Report on Stock Analysis of Bharti_Airtel

Introduction

The stock analysis of Bharti Airtel form 1-1-2017 to 24-07-2019 with the help of Python codes.

Before we play with stock information, we have to get it in some serviceable arrangement. Stock information can be acquired from Yahoo! Account, Google Finance, or various different sources, and the pandas bundle gives simple access to Yahoo! Money and Google Finance information, alongside different sources. In this address, we will get our information from Yahoo! Money.

The accompanying code shows how to make straightforwardly a DataFrame article containing stock data

Here are the codes, output and their analysis.

1) **Importing the important libraries required for the analysis**
   - import pandas_datareader as web
   - import numpy as np
   - import pandas as pd
   - import datetime
2) **Creating the start and end for Analysis Purpose.**
   - start=datetime.datetime(2017,1,1)
   - end=datetime.datetime(2019,7,24)
3) **Now, the stock of bharti airtel should be loaded in yesbank list from yahoo. The codes are**
   - Bharti_airtel= web. DataReader("Bharti_airtel.ns",'yahoo',start,end)
   - Bharti_airtel

	High	Low	Open	Close	Volume	Adj Close
Date						
2017-01-02	286.471985	278.299988	278.299988	285.553009	1612972.0	278.237366
2017-	288.261993	275.454010	286.425995	279.539001	3225790.0	272.377441

01-03						
2017-01-04	289.410004	274.673004	277.657013	288.216003	5669757.0	280.832153
2017-01-05	296.571991	284.681000	287.528015	294.826996	4189622.0	287.273773

How about we quickly talk about, open is the cost of the stock toward the start of the exchanging day (it need not be the end cost of the past exchanging day), high is the most astounding cost of the stock on that exchanging day, low the least cost of the stock on that exchanging day, and close the cost of the stock at shutting time. The volume demonstrates what number of stocks were exchanged. Balanced close is the end cost of the stock that alters the cost of the stock for corporate activities. While stock costs are viewed asset generally by brokers, stock parts (when the organization makes each surviving stock worth two and parts the cost) and profits (payout of organization benefits per share) additionally influence the cost of stock and ought to be represented.

4) **Then after finding the different EDA things from the data extracted from the yahoo.**

These are

- Bharti_airtel.index
- Bharti_airtel.columns

5) **Then after plotting some graph for analyzing the stock of bharti airtel**.

a) yesbank[['Open','Close']].plot()

In the above graph open stock price and closing stock price are interpreted from 2017 to 2019. As in the graph its showed that before of 2018 the stock Price of Bharti Airtel was highest whereas from 2019 the stock price of Bharti Airtel was decreases in the 2017 comparison. This also affect the future of stock.

b. Bharti_airtel[['High','Low']].plot()

In the above graph relation between the highest and lowest stock price of Shares are showed between the assigned period. Here, the Highest Stock Price went up to more than Rs.500. in of 2017. The Lowest Stock Price were in 2018 that's below Rs. 260. So the Stock Market of Bharti Airtel is right now going very well.

6) for monthly level

monthly_df = bharti_airtel.resample("M").mean()

print(monthly_df)

```
                 High         Low        Open       Close       Volume  \
Date
2017-01-31  298.639711  288.563667  291.530426  294.112193  4.672155e+06
2017-02-28  337.075422  325.965366  329.860424  330.515267  6.660237e+06
2017-03-31  327.969094  319.672087  324.822277  323.782999  4.450587e+06
2017-04-30  321.492554  313.522164  317.332669  318.041614  3.590338e+06
2017-05-31  339.767731  329.841001  333.198499  334.529953  6.119871e+06
```

7) finding adjustment close of each day.

```
daily_close = bharti_airtel[["Adj Close"]]

daily_pct_change = daily_close.pct_change()

daily_pct_change.fillna(0, inplace=True)

print(daily_pct_change)
```

```
            Adj Close
Date
2017-01-02   0.000000
2017-01-03  -0.021061
2017-01-04   0.031040
2017-01-05   0.022938
2017-01-06  -0.004202
```

8) Finding diff. of closing and opening price.

```
bharti_airtel["diff"] = bharti_airtel.Open - bharti_airtel.Close
bharti_airtel["diff"]
```

```
Date
2017-01-02     -7.253021
2017-01-03      6.886993
2017-01-04    -10.558990
2017-01-05     -7.298981
2017-01-06     -0.551025
```

9)

```
from matplotlib.dates import DateFormatter, WeekdayLocator,\
    DayLocator, MONDAY
```

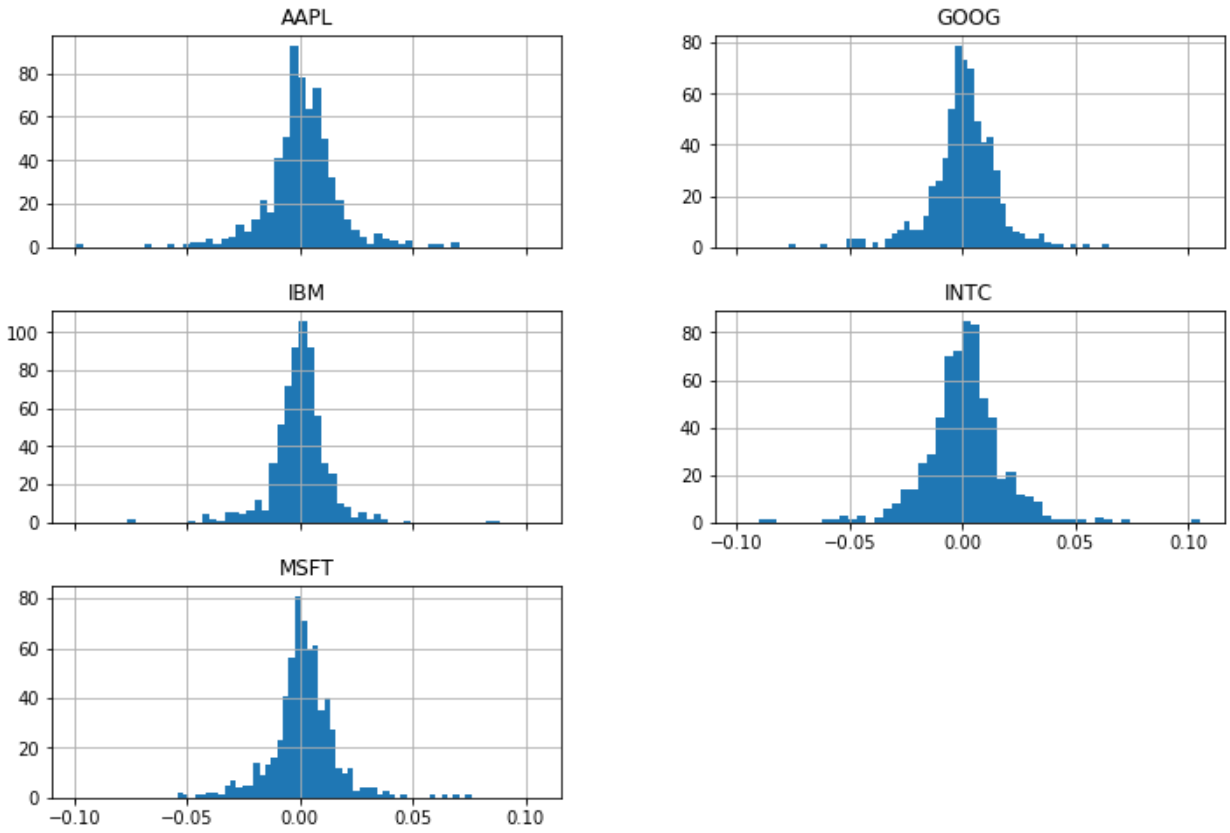
10) **Plotting the graph of 24M**

```
bharti_airtel["24m"] = np.round(bharti_airtel["Close"].rolling(window = 20, center = False).mean(), 2)
pandas_candlestick_ohlc(bharti_airtel.loc['2019-07-24':'2019-08-24',:], otherseries = "24m")
(bharti_airtel["24m"]).plot(grid=True)
plt.show()
```

11)
```
def get(tickers, startdate, enddate):
  def data(ticker):
    return (web.get_data_yahoo(ticker, start=startdate, end=enddate))
  datas = map (data, tickers)
  return(pd.concat(datas, keys=tickers, names=['Ticker', 'Date']))

tickers = ['AAPL', 'MSFT', 'IBM', 'GOOG','INTC']
all_data = get(tickers, datetime.datetime(2017, 1, 1), datetime.datetime(2019, 7, 24))
all_data.head()
```

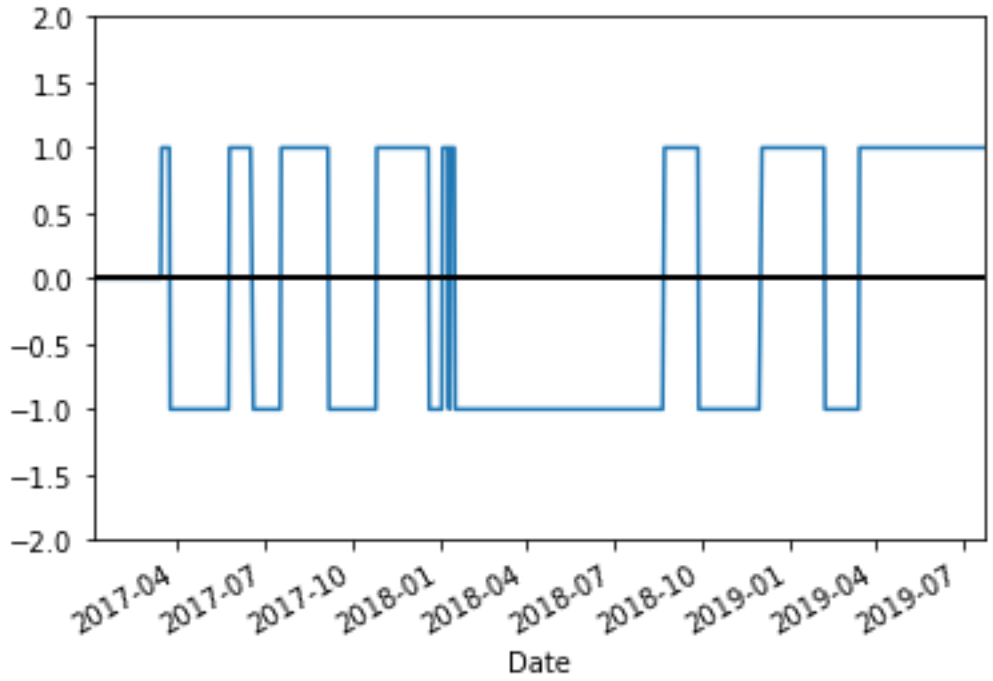
12)
```
# Isolate the `Adj Close` values and transform the DataFrame
daily_close_px = all_data[['Adj Close']].reset_index().pivot('Date', 'Ticker', 'AdjClose')

# Calculate the daily percentage change for `daily_close_px`
daily_pct_change = daily_close_px.pct_change()

# Plot the distributions
daily_pct_change.hist(bins=50, sharex=True, figsize=(12,8))

# Show the resulting plot
```

plt.show()



13) stock_change_apr = stock_change * 252 * 100 # There are 252 trading days in a year; the 100 converts to percentages
stock_change_apr.tail()
bharti_airtel["20d"] = np.round(bharti_airtel["Adj Close"].rolling(window = 20, center = False).mean(), 2)
pandas_candlestick_ohlc(bharti_airtel.loc['2019-01-04':'2019-12-31',:], otherseries = "20d")
bharti_airtel["20d"].plot(grid = True)
bharti_airtel['20d-50d'] = bharti_airtel['20d'] - bharti_airtel['50d']

```
bharti_airtel.tail()
# np.where() is a vectorized if-else function, where a condition is checked for each component of a vector, and the first argument passed is used when the condition holds, and the other passed if it does not
bharti_airtel["Regime"] = np.where(bharti_airtel['20d-50d'] > 0, 1, 0)
# We have 1's for bullish regimes and 0's for everything else. Below I replace bearish regimes's values with -1, and to maintain the rest of the vector, the second argument is apple["Regime"]
bharti_airtel["Regime"] = np.where(bharti_airtel['20d-50d'] < 0, -1, bharti_airtel["Regime"])
bharti_airtel.loc['2016-01-04':'2019-12-31',"Regime"].plot(ylim = (-2,2)).axhline(y = 0, color = "black", lw = 2)
```



14) bharti_airtel["Regime"].value_counts()

```
-1    320
 1    264
 0     49
Name: Regime, dtype: int64
```

15) # Create a DataFrame with trades, including the price at the trade and the regime under which the trade is made.

```
bharti_airtel_signals = pd.concat([
        pd.DataFrame({"Price": bharti_airtel.loc[bharti_airtel["Signal"] == 1, "Adj Close
"],
"Regime": bharti_airtel.loc[bharti_airtel["Signal"] == 1, "Regime"],
                    "Signal": "Buy"}),
          pd.DataFrame({"Price": bharti_airtel.loc[bharti_airtel["Signal"] == -1, "Adj Close"],
                "Regime": bharti_airtel.loc[bharti_airtel["Signal"] == -1, "Regime"],
                    "Signal": "Sell"}),
        ])
bharti_airtel_signals.sort_index(inplace = True)
bharti_airtel_signals
```

Date	Price	Regime	Signal
2017-03-15	324.938477	1	Buy
2017-03-24	304.630432	-1	Sell
2017-05-25	331.649017	1	Buy
2017-06-19	328.786285	-1	Sell
2017-07-18	366.292175	1	Buy
2017-09-06	361.089325	-1	Sell
2017-10-26	458.695740	1	Buy
2017-12-20	474.485046	-1	Sell

Date	Price	Regime	Signal
2018-01-03	464.885406	1	Buy
2018-01-10	454.299927	-1	Sell
2018-01-11	461.790100	1	Buy
2018-01-16	446.090668	-1	Sell
2018-08-23	336.572449	1	Buy
2018-09-28	308.046906	-1	Sell
2018-12-03	293.588013	1	Buy
2019-02-07	285.691010	-1	Sell
2019-03-15	310.069000	1	Buy
2019-07-25	340.000000	1	Sell