Stock Prediction

August 8, 2019

Stock Price Prediction

Introduction

The goal is to predict the future stock price of a selected equity in the Australia Stock Exchange (ASX) using Machine Learning Regression Algorithm(s). The data set of the stock prices will be taken from Alpha Vantage (www.alphavantage.co), which is a "leading provider of free APIs for realtime and historical data on stocks, forex (FX), and digital/crypto currencies". The stock in focus today will be Commonwealth Bank of Australia (ASX: CBA) (Alpha Vantage: cba.ax) as we will use 20 years of its stock prices and the each price is the price of CBA at the end of each trading day (closing price). The csv file of these prices is already downloaded and requires an understanding of the API documentation and a API key (free) inorder to retrieve it yourself.

Required Libraries

These following Python libraries are required for this project:

```
In [1]: # import libraries
        import pandas as pd
        from mpl_finance import volume_overlay3, candlestick_ohlc
        import matplotlib.pyplot as plt
        import matplotlib.dates as mdates
        import numpy as np
        from sklearn.linear_model import LinearRegression
        from sklearn.model_selection import train_test_split
```

Data Wrangling Process

```
In [2]: # load up csv file of the CBA 20-year stock prices
        df = pd.read_csv("daily_CBA.ax.csv")
        print("First five rows of the dataframe")
        print(df.head())

        # converting timestamp to pandas datetime format
        df['timestamp'] = pd.to_datetime(df['timestamp'])
        df["timestamp"] = df["timestamp"].apply(mdates.date2num)

        # check the variables data types
        print("\nData types of the dataframe")
        print(df.dtypes)

        # drop any rows that have all 0 values
        df = df[df['volume'] > 0]
```

```
First five rows of the dataframe
    timestamp   open   high     low  close   volume
0  2019-08-01  81.50  82.35  81.335  81.92  3129406
1  2019-07-31  83.09  83.43  82.300  82.30  3304674
2  2019-07-30  83.49  83.70  83.000  83.40  3120288
3  2019-07-29  82.80  83.45  82.380  83.25  3280357
4  2019-07-26  82.99  83.29  82.530  82.59  6715835

Data types of the dataframe
timestamp    float64
open         float64
high         float64
low          float64
close        float64
volume         int64
dtype: object
```

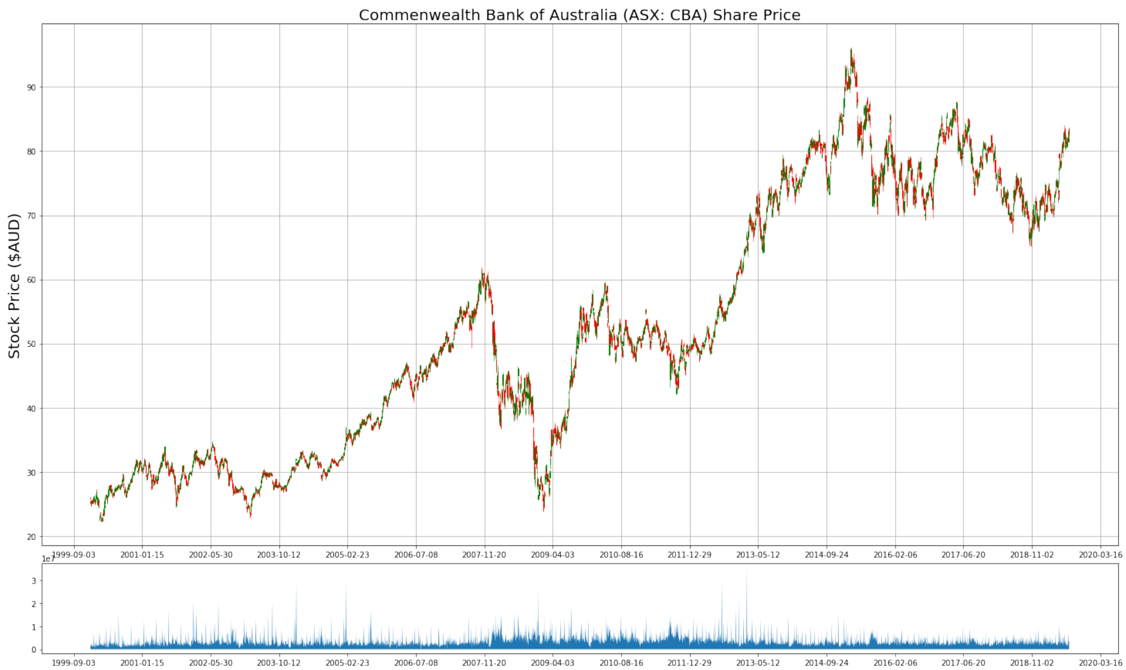
Comment: Here, it looks like 6 variables we have available which are timestamp, open, high, low, close and volume. We will use the timestamp as the labels of the stock price. The close will be the response as it is the closing price of that trading day while the rest of the variables will be the predictors. The code shows the csv file is read into a pandas dataframe which allows to manipulate and transform the data in the future. Lastly, the data looks fine enough such that we do not have to do any data cleansing as most of stock data is numerical in nature and the data types are suitable for the values the represent.

Exploratory Data Analysis

Here, we will use the mpl_finance library in Python for the graphical analysis and since this a finance problem, it would be suitable to use financial graphs, such as a candle stick graph. Also, since the open, high, low and close variables are referrring to the price of the CBA stock, it would be wise to see the relationship between these and the volume traded as volume is the amount of shares that switched hands on a certain day of trading.

```
In [3]: # plot candlestick plot of the last 20 years of the CBA price
        plt.figure(figsize = (25,15))
        fig = plt.plot()

        # CBA stock price data
        ax1 = plt.subplot2grid((6,1), (0,0), rowspan = 5, colspan = 1)
        ax2 = plt.subplot2grid((6,1), (5,0), rowspan = 1, colspan = 1, sharex = ax1)

        quotes = df[['timestamp','open', 'high', 'low', 'close']].values
        candlestick_ohlc(ax1, quotes, width = 1, colorup='green', colordown='red')
        ax1.set(xlabel = "Date", ylabel = "Stock Price ($AUD)", title = "Commenwealth Bank of A
        ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
        plt.locator_params(axis = 'x', nbins = 20)
        ax1.title.set_fontsize(20)
        ax1.xaxis.label.set_fontsize(20)
        ax1.yaxis.label.set_fontsize(20)
```

```
ax1.grid(True)

# CBA volume traded data
ax2.fill_between(df['timestamp'].values, df['volume'].values, 0)

# show combined data
plt.show()
```



Comment: So, it seems that regardless of the price surging or crashing, we see that the volume always increases if either of these two occurs and especially in a short timeframe. For example, we see during the Global Financial Crisis (GFC) the CBA stock crashed and the volume increased. Similarly, after the GFC came the current bull run which lead to the stock gaining back its price, surging past its all time high during 2007 and here again we see the surged in trading volume. Thus, by inspection it seems that during crashes and surges of the stock it looks like that the volume tends to follow the direction the stock price is heading. However, before the GFC we can see the there is still peaks of volume of the stock being traded. All this maybe attributed to the convenience of trading/investing as it is clear that in the 21th century we have more access to the stockmarket than in the past. Therefore, this may have lead to the increase in volume being traded as more people have access to the stock market rather than relying on stock brokers.

Machine Learning Regression Algorithm(s)

Multiple Linear Regression

For our multiple linear regression model to predict the future price of the stock we would first want to choose our features and the response. I will use the 'open' and 'volume' as the features and the 'close' price will be our response because the goal to is to predict the future closing price of the CBA stock.

```
In [4]: # set timestamp as the index
```

```python
df.index = df['timestamp']
df = df.drop(['timestamp'], axis = 1)
```

In [5]:
```python
# get all the features
stock_x = df[['open', 'volume']]
stock_y = df['close']

# split features into train/test data sets and
# split response intro train/test data sets corresponding with the above features
x_train, x_test, y_train, y_test = train_test_split(stock_x, stock_y, test_size = 0.30

# create linear regression model and fit the training data
stock_lm = LinearRegression().fit(x_train, y_train)

# preserve order
x_train = x_train.sort_index(axis = 0)
y_train = y_train.sort_index(axis = 0)
x_test = x_test.sort_index(axis = 0)
y_test = y_test.sort_index(axis = 0)

# test the linear model using the test data
stock_lm_predicted = stock_lm.predict(x_test)
```

Now, after training the multiple linear regression model using our training data set and using the testing data set to feed the model, we get the predicted values of the stock's price, corresponding to each of the testing data's timestamps. Now, we use the actual stock price and plot it against the predicted stock price to each corresponding time stamp to see how our model fairs. Also, we will include the 10 day moving average as a comparison to our model in order to see which performs better using the current testing data set.

In [6]:
```python
# generate 10 day simple moving average
def simplemovingaverage(interval, window_size):
    window = np.ones(int(window_size))/float(window_size)
    return np.convolve(interval, window, 'same')
sma = simplemovingaverage(y_test, 10)

# plot the actual values vs. the predicted values to access our model
plt.figure(figsize = (25,15))
plt.scatter(x_test.index[11:110], y_test.values[11:110], color = 'green')
plt.scatter(x_test.index[11:110], stock_lm_predicted[11:110], color = 'red')
plt.plot(x_test.index[11:110], y_test.values[11:110], color = 'green', linestyle='solic
plt.plot(x_test.index[11:110], stock_lm_predicted[11:110], color = 'red', linestyle='sc
plt.plot(x_test.index[11:110], sma[11:110], color = 'blue', linestyle='solid')
plt.title('Actual vs. Fitted CBA stock price using the testing data set', fontsize = 2C
plt.xlabel('Trading Date', fontsize = 20)
plt.ylabel('Stock Price ($AUD)', fontsize = 20)
plt.legend(['Actual Stock Price','Predicted Stock Price', '10 Day SMA'], loc='upper lef
plt.locator_params(axis = 'x', nbins = 20)
```

```
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
plt.show()
```

Comment: By using an out of sample analysis, we plot the predicted values against the actual values in the testing data set in order to see how our model faired when we use the training data set to train it. Here, it is clear that the predicted values spit out by the model, were pretty accurate when compared to the actual values. This shows that the Multiple Linear Regression Algorithm has promise when using the features of 'open' and 'volume' to predict the 'close' price of the CBA stock. Furthermore, when compared against the 10 day simple moving average, the multiple linear regression model still has the better fit as it closely follows the points of the correspoding stock prices.

Conclusion

Here, it is clear that the multiple regression algorithm predicts close the the true share price in the out of sample analysis, however the accuracy is still not enough as we may have to test for overfitting with new data. Here, we would also factor in other qualitative data such as news on the CBA stock and do sentiment analysis on them to determine how traders/investors will feel about the stock. This would also apply to any company annoucements by CBA and we can do a sentiment analysis on the annoucements to check if this is positive news or negative which will also affect the price.