Journal of Electronic Research and Application, 2024, Volume 8, Issue 5
http://ojs.bbwpublisher.com/index.php/JERA
ISSN Online: 2208-3510
ISSN Print: 2208-3502

Using Python to Analyze Financial Big Data

Xuanrui Zhu*

Nanjing No.1 High School, Nanjing 210001, Jiangsu Province, China

Corresponding author: Xuanrui Zhu, rui_jules@163.com

Abstract: As technology and the internet develop, more data are generated every day. These data are in large sizes, high dimensions, and complex structures. The combination of these three features is the "Big Data" [1]. Big data is revolutionizing all industries, bringing colossal impacts to them [2]. Many researchers have pointed out the huge impact that big data can have on our daily lives [3]. We can utilize the information we obtain and help us make decisions. Also, the conclusions we drew from the big data we analyzed can be used as a prediction for the future, helping us to make more accurate and benign decisions earlier than others. If we apply these technics in finance, for example, in stock, we can get detailed information for stocks. Moreover, we can use the analyzed data to predict certain stocks. This can help people decide whether to buy a stock or not by providing predicted data for people at a certain convincing level, helping to protect them from potential losses.

Keywords: Big data finance; Big data in financial services; Big data in risk management; AI; Machine learning

Online publication: August 26, 2024

1. Introduction

Technological development and innovation have generated vast amounts of data, making it both accessible and valuable, especially in industries such as automation and beyond. Among all innovations, financial innovations are considered some of the fastest-emerging developments in financial services [4]. They encompass a wide range of areas, generating vast amounts of data every minute. Any loss or damage to this data could be catastrophic for the financial industry. Therefore, the importance of having access to and the ability to organize this data effectively is self-evident.

This article uses Python to draw candle charts to present the trends of input stock. The chart shows the rising and distinctly falling of that stock where users can easily know the past information of that stock, relating to whether it rises or falls, what is its lowest price, and its highest price on a certain day. Also, this article uses Python to build a model, or Artificial Intelligence (AI) to predict the trend of a stock. To achieve this goal, certain Python libraries are necessary, so Anaconda is used to build the program and display the code.

2. Methodology

(1) The program needs to collect data first before it can analyze it. It will retrieve the data from Yahoo Finance using “pandas_datareader” and “yfinance.”

(2) After obtaining the data, we can use “mplfinance” to create plots. These plots are candlestick charts, with various styles to choose from, such as “charles” or “classic” [5].

3. Detailed procedures

(1) Firstly, we need to download the libraries. Several ways to install them are provided.

```
conda install yfinance
conda install -c conda-forge yfinance
!pip install pandas_datareader
```

Figure 1. Installation of the libraries

(2) Import necessary libraries.

```
[2]: import numpy as np
     import pandas as pd
     from pandas_datareader import data
     import yfinance as yf
     yf.pdr_override()
```

Figure 2. Importing libraries

(3) Get the information about the stock. The information is from Yahoo Finance, and the example used is Microsoft Corporation (MSFT). This gives us a “DataFrame,” so we can look at the values directly.

```
[3]: msft = data.get_data_yahoo('MSFT', start='2023-01-01', end='2024-07-12')
     msft
```

```
[**********100%%**********]  1 of 1 completed
```

[3]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-01-03	243.080002	245.750000	237.399994	239.580002	236.609207	25740000
2023-01-04	232.279999	232.869995	225.960007	229.100006	226.259186	50623400
2023-01-05	227.199997	227.550003	221.759995	222.309998	219.553375	39585600
2023-01-06	223.000000	225.759995	219.350006	224.929993	222.140869	43613600
2023-01-09	226.449997	231.240005	226.410004	227.119995	224.303726	27369800
...	...	...	...	...	...	...
2024-07-05	459.609985	468.350006	458.970001	467.559998	467.559998	16000300
2024-07-08	466.549988	467.700012	464.459991	466.239990	466.239990	12962300
2024-07-09	467.000000	467.329987	458.000000	459.540009	459.540009	17207200
2024-07-10	461.220001	466.459991	458.859985	466.250000	466.250000	18196100
2024-07-11	462.980011	464.779999	451.549988	454.700012	454.700012	23064800

382 rows × 6 columns

Figure 3. Stock information

However, as shown in the figure, Anaconda does not show all the stock information form 2023-01-01 to 2024-07-12. It is because the information is too much and displaying them all is not necessary. Thus, Python only shows part of them. But at the bottom, we can see a line showing “382 rows × 6 columns.” This tells us the exact amount of data we have. The longer the time, the larger the chart. Additionally, we can use the head() and tail() functions to obtain certain lines of information. Details are shown below.

```
[4]: msft.head(5)
```

[4]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-01-03	243.080002	245.750000	237.399994	239.580002	236.609207	25740000
2023-01-04	232.279999	232.869995	225.960007	229.100006	226.259186	50623400
2023-01-05	227.199997	227.550003	221.759995	222.309998	219.553375	39585600
2023-01-06	223.000000	225.759995	219.350006	224.929993	222.140869	43613600
2023-01-09	226.449997	231.240005	226.410004	227.119995	224.303726	27369800

Figure 4. Get the first 5 lines of data

```
[5]: msft.tail(5)
```

[5]:

Date	Open	High	Low	Close	Adj Close	Volume
2024-07-05	459.609985	468.350006	458.970001	467.559998	467.559998	16000300
2024-07-08	466.549988	467.700012	464.459991	466.239990	466.239990	12962300
2024-07-09	467.000000	467.329987	458.000000	459.540009	459.540009	17207200
2024-07-10	461.220001	466.459991	458.859985	466.250000	466.250000	18196100
2024-07-11	462.980011	464.779999	451.549988	454.700012	454.700012	23064800

Figure 5. Get the last 5 lines of data

Several other operations can help analyze the stock information and do some statistical calculations.

```
[6]: msft.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 382 entries, 2023-01-03 to 2024-07-11
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       382 non-null    float64
 1   High       382 non-null    float64
 2   Low        382 non-null    float64
 3   Close      382 non-null    float64
 4   Adj Close  382 non-null    float64
 5   Volume     382 non-null    int64
dtypes: float64(5), int64(1)
memory usage: 20.9 KB
```

Figure 6. Check the MSFT dataset

```
[7]: msft.describe()
```

[7]:

	Open	High	Low	Close	Adj Close	Volume
count	382.000000	382.000000	382.000000	382.000000	382.000000	3.820000e+02
mean	349.182487	352.273771	346.111623	349.421650	347.643337	2.524547e+07
std	60.278310	60.279301	59.998974	60.263377	61.130859	1.002994e+07
min	223.000000	225.759995	219.350006	222.309998	219.553375	9.932800e+06
25%	312.732491	315.580009	310.199989	313.452515	311.451149	1.857320e+07
50%	339.000000	341.649994	335.585007	338.129990	335.885239	2.306490e+07
75%	406.104996	409.257492	403.307510	406.077499	404.672897	2.817890e+07
max	467.000000	468.350006	464.459991	467.559998	467.559998	7.847820e+07

Figure 7. Some statistical data was collected on the stock we chose in the period we specified.

```
[11]: import mplfinance as mpf
mpf.plot(
    msft,
    type='candle', style='charles',
    title='msft 2024',
    figratio=(10,6),
    mav=(20,40,100),
    volume=True,
)
```

Figure 8. Make a candlestick plot for the stock user-specified

We need to import the mplfinance tool first, then we choose the type and the style to use [6].

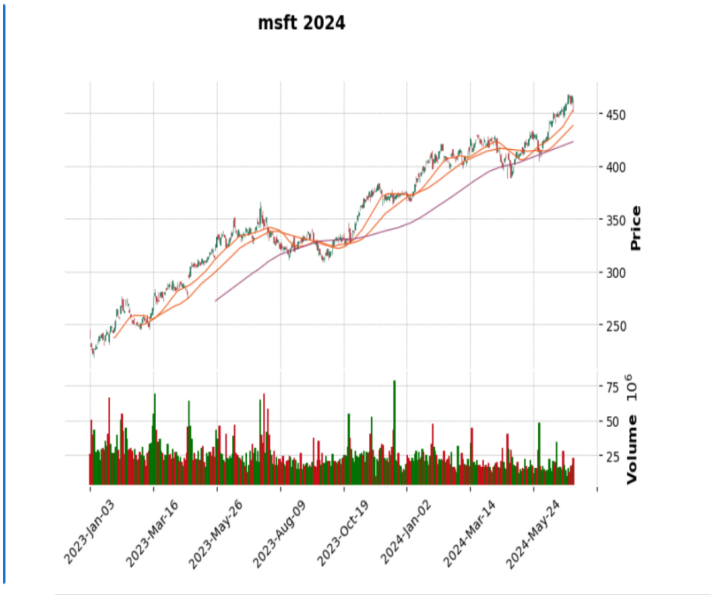

Figure 9. The graph plotted by Python

Another method to show information about the stock is to get the prices of the stock when it is open in a day [7].

```
[12]: msft['Open']

[12]: Date
      2023-01-03    243.080002
      2023-01-04    232.279999
      2023-01-05    227.199997
      2023-01-06    223.000000
      2023-01-09    226.449997
                       ...
      2024-07-05    459.609985
      2024-07-08    466.549988
      2024-07-09    467.000000
      2024-07-10    461.220001
      2024-07-11    462.980011
      Name: Open, Length: 382, dtype: float64
```

Figure 10. The prices for MSFT when it is open for sale in a day

In the output, Python still shows part of the results, the same as in **Figure 3**.

```
[13]: msft['Open'].rolling(5).mean()

[13]: Date
      2023-01-03           NaN
      2023-01-04           NaN
      2023-01-05           NaN
      2023-01-06           NaN
      2023-01-09    230.401999
                       ...
      2024-07-05    454.546002
      2024-07-08    457.241998
      2024-07-09    460.909998
      2024-07-10    462.513995
      2024-07-11    463.471997
      Name: Open, Length: 382, dtype: float64
```

Figure 11. 5 days moving average

This graph shows the average of the open prices for MSFT, the stock used as the example.

```
[15]: mpf.plot(
          msft,
          type='candle', style='classic',
          title='MTR 2024',
          figratio=(10,6),
          mav=(20,40,100),
          volume=True,
      )
```

Figure 12. Different styles chosen generate different outputs

There are many styles to choose from. We can check how many styles to use by using the function available_styles().

```
[17]: mpf.available_styles()

[17]: ['binance',
       'binancedark',
       'blueskies',
       'brasil',
       'charles',
       'checkers',
       'classic',
       'default',
       'ibd',
       'kenan',
       'mike',
       'nightclouds',
       'sas',
       'starsandstripes',
       'tradingview',
       'yahoo']
```

Figure 13. Available styles

4. Predicting stock prices and machine learning

There are four main learning methods and thirteen algorithms used commonly in the field of machine learning. For example, there are supervised learning, unsupervised learning, reinforcement learning in learning methods, and decision trees, Naive Bayes, and K-Means Clustering algorithms. This article will use several methods to predict the prices and evaluate the outcomes of each type of model.

4.1. Linear Regression model

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas_datareader import data
import yfinance as yf
from sklearn.model_selection import train_test_split
from sklearn import linear_model
yf.pdr_override()
```

Figure 14. Import necessary libraries

```
msft = data.get_data_yahoo('MSFT', start='2020-01-01', end='2022-07-01')
print(msft)
```

Figure 15. Get the stock we want

```
today = msft['High'].iloc[:len(msft)-1].reset_index(drop=True)
nextday = msft['Open'].iloc[1:].reset_index(drop=True)
```

Figure 16. Get feature words and remove unwanted features

```
today = np.array(today).reshape(-1,1)
nextday = np.array(nextday).reshape(-1,1)

X_train, X_test, y_train, y_test = train_test_split(
    today, nextday, test_size=0.3,
    shuffle=False,
)
```

Figure 17. Get the data we need and declare training and testing data.
Note: in this model, 80% of the data we obtained is used to train this model

```
model=LinearRegression()
model.fit(X_train,y_train)
```

Figure 18. Build a linear regression model and train the model

```
y_pred = model.predict(X_test)
mse=mean_squared_error(y_test,y_pred)
print(mse)
```

Figure 19. Use the model to predict the stock prices and compare the predicted results to the actual results

```
y_pred = model.predict(X_test)
mse=mean_squared_error(y_test,y_pred)
print(mse)
```

```
[**********100%%**********]  1 of 1 completed
24.85009398106079
```

Figure 20. Get the mean-squared-error (MSE) of the predicted results and the actual result

The mean-squared-error is calculated as the below equation:

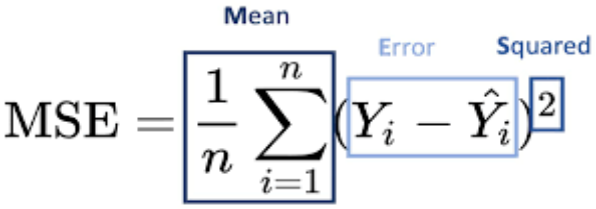

Figure 21. How to calculate the mean-squared-error

We can also visualize the accuracy of the model's predictions. If the model is predicting accurately, the dots on the following graph will be closer to the diagonal line of this graph. If the model is perfectly correct, then all dots will be on the same line and the MSE for this model will be zero. The code is:

plt.figure(figsize=(16,10))

plt.scatter(y_test, y_pred)

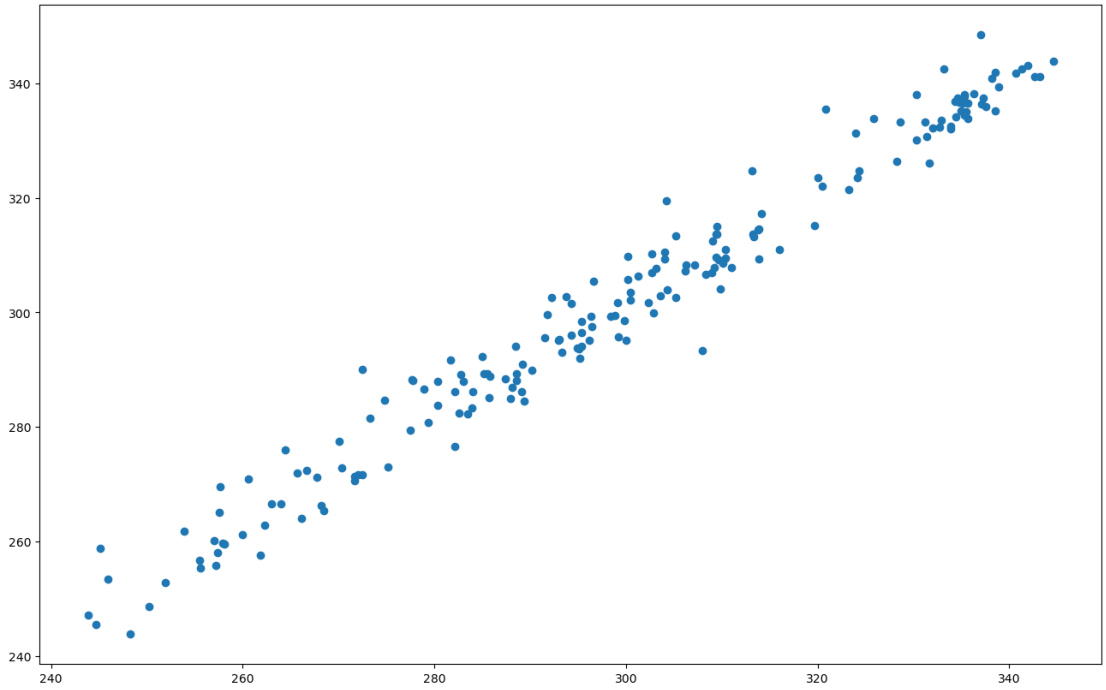

Figure 22. Plot the graph

The mean-squared-error of the predicted values and the actual value are quite big. This indicates that the model is not performing very well. To improve the accuracy of the model, we can give the model more data to train it or we can give it a different training and testing data sizes. Also, we can use more complicated models. For example, the Neural Network.

4.2. Neural Network

```
from sklearn.neural_network import MLPRegressor

regNN = MLPRegressor(
    hidden_layer_sizes=(256, 128, 64, 32),
    max_iter=5000,
    random_state=1,
  )
regNN = regNN.fit(X_train, y_train.reshape(-1))#feed in the training data to the model

y_pred_NN = regNN.predict(X_test)
y_pred_NN
```

Figure 23. Set up the model

Here we build a Neural Network model using the same training data X_train and y_train we prepared before and predict using the same test data X_test.

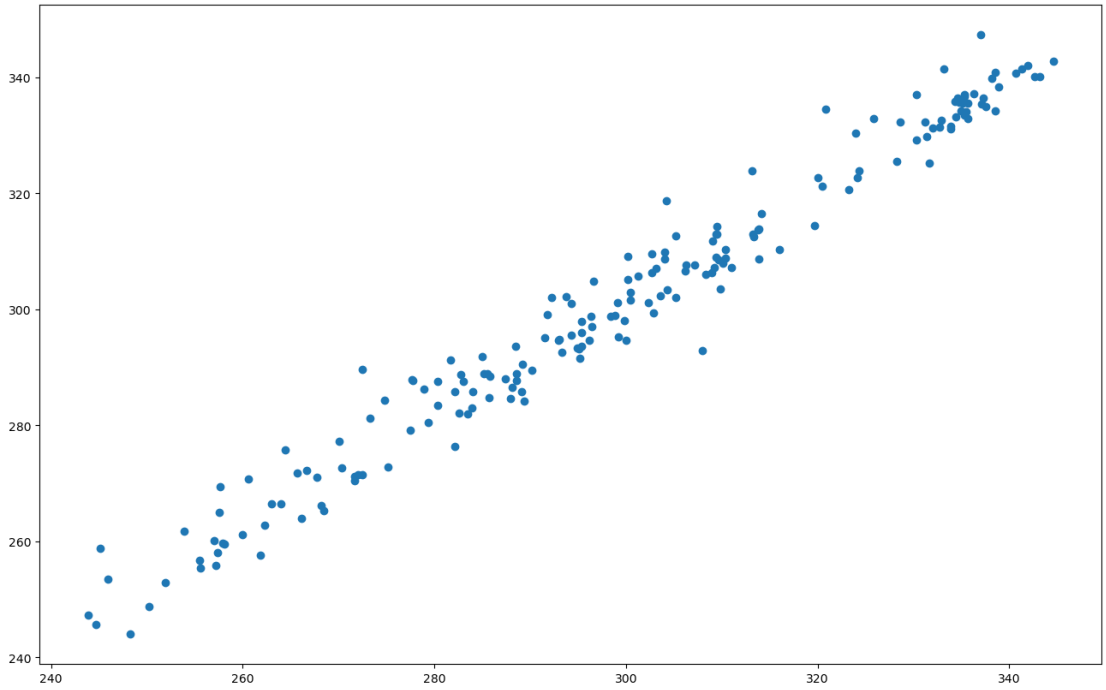

Figure 24. Show prediction accuracy by plotting

```
print(mean_squared_error(y_test, y_pred_NN))
```

```
23.15637863842539
```

Figure 25. Show the MSE of this model

Now, the MSE of this model has decreased compared to the Linear Regression model, proving that the Neural Network can perform better than the Linear Regression Model in this case.

5. Conclusion

The integration of big data into the financial sector has revolutionized the way financial institutions operate, make decisions, and communicate with their customers. The large volumes and variety of data available today provide companies with unprecedented insight into market trends, customer behaviors, and risk management [8]. By using advanced analytics, financial firms can improve their predictive capabilities, optimize their investment strategies, and improve their operational efficiency.

Financial data visualization plays an important role in this area. By plotting and describing the data directly can let user obtain the information as quickly as possible. Hence, they can make decisions quickly. Powerful visualization tools allow users to see complex patterns and correlations that may be hidden in the raw data. This might inspire users.

Furthermore, the use of artificial intelligence algorithms in the financial data analysis sector further enhances the potential of big data. Machine learning models can analyze vast data sets at incredible speeds, revealing hidden insights and automating processes that were previously time-consuming and prone to human error. The AI will be more precise as the algorithms and computer quality improve, probably leading to innovation and bringing better user experience.

Likely, the future of the financial industry is inextricably linked to the effective use of vast amounts of data

analysis, powerful visualization techniques, and intelligent algorithms [9]. Utilizing these advanced technologies not only improves operational capabilities but also paves the way for innovation in the financial industry to meet the demands of a rapidly changing world.

Disclosure statement

The author declares no conflict of interest.

References

[1] Shui Y, Song G, 2016, Big Data Concepts, Theories, and Applications. Springer International Publishing. https://doi.org/10.1007/978-3-319-27763-9

[2] Wolfert S, Ge L, Verdouw C, et al., 2017, Big Data in Smart Farming—A Review. Agricultural Systems, 153: 69–80. https://doi.org/10.1016/j.agsy.2017.01.023

[3] Dash S, Shakyawar SK, Sharma M, et al., 2019, Big Data in Healthcare: Management, Analysis and Future Prospects. Journal of Big Data, 6: 54.

[4] Goldstein I, Spatt CS, Mao Y, 2021, Big Data in France. The Review of Financial Studies, 34(7): 3213–3225.

[5] Goldfarb D, 2022, Mplfinance Styles. https://github.com/matplotlib/mplfinance/blob/master/examples/styles.ipynb

[6] Goldfarb D, 2022, Mplfinance Plot Customizations. https://github.com/matplotlib/mplfinance/blob/master/examples/plot_customizations.ipynb

[7] Goldfarb D, 2023, Financial Markets Data Visualization using Matpolib. https://github.com/matplotlib/mplfinance?tab=readme-ov-file#usage

[8] Hasan MM, Popp J, Olah J, 2020, Current Landscape and Influence of Big Data on Finance. Journal of Big Data, 7: 21.

[9] Mahesh B, 2020, Machine Learning Algorithms—A Review. International Journal of Science and Research (IJSR), 9(1): 381–386.