DR. D. Y. PATIL SCHOOL OF SCIENCE AND TECHNOLOGY

TATHAWADE, PUNE

A Foundation of Data Science-Report on S&P-500 Index Price Prediction.

SUBMITTED BY:

NAME OF STUDENT	ROLL NUMBER
1.TEJAS BHAVSAR	BTAI-204
2. ROHAN MASHERE	BTAI-228
3.PRATHEMESH NANGARE	BTAI-230

GUIDED BY:

Mrs. Mily Lal.

Data Preprocessing.

I. perform Data Cleaning:

Load the datasets

```
os.makedirs('data', exist_ok=True)


start = datetime.datetime(2000, 1, 1)
end = datetime.datetime(2020, 5, 31)

data_500 = yf.download("^GSPC", start=start, end=end)

# Convert index to string date format
data_500['Date'] = data_500.index.strftime('%Y-%m-%d')

# Reset index and keep the original columns
data_500.reset_index(drop=True, inplace=True)

# Get the original column names from the MultiIndex
original_columns = [col[0] for col in data_500.columns if col[0] != 'Date']
# add 'Date' to the beginning of the list
cols = ['Date'] + original_columns

# Reorder columns, including the MultiIndex columns, if needed
data_500 = data_500[cols]


# Save to CSV
data_500.to_csv(r'data/sp500_data.csv', index=False)
print("Saved successfully!")
```

```
[**********100%**********]  1 of 1 completedSaved successfully!
```

```
data_500 = pd.read_csv('sp500_data.csv')
```

```
data_500.head()
```

	Date	High	Low	Open	Close	Volume	Adj Close
0	2000-01-03	1478.000000	1438.359985	1469.250000	1455.219971	931800000	1455.219971
1	2000-01-04	1455.219971	1397.430054	1455.219971	1399.420044	1009000000	1399.420044
2	2000-01-05	1413.270020	1377.680054	1399.420044	1402.109985	1085500000	1402.109985
3	2000-01-06	1411.900024	1392.099976	1402.109985	1403.449951	1092300000	1403.449951
4	2000-01-07	1441.469971	1400.729980	1403.449951	1441.469971	1225200000	1441.469971

```
data_500.tail()
```

	Date	High	Low	Open	Close	Volume	Adj Close
5125	2020-05-18	2968.090088	2913.860107	2913.860107	2953.909912	6364290000	2953.909912
5126	2020-05-19	2964.209961	2922.350098	2948.590088	2922.939941	4969330000	2922.939941
5127	2020-05-20	2980.290039	2953.629883	2953.629883	2971.610107	4992970000	2971.610107
5128	2020-05-21	2978.500000	2938.570068	2969.949951	2948.510010	4966940000	2948.510010
5129	2020-05-22	2956.760010	2933.590088	2948.050049	2955.449951	3952800000	2955.449951

```
data_500.shape
```

```
(5130, 7)
```

Summary of the data

```
1 data_500.describe(include=['O'])
```

	Date
count	5130
unique	5130
top	2020-05-22
freq	1

Datatype Check

```
1 data_500.dtypes
```

	0
Date	object
High	float64
Low	float64
Open	float64
Close	float64
Volume	int64
Adj Close	float64

dtype: object

NULL check

```
1 data_500.isnull().sum()
```

	0
Date	0
High	0
Low	0
Open	0
Close	0
Volume	0
Adj Close	0

dtype: int64

- Checking the Summary of the data and datatype check it is Necessary before applying machine learning models, as they require correct data types.
- The output of data_500.isnull().sum() shows that there are no missing (NULL) values in any of the columns (Date, High, Low, Open, Close, Volume, Adj Close). This means your dataset is complete with no missing entries in these columns.

Daily Returns of S&P 500 (Simple Returns)

```
# Instead of 'close', use 'Adj Close' which is still in the DataFrame
data_500['Adj Close'] = pd.to_numeric(data_500['Adj Close'], errors='coerce')

# Now calculate daily returns using 'Adj Close'
daily_close = data_500['Adj Close']
data_500['Return'] = 100 * (data_500['Adj Close'].pct_change())

# Rounding the change to 2 digits after decimal
daily_pct_chg = round(data_500['Return'], 2)

# Print `daily_pct_chg`
daily_pct_chg.head()
```

	Return
0	NaN
1	-3.83
2	0.19
3	0.10
4	2.71

dtype: float64

Distribution of S&P 500 Daily returns

```
fig, ax = plt.subplots(figsize=(10, 5))
sns.distplot(data_500['Return'],bins=1000,color='blue')
plt.title("S&P 500 Daily returns")
plt.ioff()
daily_pct_chg.describe()
```

```
<ipython-input-250-562eb9efb5f8>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data_500['Return'],bins=1000,color='blue')
```

	Return
count	5129.000000
mean	0.021685
std	1.255191
min	-11.980000
25%	-0.480000
50%	0.050000
75%	0.570000
max	11.580000

- Mean: Average daily return.
- Standard deviation (std): Volatility of returns.
- Min/Max: Worst and best daily return.

Exploratory Data Analysis (EDA)

1. Computed Summary Statistics

- Calculated mean, median, variance, standard deviation, etc.

2. Visualized Data

- Created histograms to analyse distributions.

- Correlation - S&P500 and Big 5.

- Annualized Returns - S&P500 and Big 5. See the total returns of Big 5 stocks and S&P 500 in Bar chart.

- Let's plot the daily simple returns to visualize it better.

- Distribution of S&P 500 Daily returns.

- Returns - S&P 500 and Big 5.

- Volatility.

- 14-day future closing price, 14-day and 200-day moving average and 14-day and 200-day EMA are highly correlated with Adjusted Closing price of the stock.

1. Describe Data.

Describe data

```
1 data_500.describe()
```

	High	Low	Open	Close	Volume	Adj Close
count	5130.000000	5130.000000	5130.000000	5130.000000	5.130000e+03	5130.000000
mean	1610.995689	1591.619046	1601.711898	1601.876674	3.134990e+09	1601.876674
std	613.904379	610.780283	612.468993	612.525419	1.508044e+09	612.525419
min	695.270020	666.789978	679.280029	676.530029	3.560700e+08	676.530029
25%	1164.680023	1146.172546	1156.892487	1156.852478	1.697700e+09	1156.852478
50%	1380.200012	1360.964966	1369.579956	1369.574951	3.243340e+09	1369.574951
75%	2038.142456	2015.500000	2026.562531	2026.080017	3.945430e+09	2026.080017
max	3393.520020	3378.830078	3380.449951	3386.149902	1.145623e+10	3386.149902

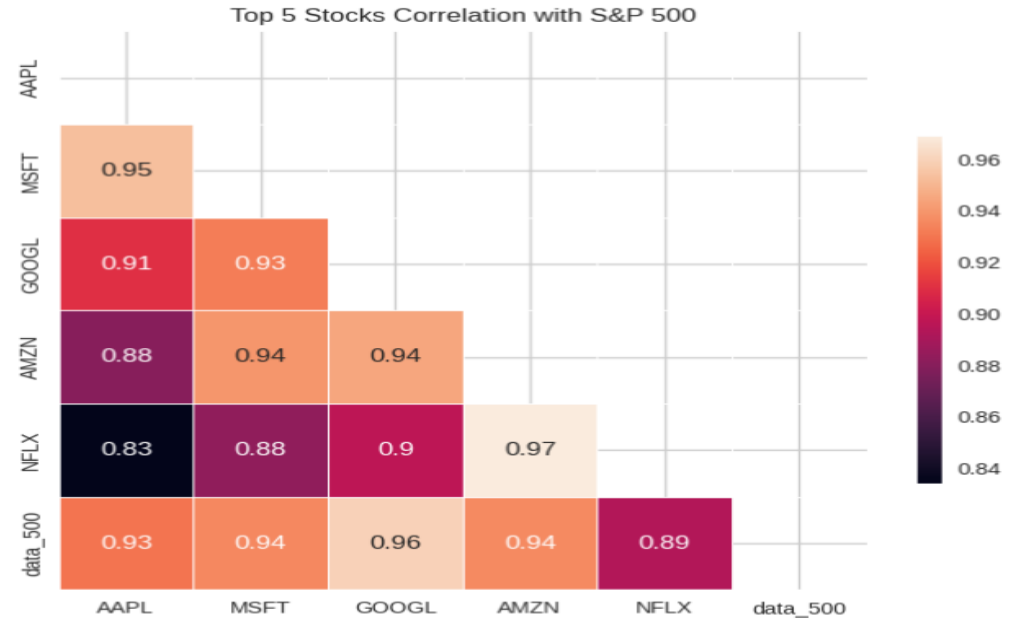
2. Data Visualization:

Correlation - S&P500 and Big 5

```
corr = top5_df.drop(columns=['Date']).corr()

mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
#correlation matrix
fig, ax = plt.subplots(figsize=(8, 8))
sns.heatmap(corr, mask=mask, square=True, linewidths=.5, annot=True,
            cbar_kws={"shrink": .5})
ax.set_title('Top 5 Stocks Correlation with S&P 500')
```

```
Text(0.5, 1.0, 'Top 5 Stocks Correlation with S&P 500')
```



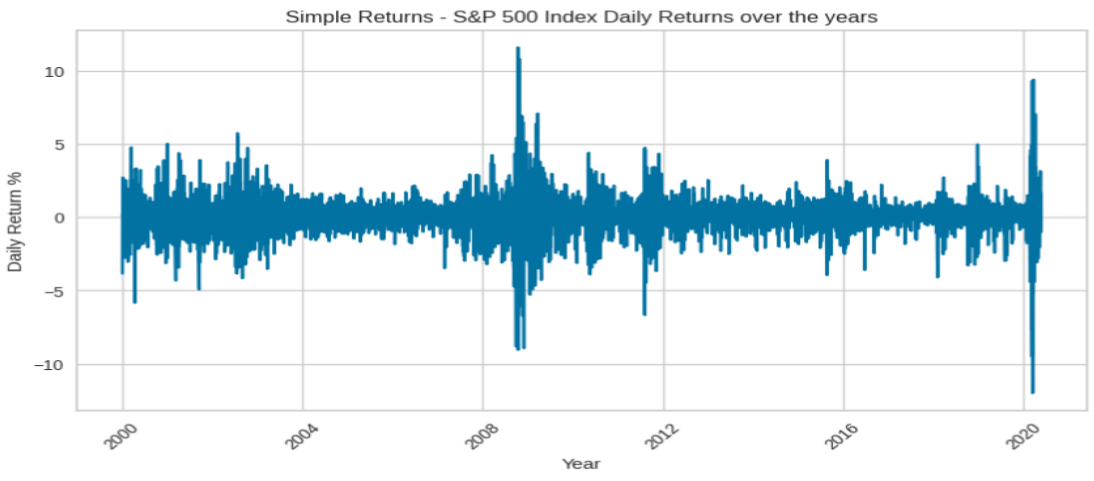
Lets plot the daily simple returns to visualize it better

```
fig, ax = plt.subplots(figsize=(10, 5))

# Convert 'Date' column to datetime objects if it's not already
data_500['Date'] = pd.to_datetime(data_500['Date'])

# Plot the 'Return' column against the 'Date' column
plt.plot(data_500['Date'], data_500['Return'], color='b')
plt.xlabel('Year')
plt.ylabel('Daily Return %')
plt.title('Simple Returns - S&P 500 Index Daily Returns over the years')

# Rotate x-axis labels for better readability (optional)
plt.xticks(rotation=45)
plt.show()
```



Returns - S&P 500 and Big 5

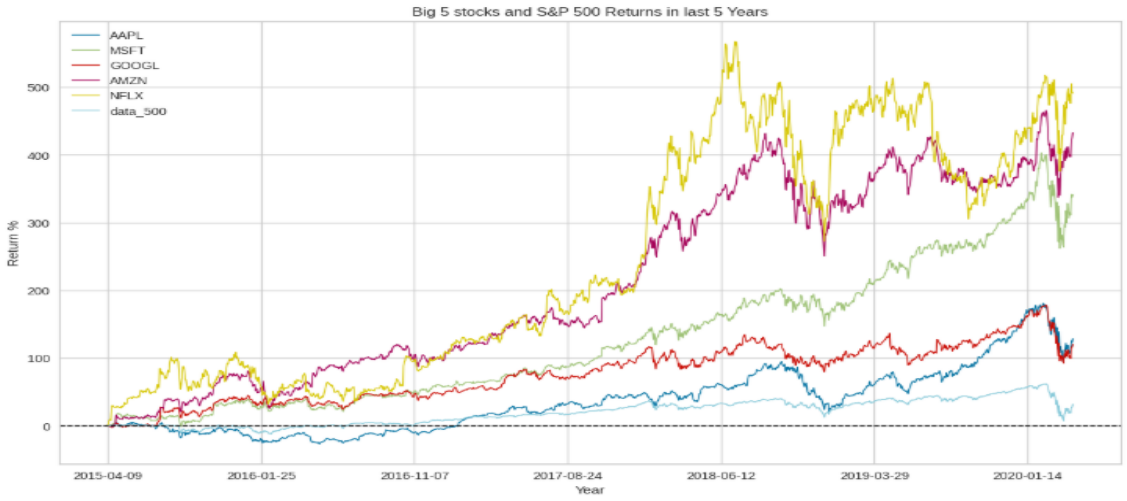
Rate of return(ROR) tells what % is gained or lost over a period of time.

** Rate of Return % = (Curent Price - Starting Price)/ Starting Price * 100 **

ploting the yearly returns of the big 5 stocks with S&P 500 in last 5 years.

```
[ ]  1 top5_df_dup = top5_df.copy()
     2 # Instead of dropping 'Date', set it as the index
     3 top5_df_dup = top5_df_dup.set_index('Date')
     4
     5 # Calculate returns as before
     6 top5_df_dup = (top5_df_dup - top5_df_dup.iloc[0, :])/top5_df_dup.iloc[0, :]*100
     7 top5_df_dup.plot(legend=True, figsize=(14, 8), linewidth=1)
     8 plt.axhline(y=0, linestyle='dashed', color='black', linewidth=1)
     9 plt.xlabel('Year')
    10 plt.ylabel('Return %')
    11 plt.title('Big 5 stocks and S&P 500 Returns in last 5 Years')
```

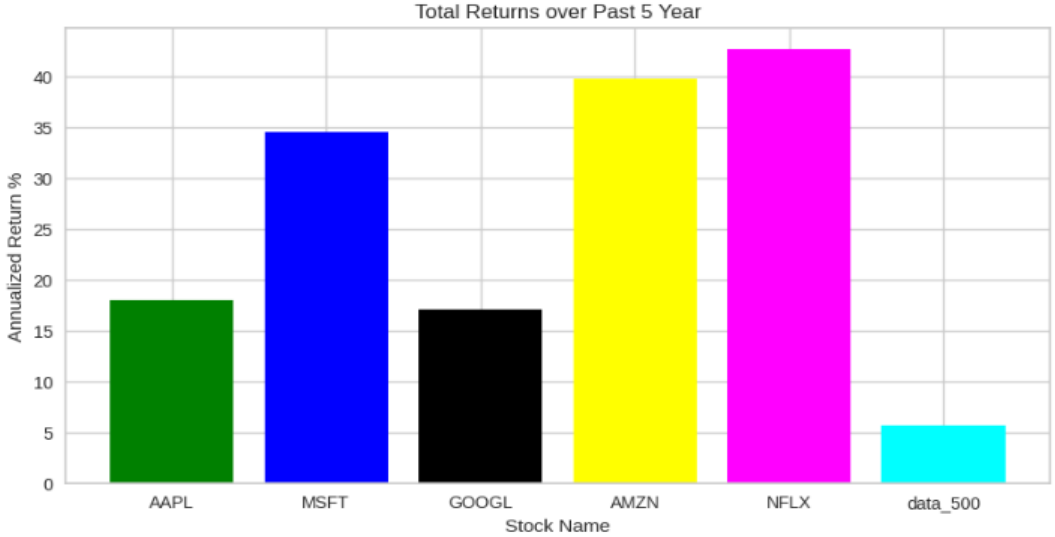
```
Text(0.5, 1.0, 'Big 5 stocks and S&P 500 Returns in last 5 Years')
```



Annualized Returns - S&P500 and Big 5

See the total returns of Big 5 stocks and S&P 500 in Bar chart.

```
 1 annual_retn = {}
 2 for t in top5_df_dup.columns:
 3     ## Annulized returns averaged for last 5 years
 4     annual_retn[t] = ((top5_df_dup[t][-1:].values/100 + 1)**(1/5) - 1)*100
 5 list2 = []
 6 for key, value in annual_retn.items():
 7     list2.append(value[0])
 8
 9 fig, ax = plt.subplots(figsize=(10, 5))
10 # Use a valid color or sequence of colors
11 plt.bar(list(annual_retn.keys()), list2, color=['green', 'blue', 'black', 'yellow', 'magenta', 'cyan', 'red'])
12 plt.xlabel('Stock Name')
13 plt.ylabel('Annualized Return %')
14 plt.title('Total Returns over Past 5 Year')
15 plt.show()
```

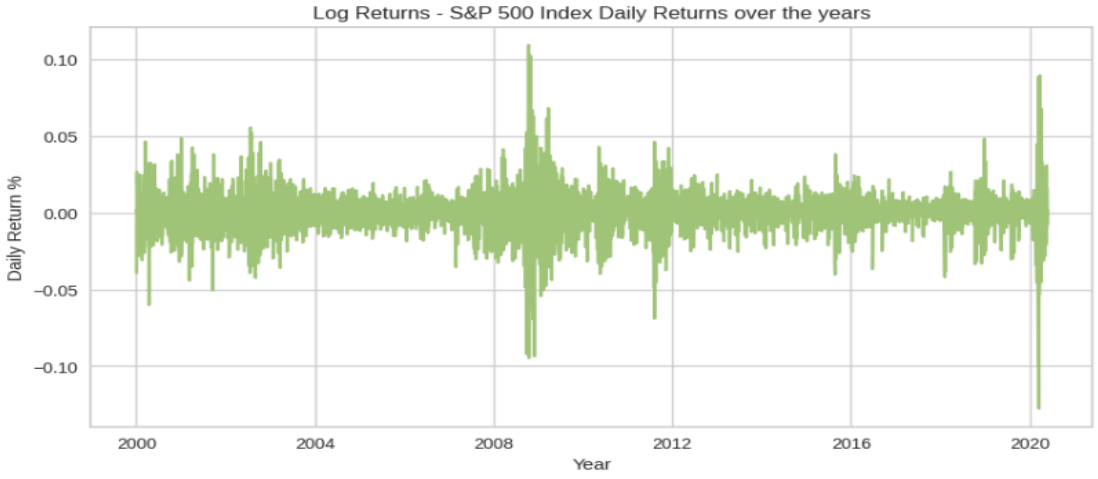


Daily Log Returns of S&P 500

```python
fig, ax = plt.subplots(figsize=(10, 5))

# Calculate daily log returns if not already calculated
if 'daily_log_returns' not in locals():  # Check if variable exists
    data_500['Log_Return'] = np.log(data_500['Close'] / data_500['Close'].shift(1))
    daily_log_returns = data_500['Log_Return']

# Now you can plot the daily log returns
plt.plot(data_500['Date'], daily_log_returns, color='g')
plt.xlabel('Year')
plt.ylabel('Daily Return %')
plt.title('Log Returns - S&P 500 Index Daily Returns over the years')
plt.show()
```

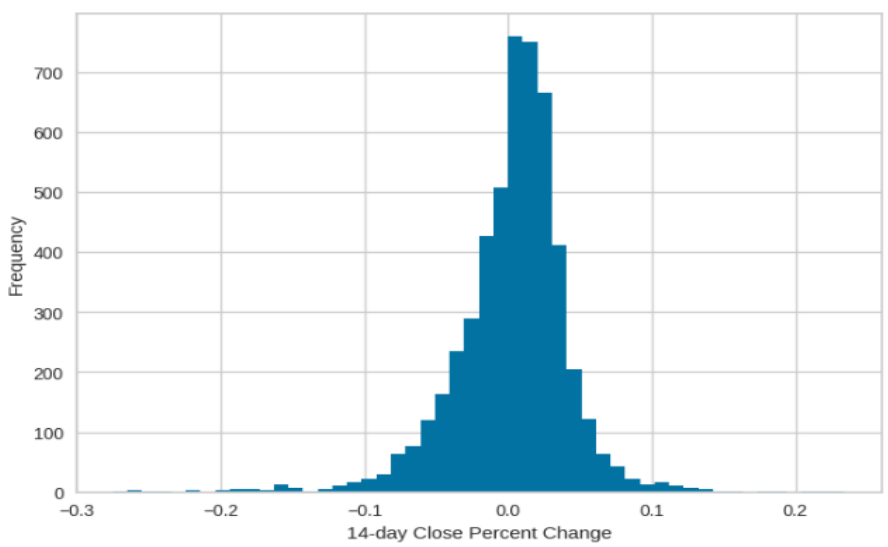


Creating Features for 14 day Future Close and 14 day Future close percent change.

```python
data_500['14d_close_pct'] = data_500['Adj Close'].pct_change(14)  # Change 'Close' to 'Adj Close'
data_500['14d_close_pct'].plot.hist(bins=50)
plt.xlabel('14-day Close Percent Change')
plt.show()

# In cell 130:

# Similarly, use 'Adj Close' or another suitable column for future calculations
data_500['14d_future_Close'] = data_500['Adj Close'].shift(-14)  # Change 'Close' to 'Adj Close'
data_500['14d_future_Close_pct'] = data_500['14d_future_Close'].pct_change(14)
```



Volatility

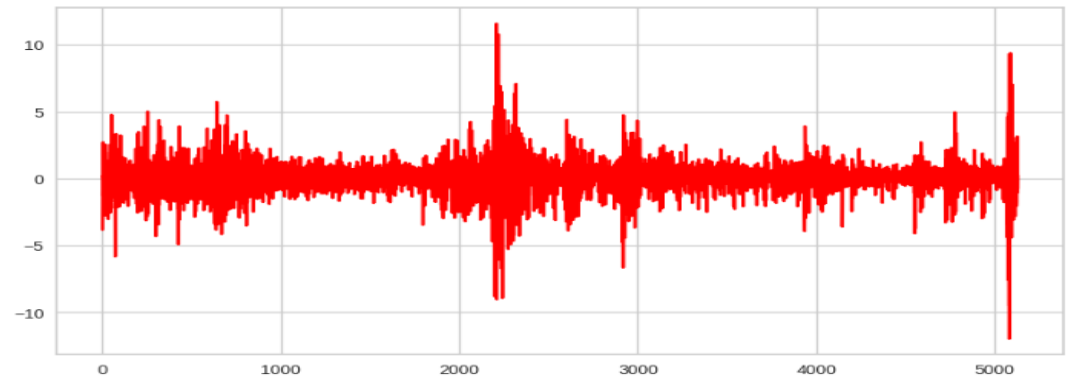
- Volatility is basically the dispersion of the financial asset returns over time
- It is important that the higher volatility then it is risky assets, let's calculate daily, monthly and annual volatility for S&P 500.

```
# Plot the price returns
fig, ax = plt.subplots(figsize=(10, 5))
plt.plot(data_500['Return'],color = 'red')
plt.show()

# Calculate daily std of returns
std_daily = data_500['Return'].std()
print('Daily volatility: ', '{:.2f}%'.format(std_daily))

# Convert daily volatility to monthly volatility
# At an average there are 21 trading days in a month
std_monthly = math.sqrt(21) * std_daily
print('Monthly volatility: ', '{:.2f}%'.format(std_monthly))

# Convert daily volatility to annaul volatility
# At an average there are 252 trading days in an year
std_annual = math.sqrt(252) * std_daily
print('Annual volatility: ', '{:.2f}%'.format(std_annual))
```

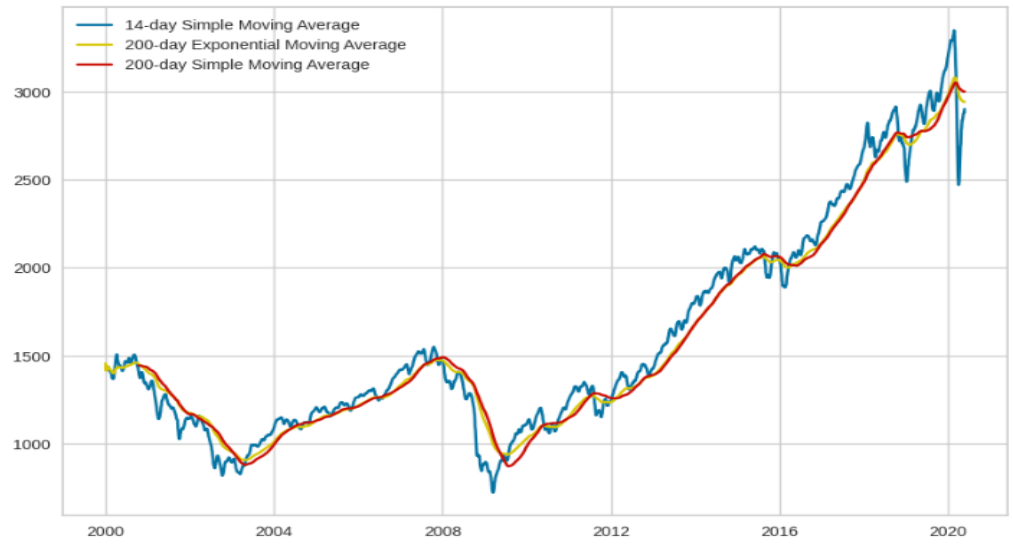



```
Daily volatility:  1.26%
Monthly volatility:  5.75%
Annual volatility:  19.93%
```

Interpretation:

- 14-day future closing price, 14-day and 200-day moving average and 14-day and 200-day EMA are highly correlated with Adjusted Closing price of the stock.

```
fig, ax = plt.subplots(figsize=(10, 7))
plt.plot(data_500['Date'], data_500['ma14'], label='14-day Simple Moving Average', color='b')
plt.plot(data_500['Date'], data_500['ema200'], label='200-day Exponential Moving Average', color='y')
plt.plot(data_500['Date'], data_500['ma200'], label='200-day Simple Moving Average', color='r')
plt.legend()
plt.show()
```



END!!