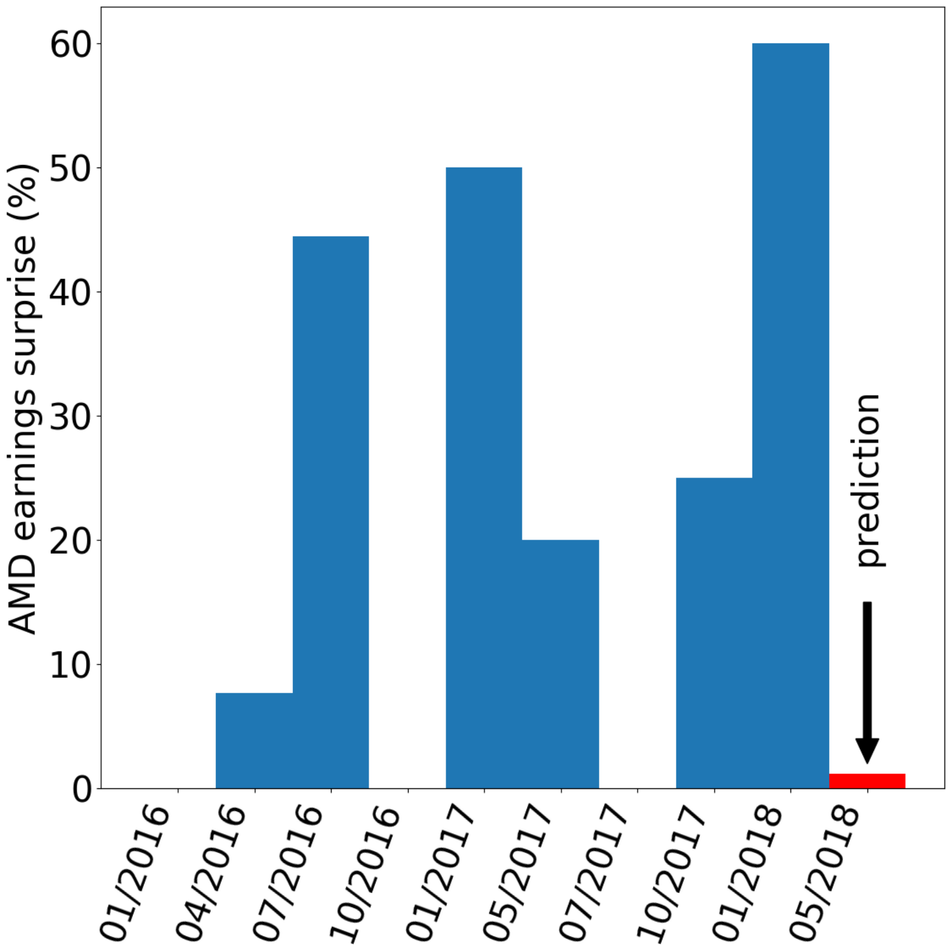
MACHINE LEARNING FOR FINANCE IN PYTHON

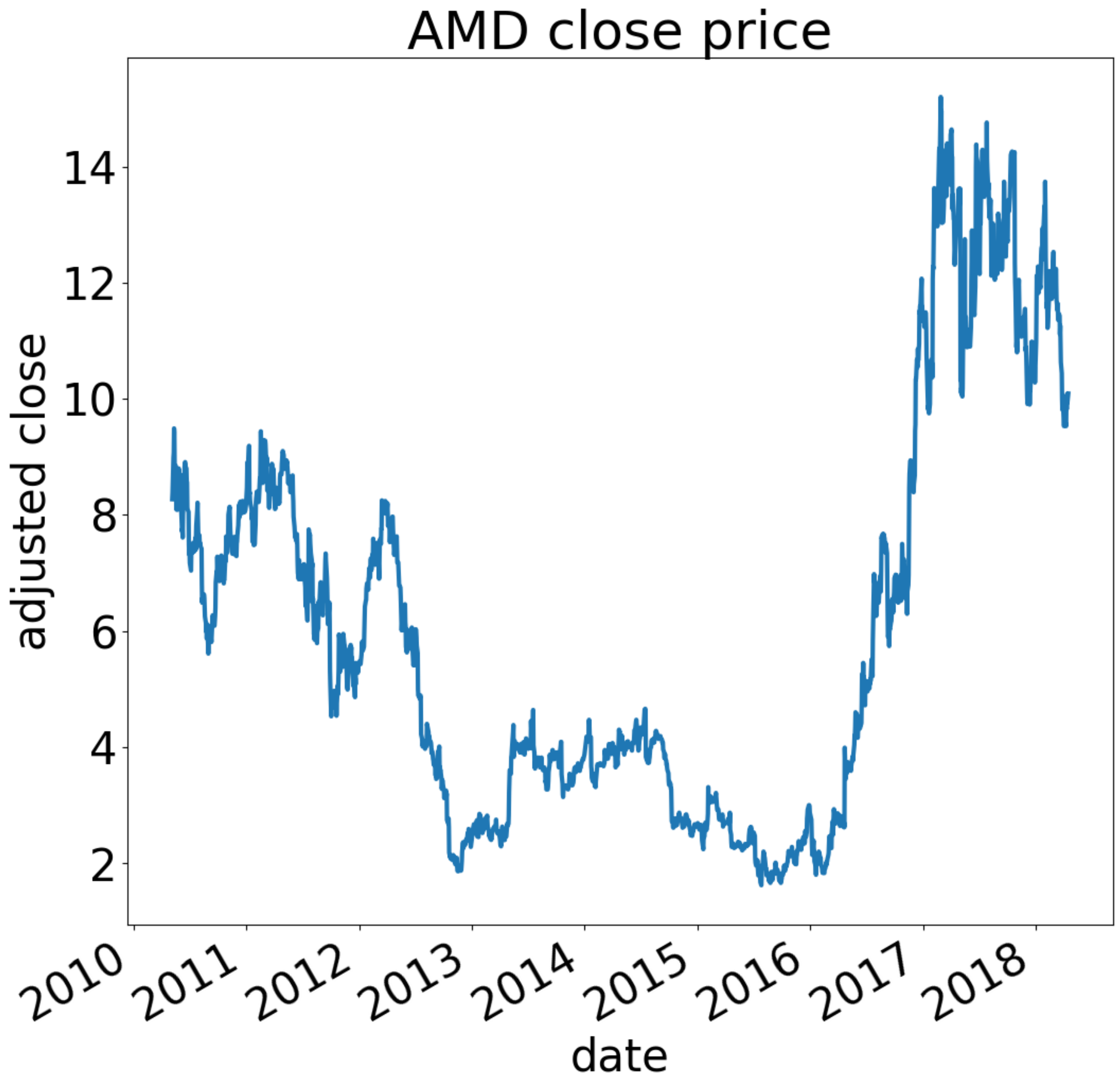
# Machine learning for finance

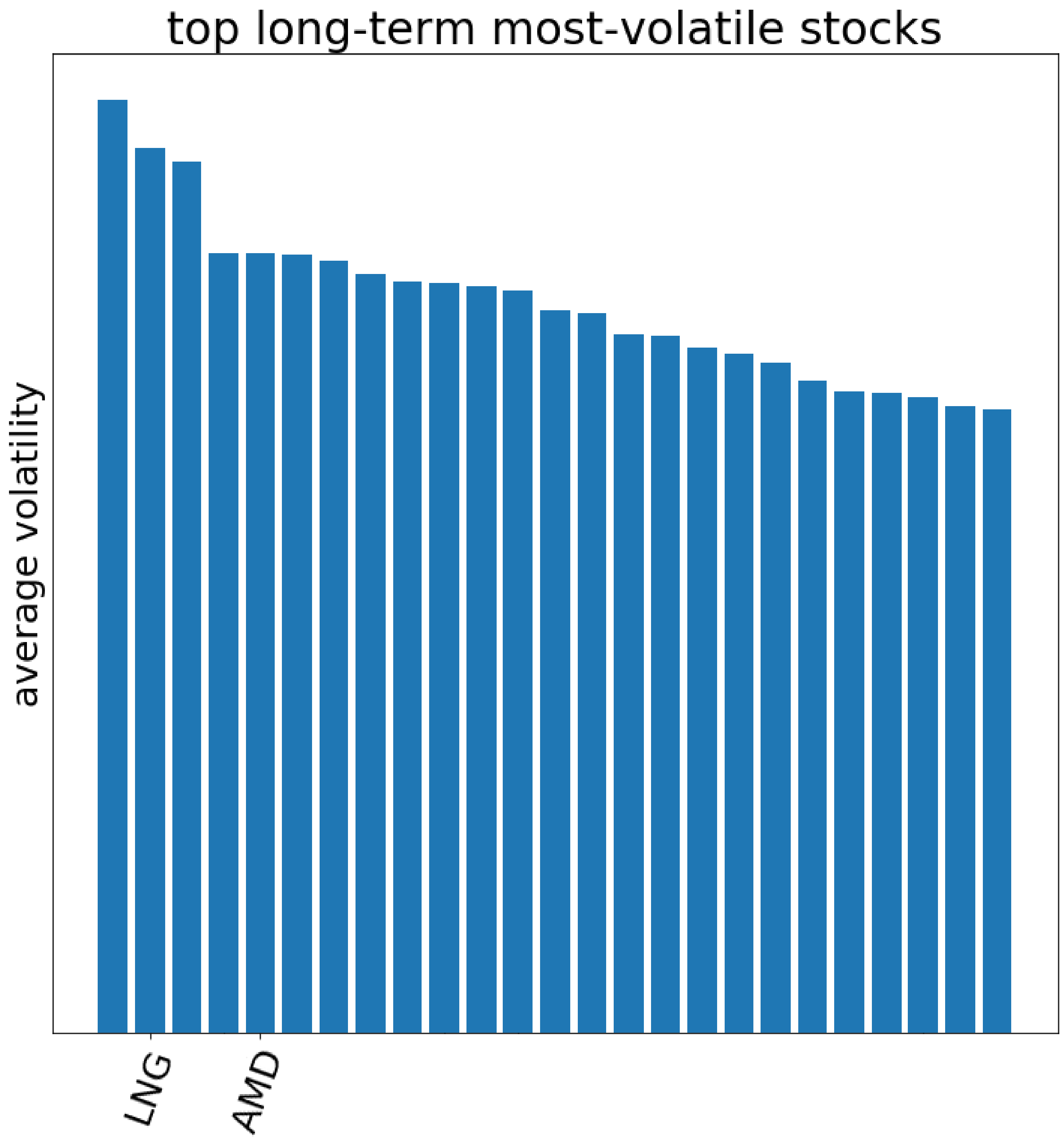
Nathan George
Data Science Professor

# Machine Learning in Finance



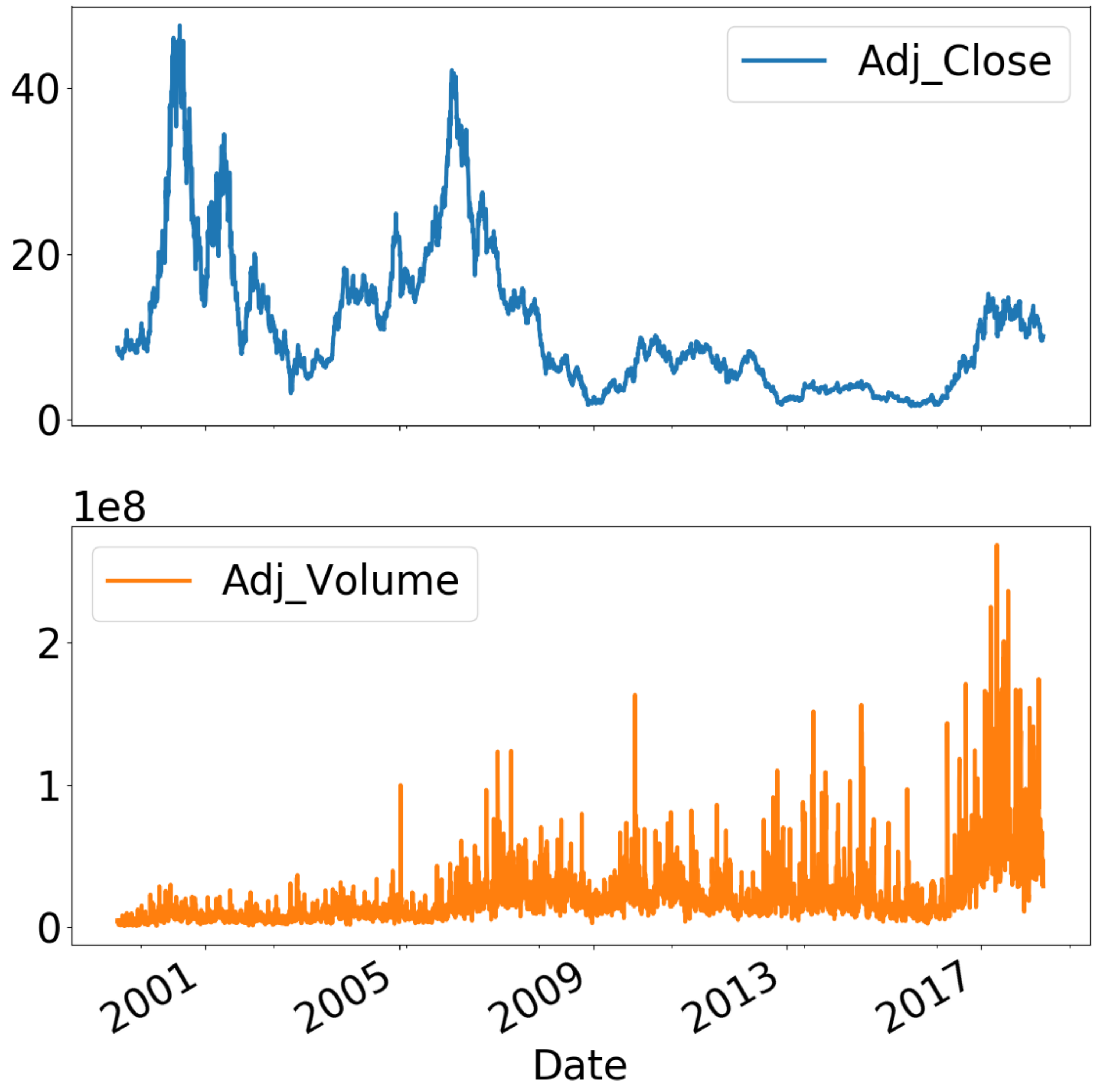

source: https://www.zacks.com/stock/quote/AMD
JPM report: http://valuesimplex.com/articles/JPM.pdf


AMD close price
adjusted close
14
12
10
8
6
4
2
2010
2011
2012
2013
2014
2015
2016
2017
2018
date


top long-term most-volatile stocks
average volatility
LNG
AMD

# Understanding the data

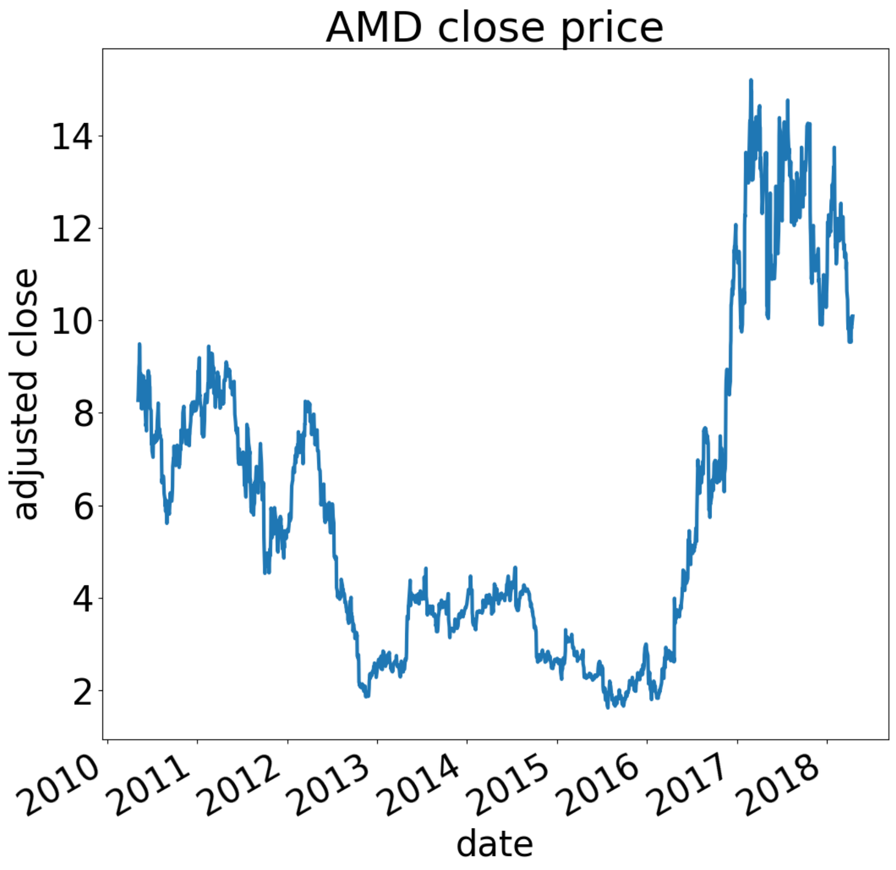
```
print(amd_df.head())

            Adj_Close  Adj_Volume
Date
1999-03-10      8.690   4871800.0
1999-03-11      8.500   3566600.0
1999-03-12      8.250   4126800.0
1999-03-15      8.155   3006400.0
1999-03-16      8.500   3511400.0
```

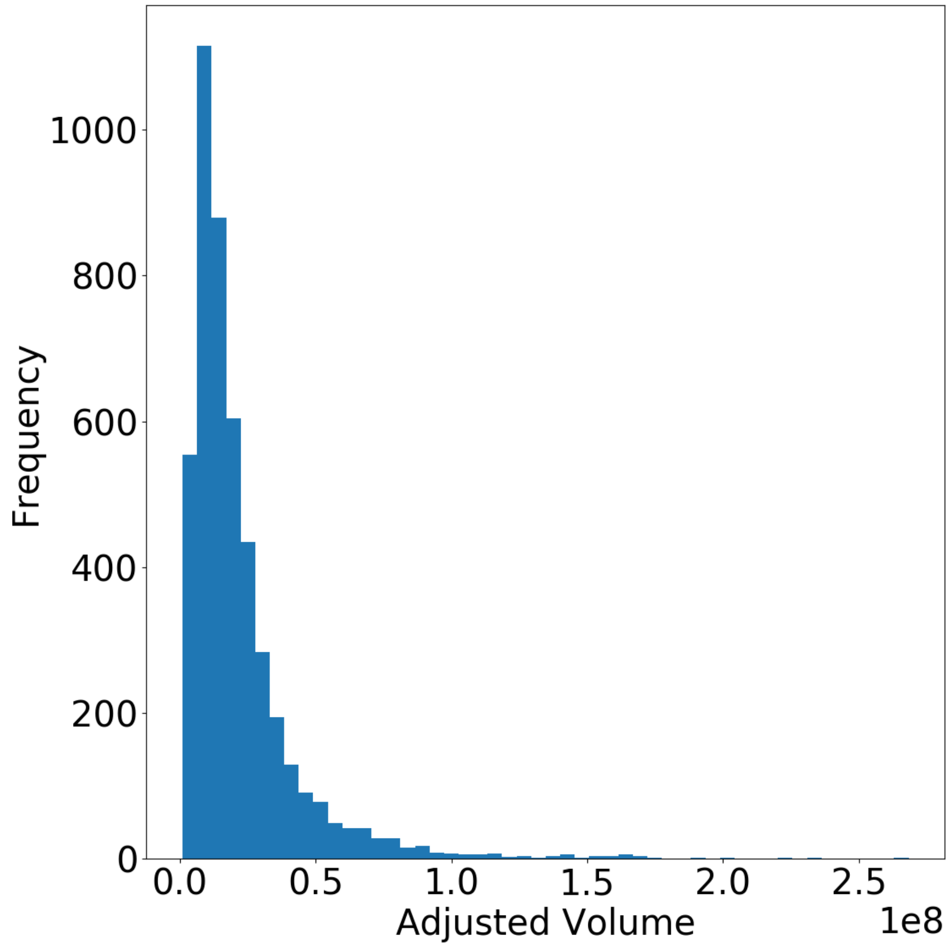

Adj_Close
40
20
0
1e8
Adj_Volume
2
1
0
2001
2005
2009
2013
2017
Date

# EDA plots

```
amd_df['Adj_Close'].plot()
plt.show()
```

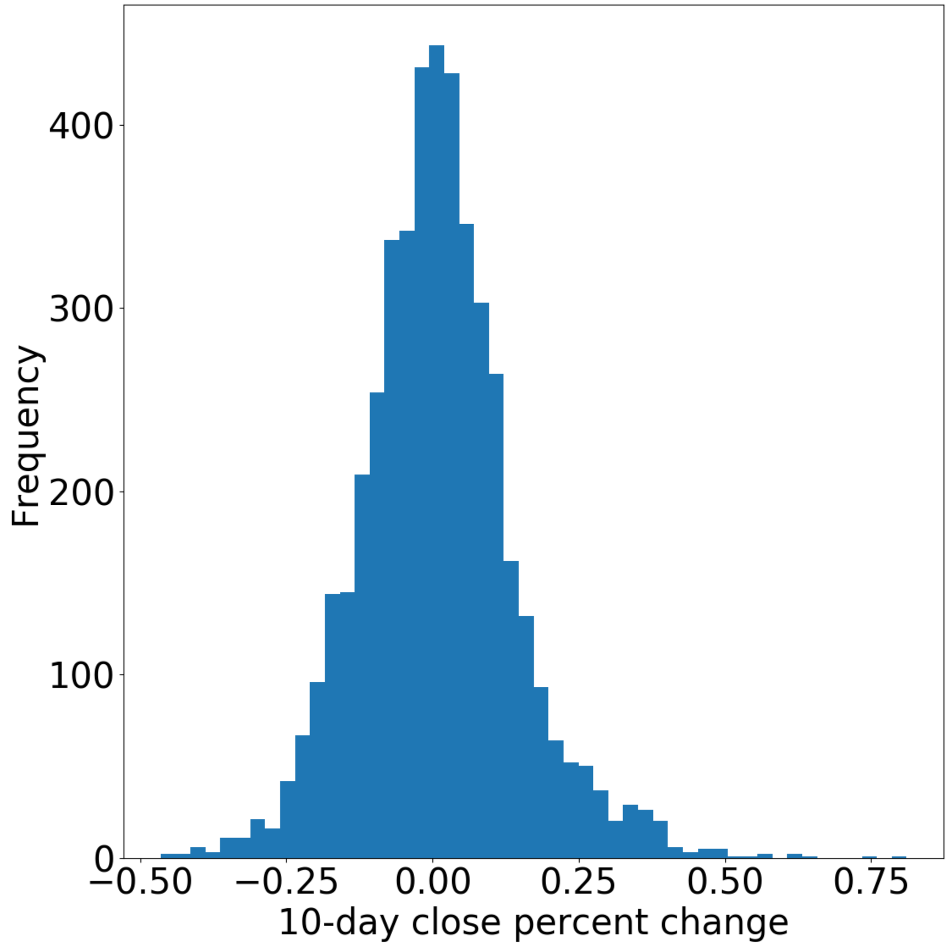
```
plt.clf()  # clears the plot area
vol = amd_df['Adj_Volume']
vol.plot.hist(bins=50)
plt.show()
```



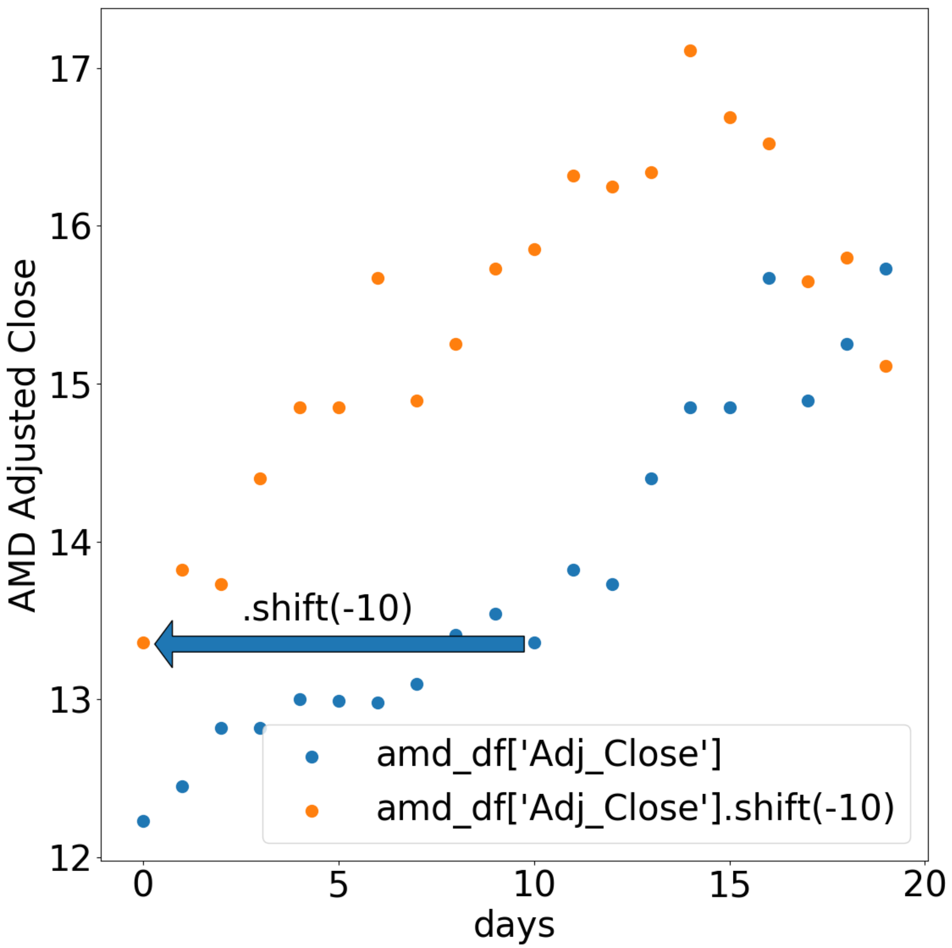
# Price changes

```
amd_df['10d_close_pct'] = amd_df['Adj_Close'].pct_change(10)

amd_df['10d_close_pct'].plot.hist(bins=50)
plt.show()
```



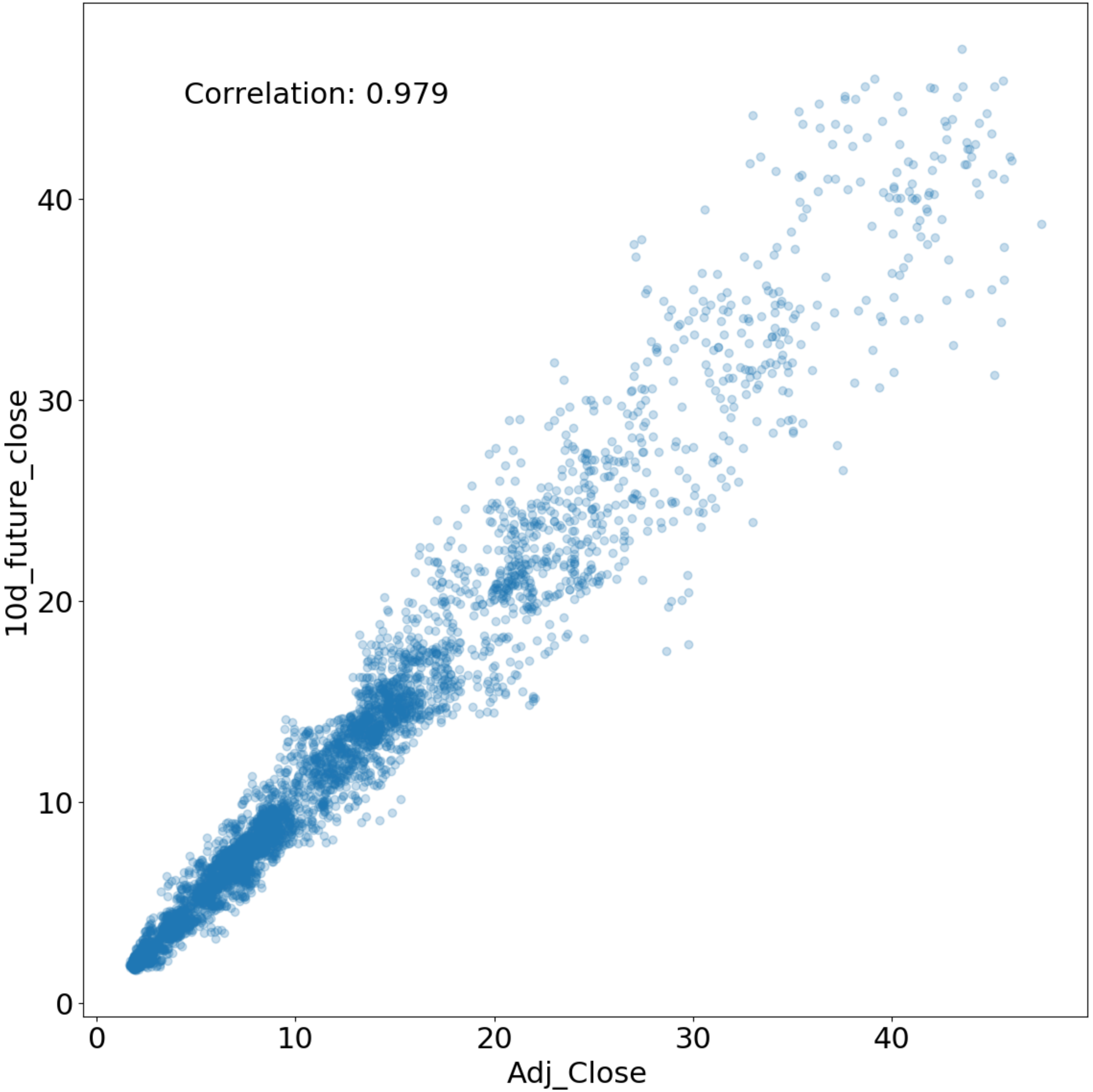
# Shift data

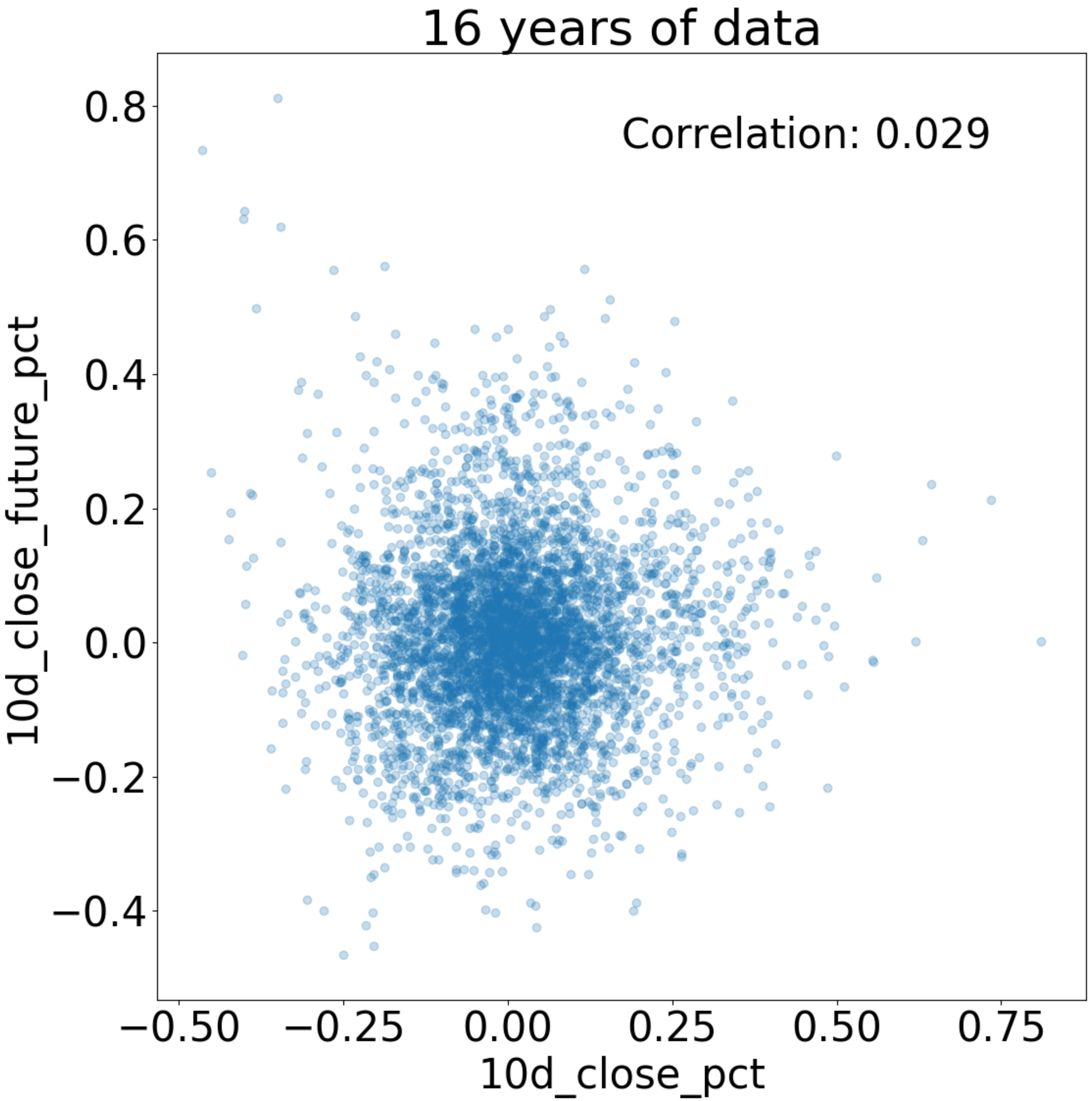
```
amd_df['10d_future_close'] = amd_df['Adj_Close'].shift(-10)
amd_df['10d_future_close_pct'] = amd_df['10d_future_close'].pct_change(10)
```

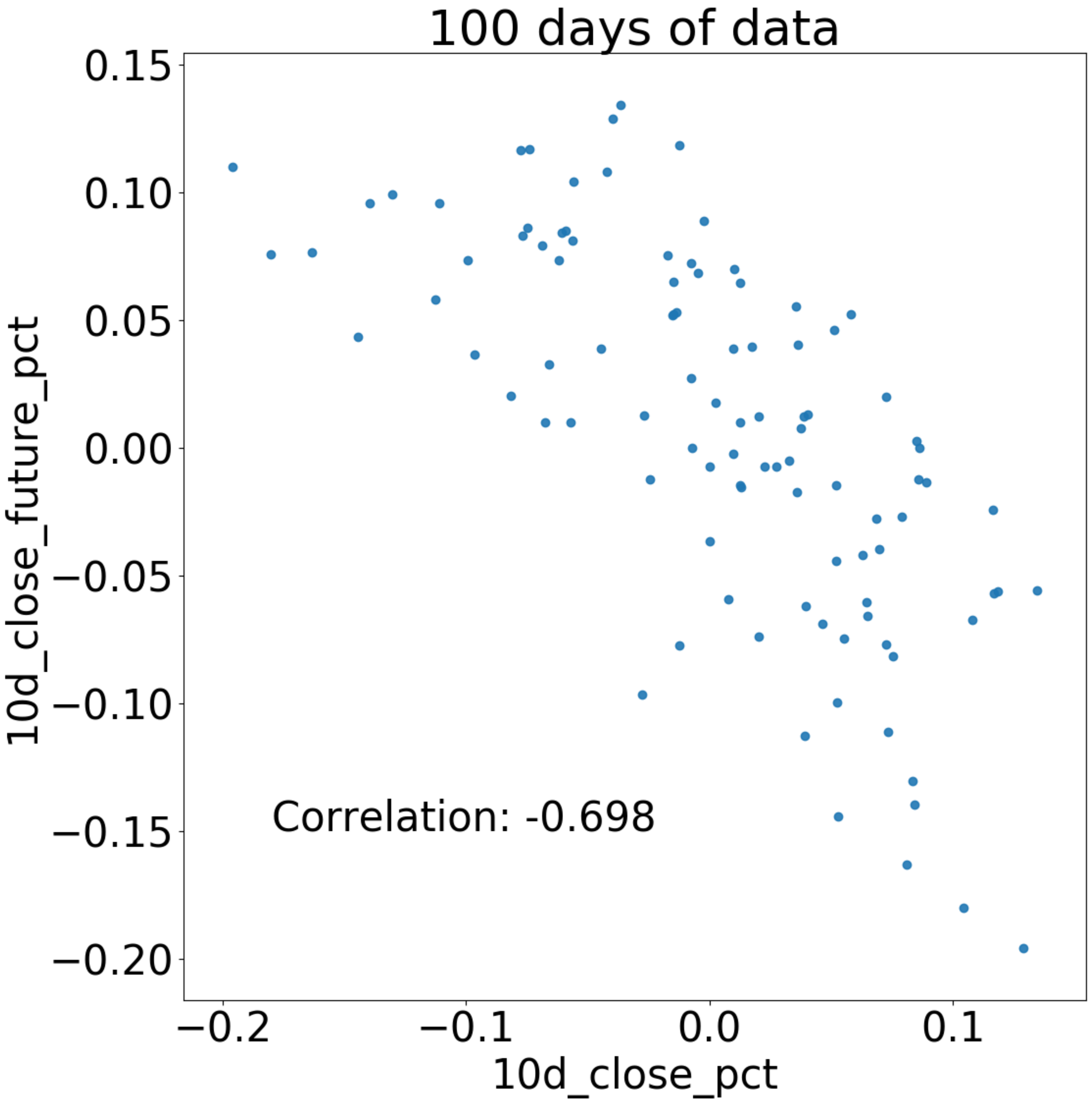


# Correlations

```
corr = amd_df.corr()
print(corr)

                    10d_future_close_pct  10d_future_close  10d_close_pct
10d_future_close_pct            1.000000          0.070742       0.030402
10d_future_close                0.070742          1.000000       0.082828
10d_close_pct                   0.030402          0.082828       1.000000
Adj_Close                      -0.083982          0.979345       0.073843
Adj_Volume                     -0.024456         -0.122473       0.044537

                    Adj_Close  Adj_Volume
10d_future_close_pct -0.083982   -0.024456
10d_future_close      0.979345   -0.122473
10d_close_pct         0.073843    0.044537
Adj_Close             1.000000   -0.119437
Adj_Volume           -0.119437    1.000000
```


Correlation: 0.979
10d_future_close
Adj_Close
0
10
20
30
40
0
10
20
30
40


16 years of data
Correlation: 0.029
10d_close_future_pct
10d_close_pct
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.50
−0.25
0.00
0.25
0.50
0.75


100 days of data
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
10d_close_future_pct
Correlation: -0.698
−0.2
−0.1
0.0
0.1
10d_close_pct

MACHINE LEARNING FOR FINANCE IN PYTHON

# Let's do some EDA!

MACHINE LEARNING FOR FINANCE IN PYTHON

# Data transforms, features, and targets

Nathan George
Data Science Professor

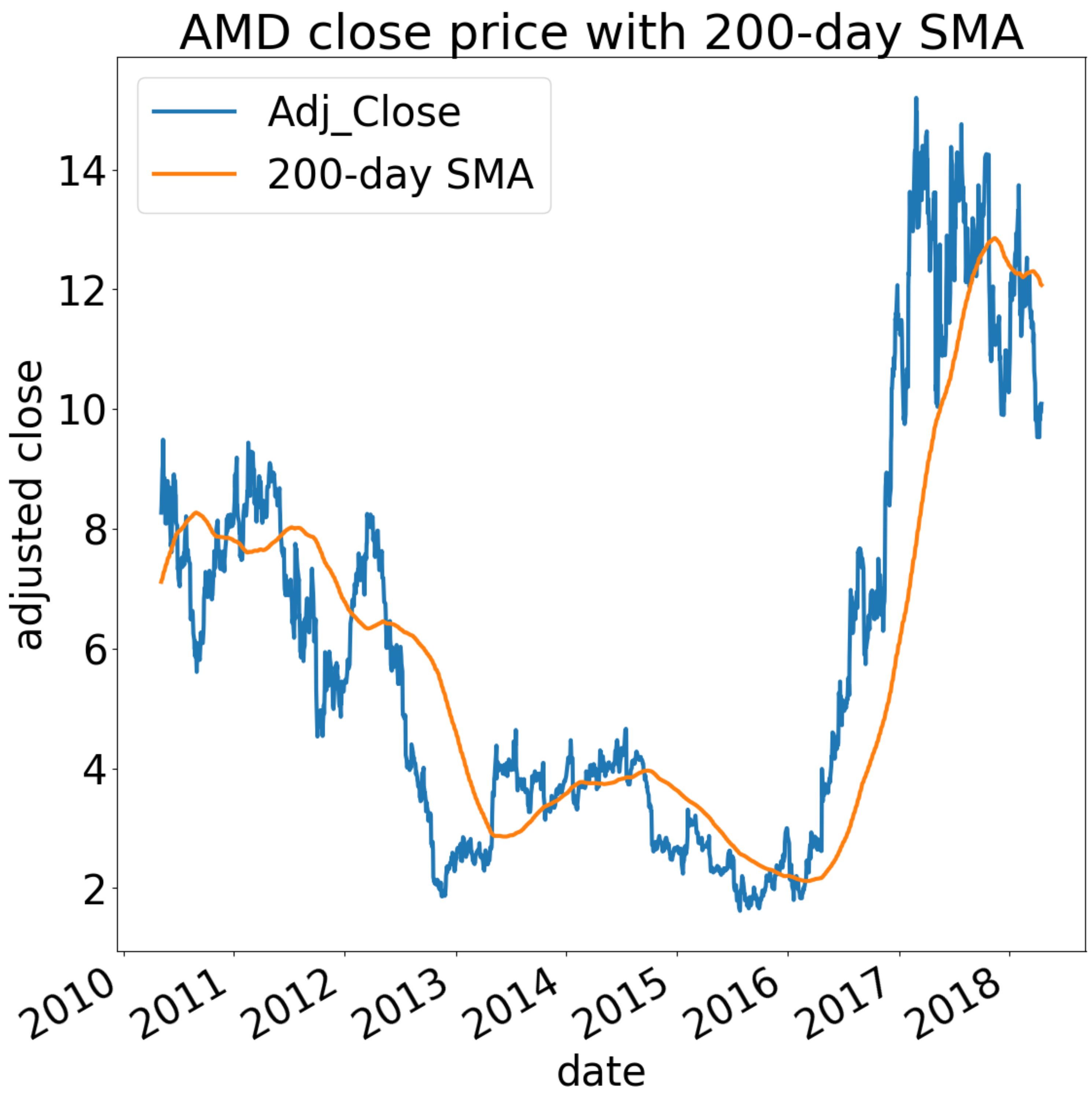
# Making features and targets

```
features = amd_df[['10d_close_pct', 'Adj_Volume']]
targets = amd_df['10d_future_close_pct']
print(type(features))
```

```
pandas.core.series.DataFrame
```

```
print(type(targets))
```

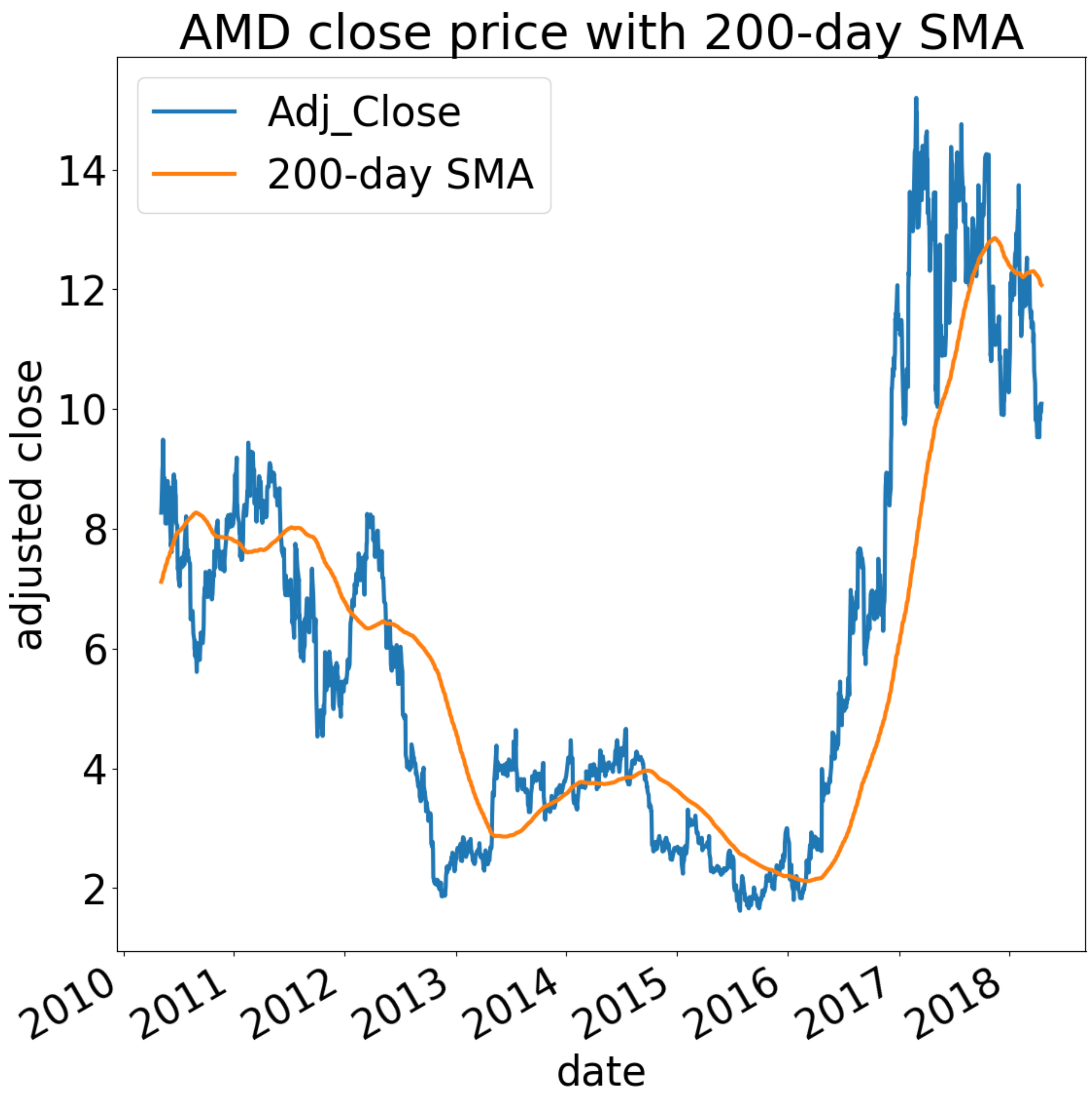
```
pandas.core.series.Series
```


AMD close price with 200-day SMA
Adj_Close
200-day SMA
adjusted close
2
4
6
8
10
12
14
date
2010
2011
2012
2013
2014
2015
2016
2017
2018

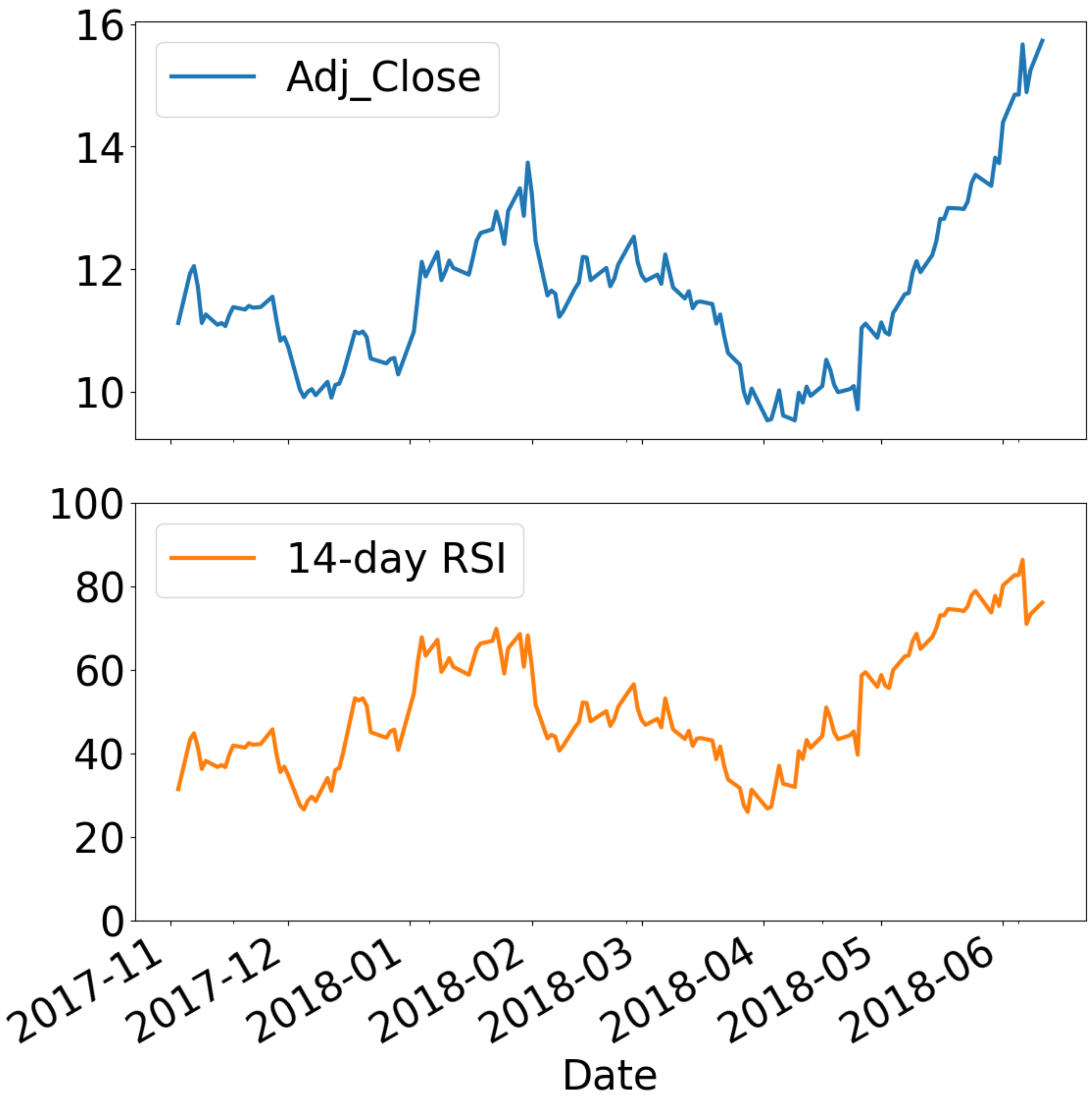
# Moving averages

Moving averages:

- use $n$ past days to get average
- common values for $n$: 14, 50, 200



AMD close price with 200-day SMA
Adj_Close
200-day SMA
adjusted close
date
2
4
6
8
10
12
14
2010
2011
2012
2013
2014
2015
2016
2017
2018


16
14
12
10
Adj_Close
100
80
60
40
20
0
14-day RSI
2017-11
2017-12
2018-01
2018-02
2018-03
2018-04
2018-05
2018-06
Date

# RSI equation

$$RSI = 100 - \frac{100}{1 + RS}$$

$$RS = \frac{\text{Average gain over } n \text{ periods}}{\text{Average loss over } n \text{ periods}}$$

# Calculating SMA and RSI

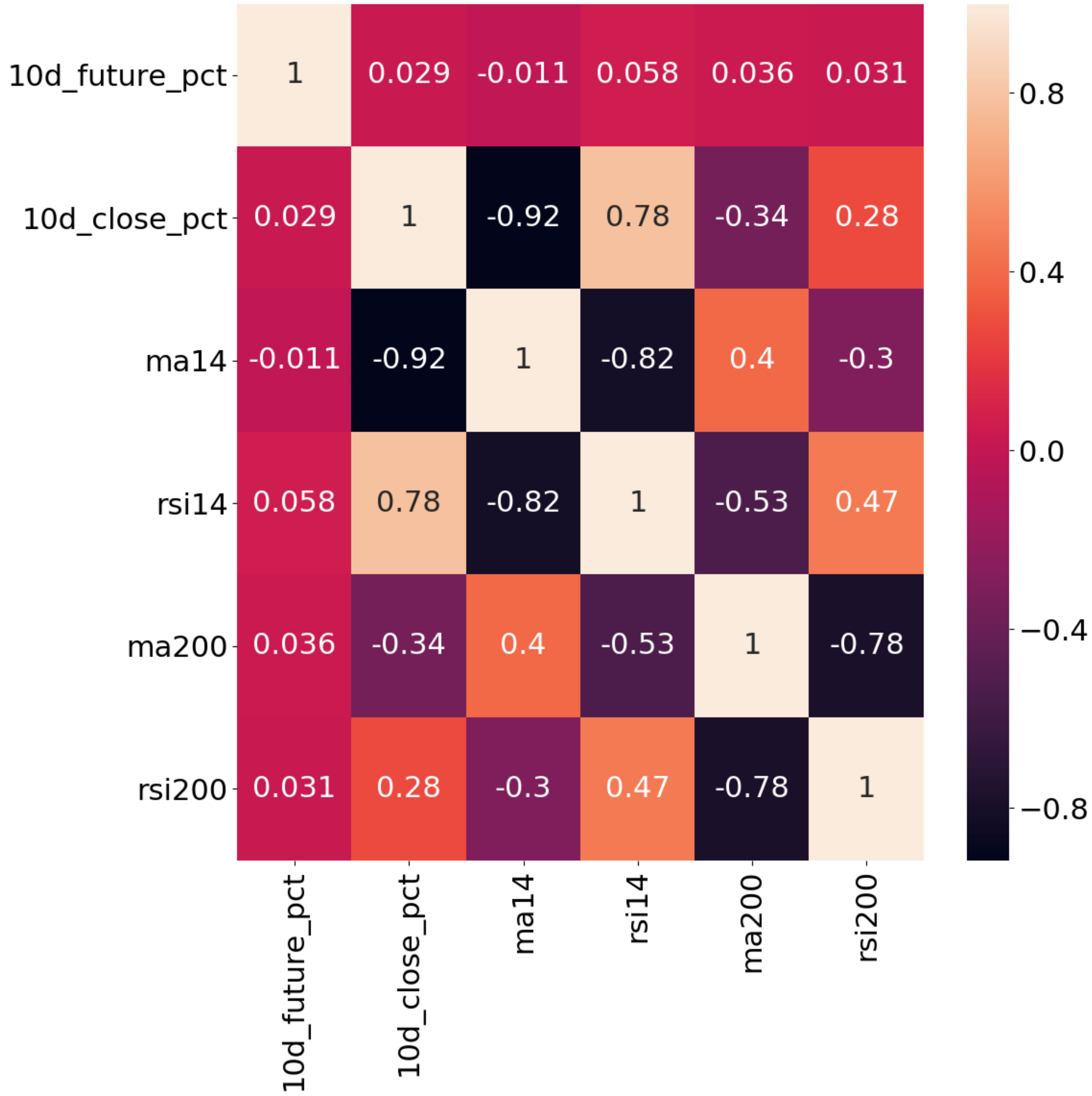
```
import talib

amd_df['ma200'] = talib.SMA(amd_df['Adj_Close'].values, timeperiod=200)

amd_df['rsi200'] = talib.RSI(amd_df['Adj_Close'].values, timeperiod=200)
```

# Finally, our features

```
feature_names = ['10d_close_pct', 'ma200', 'rsi200']
features = amd_df[feature_names]
targets = amd_df['10d_future_close_pct']

feature_target_df = amd_df[feature_names + '10d_future_close_pct']
```

# Check correlations

```
import seaborn as sns

corr = feature_target_df.corr()
sns.heatmap(corr, annot=True)
```


10d_future_pct
10d_close_pct
ma14
rsi14
ma200
rsi200
1 0.029 -0.011 0.058 0.036 0.031
0.029 1 -0.92 0.78 -0.34 0.28
-0.011 -0.92 1 -0.82 0.4 -0.3
0.058 0.78 -0.82 1 -0.53 0.47
0.036 -0.34 0.4 -0.53 1 -0.78
0.031 0.28 -0.3 0.47 -0.78 1
0.8
0.4
0.0
−0.4
−0.8

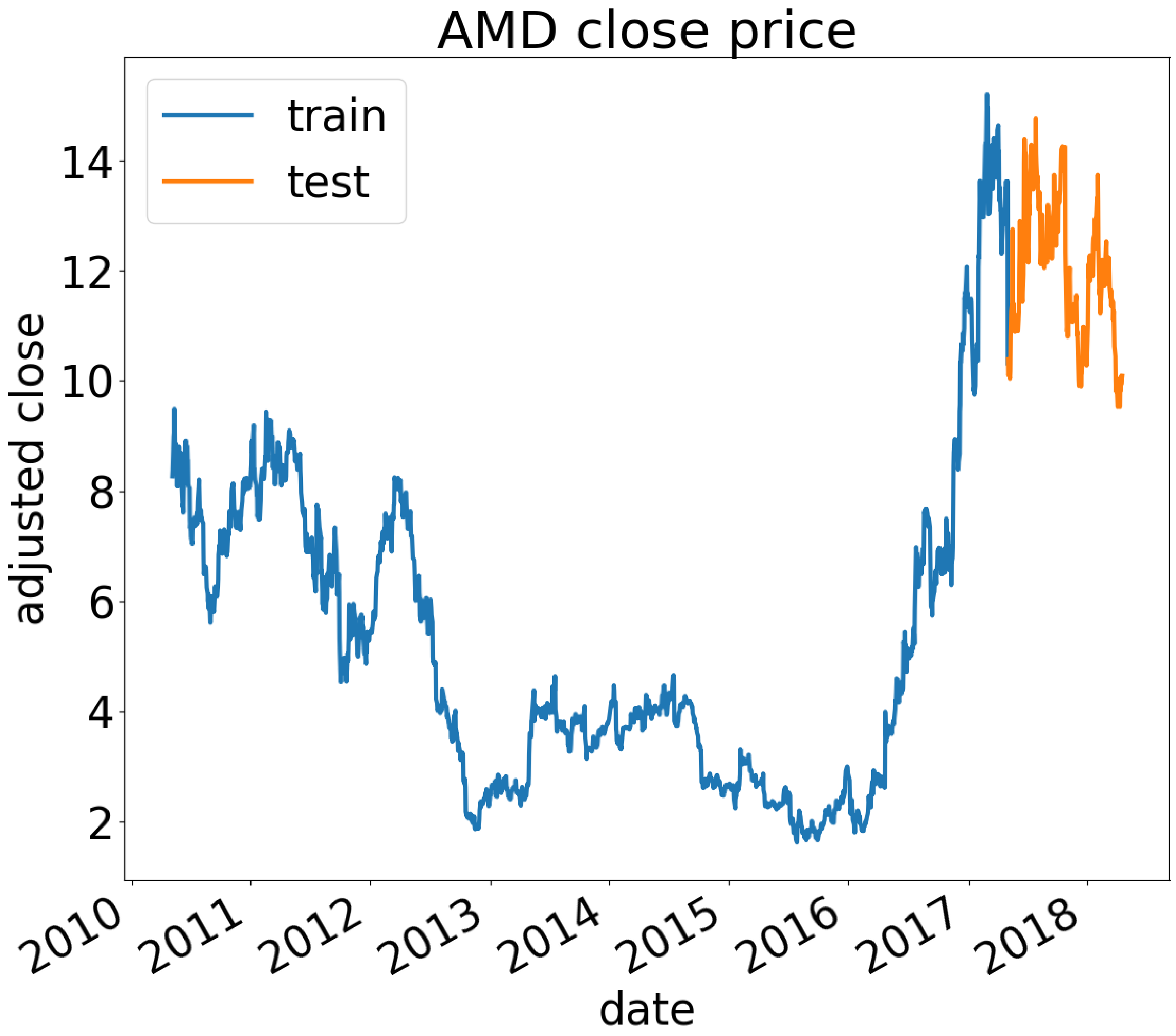
MACHINE LEARNING FOR FINANCE IN PYTHON

# Let's create features and targets!

MACHINE LEARNING FOR FINANCE IN PYTHON

# Linear modeling with financial data

Nathan George
Data Science Professor


AMD close price
train
test
adjusted close
14
12
10
8
6
4
2
2010
2011
2012
2013
2014
2015
2016
2017
2018
date

# Make train and test sets

```
import statsmodels.api as sm

linear_features = sm.add_constant(features)

train_size = int(0.85 * targets.shape[0])

train_features = linear_features[:train_size]
train_targets = targets[:train_size]
test_features = linear_features[train_size:]
test_targets = targets[train_size:]
```

```
some_list[start:stop:step]
```

# Linear modeling

```
model = sm.OLS(train_targets, train_features)
results = model.fit()
```

# Linear modeling

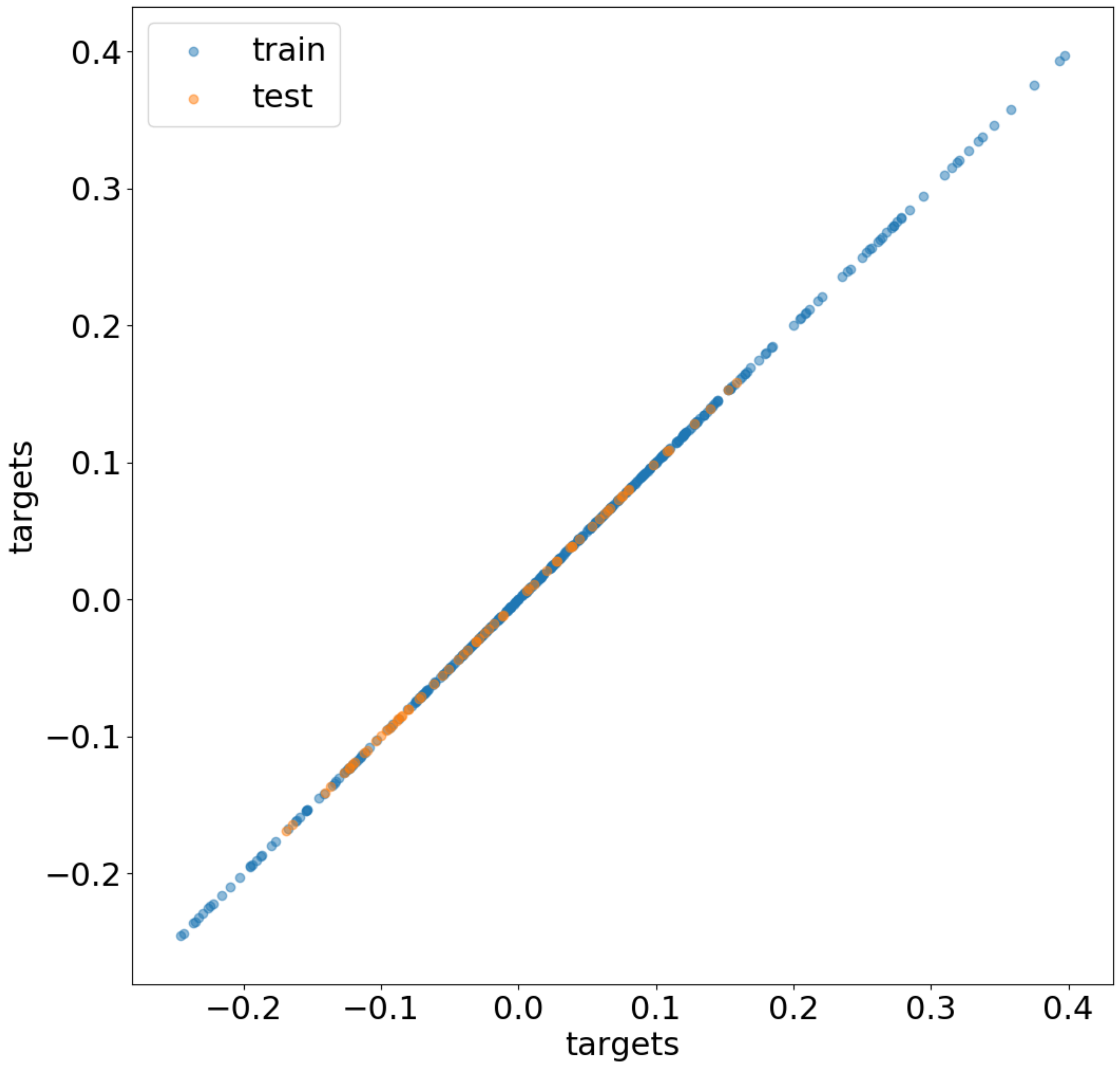
```
print(results.summary())
```

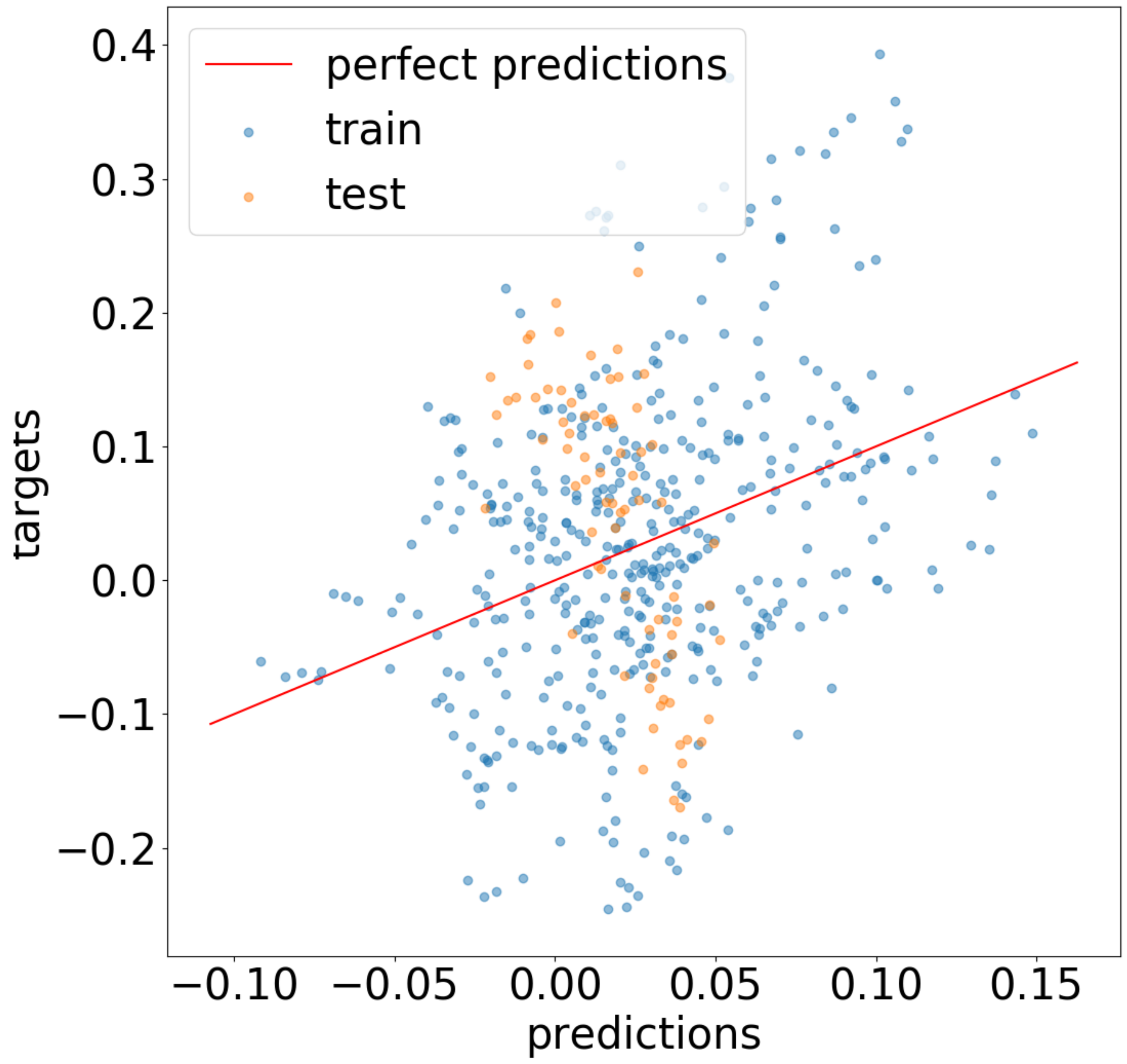
# Linear modeling

# p-values

```
print(results.pvalues)
```

```
const           4.630428e-05
10d_close_pct   3.546748e-01
ma14            1.136941e-01
rsi14           2.968699e-01
ma200           9.126405e-14
rsi200          1.169324e-10
```


train
test
targets
targets
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4


perfect predictions
train
test
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
targets
−0.10
−0.05
0.00
0.05
0.10
0.15
predictions

MACHINE LEARNING FOR FINANCE IN PYTHON

# Time to fit a linear model!