Algorithmic Trading; Risk Management; Trading Bot and Event-Based Backtesting

1121AIFQA08
MBA, IM, NTPU (M5276) (Fall 2023)
Tue 2, 3, 4 (9:10-12:00) (B3F17)

https://meet.google.com/paj-zhhj-mya

Min-Yuh Day, Ph.D,
Associate Professor

Institute of Information Management, National Taipei University

https://web.ntpu.edu.tw/~myday

2023-12-19

Syllabus

Week	Date	Subject/Topics
1	2023/09/12	Introduction to Artificial Intelligence in Finance and Quantitative Analysis
2	2023/09/19	AI in FinTech: Metaverse, Web3, DeFi, NFT, Financial Services Innovation and Applications
3	2023/09/26	Investing Psychology and Behavioral Finance
4	2023/10/03	Event Studies in Finance
5	2023/10/10	National Day (Day off)
6	2023/10/17	Case Study on AI in Finance and Quantitative Analysis I

Syllabus

Week	Date	Subject/Topics
7	2023/10/24	Finance Theory and Data-Driven Finance
8	2023/10/31	Midterm Project Report
9	2023/11/07	Financial Econometrics
10	2023/11/14	AI-First Finance
11	2023/11/21	Deep Learning in Finance; Reinforcement Learning in Finance
12	2023/11/28	Case Study on AI in Finance and Quantitative Analysis II

Syllabus

Week	Date	Subject/Topics
13	2023/12/05	Industry Practices of AI in Finance and Quantitative Analysis
14	2023/12/12	Self-study
15	2023/12/19	Algorithmic Trading; Risk Management; Trading Bot and Event-Based Backtesting
16	2023/12/26	Final Project Report I
17	2024/01/02	Final Project Report II
18	2024/01/09	Self-study

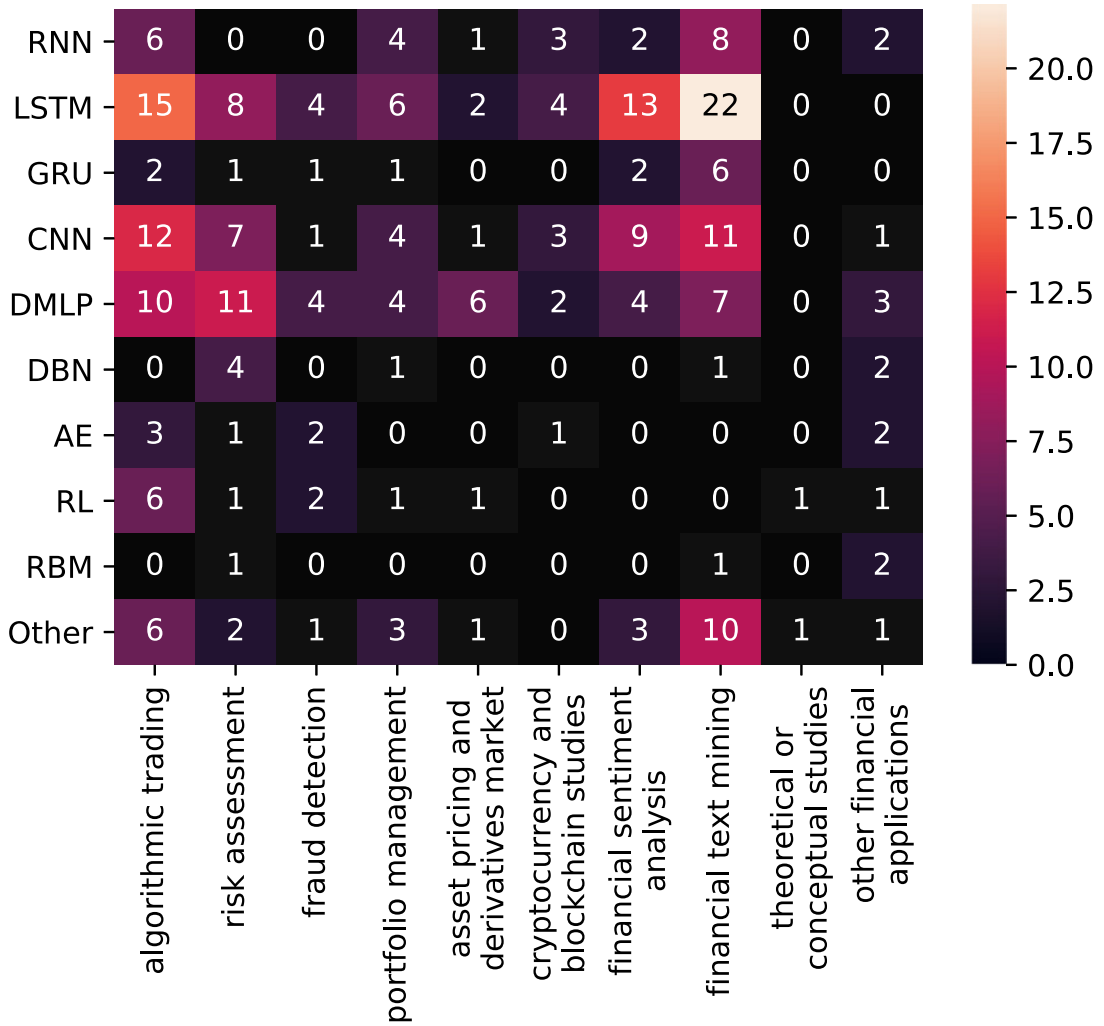
Algorithmic Trading
Risk Management
Trading Bot
Event-Based Backtesting

Outline

- Algorithmic Trading
- Risk Management
- Trading Bot
- Event-Based Backtesting

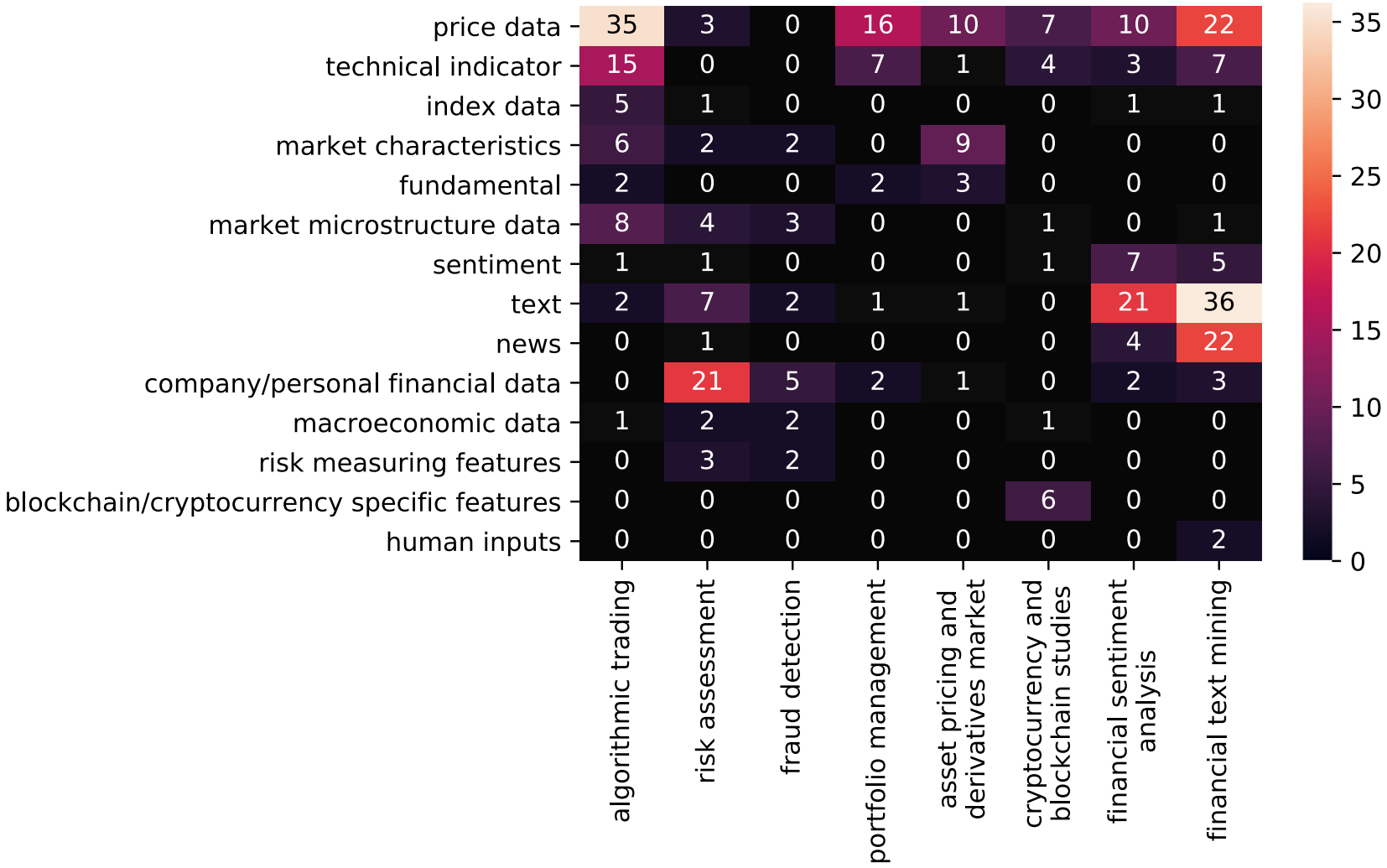
Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

Deep learning for financial applications: Topic-Model Heatmap



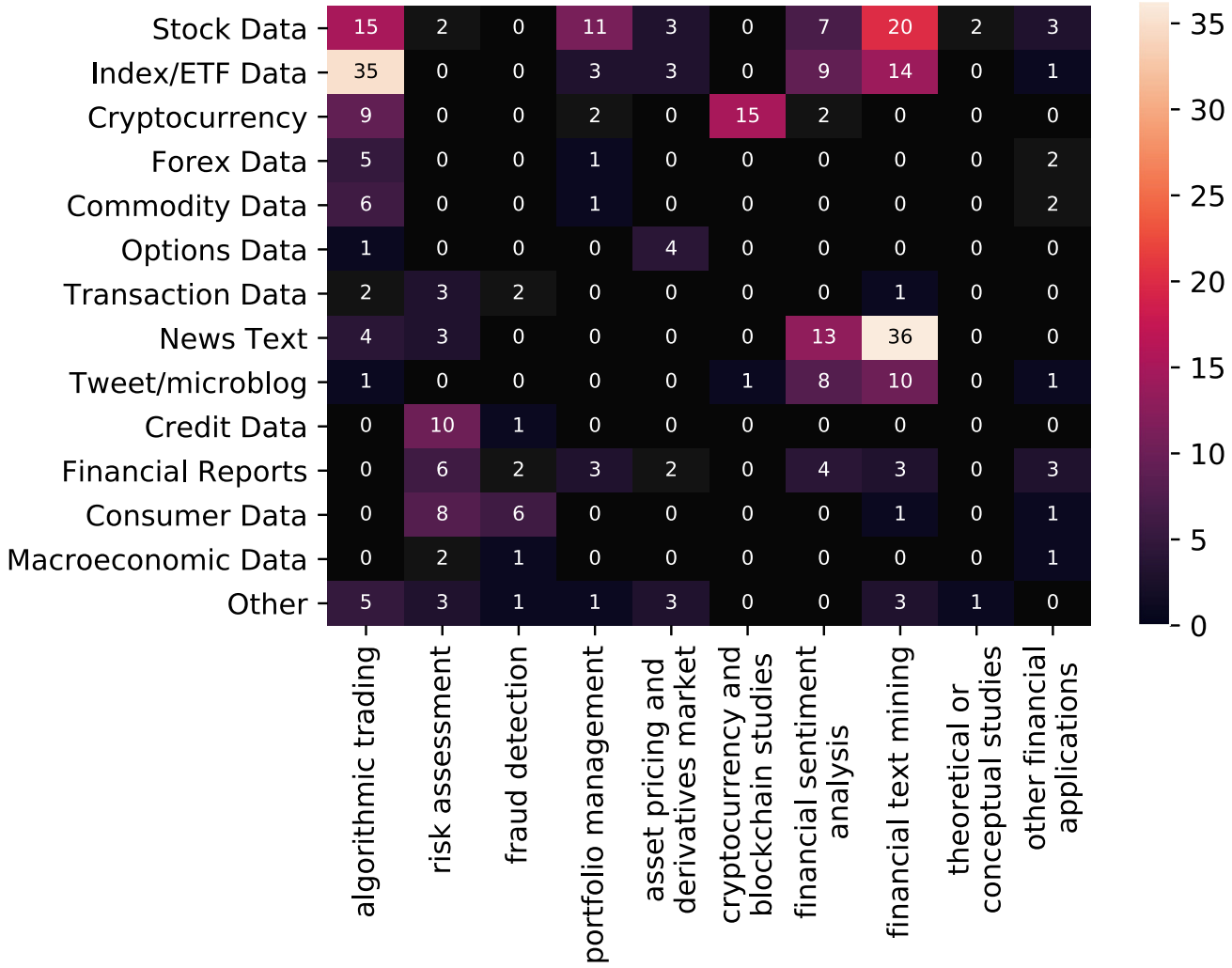
Source: Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer (2020). "Deep learning for financial applications: A survey." Applied Soft Computing (2020): 106384.

Deep learning for financial applications: Topic-Feature Heatmap



Source: Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer (2020). "Deep learning for financial applications: A survey." Applied Soft Computing (2020): 106384.

Deep learning for financial applications: Topic-Dataset Heatmap



Source: Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer (2020). "Deep learning for financial applications: A survey." Applied Soft Computing (2020): 106384.

Deep learning for financial applications:

Algo-trading applications embedded with time series forecasting models

Art.	Data set	Period	Feature set	Method	Performance criteria	Environment
[33]	GarantiBank in BIST, Turkey	2016	OCHLV, Spread, Volatility, Turnover, etc.	PLR, Graves LSTM	MSE, RMSE, MAE, RSE, Correlation R-square	Spark
[34]	CSI300, Nifty50, HSI, Nikkei 225, S&P500, DJIA	2010–2016	OCHLV, Technical Indicators	WT, Stacked autoencoders, LSTM	MAPE, Correlation coefficient, THEIL-U	–
[35]	Chinese Stocks	2007–2017	OCHLV	CNN + LSTM	Annualized Return, Mxm Retracement	Python
[36]	50 stocks from NYSE	2007–2016	Price data	SFM	MSE	–
[37]	The LOB of 5 stocks of Finnish Stock Market	2010	FI-2010 dataset: bid/ask and volume	WMTR, MDA	Accuracy, Precision, Recall, F1-Score	–
[38]	300 stocks from SZSE, Commodity	2014–2015	Price data	FDDR, DMLP+RL	Profit, return, SR, profit-loss curves	Keras
[39]	S&P500 Index	1989–2005	Price data, Volume	LSTM	Return, STD, SR, Accuracy	Python, TensorFlow, Keras, R, H2O
[40]	Stock of National Bank of Greece (ETE).	2009–2014	FTSE100, DJIA, GDAX, NIKKEI225, EUR/USD, Gold	GASVR, LSTM	Return, volatility, SR, Accuracy	Tensorflow
[41]	Chinese stock-IF-IH-IC contract	2016–2017	Decisions for price change	MODRL+LSTM	Profit and loss, SR	–
[42]	Singapore Stock Market Index	2010–2017	OCHL of last 10 days of Index	DMLP	RMSE, MAPE, Profit, SR	–
[43]	GBP/USD	2017	Price data	Reinforcement Learning + LSTM + NES	SR, downside deviation ratio, total profit	Python, Keras, Tensorflow
[44]	Commodity, FX future, ETF	1991–2014	Price Data	DMLP	SR, capability ratio, return	C++, Python
[45]	USD/GBP, S&P500, FTSE100, oil, gold	2016	Price data	AE + CNN	SR, % volatility, avg return/trans, rate of return	H2O

Source: Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer (2020). "Deep learning for financial applications: A survey." Applied Soft Computing (2020): 106384.

Deep learning for financial applications:

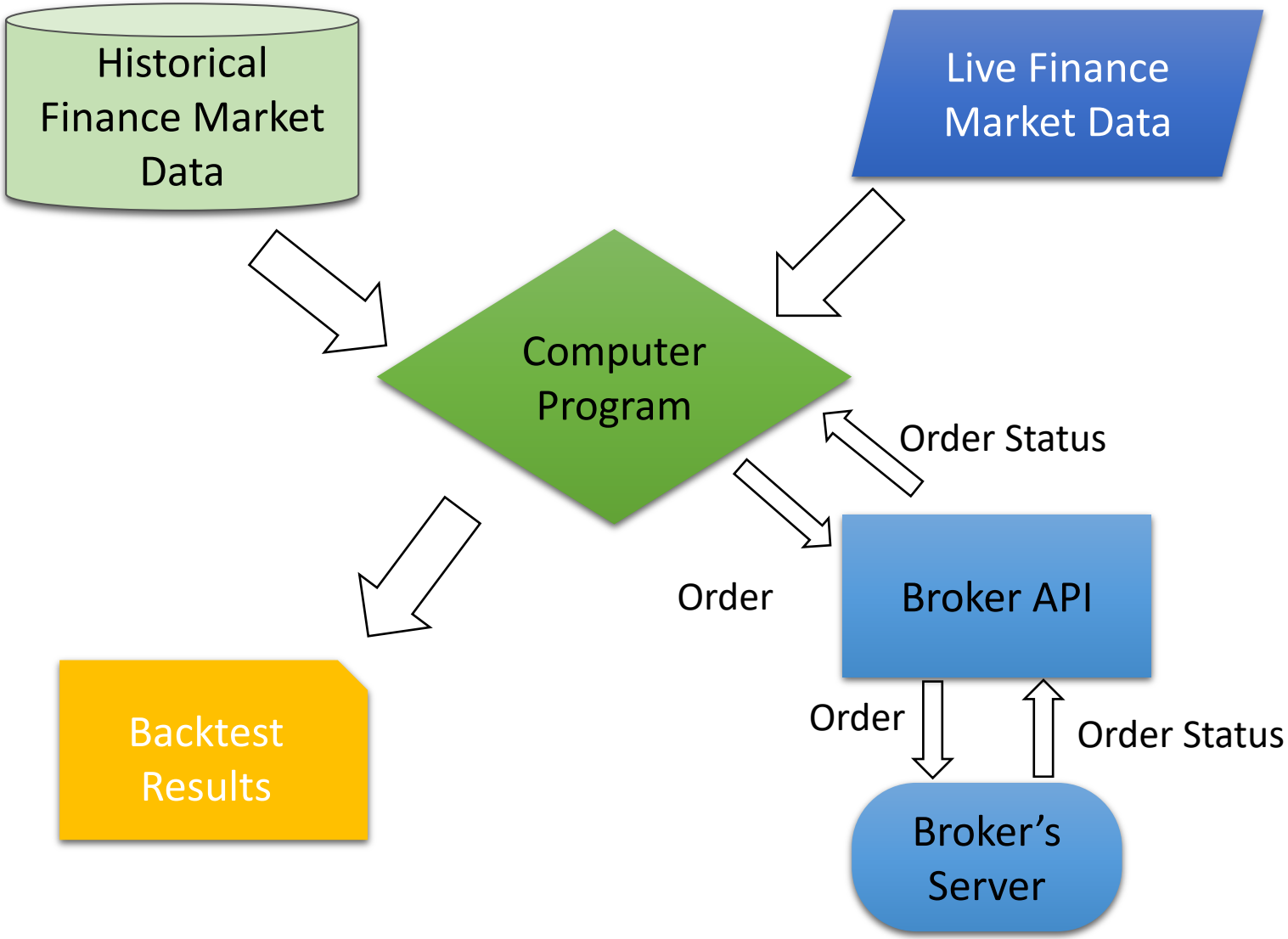
Algo-trading applications embedded with time series forecasting models

Art.	Data set	Period	Feature set	Method	Performance criteria	Environment
[46]	Bitcoin, Dash, Ripple, Monero, Litecoin, Dogecoin, Nxt, Namecoin	2014–2017	MA, BOLL, the CRIX returns, Euribor interest rates, OCHLV	LSTM, RNN, DMLP	Accuracy, F1-measure	Python, Tensorflow
[47]	S&P500, KOSPI, HSI, and EuroStoxx50	1987–2017	200-days stock price	Deep Q-Learning, DMLP	Total profit, Correlation	–
[48]	Stocks in the S&P500	1990–2015	Price data	DMLP, GBT, RF	Mean return, MDD, Calmar ratio	H2O
[49]	Fundamental and Technical Data, Economic Data	–	Fundamental , technical and market information	CNN	–	–

Source: Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer (2020). "Deep learning for financial applications: A survey." Applied Soft Computing (2020): 106384.

Algorithmic Trading

Algorithmic Trading

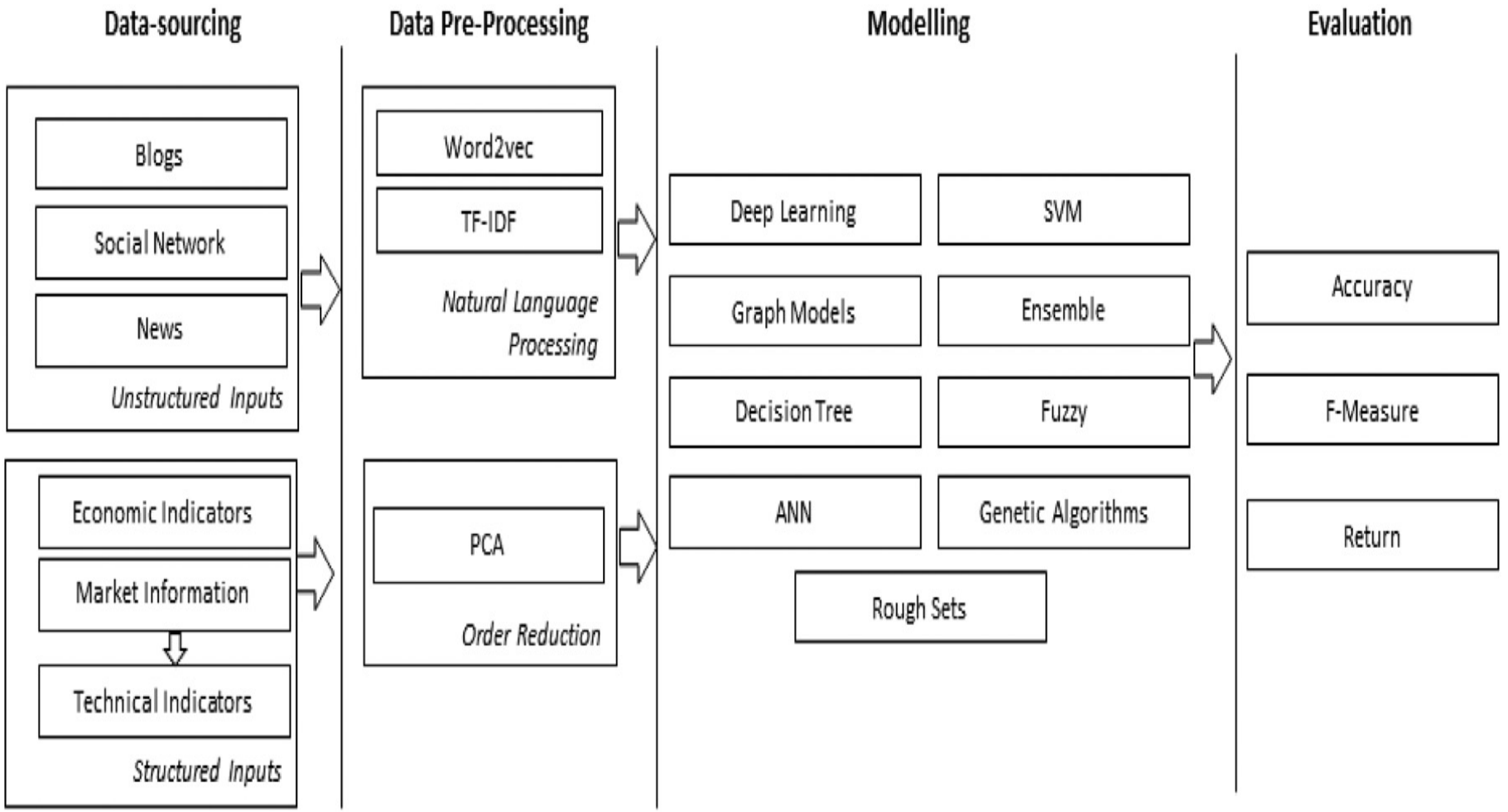


Source: Ernest P. Chan (2017), Machine Trading: Deploying Computer Algorithms to Conquer the Markets, Wiley

Process of Machine Learning in Predicting Cryptocurrency

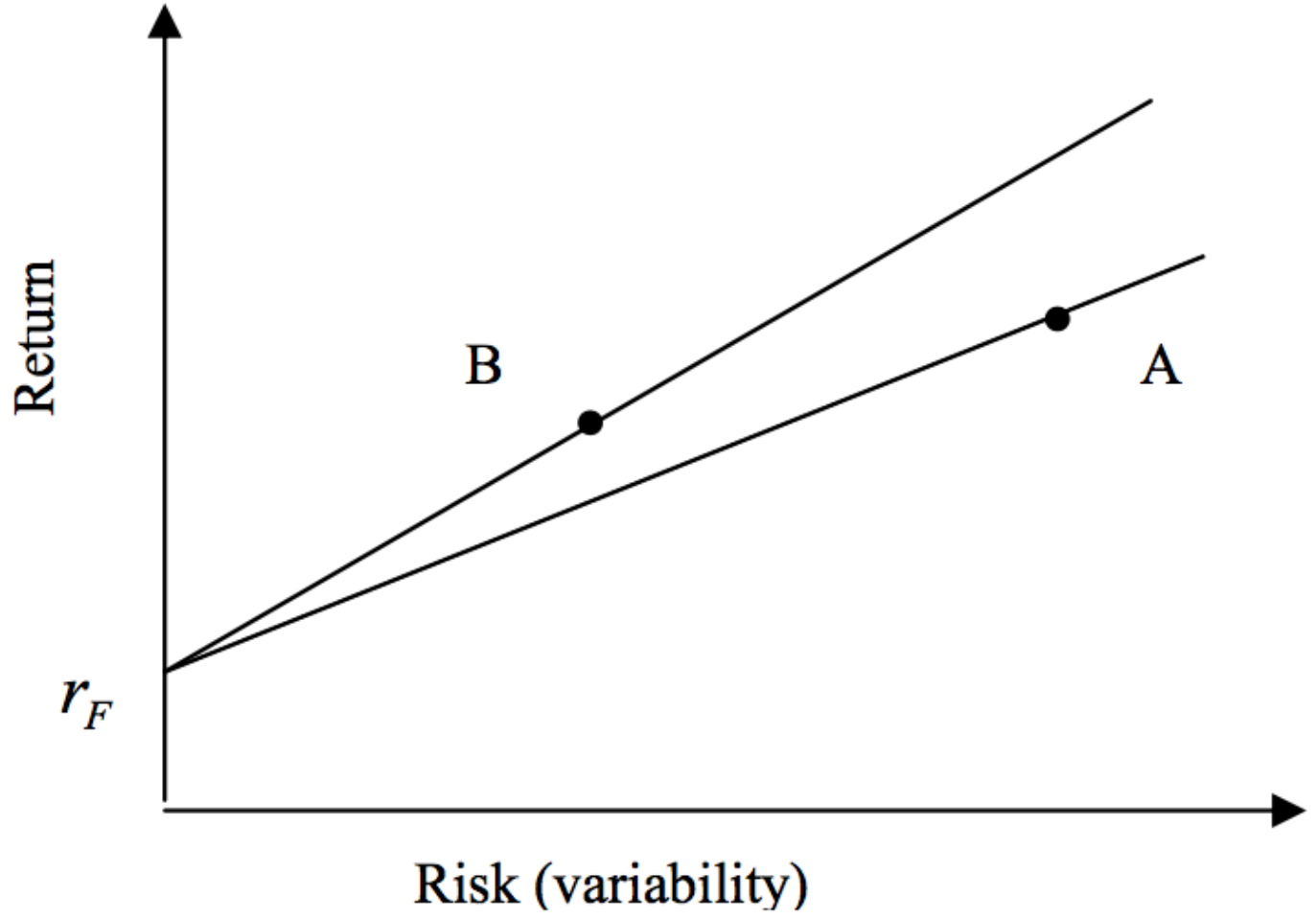
Source: Fan Fang, Carmine Ventre, Michail Basios, Leslie Kanthan, David Martinez-Rego, Fan Wu, and Lingbo Li. (2023) "Cryptocurrency trading: a comprehensive survey." Financial Innovation 8, no. 1 (2022): 1-59.

Stock Market Movement Forecast:
Phases of the stock market modeling



Source: O. Bustos and A. Pomares-Quimbaya (2020), "Stock Market Movement Forecast: A Systematic Review." Expert Systems with Applications (2020): 113464.

Risk and Return



Source: Bacon, Carl. "How sharp is the Sharpe-ratio?-Risk-adjusted Performance Measures." *Statpro White Paper* (2000).

Sharpe Ratio

$$\textbf{Sharpe Ratio} = \frac{Portofolio\ Return - Risk\ Free\ Return}{Portofolio\ Risk}$$

Source: Bacon, Carl. "How sharp is the Sharpe-ratio?-Risk-adjusted Performance Measures." *Statpro White Paper* (2000).

Sharpe Ratio

$$\textbf{Sharpe Ratio } SR = \frac{r_P - r_F}{\sigma_P}$$

Where

r_P = portfolio return

r_F = risk free rate

σ_P = portfolio risk (variability, standard deviation of return)

Source: Bacon, Carl. "How sharp is the Sharpe-ratio?-Risk-adjusted Performance Measures." *Statpro White Paper* (2000).

Sortino Ratio

$$\textbf{Sortino Ratio} = \frac{r_P - r_T}{\sigma_D}$$

Where

r_P = portfolio return

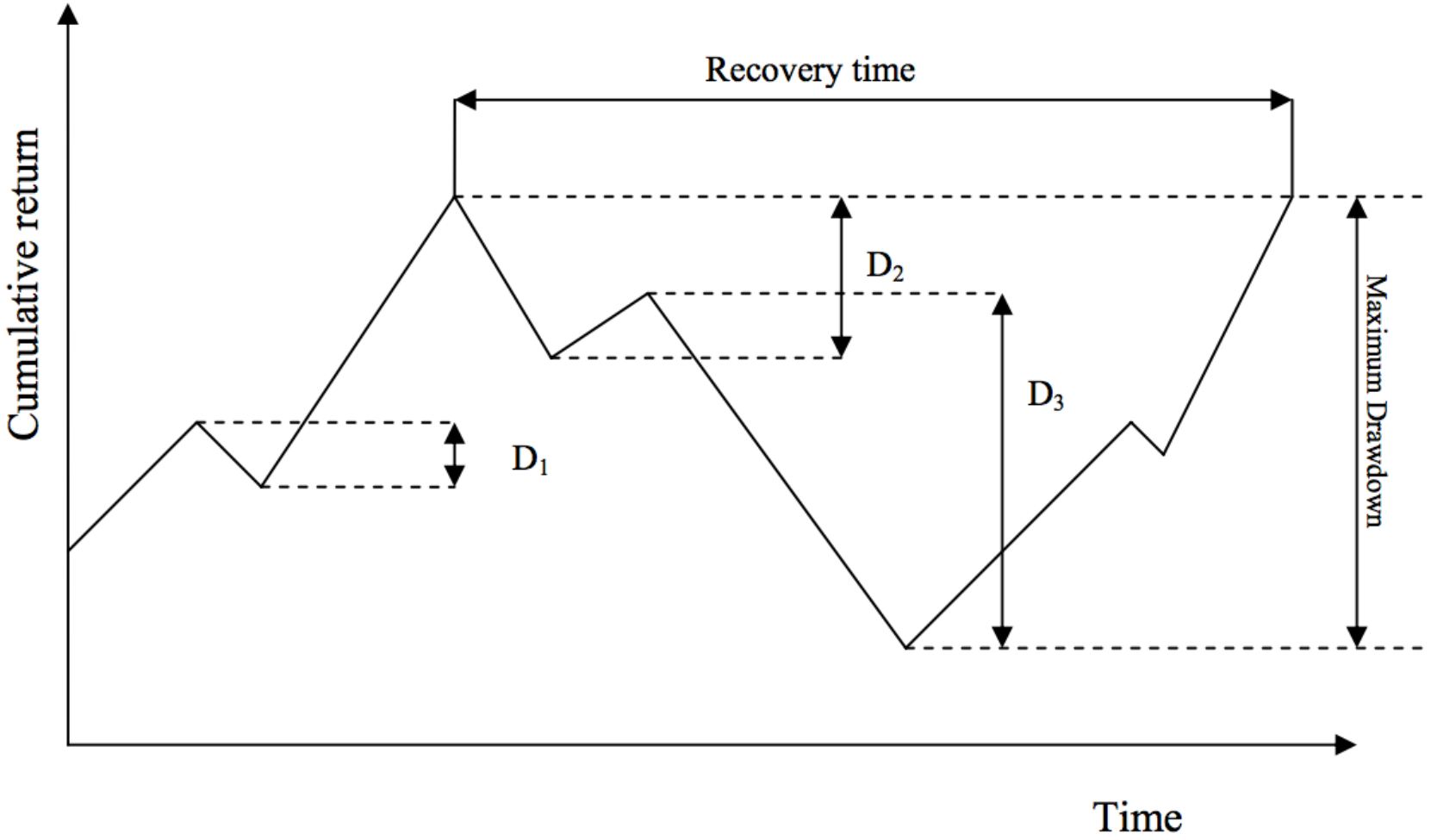
r_T = Minimum Target Return

σ_D = Downside Risk

$$\textbf{Downside Risk } \sigma_D = \sqrt{\sum_{i=1}^{n} \frac{\min[(r_i - rT), 0]^2}{n}}$$

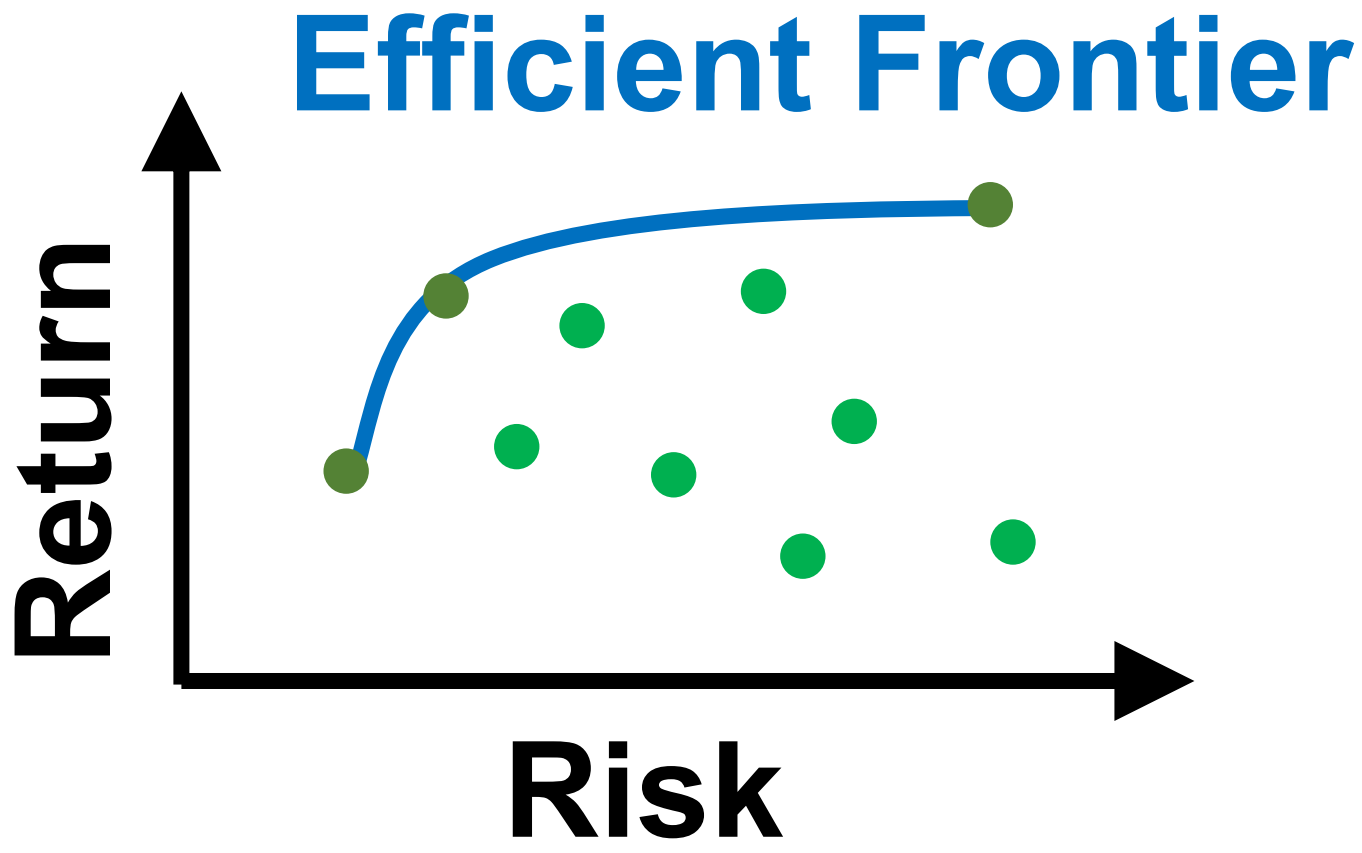
Source: Bacon, Carl. "How sharp is the Sharpe-ratio?-Risk-adjusted Performance Measures." *Statpro White Paper* (2000).

Max Drawdown



Source: Bacon, Carl. "How sharp is the Sharpe-ratio?-Risk-adjusted Performance Measures." *Statpro White Paper* (2000).

Portfolio Optimization
Efficient Frontier



Source: Tucker Balch (2012), Investment Science: Portfolio Optimization, https://www.youtube.com/watch?v=5qbMhXXq0vI

Backtesting

- Financial Functions (ffn)
  - https://pmorissette.github.io/ffn/
- backtesting.py
  - https://kernc.github.io/backtesting.py/
- Visualization
  - Plotly Express (px)
    - https://plotly.com/python/plotly-express/
  - Bokeh
    - https://bokeh.org/

Financial Functions (ffn)
plotly.express (px)

```
!pip install ffn
import ffn
import plotly.express as px
%pylab inline
#BTC-USD Bitcoin USD
df = ffn.get('btc-usd', start='2016-01-01', end='2021-12-31')
print('df')
print(df.head())
print(df.tail())
print(df.describe())
df.plot(figsize=(14,10))

returns = df.to_returns().dropna()
print('returns')
print(returns.head())
print(returns.tail())
print(returns.describe())
#ax = df.plot(figsize=(12,9))

perf = df.calc_stats()
perf.plot(figsize=(14, 10))
print(perf.display())

fig = px.line(df, x=df.index, y="btcusd", title='btcusd')
fig.update_layout(title='btcusd price', xaxis_title='Date', yaxis_title='Price')
#fig.update_traces(mode='markers+lines')
fig.show()

fig = px.line(returns, x=returns.index, y="btcusd", title='btcusd')
fig.update_layout(title='btcusd returns', xaxis_title='Date', yaxis_title='Returns')
fig.show()

fig = px.histogram(returns, x='btcusd', nbins=40, histnorm='probability', width=800, height=400)
fig.update_layout(title='btcusd returns histogram')
fig.show()

fig = px.box(returns, y='btcusd', points = 'all')
fig.update_layout(title='btcusd returns box')
fig.update_traces(boxmean='sd')
fig.show()
```

Financial Functions (ffn)
plotly.express (px)

```
# Upgrade pandas-datareader
!pip install --upgrade pandas
!pip install --upgrade pandas-datareader
```

```
!pip install ffn
import ffn
import plotly.express as px
%pylab inline
#BTC-USD Bitcoin USD
df = ffn.get('btc-usd', start='2016-01-01', end='2021-12-31')
print('df')
print(df.head())
print(df.tail())
print(df.describe())
df.plot(figsize=(14,10))
```

Financial Functions (ffn)
plotly.express (px)

```
returns = df.to_returns().dropna()
print('returns')
print(returns.head())
print(returns.tail())
print(returns.describe())
#ax = df.plot(figsize=(12,9))
```

Financial Functions (ffn)
plotly.express (px)

```
perf = df.calc_stats()
perf.plot(figsize=(14, 10))
print(perf.display())

fig = px.line(df, x=df.index, y="btcusd", title='btcusd')
fig.update_layout(title='btcusd price', xaxis_title='Date',
yaxis_title='Price')
#fig.update_traces(mode='markers+lines')
fig.show()

fig = px.line(returns, x=returns.index, y="btcusd", title='btcusd')
fig.update_layout(title='btcusd returns', xaxis_title='Date',
yaxis_title='Returns')
fig.show()
```

Financial Functions (ffn)
plotly.express (px)

```
fig = px.histogram(returns, x='btcusd', nbins=40,
histnorm='probability', width=800, height=400)

fig.update_layout(title='btcusd returns histogram')
fig.show()

fig = px.box(returns, y='btcusd', points = 'all')
fig.update_layout(title='btcusd returns box')
fig.update_traces(boxmean='sd')
fig.show()
```

Financial Functions (ffn)

```
 btcusd
Date
2016-01-01  434.334015
2016-01-02  433.437988
2016-01-03  430.010986
2016-01-04  433.091003
2016-01-05  431.959991
                btcusd
Date
2021-12-28  47588.855469
2021-12-29  46444.710938
2021-12-30  47178.125000
2021-12-31  46306.445312
2022-01-01  47686.812500
             btcusd
count    2193.000000
mean    13025.164562
std     16489.530523
min       364.330994
25%      2589.409912
50%      7397.796875
75%     11358.662109
max     67566.828125
```

Financial Functions (ffn)

calc_stats() display()

Stat	btcusd
Start	2016-01-01
End	2022-01-01
Risk-free rate	0.00%
Total Return	10879.29%
Daily Sharpe	1.18
Daily Sortino	1.95
CAGR	118.79%
Max Drawdown	-83.40%
Calmar Ratio	1.42

Financial Functions (ffn)

calc_stats() display()

```
MTD                    2.98%
3m                     -0.89%
6m                     42.04%
YTD                    2.98%
1Y                     62.34%
3Y (ann.)              131.46%
5Y (ann.)              116.71%
10Y (ann.)             -
Since Incep. (ann.)    118.79%
```

Financial Functions (ffn)

`calc_stats() display()`

```
Daily Sharpe            1.18
Daily Sortino           1.95
Daily Mean (ann.)       74.04%
Daily Vol (ann.)        62.94%
Daily Skew              -0.10
Daily Kurt              7.30
Best Day                25.25%
Worst Day               -37.17%
```

Financial Functions (ffn)

`calc_stats() display()`

```
Monthly Sharpe          1.38
Monthly Sortino         3.75
Monthly Mean (ann.)     114.20%
Monthly Vol (ann.)      82.59%
Monthly Skew            0.43
Monthly Kurt            -0.16
Best Month              69.63%
Worst Month             -36.41%
```

Financial Functions (ffn)

calc_stats() display()

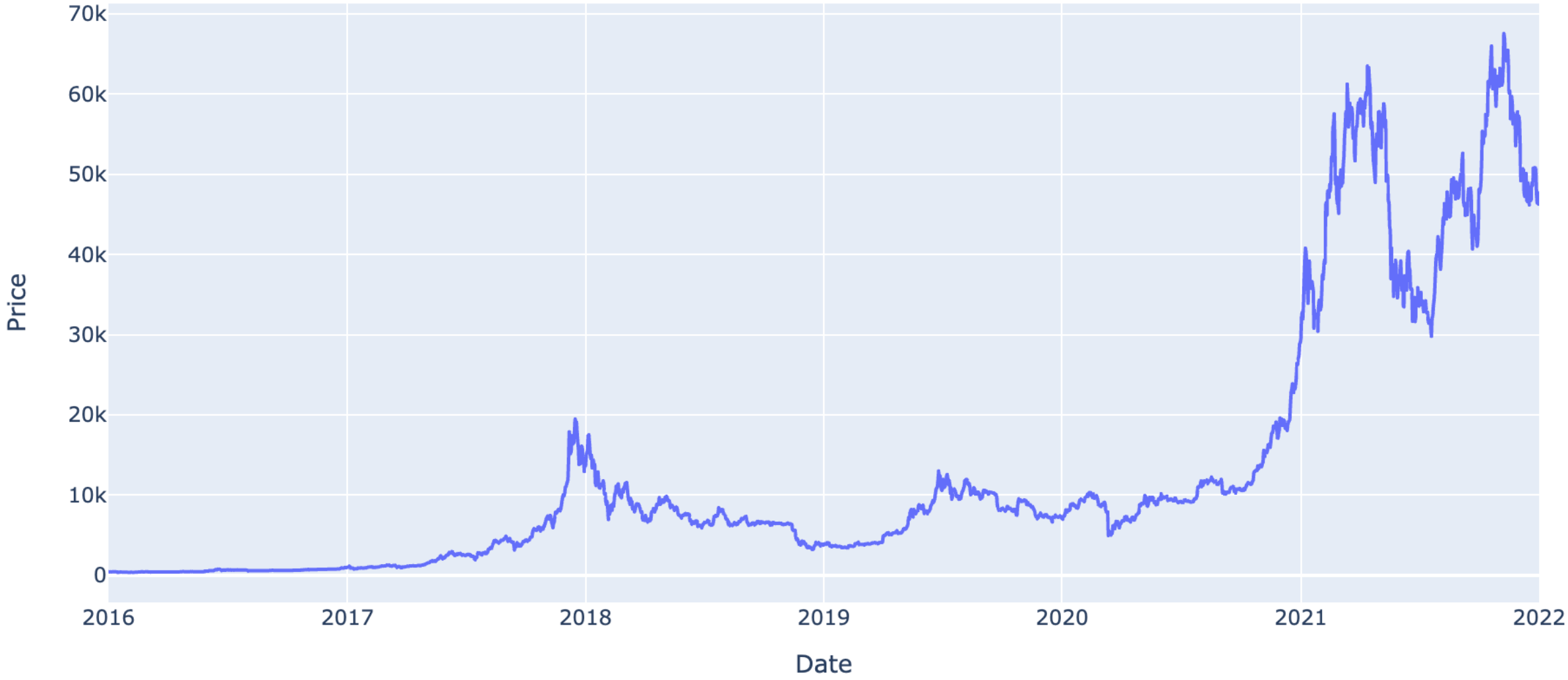
```
Yearly Sharpe             0.54
Yearly Sortino            9.73
Yearly Mean               292.22%
Yearly Vol                542.38%
Yearly Skew               2.17
Yearly Kurt               4.86
Best Year                 1368.90%
Worst Year                -73.56%
```

Financial Functions (ffn)

calc_stats() display()

```
Avg. Drawdown          -10.25%
Avg. Drawdown Days     36.55
Avg. Up Month          25.13%
Avg. Down Month        -12.35%
Win Year %             83.33%
Win 12m %              85.48%
```

Visualization
plotly.express (px)



Backtesting Output

```
backtesing output
Start                     2016-01-01 00:00:00
End                       2022-01-01 00:00:00
Duration                   2192 days 00:00:00
Exposure Time [%]                   97.993616
Equity Final [$]                4237449.058157
Equity Peak [$]                 6165339.439633
Return [%]                         4137.449058
Buy & Hold Return [%]             10879.294935
Return (Ann.) [%]                    86.557668
Volatility (Ann.) [%]               144.748975
Sharpe Ratio                          0.597985
Sortino Ratio                         1.946086
Calmar Ratio                          1.362652
Max. Drawdown [%]                   -63.521467
Avg. Drawdown [%]                   -12.142095
Max. Drawdown Duration       557 days 00:00:00
Avg. Drawdown Duration        44 days 00:00:00
# Trades                                  116
Win Rate [%]                         35.344828
Best Trade [%]                      119.026467
Worst Trade [%]                     -23.393531
Avg. Trade [%]                        3.291328
Max. Trade Duration           74 days 00:00:00
Avg. Trade Duration           19 days 00:00:00
Profit Factor                         2.293983
Expectancy [%]                        5.036865
SQN                                   1.236071
_strategy                             SmaCross
_equity_curve                              ...
_trades                      Size  Entry..
```

describe()

	High	Low	Open	Close	Volume	Adj Close	
count	2193.00	2193.00	2193.00	2193.00	2.193000e+03	2193.00	
mean	13363.00	12616.08	13005.79	13025.16	1.757591e+10	13025.16	
std	16935.24	15960.65	16480.00	16489.53	2.085247e+10	16489.53	
min	374.95	354.91	365.07	364.33	2.851400e+07	364.33	
25%	2682.26	2510.48	2577.77	2589.41	1.182870e+09	2589.41	
50%	7535.72	7233.40	7397.13	7397.80	9.175292e+09	7397.80	
75%	11570.79	11018.13	11354.30	11358.66	2.886756e+10	11358.66	
max	68789.62	66382.06	67549.73	67566.83	3.509679e+11	67566.83	

Backtesting

```
# Upgrade pandas-datareader
!pip install --upgrade pandas
!pip install --upgrade pandas-datareader

!pip install backtesting
from backtesting import Backtest, Strategy
from backtesting.lib import crossover
from backtesting.test import SMA

import pandas as pd
import pandas_datareader.data as web
df = web.DataReader("BTC-USD", 'yahoo', '2016-01-01', '2021-12-31')
df.to_csv('BTC-USD.csv')
print(df.head().round(2))
print(df.tail().round(2))
print(df.describe().round(2))

class SmaCross(Strategy):
      n1 = 5
      n2 = 20

      def init(self):
            close = self.data.Close
            self.sma1 = self.I(SMA, close, self.n1)
            self.sma2 = self.I(SMA, close, self.n2)

      def next(self):
            if crossover(self.sma1, self.sma2):
                self.buy()
            elif crossover(self.sma2, self.sma1):
                self.sell()

bt = Backtest(df, SmaCross, cash=100000, commission=.002, exclusive_orders=True)

output = bt.run()
print('backtesing output')
print(output)

bt.plot()
```

Backtesting

```
#!pip install backtesting
from backtesting import Backtest, Strategy
from backtesting.lib import crossover
from backtesting.lib import plot_heatmaps
from backtesting.test import SMA

import pandas as pd
import pandas_datareader.data as web

from google.colab import files
import time
#BTC-USD ETH-USD
v_symbol = 'BTC-USD'
v_time_start = '2016-01-01'
v_time_end = '2021-12-31'
v_to_csv_filename = v_symbol + '_' + v_time_start + '_' + v_time_end + '.csv'
df = web.DataReader(v_symbol, 'yahoo', v_time_start, v_time_end)
df.to_csv(v_to_csv_filename)

print(df.head().round(2))
print(df.tail().round(2))
print(df.describe().round(2))
v_n1 = 5 #5 #20 #60 #120
v_n2 = 200 #20 #60 #120 #240
```

Backtesting

```
class SmaCross(Strategy):
    n1 = v_n1 #5
    n2 = v_n2 #60

    def init(self):
        close = self.data.Close
        self.sma1 = self.I(SMA, close, self.n1)
        self.sma2 = self.I(SMA, close, self.n2)

    def next(self):
        if crossover(self.sma1, self.sma2):
                self.buy()
        elif crossover(self.sma2, self.sma1):
                self.sell()

bt = Backtest(df, SmaCross, cash=100000, commission=.002, exclusive_orders=True)

stats = bt.run()
```

Backtesting

```
filename = v_symbol + '_' + v_time_start + '_' + v_time_end + '_' + 'MA_' +
str(v_n1) + '_' + str(v_n2) + '.csv'
print('filename:', filename)
stats.to_csv(filename)

print('backtesing stats')
print(stats)
bt.plot()

print('filename:\t', filename)
print("stats._strategy:\t", stats._strategy)
print("# Trades:\t", stats['# Trades'])
print("stats['Equity Final [$]']:\t", round(stats['Equity Final [$]'], 4))
print("stats['Avg. Trade [%]']:\t", round(stats['Avg. Trade [%]'], 4))
print("Sharpe Ratio:\t", round(stats['Sharpe Ratio'], 4))

#download file
time.sleep(1) # time sleep 1 second
files.download(filename)
print('file downloaded:', filename)
```

Backtesting

```
print('*****bt.optimize*****')
stats, heatmap = bt.optimize(
   n1 = range(5, 65, 5),
   n2 = range(10, 205, 5),
   constraint = lambda param: param.n1 <param.n2,
   maximize = 'Avg. Trade [%]',
   max_tries = 600,
   random_state = 0,
   return_heatmap = True)

#'Equity Final [$]' 'Avg. Trade [%]'

optimize_strategy = stats._strategy
```

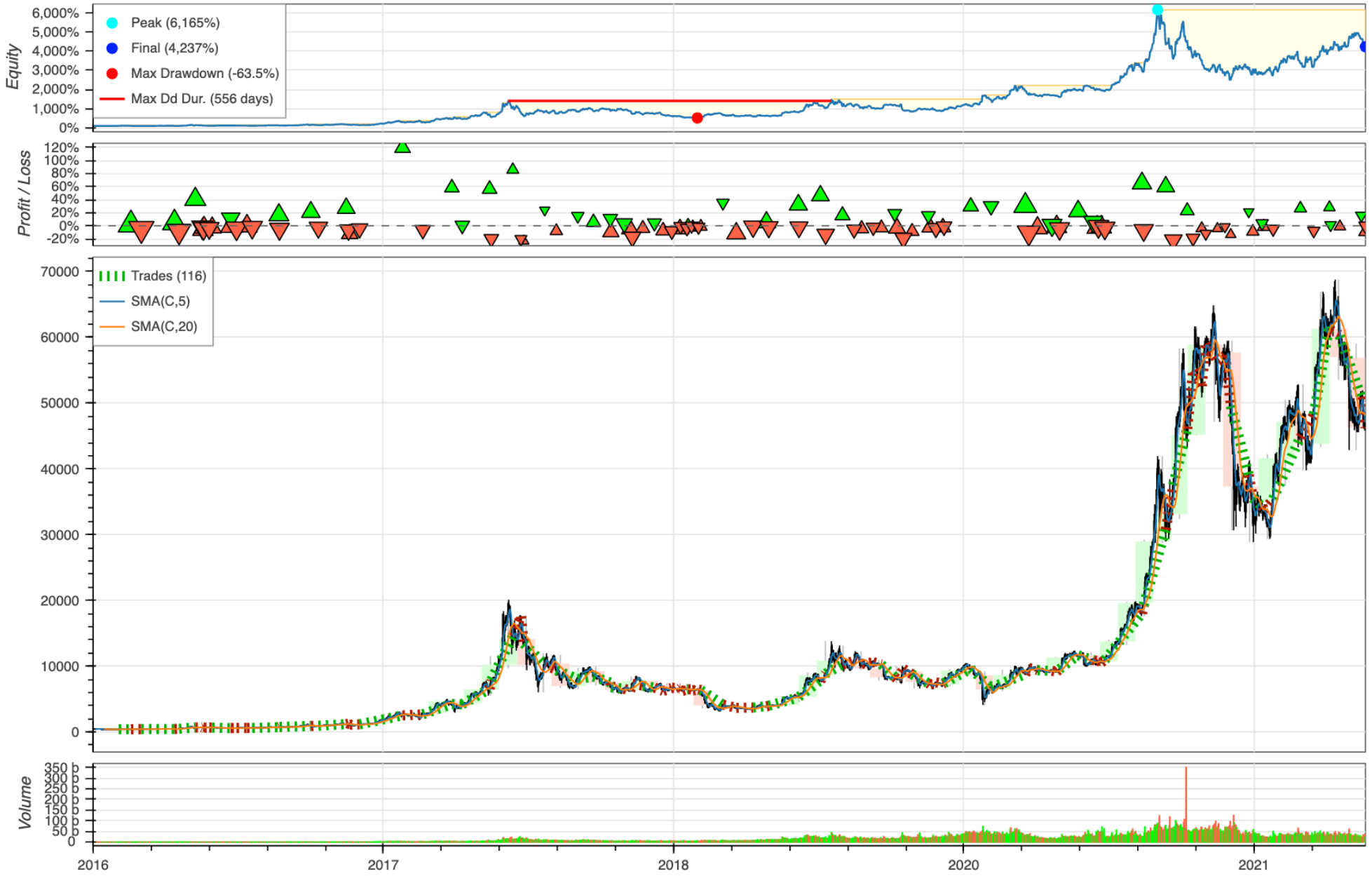
Backtesting

```
optimize_filename = v_symbol + '_' + v_time_start + '_' + v_time_end + '_' +
'bt_optimize_strategy' + str(optimize_strategy) + '.csv'
print('optimize_filename:', optimize_filename)
print('backtesing optimize strategy stats')
print(stats)
stats.to_csv(optimize_filename)
plot_heatmaps(heatmap, agg='mean', plot_width = 1800)

print('backtesting optimize strategy heatmap')
print(heatmap)
print('backtesting optimize strategy heatmap Top 10')
print(heatmap.sort_values().iloc[-10:])
hm = heatmap.groupby(['n1', 'n2']).mean().unstack()
print('backtesting optimize strategy heatmap mean')
print(hm)
hm_filename = v_symbol + '_' + v_time_start + '_' + v_time_end + '_' +
'hm_heatmap.csv'
hm.to_csv(hm_filename)
```

Backtesting

```
print("filename:\t", optimize_filename)
print("stats._strategy:\t", stats._strategy)
print("# Trades:\t", stats['# Trades'])
print("stats['Equity Final [$]']:\t", round(stats['Equity Final [$]'], 4))
print("stats['Avg. Trade [%]']:\t", round(stats['Avg. Trade [%]'], 4))
print("Sharpe Ratio:\t", round(stats['Sharpe Ratio'], 4))

#download file
time.sleep(1) # time sleep 1 second
files.download(hm_filename)
print('file downloaded:', hm_filename)
files.download(optimize_filename)
print('file downloaded:', optimize_filename)
```

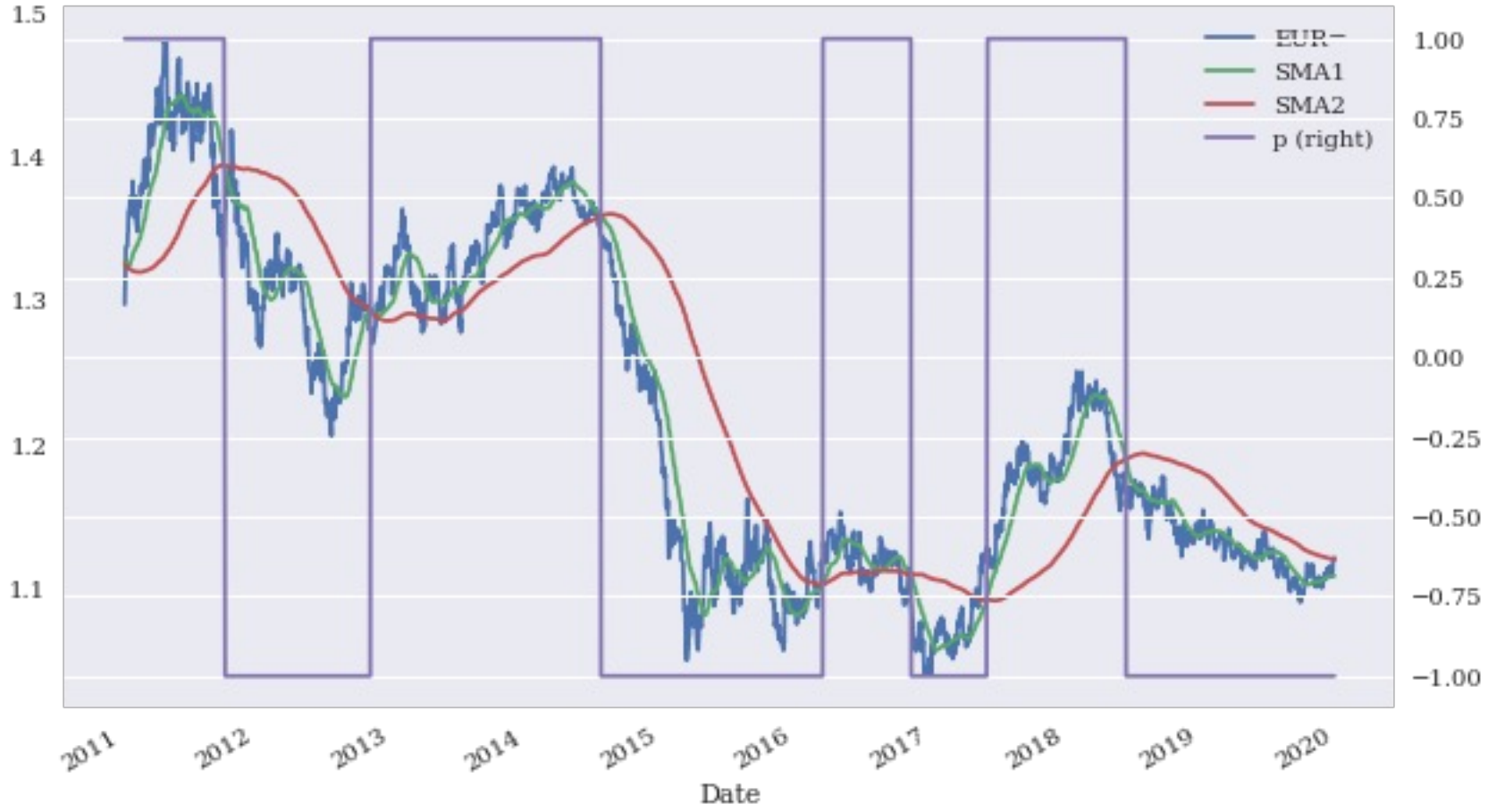
Backtesting



Time series data for EUR/USD and SMAs

Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

Time series data for EUR/USD, SMAs, and resulting positions



Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

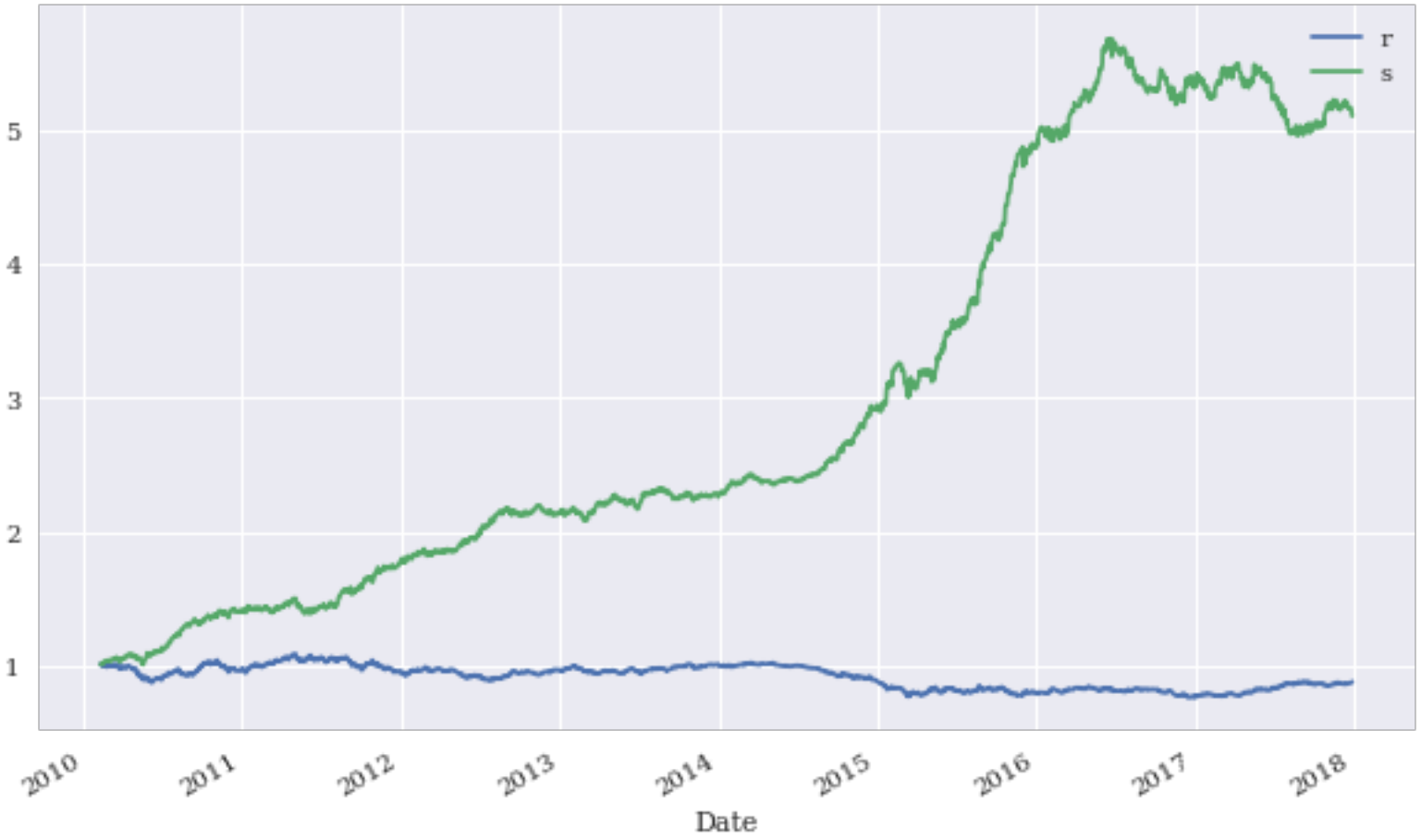
Gross performance of passive benchmark investment and SMA strategy

Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

Gross performance of the SMA strategy before and after transaction costs

Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

Gross performance of the passive benchmark investment and the daily DNN strategy (in-sample)



Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

Gross performance of the passive benchmark investment and the daily DNN strategy (out-of-sample)

Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

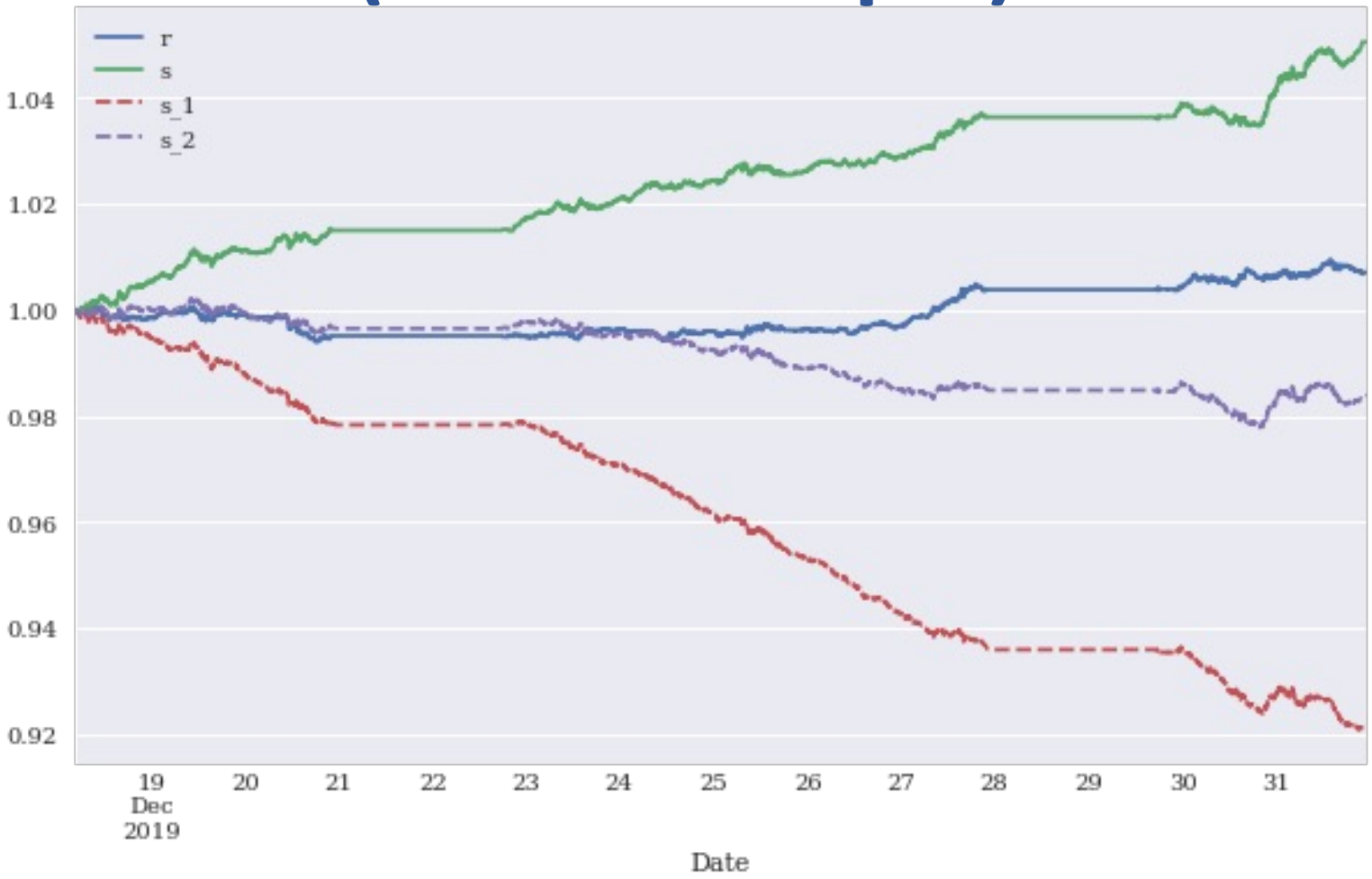
Gross performance of the daily DNN strategy before and after transaction costs (out-of-sample)

Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

Gross performance of the passive benchmark investment and the DNN intraday strategy (out-of-sample)

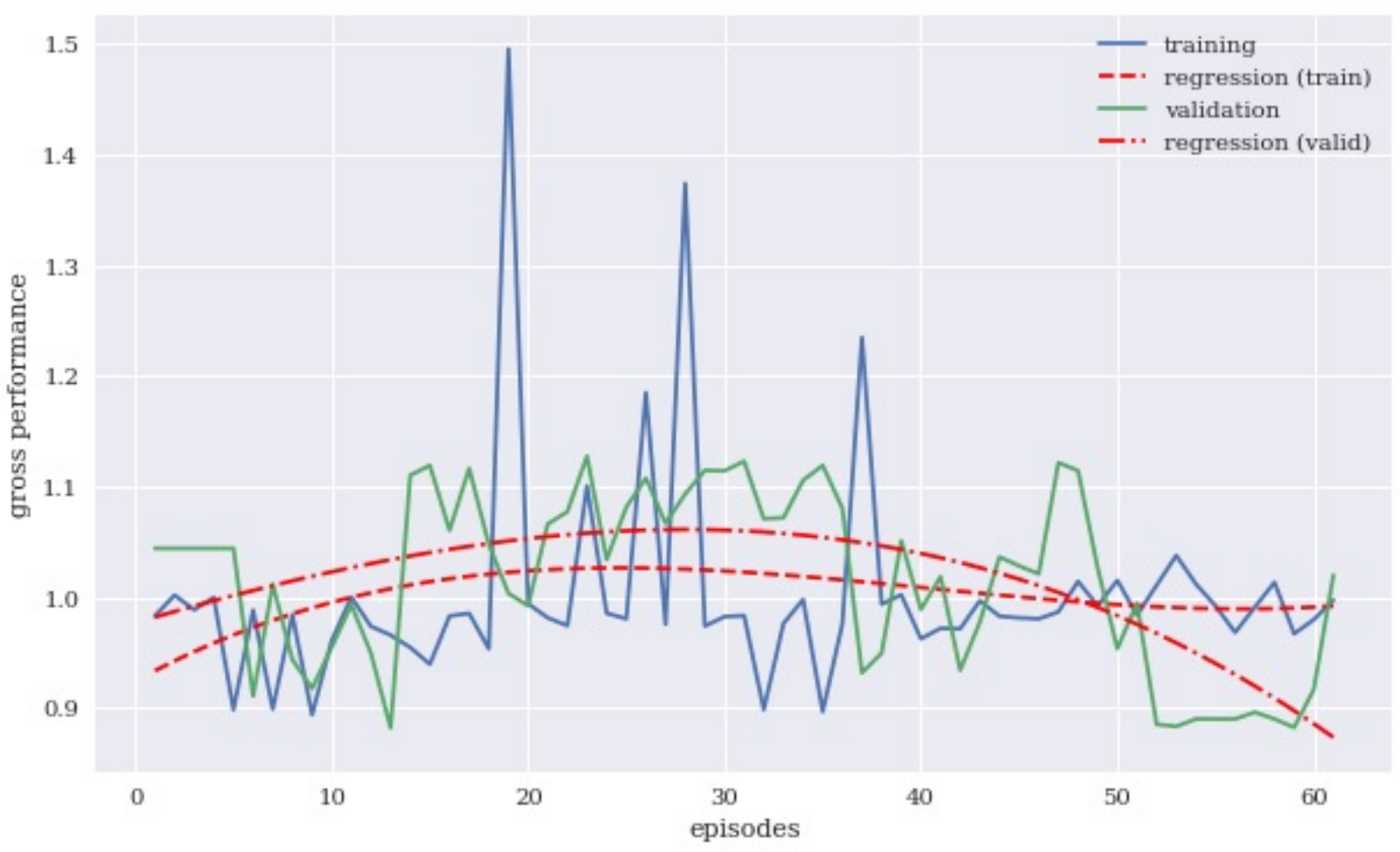
Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

Gross performance of the DNN intraday strategy before and after higher/ lower transaction costs (out-of-sample)



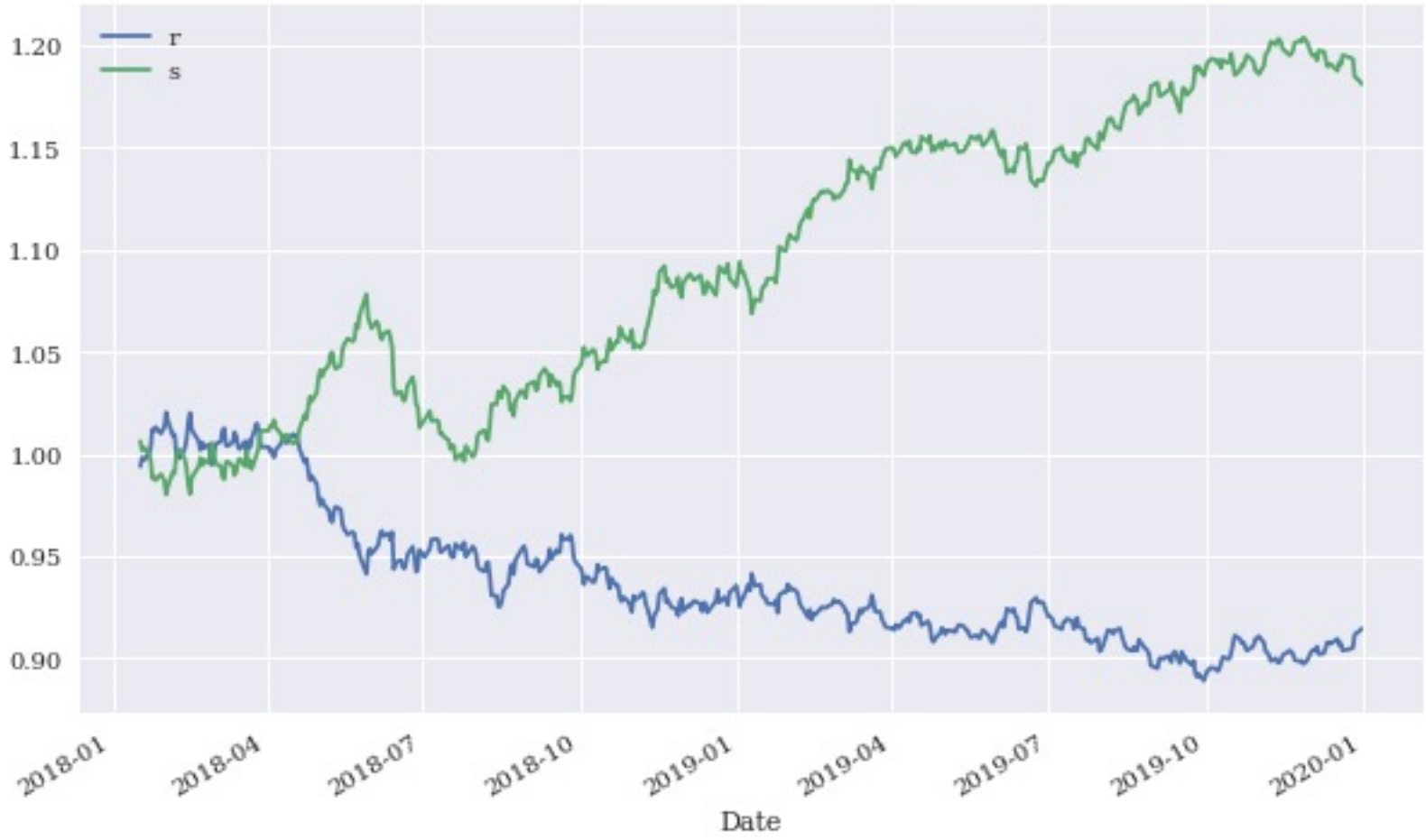
Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

Gross performance on training and validation data set



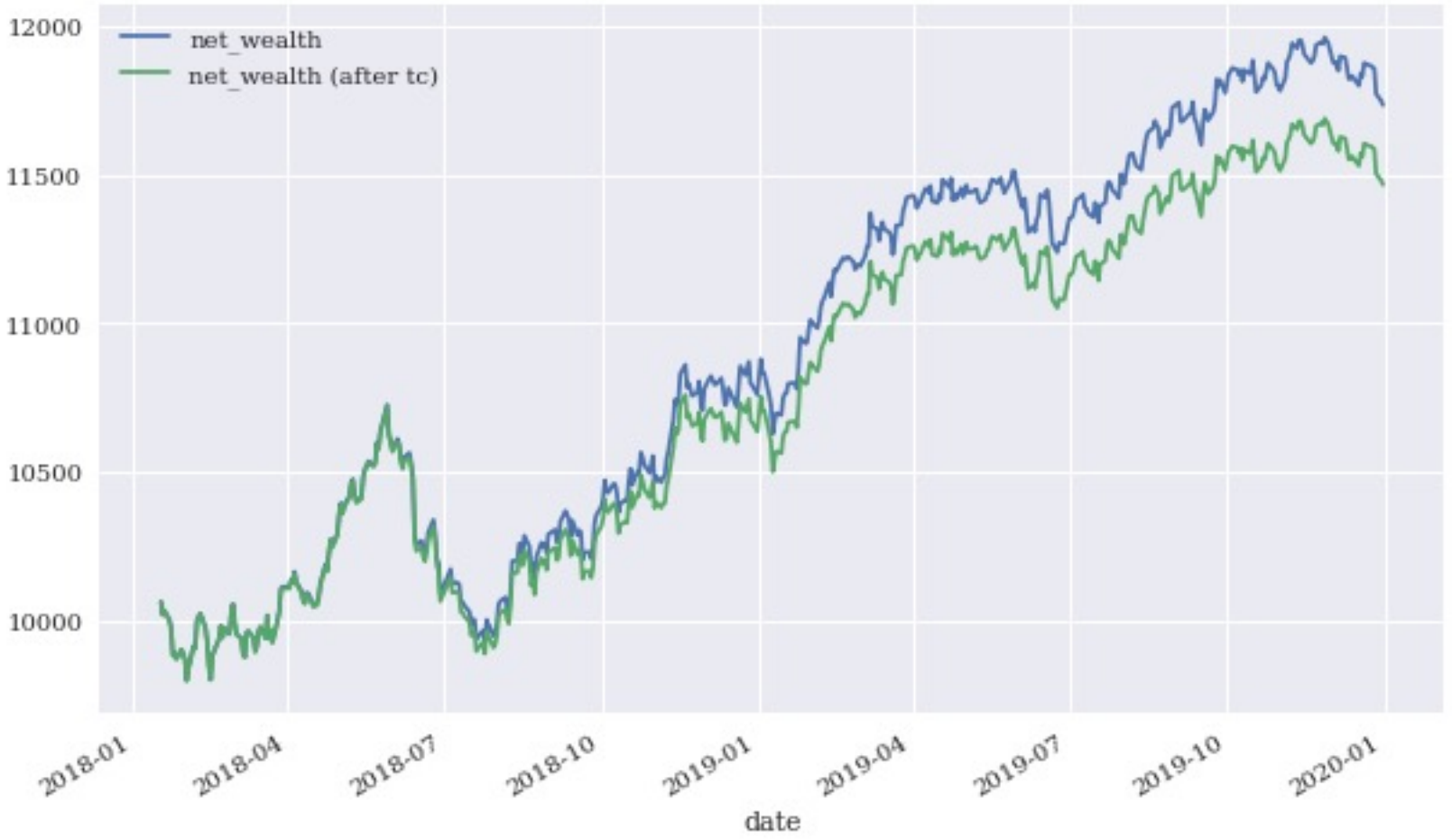
Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

Gross performance of the passive benchmark investment and the trading bot (out-of-sample)



Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

Gross performance of the trading bot before and after transaction costs (in-sample)

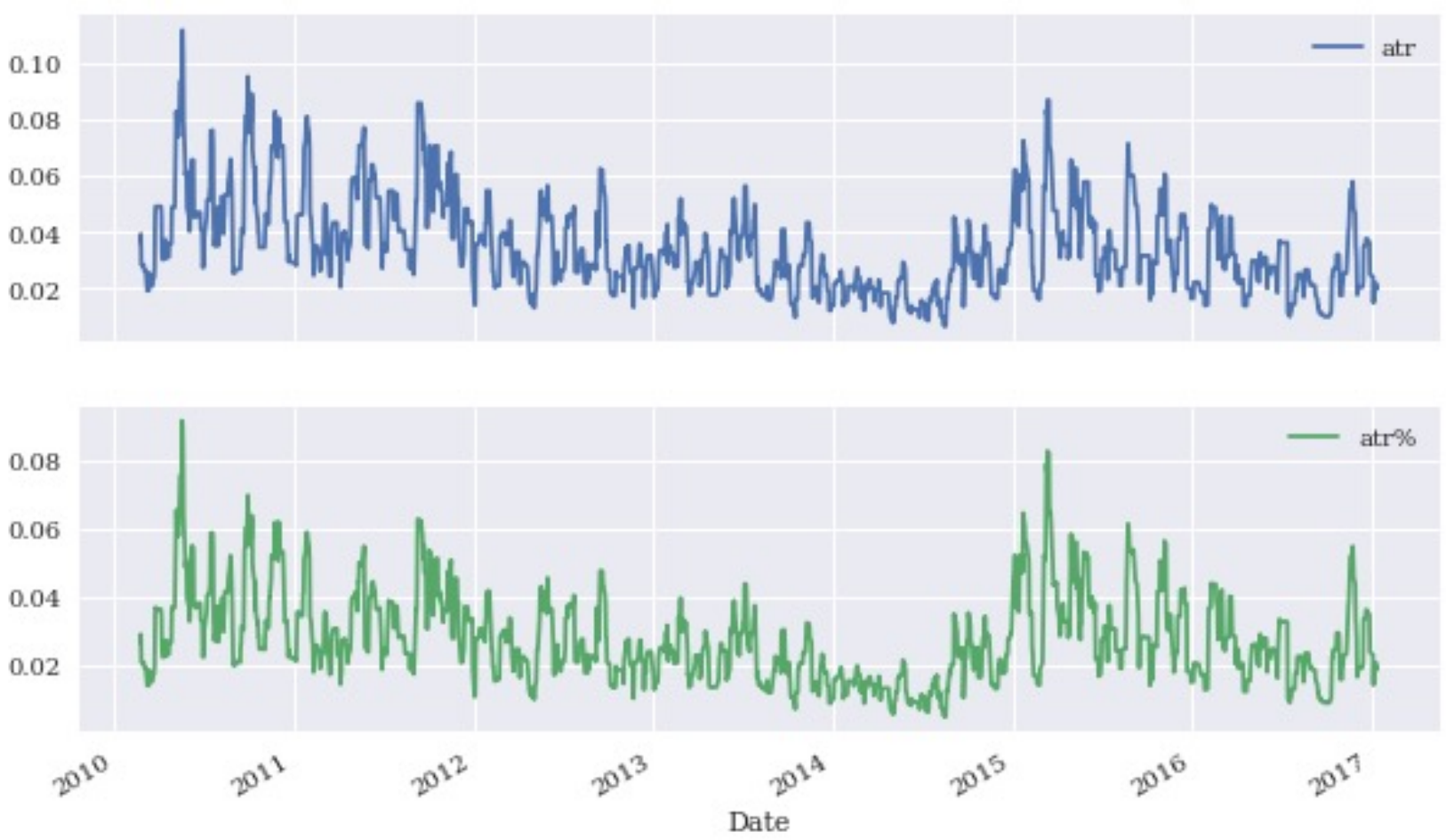


Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

Gross performance of the passive benchmark investment and the trading bot (vectorized and event-based backtesting)

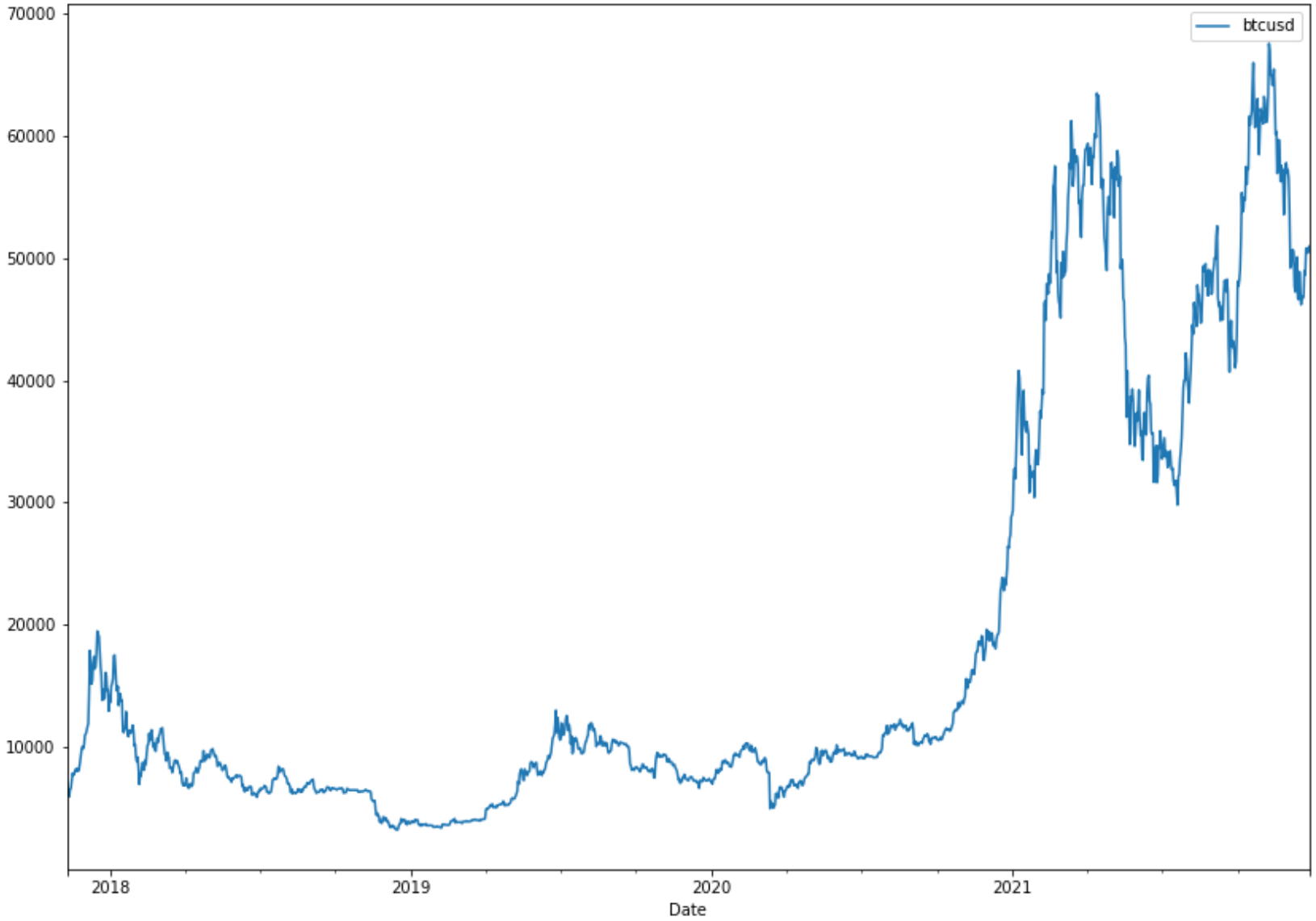
Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

Average true range (ATR) in absolute (price) and relative (%) terms



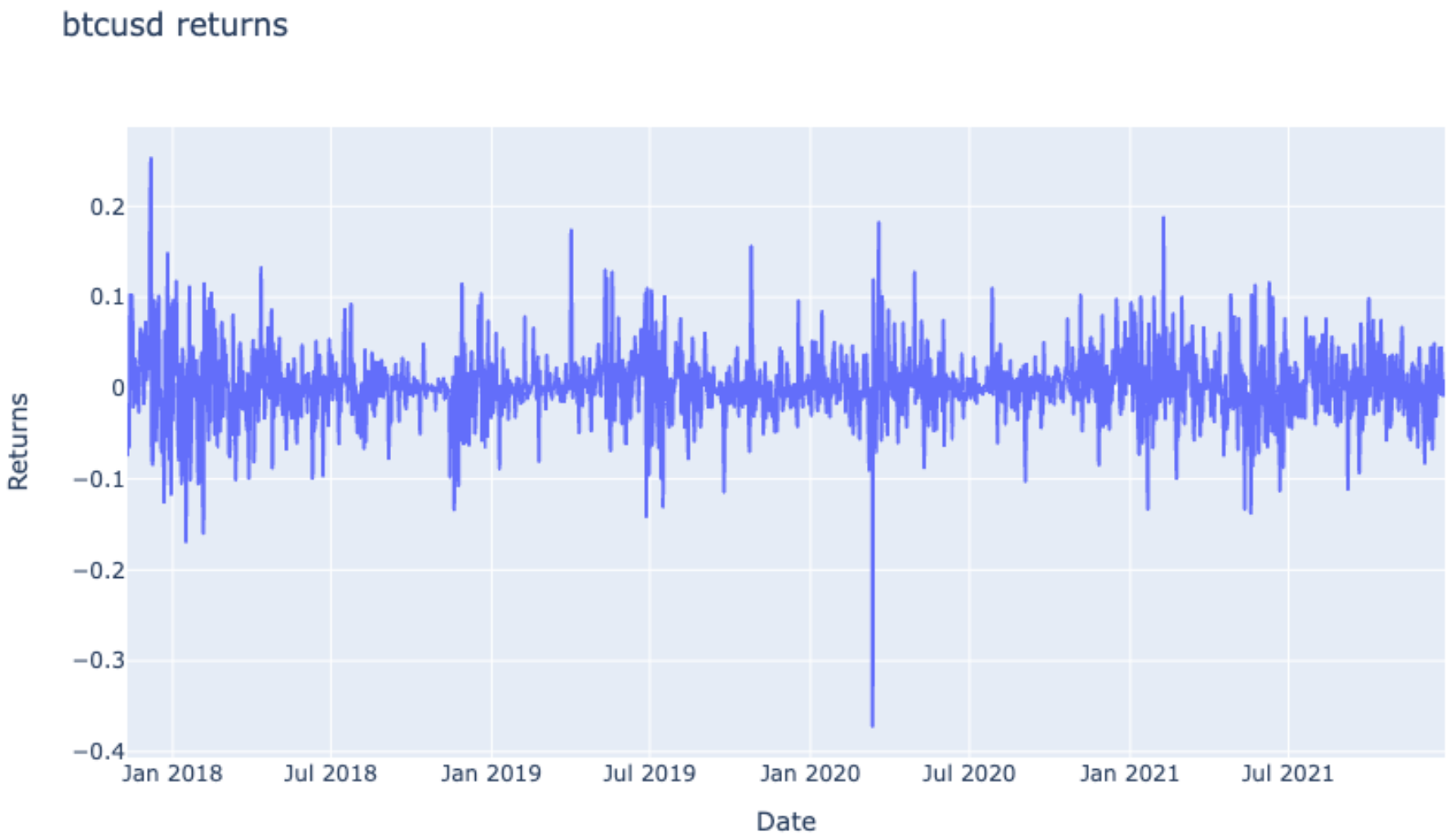
Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

BTC-USD



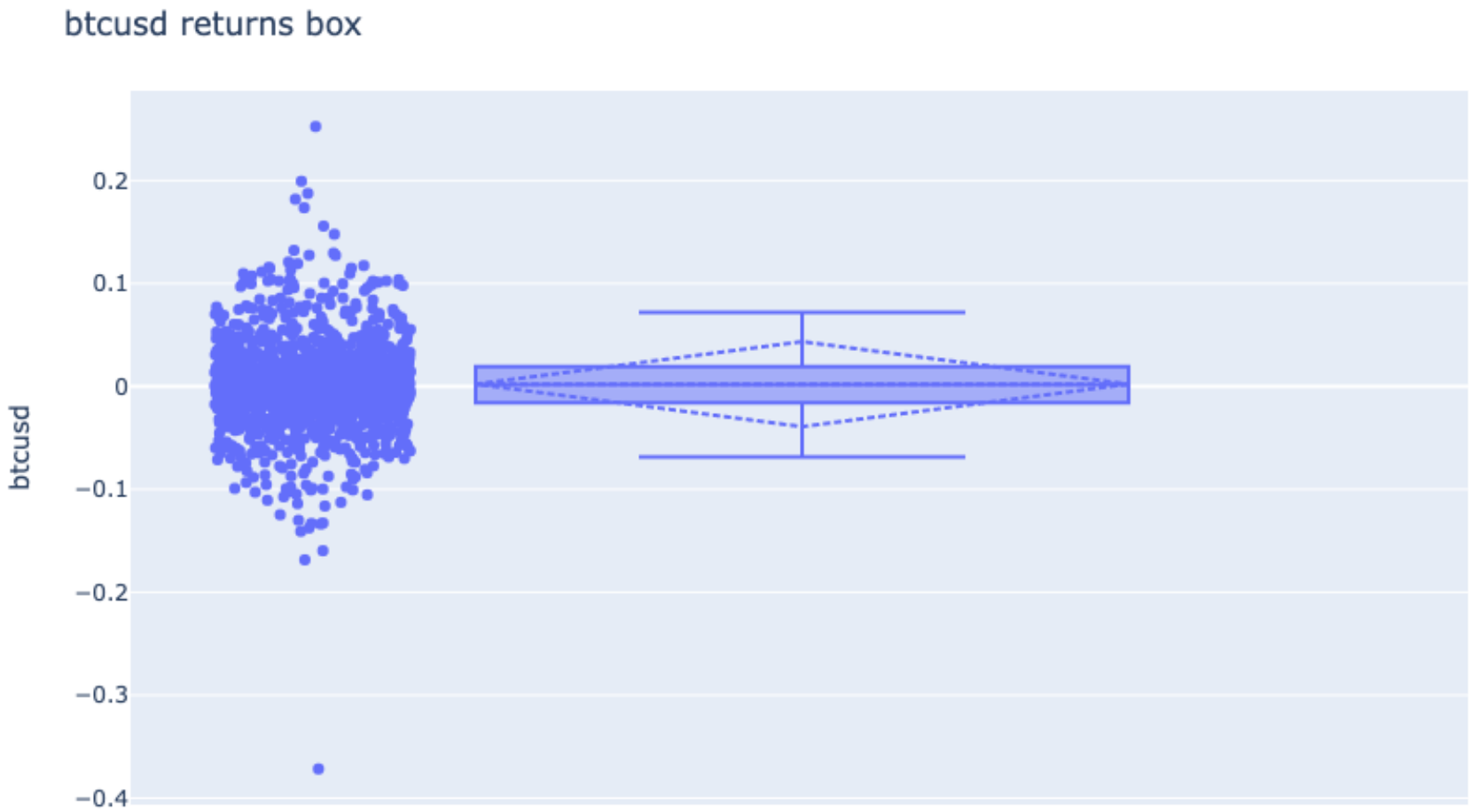
Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

BTC-USD Returns



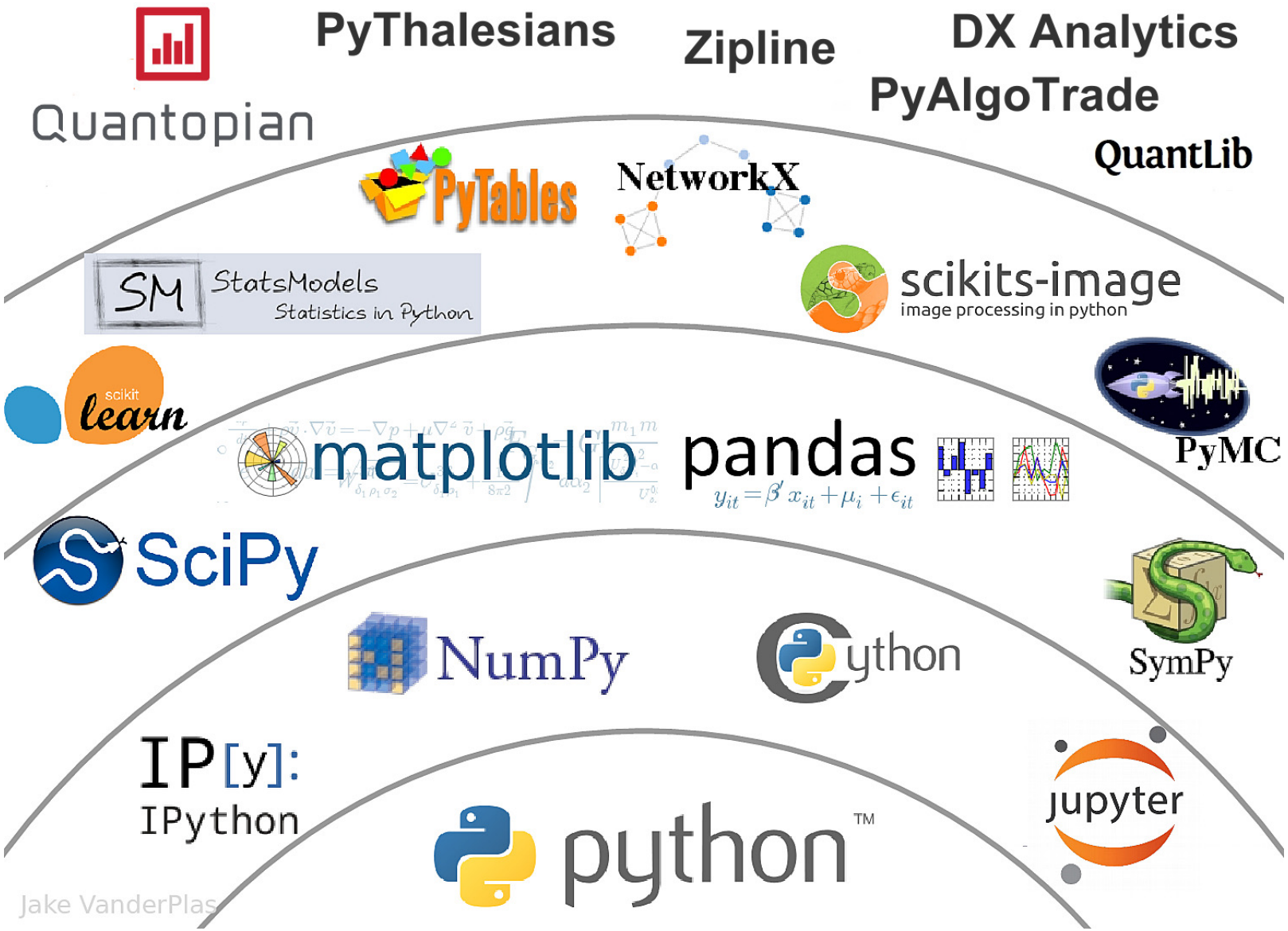
Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

BTC-USD Returns Box



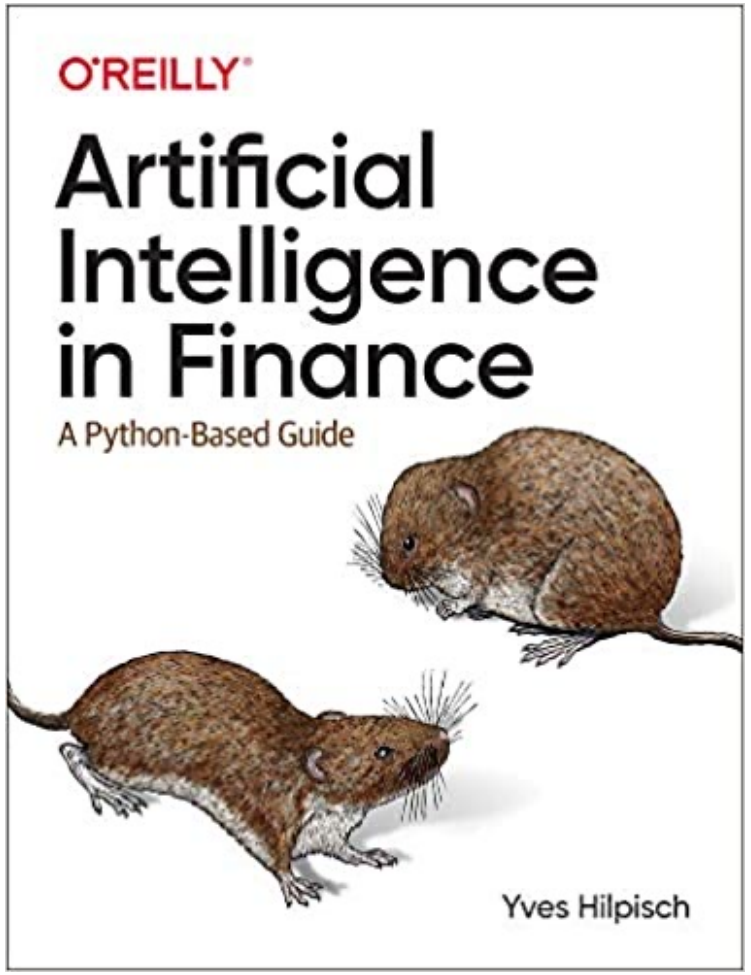
Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

The Quant Finance PyData Stack



Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#/5

Yves Hilpisch (2020),
Artificial Intelligence in Finance:
A Python-Based Guide,
O'Reilly



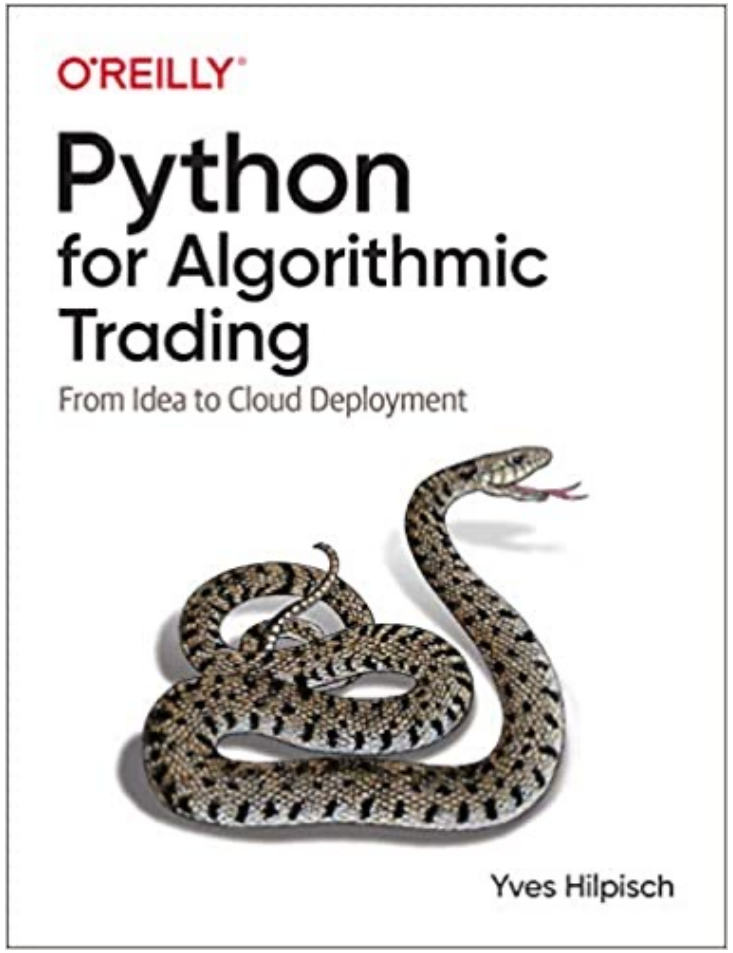
Source: https://www.amazon.com/Artificial-Intelligence-Finance-Python-Based-Guide/dp/1492055433

Yves Hilpisch (2020),

Python for Algorithmic Trading:

From Idea to Cloud Deployment,

O'Reilly



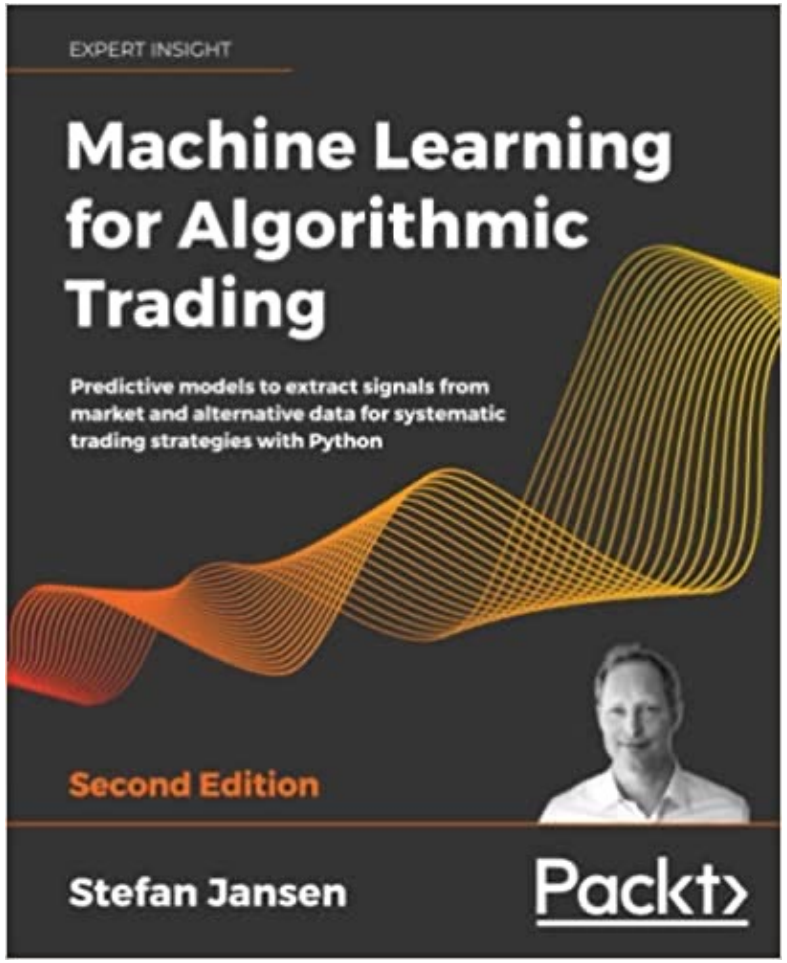
Source: https://www.amazon.com/Python-Algorithmic-Trading-Cloud-Deployment/dp/149205335X

Stefan Jansen (2020),

Machine Learning for Algorithmic Trading:

Predictive models to extract signals from market and alternative data for systematic trading strategies with Python, 2nd Edition,

Packt Publishing.



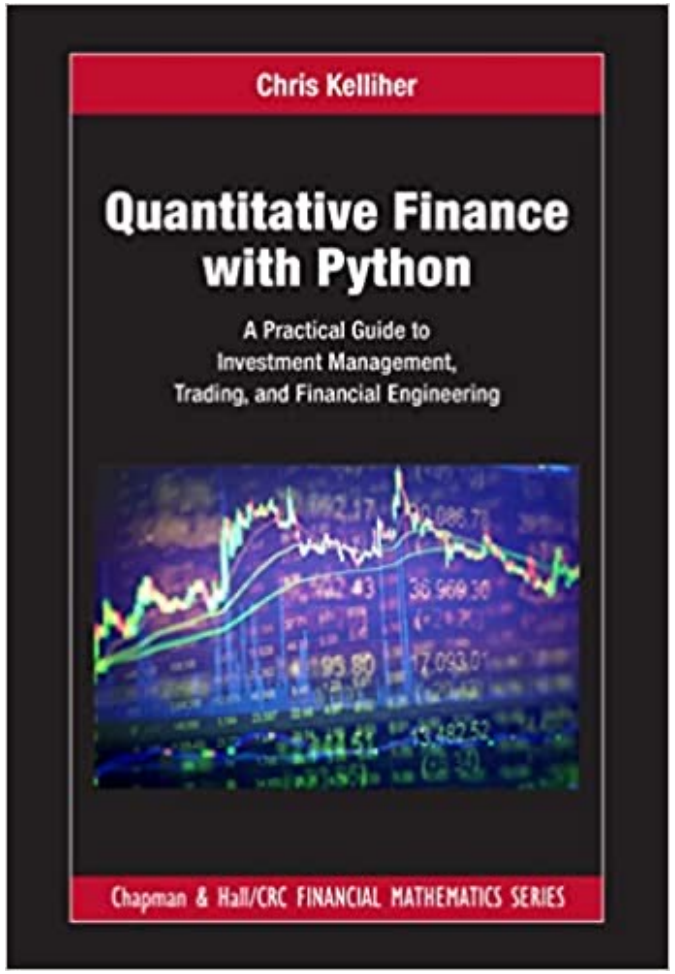
Source: https://www.amazon.com/Machine-Learning-Algorithmic-Trading-alternative/dp/1839217715/

Chris Kelliher (2022),

Quantitative Finance With Python:

A Practical Guide to Investment Management, Trading, and Financial Engineering,

Chapman and Hall/CRC.



Source: https://www.amazon.com/Quantitative-Finance-Python-Engineering-Mathematics/dp/1032014431/

Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly

yhilpisch / aiif Public

https://github.com/yhilpisch/aiif

Notifications | Star 98 | Fork 77

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights

main | 1 branch | 0 tags | Go to file | Code

yves Code updates for TF 2.3. e334251 on Dec 8, 2020 4 commits

code	Code updates for TF 2.3.	11 months ago
.gitignore	Code updates for TF 2.3.	11 months ago
LICENSE.txt	Code updates.	11 months ago
README.md	Code updates.	11 months ago

README.md

Artificial Intelligence in Finance

About this Repository

This repository provides Python code and Jupyter Notebooks accompanying the **Artificial Intelligence in Finance** book published by O'Reilly.

O'REILLY

About

Jupyter Notebooks and code for the book Artificial Intelligence in Finance (O'Reilly) by Yves Hilpisch.

home.tpq.io/books/aiif

Readme

View license

Releases

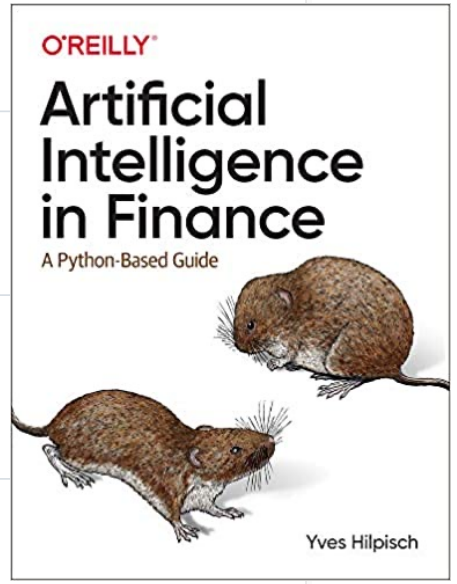
No releases published

Packages

No packages published

Languages

Jupyter Notebook 97.4% Python 2.6%



Source: https://github.com/yhilpisch/aiif

Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly

yhilpisch / aiif Public

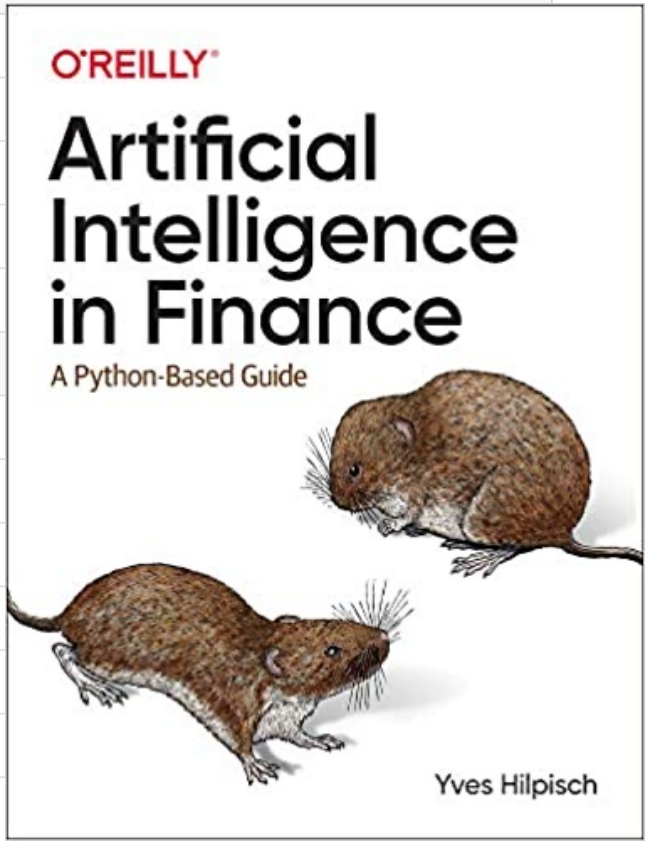
Notifications | Star 98 | Fork 77

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights

main | aiif / code / https://github.com/yhilpisch/aiif/tree/main/code | Go to file

yves Code updates for TF 2.3. e334251 on Dec 8, 2020 History

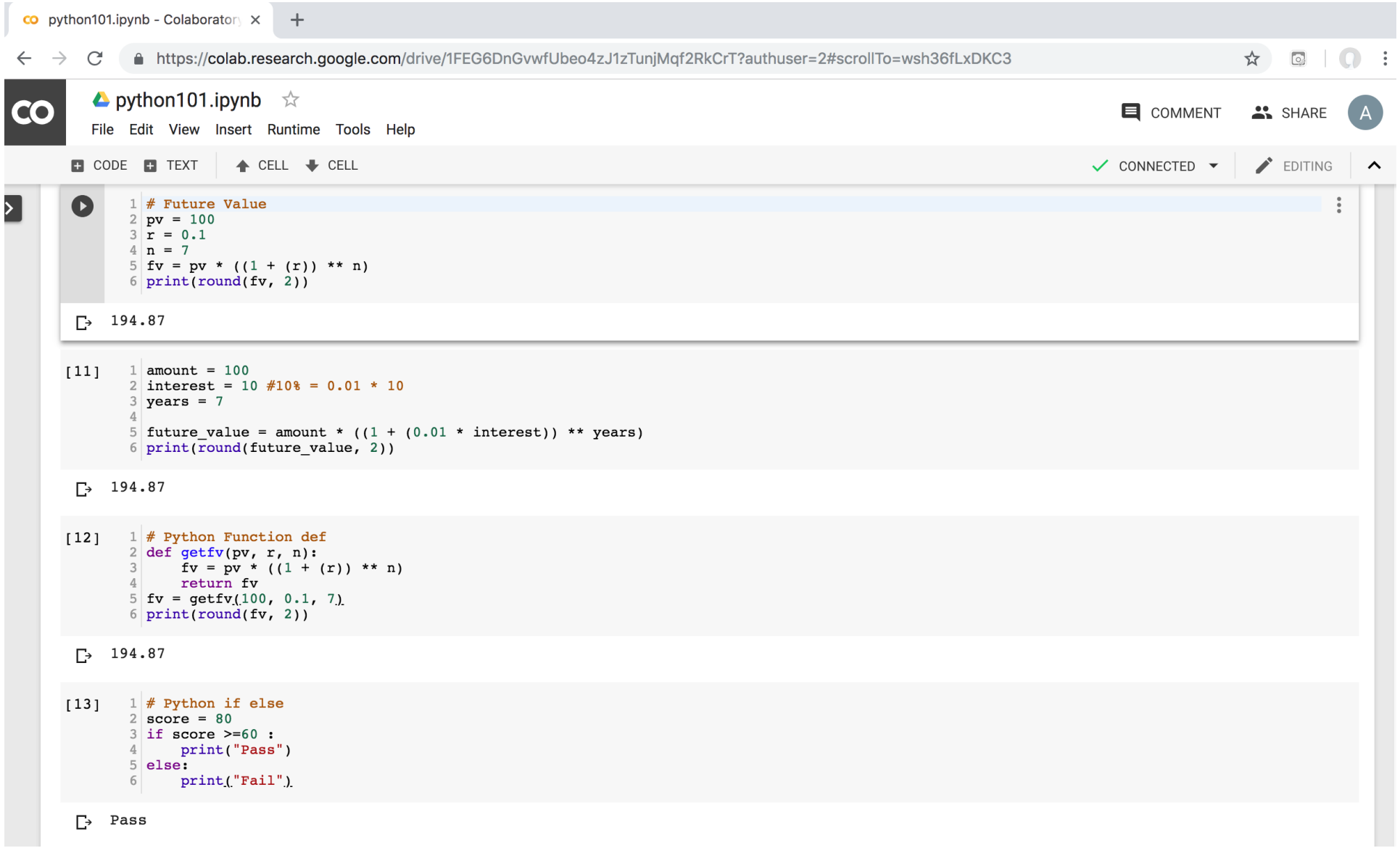
..	
oanda	Code updates for TF 2.3.
01_artificial_intelligence.ipynb	Code updates for TF 2.3.
02_superintelligence.ipynb	Code updates for TF 2.3.
03_normative_finance.ipynb	Code updates for TF 2.3.
04_data_driven_finance_a.ipynb	Initial commit.
04_data_driven_finance_b.ipynb	Initial commit.
05_machine_learning.ipynb	Code updates for TF 2.3.
06_ai_first_finance.ipynb	Code updates for TF 2.3.
07_dense_networks.ipynb	Code updates for TF 2.3.
08_recurrent_networks.ipynb	Code updates for TF 2.3.
09_reinforcement_learning_a.ipynb	Code updates.
09_reinforcement_learning_b.ipynb	Code updates for TF 2.3.



Source: https://github.com/yhilpisch/aiif/tree/main/code

Python in Google Colab (Python101)

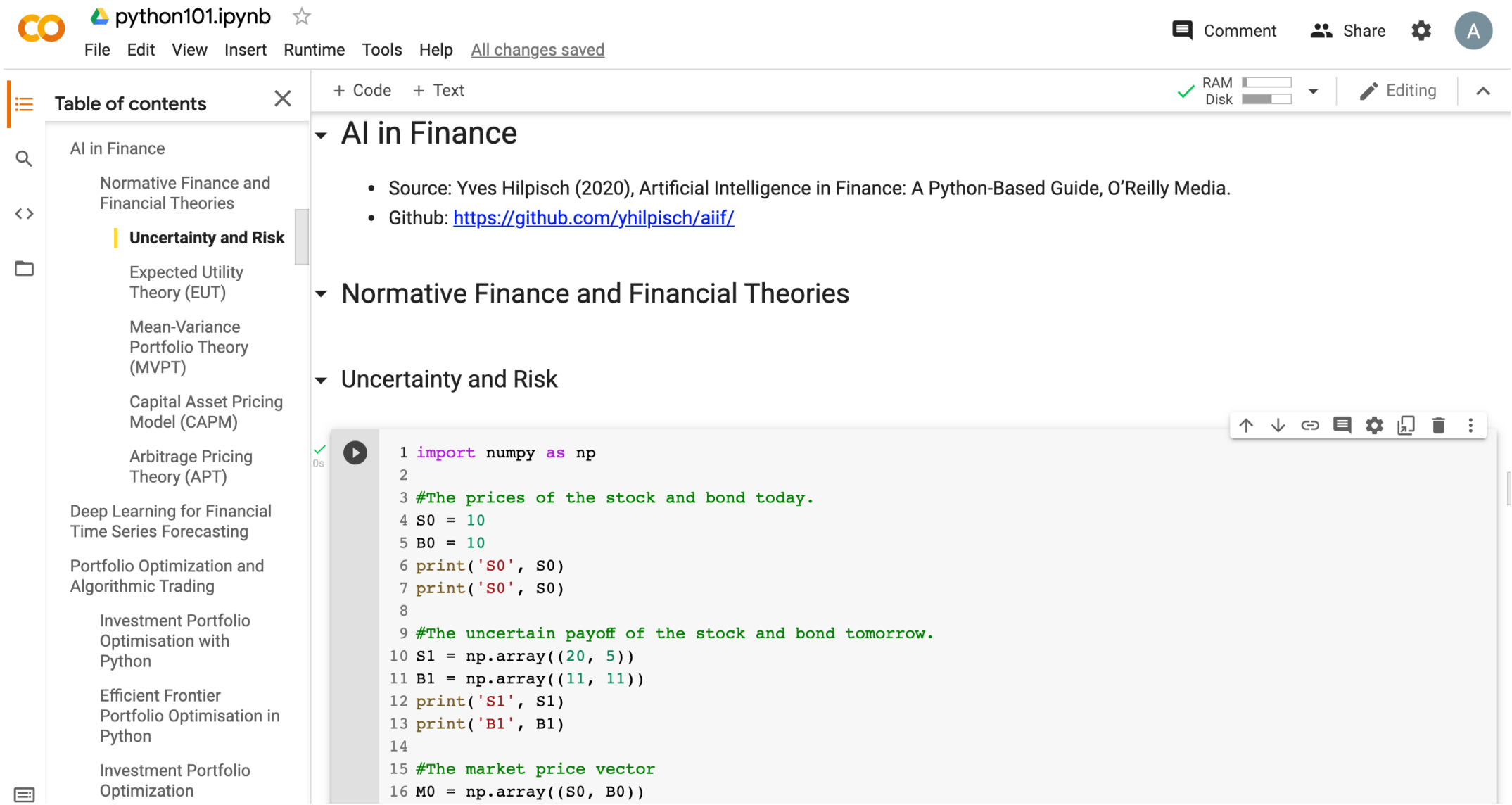
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

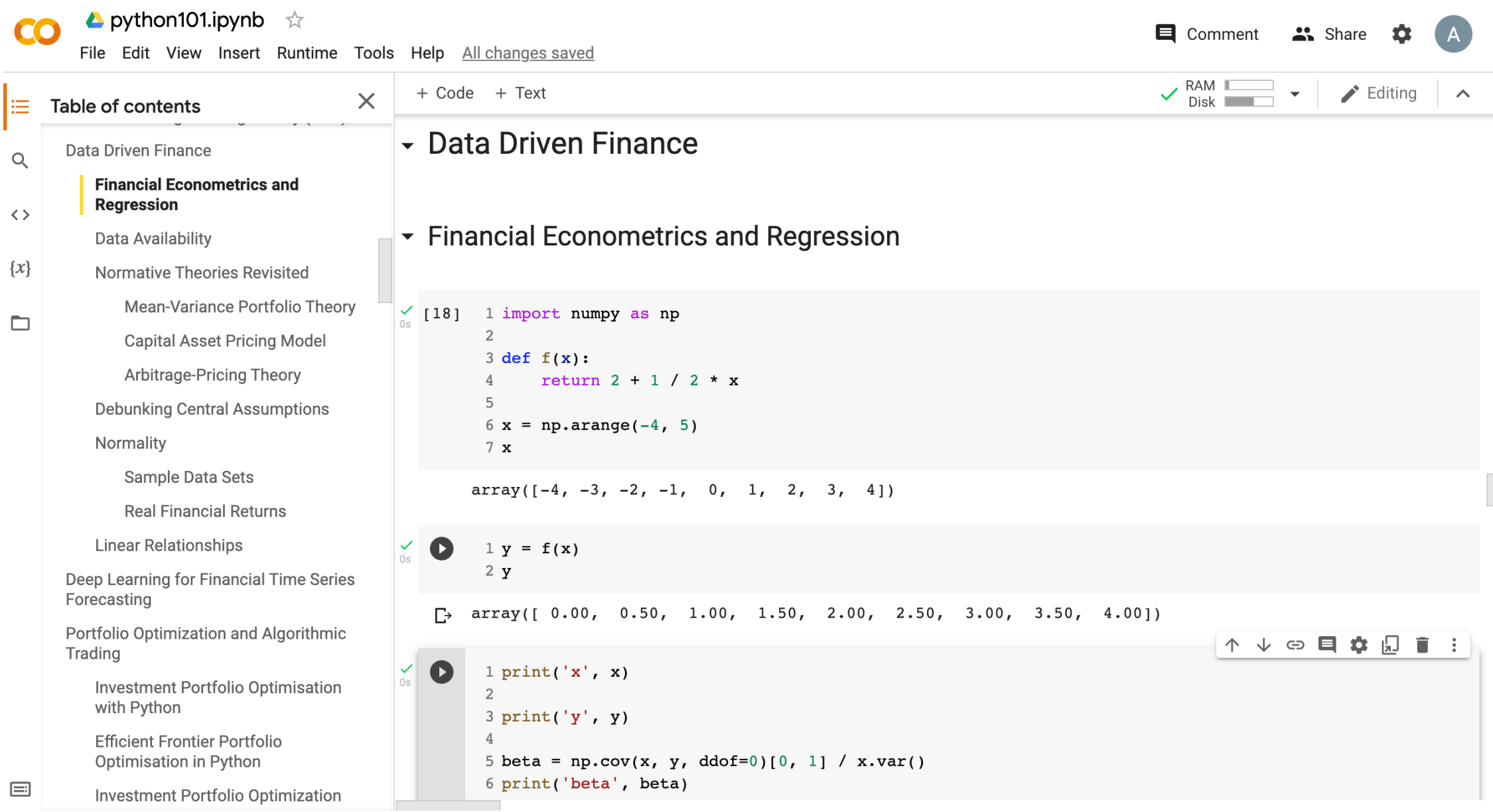
Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

Table of contents

- Financial Econometrics and Regression
- Data Availability
- Normative Theories Revisited
  - Mean-Variance Portfolio Theory
  - Capital Asset Pricing Model
  - Arbitrage-Pricing Theory
- Debunking Central Assumptions
- Normality
  - Sample Data Sets
  - Real Financial Returns
- Linear Relationships

Financial Econometrics and Machine Learning

- **Machine Learning**
- Data
- Success
- Capacity
- Evaluation
- Bias & Variance
- Cross-Validation

+ Code + Text RAM Disk Editing

Machine Learning

Data

```
1 import numpy as np
2 import pandas as pd
3 from pylab import plt, mpl
4 np.random.seed(100)
5 plt.style.use('seaborn')
6 mpl.rcParams['savefig.dpi'] = 300
7 mpl.rcParams['font.family'] = 'serif'
8
9 url = 'http://hilpisch.com/aiif_eikon_eod_data.csv'
10
11 raw = pd.read_csv(url, index_col=0, parse_dates=True)['EUR=']
12 raw.head()
```

```
Date
2010-01-01    1.4323
2010-01-04    1.4411
2010-01-05    1.4368
2010-01-06    1.4412
2010-01-07    1.4318
Name: EUR=, dtype: float64
```

```
[2]  1 raw.tail()
```

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

Table of contents

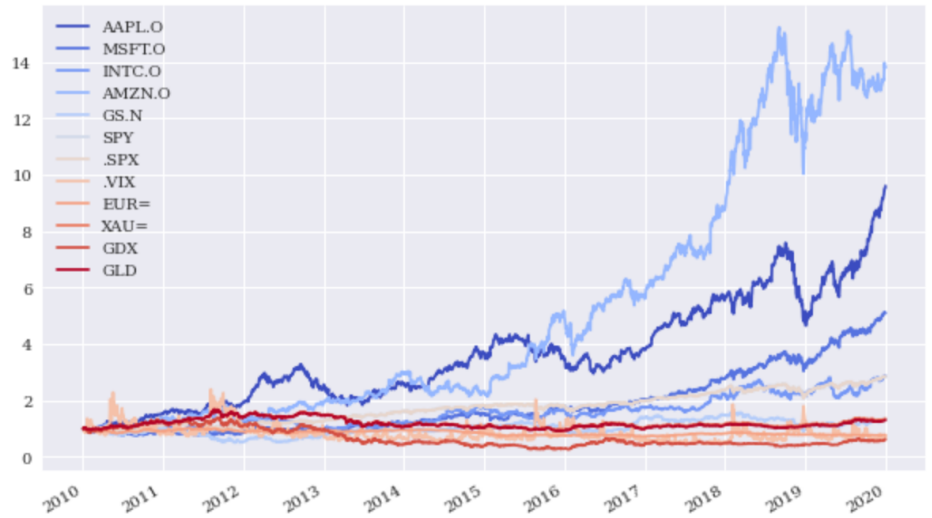
- Mean-Variance Portfolio Theory
- Capital Asset Pricing Model
- Arbitrage-Pricing Theory
- Debunking Central Assumptions
- Normality
- Sample Data Sets
- Real Financial Returns
- Linear Relationships
- Financial Econometrics and Machine Learning
- Machine Learning
- Data
- Success
- Capacity
- Evaluation
- Bias & Variance
- Cross-Validation
- AI-First Finance
- **Efficient Markets**
- Market Prediction Based on Returns Data
- Market Prediction With More Features
- Market Prediction Intraday

+ Code + Text RAM Disk Editing

Efficient Markets

```
1 import numpy as np
2 import pandas as pd
3 from pylab import plt, mpl
4 plt.style.use('seaborn')
5 mpl.rcParams['savefig.dpi'] = 300
6 mpl.rcParams['font.family'] = 'serif'
7 pd.set_option('precision', 4)
8 np.set_printoptions(suppress=True, precision=4)
9
10 url = 'http://hilpisch.com/aiif_eikon_eod_data.csv'
11 data = pd.read_csv(url, index_col=0, parse_dates=True).dropna()
12 (data / data.iloc[0]).plot(figsize=(10, 6), cmap='coolwarm')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f29f972f210>
```



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

Table of contents

- Deep Learning (DL) in Finance
  - Dense Neural Networks (DNN)
    - Baseline Prediction
    - Normalization
    - Dropout
    - Regularization
    - Bagging
    - Optimizers
  - Recurrent Neural Networks (RNN)
    - First Example
    - Second Example
    - Financial Price Series
    - Financial Return Series
    - Financial Features
    - Deep RNNs
  - Convolutional Neural Networks (CNN)
- Reinforcement Learning (RL) in Finance

+ Code + Text Connect Editing

Deep Learning (DL) in Finance

- Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.
- Github: https://github.com/yhilpisch/aiif/

Dense Neural Networks (DNN)

```
1 import os
2 import numpy as np
3 import pandas as pd
4 from pylab import plt, mpl
5 plt.style.use('seaborn')
6 mpl.rcParams['savefig.dpi'] = 300
7 mpl.rcParams['font.family'] = 'serif'
8 pd.set_option('precision', 4)
9 np.set_printoptions(suppress=True, precision=4)
10 os.environ['PYTHONHASHSEED'] = '0'
```

```
[ ] 1 url = 'http://hilpisch.com/aiif_eikon_id_eur_usd.csv'
    2 symbol = 'EUR_USD'
    3 raw = pd.read_csv(url, index_col=0, parse_dates=True)
    4 raw.head()
```

	HIGH	LOW	OPEN	CLOSE

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

Table of contents

Financial Features
Deep RNNs
Convolutional Neural Networks (CNN)
Reinforcement Learning (RL) in Finance
Reinforcement Learning (RL)
CartPole Environment
Dimensionality Reduction
Action Rule
Total Reward per Episode
Simple Learning
Testing the Results
DNN Learning
Q Learning
Finance Environment
Improved Finance Environment
Improved Financial QL Agent

+ Code + Text Connect Editing

Reinforcement Learning (RL) in Finance

- Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.
- Github: https://github.com/yhilpisch/aiif/

Reinforcement Learning (RL)

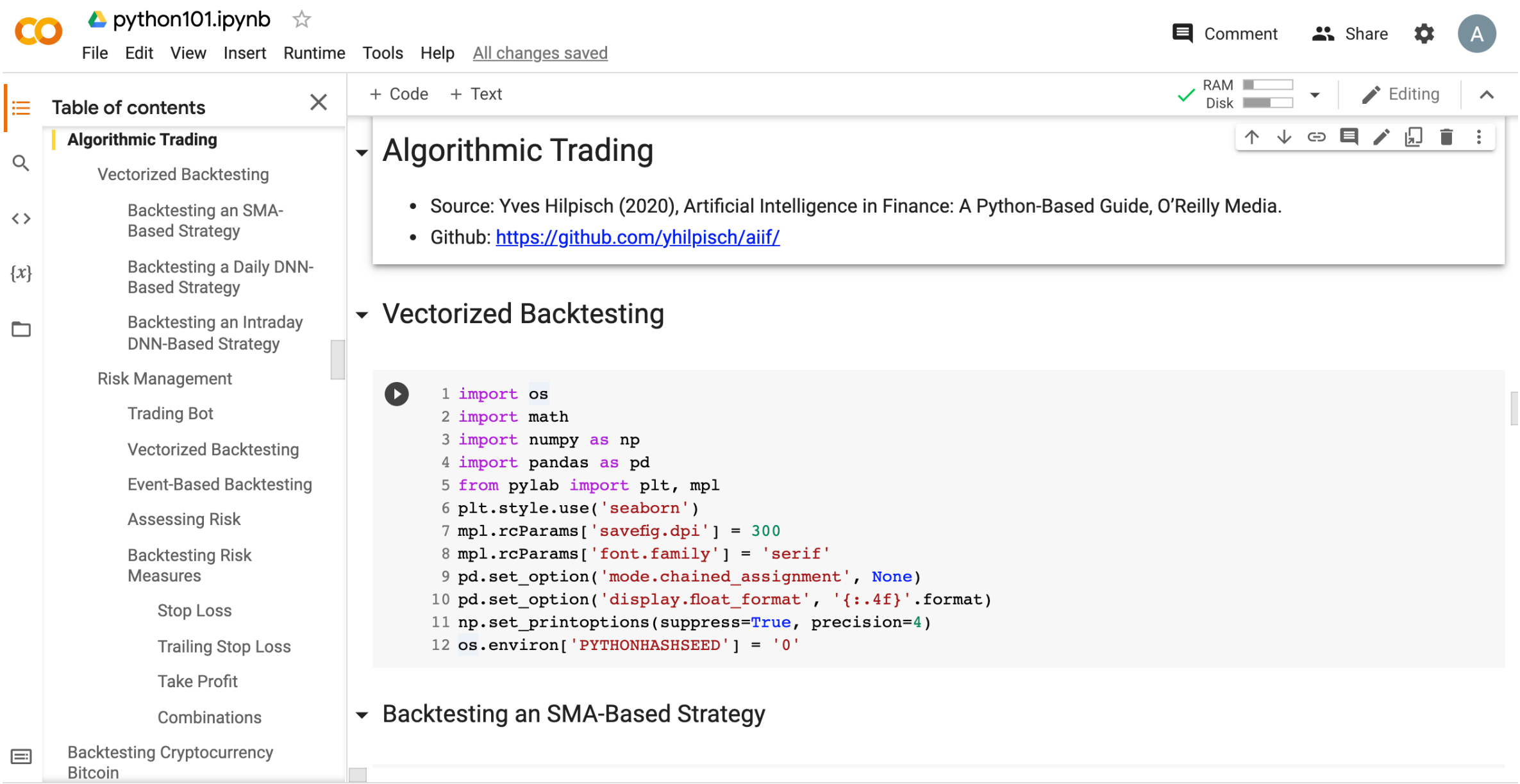
```
1 import os
2 import math
3 import random
4 import numpy as np
5 import pandas as pd
6 from pylab import plt, mpl
7 plt.style.use('seaborn')
8 mpl.rcParams['savefig.dpi'] = 300
9 mpl.rcParams['font.family'] = 'serif'
10 np.set_printoptions(precision=4, suppress=True)
11 os.environ['PYTHONHASHSEED'] = '0'
```

CartPole Environment

```
[ ]  1 import gym
     2
```

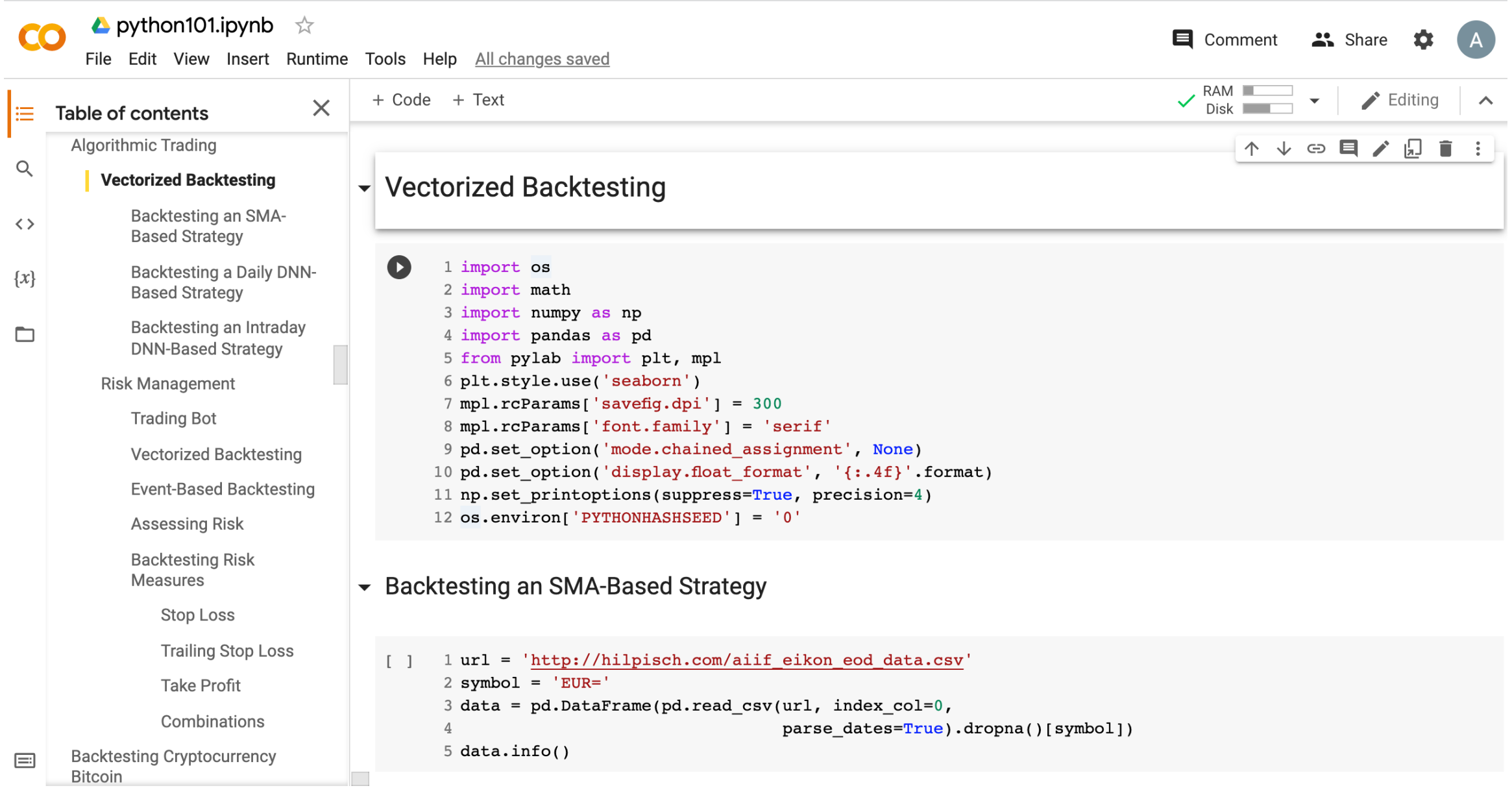
https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

Table of contents

- Algorithmic Trading
  - **Vectorized Backtesting**
    - Backtesting an SMA-Based Strategy
    - Backtesting a Daily DNN-Based Strategy
    - Backtesting an Intraday DNN-Based Strategy
  - Risk Management
    - Trading Bot
    - Vectorized Backtesting
    - Event-Based Backtesting
    - Assessing Risk
    - Backtesting Risk Measures
      - Stop Loss
      - Trailing Stop Loss
      - Take Profit
      - Combinations
- Backtesting Cryptocurrency Bitcoin

+ Code + Text

RAM Disk Editing

```
[ ]  1 data['r'] = np.log(data[symbol] / data[symbol].shift(1))
     2 data.dropna(inplace=True)
     3 data['s'] = data['p'] * data['r']
     4 data[['r', 's']].sum().apply(np.exp)  # gross performance
     5 data[['r', 's']].sum().apply(np.exp) - 1  # net performance
     6 data[['r', 's']].cumsum().apply(np.exp).plot(figsize=(10, 6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd9f404fed0>
```

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

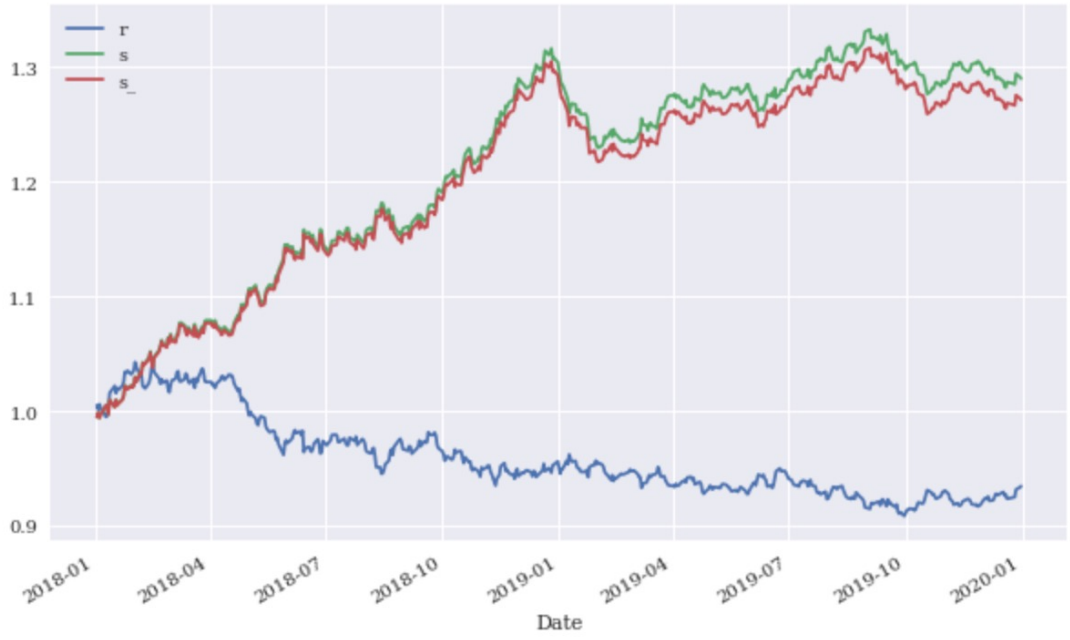
Table of contents

- Algorithmic Trading
  - Vectorized Backtesting
    - Backtesting an SMA-Based Strategy
    - **Backtesting a Daily DNN-Based Strategy**
    - Backtesting an Intraday DNN-Based Strategy
  - Risk Management
    - Trading Bot
    - Vectorized Backtesting
    - Event-Based Backtesting
    - Assessing Risk
    - Backtesting Risk Measures
      - Stop Loss
      - Trailing Stop Loss
      - Take Profit
      - Combinations
- Backtesting Cryptocurrency Bitcoin

+ Code + Text RAM Disk Editing

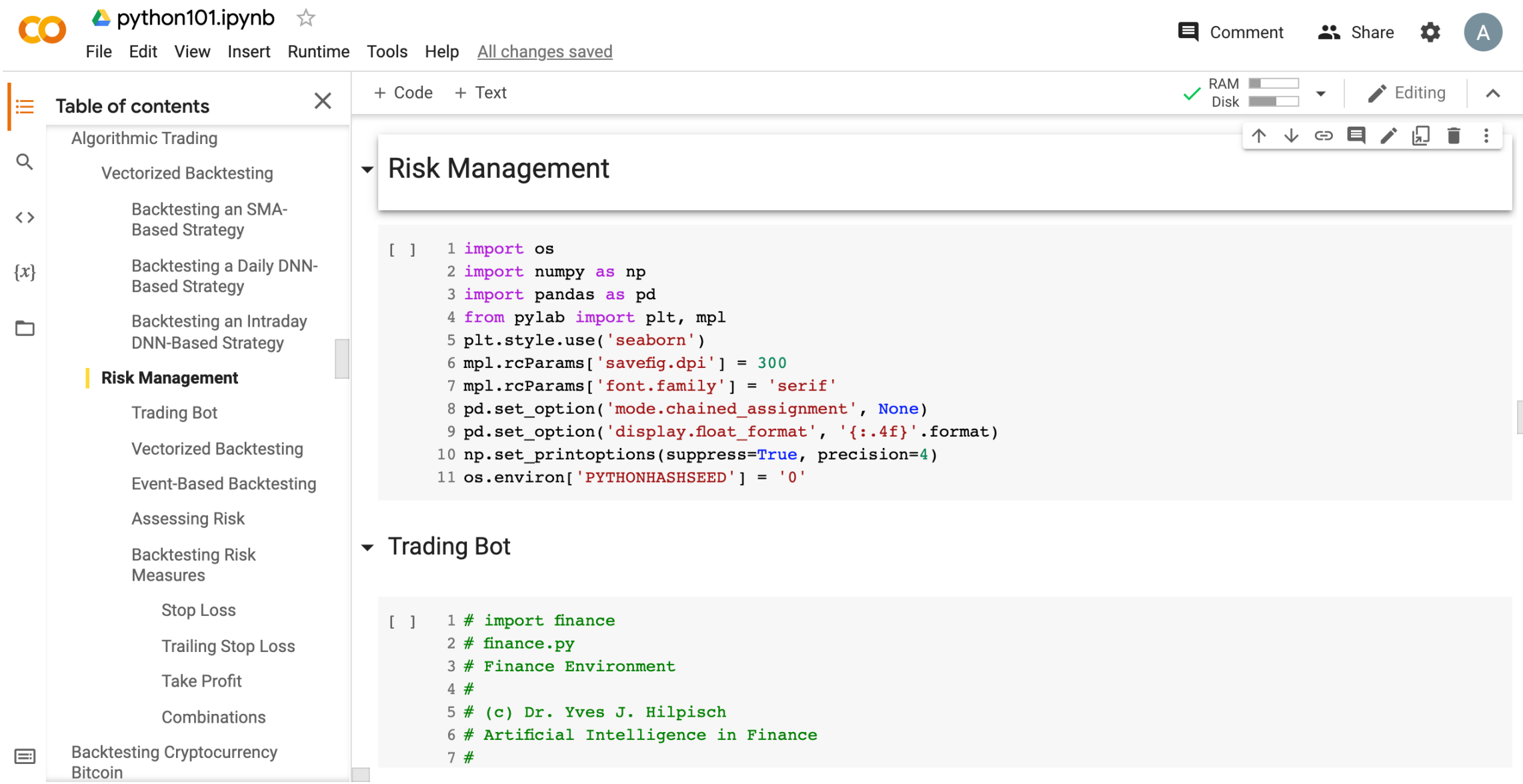
```
1 test['s_'] = np.where(test['p'].diff() != 0,
2                       test['s'] - pc, test['s'])
3 # test['s_'].iloc[0] -= pc
4 test['s_'].iloc[-1] -= pc
5 test[['r', 's', 's_']].sum().apply(np.exp)
6 test[['r', 's', 's_']].sum().apply(np.exp) - 1
7 test[['r', 's', 's_']].cumsum().apply(np.exp).plot(figsize=(10, 6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd901d89910>
```



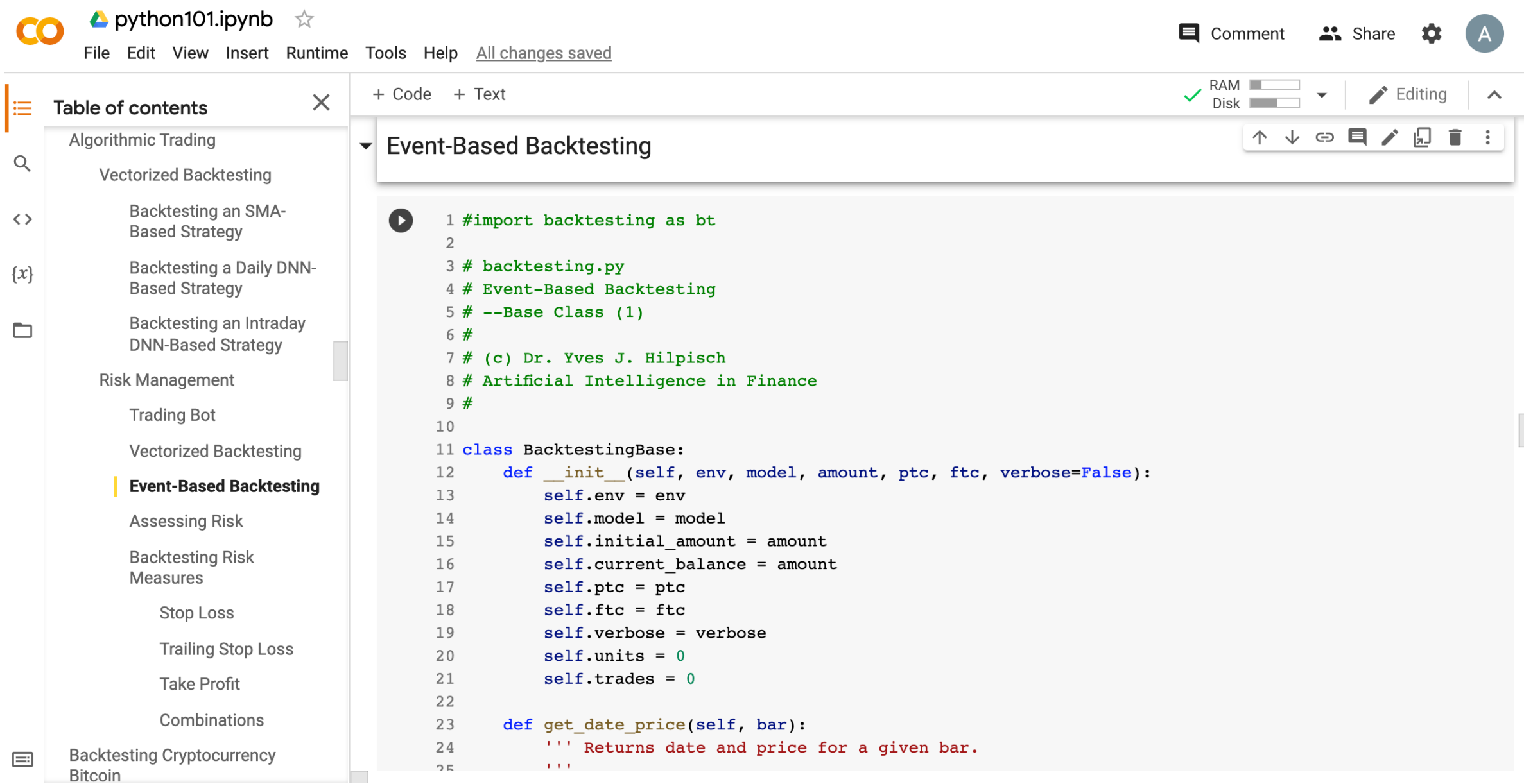
https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)



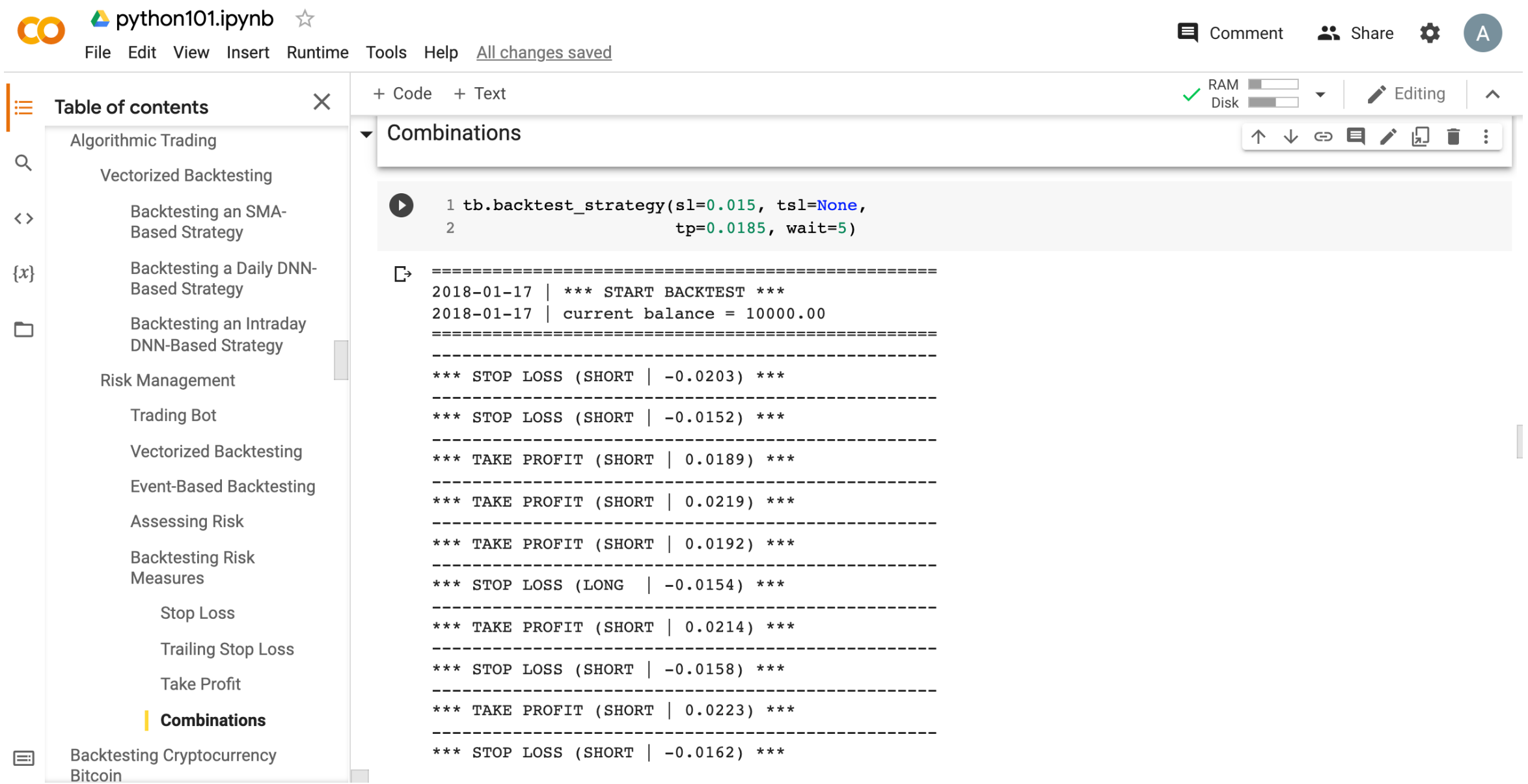
https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

Table of contents

- Algorithmic Trading
  - Vectorized Backtesting
    - Backtesting an SMA-Based Strategy
    - Backtesting a Daily DNN-Based Strategy
    - Backtesting an Intraday DNN-Based Strategy
  - Risk Management
    - Trading Bot
    - Vectorized Backtesting
    - Event-Based Backtesting
    - Assessing Risk
    - Backtesting Risk Measures
      - Stop Loss
      - Trailing Stop Loss
      - Take Profit
      - Combinations
- **Backtesting Cryptocurrency Bitcoin**

+ Code + Text RAM Disk Editing

Backtesting Cryptocurrency Bitcoin

- Financial Functions (ffn): https://pmorissette.github.io/ffn/
- backtesting.py: https://kernc.github.io/backtesting.py/

15s

```
1 !pip install ffn
2 import ffn
3 import plotly.express as px
4 %pylab inline
5 #BTC-USD Bitcoin USD
6 df = ffn.get('btc-usd', start='2016-01-01', end='2021-12-31')
7 print('df')
8 print(df.head())
9 print(df.tail())
10 print(df.describe())
11 df.plot(figsize=(14,10))
12
13 returns = df.to_returns().dropna()
14 print('returns')
15 print(returns.head())
16 print(returns.tail())
17 print(returns.describe())
18 #ax = df.plot(figsize=(12,9))
19
20 perf = df.calc_stats()
21 perf.plot(figsize=(14, 10))
```

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

Table of contents

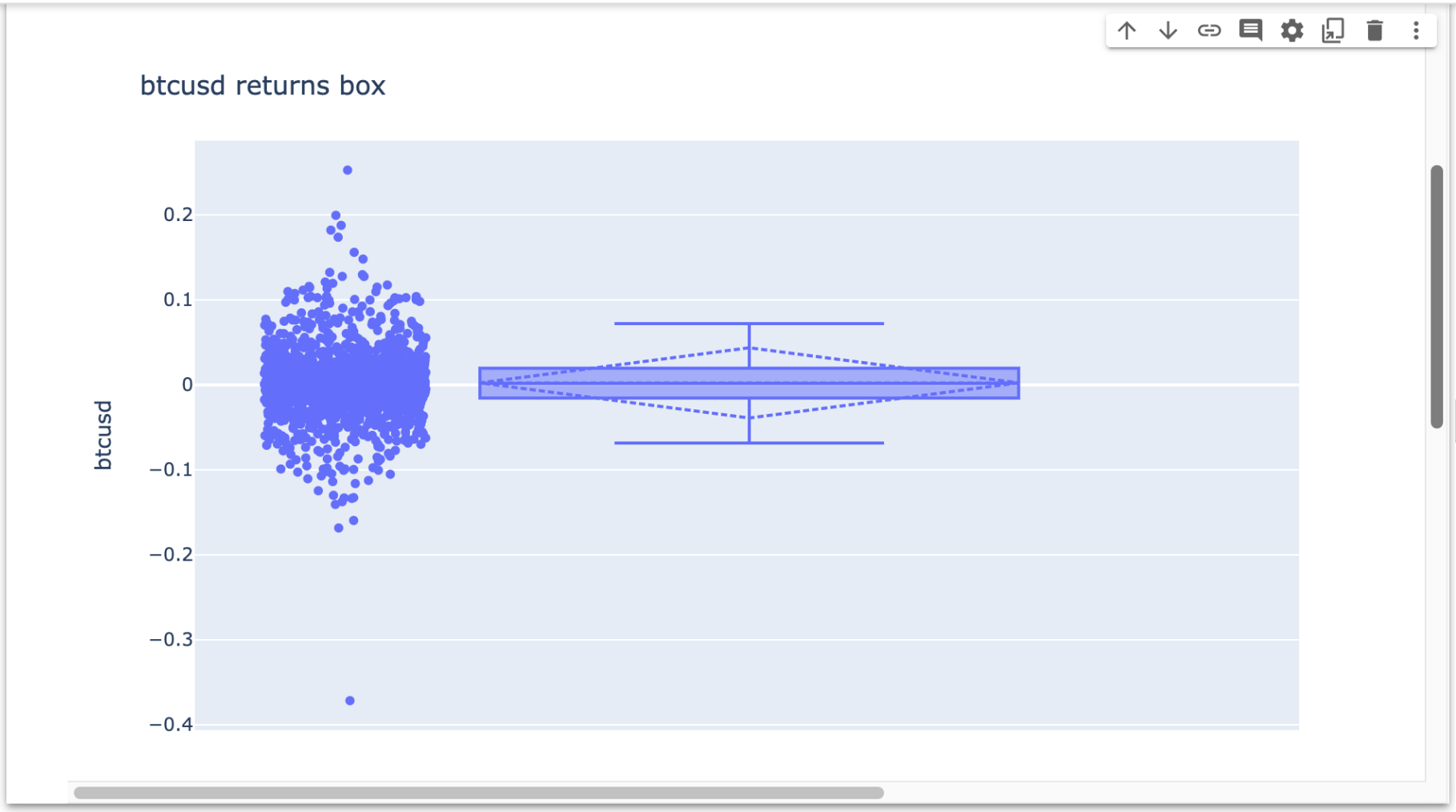
- Algorithmic Trading
  - Vectorized Backtesting
    - Backtesting an SMA-Based Strategy
    - Backtesting a Daily DNN-Based Strategy
    - Backtesting an Intraday DNN-Based Strategy
  - Risk Management
    - Trading Bot
    - Vectorized Backtesting
    - Event-Based Backtesting
    - Assessing Risk
    - Backtesting Risk Measures
      - Stop Loss
      - Trailing Stop Loss
      - Take Profit
      - Combinations

Backtesting Cryptocurrency Bitcoin

+ Code + Text

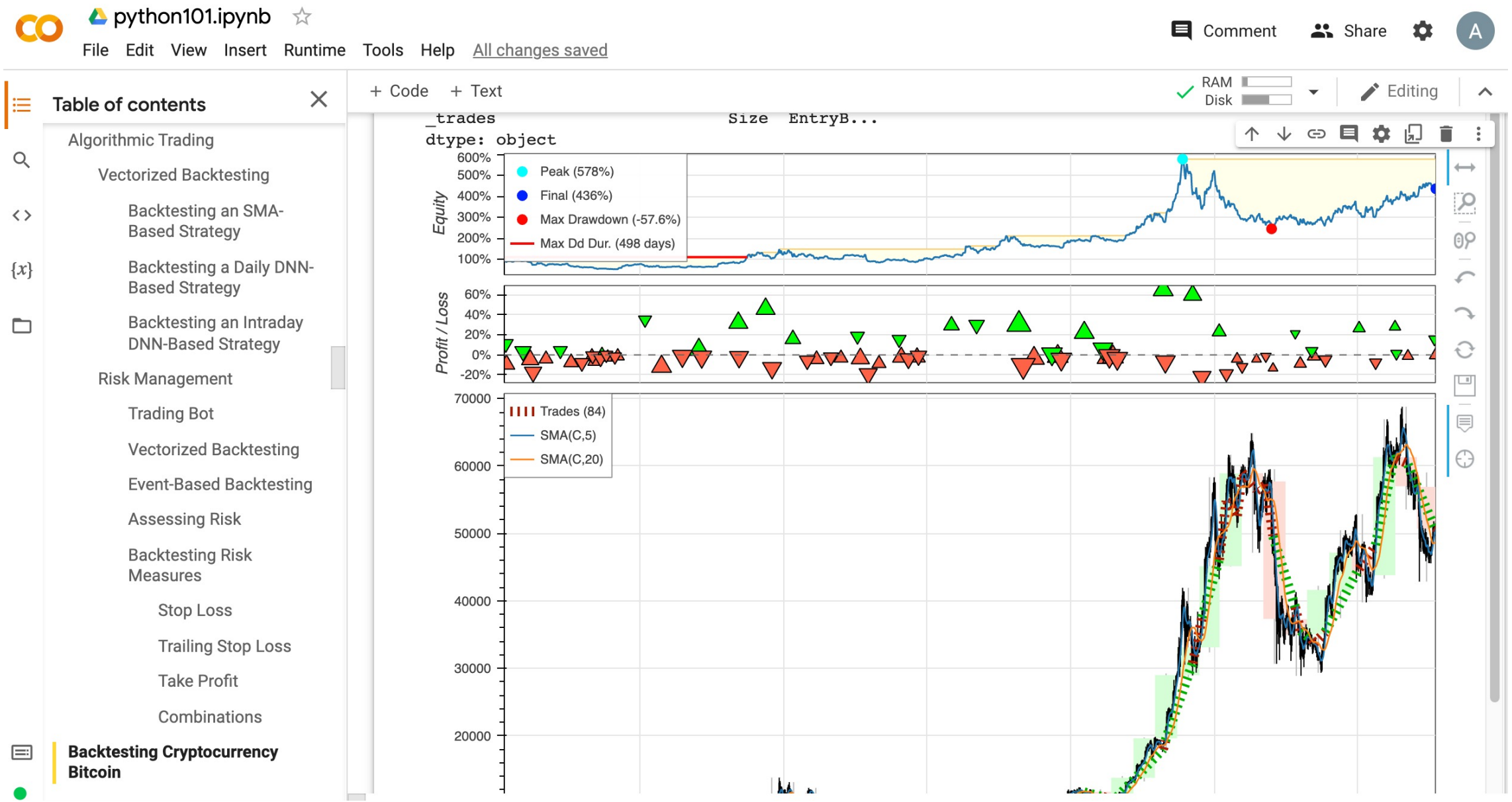
RAM Disk

Editing

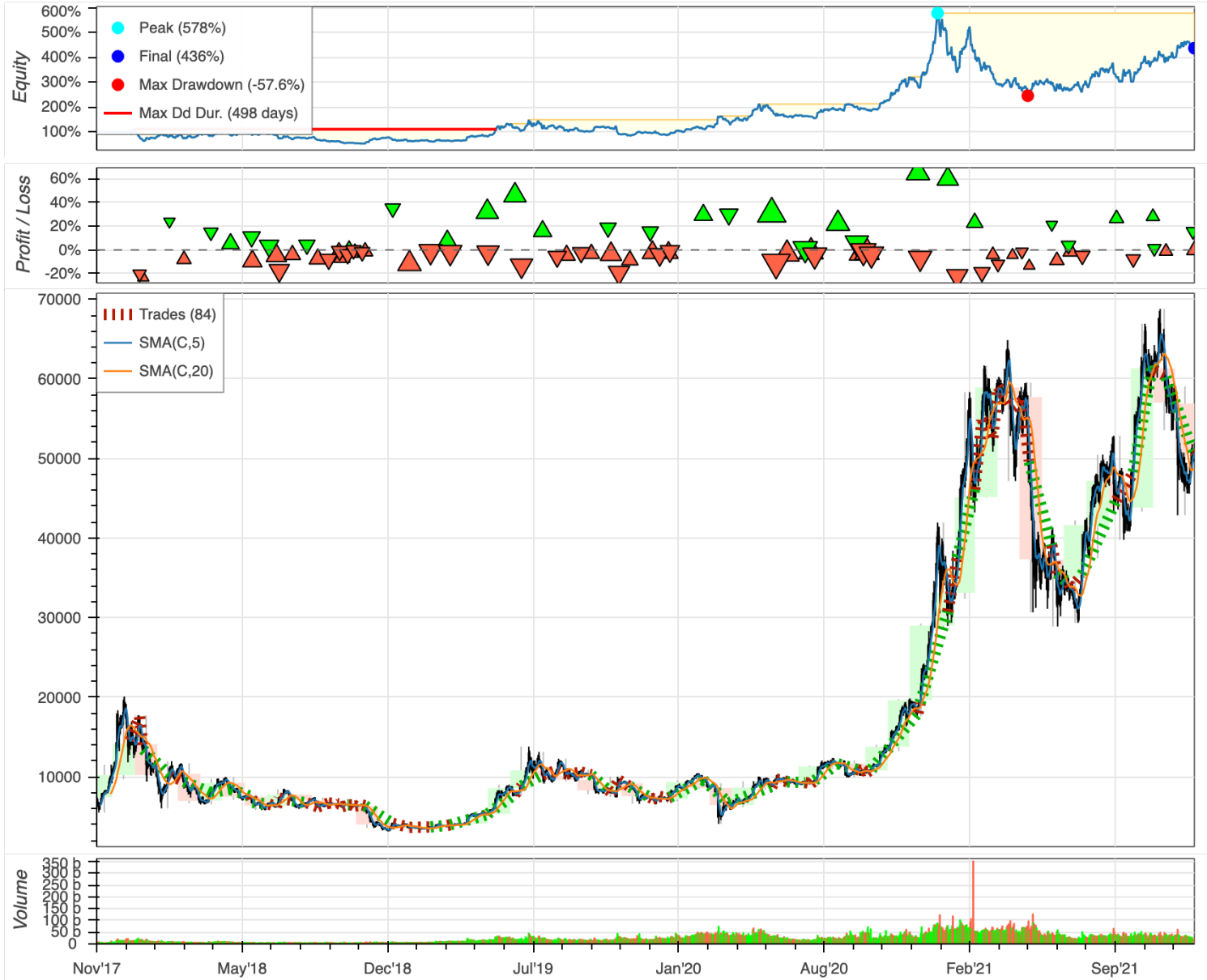


https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)



https://tinyurl.com/aintpupython101



https://tinyurl.com/aintpupython101

Summary

- **Algorithmic Trading**
- **Risk Management**
- **Trading Bot**
- **Event-Based Backtesting**

Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.

References

- Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media, https://github.com/yhilpisch/aiif .
- Yves Hilpisch (2020), Python for Algorithmic Trading: From Idea to Cloud Deployment, O'Reilly Media.
- Stefan Jansen (2020), Machine Learning for Algorithmic Trading: Predictive models to extract signals from market and alternative data for systematic trading strategies with Python, 2nd Edition, Packt Publishing.
- Aurélien Géron (2022), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 3rd Edition, O'Reilly Media.
- Hariom Tatsat, Sahil Puri, Brad Lookabaugh (2020), Machine Learning and Data Science Blueprints for Finance: From Building Trading Strategies to Robo-Advisors Using Python, O'Reilly Media
- Chris Kelliher (2022), Quantitative Finance With Python: A Practical Guide to Investment Management, Trading, and Financial Engineering, Chapman and Hall/CRC.
- Abdullah Karasan (2021), Machine Learning for Financial Risk Management with Python: Algorithms for Modeling Risk, O'Reilly Media.
- Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer (2020). "Deep learning for financial applications: A survey." Applied Soft Computing (2020): 106384.
- Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu (2020), "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019." Applied Soft Computing 90 (2020): 106181.
- Yuanhang Zheng, Zeshui Xu, and Anran Xiao (2023). "Deep learning in economics: a systematic and critical review." Artificial Intelligence Review (2023): 1-43.
- Ajitha Kumari Vijayappan Nair Biju, Ann Susan Thomas, and J. Thasneem (2023). "Examining the research taxonomy of artificial intelligence, deep learning & machine learning in the financial sphere—a bibliometric analysis." Quality & Quantity (2023): 1-30.
- Min-Yuh Day, Ching-Ying Yang, and Yensen Ni (2023), "Portfolio dynamic trading strategies using deep reinforcement learning." Soft Computing (2023): 1-16.
- Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer (2020), "Deep learning for financial applications: A survey." Applied Soft Computing (2020): 106384.
- Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu (2020), "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019." Applied Soft Computing 90 (2020): 106181.
- Fan Fang, Carmine Ventre, Michail Basios, Leslie Kanthan, David Martinez-Rego, Fan Wu, and Lingbo Li. (2023) "Cryptocurrency trading: a comprehensive survey." Financial Innovation 8, no. 1 (2022): 1-59.
- Min-Yuh Day (2023), Python 101, https://tinyurl.com/aintpupython101