Derivative_Linear_Equation

September 29, 2021

1 Derivative Linear Equation Stock Data

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

import warnings
warnings.filterwarnings("ignore")

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

[2]:
```
# input
symbol = 'AMD'
start = '2017-01-01'
end = '2019-01-01'

# Read data
dataset = yf.download(symbol,start,end)['Adj Close']

# View Columns
dataset.head()
```

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

[2]:
```
Date
2017-01-03    11.43
2017-01-04    11.43
2017-01-05    11.24
2017-01-06    11.32
2017-01-09    11.49
Name: Adj Close, dtype: float64
```

[3]:
```
df = dataset.reset_index()
```

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

[4]:
```
        Date  Adj Close
0 2017-01-03      11.43
1 2017-01-04      11.43
2 2017-01-05      11.24
3 2017-01-06      11.32
4 2017-01-09      11.49
```

[5]:
```
df.tail()
```

[5]:
```
           Date  Adj Close
497 2018-12-24  16.650000
498 2018-12-26  17.900000
499 2018-12-27  17.490000
500 2018-12-28  17.820000
501 2018-12-31  18.459999
```

[6]:
```
max_p = df['Adj Close'].max()
min_p = df['Adj Close'].min()
avg_p = df['Adj Close'].mean()
```

[7]:
```
data = df.drop(['Date'], axis=1)
data
```

[7]:
```
     Adj Close
0    11.430000
1    11.430000
2    11.240000
3    11.320000
4    11.490000
5    11.440000
6    11.200000
7    10.760000
8    10.580000
9     9.820000
10    9.880000
11    9.770000
12    9.750000
13    9.910000
14   10.440000
15   10.350000
16   10.520000
17   10.670000
18   10.610000
19   10.370000
20   12.060000
```

```
21   12.280000
22   12.240000
23   13.630000
24   13.290000
25   13.560000
26   13.420000
27   13.580000
28   13.490000
29   13.260000
..         …
472  21.490000
473  20.660000
474  19.110001
475  19.209999
476  18.730000
477  19.379999
478  20.080000
479  21.049999
480  21.340000
481  21.430000
482  21.299999
483  23.709999
484  21.120001
485  21.299999
486  19.459999
487  19.990000
488  19.980000
489  20.480000
490  19.860001
491  19.900000
492  18.830000
493  19.500000
494  18.160000
495  17.940001
496  16.930000
497  16.650000
498  17.900000
499  17.490000
500  17.820000
501  18.459999

[502 rows x 1 columns]
```

[8]:
```
data = data.reset_index()
```

[9]:
```
data.as_matrix()
```

[9]:
```
array([[  0.        ,  11.43000031],
       [  1.        ,  11.43000031],
       [  2.        ,  11.23999977],
       …,
       [499.        ,  17.48999977],
       [500.        ,  17.81999969],
       [501.        ,  18.45999908]])
```

[10]:
```
from numpy import ones,vstack
from numpy.linalg import lstsq
```

[11]:
```
points = data.as_matrix()
```

[12]:
```
x_coords, y_coords = zip(*points)
A = vstack([x_coords,ones(len(x_coords))]).T
m, c = lstsq(A, y_coords)[0]
```

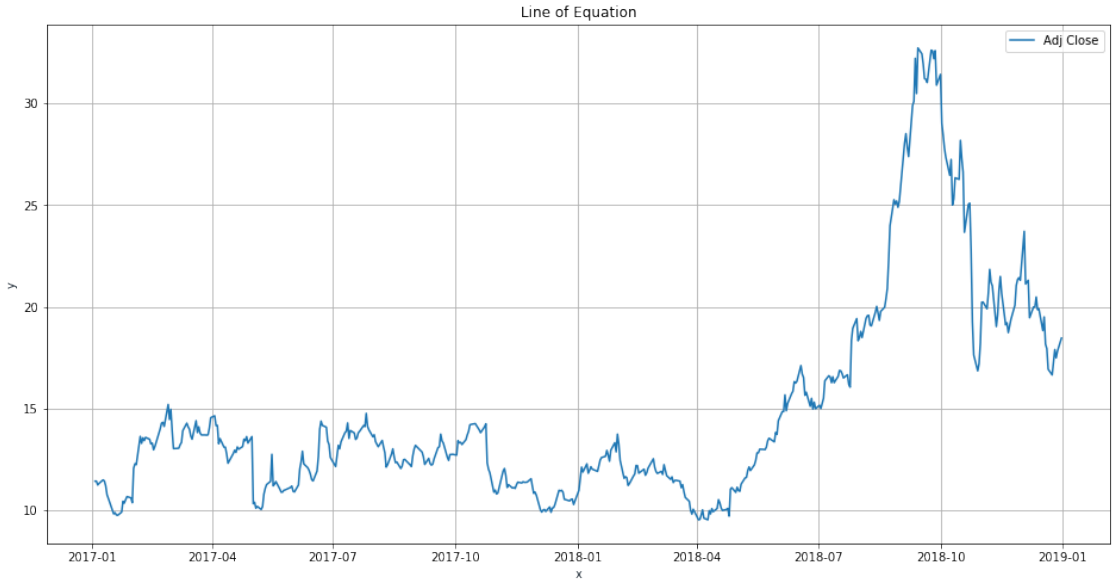
[13]:
```
print("Line Equation is y = {m}x + {c}".format(m=m,c=c))
```

```
Line Equation is y = 0.021718614923358824x + 9.372574584656498
```

[14]:
```
equation_of_line = print("y = {m}x + {c}".format(m=m,c=c))
```

```
y = 0.021718614923358824x + 9.372574584656498
```

[15]:
```
plt.figure(figsize=(16,8))
plt.plot(dataset)
plt.title('Line of Equation', equation_of_line)
plt.xlabel('x', color='#1C2833')
plt.ylabel('y', color='#1C2833')
plt.legend(loc='best')
plt.grid()
plt.show()
```



[16]:
```
from sympy import *
```

[17]:
```
x = Symbol('x')
```

[18]:
```
y = 0.021718614923358824*x + 9.372574584656498
```

[19]:
```
yder = y.diff(x)
yder
```

[19]:
```
0.0217186149233588
```

[20]:
```
y =  0.021718614923358824*(df.index) + 9.372574584656498
```

[21]:
```
y
```

[21]:
```
Float64Index([ 9.372574584656498,  9.394293199579856,  9.416011814503216,
                9.437730429426574,  9.459449044349933,  9.481167659273291,
                9.502886274196651,   9.52460488912001,  9.546323504043368,
                9.568042118966726,
               …
               20.058133126949038, 20.079851741872396, 20.101570356795754,
               20.123288971719116, 20.145007586642475, 20.166726201565833,
                20.18844481648919, 20.210163431412553,  20.23188204633591,
                20.25360066125927],
              dtype='float64', length=502)
```

[22]:
```
pd.DataFrame(y, columns=['Forecast'])
```

[22]:

	Forecast
0	9.372575
1	9.394293
2	9.416012
3	9.437730
4	9.459449
5	9.481168
6	9.502886
7	9.524605
8	9.546324
9	9.568042
10	9.589761
11	9.611479
12	9.633198
13	9.654917
14	9.676635
15	9.698354
16	9.720072
17	9.741791
18	9.763510
19	9.785228
20	9.806947
21	9.828665
22	9.850384
23	9.872103
24	9.893821
25	9.915540
26	9.937259
27	9.958977
28	9.980696
29	10.002414
..	…
472	19.623761
473	19.645479
474	19.667198
475	19.688917
476	19.710635
477	19.732354
478	19.754073
479	19.775791
480	19.797510
481	19.819228
482	19.840947
483	19.862666
484	19.884384
485	19.906103
486	19.927821

```
487  19.949540
488  19.971259
489  19.992977
490  20.014696
491  20.036415
492  20.058133
493  20.079852
494  20.101570
495  20.123289
496  20.145008
497  20.166726
498  20.188445
499  20.210163
500  20.231882
501  20.253601

[502 rows x 1 columns]
```

[23]:
```
dataset
```

[23]:
```
Date
2017-01-03    11.430000
2017-01-04    11.430000
2017-01-05    11.240000
2017-01-06    11.320000
2017-01-09    11.490000
2017-01-10    11.440000
2017-01-11    11.200000
2017-01-12    10.760000
2017-01-13    10.580000
2017-01-17     9.820000
2017-01-18     9.880000
2017-01-19     9.770000
2017-01-20     9.750000
2017-01-23     9.910000
2017-01-24    10.440000
2017-01-25    10.350000
2017-01-26    10.520000
2017-01-27    10.670000
2017-01-30    10.610000
2017-01-31    10.370000
2017-02-01    12.060000
2017-02-02    12.280000
2017-02-03    12.240000
2017-02-06    13.630000
2017-02-07    13.290000
2017-02-08    13.560000
```

```
2017-02-09    13.420000
2017-02-10    13.580000
2017-02-13    13.490000
2017-02-14    13.260000
                 …
2018-11-15    21.490000
2018-11-16    20.660000
2018-11-19    19.110001
2018-11-20    19.209999
2018-11-21    18.730000
2018-11-23    19.379999
2018-11-26    20.080000
2018-11-27    21.049999
2018-11-28    21.340000
2018-11-29    21.430000
2018-11-30    21.299999
2018-12-03    23.709999
2018-12-04    21.120001
2018-12-06    21.299999
2018-12-07    19.459999
2018-12-10    19.990000
2018-12-11    19.980000
2018-12-12    20.480000
2018-12-13    19.860001
2018-12-14    19.900000
2018-12-17    18.830000
2018-12-18    19.500000
2018-12-19    18.160000
2018-12-20    17.940001
2018-12-21    16.930000
2018-12-24    16.650000
2018-12-26    17.900000
2018-12-27    17.490000
2018-12-28    17.820000
2018-12-31    18.459999
Name: Adj Close, Length: 502, dtype: float64
```

[24]:
```
forecast = pd.DataFrame(y, columns=['Forecast'])
forecast
```

[24]:
```
      Forecast
0     9.372575
1     9.394293
2     9.416012
3     9.437730
4     9.459449
5     9.481168
```

```
6     9.502886
7     9.524605
8     9.546324
9     9.568042
10    9.589761
11    9.611479
12    9.633198
13    9.654917
14    9.676635
15    9.698354
16    9.720072
17    9.741791
18    9.763510
19    9.785228
20    9.806947
21    9.828665
22    9.850384
23    9.872103
24    9.893821
25    9.915540
26    9.937259
27    9.958977
28    9.980696
29   10.002414
..         …
472  19.623761
473  19.645479
474  19.667198
475  19.688917
476  19.710635
477  19.732354
478  19.754073
479  19.775791
480  19.797510
481  19.819228
482  19.840947
483  19.862666
484  19.884384
485  19.906103
486  19.927821
487  19.949540
488  19.971259
489  19.992977
490  20.014696
491  20.036415
492  20.058133
493  20.079852
```

```
494  20.101570
495  20.123289
496  20.145008
497  20.166726
498  20.188445
499  20.210163
500  20.231882
501  20.253601

[502 rows x 1 columns]
```

[25]:
```
df = dataset.reset_index()
```

[26]:
```
df = df.join(forecast)
```

[27]:
```
df
```

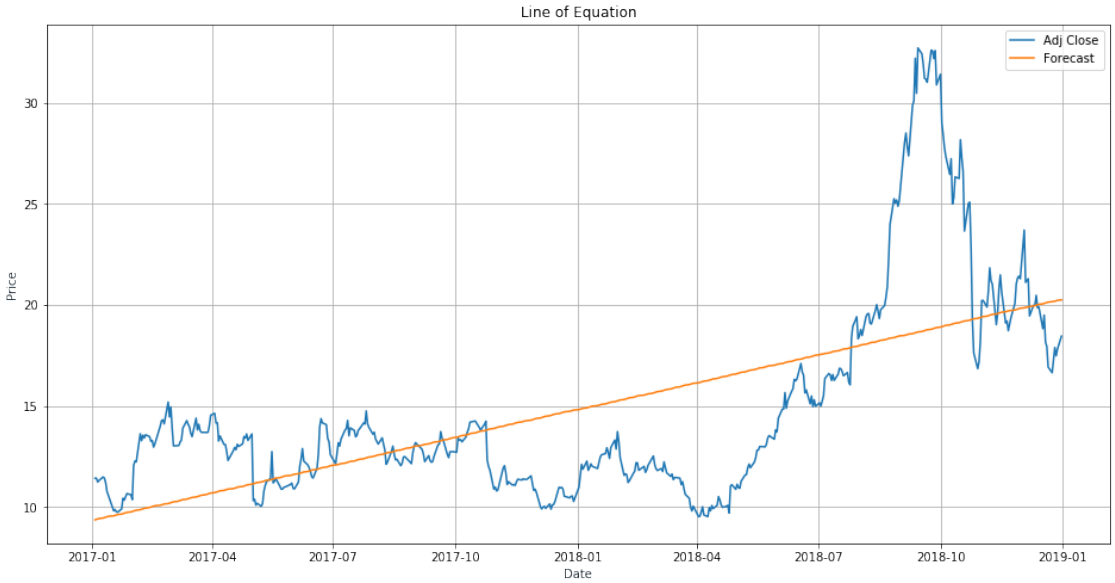
[27]:

	Date	Adj Close	Forecast
0	2017-01-03	11.430000	9.372575
1	2017-01-04	11.430000	9.394293
2	2017-01-05	11.240000	9.416012
3	2017-01-06	11.320000	9.437730
4	2017-01-09	11.490000	9.459449
5	2017-01-10	11.440000	9.481168
6	2017-01-11	11.200000	9.502886
7	2017-01-12	10.760000	9.524605
8	2017-01-13	10.580000	9.546324
9	2017-01-17	9.820000	9.568042
10	2017-01-18	9.880000	9.589761
11	2017-01-19	9.770000	9.611479
12	2017-01-20	9.750000	9.633198
13	2017-01-23	9.910000	9.654917
14	2017-01-24	10.440000	9.676635
15	2017-01-25	10.350000	9.698354
16	2017-01-26	10.520000	9.720072
17	2017-01-27	10.670000	9.741791
18	2017-01-30	10.610000	9.763510
19	2017-01-31	10.370000	9.785228
20	2017-02-01	12.060000	9.806947
21	2017-02-02	12.280000	9.828665
22	2017-02-03	12.240000	9.850384
23	2017-02-06	13.630000	9.872103
24	2017-02-07	13.290000	9.893821
25	2017-02-08	13.560000	9.915540
26	2017-02-09	13.420000	9.937259
27	2017-02-10	13.580000	9.958977
28	2017-02-13	13.490000	9.980696

```
29  2017-02-14  13.260000  10.002414
..         …         …         …
472 2018-11-15  21.490000  19.623761
473 2018-11-16  20.660000  19.645479
474 2018-11-19  19.110001  19.667198
475 2018-11-20  19.209999  19.688917
476 2018-11-21  18.730000  19.710635
477 2018-11-23  19.379999  19.732354
478 2018-11-26  20.080000  19.754073
479 2018-11-27  21.049999  19.775791
480 2018-11-28  21.340000  19.797510
481 2018-11-29  21.430000  19.819228
482 2018-11-30  21.299999  19.840947
483 2018-12-03  23.709999  19.862666
484 2018-12-04  21.120001  19.884384
485 2018-12-06  21.299999  19.906103
486 2018-12-07  19.459999  19.927821
487 2018-12-10  19.990000  19.949540
488 2018-12-11  19.980000  19.971259
489 2018-12-12  20.480000  19.992977
490 2018-12-13  19.860001  20.014696
491 2018-12-14  19.900000  20.036415
492 2018-12-17  18.830000  20.058133
493 2018-12-18  19.500000  20.079852
494 2018-12-19  18.160000  20.101570
495 2018-12-20  17.940001  20.123289
496 2018-12-21  16.930000  20.145008
497 2018-12-24  16.650000  20.166726
498 2018-12-26  17.900000  20.188445
499 2018-12-27  17.490000  20.210163
500 2018-12-28  17.820000  20.231882
501 2018-12-31  18.459999  20.253601

[502 rows x 3 columns]
```

[29]:
```
plt.figure(figsize=(16,8))
plt.plot(df.Date, df['Adj Close'])
plt.plot(df.Date, df['Forecast'])
plt.title('Line of Equation', equation_of_line)
plt.xlabel('Date', color='#1C2833')
plt.ylabel('Price', color='#1C2833')
plt.legend(loc='best')
plt.grid()
plt.show()
```


Line of Equation
Adj Close
Forecast
Price
Date
30
25
20
15
10
2017-01
2017-04
2017-07
2017-10
2018-01
2018-04
2018-07
2018-10
2019-01