AMAZON Stock Analysis

Contents

1. Stock performance history
2. Exploratory data analysis
3. Data pre-processing
4. Candidate models
5. Strategies
6. Strategy performance evaluation
7. Conclusion

History of Stock price

If you bought $14,000 worth of amazon stock in 2001, you'd be a millionaire this morning!

Amazon stock price has risen more than 400% in last five years.

The revenue grew from $48 billion in 2011 fiscal year to $136 billion in 2016

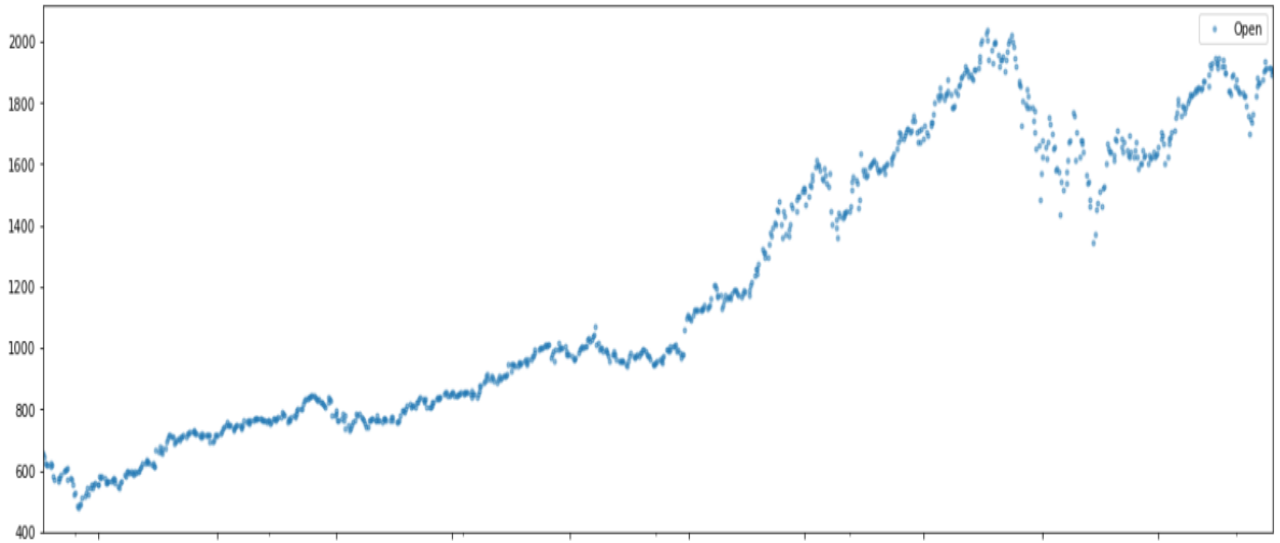
The all-time high Amazon stock closing price was **2039.51** on **September 04, 2018**.

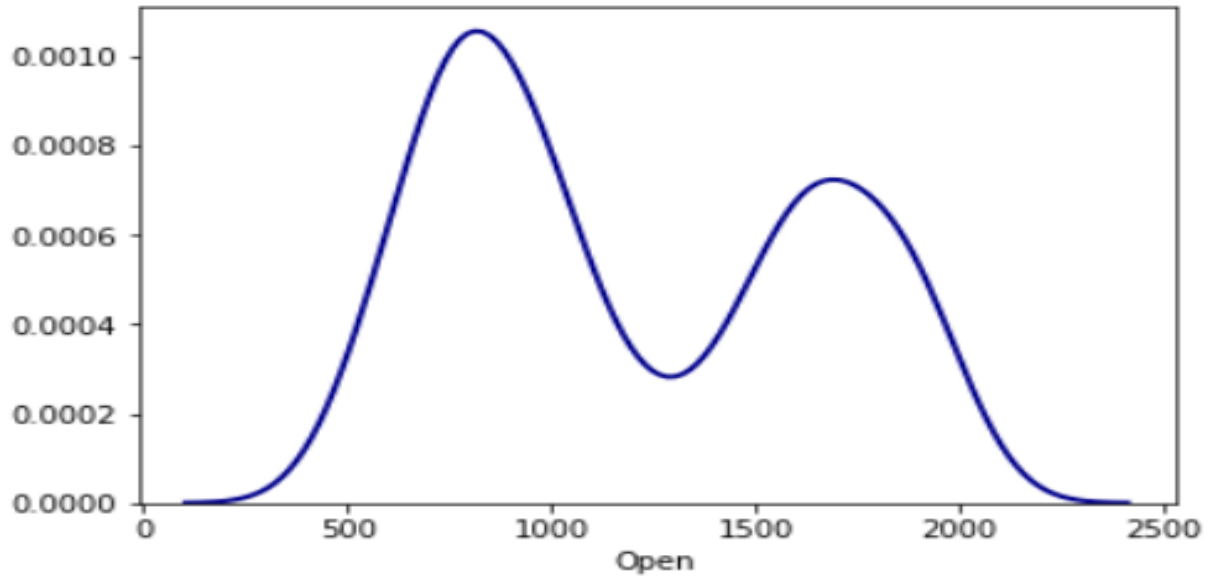
The Amazon 52-week high stock price is **2035.80**, which is **16.2%** above the current share price.

The Amazon 52-week low stock price is **1307.00**, which is **25.4%** below the current share price.

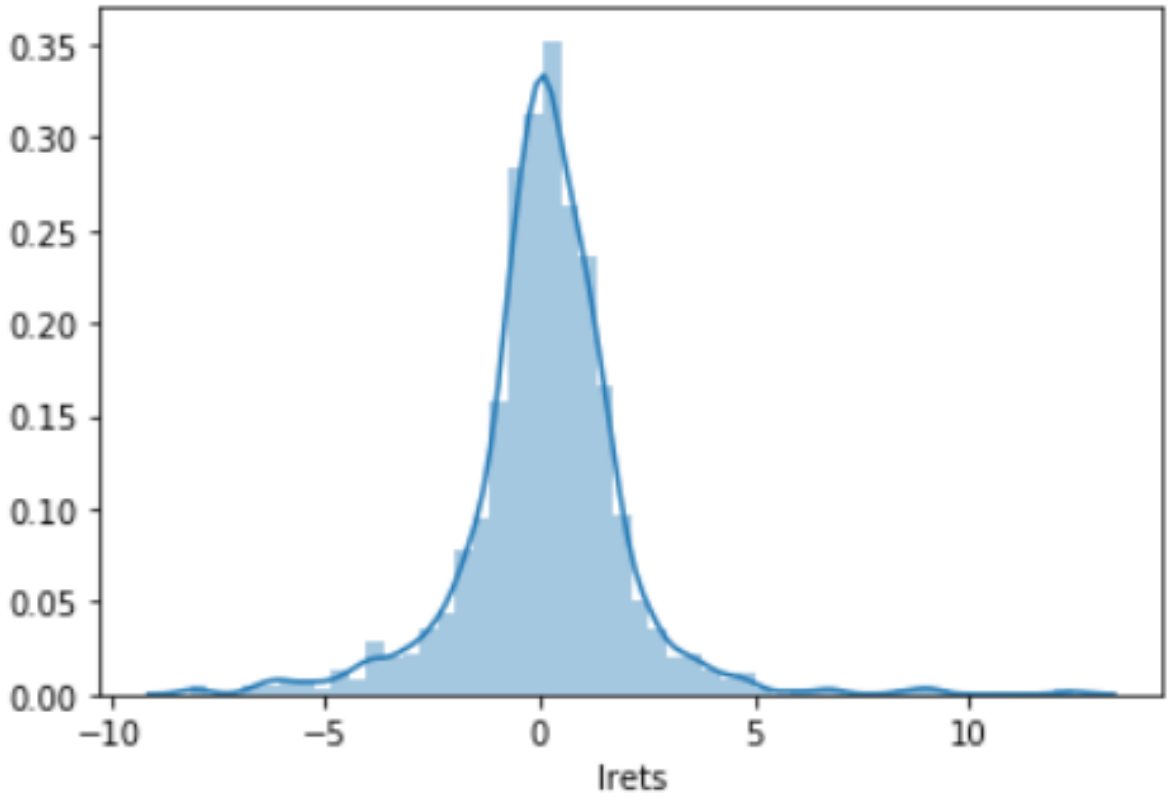
The average Amazon stock price for the last 52 weeks is **1773.07**.

Exploratory Data Analysis

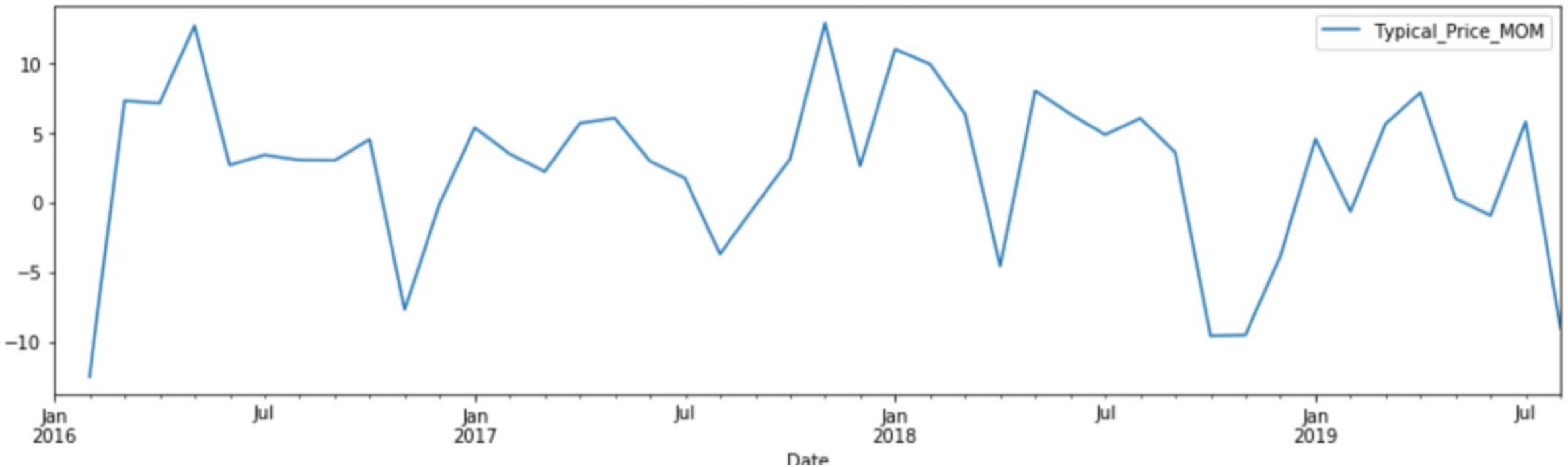




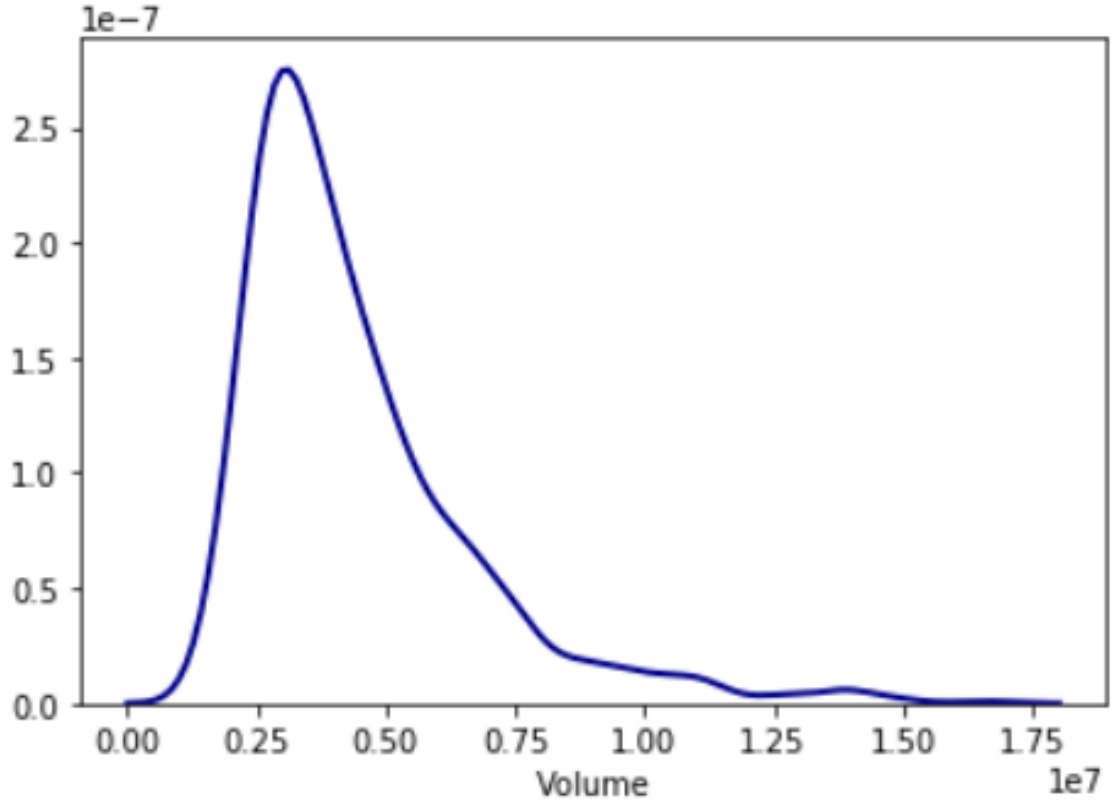
Distribution of returns



Monthly returns



Volume Density plot



Data pre-processing

Date	Open	Close	High	Low	Volume	Typical_Price	Mkt-RF	SMB	HML	RF	ADS_Index_102419
2016-01-31	603.434204	601.061578	613.208416	589.950526	6.852679e+06	601.406840	-0.301579	-0.186842	0.125263	0.000	-0.231622
2016-02-29	531.865005	530.620000	539.900002	521.103500	6.207240e+06	530.541167	0.003000	0.041000	-0.025500	0.001	-0.420509
2016-03-31	570.431366	572.374090	577.272272	563.422730	4.273159e+06	571.023031	0.308636	0.041364	0.050455	0.001	-0.623420
2016-04-30	613.530951	613.594288	619.647618	607.133333	3.736390e+06	613.458413	0.045714	0.034286	0.154762	0.000	-0.529730
2016-05-31	694.639526	697.473807	702.979042	690.159526	4.314976e+06	696.870791	0.087143	-0.011429	-0.086190	0.001	-0.447915

Candidate models

- Kalman Filter
- Moving Average
- MIDAS
- Keltner Chanel
- Random Forest
- GARCH
- Model selection using Random forest

Trading Strategies

- **Buy and hold**

  - If prediction > yesterday's close price -> buy;

  - If already bought -> hold

  - Else sell

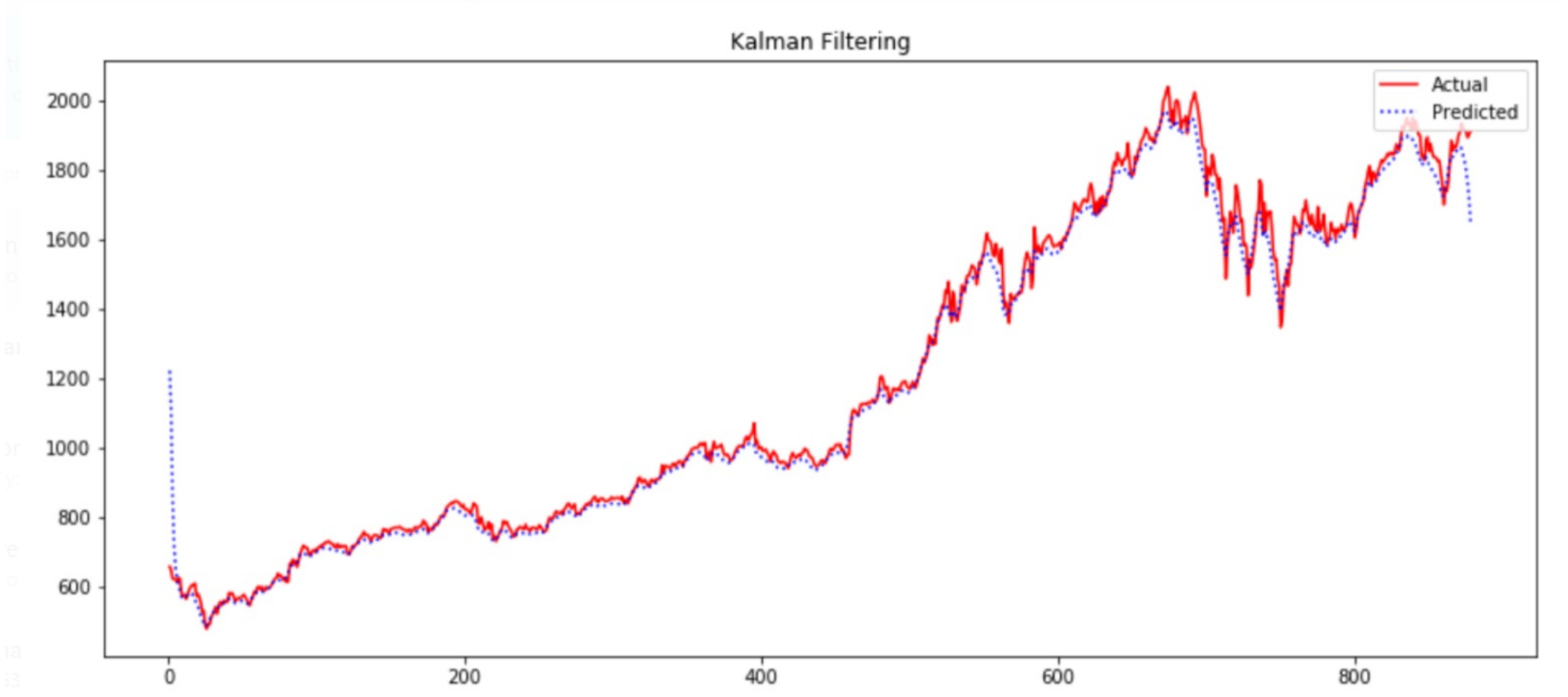
- **Long and Short**

  - If predicted > yesterday's close price -> go long else go short

- **Day trading**

  - If predicted > yesterday's close price -> buy it at open and sell at close

Kalman Filter



Strategy for Kalman Filter

Long Short Day trading:

#if predicted > yesterdays close, buy and sell at end of day

#if predicted < yesterdays close, sell and buy at end of day

```
amount = 10000
signal = 0
Amount = []
balance = 0
action = []
portfolio = 0
Portfolio = []
stocks = 0
Stocks = []

for i in range(len(results)):
    if results['Predicted'][i] > results['Actual'][i-1]:
        action.append('Buy at Open')
        stocks = int(amount/results['Open'][i])
        balance = int(amount%results['Close'][i])
        portfolio = stocks * results ['Open'][i]
        print(i,'Buy at Open',round(portfolio,2),stocks,round(balance,2))


        action.append('Sell at End')
        portfolio = stocks * results['Close'][i]
        signal = 0
        stocks = 0
        amount = balance + portfolio
        portfolio = 0
        balance = 0
        print(i,'Sell at Close',round(amount,2),balance)
```

```
    else:
        action.append('Sell at Open')
        stocks = int(amount/results['Open'][i])
        balance = int(amount%results['Close'][i])
        portfolio = stocks * results ['Open'][i]
        print(i,'Sell at Open',round(portfolio,2),'-',stocks,round(balance,2))


        action.append('Buy at Close')
        portfolio = stocks * results['Close'][i]
        signal = 0
        stocks = 0
        amount = balance + portfolio
        portfolio = 0
        balance = 0
        print(i,'Buy Back at Close',round(amount,2),balance)
Amount.append(amount)
print(Amount)
```

Back testing Result

```
0 Sell at Open 9844.35 - 15 445
0 Buy Back at Close 9999.85 0
1 Buy at Open 9702.9 15 493
1 Sell at Close 9999.85 0
2 Buy at Open 9952.0 16 510
2 Sell at Close 10632.4 0
3 Buy at Open 10570.6 17 297
3 Sell at Close 10631.98 0
4 Buy at Open 10534.22 17 312
4 Sell at Close 10631.85 0
5 Buy at Open 10412.16 17 130
5 Sell at Close 10631.58 0
6 Sell at Open 10629.25 - 17 127
6 Buy Back at Close 10631.13 0
```

Moving Average

- It is used indicator in technical analysis that helps smooth outprice action
- Two basic and commonly used moving averages are simple moving average (SMA) and exponential moving average(EMA)

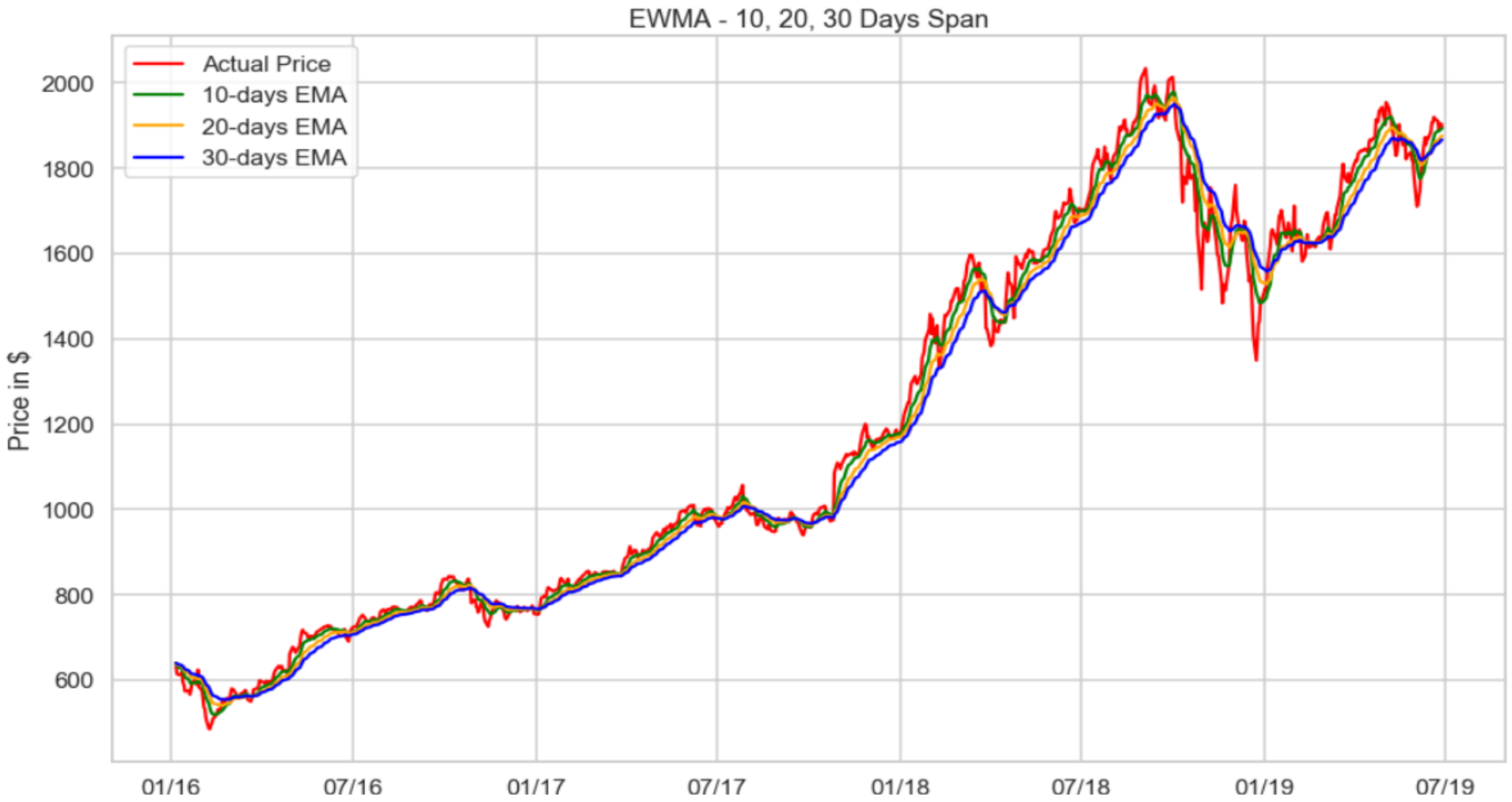
Combining data

Date	Open	Close	High	Low	Volume	Typical_Price	lrets
2019-06-24	1912.660034	1913.900024	1916.859985	1901.300049	2283000	1910.686686	0.135939
2019-06-25	1911.839966	1878.270020	1916.390015	1872.420044	3012300	1889.026693	-1.879191
2019-06-26	1892.479980	1897.829956	1903.800049	1887.319946	2441900	1896.316650	1.035995
2019-06-27	1902.000000	1904.280029	1911.239990	1898.040039	2141700	1904.520019	0.339289
2019-06-28	1909.099976	1893.630005	1912.939941	1884.000000	3037400	1896.856649	-0.560838

Date	Open	High	Low	Close	Adj Close	Volume	SP500
2016-01-04	2038.199951	2038.199951	1989.680054	2012.660034	2012.660034	4304880000	2013.513346
2016-01-05	2013.780029	2021.939941	2004.170044	2016.709961	2016.709961	3706620000	2014.273315
2016-01-06	2011.709961	2011.709961	1979.050049	1990.260010	1990.260010	4336660000	1993.673340
2016-01-07	1985.319946	1985.319946	1938.829956	1943.089966	1943.089966	5076590000	1955.746623
2016-01-08	1945.969971	1960.400024	1918.459961	1922.030029	1922.030029	4664940000	1933.630005

Date	SP500	Typical_Price
2016-01-04	2013.513346	640.739990
2016-01-05	2014.273315	636.116658
2016-01-06	1993.673340	630.916667
2016-01-07	1955.746623	614.383341
2016-01-08	1933.630005	612.396668
2016-01-11	1920.140015	612.053324
2016-01-12	1933.470012	618.706665
2016-01-13	1909.006673	593.949992
2016-01-14	1911.746664	588.376668
2016-01-15	1884.946655	573.366659

EMA for different time periods



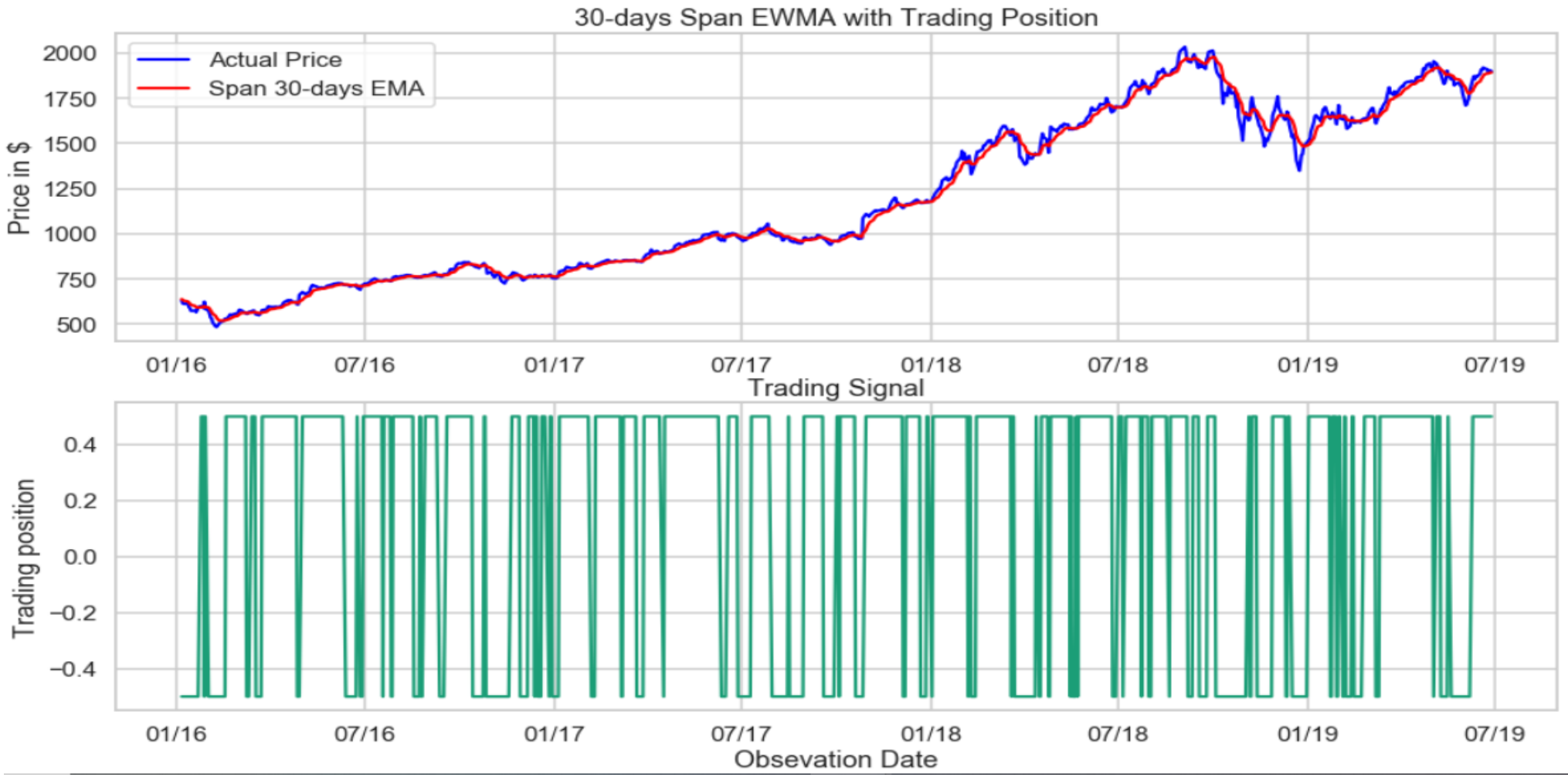
Trading Strategy for Moving Average

```
#implementation
trad_strat = ema_ten.apply(np.sign)

for index, row in trad_strat.iterrows():
    if ema_ten.loc[index,'Typical_Price'] > ema_thirty.loc[index,'Typical_Price'] and ema_twenty.loc[index,'Typical_Price'] 
        trad_strat.loc[index,'Typical_Price'] = 1
    elif ema_ten.loc[index,'Typical_Price'] < ema_thirty.loc[index,'Typical_Price'] and ema_twenty.loc[index,'Typical_Price'
        trad_strat.loc[index,'Typical_Price'] = -1
    else:
        trad_strat.loc[index,'Typical_Price'] = 0
```

```
# initial price 484.81
# last price 2034.33
initial_amt = 100000
total_shares = 30
print('Account balance ',initial_amt)
print('Initial number of shares owned ',total_shares)
avg_price = 484.81
initial_val = 114544.3  #initial amount+total_shares*avg_price

for index, row in trad_strat.iterrows():
    if trad_strat.loc[index,'Typical_Price'] == 1:
        if (initial_amt - df.loc[index,'Typical_Price'])> 0 :
            initial_amt -= df.loc[index,'Typical_Price']
            avg_price = ((avg_price*total_shares)+df.loc[index,'Typical_Price'])/(total_shares+1)
            total_shares = total_shares + 1;
            df.loc[index,'Signal'] = "Buy"
        else:
            df.loc[index,'Signal'] = "Buy Alert"
    elif trad_strat.loc[index,'Typical_Price'] == -1:
        if total_shares - 1 > 0 :
            df.loc[index,'Signal'] = "Sell"
            avg_price = ((avg_price*total_shares)-df.loc[index,'Typical_Price'])/(total_shares-1)
            initial_amt += df.loc[index,'Typical_Price'];
            total_shares = total_shares - 1;
        else:
            df.loc[index,'Signal'] = "Sell Alert"
    else:
        df.loc[index,'Signal'] = "Hold"

print('----------Signals Given During Day To Day Trade----------')

total_val = initial_amt + total_shares * 2034.33
```

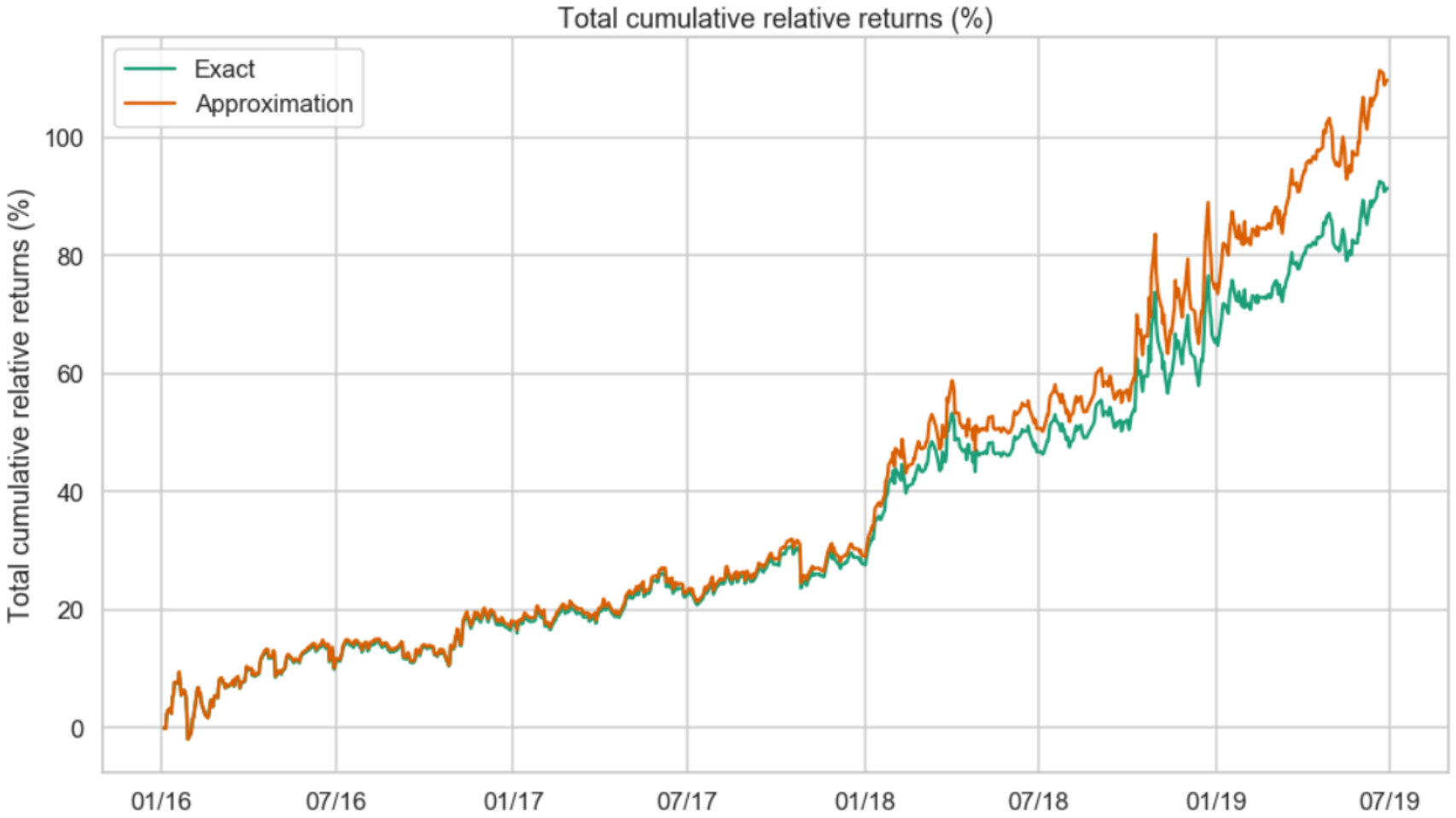
```
Account balance  100000
Initial number of shares owned  30
----------Signals Given During Day To Day Trade--
                SP500  Typical_Price Signal
Date
2016-01-04  2013.513346     640.739990   Hold
2016-01-05  2014.273315     636.116658   Sell
2016-01-06  1993.673340     630.916667   Sell
2016-01-07  1955.746623     614.383341   Sell
2016-01-08  1933.630005     612.396668   Sell
2016-01-11  1920.140015     612.053324   Sell
2016-01-12  1933.470012     618.706665   Sell
2016-01-13  1909.006673     593.949992   Sell
2016-01-14  1911.746664     588.376668   Sell
2016-01-15  1884.946655     573.366659   Sell
2016-01-19  1882.456624     574.976664   Sell
2016-01-20  1849.266683     565.800008   Sell
2016-01-21  1869.273315     577.349996   Sell
2016-01-22  1897.716675     593.529989   Sell
2016-01-25  1886.443319     599.863342   Sell
2016-01-26  1896.383341     598.710002   Sell
2016-01-27  1890.879964     588.506673   Sell
2016-01-28  1889.989990     623.653321   Sell
2016-01-29  1924.826660     583.333333   Sell
2016-02-01  1935.626668     575.639995   Sell
----------Results--------------------------
Balance left in Account 1794.5664839999683
Total Number of shares  147
Total value of the shares  300841.07648399996
Percetage Profit  162.6416822871151
```

Trading position for 30 days

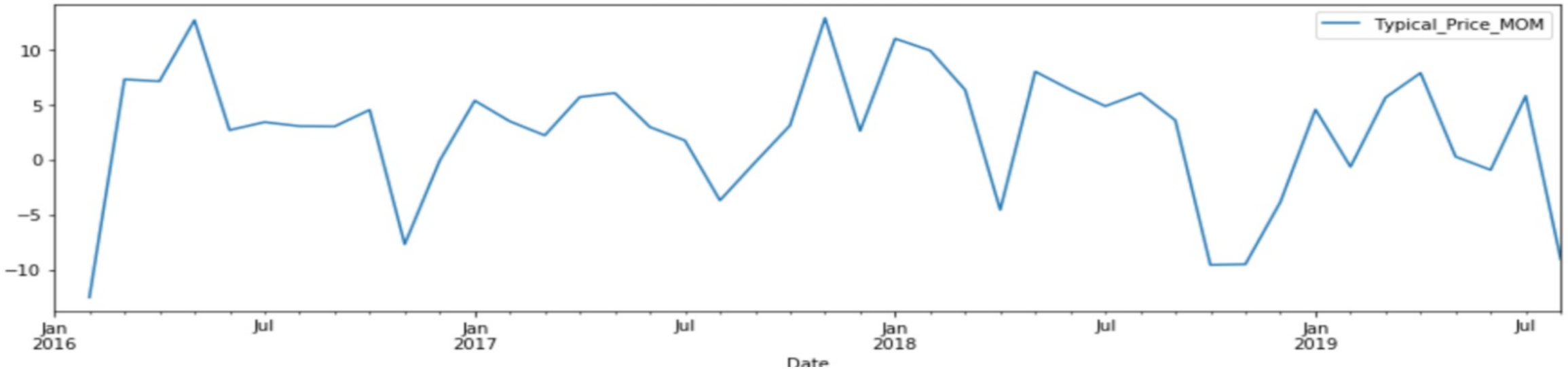


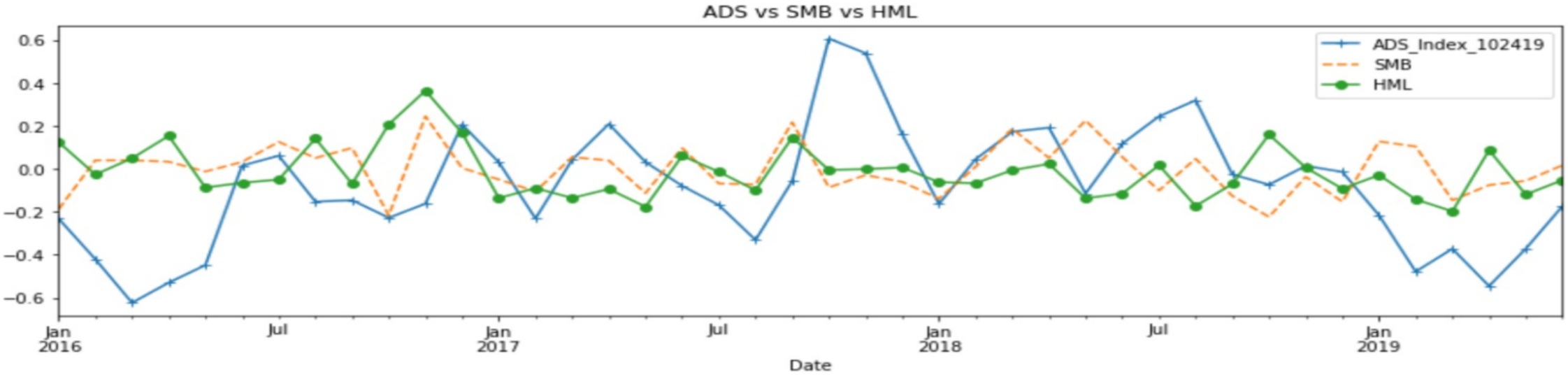
Relative returns

Cumulative Returns

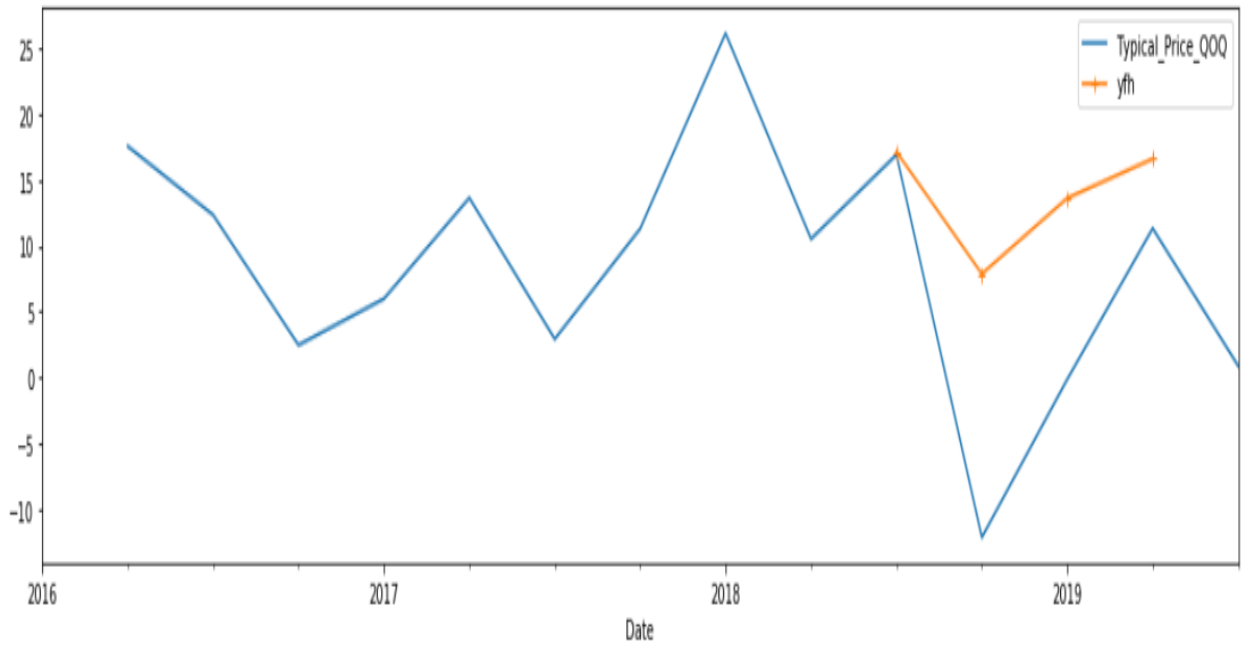


MIDAS

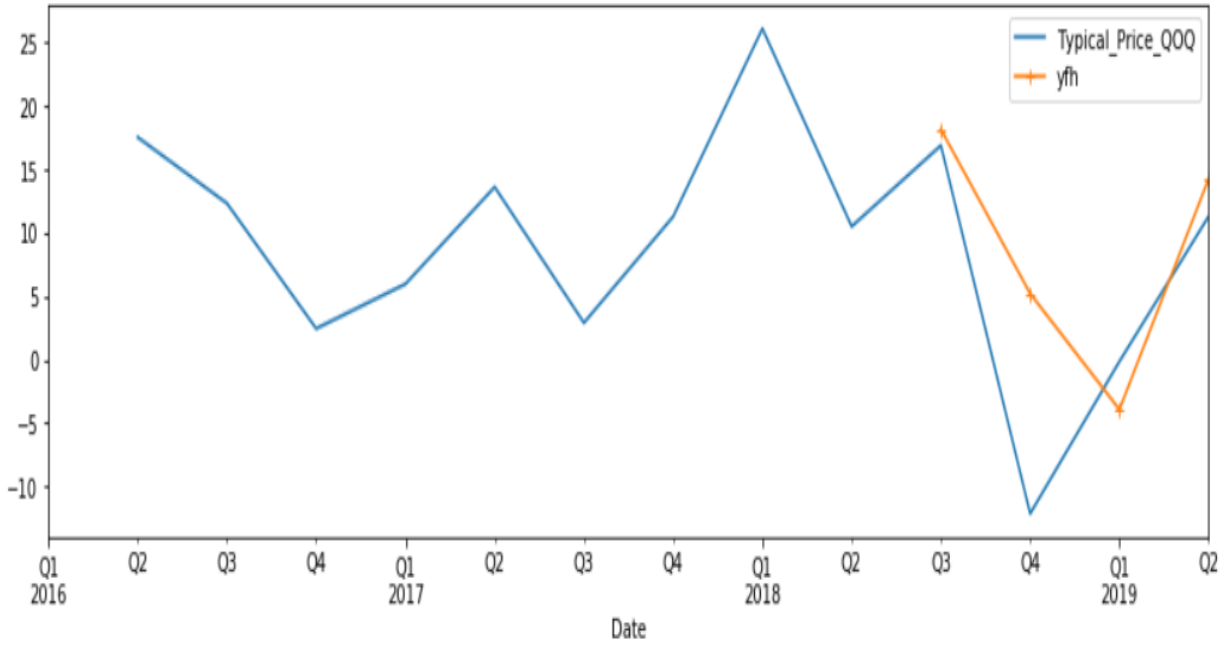

Typical_Price_MOM
10
5
0
−5
−10
Jan
2016
Jul
Jan
2017
Jul
Jan
2018
Jul
Jan
2019
Jul
Date


ADS vs SMB vs HML
ADS_Index_102419
SMB
HML
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
Jan
2016
Jul
Jan
2017
Jul
Jan
2018
Jul
Jan
2019
Date

Trial Models

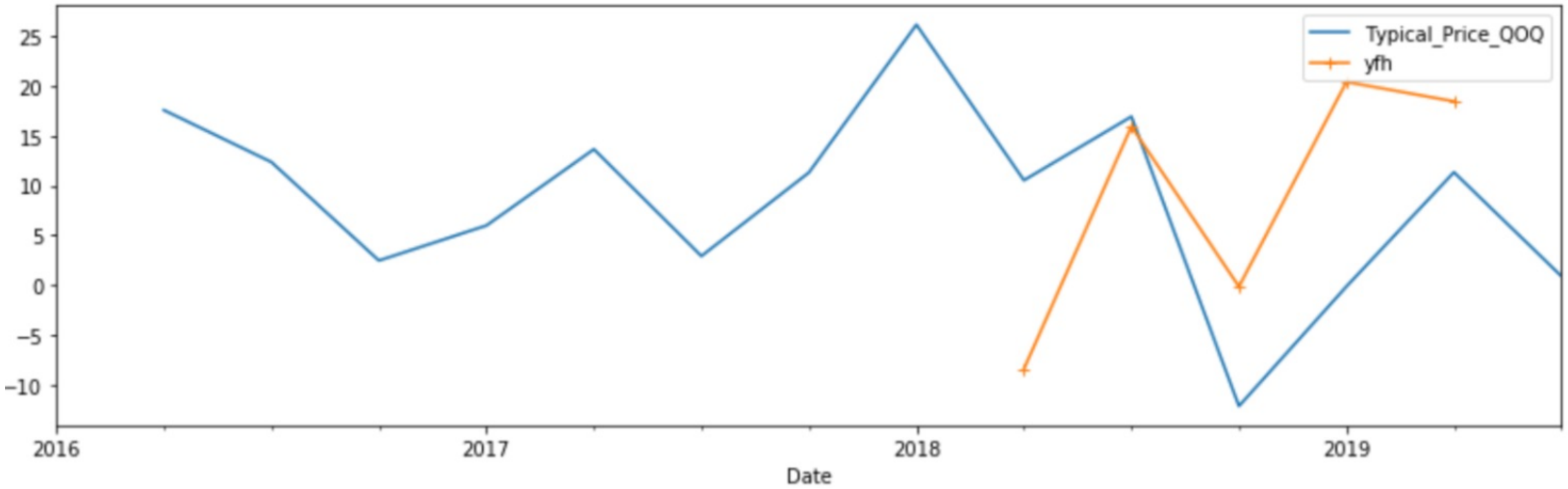


Lf = typical price mom
Hf = typical price qoq



Lf1 = typical price mom
Lf2 = volume mom
Hf = typical price qoq

Final Model



Lf = typical price mom

Hf = typical price qoq

RMSE - $13.96 for log returns

Strategy for MIDAS

Trading Strategy

```
# Buy if we have next days predicted_value greater than todays close value and hold if already bought
# Sell if we have next days predicted_value lesser than todays close value and dont buy until rule 1
```

```
signal = 0
amount = 10000
Amount = []
balance = 0
action = []
portfolio = 0
Portfolio = []
stocks = 0
Stocks = []


for i in range(len(Predicted)-1):
    if Predicted['Predicted_Close_Price'][i+1] > Predicted['Actual_Close_Price'][i]:
        if signal == 0:
            action.append('Buy')
            stocks = int(amount / Predicted['Actual_Close_Price'][i])
            balance = int(amount % Predicted['Actual_Close_Price'][i])
            portfolio = stocks * Predicted['Actual_Close_Price'][i]
            signal = 1
            amount = portfolio + balance
            print('Stock:',Predicted['Actual_Close_Price'][i] ,'Action:',action[i],'Portfolio:',round(portfolio,2),'
            Portfolio.append(round(portfolio,5))
            Amount.append(round(amount,0))
            Stocks.append(stocks)
        else:
            action.append('Bought--Holding')
            portfolio = stocks * Predicted['Actual_Close_Price'][i]
            amount = portfolio + balance
            print('Stock:',Predicted['Actual_Close_Price'][i],'Action:',action[i],'Portfolio:',round(portfolio,2),'S
            Portfolio.append(round(portfolio,5))
            Amount.append(round(amount,0))
            Stocks.append(stocks)
```

```
elif Predicted['Predicted_Close_Price'][i+1] < Predicted['Actual_Close_Price'][i]:
    if signal == 1:
        action.append('Sell')
        portfolio = stocks * Predicted['Actual_Close_Price'][i]

        signal = 0
        stocks = 0
        amount = balance + portfolio
        portfolio = 0
        balance = 0
        print('Stock:',Predicted['Actual_Close_Price'][i],'Action:',action[i],'Portfolio:',round(portfolio,2),'S
        Portfolio.append(round(portfolio,5))
        Amount.append(round(amount,0))
        Stocks.append(stocks)
    else:
        action.append('Price-Prediction-Already-Lower')
        print('Stock:',Predicted['Actual_Close_Price'][i],'Action:',action[i],'Portfolio:',round(portfolio,2),'S
        Portfolio.append(round(portfolio,5))
        Amount.append(round(amount,0))
        Stocks.append(stocks)
```

Quaterly Strategy Movement

```
Stock: 1879.5161914444452 Action: Price-Prediction-Already-Lower Portfolio: 0 Stocks: 0 Balance_init: 0 total($) 10
000
Stock: 1664.876507460317 Action: Buy Portfolio: 9989.26 Stocks: 6 Balance_init: 10 total($) 9999.26
Stock: 1662.7561262349723 Action: Bought--Holding Portfolio: 9976.54 Stocks: 6 Balance_init: 10 total($) 9986.54
Stock: 1862.1950760846562 Action: Bought--Holding Portfolio: 11173.17 Stocks: 6 Balance_init: 10 total($) 11183.17
```

Date	Actual_Typical_Price	Predicted_Typical_Price
2018-09-30	1879.516191	1587.068490
2018-12-31	1664.876507	1459.148698
2019-03-31	1662.756126	1710.977401
2019-06-30	1862.195076	1708.630883
2019-09-30	1879.050837	2095.002682

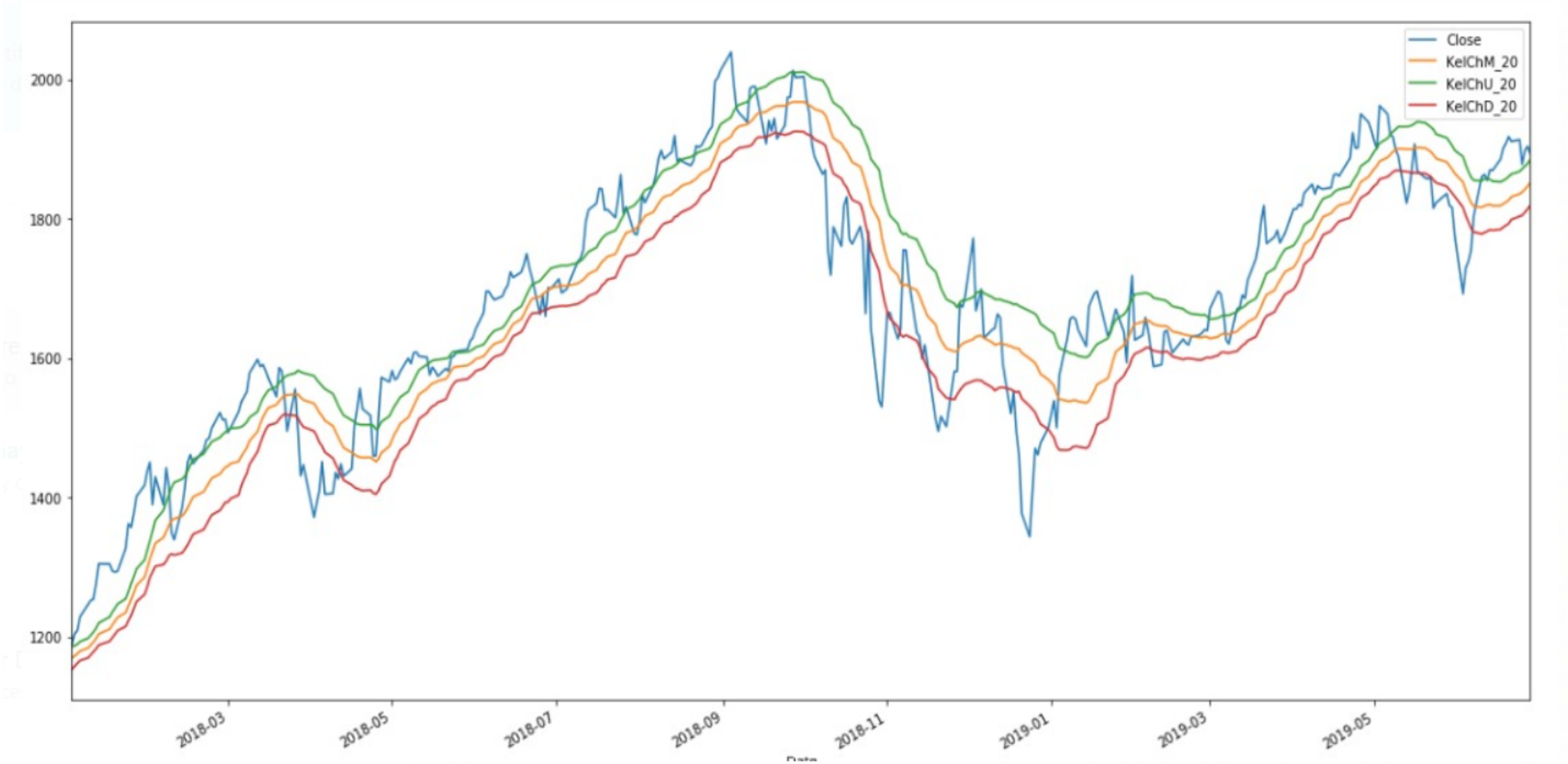
Keltner Channel

- A technical indicator which tells us the volatality of the market

- ATR (Highest of)
- Current high - current low
- Abs(current low) – previous close
- Abs(current high) – previous close

- Middle band = EMA
- Upper band = EMA + 2(ATR)
- Lower band = EMA – 2(ATR)

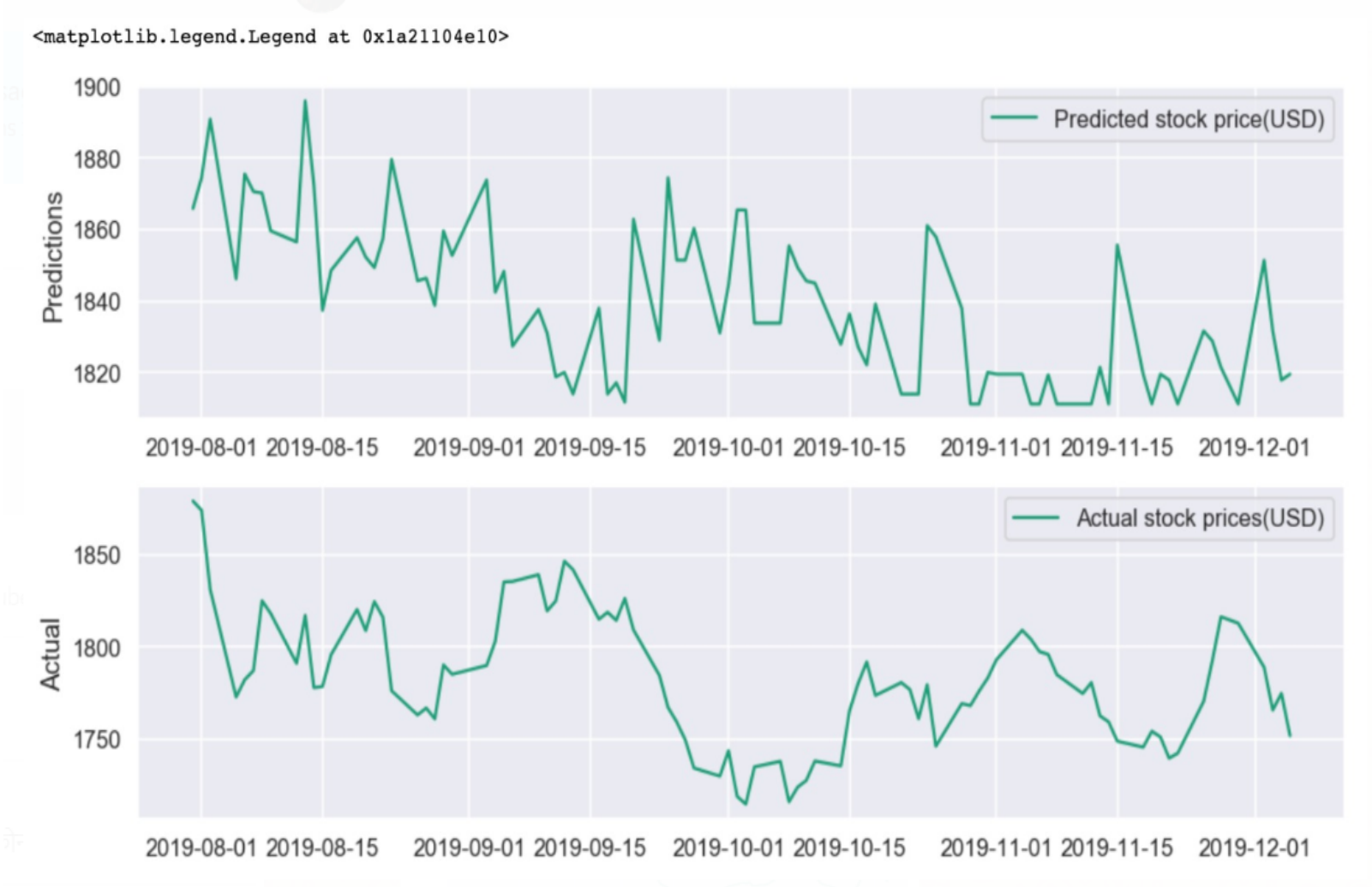
Keltner Channel



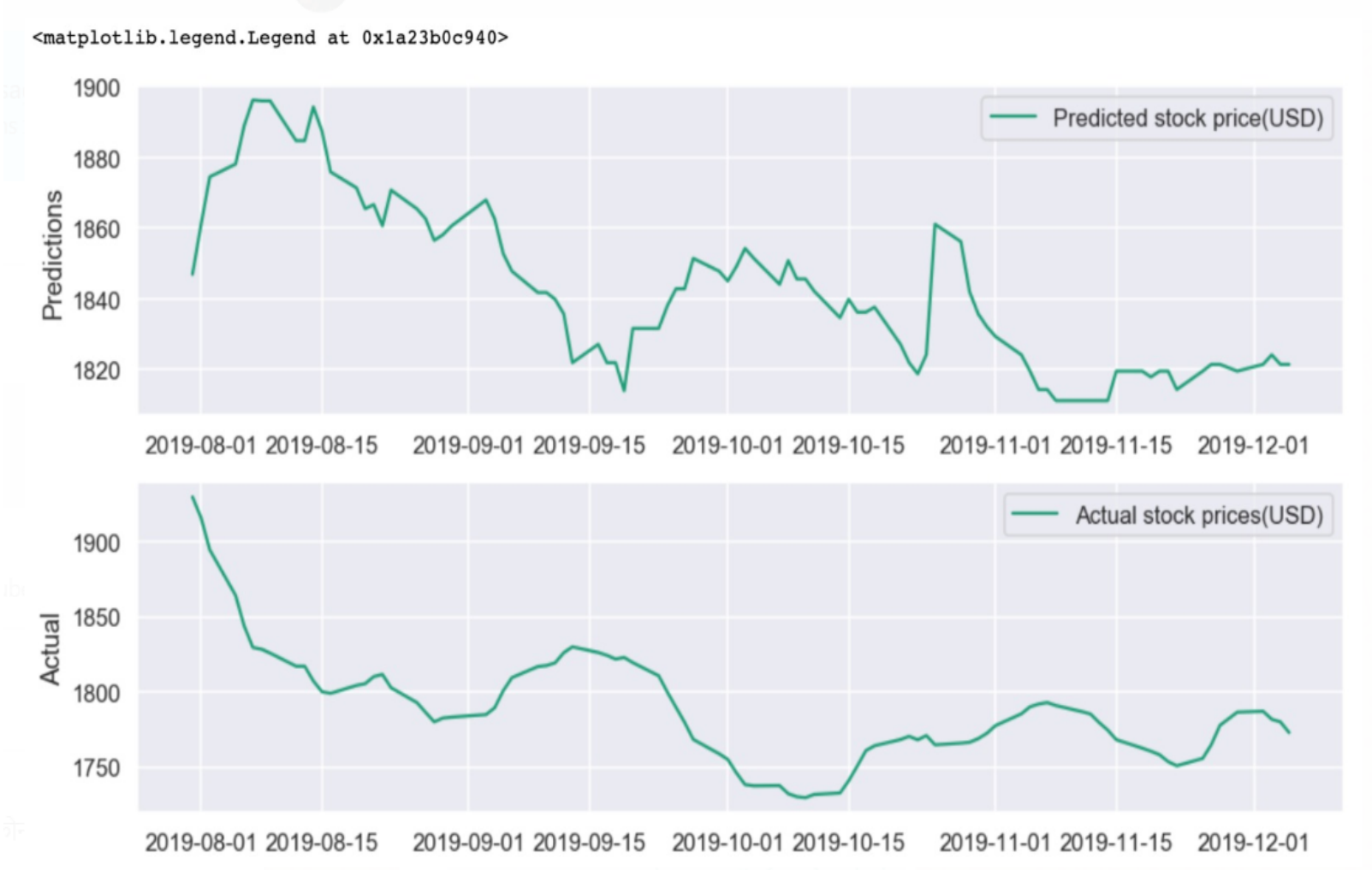
Random Forest

- Random is a meta data estimator that fits number of classifying decision trees on various sub samples of dataset and uses averaging to improve the predictive accuracy
- The basic idea is combining multiple decision trees in determining the final output rather than relying on individual decision trees.

Comparing predicted and actual values for last 90 days



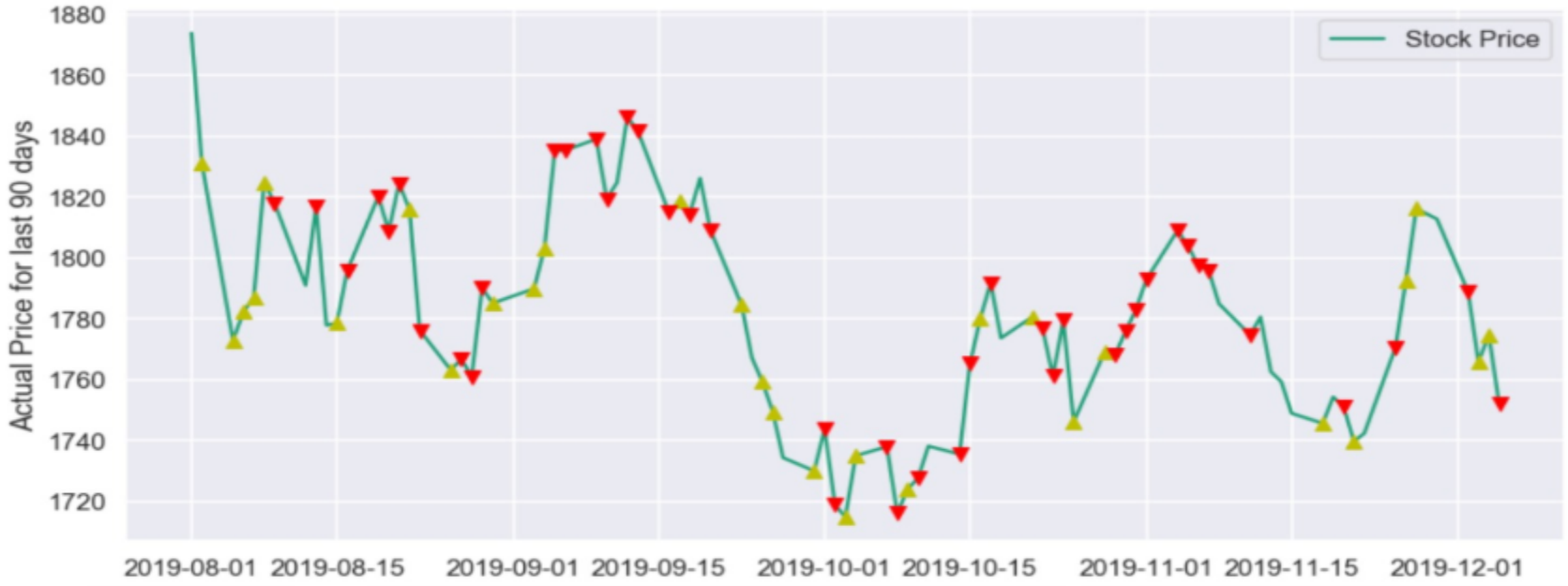
Comparison using Moving Average



Trading signals

In [58]:
```
fig, (ax2) = plt.subplots(1, 1, figsize=(16,7))
duration = 89
y_temp = pd.DataFrame(y_short.tail(duration),index =X.tail(duration).index)
ax2.plot(X.tail(duration).index,y.tail(duration),label="Stock Price")

ax2.set_ylabel("Actual Price for last 90 days")

ax2.legend(loc='best')

# # Plot the buy signals
ax2.plot(trading_signal_week.tail(duration).loc[trading_signal_week.signal == 1.0].index,
         y.tail(duration)[trading_signal_week.signal == 1.0],'^', markersize=10, color='y')

# Plot the sell signals
ax2.plot(trading_signal_week.tail(duration).loc[trading_signal_week.signal == -1.0].tail(duration).index,
         y.tail(duration)[trading_signal_week.signal == -1.0],'v', markersize=10, color='r')
```

Out[58]: [<matplotlib.lines.Line2D at 0x1a24789a58>]



Trading Strategy

```
# Set the initial capital
initial_capital= float(10000.0)

# Create a DataFrame 'positions'
positions = pd.DataFrame(index=trading_signal_week.index).fillna(0.0)

# Buy
positions['stock_price'] = trading_signal_week['signal']

# Initialize the portfolio with value owned
portfolio = positions.multiply(y_short.tail(90), axis=0)

# Store the difference in shares owned
pos_diff = positions.diff()

# Add 'holdings' to portfolio
portfolio['holdings'] = (positions.multiply(y_short.tail(90), axis=0)).sum(axis=1)

# Add 'cash' to portfolio
portfolio['cash'] = initial_capital - (pos_diff.multiply(y_short.tail(90), axis=0)).sum(axis=1).cumsum()

# Add 'total' & 'returns' to portfolio
portfolio['total'] = portfolio['cash'] + portfolio['holdings']

portfolio['returns'] = portfolio['total'].pct_change()
```

Returns

Date	stock_price	holdings	cash	total	returns
2019-12-03	1781.719017	1781.719017	10030.631928	11812.350945	4.513284e-04
2019-12-04	1779.982378	1779.982378	10030.631928	11810.614306	-1.470189e-04
2019-12-05	-1772.954282	-1772.954282	13576.540492	11803.586210	-5.950661e-04

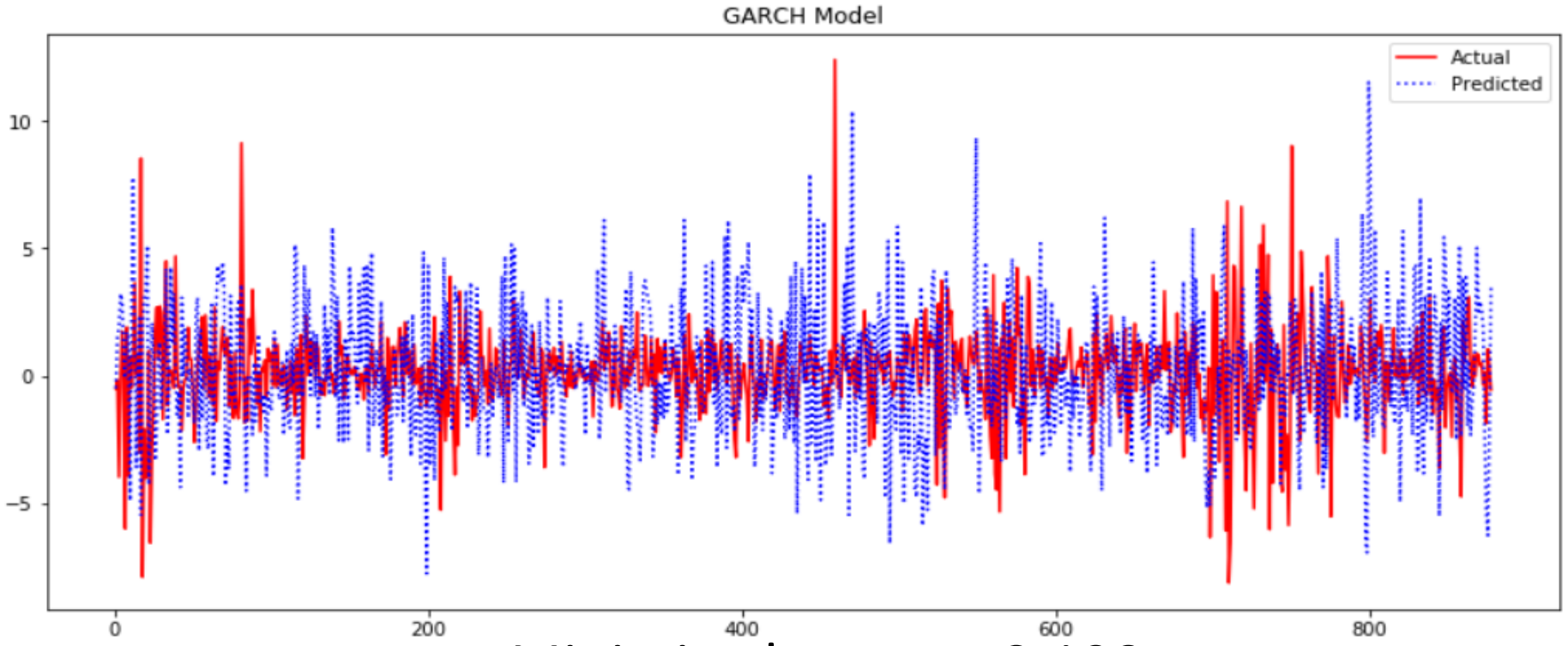
Time series prediction with LSTM

```python
# Part 2 - Building the RNN

# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout

# Initialising the RNN
regressor = Sequential()

# Adding the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

# Adding a second LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

```
Epoch 92/100
798/798 [==============================] - 16s 20ms/step - loss: 0.0020
Epoch 93/100
798/798 [==============================] - 16s 19ms/step - loss: 0.0019
Epoch 94/100
798/798 [==============================] - 18s 22ms/step - loss: 0.0019
Epoch 95/100
798/798 [==============================] - 20s 25ms/step - loss: 0.0021
Epoch 96/100
798/798 [==============================] - 23s 28ms/step - loss: 0.0020
Epoch 97/100
798/798 [==============================] - 20s 25ms/step - loss: 0.0020
Epoch 98/100
798/798 [==============================] - 18s 22ms/step - loss: 0.0019
Epoch 99/100
798/798 [==============================] - 16s 20ms/step - loss: 0.0021
Epoch 100/100
798/798 [==============================] - 20s 24ms/step - loss: 0.0019
```

Comparison of actual v/s predicted

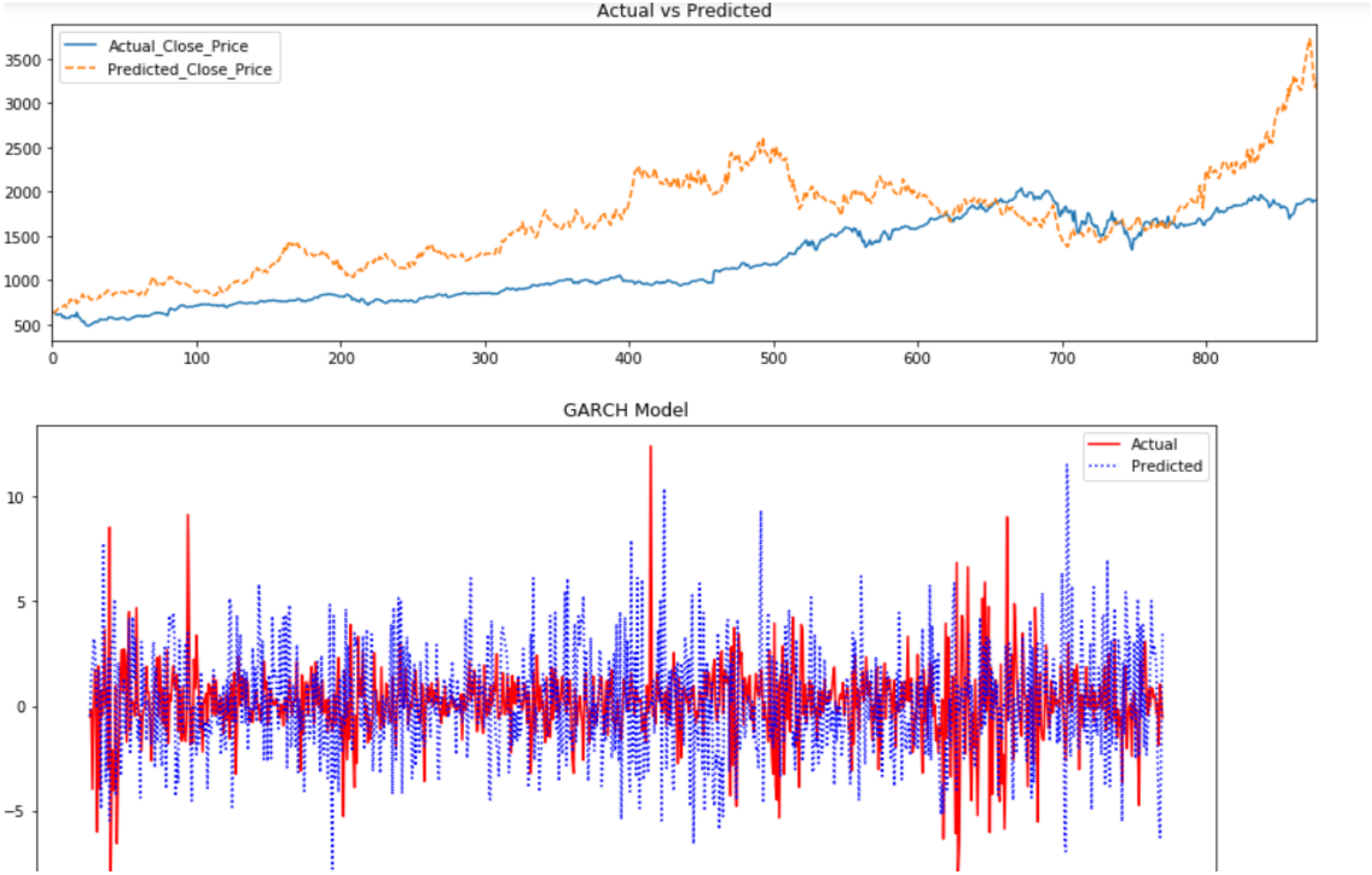
Garch



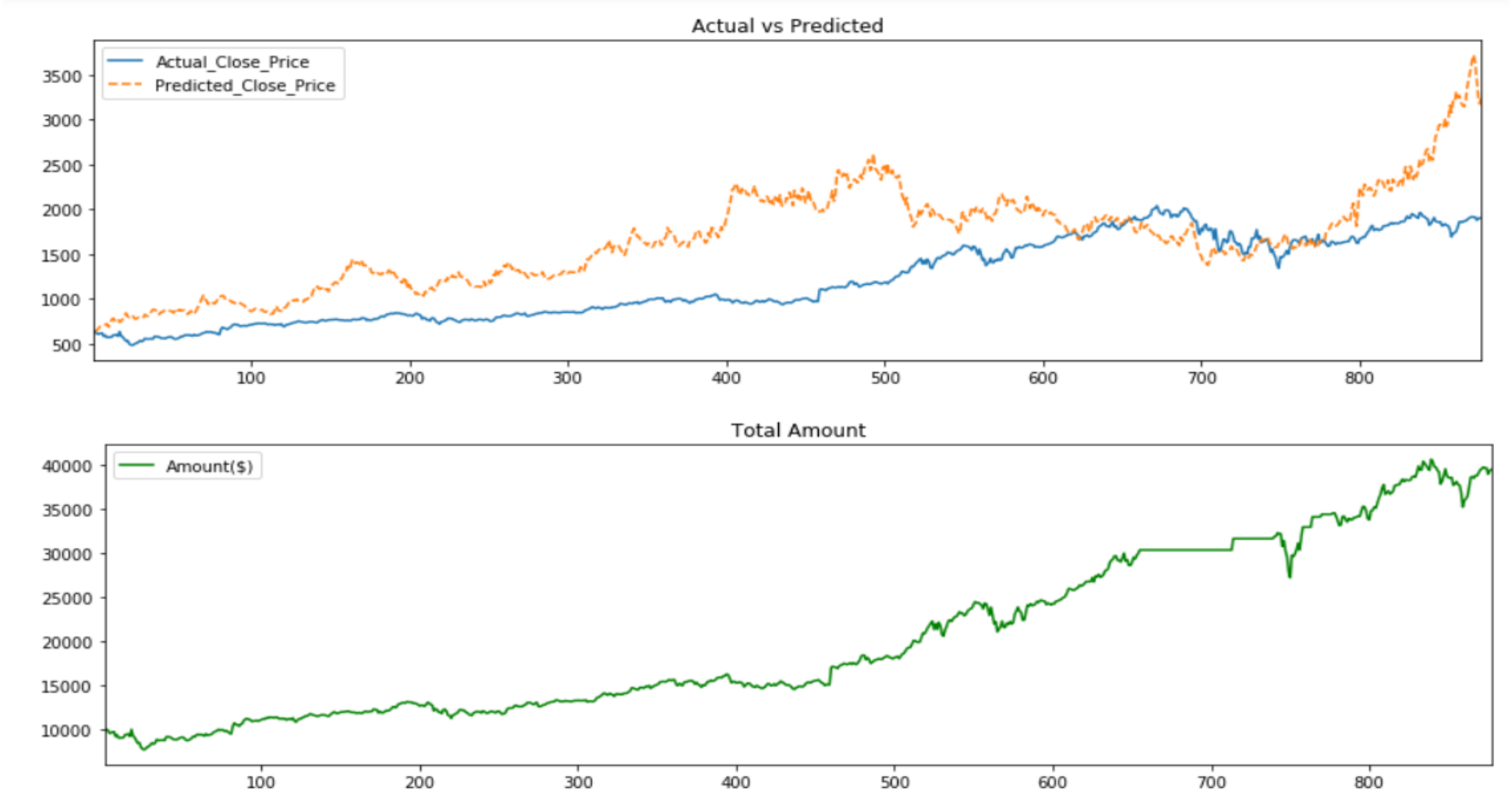
Minimized mean = 0.123

Standard Deviation = 1.88

Volatility measure against prediction



Portfolio Holdings



Trading Strategy for GARCH

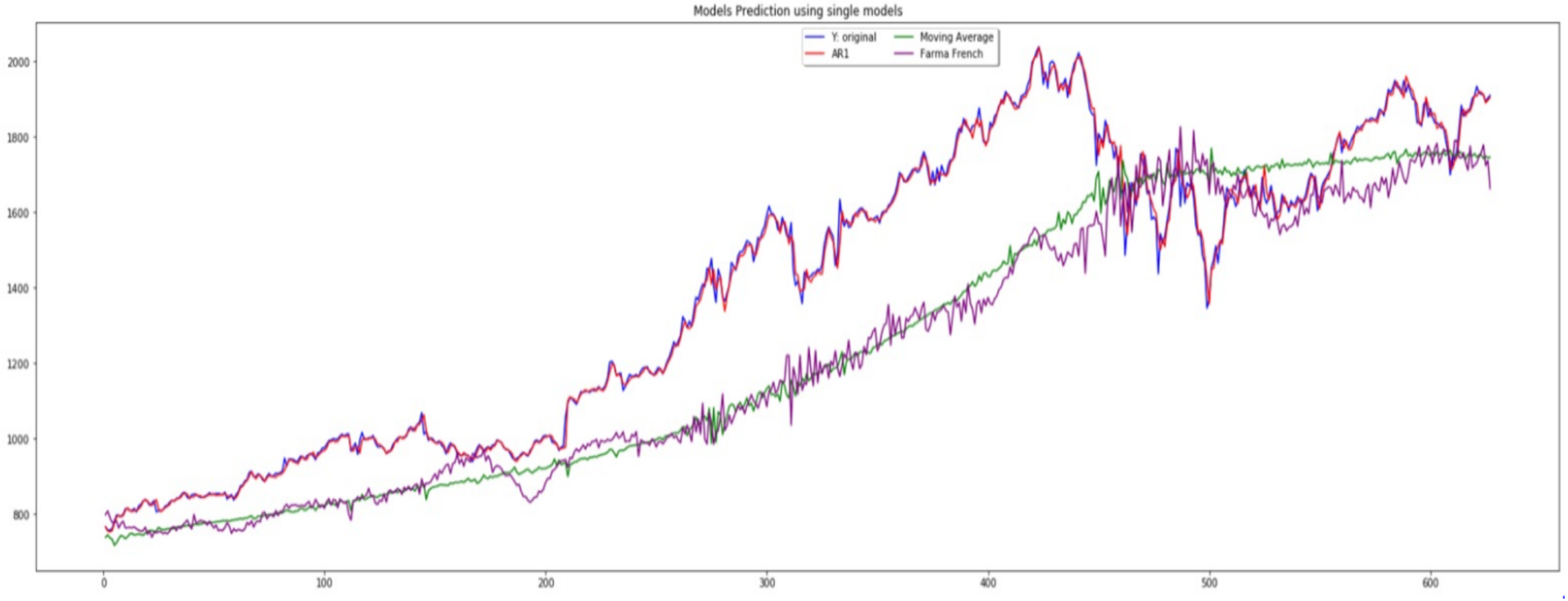
```
signal = 0
amount = 10000
Amount = []
balance = 0
action = []
portfolio = 0
Portfolio = []
stocks = 0
Stocks = []


for i in range(len(Predicted)-1):
    if Predicted['Predicted_Close_Price'][i+1] > Predicted['Actual_Close_Price'][i]:
        if signal == 0:
            action.append('Buy')
            stocks = int(amount / Predicted['Actual_Close_Price'][i])
            balance = int(amount % Predicted['Actual_Close_Price'][i])
            portfolio = stocks * Predicted['Actual_Close_Price'][i]
            signal = 1
            amount = portfolio + balance
            print('Stock:',Predicted['Actual_Close_Price'][i] ,'Action:',action[i],'Portfolio:',round(portfolio,2),'
            Portfolio.append(round(portfolio,5))
            Amount.append(round(amount,0))
            Stocks.append(stocks)
        else:
            action.append('Bought--Holding')
            portfolio = stocks * Predicted['Actual_Close_Price'][i]
            amount = portfolio + balance
            print('Stock:',Predicted['Actual_Close_Price'][i],'Action:',action[i],'Portfolio:',round(portfolio,2),'S
            Portfolio.append(round(portfolio,5))
            Amount.append(round(amount,0))
            Stocks.append(stocks)
```

```
elif Predicted['Predicted_Close_Price'][i+1] < Predicted['Actual_Close_Price'][i]:
    if signal == 1:
        action.append('Sell')
        portfolio = stocks * Predicted['Actual_Close_Price'][i]

        signal = 0
        stocks = 0
        amount = balance + portfolio
        portfolio = 0
        balance = 0
        print('Stock:',Predicted['Actual_Close_Price'][i],'Action:',action[i],'Portfolio:',round(portfolio,2),'S
        Portfolio.append(round(portfolio,5))
        Amount.append(round(amount,0))
        Stocks.append(stocks)
    else:
        action.append('Price-Prediction-Already-Lower')
        print('Stock:',Predicted['Actual_Close_Price'][i],'Action:',action[i],'Portfolio:',round(portfolio,2),'S
        Portfolio.append(round(portfolio,5))
        Amount.append(round(amount,0))
        Stocks.append(stocks)
```

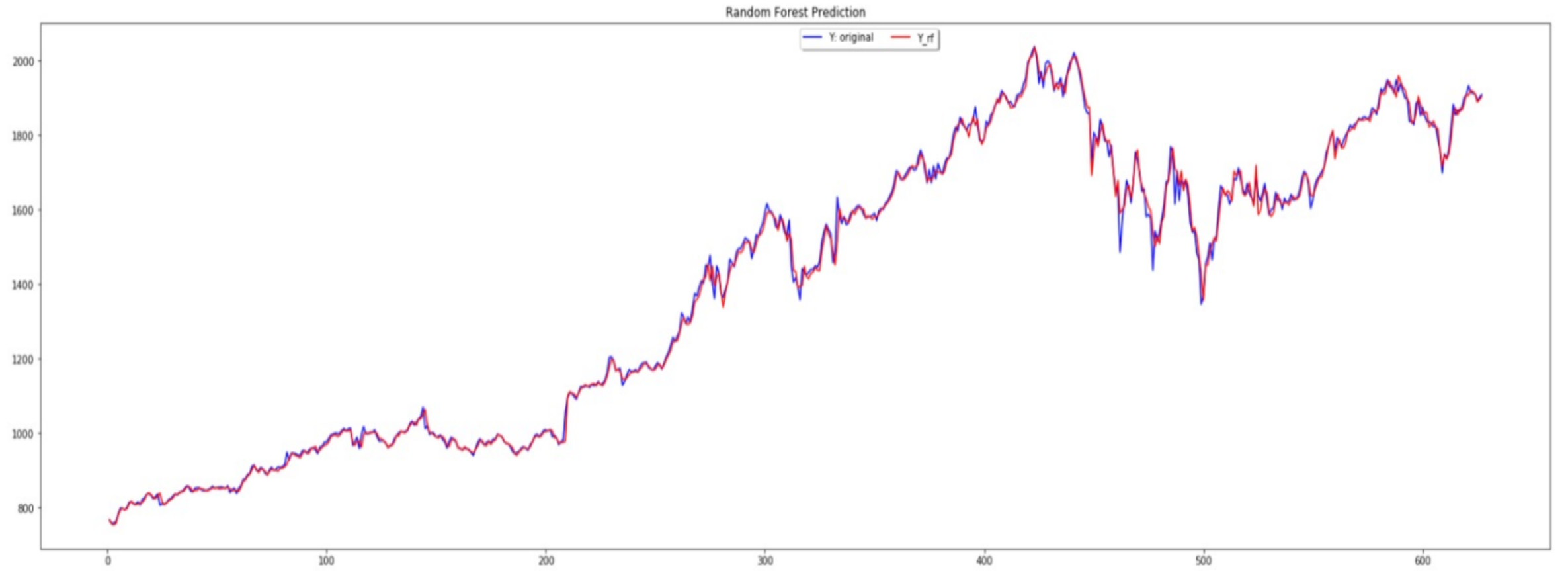
Daily strategy movements

```
Stock: 1870.300049 Action: Bought--Holding Portfolio: 37406.0 Stocks: 20 Balance_init: 1324 total($) 38730.0
Stock: 1869.670044 Action: Bought--Holding Portfolio: 37393.4 Stocks: 20 Balance_init: 1324 total($) 38717.4
Stock: 1886.030029 Action: Bought--Holding Portfolio: 37720.6 Stocks: 20 Balance_init: 1324 total($) 39044.6
Stock: 1901.369995 Action: Bought--Holding Portfolio: 38027.4 Stocks: 20 Balance_init: 1324 total($) 39351.4
Stock: 1908.790039 Action: Bought--Holding Portfolio: 38175.8 Stocks: 20 Balance_init: 1324 total($) 39499.8
Stock: 1918.189941 Action: Bought--Holding Portfolio: 38363.8 Stocks: 20 Balance_init: 1324 total($) 39687.8
Stock: 1911.300049 Action: Bought--Holding Portfolio: 38226.0 Stocks: 20 Balance_init: 1324 total($) 39550.0
Stock: 1913.900024 Action: Bought--Holding Portfolio: 38278.0 Stocks: 20 Balance_init: 1324 total($) 39602.0
Stock: 1878.27002 Action: Bought--Holding Portfolio: 37565.4 Stocks: 20 Balance_init: 1324 total($) 38889.4
Stock: 1897.829956 Action: Bought--Holding Portfolio: 37956.6 Stocks: 20 Balance_init: 1324 total($) 39280.6
Stock: 1904.280029 Action: Bought--Holding Portfolio: 38085.6 Stocks: 20 Balance_init: 1324 total($) 39409.6
```

Model selection using Random Forest



Random Forest prediction



RMSE for Random Forest

```
ar1_RMSE:  19.571529203681898
ma_RMSE:  232.87868522067123
FF_RMSE:  243.86301947915953
random forest rmse:  17.930652853681913
```

Performance Evaluation

Models	RMSE	Sharpe Ratio	Cost	Strategies
Kalman Filter	$41.47	0.59	$11,575	Long shot day trade
MIDAS	$13.96(log returns)		$11,000(holding)	Buy & hold quaterly
Random Forest	$65.44	0.96	$11,803	Long short Hold
GARCH	$40.18	2.29	$29,057	Buy & hold