Finance Data Project For Data Analysis

Created By

Rudraneel Chakraborty

Project Description

This data project is focused on exploratory data analysis of stock prices. Keep in mind, this project is just meant to practice visualization and pandas' skills, it is not meant to be a robust financial analysis or be taken as financial advice.

It's focused on bank stocks and see how they progressed throughout the financial crisis all the way to early 2023.

This project used the YFinance library to get stock information for the following banks:

- Bank of America
- Citi Group
- Goldman Sachs
- JPMorgan Chase
- Morgan Stanley
- Wells Fargo

The Financial Stock Market values for a stock depend on public sentiment as well as the Condition of the Financial Market for that year.

From December 2007 to June 2009, The Great Recession happened in the US, which resulted in stock value crashes. Here to analyse the financial stock data we took 6 Bank Stocks and will check how that was react though out the Years.

DATA DESCRIPTION

The dataset downloaded through YFinance Library to analyse on Stock Market data from early 2006 to early 2023.

Below following modules are used to do DATA VISULIZATION, ANALYSIS:

- **NUMPY** or numeric python is basically built to work on numerical data
- **PANDAS** is basically built on NUMPY to do certain jobs on the basis of numeric data.
- **MATPLOTLIB.PYPLOT** to implement the graphs such as Bar plot, Histogram, Box plot, Count plot, Bell curve to analyse our data.
- **SEABORN** is used to do critical statistical analysis in very simple & effective way, which gives better results comparing to MATPLOTLIB.
- **DATETIME** module supplies classes for manipulating dates and times.
- **PLOTLY** graphing library makes interactive, publication-quality graphs.
- **Cufflinks** binds the power of plotly with the flexibility of pandas for easy plotting.
- **YFinance** is a Python library that provides convenient access to the Yahoo Finance API.

CONTENT OF DATASET

The dataset contains several parameters which are considered important during analysing. The parameters are shown in the data which is given below:

```
Bank_Stocks.head()
```

Bank Ticker	BAC						C				...	MS		
Stock Info	Open	High	Low	Close	Adj Close	Volume	Open	High	Low	Close	...	Low	Close	Adj Close
Date														
2006-01-03	46.919998	47.180000	46.150002	47.080002	32.695641	16296700	490.000000	493.799988	481.100006	492.899994	...	56.740002	58.310001	33.995918
2006-01-04	47.000000	47.240002	46.450001	46.580002	32.348431	17757900	488.600006	491.000000	483.500000	483.799988	...	58.349998	58.349998	34.019238
2006-01-05	46.580002	46.830002	46.320000	46.639999	32.390095	14970700	484.399994	487.799988	484.000000	486.200012	...	58.020000	58.509998	34.112511
2006-01-06	46.799999	46.910000	46.349998	46.570000	32.341480	12599800	488.799988	489.000000	482.000000	486.200012	...	58.049999	58.570000	34.147499
2006-01-09	46.720001	46.970001	46.360001	46.599998	32.362320	15619400	486.000000	487.399994	483.000000	483.899994	...	58.619999	59.189999	34.508980

5 rows × 36 columns

The Data picked from 2006-01-01 to 2023-01-01, where the ticket_list contains the ticker for 6 banks:

start = dt.date(2006,1,1)

end = dt.date(2023,1,1)

ticker_list = ['BAC','C','GS','JPM','MS','WFC']

Once the data has been downloaded for those tickers on mentioned start date and end date then Pandas Concat Function called to make those in a single DataFrame by passing the axis value as 1, and the key would be ticker_list.

```
# Bank of America
BAC = yf.download("BAC", start=start,end=end)

# CitiGroup
C = yf.download("C", start=start,end=end)

# Goldman Sachs
GS = yf.download("GS", start=start,end=end)

# JPMorgan Chase
JPM = yf.download("JPM", start=start,end=end)

# Morgan Stanley
MS = yf.download("MS", start=start,end=end)

# Wells Fargo
WFC = yf.download("WFC", start=start,end=end)
```

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

In [96]:
```
Bank_Stocks = pd.concat([BAC, C, GS, JPM, MS, WFC],axis=1,keys=ticker_list)
```

In [97]:
```
Bank_Stocks.columns.names = ['Bank Ticker','Stock Info']
```

In [98]:
```
Bank_Stocks.head()
```

Out[98]:

Bank Ticker	BAC						C				...	MS		
Stock Info	Open	High	Low	Close	Adj Close	Volume	Open	High	Low	Close	...	Low	Close	Adj Close
Date														
2006-01-03	46.919998	47.180000	46.150002	47.080002	32.695641	16296700	490.000000	493.799988	481.100006	492.899994	...	56.740002	58.310001	33.995918
2006-01-04	47.000000	47.240002	46.450001	46.580002	32.348431	17757900	488.600006	491.000000	483.500000	483.799988	...	58.349998	58.349998	34.019238
2006-01-05	46.580002	46.830002	46.320000	46.639999	32.390095	14970700	484.399994	487.799988	484.000000	486.200012	...	58.020000	58.509998	34.112511
2006-01-06	46.799999	46.910000	46.349998	46.570000	32.341480	12599800	488.799988	489.000000	482.000000	486.200012	...	58.049999	58.570000	34.147499
2006-01-09	46.720001	46.970001	46.360001	46.599998	32.362320	15619400	486.000000	487.399994	483.000000	483.899994	...	58.619999	59.189999	34.508980

5 rows × 36 columns

Exploratory data analysis (EDA)

In EDA, we will check the values for the stocks on little precisely. So, let's first have a look the max Close price for each bank's stock throughout the time period.

```
In [99]: #EDA

         #What is the max Close price for each bank's stock throughout the time period?
         # for tick in ticker_list:
         #     print(tick,Bank_Stocks[tick]['Close'].max())
         Bank_Stocks.xs(key='Close',axis=1,level='Stock Info').max()

Out[99]: Bank Ticker
         BAC     54.900002
         C      564.099976
         GS     423.850006
         JPM    171.779999
         MS     108.730003
         WFC     65.930000
         dtype: float64
```

As, we can see from 2006 to 2023 the mentioned Values are the Maximum price of Closer on those Stock.

The Values showing about is in Doller and the whole analysis happen on base of dollar as we are looking for US Stocks Data.

Now, we will check returns for each bank's stock. The Formula to calculate the return is below:

$$r_t = \frac{p_t - p_{t-1}}{p_{t-1}} = \frac{p_t}{p_{t-1}} - 1$$

Below is the same functionality implemented though Pandas.

```
#This dataframe will contain the returns for each bank's stock
returns = pd.DataFrame()
```

```
for tick in ticker_list:
    returns[tick + ' Return'] = Bank_Stocks[tick]['Close'].pct_change()
```

```
returns.head()
```

Date	BAC Return	C Return	GS Return	JPM Return	MS Return	WFC Return
2006-01-03	NaN	NaN	NaN	NaN	NaN	NaN
2006-01-04	-0.010620	-0.018462	-0.013812	-0.014183	0.000686	-0.011599
2006-01-05	0.001288	0.004961	-0.000393	0.003029	0.002742	-0.001110
2006-01-06	-0.001501	0.000000	0.014169	0.007046	0.001025	0.005874
2006-01-09	0.000644	-0.004731	0.012030	0.016242	0.010586	-0.000158

Now Let's figure out from the PCT_Change(), In whole timeframe which was the days that gets the highest and lowest single day return.

```
returns.idxmin()
```

```
BAC Return    2009-01-20
C Return      2009-02-27
GS Return     2009-01-20
JPM Return    2009-01-20
MS Return     2008-10-09
WFC Return    2009-01-20
dtype: datetime64[ns]
```

```
returns.idxmax()
```

```
BAC Return    2009-04-09
C Return      2008-11-24
GS Return     2008-11-24
JPM Return    2009-01-21
MS Return     2008-10-13
WFC Return    2008-07-16
dtype: datetime64[ns]
```

From min and max, we can see it's targeting the Recession Time period for highest and lowest single day return. And Even id we check that JP Morgan Chase having highest and lowest in just one day.

What happen in one day that drops the price and again rise the price?

So, out of Curiosity if we look into the below data, we would see that the price fluctuates so much in one day that it didn't fluctuate on the whole-time frame.

```
Bank_Stocks.loc['2009-01-20':'2009-01-21']['JPM']
```

Stock Info Date	Open	High	Low	Close	Adj Close	Volume
2009-01-20	21.000000	21.27	17.700001	18.090000	12.624361	142090400
2009-01-21	19.290001	22.84	18.879999	22.629999	15.792667	136421400

So, there definitely a reason behind this, And if we look on the day 20-January-2009, we would see below result:

Which shows us that the Market Value is totally dependent on Public's Sentiment.

Now, Let's move and check which one from this was having the riskiest stock to purchase on the timeframe from 2008 to 2009, As the data up above is targeting The Great Recession Time, we will check on that time itself which was riskier among those 6 banks.

To, know about risk among those, we will check the Standard Deviation on that timeframe for 6 Bank Stocks. If a Stock having very high standard deviation on returns that means that stock is riskier.

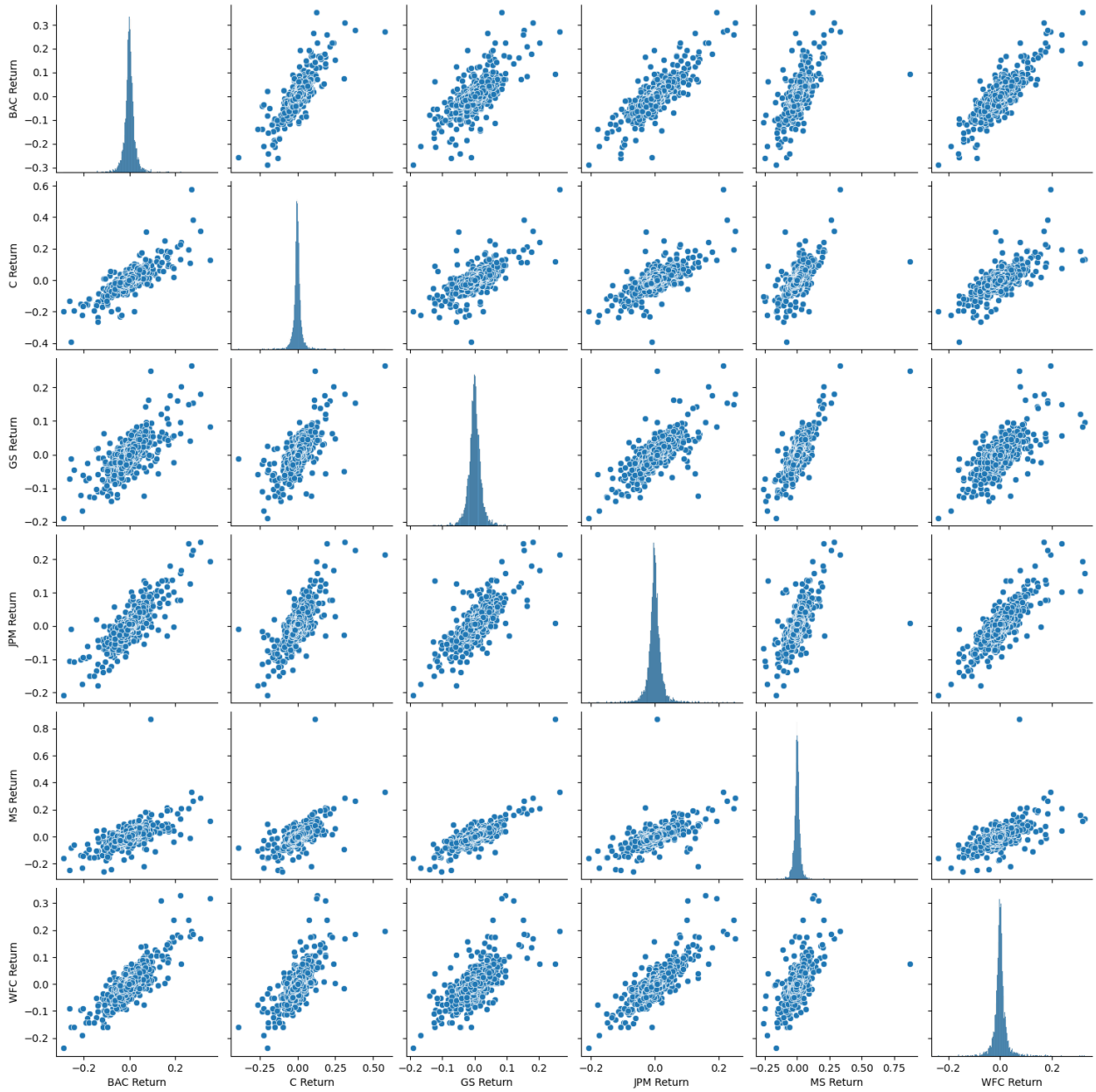
```
In [129]: returns.loc['2008-01-01':'2009-12-31'].std()

Out[129]: BAC Return    0.070748
          C Return      0.076439
          GS Return     0.044563
          JPM Return    0.052131
          MS Return     0.071731
          WFC Return    0.059993
          dtype: float64
```

As we can see among those stocks BAC, C & MS having higher STD, which means those was the risker on buy that time because the price was fluctuating from high to low.

And we will take another step to make more analysis towards the data though Plots on next step.

Data Analysis Through Plots

Let’s take a look on PairPlot of those 6 banks from 2006 to 2023 data.

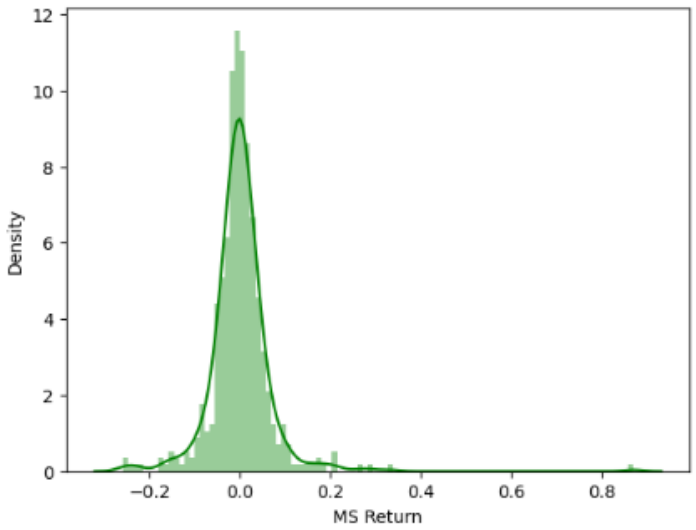


And to make clear picture on 2008-2009 let’s take a look on MS Return and now we will see in distplot.

```
sns.distplot(returns.loc['2008-01-01':'2009-12-31']['MS Return'],color='green',bins=100)
```

```
E:\SOFTWARE\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt
figure-level function with similar flexibility) or `histplot` (an axes-level function for
```

```
<AxesSubplot:xlabel='MS Return', ylabel='Density'>
```

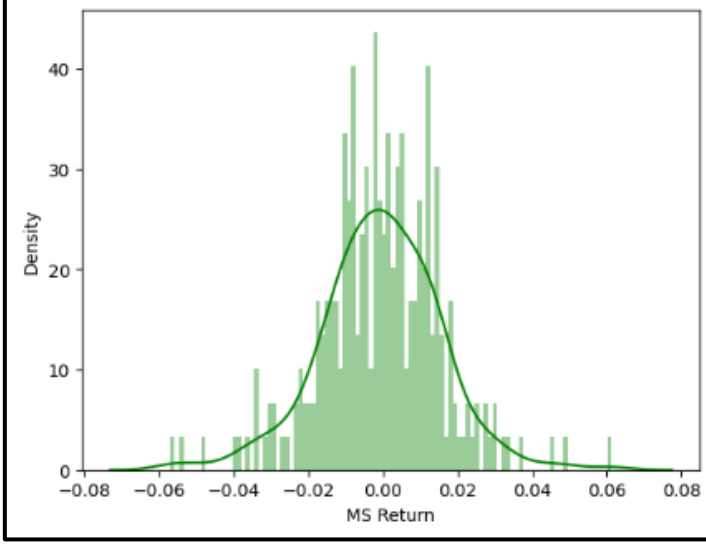


After seeing the KDE printed a smooth line on Distribution, we can see that the plot Is a right skewed. And if we differentiate the same plot with 2015 data, we would see that the plot will create a Bell Curve.

```
sns.distplot(returns.loc['2015-01-01':'2015-12-31']['MS Return'],color='green',bins=100)
```

```
E:\SOFTWARE\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt
figure-level function with similar flexibility) or `histplot` (an axes-level function for
```

```
<AxesSubplot:xlabel='MS Return', ylabel='Density'>
```



For more visibility and Understanding, Let's create a Line Plot by using Plotly to clearly understand how 2008's recession hit the stock market.

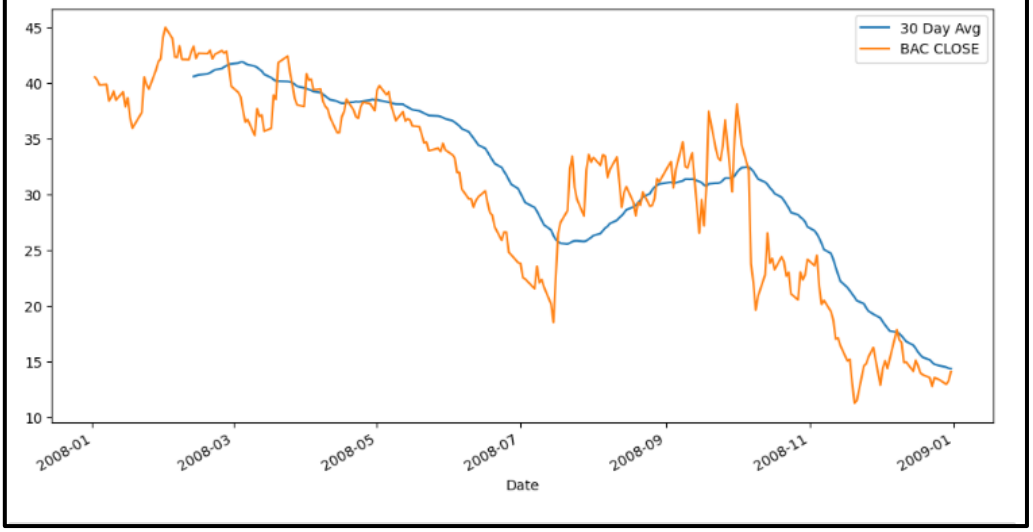
```
#Using Iplot Plotly
Bank_Stocks.xs(key='Close',axis=1,level='Stock Info').iplot()
```

So, from this we can see that on 2008-2009 All Bank Stocks ae just crashed, But the GS Bank recovered very fast but others were not able to recover so much faster.

And in below some other plots which provide us some more insights about the Timeperiod, So below is 30 days moving average data with BAC Close on 2008 -2009.

```
plt.figure(figsize=(12,6))
BAC['Close'].loc['2008-01-01':'2009-01-01'].rolling(window=30).mean().plot(label='30 Day Avg')
BAC['Close'].loc['2008-01-01':'2009-01-01'].plot(label='BAC CLOSE')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x20a9d91daf0>
```



And below is the Bollinger Curve for BAC Bank for 2008-2009 Timeframe.

So, At the End After Analysis we can tell that, The stock market is totally dependent on Public Sentiment as well as that depend on Market Situation. In the Analysis I targeted on Recession timeframe as that was the time when financial crisis happened and we saw the significant example to see how much that made a difference on Stock Market.

I made this analysis on 6 stocks but if we see any other stocks on that time frame, we would see same amount of significant difference though out the Period.