Financial Data Science - Assignment 2: 3rd March, 2023

Author: Gavin Connolly
Student Number: 18308483

Module: Financial Data Science

Academic Year - 2022/23
Submitted: 3rd March, 2023

Q1:

You have been assigned a stock ticker in the file ClassList.csv

```
student_number = 18308483
class_list = pd.read_csv("ClassList1.csv", sep=",")

ticker_name = class_list.loc[class_list["Student ID"] == student_number, "Assigned Ticker"].item()
yticker = yf.Ticker(ticker_name)
```

Q2:

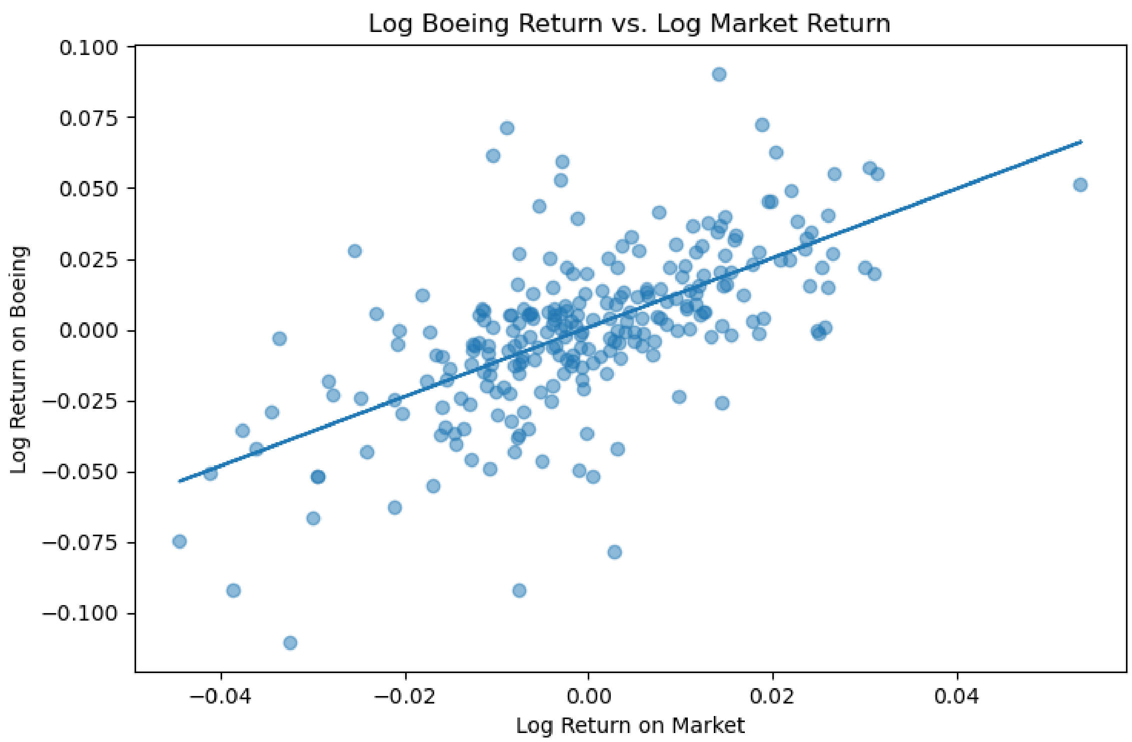
Using the yfinance and mpl packages (or any alternative packages you prefer) download the last year of price information for your assigned stock and for the S&P 500 and create log returns.

```
BA = yticker.history(period="1y")
SPY = yf.Ticker('SPY').history(period="1y")

# Calculate log returns
BA['Log Return'] = np.log(BA['Close'].pct_change()+1)
SPY['Log Return'] = np.log(SPY['Close'].pct_change()+1)
```

Q3:

Create a Scatter Plot of your stock returns Vs. market returns.

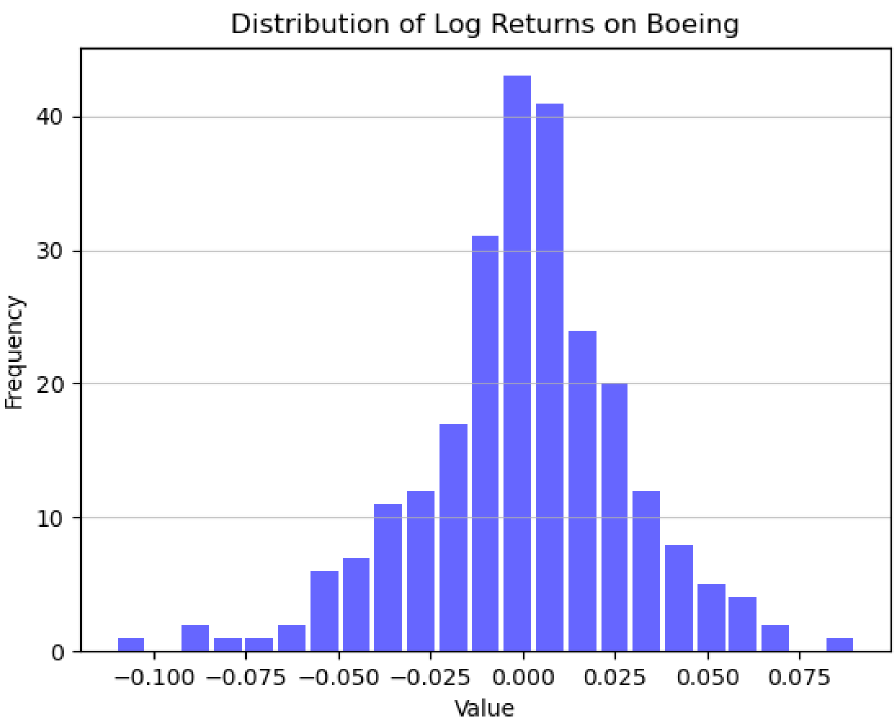
```
# Fit trend line
fitted = np.polyfit(SPY['Log Return'].iloc[2:], BA['Log Return'].iloc[2:], 1)
trend_line = np.poly1d(fitted)

# Plot results
plt.scatter(SPY['Log Return'], BA['Log Return'], alpha=0.5)
plt.plot(SPY['Log Return'], trend_line(SPY['Log Return']))

plt.xlabel("Log Return on Market")
plt.ylabel("Log Return on Boeing")
plt.title("Log Boeing Return vs. Log Market Return")
plt.show()
```



Q4:

Plot a histogram of your stock returns.

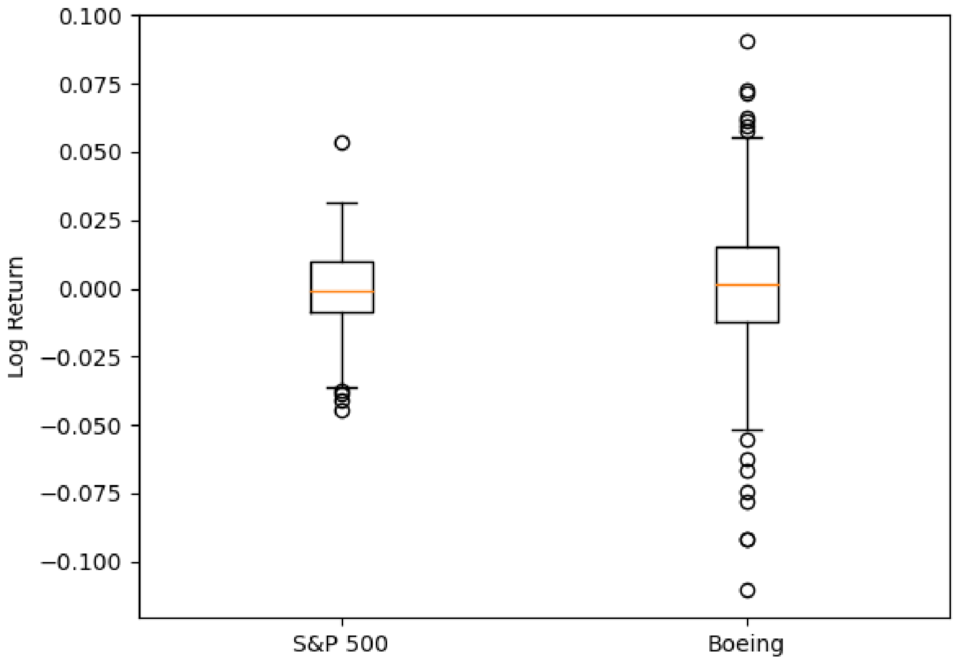
```
n, bins, patches = plt.hist(x=BA['Log Return'], bins='auto', color='b',
                            alpha=0.75, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Distribution of Log Returns on Boeing')
plt.show()
```



Q5:

Compare Box Plots of your stock returns and the market returns.

```
box_data = [SPY["Log Return"].values[2:], BA["Log Return"].values[2:]]

plt.boxplot(box_data)
plt.xticks([1, 2], ['S&P 500', 'Boeing'])
plt.ylabel("Log Return")
plt.show()
```

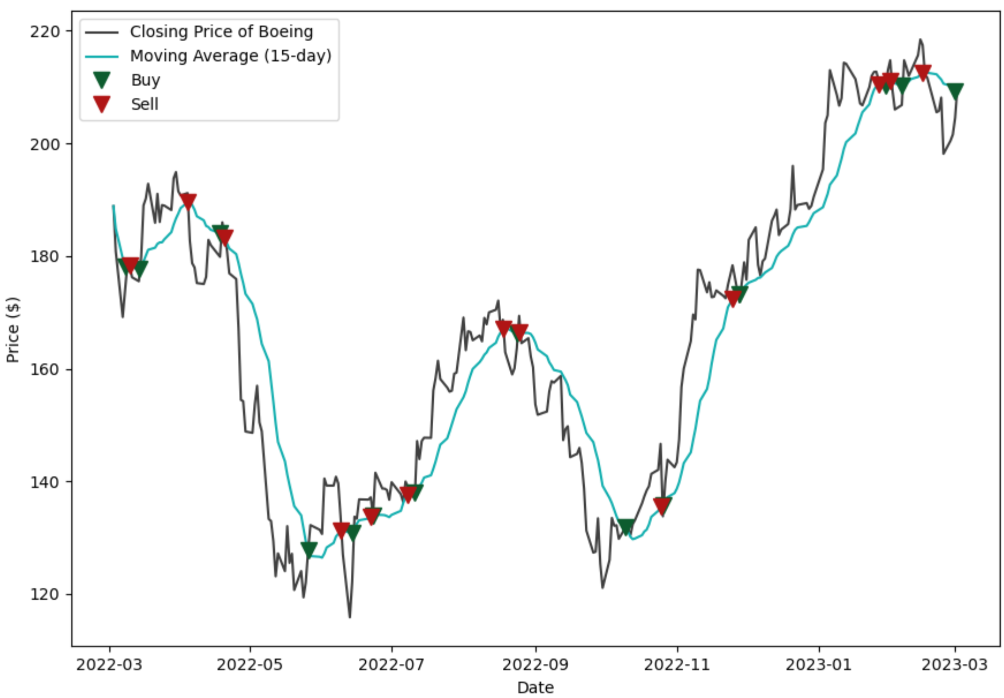


We see that Boeing's returns are much more dispersed than that of the S&P 500. This is in line with what we would likely expect, as the S&P is comprised of a diverse range of companies, meaning that the specific risk associated with any individual company will have less of an impact on its overall return. Boeing's median return is roughly in line with if not slightly higher than the S&P median, meaning its average return is roughly the same. We see a lot more outliers far from the upper/lower quantiles of the whiskers of the plot when looking at Boeing's individual returns, with this being especially notable in the abnormally low returns observed during the sample period. We see log returns of -10%, whereas the S&P's greatest loss is under 5%.

Q6:

Plot a moving average indicator alongside the stock price - to flag buy/sell signals.

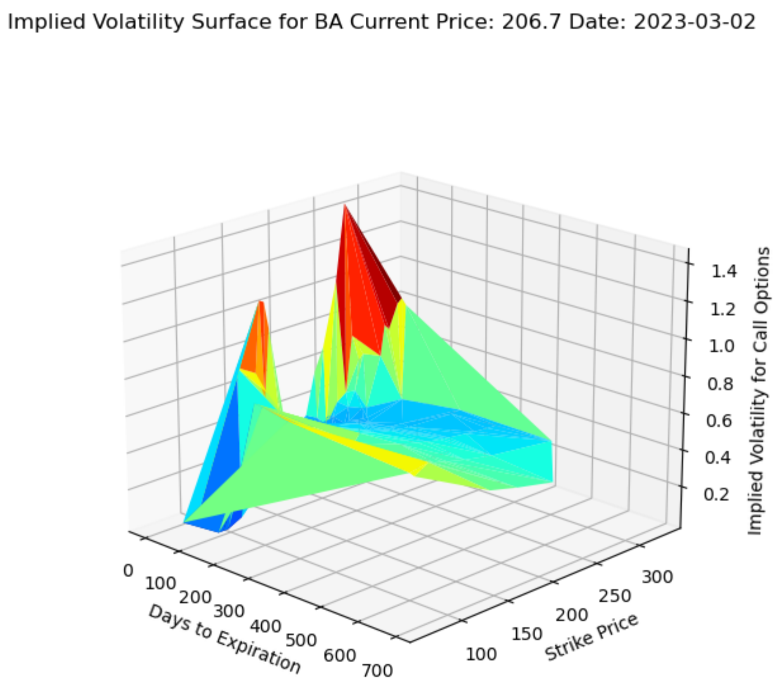
```
BA["Price MA 15"] = BA["Close"].rolling(window = 15, min_periods = 1).mean()
BA["Position"] = (BA["Price MA 15"] - BA["Close"]).apply(np.sign)
BA["Change in Position"] = (BA["Position"]-BA["Position"].shift(-1)).apply(np.sign)
BA.iloc[0,-1] = 0

# Plot Closing Price & MA
plt.plot(BA["Close"], label = 'Closing Price of Boeing', color = '#404040')
plt.plot(BA["Price MA 15"], label = "Moving Average (15-day)", color = '#15b0b0')

# Plot Buy/Sell Markers
plt.plot(BA[BA['Change in Position'] == 1].index,
         BA["Price MA 15"][BA['Change in Position'] == 1],
         'v', markersize = 10, color = '#0d5c2f', label = 'Buy')
plt.plot(BA[BA['Change in Position'] == -1].index,
         BA["Price MA 15"][BA['Change in Position'] == -1],
         'v', markersize = 10, color = '#b01515', label = 'Sell')

plt.xlabel('Date')
plt.ylabel('Price ($)')
plt.legend()
plt.show()
```



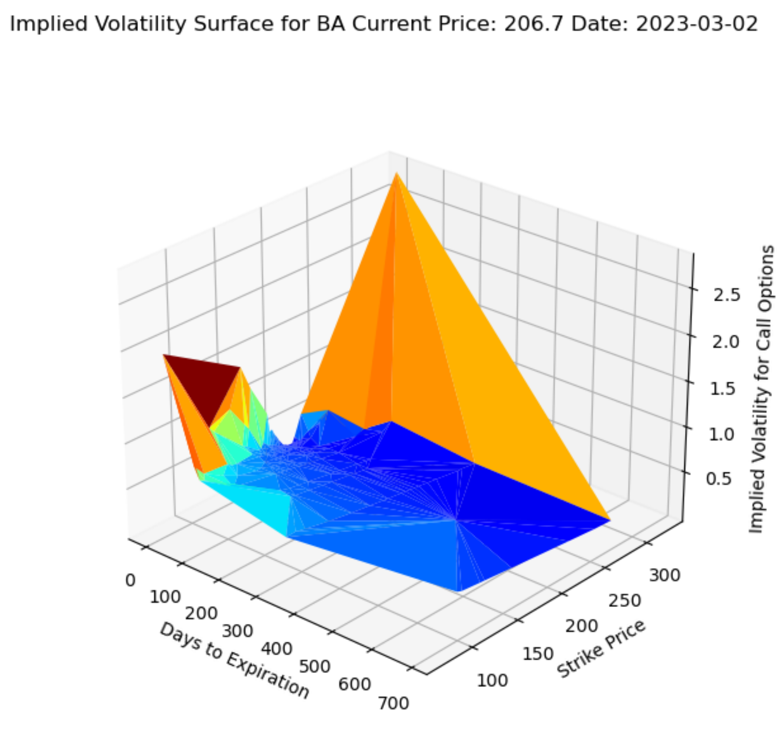
Q7:

Pull all incoming option data for your stock from yahoo finance and plot a 3D implied volatility surface

```
import yfinance as yf
import pandas as pd
import datetime as dt
from matplotlib import cm
from matplotlib import pyplot as plt

def options_chain(symbol):

    tk = yf.Ticker(symbol)
    # Expiration dates
    exps = tk.options

    # Get options for each expiration
    options = pd.DataFrame()
    for e in exps:
        opt = tk.option_chain(e)
        opt = pd.concat([pd.DataFrame(), opt.calls, opt.puts])
        opt['expirationDate'] = e
        options = pd.concat([options, opt], ignore_index=True)

    # Bizarre error in yfinance that gives the wrong expiration date
    # Add 1 day to get the correct expiration date
    options['expirationDate'] = pd.to_datetime(options['expirationDate']) + dt.timedelta(days = 1)
    options['dte'] = (options['expirationDate'] - dt.datetime.today()).dt.days

    # Boolean column if the option is a CALL
    options['CALL'] = options['contractSymbol'].str[4:].apply(lambda x: "C" in x)

    options[['bid', 'ask', 'strike']] = options[['bid', 'ask', 'strike']].apply(pd.to_numeric)
    options['mark'] = (options['bid'] + options['ask']) / 2 # Calculate the midpoint of the bid-ask

    options = options.drop(columns = ['contractSize', 'currency', 'change', 'percentChange', 'lastTradeDate', 'lastPrice'])
    return options

options_data = options_chain(ticker_name)

options_data = options_data[(options_data.volume > 10) & (options_data.CALL == True)]

# generate x,y,z values
xaxis=options_data['dte'] #days to expiry
yaxis=options_data['strike']
zaxis=options_data['impliedVolatility']
fig4 = plt.figure(2)
ax = fig4.add_subplot(111, projection='3d')
ax.plot_trisurf(xaxis, yaxis, zaxis, cmap=cm.jet)
ax.set_xlabel('Days to Expiration')
ax.set_ylabel('Strike Price')
ax.set_zlabel('Implied Volatility for Call Options')
plt.suptitle('Implied Volatility Surface for %s Current Price: %s Date: %s' %
                 (ticker_name, '{0:.4g}'.format(BA['Close'].values[-1]), dt.date.today()))

plt.show()
```



Implied Volatility Surface calculated from Call Options



Implied Volatility Surface calculated from Put Options