

TDM	729.89	915.51	185.62	▲25.43%	FLR	660.27	745.28	85.01	▲12.88%
HUM	749.73	924.29	174.56	▲23.28%	UVD	155.59	181.57	25.98	▲16.70%
DMW	833.72	1004.01	170.29	▲20.43%	QUV	440.55	540.21	99.66	▲22.62%
YZJ	903.49	1127.46	223.97	▲24.79%	HZT	285.51	344.98	59.47	▲20.83%
SLV	993.07	1216.99	223.92	▲24.17%	PCW	811.44	1029.66	218.22	▲26.89%
VDA	113.74	143.41	29.67	▲26.09%	AIK	361.77	451.39	89.62	▲24.77%
UVV	468.08	535.41	67.33	▲14.38%	ZJJ	858.36	994.57	136.21	▲15.87%
HJS	545.49	659.05	113.56	▲20.82%	RHJ	894.79	1046.68	151.89	▲16.97%
ECC	566.96	664.69	97.73	▲17.24%	VGV	425.08	509.95	84.87	▲19.97%

PJ	912.63	1038.36	125.73	▲13.78%	ZGK	391.59	491.48	99.89	▲25.51%
UAQ	1309.55	1655.62	346.07	▲26.43%	BNY	969.21	1130.65	161.44	▲16.66%
U	1017.17	1641.66	346.49	▲26.75%	SDM	735.44	913.39	177.95	▲24.20%
U	651.53	775.84	121.51	▲18.57%	TQQ	1323.91	1646.42	322.51	▲24.36%
U	1017.17	1641.66	346.49	▲26.75%	QIS	543.42	667.24	123.82	▲22.79%
U	1017.17	1641.66	346.49	▲26.75%	U	1017.17	1641.66	346.49	▲26.75%

ALGORITHMIC TRADING IN THE STOCK MARKET

Aidan Hammond

Professor Prasad

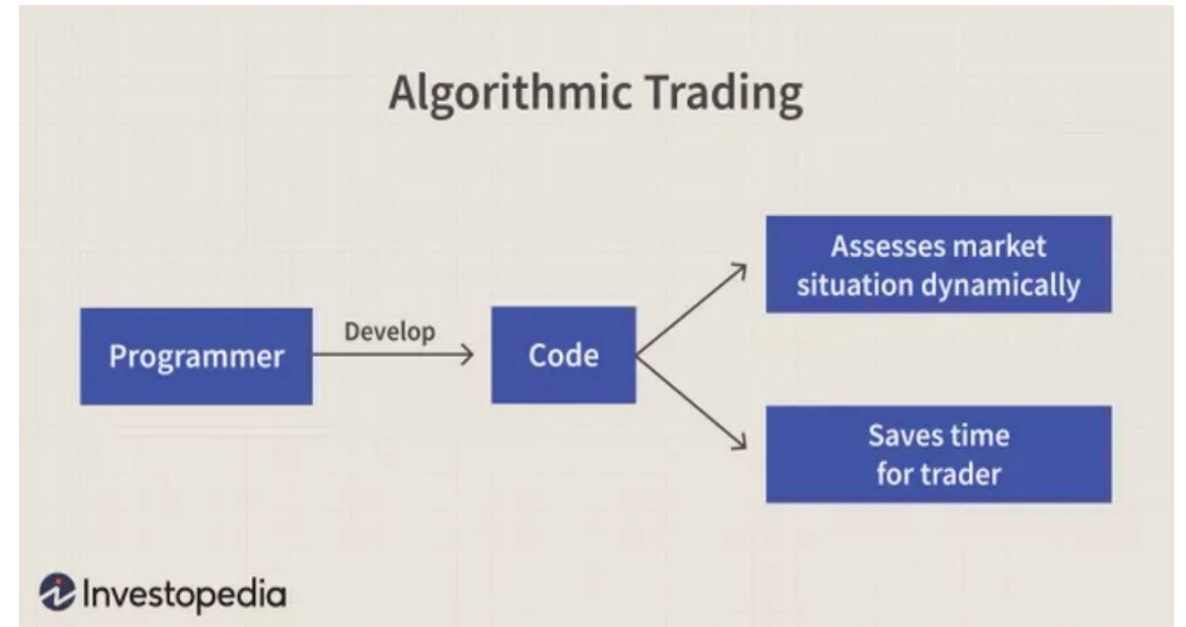
CS106

5/3/22



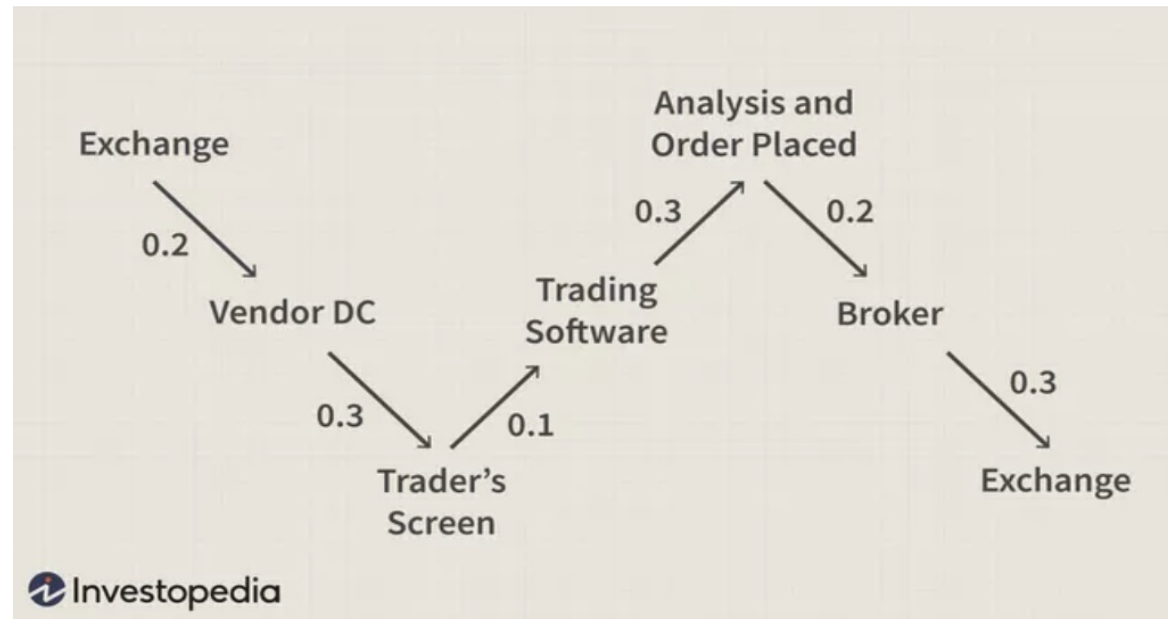
What is Algorithmic Trading?

- ❖ The use of pre-programmed code to execute automated asset trading
- ❖ Unmatched speed and volume compared to human capabilities
- ❖ Algo-trading coming to dominate the financial markets



Processes of Implementation

- ❖ Latency is critical
- ❖ Most commonly developed in Python and R



Visualizing Algo-trading

- ❖ Example of algo-trading script using a for-loop:
 - ❖ Establish [stocks] list
 - ❖ Loop through [stocks]
 - ❖ Initiate your buy signal and sell signal as variables
 - ❖ Using conditions, execute trades whenever one or more of the pre-programmed conditions are true

```
import quantopian.algorithm as algo
from quantopian.pipeline import Pipeline
from quantopian.pipeline.data.builtin import USEquityPricing
from quantopian.pipeline.filters import QTradableStocksUS
import talib as ta

def trading(context, data):
    context.stocks = [sid(39840), sid(2190), sid(24), sid(8554)]

    for stock in context.stocks:
        # Closing prices for each stock
        close = data.history(stock, "close", 400, "1d")

        # Moving Averages for each stock
        MA_50 = ta.MA(close, timeperiod=50, matype=0)
        MA_200 = ta.MA(close, timeperiod=200, matype=0)

        # Crossing Moving Averages
        bullish_cross = MA_50[-1]>MA_200[-1] and MA_50[-2]<MA_200[-2]
        bearish_cross = MA_50[-1]<MA_200[-1] and MA_50[-2]>MA_200[-2]

        # Checking leverage
        open_lev = len(get_open_orders()) < (len(context.stocks) - len(context.portfolio.positions))

        # Opening a position in each stock
        if open_lev:
            if bullish_cross:
                order_target_percent(stock, 0.25)
                log.info("Opening long position {}".format(stock))
            elif bearish_cross:
                order_target_percent(stock, -0.25)
                log.info("Opening short position {}".format(stock))
```

Source: <https://medium.com/swlh/coding-your-way-to-wall-street-bf21a500376f>

Work Cited

- ❖ <https://www.investopedia.com/terms/a/algorithmictrading.asp>
 - ❖ <https://www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp>
 - ❖ <https://www.datacamp.com/community/tutorials/finance-python-trading#a-simple-trading-strategy>
 - ❖ <https://www.nasdaq.com/articles/advantages-algorithmic-trading-2019-06-07>
 - ❖ <https://medium.com/swlh/coding-your-way-to-wall-street-bf21a500376f>
-