '5. volume': '11
086313'}, '2023-05-30': {'1. open': '129.5600', '2. high': '130.0699', '3. low': '12
8.2600', '4. close': '129.4800', '5. volume': '3741050'}, '2023-05-26': {'1. open':
'127.0600', '2. high': '129.6600', '3. low': '126.8100', '4. close': '128.8900', '5.
volume': '5612570'}, '2023-05-25': {'1. open': '125.6100', '2. high': '127.2300', '3.
low': '125.0100', '4. close': '126.7600', '5. volume': '4102854'}, '2023-05-24': {'1.
open': '127.8200', '2. high': '127.9000', '3. low': '125.4700', '4. close': '125.680
0', '5. volume': '3915505'}, '2023-05-23': {'1. open': '127.2400', '2. high': '129.09
00', '3. low': '127.1300', '4. close': '128.1800', '5. volume': '4592280'}, '2023-05-
22': {'1. open': '127.5000', '2. high': '128.1900', '3. low': '127.1500', '4. close':
'127.5000', '5. volume': '2806770'}, '2023-05-19': {'1. open': '126.7900', '2. high':
'128.2900', '3. low': '126.5500', '4. close': '127.2600', '5. volume': '4306657'}, '2
023-05-18': {'1. open': '125.3000', '2. high': '126.5100', '3. low': '125.1894', '4.
close': '126.1500', '5. volume': '3797883'}}

As shown by this sample output, the API can return daily prices for the stock market for any stock. The data I call will return the past 100 days of stock prices for the specified company. Now, I can do this with the past 20+ years of historic data, but that might not be necessary. I will make more decisions on that when it comes to building the time series model.

Overall, this project is ready to be underway and get to solving the problem of keeping stock price predictions simple and easy to use.

Getting Data from User Input

In this section I will be all about gathering the time series data from the user's input. This will be making sure the user input is usable for the API call and making sure the user has a chance to input again if they mess up.

In [4]:
```
while True:
    symbol = input('Please enter a stock ticker (symbol) for the company you wish to a
    url = f'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&outputsize=co
    r = requests.get(url)
```

```
    data = r.json()
    if next(iter(data)) == 'Error Message':
        print('You did not enter a valid stock symbol, please try again.')
        continue
    else:
        break
```

```
Please enter a stock ticker (symbol) for the company you wish to analyze: Ibm
```

In [5]:
```
df = pd.DataFrame.from_dict(data['Time Series (Daily)']).reset_index()
df = df.T
df.columns = df.iloc[0]
df = df[1:]
df = df.reset_index()
df.rename(columns = {'index':'Date', '1. open': 'open', '2. high': 'high', '3. low': '
df = df.iloc[::-1]
df.head()
```

Out[5]:

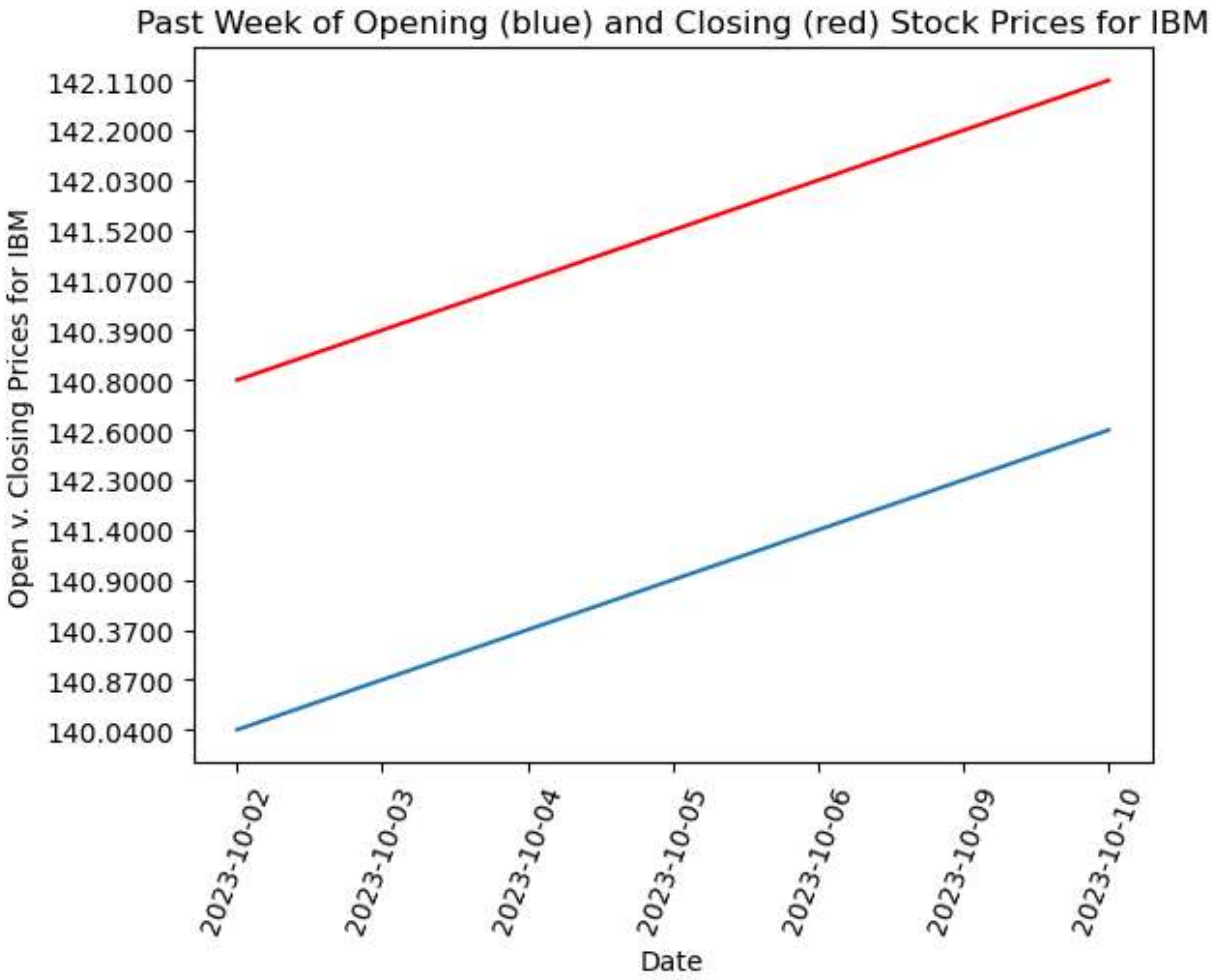
index	Date	open	high	low	close	volume
99	2023-05-18	125.3000	126.5100	125.1894	126.1500	3797883
98	2023-05-19	126.7900	128.2900	126.5500	127.2600	4306657
97	2023-05-22	127.5000	128.1900	127.1500	127.5000	2806770
96	2023-05-23	127.2400	129.0900	127.1300	128.1800	4592280
95	2023-05-24	127.8200	127.9000	125.4700	125.6800	3915505

Generating Visuals for the User

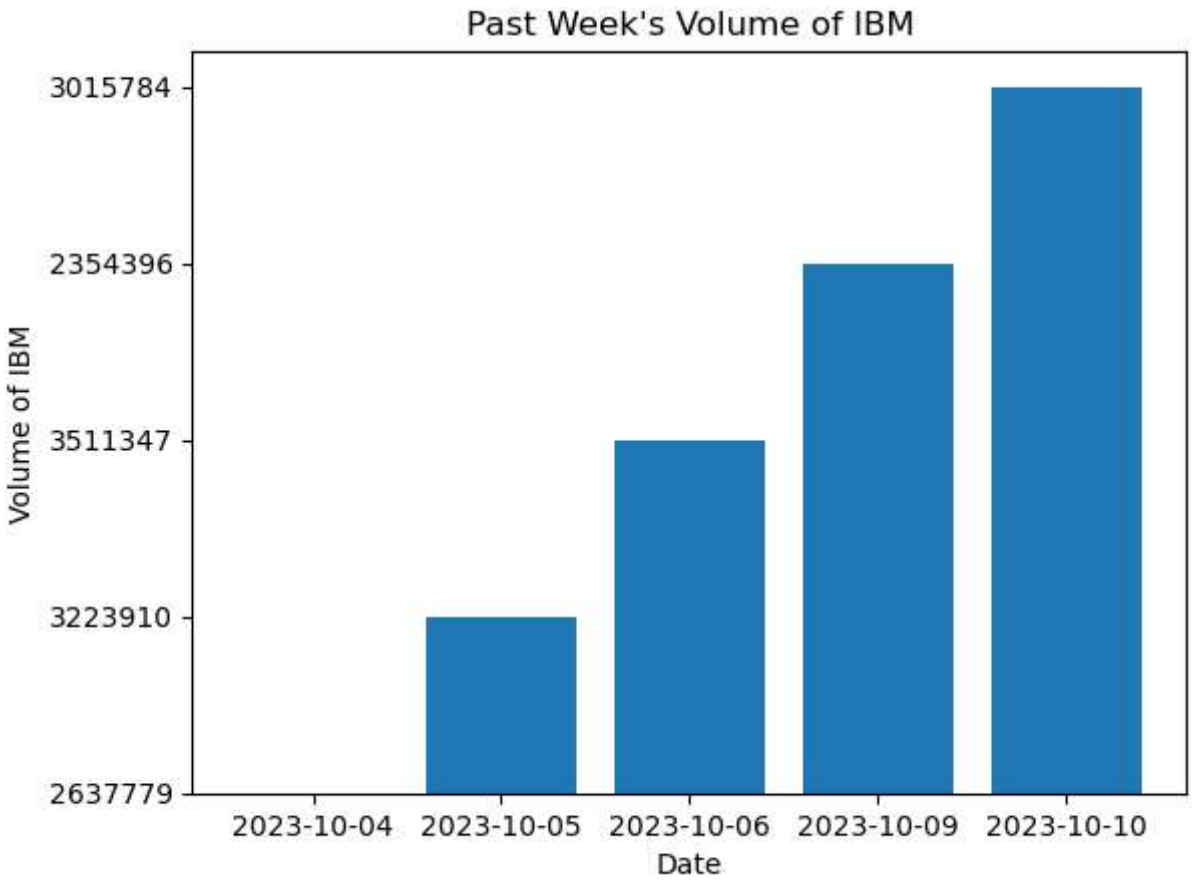
In this section of the project, I want to create some useful plots to show off the stock's performance. This will hopefully give initial insight on what the user will see in the results of the time series analysis.

In [13]:
```
# Open vs. Close Plot:

plt.plot(df.Date.tail(7), df.open.tail(7))
plt.plot(df.Date.tail(7), df.close.tail(7), color = 'r')
plt.xlabel('Date')
plt.xticks(rotation = 70)
plt.ylabel('Open v. Closing Prices for '+symbol.upper())
plt.title('Past Week of Opening (blue) and Closing (red) Stock Prices for '+symbol.upp
plt.show()
```



In [17]:
```
plt.bar(df.Date.tail(), df.volume.tail())
plt.xlabel('Date')
plt.ylabel('Volume of ' + symbol.upper())
plt.title('Past Week\'s Volume of ' + symbol.upper())
plt.show()
```



Time-Series Modeling

In this seciton of the project I will be generating the time-series model for the specified stock and create the predictions of the stock's price. Once the model is created, I will create a visual of the time-serie's predictions for the user. I first need to check/update some of the data types for this model to work, then I can get into creating the model.

In [7]:
```
df.dtypes
```

Out[7]:
```
index
Date      object
open      object
high      object
low       object
close     object
volume    object
dtype: object
```

In [37]:
```
df['Date'] = pd.to_datetime(df['Date'], format='%Y-%m-%d')#.astype('int64')
```

Next, I will test for Stationarity for the time-series data. This is important for the model because this will test whether or not the data has trends or is constant. To do this, I will utilize the Dickey Fuller which creates helpful statistics to let me know if this data passes or fails the Stationarity test. From this point on, I will be using this model using the 'open' variable since I want to predict what prices are going to open the next day with.

In [192…
```
df_open = df[['Date', 'open']]
```

In [193…
```
df_open.index = df_open['Date']
del df_open['Date']
```

In [194…
```
df_open.head()
```

Out[194]:

Date	index open
2023-05-18	125.3000
2023-05-19	126.7900
2023-05-22	127.5000
2023-05-23	127.2400
2023-05-24	127.8200

In [195…
```
adft = adfuller(df_open,autolag="AIC")
```

In [196…
```
adft_output_df = pd.DataFrame({"Values":[adft[0],adft[1],adft[2],adft[3],
                                         adft[4]['1%'], adft[4]['5%'], adft[4]['10%']],
                               "Metric":["Test Statistics","p-value","No. of lags used","Nu
                                         "critical value (1%)", "critical value (5%)", "cri
```

In [197…
```
adft_output_df
```

Out[197]:

	Values	Metric
0	-2.046432	Test Statistics
1	0.266592	p-value
2	0.000000	No. of lags used
3	99.000000	Number of observations used
4	-3.498198	critical value (1%)
5	-2.891208	critical value (5%)
6	-2.582596	critical value (10%)

In [198…
```
if adft[1] > .05:
    print('This time-series has no Stationarity, good for time-series modeling')
else:
    print('This time-series has Stationarity and is not a good fit for a time-series m
```

```
This time-series has no Stationarity, good for time-series modeling
```

I will now step into tracking the autocorrelation which will let me know if the future prices will be correlated with the trends of the historic prices.

In [199…
```
df_open['open'] = pd.to_numeric(df_open['open'])
```

```
C:\Users\corbi\AppData\Local\Temp\ipykernel_13992\510266844.py:1: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df_open['open'] = pd.to_numeric(df_open['open'])
```

In [200…

```
ac3 = df_open['open'].autocorr(lag=3)
print("3 Month Lag: ", ac3)

ac6 = df_open['open'].autocorr(lag=6)
print("6 Month Lag: ", ac6)

ac9 = df_open['open'].autocorr(lag=9)
print("9 Month Lag: ", ac9)
```

```
3 Month Lag:  0.9324055328111263
6 Month Lag:  0.837995405458476
9 Month Lag:  0.7478923705729305
```

For a 3, 6, and even 9 month lag of the autocorrelation, there is a high correlation amongst the trending upwards of this specific stock.

Now is time to create the time-series model where I will generate predictions! This will be a cool and powerful time-series method called ARIMA.

In [201…

```
df_open['Dt'] = df_open.index
#df_open['Date'] = df_open['Date']
df_open = df_open.resample('D').ffill().reset_index()
df_open.head()
```

```
C:\Users\corbi\AppData\Local\Temp\ipykernel_13992\84236231.py:1: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df_open['Dt'] = df_open.index
```

Out[201]:

index	Date	open	Dt
0	2023-05-18	125.30	2023-05-18
1	2023-05-19	126.79	2023-05-19
2	2023-05-20	126.79	2023-05-19
3	2023-05-21	126.79	2023-05-19
4	2023-05-22	127.50	2023-05-22

In [202…

```
df_open.index = df_open['Date']
del df_open['Dt']
```

In [203…

```
train = df_open[df_open['Date'] < pd.Timestamp('now').floor('D') + pd.Timedelta(-7, ur
train.head()
```

Out[203]:

	index	Date	open
Date			
2023-05-18	2023-05-18	125.30
2023-05-19	2023-05-19	126.79
2023-05-20	2023-05-20	126.79
2023-05-21	2023-05-21	126.79
2023-05-22	2023-05-22	127.50

In [204…

```
test = df_open[df_open['Date'] >= pd.Timestamp('now').floor('D') + pd.Timedelta(-7, ur
test.head()
```

Out[204]:

	index	Date	open
Date			
2023-10-04	2023-10-04	140.37
2023-10-05	2023-10-05	140.90
2023-10-06	2023-10-06	141.40
2023-10-07	2023-10-07	141.40
2023-10-08	2023-10-08	141.40

In [205…

```
train['train'] = train['open']
test['test'] = test['open']
```

```
C:\Users\corbi\AppData\Local\Temp\ipykernel_13992\148918458.py:1: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  train['train'] = train['open']
C:\Users\corbi\AppData\Local\Temp\ipykernel_13992\148918458.py:2: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  test['test'] = test['open']
```

In [206…

```
del train['Date']
del test['Date']
```

In [207…

```
del train['open']
del test['open']
```

In [208…

```
train.head()
```

Out[208]:

Date	index train
2023-05-18	125.30
2023-05-19	126.79
2023-05-20	126.79
2023-05-21	126.79
2023-05-22	127.50

In [209…

```
test.head()
```

Out[209]:

Date	index test
2023-10-04	140.37
2023-10-05	140.90
2023-10-06	141.40
2023-10-07	141.40
2023-10-08	141.40

In [213…

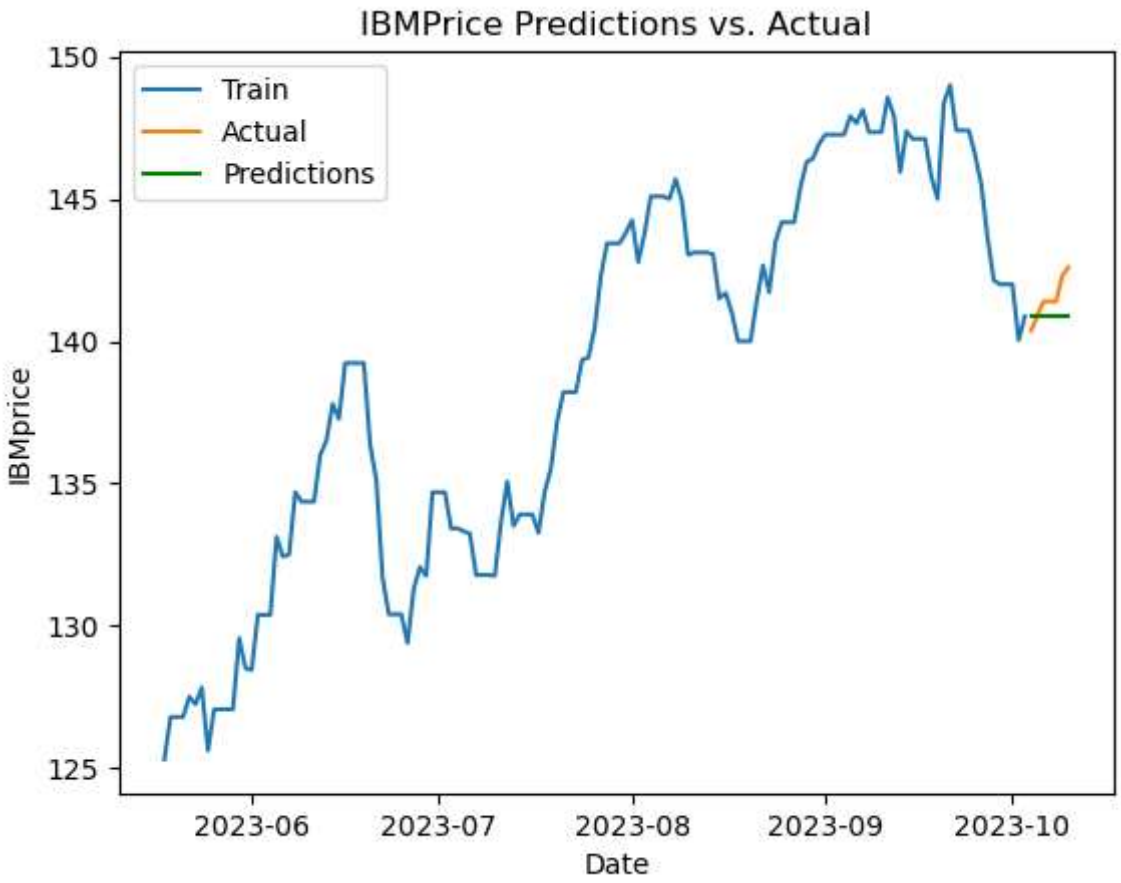
```
model = auto_arima(train, trace=True, error_action='ignore', suppress_warnings=True)
model.fit(train)
forecast = model.predict(n_periods=len(test))
forecast = pd.DataFrame(forecast,index = test.index,columns=['Prediction'])
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.67 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=421.862, Time=0.02 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=423.481, Time=0.06 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=423.496, Time=0.07 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=421.307, Time=0.03 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=424.121, Time=0.31 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0]
Total fit time: 1.183 seconds
```

In [214…

```
# plotting the time-series
plt.plot(train, label = 'Train')
plt.plot(test, label = 'Actual')
plt.plot(forecast, color = 'g', label = 'Predictions')
plt.xlabel('Date')
plt.ylabel(symbol.upper() + 'price')
plt.title(symbol.upper() + 'Price Predictions vs. Actual')
plt.legend()
plt.show()
```



Finally, I want to check the RMSE for this model to get an idea for it's reliability.

In [219…

```
rmse = sqrt(mean_squared_error(test,forecast))
print("RMSE: ", rmse)
print('Your predicted open prices are:')
print(forecast)
print('WARNING: These predictions are based on a time-series model utilizing the last
```

```
RMSE:  0.9359029253690172
Your predicted open prices are:
            Prediction
Date
2023-10-04      140.87
2023-10-05      140.87
2023-10-06      140.87
2023-10-07      140.87
2023-10-08      140.87
2023-10-09      140.87
2023-10-10      140.87
WARNING: These predictions are based on a time-series model utilizing the last 100 da
ys of IBM stock. This is not finacial advice, this is simply a prediction from a pred
ictive model.
```

Conclusion

Looking at the results of the time-series predictions, I'm not super satisfyed with them. The RMSE is too high, meaning the model isn't a good fit for predicting the price of the user

specified stock. Hopefully, the model created for a different stock would come out a better fit. This is something that I'm glad I address in the ethical considerations and the output.

Like I noted in the challenges of this project, I knew this idea would be tough to replicate for each individual stock. So, though I'm not happy with the stock predictions for IBM, there could be better predictions for other stocks. It all comes with testing which I will do more of when putting this project into some 'practical' use.

Using this project and application of this could hypothetically create an easy way for non-experienced stock traders, learn how to see where stock prices are going based on the stocks they like.