**B.3**

For this task, I created two functions, one for a candlestick chart, and one for a boxplot chart. I started with the candlestick chart. I had a lot of trouble getting started properly on this, as I had a lot of trouble with the Date column, but after I figured out what was wrong and understood how it was working, I got through the charts pretty quickly. So I spent most of the time just debugging and understanding a concept related to the load\_data() function.

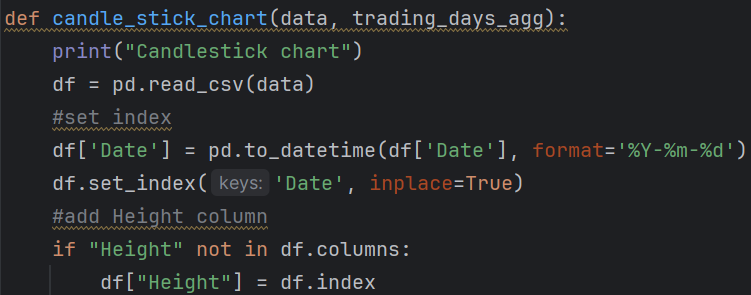
To start off, I added a string variable called “data” to the paraments, so that when called, the function will access the file specified. Later I added the Boolean trading\_days\_agg, but I will come back to that.

**Candlestick Chart**

Function call at end of file



Function



First, I assigned the dataframe to be used with pd.read\_csv(data), as the data has already been loaded and saved to a .csv file with the load\_data() function.

I converted df[‘Date’] to datetime and then set the index to Date. At first I didn’t think I needed to do this since I already did that in the load\_data() function, but not having it was creating an error with the dates being displayed on the X axis of the chart figure.

Next, I added Height as a column as the candlesticks needed their own column for the calculations.

A computer screen shot of a program

Description automatically generated

I created a condition for the *trading\_days\_agg paramenter*. If True is parsed through the parameter, the data will resample and aggregate to display data according to *n* number of trading days (the D within the resample parameter represents days, if I . If False is parsed through, the candlestick chart wrote M, it would resample by *n* number of trading months. will display the data without aggregating.

The candles need to be defined by making a copy of the dataframe, where Close is greater than Open for green, and Open is greater than Close for red. If *trading\_days\_agg* is set to True, a new dataframe name (n\_df) is used with the assigned aggregated data.

After the candles are first declared, they need to be altered to subtract Open or Close from each to show the difference during each trading timeframe. The green candles show the increase of prices, while the red candles show the decline in prices. These will be plotted with plt.bar. While plt.vlines will create the lines or “wicks” that go through them, that represents the overall height of the prices in trading days, with the top of the wick being the highest price reached, and the bottom of the wick being the lowest price reached.

A screen shot of a computer

Description automatically generated

I created two constant variables for easy colour reference, to be used when plotting the chart. Then set the figure size, followed by labels. I noticed that plt.figure() should be coded before the labels, otherwise the labels will not show up on the figure.

Next, I wanted the dates to be displayed on the X axis by month, so that it’s not over-saturated with labels and smashing text together, so I used *plt.gca().xaxis.set\_major\_locator(mdates.MonthLocator())* to space out the time labels by month, *and plt.gca().xaxis.set\_major\_formatter(plt.matplotlib.dates.DateFormatter(‘%b-%Y’))*  to format the labels to show a shorthand label for each month, a dash, and then the year (with a capital Y, as I learnt that capitals format 4 digits while lower case formats 2 digits).

For plotting the candles, it was imperative that the index be set to ‘Date’, so that I could set *x=green/red\_candle.index*, as *green/red\_candle[‘Date’]* did not work for some reason, even though the man in supplied tutorial video did that without any issue.

A computer screen shot of text

Description automatically generated

**Results**

For this task, I set the dates to a shorter timeframe to make debugging faster.

A black background with white text

Description automatically generated

With trading\_days\_agg set to False

A graph of candlesticks and numbers

Description automatically generated

With it set to True, and n is 5

When the data is aggregated to represent n trading days, the candlesticks should get longer, with fewer appearing as the days (or n) increases.

A graph with red and green lines

Description automatically generated

With n set to 20

A graph with green and red points

Description automatically generated

After reducing the TRAIN\_START\_DATE by a year (just for demonstrating)

A graph with red and green lines

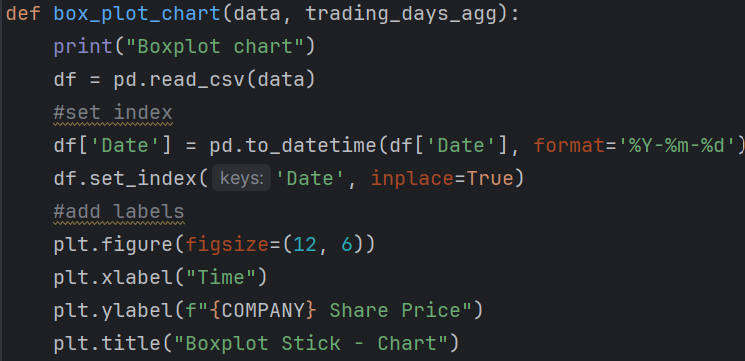
Description automatically generated

A black screen with white text

Description automatically generated

**Boxplot Chart**

The boxplot chart was a lot simpler to work with, and mainly just involved parsing the dataframe into the plt.boxplot() function

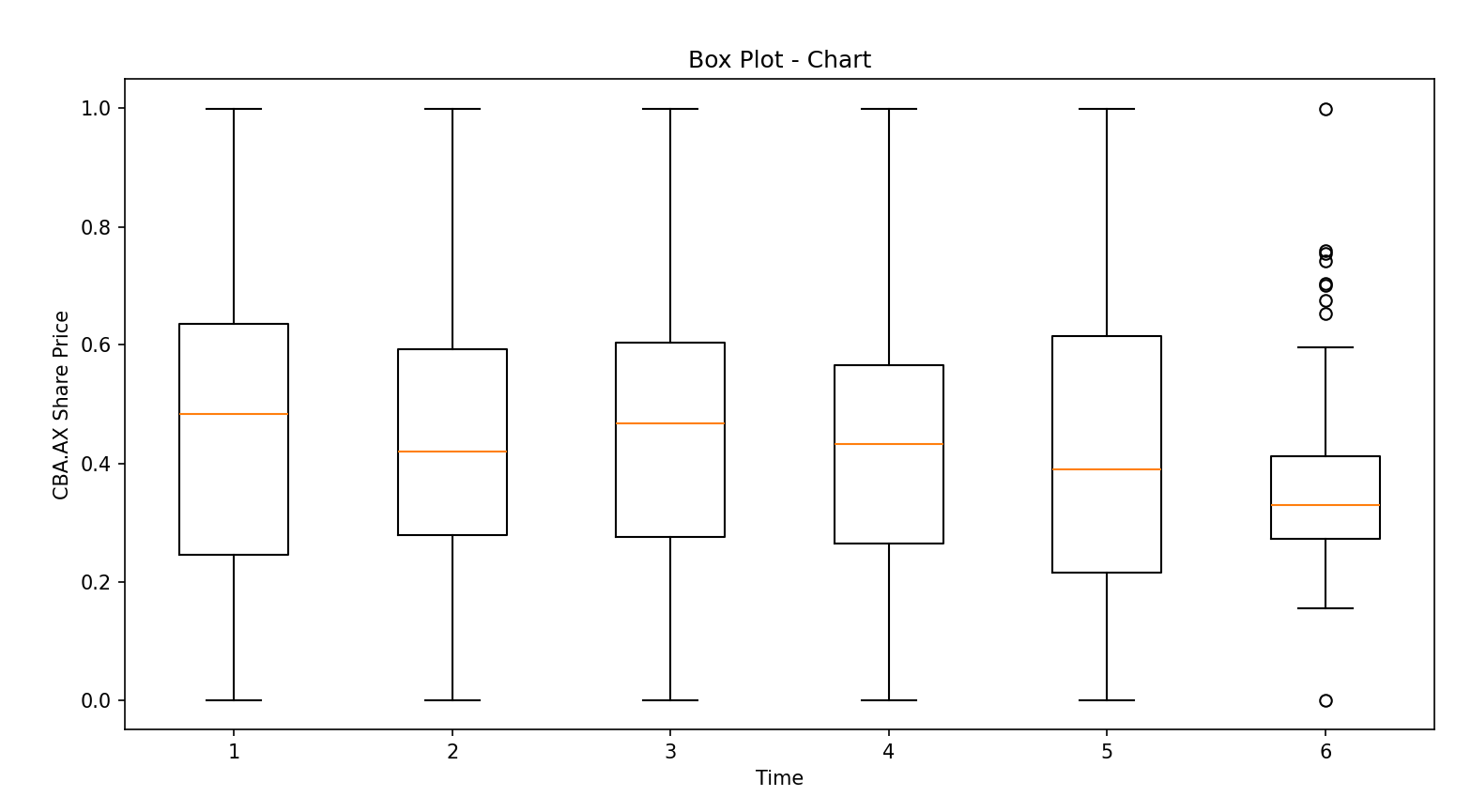


A computer screen shot of a program code

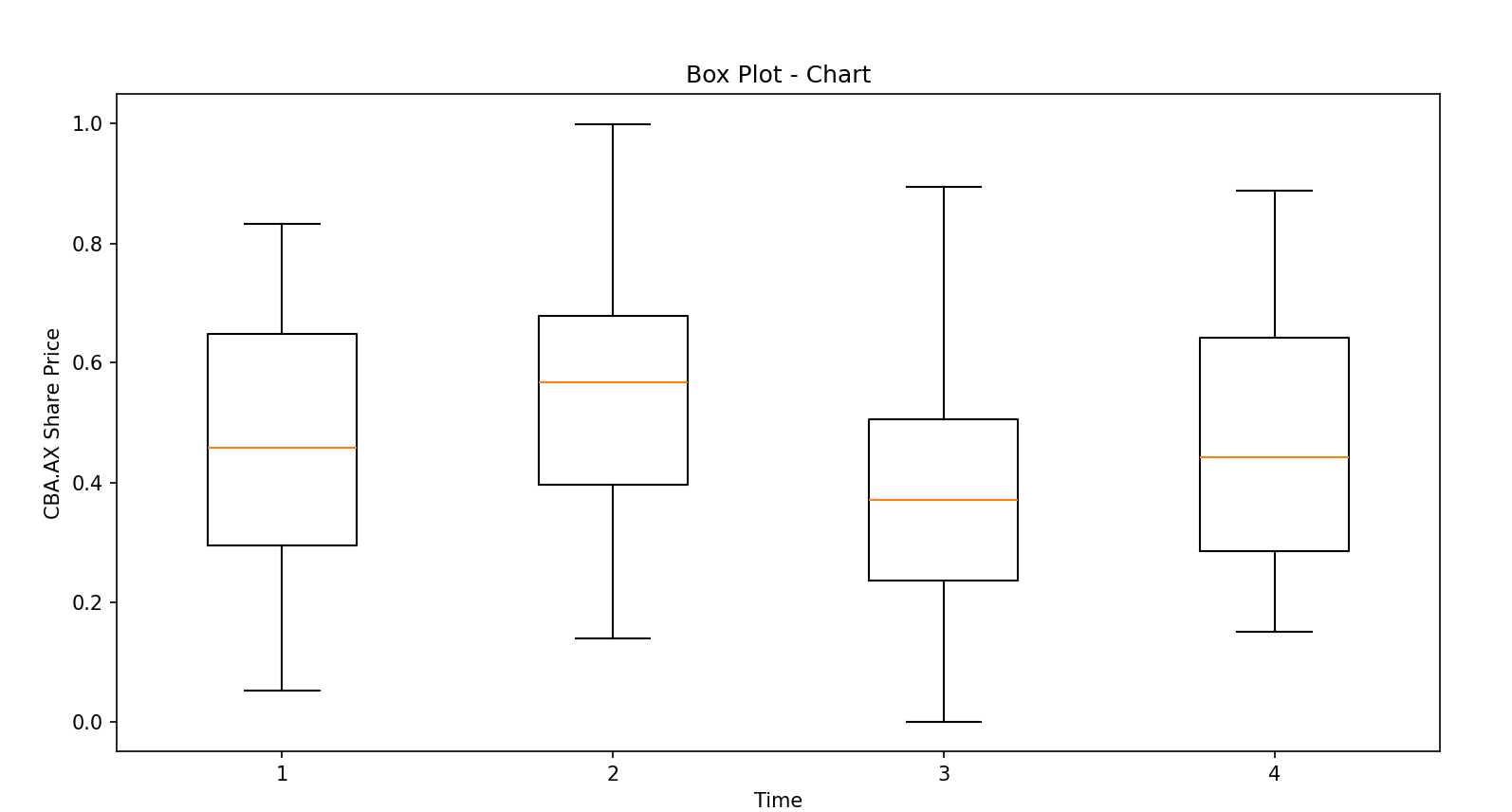
Description automatically generated

**Results**

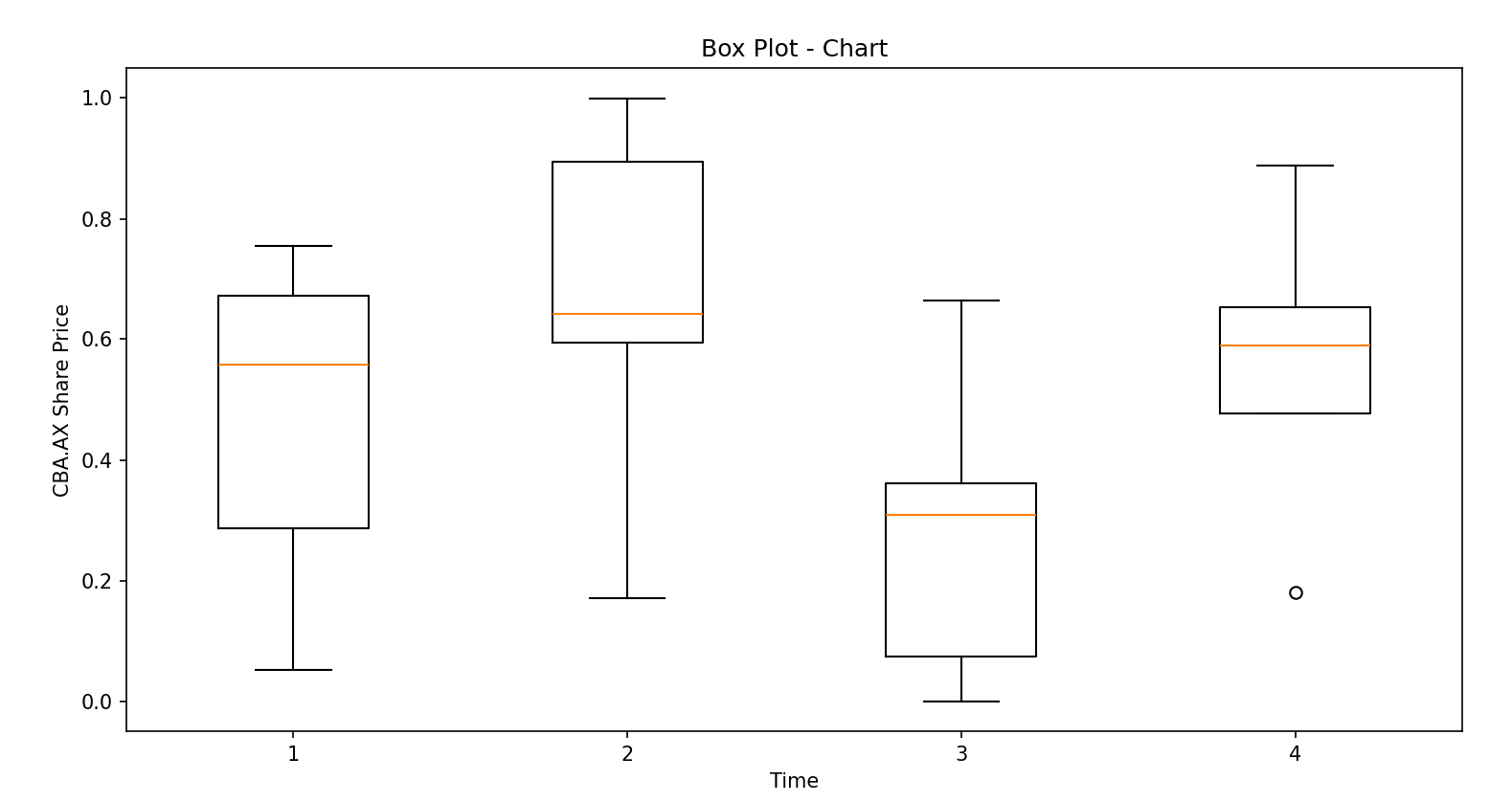
With trading\_days\_agg set to False



With it set to True, with n being 5



With n set to 20



Example with the TRAIN\_START\_DATE set to 2023 (for demonstration)

A graph of a number of boxes

Description automatically generated with medium confidence

**Useful resources:**

<https://www.earthdatascience.org/courses/use-data-open-source-python/use-time-series-data-in-python/date-time-types-in-pandas-python/resample-time-series-data-pandas-python/>

<https://www.geeksforgeeks.org/box-plot-in-python-using-matplotlib/>

<https://wellbeingatschool.org.nz/information-sheet/understanding-and-interpreting-box-plots>

<https://www.coinbase.com/en-au/learn/tips-and-tutorials/how-to-read-candlestick-charts>

**File path:** *project/venv/scripts/stock\_prediction3.py*