

Linear_Regression

September 29, 2021

1 Linear Regression Indicator

<https://commodity.com/technical-analysis/lin-reg-line/>

<https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/linear-regression>

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")

# fix_yahoo_finance is used to fetch data
import fix_yahoo_finance as yf
yf.pdr_override()
```

```
[2]: # input
symbol1 = 'AAPL'
symbol2 = 'QQQ'
start = '2018-08-01'
end = '2019-01-01'

# Read data
df1 = yf.download(symbol1,start,end)
df2 = yf.download(symbol2,start,end)
```

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

```
[3]: # View Columns
df1.head()
```

```
[3]:
```

	Open	High	Low	Close	Adj Close	\
Date						
2018-08-01	199.130005	201.759995	197.309998	201.500000	198.478760	
2018-08-02	200.580002	208.380005	200.350006	207.389999	204.280457	

2018-08-03	207.029999	208.740005	205.479996	207.990005	204.871445
2018-08-06	208.000000	209.250000	207.070007	209.070007	205.935257
2018-08-07	209.320007	209.500000	206.759995	207.110001	204.004639

	Volume
Date	
2018-08-01	67935700
2018-08-02	62404000
2018-08-03	33447400
2018-08-06	25425400
2018-08-07	25587400

```
[4]: df2.head()
```

```
[4]:
```

	Open	High	Low	Close	Adj Close	\
Date						
2018-08-01	176.860001	177.649994	176.100006	177.119995	175.977173	
2018-08-02	175.869995	179.740005	175.789993	179.529999	178.371628	
2018-08-03	179.869995	180.089996	179.080002	180.080002	178.918091	
2018-08-06	179.960007	181.190002	179.740005	181.139999	179.971237	
2018-08-07	181.649994	182.139999	181.259995	181.800003	180.626999	

	Volume
Date	
2018-08-01	37101900
2018-08-02	47178200
2018-08-03	28934400
2018-08-06	24808800
2018-08-07	29895700

```
[5]: avg1 = df1['Adj Close'].mean()
avg2 = df2['Adj Close'].mean()
df1['AVGS1_S1'] = avg1 - df1['Adj Close']
df1['AVGS2_S2'] = avg2 - df2['Adj Close']
df1['Average_SQ'] = df1['AVGS1_S1']**2
df1['AVG_AVG'] = df1['AVGS1_S1']*df1['AVGS2_S2']
```

```
[6]: df1.head(20)
```

```
[6]:
```

	Open	High	Low	Close	Adj Close	\
Date						
2018-08-01	199.130005	201.759995	197.309998	201.500000	198.478760	
2018-08-02	200.580002	208.380005	200.350006	207.389999	204.280457	
2018-08-03	207.029999	208.740005	205.479996	207.990005	204.871445	
2018-08-06	208.000000	209.250000	207.070007	209.070007	205.935257	
2018-08-07	209.320007	209.500000	206.759995	207.110001	204.004639	
2018-08-08	206.050003	207.809998	204.520004	207.250000	204.142532	

2018-08-09	207.279999	209.779999	207.199997	208.880005	205.748108
2018-08-10	207.360001	209.100006	206.669998	207.529999	205.135254
2018-08-13	207.699997	210.949997	207.699997	208.869995	206.459793
2018-08-14	210.160004	210.559998	208.259995	209.750000	207.329651
2018-08-15	209.220001	210.740005	208.330002	210.240005	207.813995
2018-08-16	211.750000	213.809998	211.470001	213.320007	210.858459
2018-08-17	213.440002	217.949997	213.160004	217.580002	215.069290
2018-08-20	218.100006	219.179993	215.110001	215.460007	212.973755
2018-08-21	216.800003	217.190002	214.029999	215.039993	212.558609
2018-08-22	214.100006	216.360001	213.839996	215.050003	212.568481
2018-08-23	214.649994	217.050003	214.600006	215.490005	213.003418
2018-08-24	216.600006	216.899994	215.110001	216.160004	213.665680
2018-08-27	217.149994	218.740005	216.330002	217.940002	215.425140
2018-08-28	219.009995	220.539993	218.919998	219.699997	217.164825

	Volume	AVGS1_S1	AVGS2_S2	Average_SQ	AVG_AVG
Date					
2018-08-01	67935700	2.593095	-3.527169	6.724141	-9.146283
2018-08-02	62404000	-3.208602	-5.921624	10.295127	19.000134
2018-08-03	33447400	-3.799590	-6.468087	14.436884	24.576078
2018-08-06	25425400	-4.863402	-7.521233	23.652679	36.578778
2018-08-07	25587400	-2.932784	-8.176995	8.601222	23.981359
2018-08-08	22525500	-3.070677	-8.395562	9.429057	25.780058
2018-08-09	23469200	-4.676253	-8.286278	21.867342	38.748731
2018-08-10	24611200	-4.063399	-6.905236	16.511212	28.058728
2018-08-13	25869100	-5.387938	-6.706536	29.029876	36.134399
2018-08-14	20748000	-6.257796	-7.829247	39.160011	48.993829
2018-08-15	28807600	-6.742140	-5.623559	45.456452	37.914820
2018-08-16	28500400	-9.786604	-6.209771	95.777619	60.772567
2018-08-17	35427000	-13.997435	-6.249505	195.928188	87.477036
2018-08-20	30287700	-11.901900	-6.090523	141.655225	72.488792
2018-08-21	26159800	-11.486754	-6.746270	131.945518	77.492740
2018-08-22	19018100	-11.496626	-7.441765	132.172410	85.555186
2018-08-23	18883200	-11.931563	-7.183434	142.362197	85.709592
2018-08-24	18476400	-12.593825	-8.852593	158.604429	111.488003
2018-08-27	20525100	-14.353285	-10.700585	206.016791	153.588542
2018-08-28	22776800	-16.092970	-10.968850	258.983685	176.521369

```
[7]: sum_sq = df1['Average_SQ'].sum()
      sum_avg = df1['AVG_AVG'].sum()
      slope = sum_avg/sum_sq
      intercept = avg2-(slope*avg1)
```

```
[8]: df1['Linear_Regression'] = intercept + slope*(df1['Adj Close'])
```

```
[9]: df1 = df1.drop(['AVGS1_S1', 'AVGS2_S2', 'Average_SQ', 'AVG_AVG'], axis=1)
      df1.head()
```

```
[9]:
```

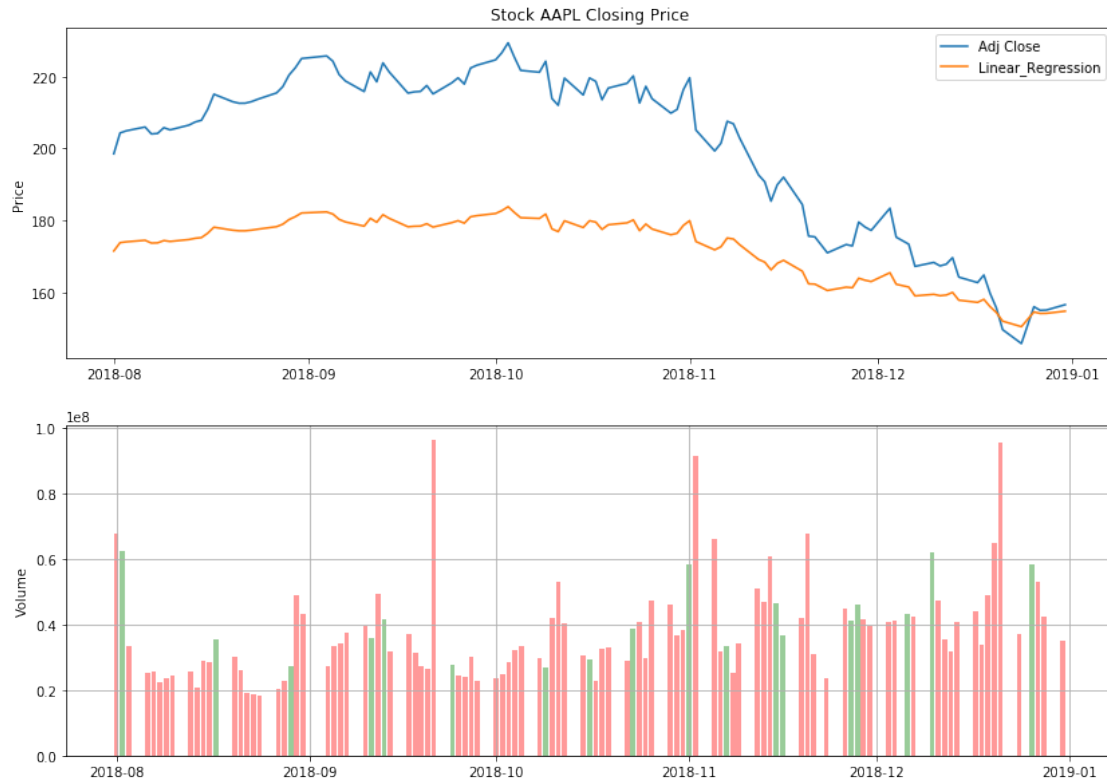
	Open	High	Low	Close	Adj Close \
Date					
2018-08-01	199.130005	201.759995	197.309998	201.500000	198.478760
2018-08-02	200.580002	208.380005	200.350006	207.389999	204.280457
2018-08-03	207.029999	208.740005	205.479996	207.990005	204.871445
2018-08-06	208.000000	209.250000	207.070007	209.070007	205.935257
2018-08-07	209.320007	209.500000	206.759995	207.110001	204.004639

	Volume	Linear_Regression
Date		
2018-08-01	67935700	171.415488
2018-08-02	62404000	173.730078
2018-08-03	33447400	173.965852
2018-08-06	25425400	174.390261
2018-08-07	25587400	173.620040

```
[10]: fig = plt.figure(figsize=(14,10))
ax1 = plt.subplot(2, 1, 1)
ax1.plot(df1['Adj Close'])
ax1.plot(df1['Linear_Regression'], label='Linear_Regression')
ax1.set_title('Stock ' + symbol1 + ' Closing Price')
ax1.set_ylabel('Price')
ax1.legend(loc='best')

ax2 = plt.subplot(2, 1, 2)
df1['VolumePositive'] = df1['Open'] < df1['Adj Close']
colors = df1.VolumePositive.map({True: 'g', False: 'r'})
ax2.bar(df1.index, df1['Volume'], color=colors, alpha=0.4)
ax2.grid()
ax2.set_ylabel('Volume')
```

```
[10]: Text(0,0.5,'Volume')
```



1.1 Candlestick with Linear Regression

```
[11]: from matplotlib import dates as mdates
import datetime as dt

dfc = df1.copy()
dfc['VolumePositive'] = dfc['Open'] < dfc['Adj Close']
#dfc = dfc.dropna()
dfc = dfc.reset_index()
dfc['Date'] = mdates.date2num(dfc['Date'].astype(dt.date))
dfc.head()
```

```
[11]:
```

	Date	Open	High	Low	Close	Adj Close \
0	736907.0	199.130005	201.759995	197.309998	201.500000	198.478760
1	736908.0	200.580002	208.380005	200.350006	207.389999	204.280457
2	736909.0	207.029999	208.740005	205.479996	207.990005	204.871445
3	736912.0	208.000000	209.250000	207.070007	209.070007	205.935257
4	736913.0	209.320007	209.500000	206.759995	207.110001	204.004639

	Volume	Linear_Regression	VolumePositive
0	67935700	171.415488	False
1	62404000	173.730078	True

2	33447400	173.965852	False
3	25425400	174.390261	False
4	25587400	173.620040	False

```
[12]: from mpl_finance import candlestick_ohlc

fig = plt.figure(figsize=(14,10))
ax1 = plt.subplot(2, 1, 1)
candlestick_ohlc(ax1,dfc.values, width=0.5, colorup='g', colordown='r', alpha=1.
↪0)
ax1.plot(df1['Linear_Regression'], label='Linear_Regression')
ax1.xaxis_date()
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%d-%m-%Y'))
ax1.grid(True, which='both')
ax1.minorticks_on()
ax1v = ax1.twinx()
colors = dfc.VolumePositive.map({True: 'g', False: 'r'})
ax1v.bar(dfc.Date, dfc['Volume'], color=colors, alpha=0.4)
ax1v.axes.yaxis.set_ticklabels([])
ax1v.set_ylim(0, 3*df1.Volume.max())
ax1.set_title('Stock ' + symbol1 + ' Closing Price')
ax1.set_ylabel('Price')
ax1.legend(loc='best')

ax2 = plt.subplot(2, 1, 2)
df1['VolumePositive'] = df1['Open'] < df1['Adj Close']
colors = df1.VolumePositive.map({True: 'g', False: 'r'})
ax2.bar(df1.index, df1['Volume'], color=colors, alpha=0.4)
ax2.grid()
ax2.set_ylabel('Volume')
```

```
[12]: Text(0,0.5,'Volume')
```

