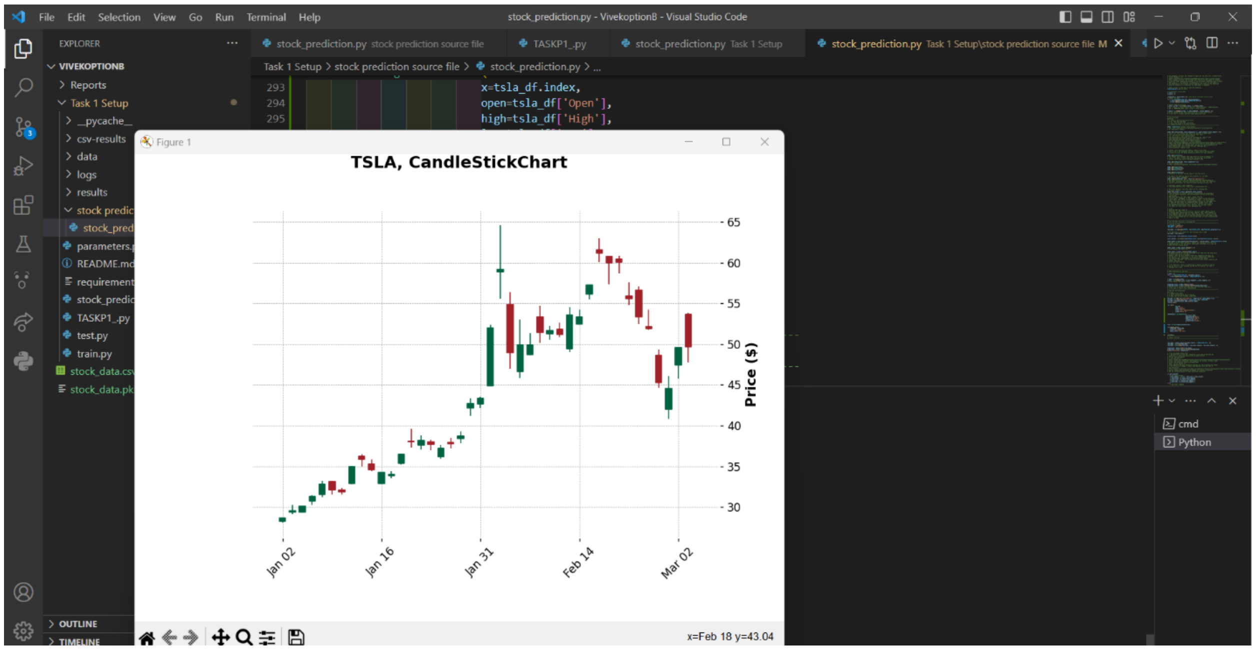
Option B - Task 3: Data processing 2

Starting with the first task in which we have to build a candlestick chart, the following provides you a clear idea of how it looks like:

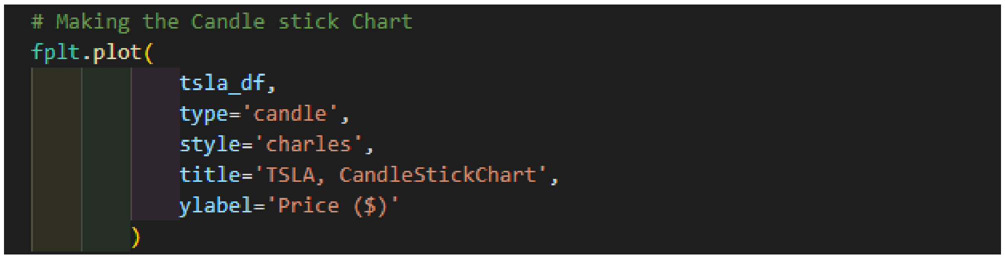


```
x=tsla_df.index,
open=tsla_df['Open'],
high=tsla_df['High'],
```

This data has been saved and made into a csv file called as “stock_data.csv” and then passed into a variable.

```
# creating a variable and storing the data
tsla_df = pd.read_csv('stock_data.csv', index_col= 0 ,parse_dates= True)
dt_range = pd.date_range(start = TRAIN_START, end = TRAIN_END)
tsla_df = tsla_df[tsla_df.index.isin(dt_range)]
tsla_df.head()
# Making the Candle stick Chart
```

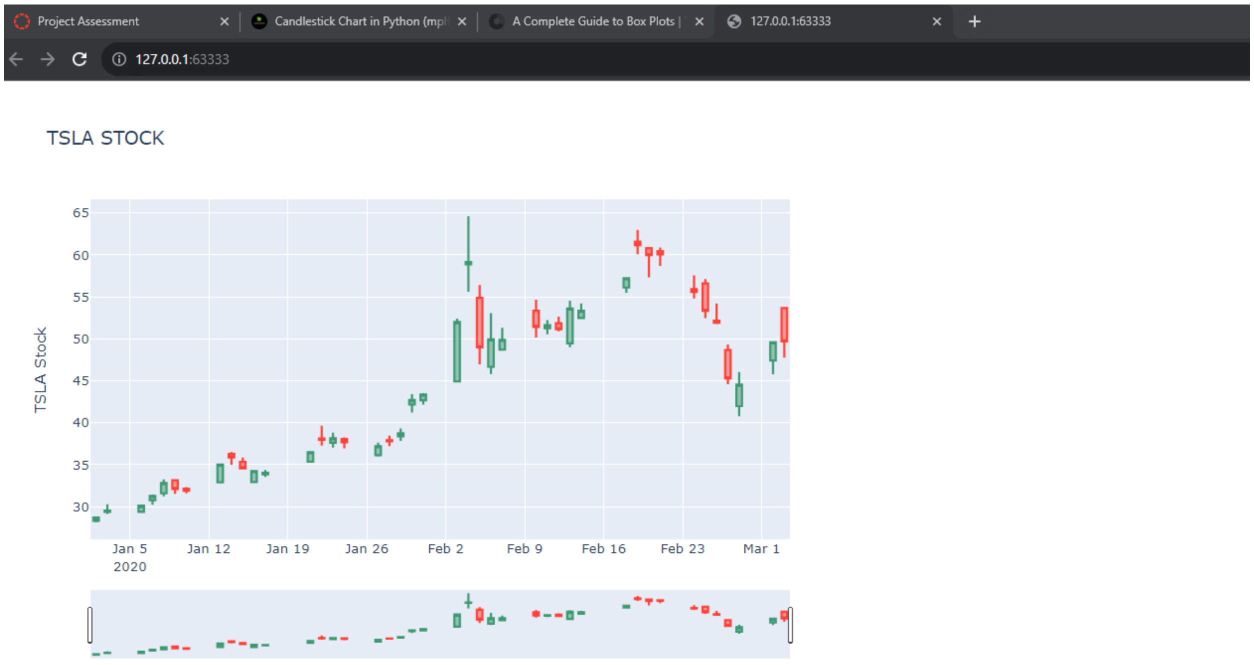
We then move onto using the function to make the candlestick Chart. I imported the mplfinance library to make this work after installing it from pip. There were different types of charts that I discovered like “Charles”, “mike”, “nightclouds” etc. which gives us different types of prebuilt custom charts.

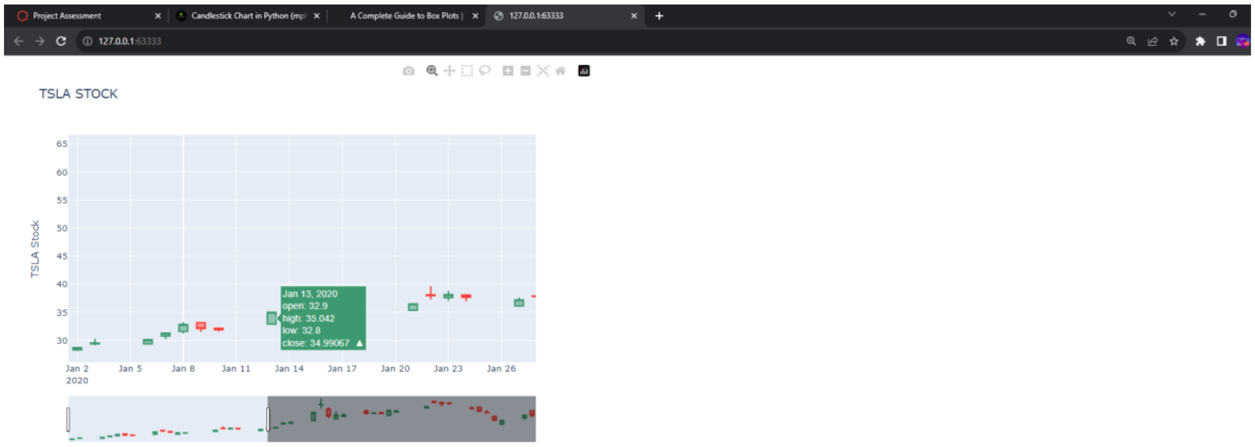


```
# Making the Candle stick Chart
fplt.plot(
            tsla_df,
            type='candle',
            style='charles',
            title='TSLA, CandleStickChart',
            ylabel='Price ($)'
        )
```

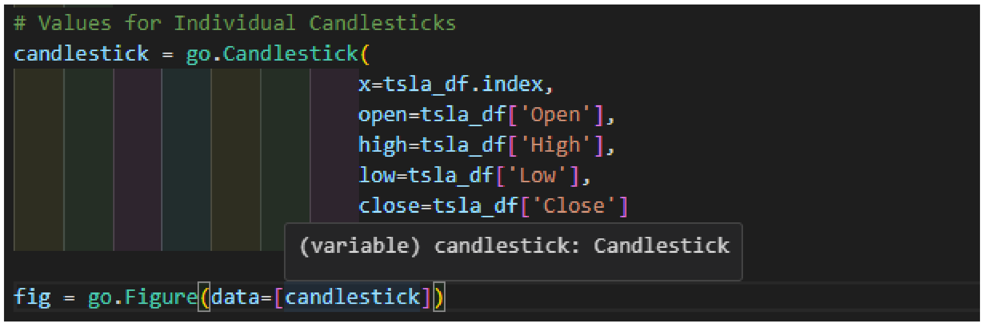
This function will give us a Candlestick Chart. Now some things to keep in mind is that I have used the start date and end date for only 2 months because feeding it more value not only consumes more time to train but also it makes the chart super messy. As I referred to the lecture present in the material given, we can also take the last 30 or maybe 50 values to show in the chart.

Moving on to the Second part of the requirement where we need to express the info in the individual candlesticks would look something like this:





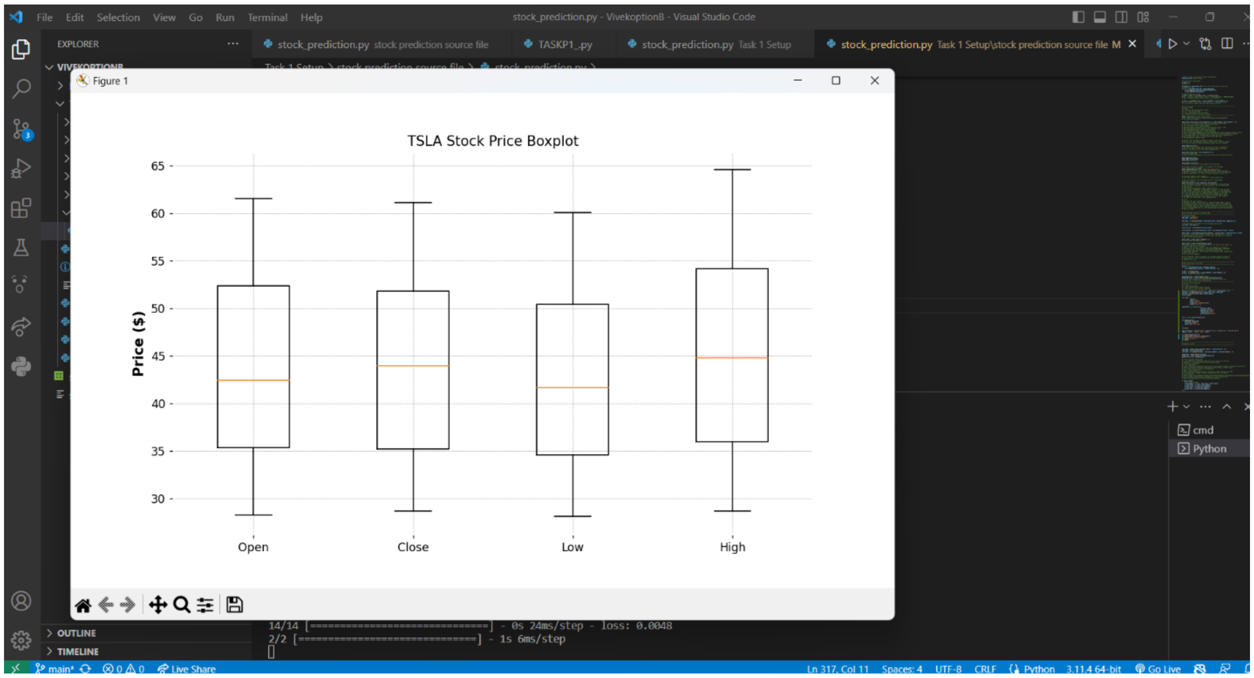
We can adjust the info with a slider that will only show the relevant info according to the number of n trading days that we want to see.



```
# Values for Individual Candlesticks
candlestick = go.Candlestick(
                            x=tsla_df.index,
                            open=tsla_df['Open'],
                            high=tsla_df['High'],
                            low=tsla_df['Low'],
                            close=tsla_df['Close']
                        (variable) candlestick: Candlestick

fig = go.Figure(data=[candlestick])
```

This is the function which was refereed from https://coderzcolumn.com/tutorials/data-science/candlestick-chart-in-python-mplfinance-plotly-bokeh

And at last we need to make a boxplot chart , I Used the same dataset I used for the other chart and referred to https://www.educative.io/answers/how-to-make-a-boxplot-in-pandas to make the function which results in something like this:



```
14/14 [==============================] - 0s 24ms/step - loss: 0.0048
2/2 [==============================] - 1s 6ms/step
```

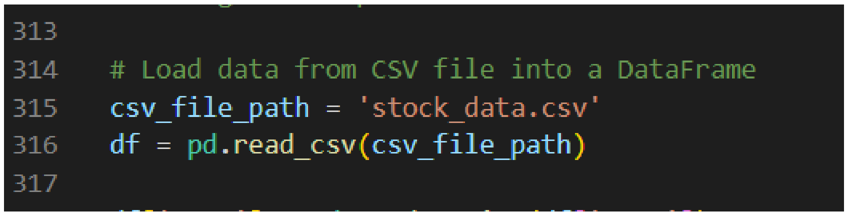
Version corrections:

I have done some corrections because I was not sure for the boxplot but after the class that I have attended I landed upon a solution for the boxplots because the boxplots has some issues.

The issues I faced includes the wrong data being read for the output.

As mentioned in the figure above , we can see only 4 boxplots being plotted for the whole dataset where we need the data for n number of days.

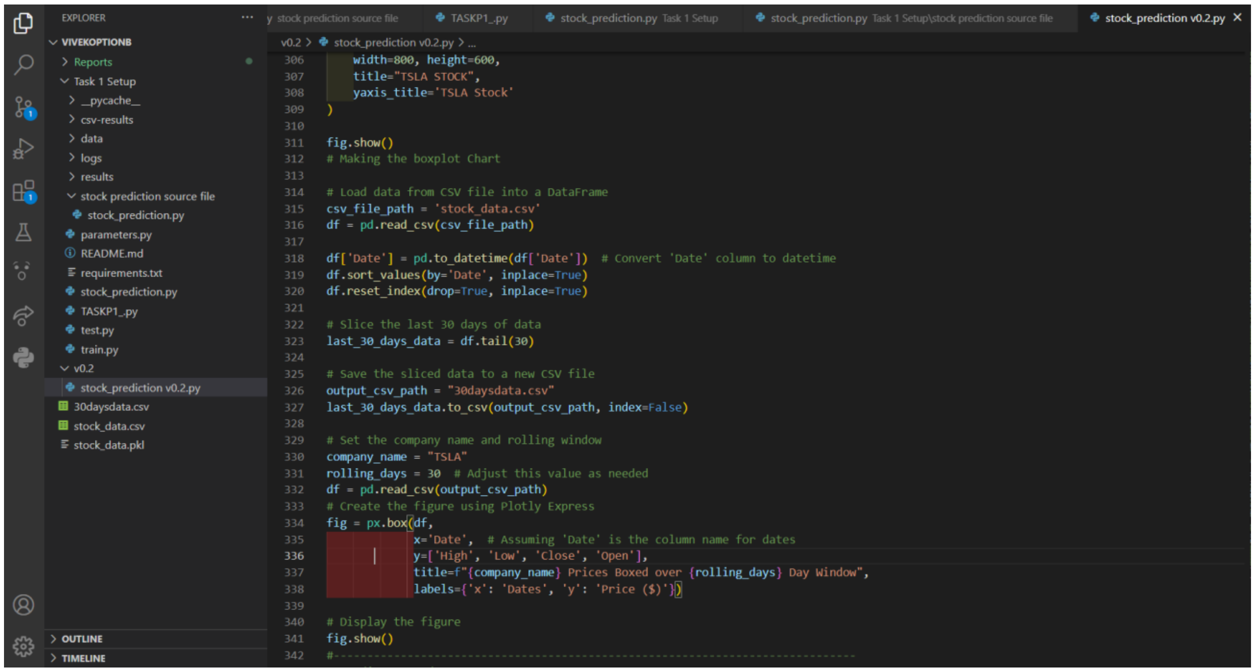
So to correct that I did some research online and came to the conclusion I should make the code read the values from the csv file. I corrected the code and instead of reading the first 4 columns now it is reading the whole dataset.



```

    # Load data from CSV file into a DataFrame
    csv_file_path = 'stock_data.csv'
    df = pd.read_csv(csv_file_path)

```

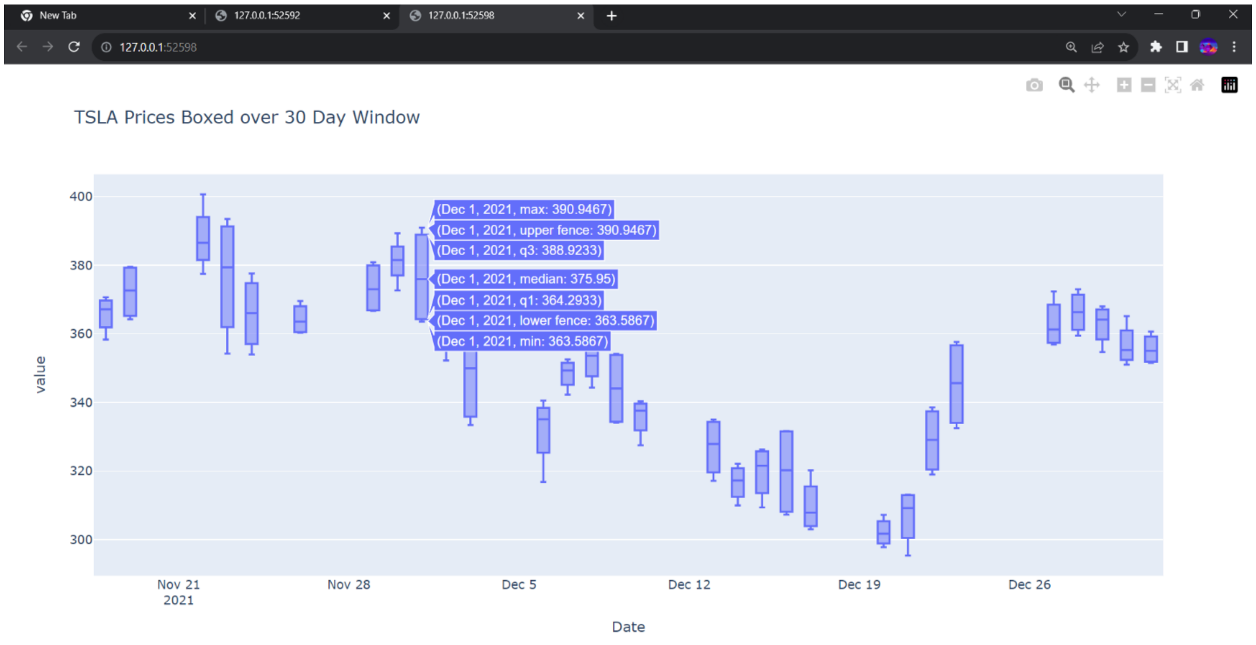
The boxplot was really messy so I had to cut down the data to show only a reasonable data. I used the slice down function to slice down the data. Thanks to stack overflow where I took the reference https://stackoverflow.com/questions/48423935/slicing-rows-from-csv-file

After some trying I managed to somehow alter the data and come up with a solution.



```python
    width=800, height=600,
    title="TSLA STOCK",
    yaxis_title='TSLA Stock'
)

fig.show()
# Making the boxplot Chart

# Load data from CSV file into a DataFrame
csv_file_path = 'stock_data.csv'
df = pd.read_csv(csv_file_path)

df['Date'] = pd.to_datetime(df['Date'])  # Convert 'Date' column to datetime
df.sort_values(by='Date', inplace=True)
df.reset_index(drop=True, inplace=True)

# Slice the last 30 days of data
last_30_days_data = df.tail(30)

# Save the sliced data to a new CSV file
output_csv_path = "30daysdata.csv"
last_30_days_data.to_csv(output_csv_path, index=False)

# Set the company name and rolling window
company_name = "TSLA"
rolling_days = 30  # Adjust this value as needed
df = pd.read_csv(output_csv_path)
# Create the figure using Plotly Express
fig = px.box(df,
             x='Date',  # Assuming 'Date' is the column name for dates
             y=['High', 'Low', 'Close', 'Open'],
             title=f"{company_name} Prices Boxed over {rolling_days} Day Window",
             labels={'x': 'Dates', 'y': 'Price ($)'})

# Display the figure
fig.show()
#------------------------------------------------------------------------
```

This is the final output where we can see that the current file was "stockdata.csv" and I cut some data and then placed it in a new csv file so that it only shows the last 30 days. I used plotly library to make the boxplot work because it was simple and straight to the point. Problem solved and the final out put looks like this :



In conclusion, we used the certain libraries that come out from mathplotlib and mplfinance , which helps us to visualize the data and enhanced how we look at the data That we have predicted. After some challenges and obstacles I think all the requirements are met and I am open to feed back. Thanks for your time :)