

## Algorithmic Trading, Risk Management, Trading Bot and Event-Based Backtesting

1111AIFQA10

MBA, IM, NTPU (M6132) (Fall 2022)

Tue 2, 3, 4 (9:10-12:00) (B8F40)



<https://meet.google.com/paj-zhji-mya>



Min-Yuh Day, Ph.D,  
Associate Professor

Institute of Information Management, National Taipei University

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



# Syllabus

<b>Week</b>	<b>Date</b>	<b>Subject/Topics</b>
<b>1</b>	<b>2022/09/13</b>	<b>Introduction to Artificial Intelligence in Finance and Quantitative Analysis</b>
<b>2</b>	<b>2022/09/20</b>	<b>AI in FinTech: Metaverse, Web3, DeFi, NFT, Financial Services Innovation and Applications</b>
<b>3</b>	<b>2022/09/27</b>	<b>Investing Psychology and Behavioral Finance</b>
<b>4</b>	<b>2022/10/04</b>	<b>Event Studies in Finance</b>
<b>5</b>	<b>2022/10/11</b>	<b>Case Study on AI in Finance and Quantitative Analysis I</b>
<b>6</b>	<b>2022/10/18</b>	<b>Finance Theory</b>

# Syllabus

Week	Date	Subject/Topics
7	2022/10/25	Data-Driven Finance
8	2022/11/01	Midterm Project Report
9	2022/11/08	Financial Econometrics and Machine Learning
10	2022/11/15	AI-First Finance
11	2022/11/22	Deep Learning in Finance; Reinforcement Learning in Finance
12	2022/11/29	Case Study on AI in Finance and Quantitative Analysis II

# Syllabus

Week	Date	Subject/Topics
13	2022/12/06	Industry Practices of AI in Finance and Quantitative Analysis
14	2022/12/13	Algorithmic Trading; Risk Management; Trading Bot and Event-Based Backtesting
15	2022/12/20	Final Project Report I
16	2022/12/27	Final Project Report II
17	2023/01/03	Self-learning
18	2023/01/10	Self-learning

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

# Outline

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

# **Deep learning for financial applications: A survey**

## **Applied Soft Computing (2020)**

Source:

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

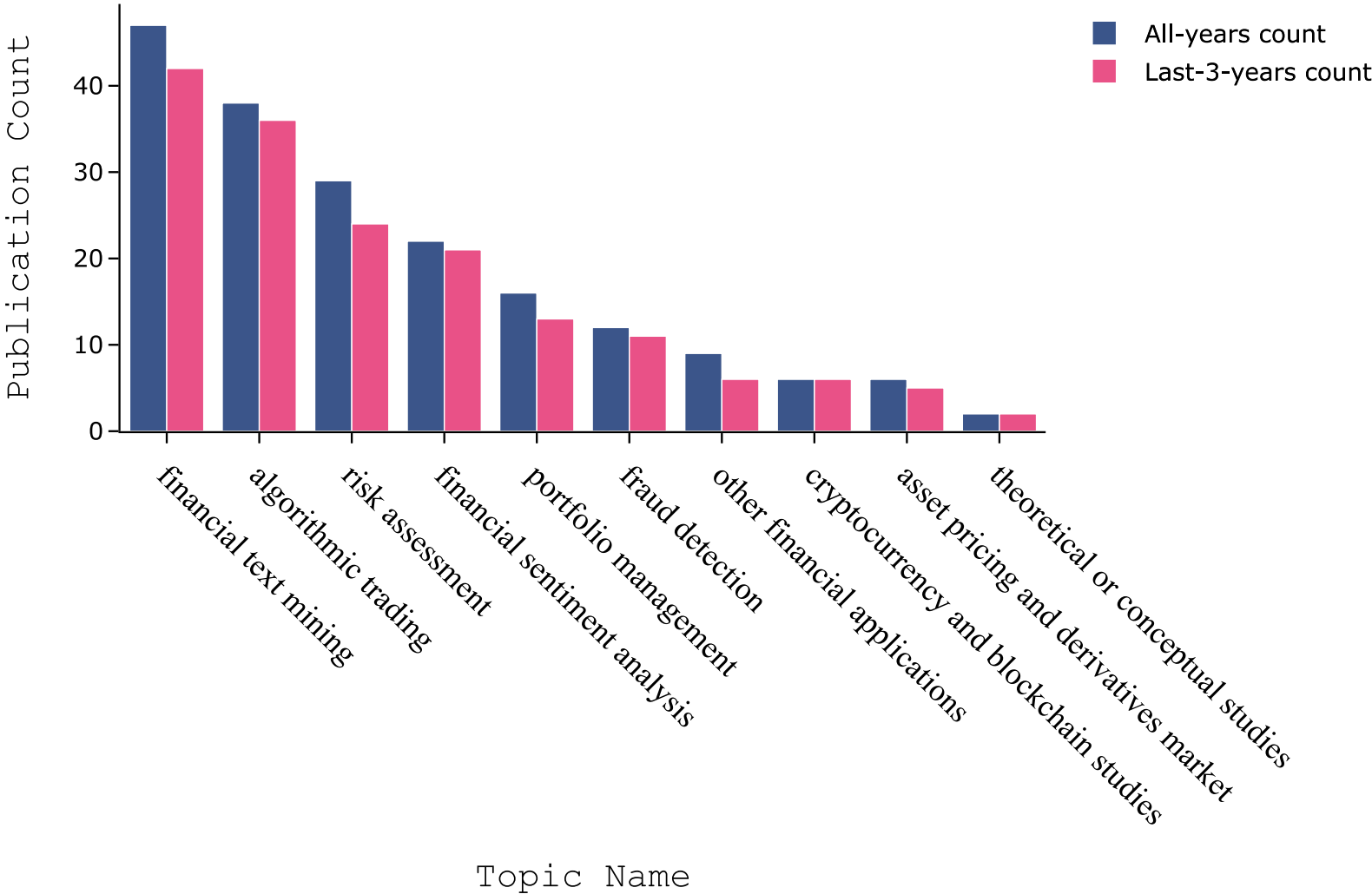
**Financial  
time series forecasting with  
deep learning:  
A systematic literature review:  
2005–2019  
Applied Soft Computing (2020)**

Source:

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.

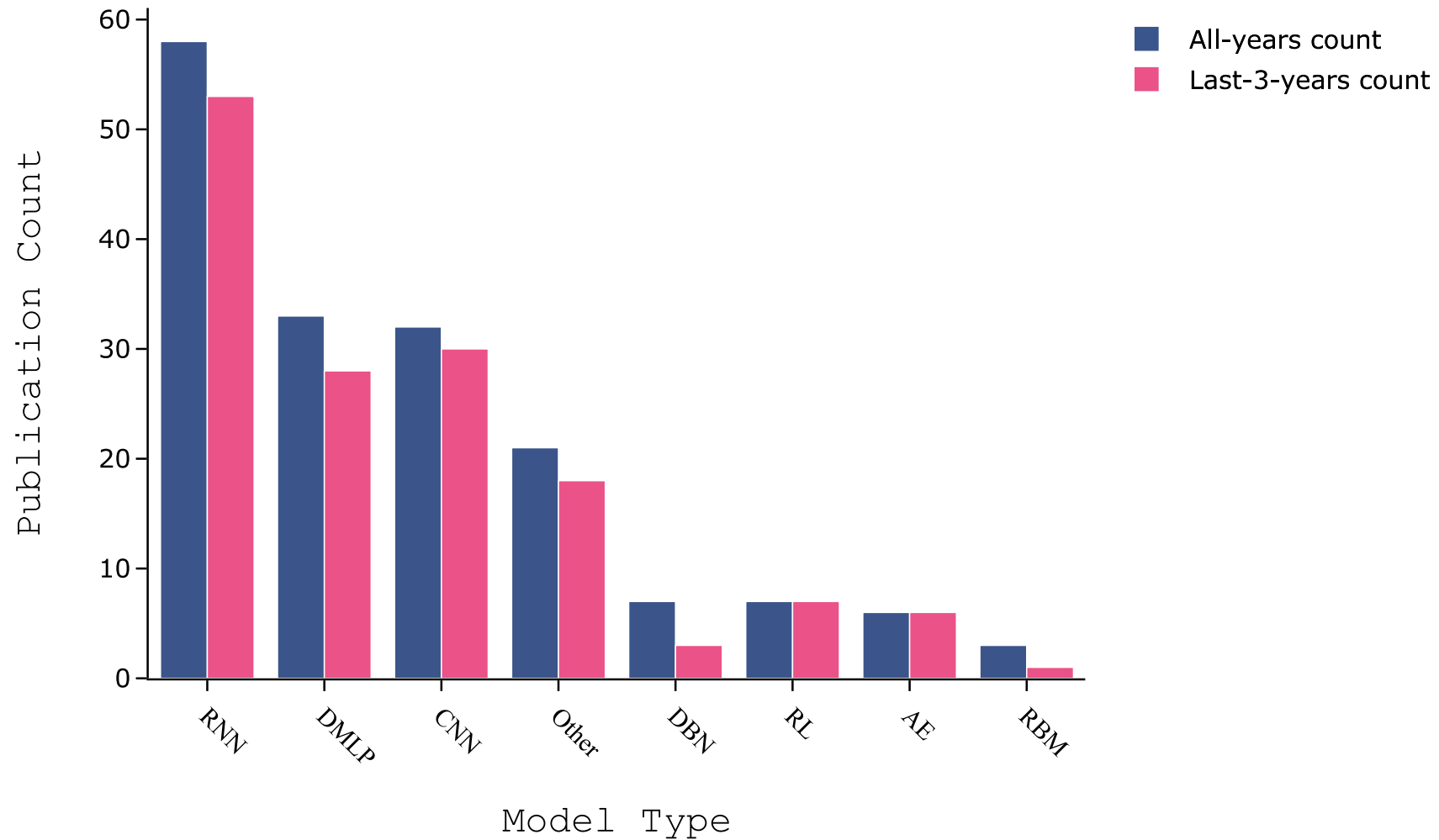


# Deep learning for financial applications: Topics

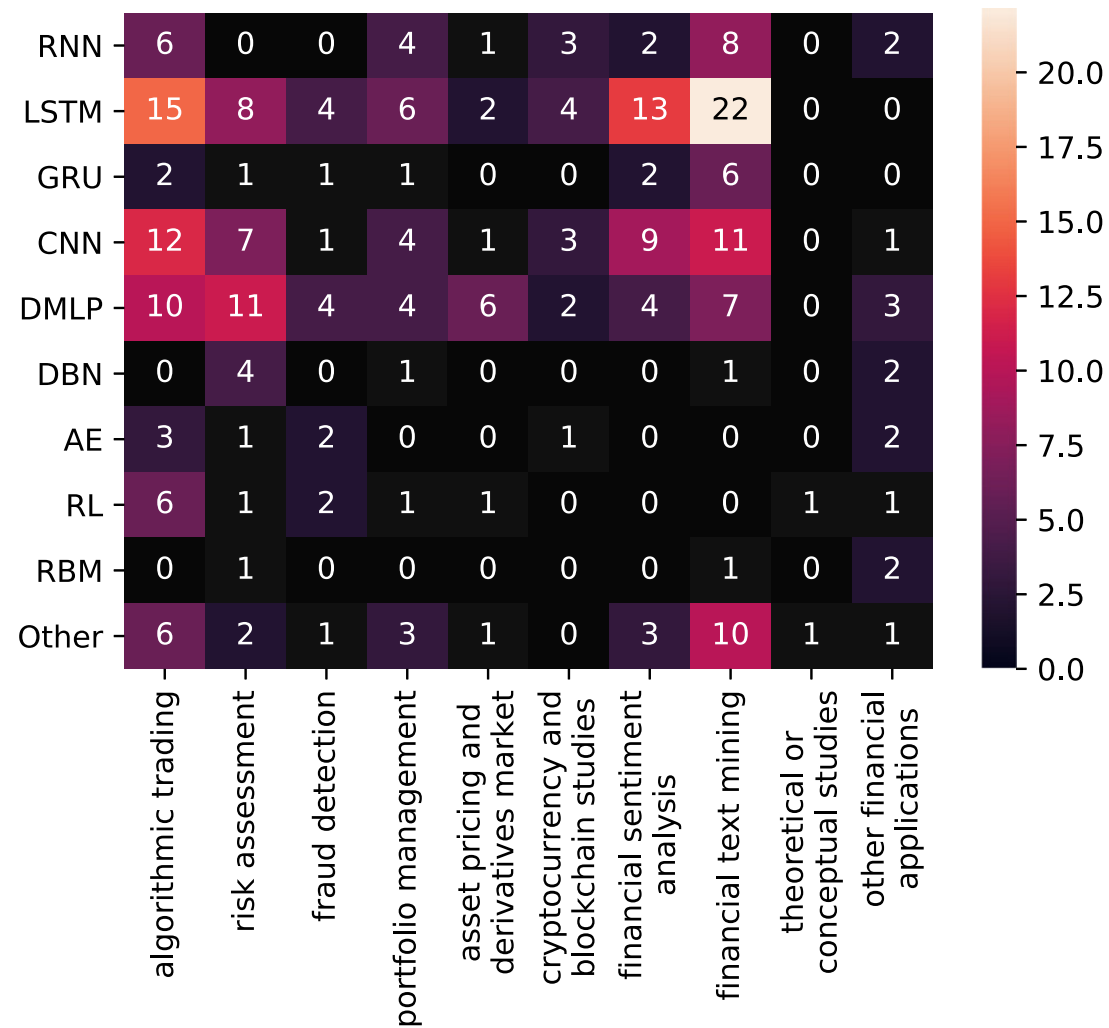


Source: Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer (2020). "Deep learning for financial applications: A survey." Applied Soft Computing (2020): 106384.

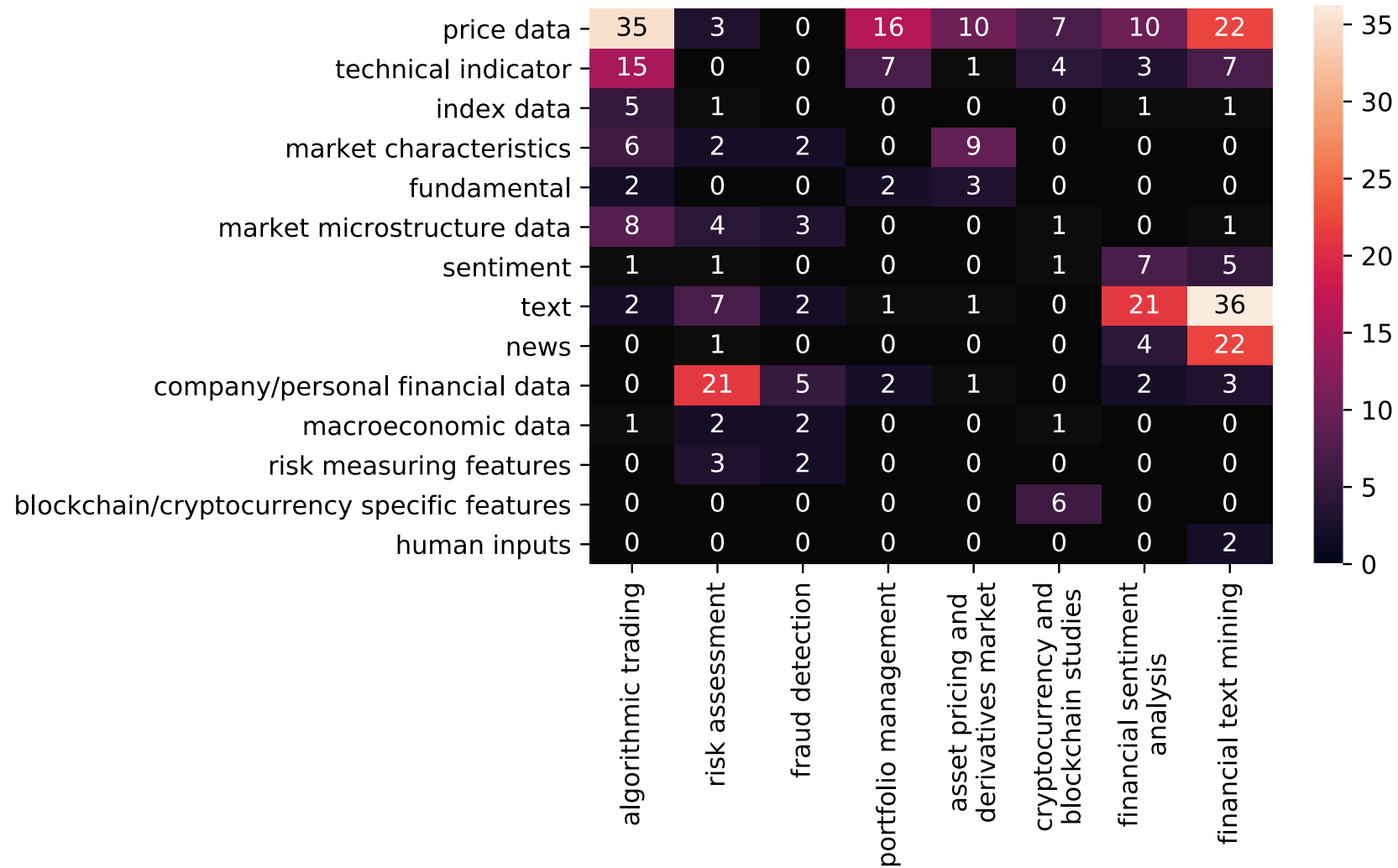
# Deep learning for financial applications: Deep Learning Models



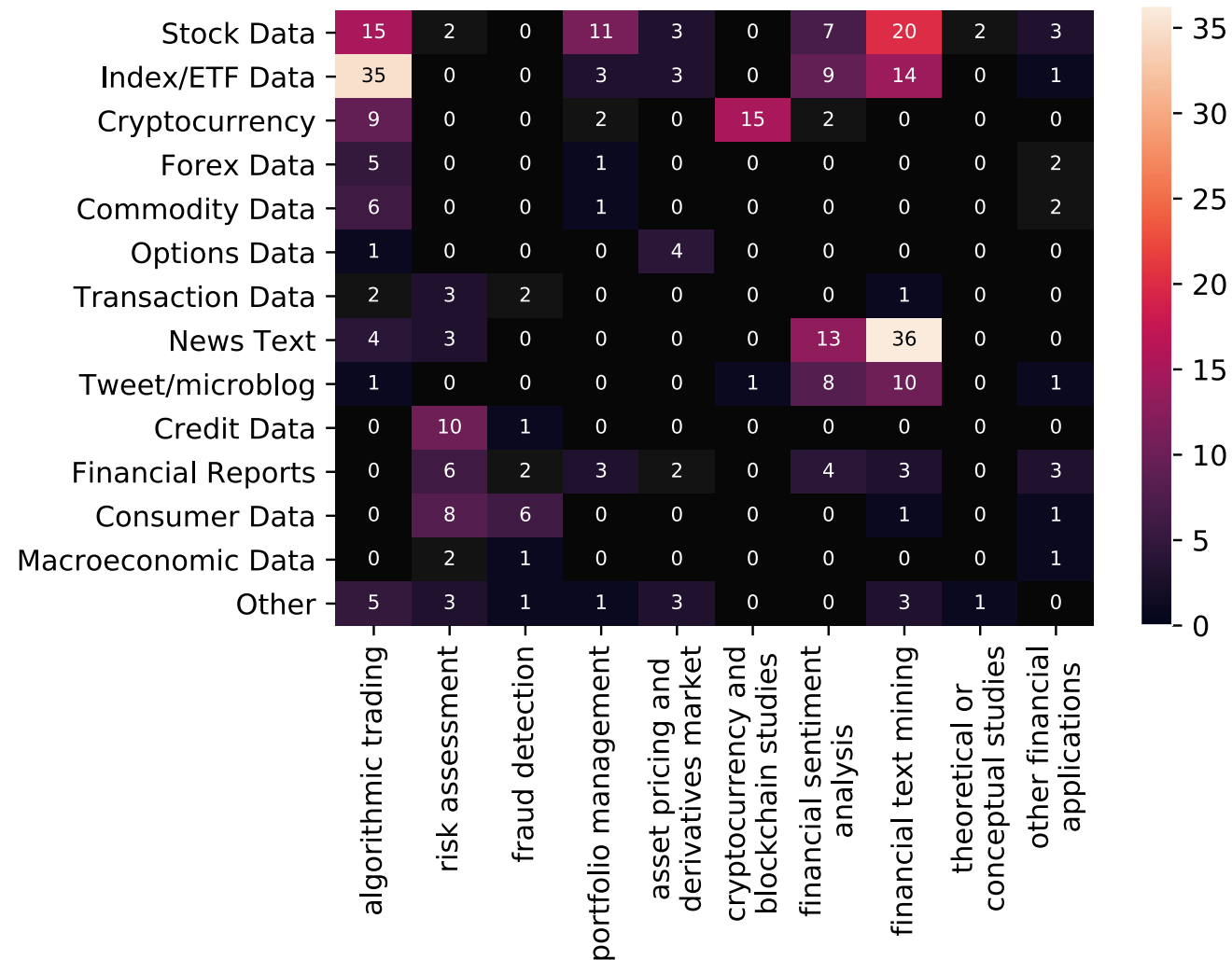
# Deep learning for financial applications: Topic-Model Heatmap



# Deep learning for financial applications: Topic-Feature Heatmap



# Deep learning for financial applications: Topic-Dataset Heatmap



# 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	–	–

# Deep learning for financial applications:

## Classification (buy–sell signal, or trend detection) based algo-trading models

Art.	Data set	Period	Feature set	Method	Performance criteria	Environment
[51]	Stocks in Dow30	1997–2017	RSI	DMLP with genetic algorithm	Annualized return	Spark MLlib, Java
[52]	SPY ETF, 10 stocks from S&P500	2014–2016	Price data	FFNN	Cumulative gain	MatConvNet, Matlab
[53]	Dow30 stocks	2012–2016	Close data and several technical indicators	LSTM	Accuracy	Python, Keras, Tensorflow, TALIB
[54]	High-frequency record of all orders	2014–2017	Price data, record of all orders, transactions	LSTM	Accuracy	–
[55]	Nasdaq Nordic (Kesko Oyj, Outokumpu Oyj, Sampo, Rautaruukki, Wartsila Oyj)	2010	Price and volume data in LOB	LSTM	Precision, Recall, F1-score, Cohen's k	–
[56]	17 ETFs	2000–2016	Price data, technical indicators	CNN	Accuracy, MSE, Profit, AUROC	Keras, Tensorflow
[57]	Stocks in Dow30 and 9 Top Volume ETFs	1997–2017	Price data, technical indicators	CNN with feature imaging	Recall, precision, F1-score, annualized return	Python, Keras, Tensorflow, Java
[58]	FTSE100	2000–2017	Price data	CAE	TR, SR, MDD, mean return	–
[59]	Nasdaq Nordic (Kesko Oyj, Outokumpu Oyj, Sampo, Rautaruukki, Wartsila Oyj)	2010	Price, Volume data, 10 orders of the LOB	CNN	Precision, Recall, F1-score, Cohen's k	Theano, Scikit learn, Python
[60]	Borsa Istanbul 100 Stocks	2011–2015	75 technical indicators and OCHLV	CNN	Accuracy	Keras
[61]	ETFs and Dow30	1997–2007	Price data	CNN with feature imaging	Annualized return	Keras, Tensorflow
[62]	8 experimental assets from bond/derivative market	–	Asset prices data	RL, DMLP, Genetic Algorithm	Learning and genetic algorithm error	–
[63]	10 stocks from S&P500	–	Stock Prices	TDNN, RNN, PNN	Missed opportunities, false alarms ratio	–
[64]	London Stock Exchange	2007–2008	Limit order book state, trades, buy/sell orders, order deletions	CNN	Accuracy, kappa	Caffe
[65]	Cryptocurrencies, Bitcoin	2014–2017	Price data	CNN, RNN, LSTM	Accumulative portfolio value, MDD, SR	–

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:

## Stand-alone and/or other algorithmic models

Art.	Data set	Period	Feature set	Method	Performance criteria	Environment
[66]	DAX, FTSE100, call/put options	1991–1998	Price data	Markov model, RNN	Ewa-measure, iv, daily profits' mean and std	–
[67]	Taiwan Stock Index Futures, Mini Index Futures	2012–2014	Price data to image	Visualization method + CNN	Accumulated profits, accuracy	–
[68]	Energy-Sector/ Company-Centric Tweets in S&P500	2015–2016	Text and Price data	LSTM, RNN, GRU	Return, SR, precision, recall, accuracy	Python, Tweepy API
[69]	CME FIX message	2016	Limit order book, time-stamp, price data	RNN	Precision, recall, F1-measure	Python, TensorFlow, R
[70]	Taiwan stock index futures (TAIFEX)	2017	Price data	Agent based RL with CNN pre-trained	Accuracy	–
[71]	Stocks from S&P500	2010–2016	OCHLV	DCNL	PCC, DTW, VWL	Pytorch
[72]	News from NowNews, AppleDaily, LTN, MoneyDJ for 18 stocks	2013–2014	Text, Sentiment	DMLP	Return	Python, Tensorflow
[73]	489 stocks from S&P500 and NASDAQ-100	2014–2015	Limit Order Book	Spatial neural network	Cross entropy error	NVIDIA's cuDNN
[74]	Experimental dataset	–	Price data	DRL with CNN, LSTM, GRU, DMLP	Mean profit	Python

# Deep learning for financial applications: Credit scoring or classification studies

Art.	Data set	Period	Feature set	Method	Performance criteria	Env.
[77]	The XR 14 CDS contracts	2016	Recovery rate, spreads, sector and region	DBN+RBM	AUROC, FN, FP, Accuracy	WEKA
[78]	German, Japanese credit datasets	–	Personal financial variables	SVM + DBN	Weighted-accuracy, TP, TN	–
[79]	Credit data from Kaggle	–	Personal financial variables	DMLP	Accuracy, TP, TN, G-mean	–
[80]	Australian, German credit data	–	Personal financial variables	GP + AE as Boosted DMLP	FP	Python, Scikit-learn
[81]	German, Australian credit dataset	–	Personal financial variables	DCNN, DMLP	Accuracy, False/Missed alarm	–
[82]	Consumer credit data from Chinese finance company	–	Relief algorithm chose the 50 most important features	CNN + Relief	AUROC, K-s statistic, Accuracy	Keras
[83]	Credit approval dataset by UCI Machine Learning repo	–	UCI credit approval dataset	Rectifier, Tanh, Maxout DL	–	AWS EC2, H2O, R

# Deep learning for financial applications:

## Financial distress, bankruptcy, bank risk, mortgage risk, crisis forecasting studies.

Art.	Data set	Period	Feature set	Method	Performance criteria	Env.
[84]	966 french firms	-	Financial ratios	RBM+SVM	Precision, Recall	-
[85]	883 BHC from EDGAR	2006-2017	Tokens, weighted sentiment polarity, leverage and ROA	CNN, LSTM, SVM, RF	Accuracy, Precision, Recall, F1-score	Keras, Python, Scikit-learn
[86]	The event data set for large European banks, news articles from Reuters	2007-2014	Word, sentence	DMLP +NLP preprocess	Relative usefulness, F1-score	-
[87]	Event dataset on European banks, news from Reuters	2007-2014	Text, sentence	Sentence vector + DFFN	Usefulness, F1-score, AUROC	-
[88]	News from Reuters, fundamental data	2007-2014	Financial ratios and news text	doc2vec + NN	Relative usefulness	Doc2vec
[89]	Macro/Micro economic variables, Bank characteristics/performance variables from BHC	1976-2017	Macro economic variables and bank performances	CGAN, MVN, MV-t, LSTM, VAR, FE-QAR	RMSE, Log likelihood, Loan loss rate	-
[90]	Financial statements of French companies	2002-2006	Financial ratios	DBN	Recall, Precision, F1-score, FP, FN	-
[91]	Stock returns of American publicly-traded companies from CRSP	2001-2011	Price data	DBN	Accuracy	Python, Theano
[92]	Financial statements of several companies from Japanese stock market	2002-2016	Financial ratios	CNN	F1-score, AUROC	-
[93]	Mortgage dataset with local and national economic factors	1995-2014	Mortgage related features	DMLP	Negative average log-likelihood	AWS
[94]	Mortgage data from Norwegian financial service group, DNB	2012-2016	Personal financial variables	CNN	Accuracy, Sensitivity, Specificity, AUROC	-
[95]	Private brokerage company's real data of risky transactions	-	250 features: order details, etc.	CNN, LSTM	F1-Score	Keras, Tensorflow
[96]	Several datasets combined to create a new one	1996-2017	Index data, 10-year Bond yield, exchange rates,	Logit, CART, RF, SVM, NN, XGBoost, DMLP	AUROC, KS, G-mean, likelihood ratio, DP, BA, WBA	R

# Deep learning for financial applications:

## Fraud detection studies

Art.	Data set	Period	Feature set	Method	Performance criteria	Env.
[114]	Debit card transactions by a local Indonesia bank	2016–2017	Financial transaction amount on several time periods	CNN, Stacked-LSTM, CNN-LSTM	AUROC	–
[115]	Credit card transactions from retail banking	2017	Transaction variables and several derived features	LSTM, GRU	Accuracy	Keras
[116]	Card purchases' transactions	2014–2015	Probability of fraud per currency/origin country, other fraud related features	DMLP	AUROC	–
[117]	Transactions made with credit cards by European cardholders	2013	Personal financial variables to PCA	DMLP, RF	Recall, Precision, Accuracy	–
[118]	Credit-card transactions	2015	Transaction and bank features	LSTM	AUROC	Keras, Scikit-learn
[119]	Databases of foreign trade of the Secretariat of Federal Revenue of Brazil	2014	8 Features: Foreign Trade, Tax, Transactions, Employees, Invoices, etc	AE	MSE	H2O, R
[120]	Chamber of Deputies open data, Companies data from Secretariat of Federal Revenue of Brazil	2009–2017	21 features: Brazilian State expense, party name, Type of expense, etc.	Deep Autoencoders	MSE, RMSE	H2O, R
[121]	Real-world data for automobile insurance company labeled as fraudulent	–	Car, insurance and accident related features	DMLP + LDA	TP, FP, Accuracy, Precision, F1-score	–
[122]	Transactions from a giant online payment platform	2006	Personal financial variables	GBDT+DMLP	AUROC	–
[123]	Financial transactions	–	Transaction data	LSTM	t-SNE	–
[124]	Empirical data from Greek firms	–	–	DQL	Revenue	Torch

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:

## Portfolio management studies

Art.	Data set	Period	Feature set	Method	Performance criteria	Env.
[65]	Cryptocurrencies, Bitcoin	2014–2017	Price data	CNN, RNN, LSTM	Accumulative portfolio value, MDD, SR	–
[127]	Stocks from NYSE, AMEX, NASDAQ	1965–2009	Price data	Autoencoder + RBM	Accuracy, confusion matrix	–
[128]	20 stocks from S&P500	2012–2015	Technical indicators	DMLP	Accuracy	Python, Scikit Learn, Keras, Theano
[129]	Chinese stock data	2012–2013	Technical, fundamental data	Logistic Regression, RF, DMLP	AUC, accuracy, precision, recall, f1, tpr, fpr	Keras, Tensorflow, Python, Scikit learn
[130]	Top 5 companies in S&P500	–	Price data and Financial ratios	LSTM, Auto-encoding, Smart indexing	CAGR	–
[131]	IBB biotechnology index, stocks	2012–2016	Price data	Auto-encoding, Calibrating, Validating, Verifying	Returns	–
[132]	Taiwans stock market	–	Price data	Elman RNN	MSE, return	–
[133]	FOREX (EUR/USD, etc.), Gold	2013	Price data	Evolino RNN	Return	Python
[134]	Stocks in NYSE, AMEX, NASDAQ, TAQ intraday trade	1993–2017	Price, 15 firm characteristics	LSTM+DMLP	Monthly return, SR	Python, Keras, Tensorflow in AWS
[135]	S&P500	1985–2006	monthly and daily log-returns	DBN+MLP	Validation, Test Error	Theano, Python, Matlab
[136]	10 stocks in S&P500	1997–2016	OCHLV, Price data	RNN, LSTM, GRU	Accuracy, Monthly return	Keras, Tensorflow
[137]	Analyst reports on the TSE and Osaka Exchange	2016–2018	Text	LSTM, CNN, Bi-LSTM	Accuracy, $R^2$	R, Python, MeCab
[138]	Stocks from Chinese/American stock market	2015–2018	OCHLV, Fundamental data	DDPG, PPO	SR, MDD	–
[139]	Hedge fund monthly return data	1996–2015	Return, SR, STD, Skewness, Kurtosis, Omega ratio, Fund alpha	DMLP	Sharpe ratio, Annual return, Cum. return	–
[140]	12 most-volumed cryptocurrency	2015–2016	Price data	CNN + RL	SR, portfolio value, MDD	–

# Deep learning for financial applications:

## Asset pricing and derivatives market studies

Art.	Der. type	Data set	Period	Feature set	Method	Performance criteria	Env.
[137]	Asset pricing	Analyst reports on the TSE and Osaka Exchange	2016–2018	Text	LSTM, CNN, Bi-LSTM	Accuracy, $R^2$	R, Python, MeCab
[142]	Options	Simulated a range of call option prices	–	Price data, option strike/maturity, dividend/risk free rates, volatility	DMLP	RMSE, the average percentage pricing error	Tensorflow
[143]	Futures, Options	TAIEX Options	2017	OCHLV, fundamental analysis, option price	DMLP, DMLP with Black scholes	RMSE, MAE, MAPE	–
[144]	Equity returns	Returns in NYSE, AMEX, NASDAQ	1975–2017	57 firm characteristics	Fama–French n-factor model DL	$R^2$ , RMSE	Tensorflow

# Deep learning for financial applications:

## Cryptocurrency and blockchain studies

Art.	Data set	Period	Feature set	Method	Performance criteria	Env.
[46]	Bitcoin, Dash, Ripple, Monero, Litecoin, Dogecoin, Nxt, Namecoin	2014–2017	MA, BOLL, the CRIX daily returns, Euribor interest rates, OCHLV of EURO/UK, EURO/USD, US/JPY	LSTM, RNN, DMLP	Accuracy, F1-measure	Python, Tensorflow
[65]	Cryptocurrencies, Bitcoin	2014–2017	Price data	CNN	Accumulative portfolio value, MDD, SR	–
[140]	12 most-volumed cryptocurrency	2015–2016	Price data	CNN + RL	SR, portfolio value, MDD	–
[145]	Bitcoin data	2010–2017	Hash value, bitcoin address, public/private key, digital signature, etc.	Takagi–Sugeno Fuzzy cognitive maps	Analytical hierarchy process	–
[146]	Bitcoin data	2012, 2013, 2016	TransactionId, input/output Addresses, timestamp	Graph embedding using heuristic, laplacian eigen-map, deep AE	F1-score	–
[147]	Bitcoin, Litecoin, StockTwits	2015–2018	OCHLV, technical indicators, sentiment analysis	CNN, LSTM, State Frequency Model	MSE	Keras, Tensorflow
[148]	Bitcoin	2013–2016	Price data	Bayesian optimized RNN, LSTM	Sensitivity, specificity, precision, accuracy, RMSE	Keras, Python, Hyperas

# Deep learning for financial applications:

## Financial sentiment studies coupled with text mining for forecasting

Art.	Data set	Period	Feature set	Method	Performance criteria	Env.
[137]	Analyst reports on the TSE and Osaka Exchange	2016–2018	Text	LSTM, CNN, Bi-LSTM	Accuracy, $R^2$	R, Python, MeCab
[150]	Sina Weibo, Stock market records	2012–2015	Technical indicators, sentences	DRSE	F1-score, precision, recall, accuracy, AUROC	Python
[151]	News from Reuters and Bloomberg for S&P500 stocks	2006–2015	Financial news, price data	DeepClue	Accuracy	Dynet software
[152]	News from Reuters and Bloomberg, Historical stock security data	2006–2013	News, price data	DMLP	Accuracy	–
[153]	SCI prices	2008–2015	OCHL of change rate, price	Emotional Analysis + LSTM	MSE	–
[154]	SCI prices	2013–2016	Text data and Price data	LSTM	Accuracy, F1-Measure	Python, Keras
[155]	Stocks of Google, Microsoft and Apple	2016–2017	Twitter sentiment and stock prices	RNN	–	Spark, Flume, Twitter API,
[156]	30 DJIA stocks, S&P500, DJI, news from Reuters	2002–2016	Price data and features from news articles	LSTM, NN, CNN and word2vec	Accuracy	VADER
[157]	Stocks of CSI300 index, OCHLV of CSI300 index	2009–2014	Sentiment Posts, Price data	Naive Bayes + LSTM	Precision, Recall, F1-score, Accuracy	Python, Keras
[158]	S&P500, NYSE Composite, DJIA, NASDAQ Composite	2009–2011	Twitter moods, index data	DNN, CNN	Error rate	Keras, Theano



# Deep learning for financial applications:

## Text mining studies without sentiment analysis for forecasting

Art.	Data set	Period	Feature set	Method	Performance criteria	Env.
[68]	Energy-Sector/ Company-Centric Tweets in S&P500	2015–2016	Text and Price data	RNN, KNN, SVR, LinR	Return, SR, precision, recall, accuracy	Python, Tweepy API
[165]	News from Reuters, Bloomberg	2006–2013	Financial news, price data	Bi-GRU	Accuracy	Python, Keras
[166]	News from Sina.com, ACE2005 Chinese corpus	2012–2016	A set of news text	Their unique algorithm	Precision, Recall, F1-score	–
[167]	CDAX stock market data	2010–2013	Financial news, stock market data	LSTM	MSE, RMSE, MAE, Accuracy, AUC	TensorFlow, Theano, Python, Scikit-Learn
[168]	Apple, Airbus, Amazon news from Reuters, Bloomberg, S&P500 stock prices	2006–2013	Price data, news, technical indicators	TGRU, stock2vec	Accuracy, precision, AUROC	Keras, Python
[169]	S&P500 Index, 15 stocks in S&P500	2006–2013	News from Reuters and Bloomberg	CNN	Accuracy, MCC	–
[170]	S&P500 index news from Reuters	2006–2013	Financial news titles, Technical indicators	SI-RCNN (LSTM + CNN)	Accuracy	–
[171]	10 stocks in Nikkei 225 and news	2001–2008	Textual information and Stock prices	Paragraph Vector + LSTM	Profit	–
[172]	NIFTY50 Index, NIFTY Bank/Auto/IT/Energy Index, News	2013–2017	Index data, news	LSTM	MCC, Accuracy	–
[173]	Price data, index data, news, social media data	2015	Price data, news from articles and social media	Coupled matrix and tensor	Accuracy, MCC	Jieba
[174]	HS300	2015–2017	Social media news, price data	RNN-Boost with LDA	Accuracy, MAE, MAPE, RMSE	Python, Scikit-learn

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:

## Text mining studies without sentiment analysis for forecasting

Art.	Data set	Period	Feature set	Method	Performance criteria	Env.
[175]	News and Chinese stock data	2014–2017	Selected words in a news	HAN	Accuracy, Annual return	–
[176]	News, stock prices from Hong Kong Stock Exchange	2001	Price data and TF-IDF from news	ELM, DLR, PCA, BELM, KELM, NN	Accuracy	Matlab
[177]	TWSE index, 4 stocks in TWSE	2001–2017	Technical indicators, Price data, News	CNN + LSTM	RMSE, Profit	Keras, Python, TALIB
[178]	Stock of Tsugami Corporation	2013	Price data	LSTM	RMSE	Keras, Tensorflow
[179]	News, Nikkei Stock Average and 10-Nikkei companies	1999–2008	news, MACD	RNN, RBM+DBN	Accuracy, <i>P</i> -value	–
[180]	ISMIS 2017 Data Mining Competition dataset	–	Expert identifier, classes	LSTM + GRU + FFNN	Accuracy	–
[181]	Reuters, Bloomberg News, S&P500 price	2006–2013	News and sentences	LSTM	Accuracy	–
[182]	APPL from S&P500 and news from Reuters	2011–2017	Input news, OCHLV, Technical indicators	CNN + LSTM, CNN+SVM	Accuracy, F1-score	Tensorflow
[183]	Nikkei225, S&P500, news from Reuters and Bloomberg	2001–2013	Stock price data and news	DGM	Accuracy, MCC, %profit	–
[184]	Stocks from S&P500	2006–2013	Text (news) and Price data	LAR+News, RF+News	MAPE, RMSE	–

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:

## Financial sentiment studies coupled with text mining without forecasting

Art.	Data set	Period	Feature set	Method	Performance criteria	Env.
[85]	883 BHC from EDGAR	2006–2017	Tokens, weighted sentiment polarity, leverage and ROA	CNN, LSTM, SVM, Random Forest	Accuracy, Precision, Recall, F1-score	Keras, Python, Scikit-learn
[185]	SemEval-2017 dataset, financial text, news, stock market data	2017	Sentiments in Tweets, News headlines	Ensemble SVR, CNN, LSTM, GRU	Cosine similarity score, agreement score, class score	Python, Keras, Scikit Learn
[186]	Financial news from Reuters	2006–2015	Word vector, Lexical and Contextual input	Targeted dependency tree LSTM	Cumulative abnormal return	–
[187]	Stock sentiment analysis from StockTwits	2015	StockTwits messages	LSTM, Doc2Vec, CNN	Accuracy, precision, recall, f-measure, AUC	–
[188]	Sina Weibo, Stock market records	2012–2015	Technical indicators, sentences	DRSE	F1-score, precision, recall, accuracy, AUROC	Python
[189]	News from NowNews, AppleDaily, LTN, MoneyDJ for 18 stocks	2013–2014	Text, Sentiment	LSTM, CNN	Return	Python, Tensorflow
[190]	StockTwits	2008–2016	Sentences, StockTwits messages	CNN, LSTM, GRU	MCC, WSURT	Keras, Tensorflow
[191]	Financial statements of Japan companies	–	Sentences, text	DMLP	Precision, recall, f-score	–
[192]	Twitter posts, news headlines	–	Sentences, text	Deep-FASP	Accuracy, MSE, R <sup>2</sup>	–
[193]	Forums data	2004–2013	Sentences and keywords	Recursive neural tensor networks	Precision, recall, f-measure	–
[194]	News from Financial Times related US stocks	–	Sentiment of news headlines	SVR, Bidirectional LSTM	Cosine similarity	Python, Scikit Learn, Keras, Tensorflow

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:

## Other text mining studies

Art.	Data set	Period	Feature set	Method	Performance criteria	Env.
[72]	News from NowNews, AppleDaily, LTN, MoneyDJ for 18 stocks	2013–2014	Text, Sentiment	DMLP	Return	Python, Tensorflow
[86]	The event data set for large European banks, news articles from Reuters	2007–2014	Word, sentence	DMLP +NLP preprocess	Relative usefulness, F1-score	–
[87]	Event dataset on European banks, news from Reuters	2007–2014	Text, sentence	Sentence vector + DFFN	Usefulness, F1-score, AUROC	–
[88]	News from Reuters, fundamental data	2007–2014	Financial ratios and news text	doc2vec + NN	Relative usefulness	Doc2vec
[121]	Real-world data for automobile insurance company labeled as fraudulent	–	Car, insurance and accident related features	DMLP + LDA	TP, FP, Accuracy, Precision, F1-score	–
[123]	Financial transactions	–	Transaction data	LSTM	t-SNE	–
[195]	Taiwan's National Pension Insurance	2008–2014	Insured's id, area-code, gender, etc.	RNN	Accuracy, total error	Python
[196]	StockTwits	2015–2016	Sentences, StockTwits messages	Doc2vec, CNN	Accuracy, precision, recall, f-measure, AUC	Python, Tensorflow

# Deep learning for financial applications:

## Other theoretical or conceptual studies

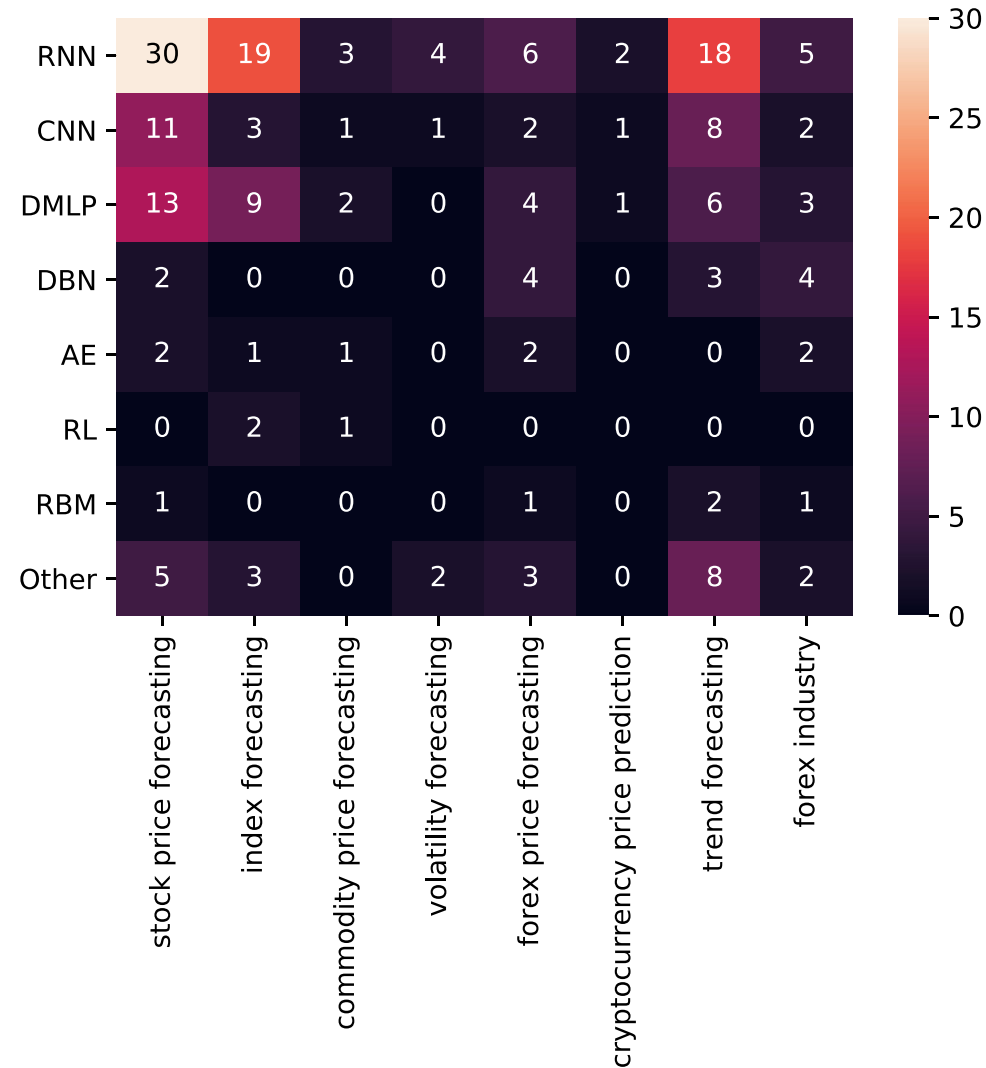
Art.	SubTopic	IsTimeSeries?	Data set	Period	Feature set	Method
[197]	Analysis of AE, SVD	Yes	Selected stocks from the IBB index and stock of Amgen Inc.	2012–2014	Price data	AE, SVD
[198]	Fraud Detection in Banking	No	Risk Management / Fraud Detection	–	–	DRL

# Deep learning for financial applications:

## Other financial applications

Art.	Subtopic	Data set	Period	Feature set	Method	Performance criteria	Env.
[47]	Improving trading decisions	S&P500, KOSPI, HSI, and EuroStoxx50	1987–2017	200-days stock price	Deep Q-Learning and DMLP	Total profit, Correlation	–
[193]	Identifying Top Sellers In Underground Economy	Forums data	2004–2013	Sentences and keywords	Recursive neural tensor networks	Precision, recall, f-measure	–
[195]	Predicting Social Ins. Payment Behavior	Taiwan's National Pension Insurance	2008–2014	Insured's id, area-code, gender, etc.	RNN	Accuracy, total error	Python
[199]	Speedup	45 CME listed commodity and FX futures	1991–2014	Price data	DNN	–	–
[200]	Forecasting Fundamentals	Stocks in NYSE, NASDAQ or AMEX exchanges	1970–2017	16 fundamental features from balance sheet	DMLP, LFM	MSE, Compound annual return, SR	–
[201]	Predicting Bank Telemarketing	Phone calls of bank marketing data	2008–2010	16 finance-related attributes	CNN	Accuracy	–
[202]	Corporate Performance Prediction	22 pharmaceutical companies data in US stock market	2000–2015	11 financial and 4 patent indicator	RBM, DBN	RMSE, profit	–

# Financial time series forecasting with deep learning: Topic-model heatmap



# Stock price forecasting using only raw time series data

Art.	Data set	Period	Feature set	Lag	Horizon	Method	Performance criteria	Env.
[80]	38 stocks in KOSPI	2010–2014	Lagged stock returns	50 min	5 min	DNN	NMSE, RMSE, MAE, MI	–
[81]	China stock market, 3049 Stocks	1990–2015	OCHLV	30 d	3 d	LSTM	Accuracy	Theano, Keras
[82]	Daily returns of 'BRD' stock in Romanian Market	2001–2016	OCHLV	–	1 d	LSTM	RMSE, MAE	Python, Theano
[83]	297 listed companies of CSE	2012–2013	OCHLV	2 d	1 d	LSTM, SRNN, GRU	MAD, MAPE	Keras
[84]	5 stock in NSE	1997–2016	OCHLV, Price data, turnover and number of trades.	200 d	1..10 d	LSTM, RNN, CNN, MLP	MAPE	–
[85]	Stocks of Infosys, TCS and CIPLA from NSE	2014	Price data	–	–	RNN, LSTM and CNN	Accuracy	–
[86]	10 stocks in S&P500	1997–2016	OCHLV, Price data	36 m	1 m	RNN, LSTM, GRU	Accuracy, Monthly return	Keras, Tensorflow
[87]	Stocks data from S&P500	2011–2016	OCHLV	1 d	1 d	DBN	MSE, norm-RMSE, MAE	–
[88]	High-frequency transaction data of the CSI300 futures	2017	Price data	–	1 min	DNN, ELM, RBF	RMSE, MAPE, Accuracy	Matlab
[89]	Stocks in the S&P500	1990–2015	Price data	240 d	1 d	DNN, GBT, RF	Mean return, MDD, Calmar ratio	H2O
[90]	ACI Worldwide, Staples, and Seagate in NASDAQ	2006–2010	Daily closing prices	17 d	1 d	RNN, ANN	RMSE	–
[91]	Chinese Stocks	2007–2017	OCHLV	30 d	1..5 d	CNN + LSTM	Annualized Return, Mxm Retracement	Python
[92]	20 stocks in S&P500	2010–2015	Price data	–	–	AE + LSTM	Weekly Returns	–
[93]	S&P500	1985–2006	Monthly and daily log-returns	*	1 d	DBN+MLP	Validation, Test Error	Theano, Python, Matlab
[94]	12 stocks from SSE Composite Index	2000–2017	OCHLV	60 d	1..7 d	DWNN	MSE	Tensorflow
[95]	50 stocks from NYSE	2007–2016	Price data	–	1d, 3 d, 5 d	SFM	MSE	–

Source: 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.



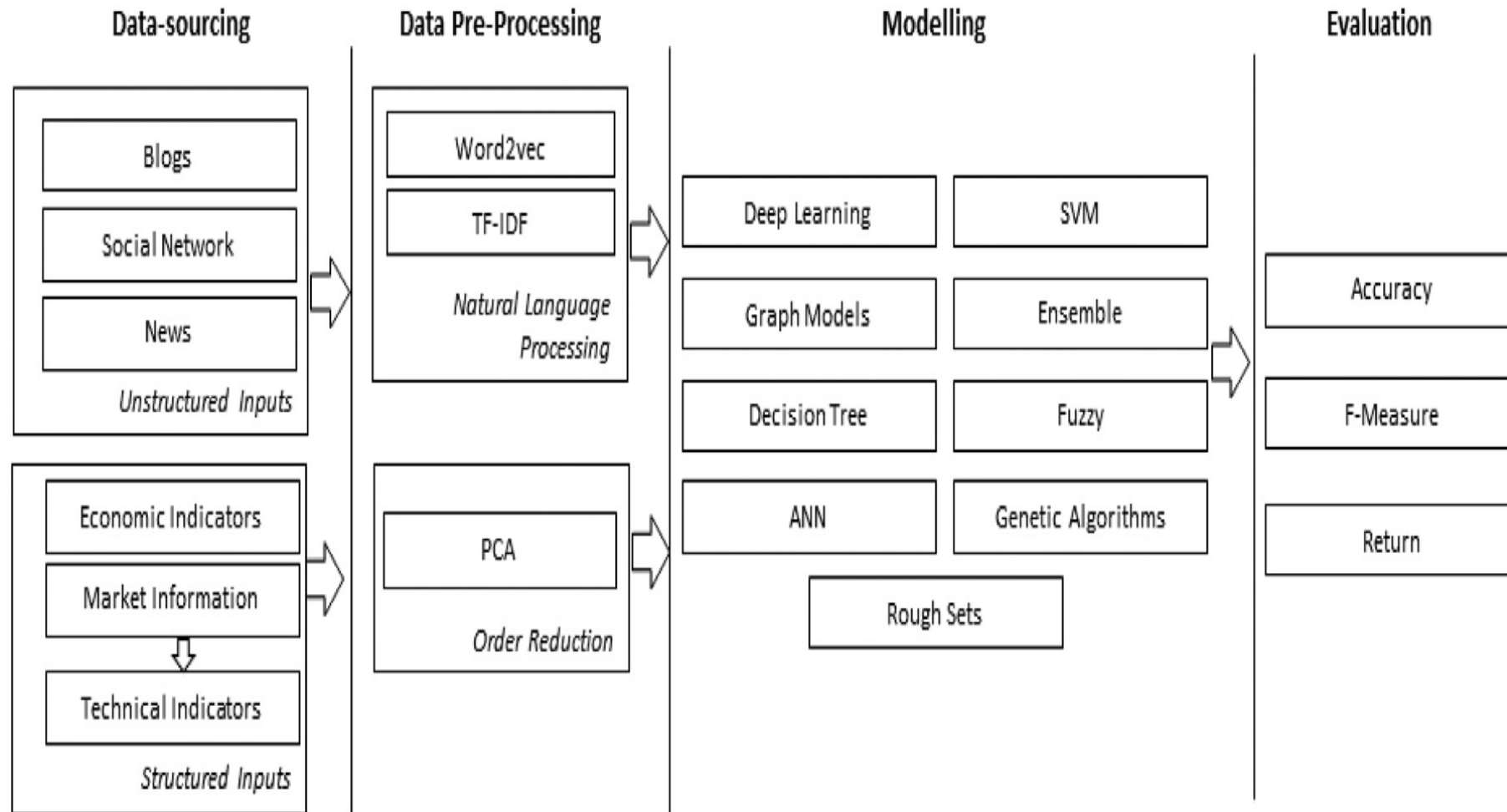
# Stock price forecasting using various data

Art.	Data set	Period	Feature set	Lag	Horizon	Method	Performance criteria	Env.
[96]	Japan Index constituents from WorldScope	1990–2016	25 Fundamental Features	10 d	1 d	DNN	Correlation, Accuracy, MSE	Tensorflow
[97]	Return of S&P500	1926–2016	Fundamental Features:	–	1 s	DNN	MSPE	Tensorflow
[98]	U.S. low-level disaggregated macroeconomic time series	1959–2008	GDP, Unemployment rate, Inventories, etc.	–	–	DNN	R <sup>2</sup>	–
[99]	CDAX stock market data	2010–2013	Financial news, stock market data	20 d	1 d	LSTM	MSE, RMSE, MAE, Accuracy, AUC	TensorFlow, Theano, Python, Scikit-Learn
[100]	Stock of Tsugami Corporation	2013	Price data	–	–	LSTM	RMSE	Keras, Tensorflow
[101]	Stocks in China's A-share	2006–2007	11 technical indicators	–	1 d	LSTM	AR, IR, IC	–
[102]	SCI prices	2008–2015	OCHL of change rate, price	7 d	–	EmotionalAnalysis + LSTM	MSE	–
[103]	10 stocks in Nikkei 225 and news	2001–2008	Textual information and Stock prices	10 d	–	Paragraph Vector + LSTM	Profit	–
[104]	TKC stock in NYSE and QQQQ ETF	1999–2006	Technical indicators, Price	50 d	1 d	RNN (Jordan–Elman)	Profit, MSE	Java
[105]	10 Stocks in NYSE	–	Price data, Technical indicators	20 min	1 min	LSTM, MLP	RMSE	–
[106]	42 stocks in China's SSE	2016	OCHLV, Technical Indicators	242 min	1 min	GAN (LSTM, CNN)	RMSRE, DPA, GAN-F, GAN-D	–
[107]	Google's daily stock data	2004–2015	OCHLV, Technical indicators	20 d	1 d	(2D) <sup>2</sup> PCA + DNN	SMAPE, PCD, MAPE, RMSE, HR, TR, R <sup>2</sup>	R, Matlab
[108]	GarantiBank in BIST, Turkey	2016	OCHLV, Volatility, etc.	–	–	PLR, Graves LSTM	MSE, RMSE, MAE, RSE, R <sup>2</sup>	Spark
[109]	Stocks in NYSE, AMEX, NASDAQ, TAQ intraday trade	1993–2017	Price, 15 firm characteristics	80 d	1 d	LSTM+MLP	Monthly return, SR	Python,Keras, Tensorflow in AWS
[110]	Private brokerage company's real data of risky transactions	–	250 features: order details, etc.	–	–	CNN, LSTM	F1-Score	Keras, Tensorflow
[111]	Fundamental and Technical Data, Economic Data	–	Fundamental , technical and market information	–	–	CNN	–	–
[112]	The LOB of 5 stocks of Finnish Stock Market	2010	FI-2010 dataset: bid/ask and volume	–	*	WMTR, MDA	Accuracy, Precision, Recall, F1-Score	–
[113]	Returns in NYSE, AMEX, NASDAQ	1975–2017	57 firm characteristics	*	–	Fama–French n-factor model DL	R <sup>2</sup> , RMSE	Tensorflow

Source: 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.

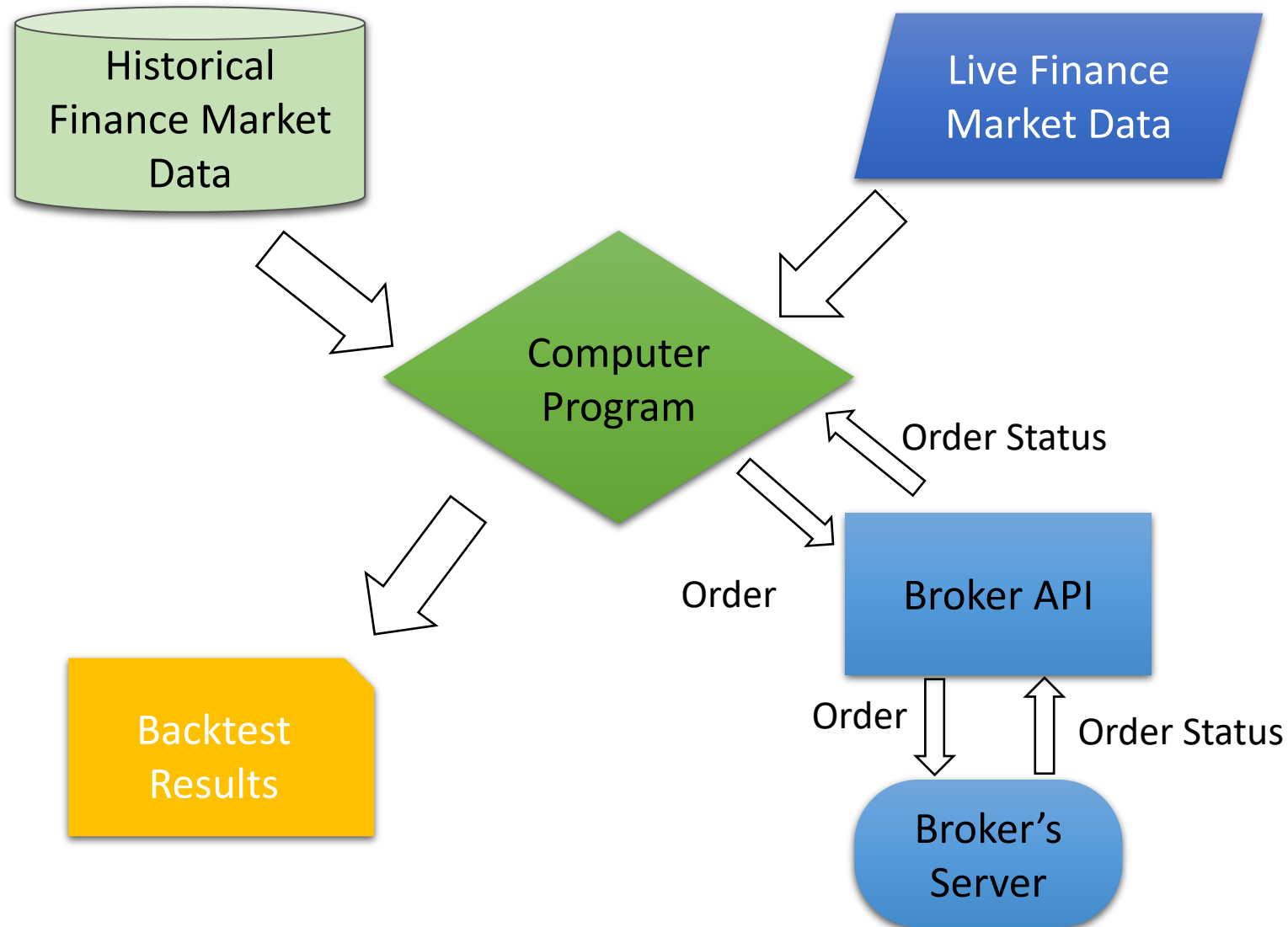
# Stock Market Movement Forecast:

## Phases of the stock market modeling

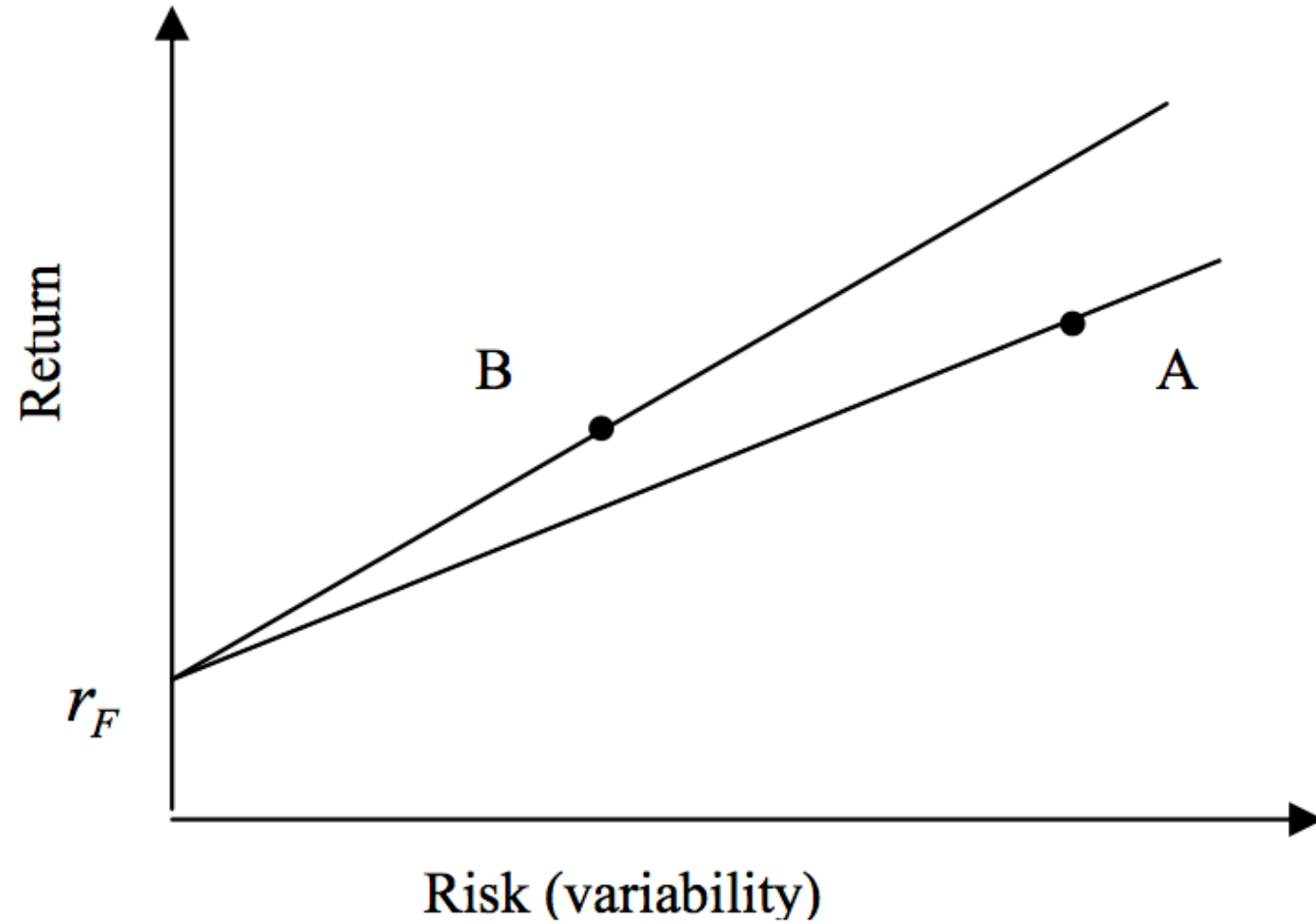


# Algorithmic Trading

# Algorithmic Trading



# Risk and Return



# Sharpe Ratio

$$\text{Sharpe Ratio} = \frac{\text{Portfolio Return} - \text{Risk Free Return}}{\text{Portfolio Risk}}$$

# Sharpe Ratio

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

Where

$r_P$  = portfolio return

$r_F$  = risk free rate

$\sigma_P$  = portfolio risk (variability, standard deviation of return)

# Sortino Ratio

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

Where

$r_P$  = portfolio return

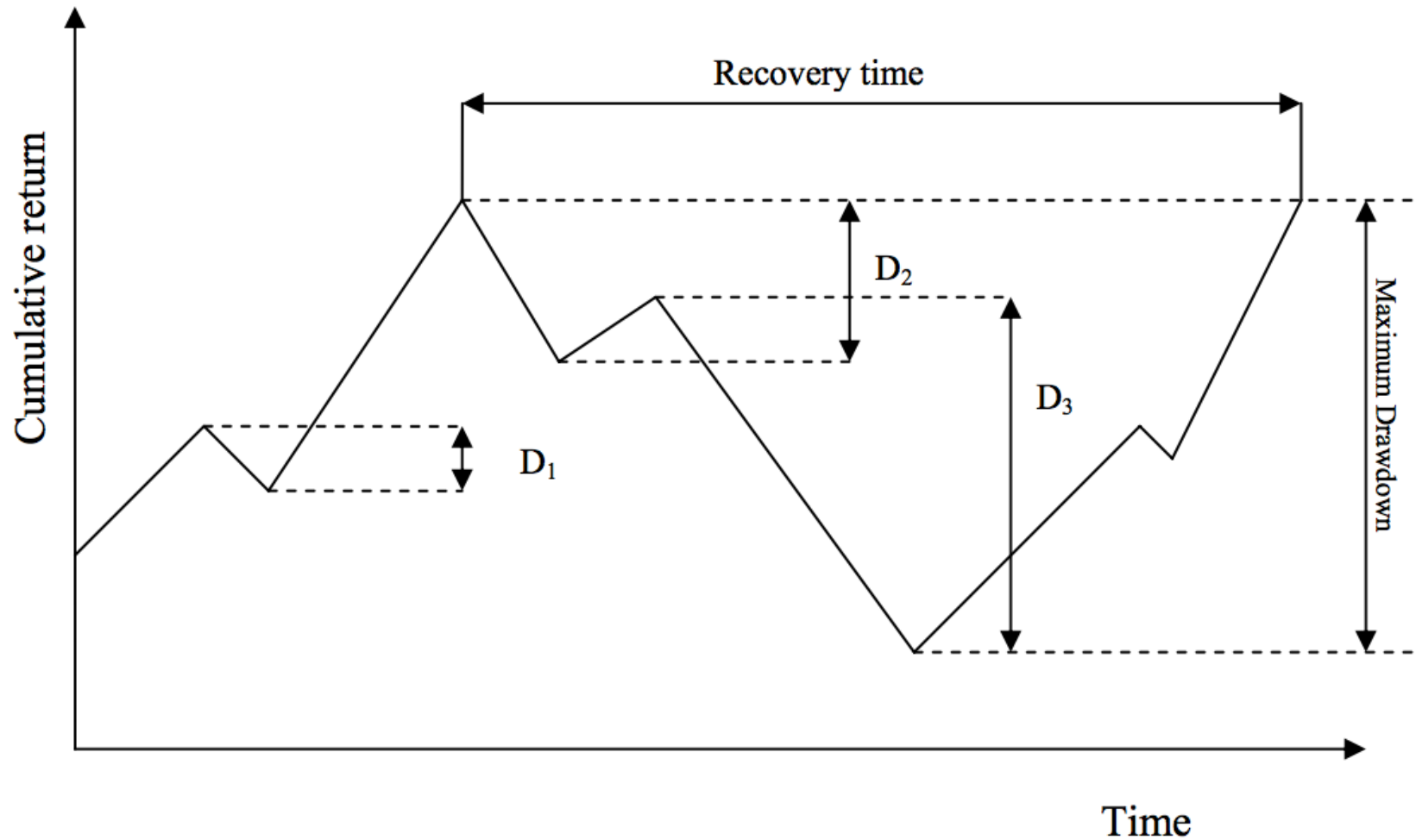
$r_T$  = Minimum Target Return

$\sigma_D$  = Downside Risk

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

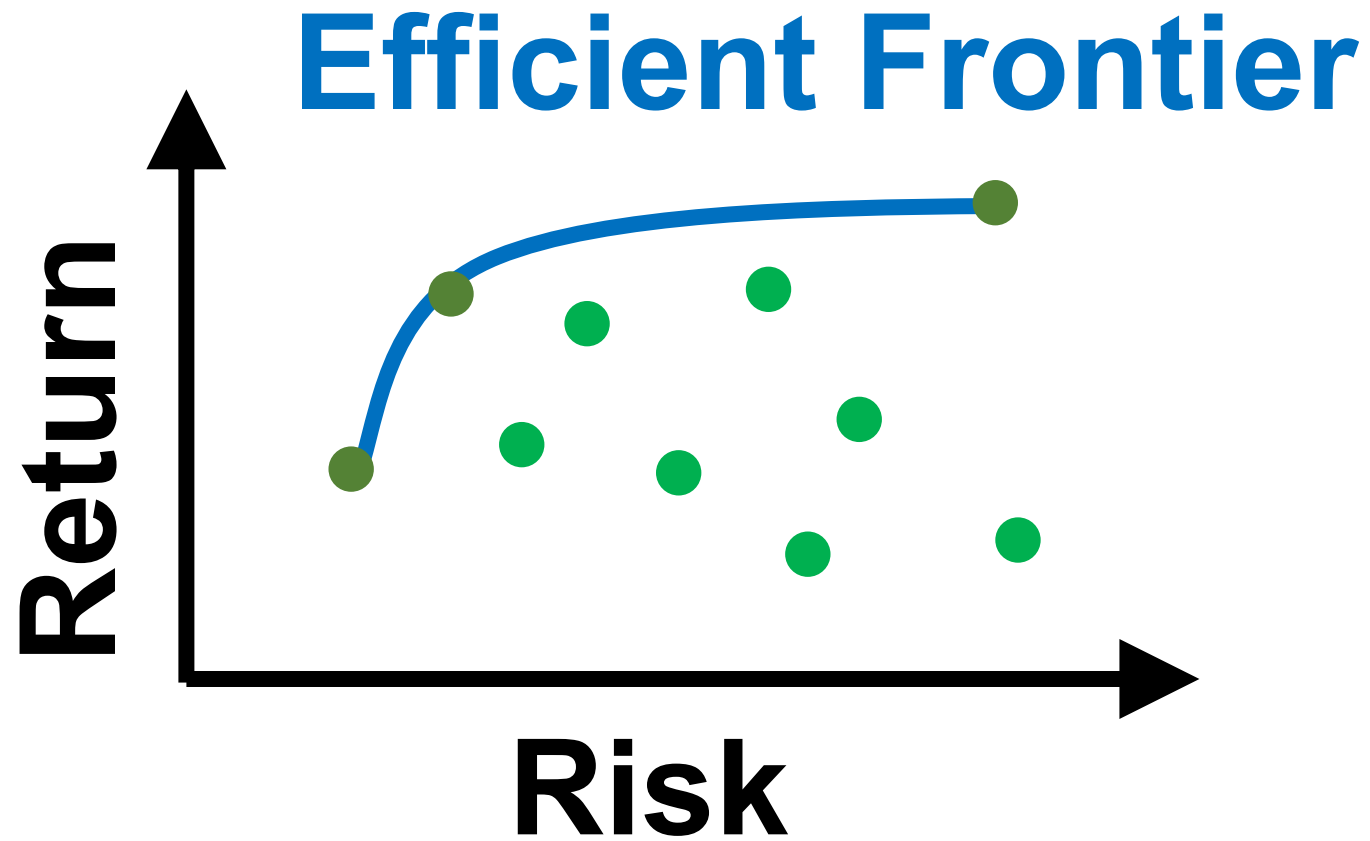


# Max Drawdown



# Portfolio Optimization

## Efficient Frontier



# 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)

btccusd

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

btccusd

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

btccusd

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

```
backtesting 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('backtesting 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('backtesting 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('backtesting 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)
```

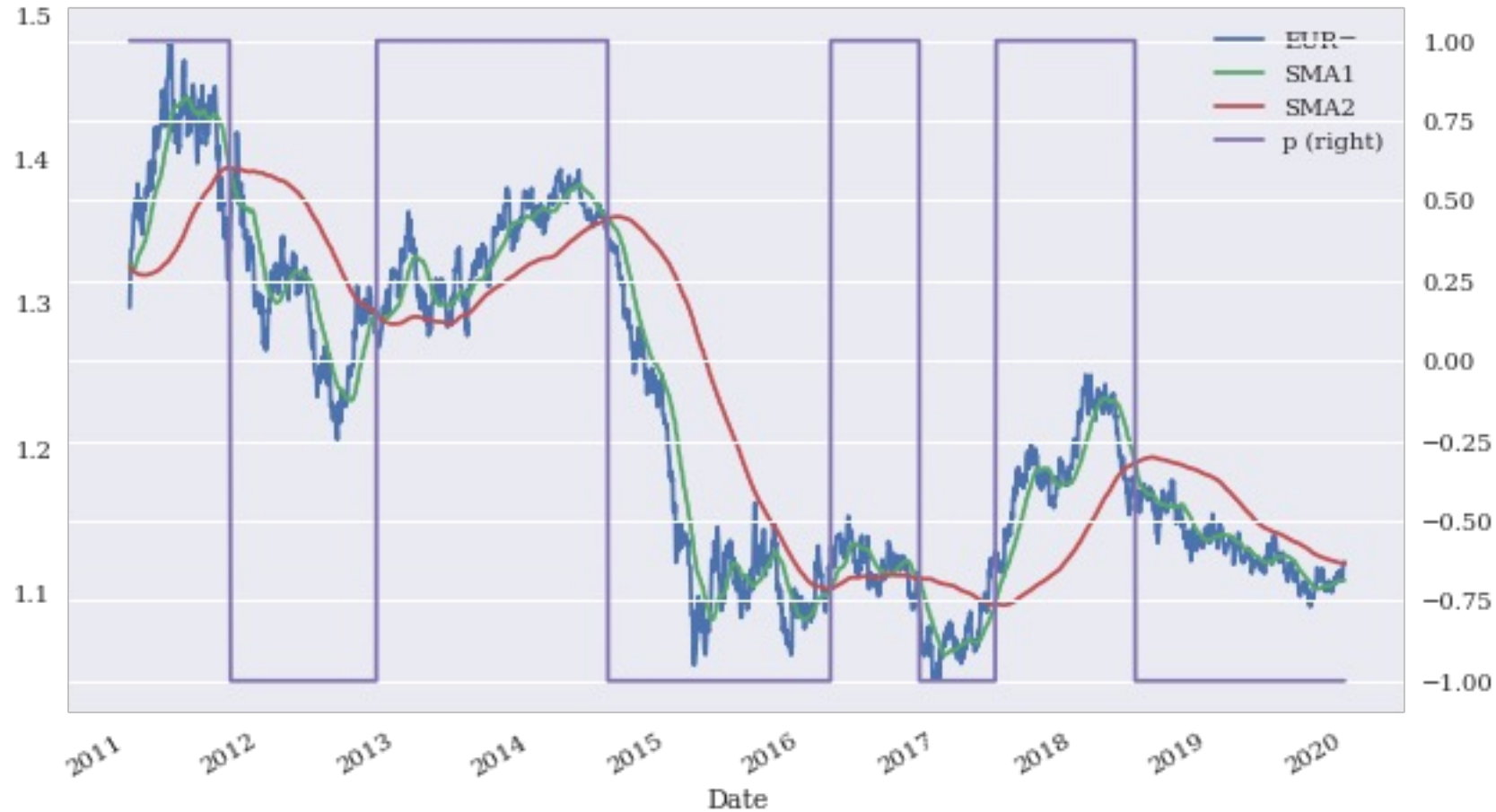
# Backtesting



# Time series data for EUR/USD and SMAs



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



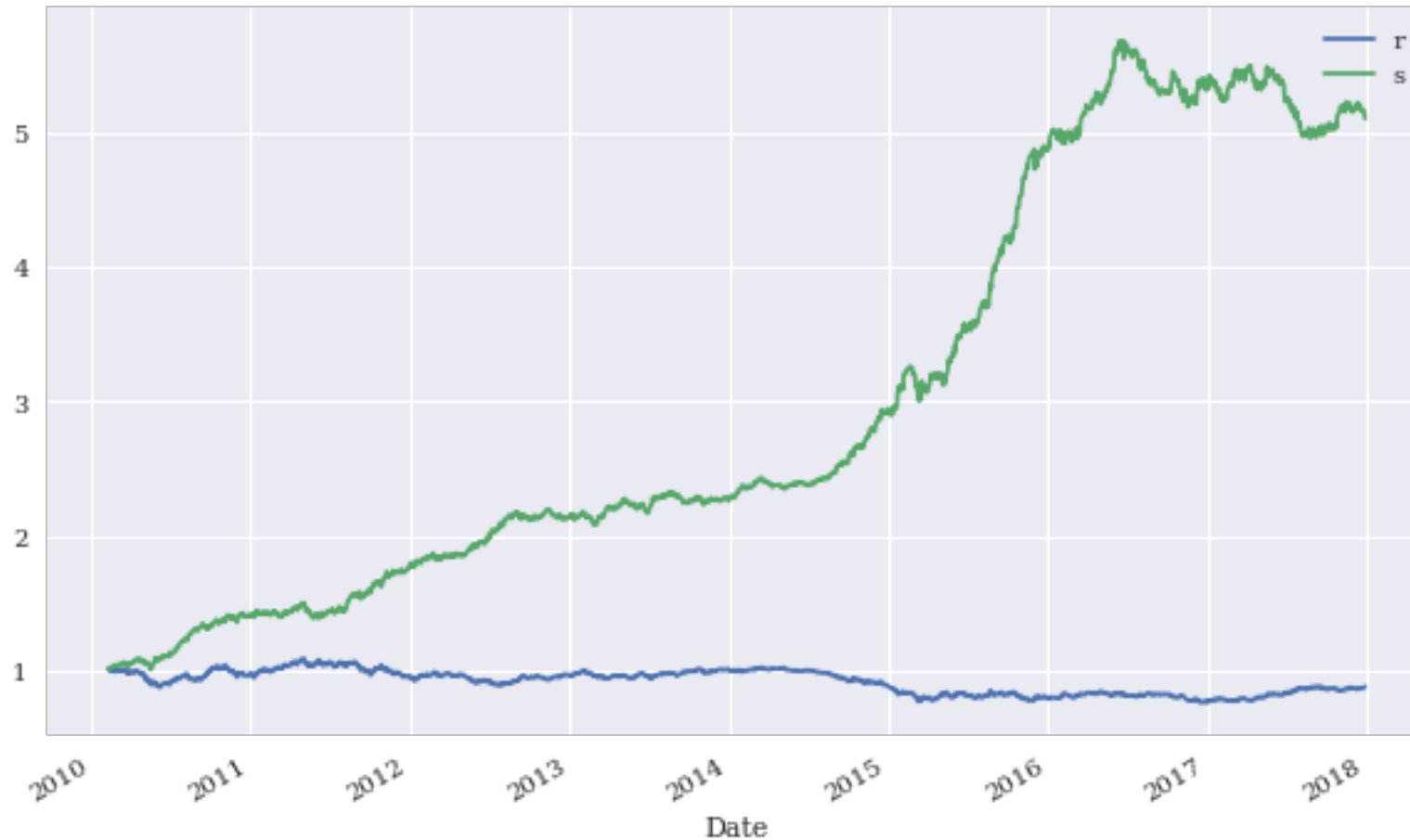
# Gross performance of passive benchmark investment and SMA strategy



# Gross performance of the SMA strategy before and after transaction costs



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



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





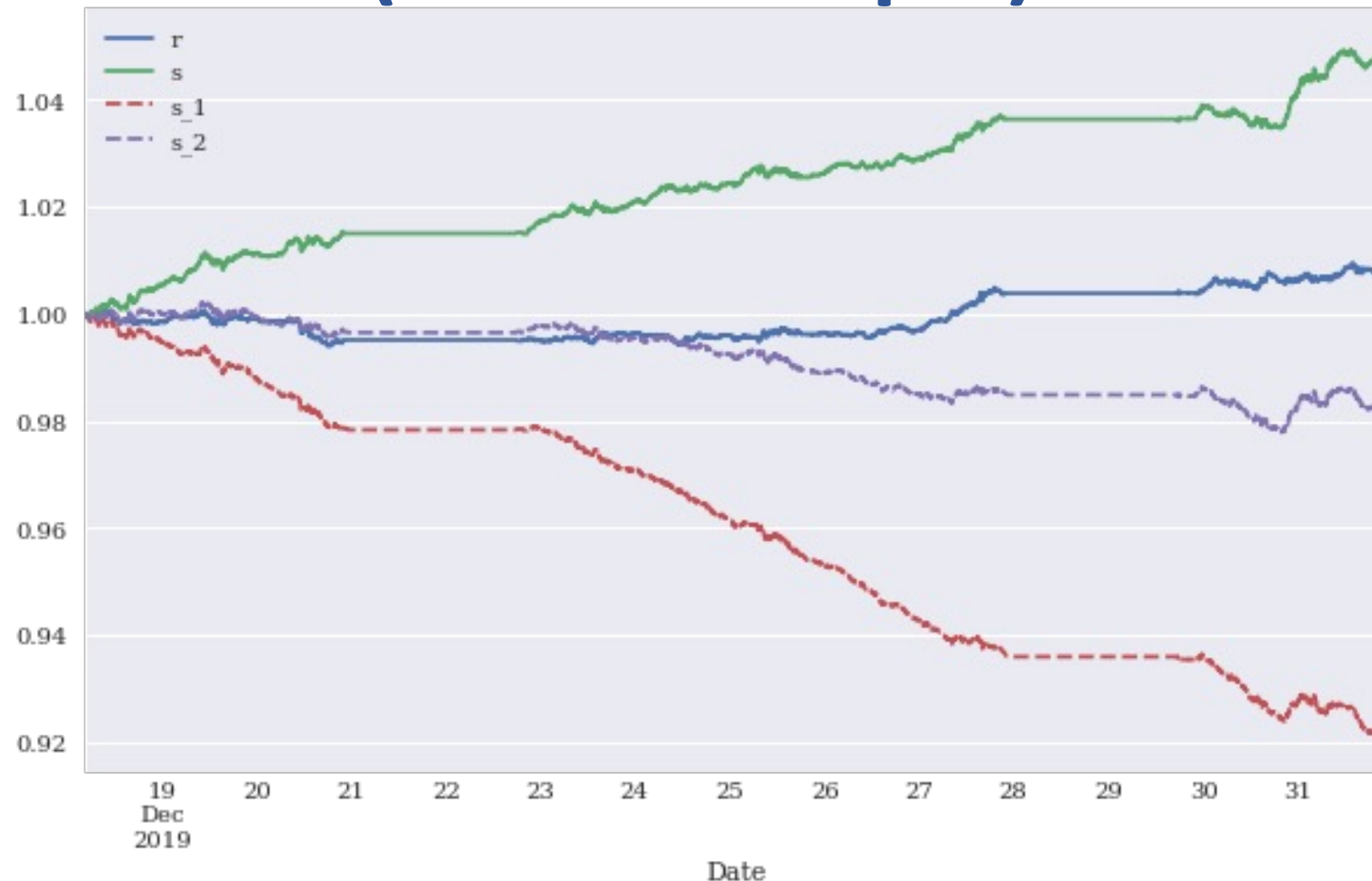
# Gross performance of the daily DNN strategy before and after transaction costs (out-of-sample)



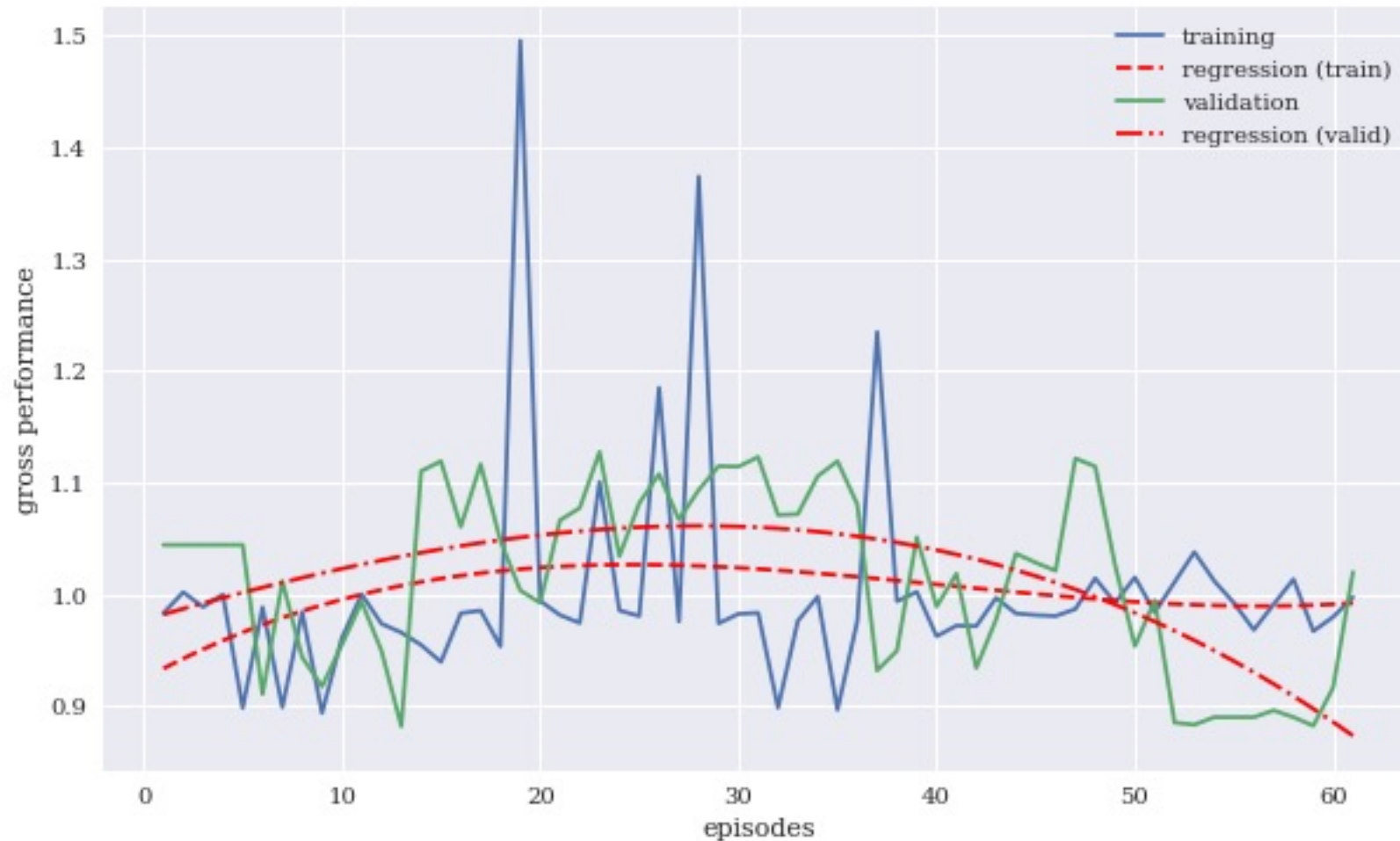
# Gross performance of the passive benchmark investment and the DNN intraday strategy (out-of-sample)



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



# Gross performance on training and validation data set



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



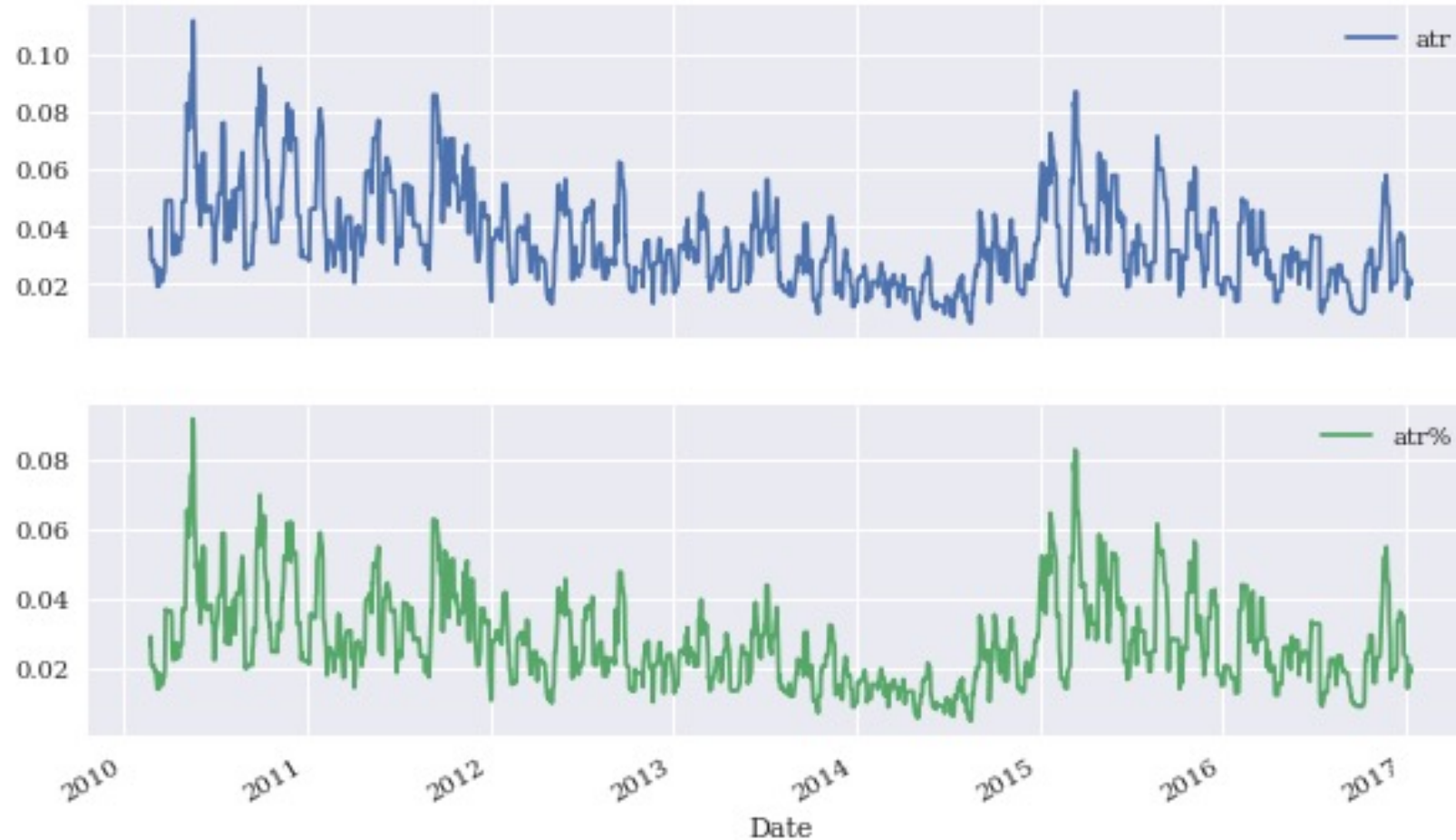
# Gross performance of the trading bot before and after transaction costs (in-sample)



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

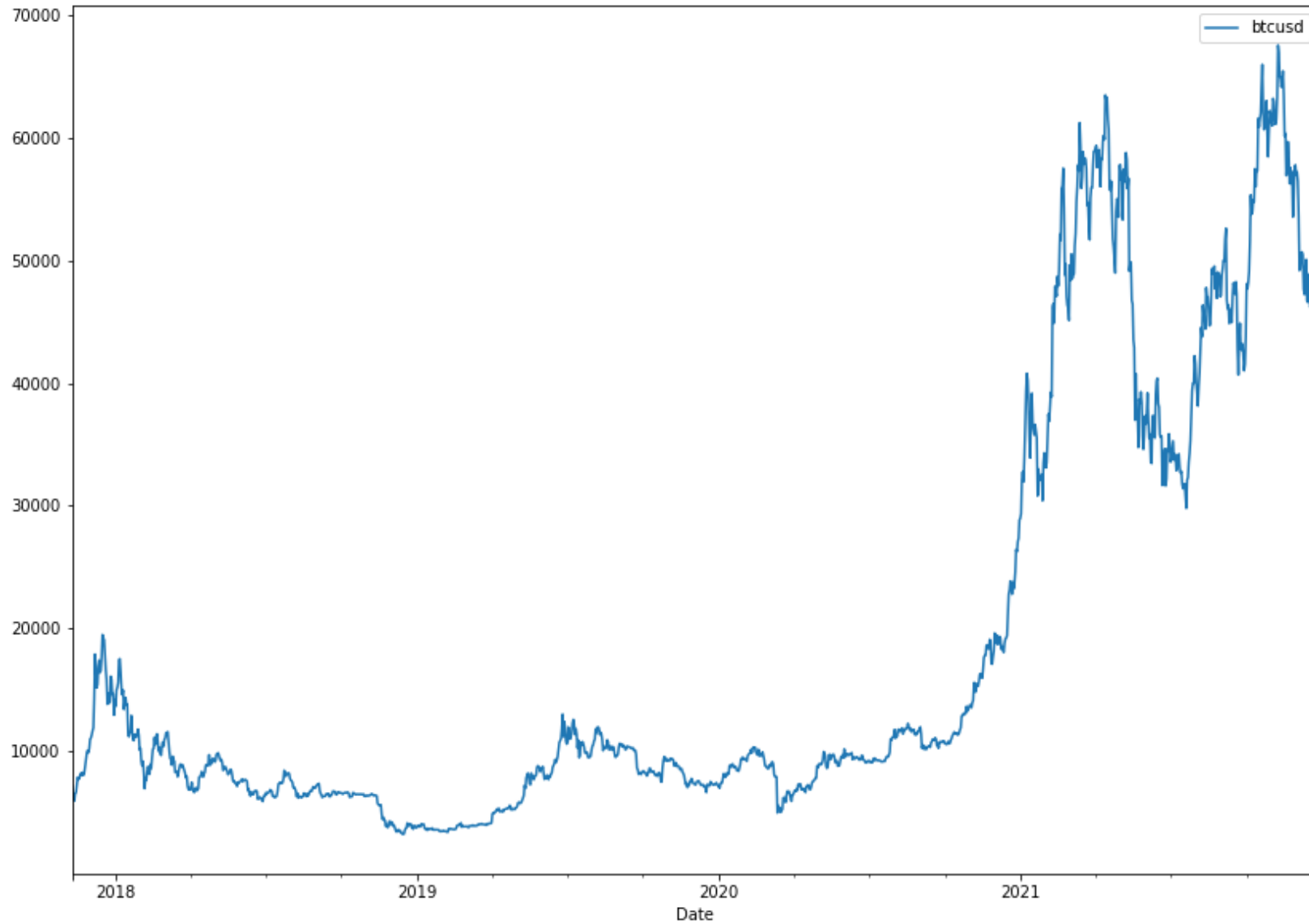


# Average true range (ATR) in absolute (price) and relative (%) terms



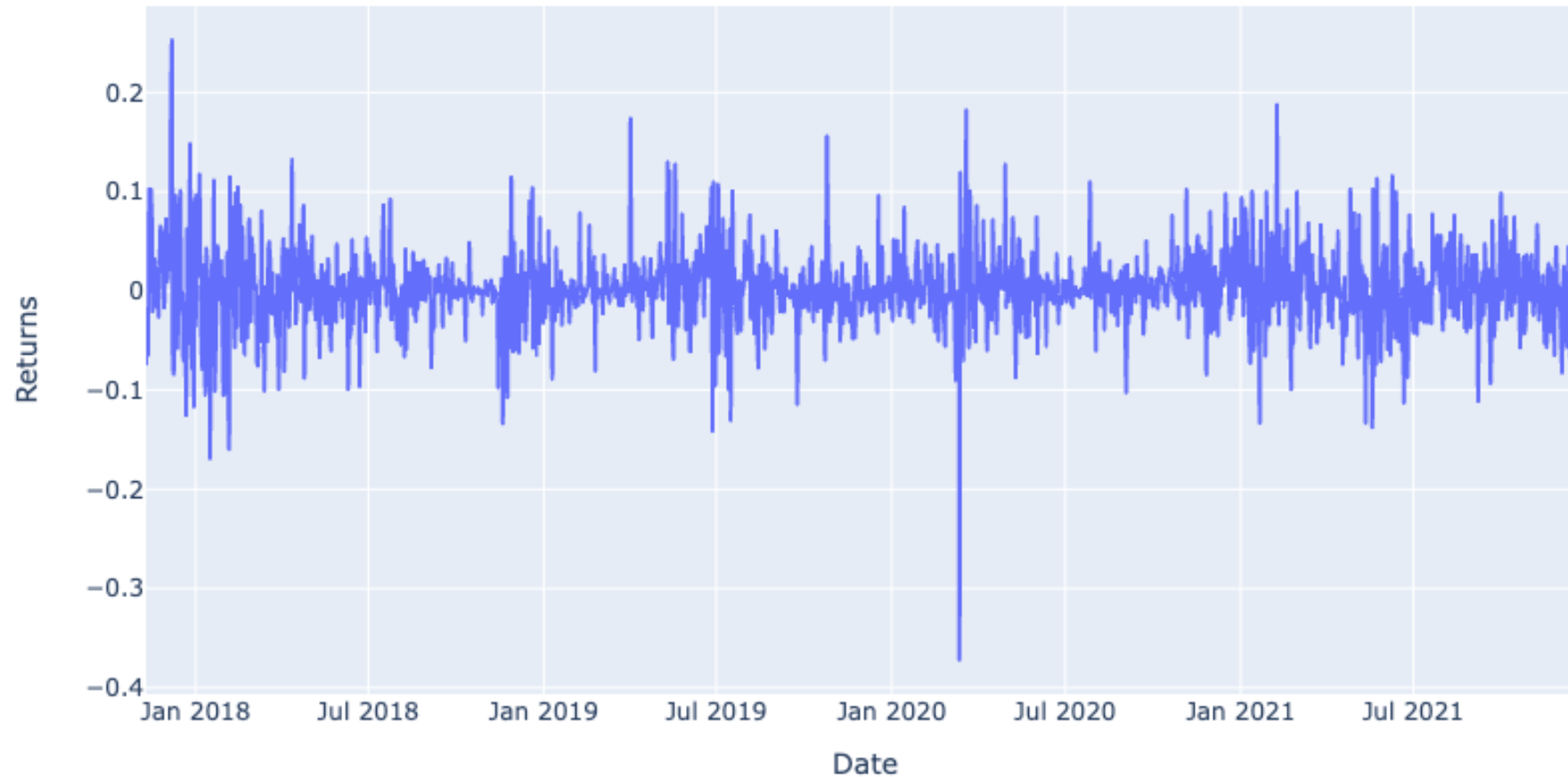


# BTC-USD



# BTC-USD Returns

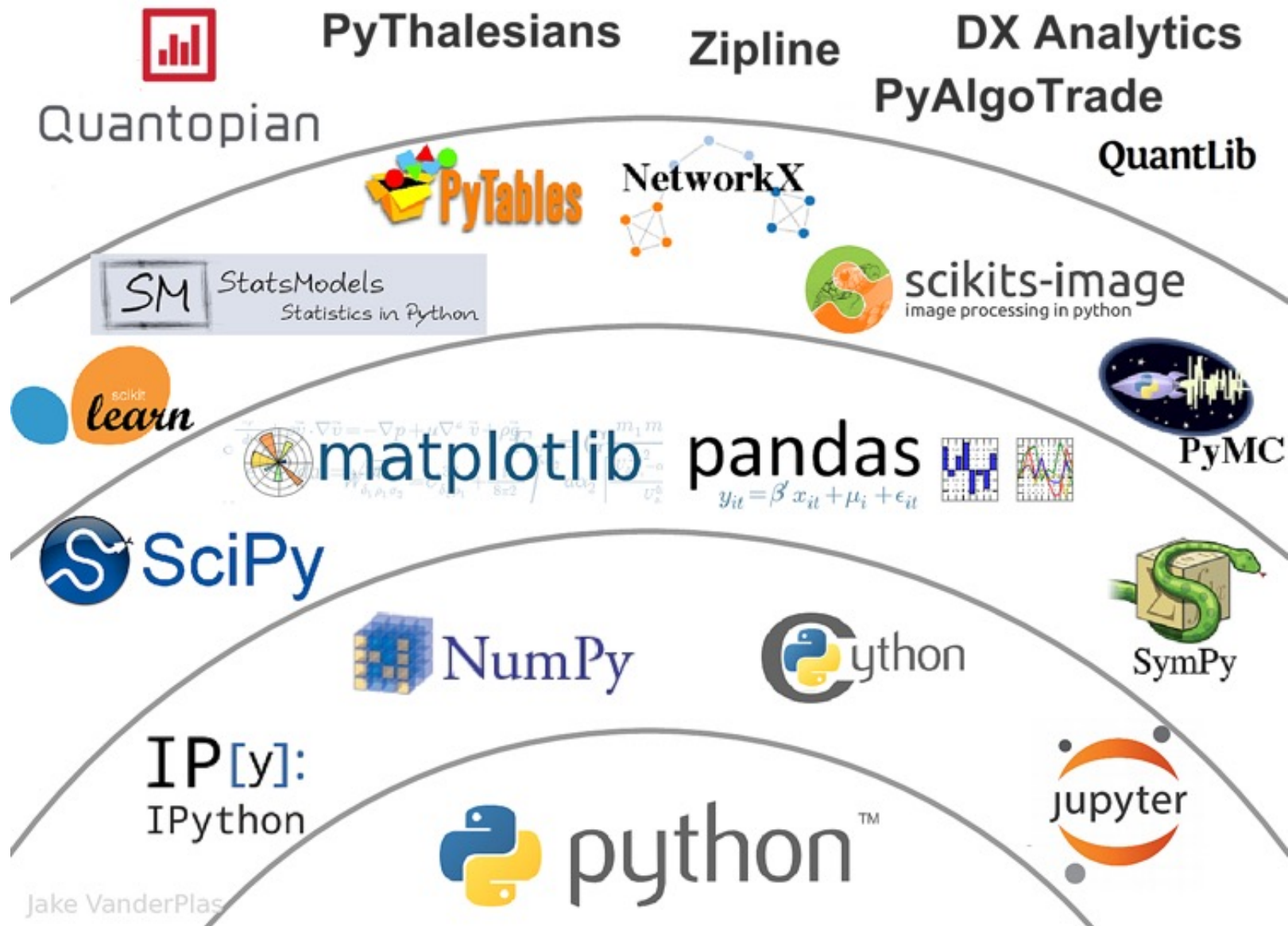
btcsud returns



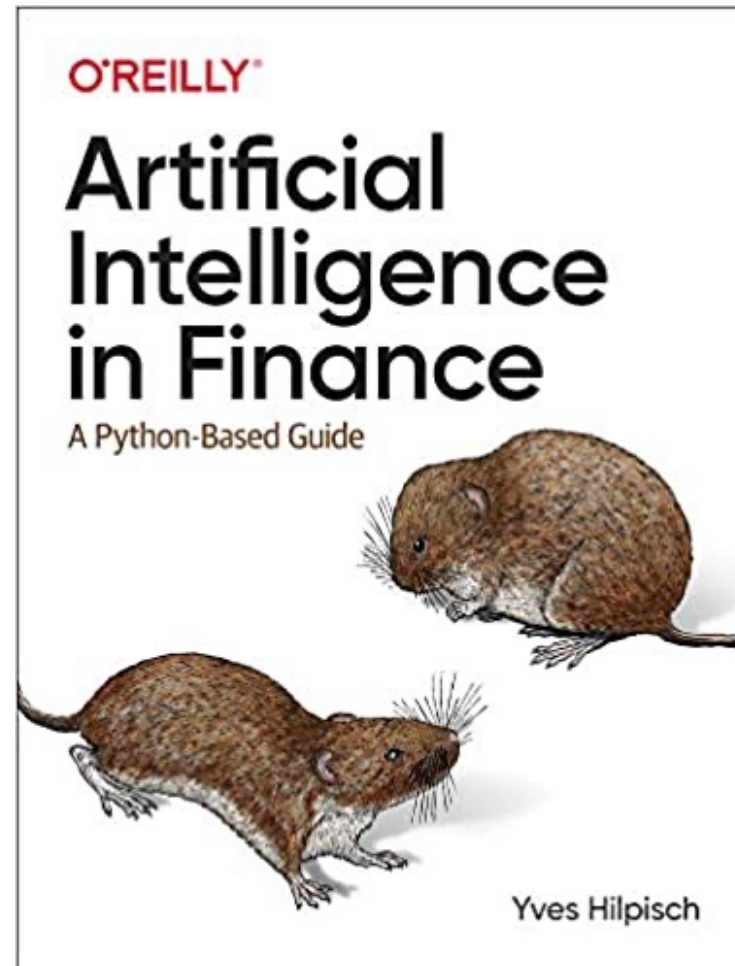
# BTC-USD Returns Box



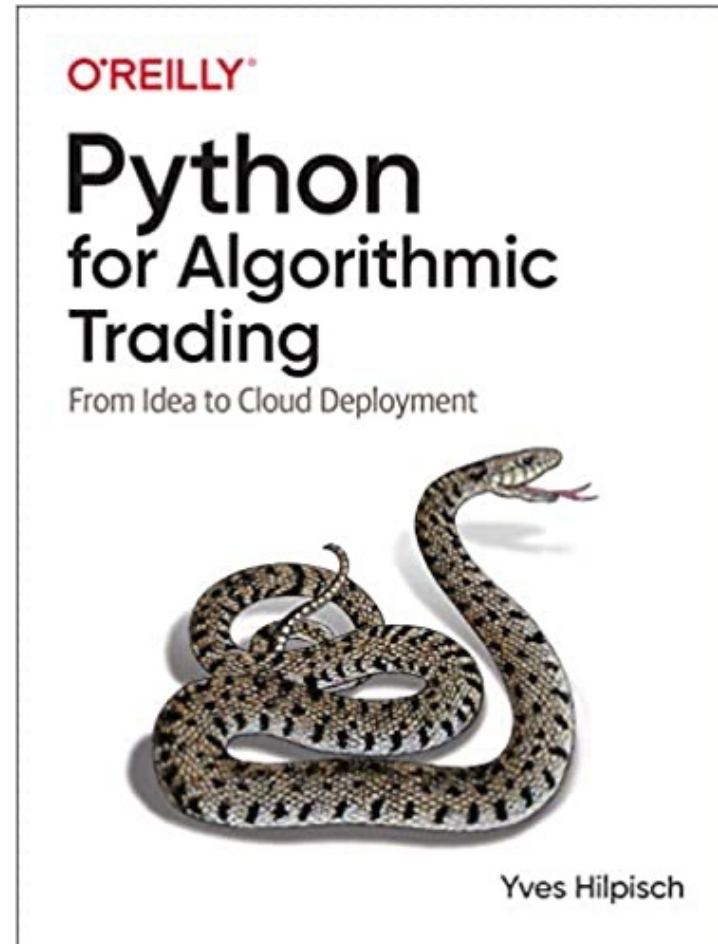
# The Quant Finance PyData Stack



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



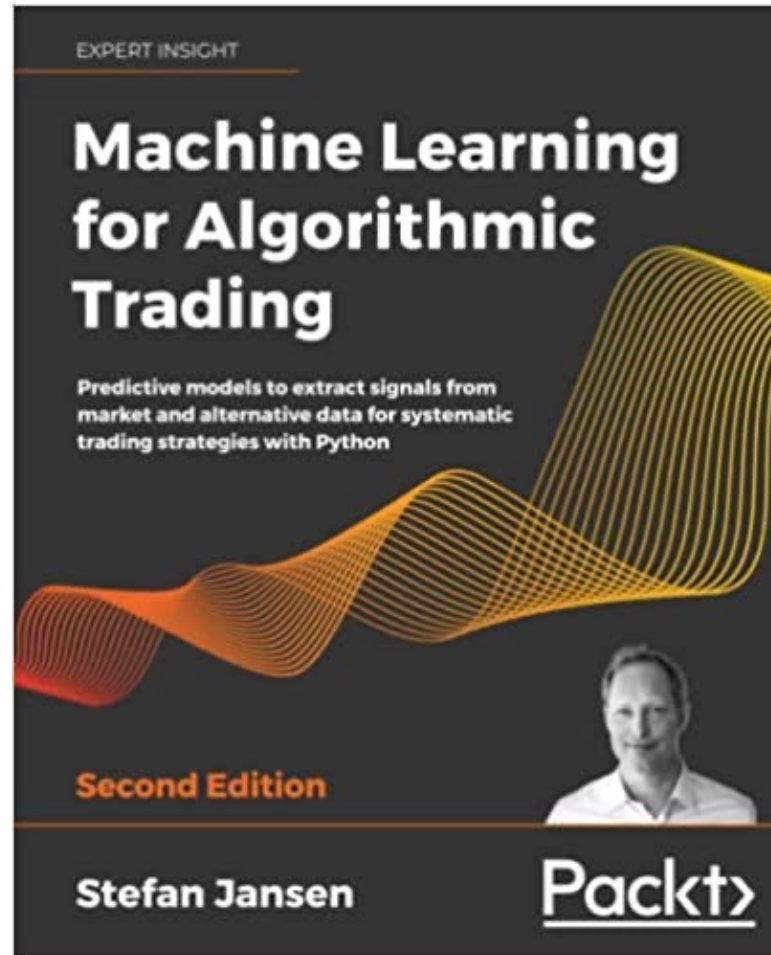
**Yves Hilpisch (2020),**  
**Python for Algorithmic Trading:**  
**From Idea to Cloud Deployment,**  
**O'Reilly**



Stefan Jansen (2020),

# Machine Learning for Algorithmic Trading:

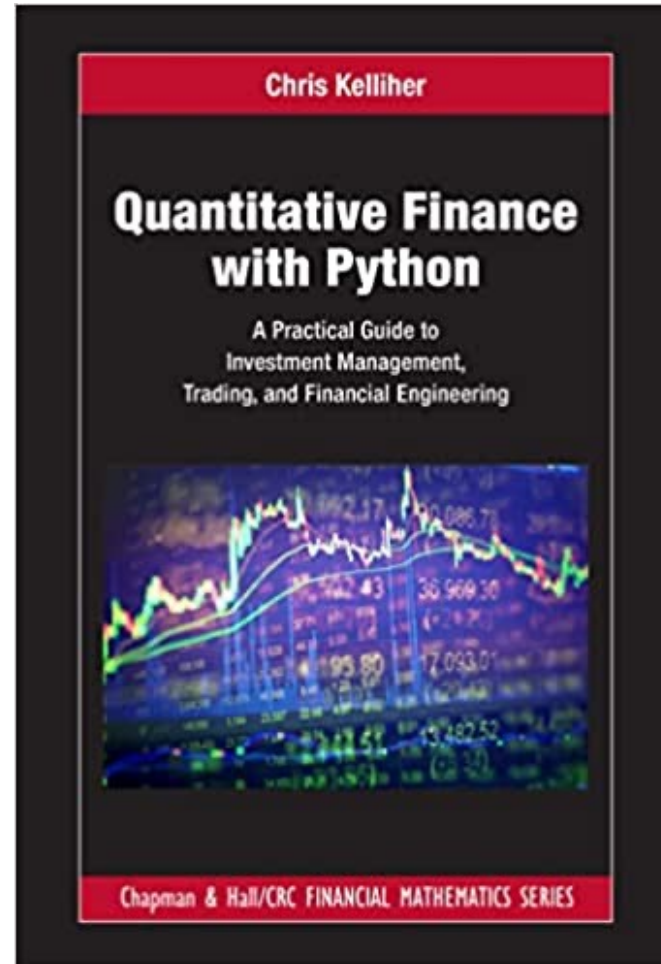
Predictive models to extract signals from market and alternative data for systematic trading strategies with Python, 2nd Edition,  
Packt Publishing.



**Chris Kelliher (2022),**

**Quantitative Finance With Python:**

**A Practical Guide to Investment Management, Trading, and Financial Engineering,  
Chapman and Hall/CRC.**





# 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](#).



**About**

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

[home.tpq.io/books/aiif](https://home.tpq.io/books/aiif)

Readme View license

**Releases**

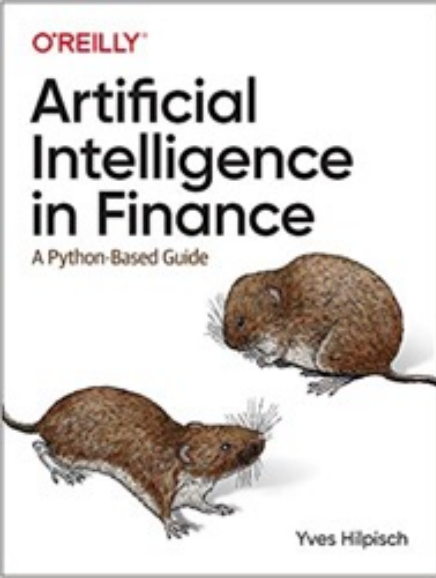
No releases published

**Packages**

No packages published

**Languages**

- Jupyter Notebook 97.4%
- Python 2.6%



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

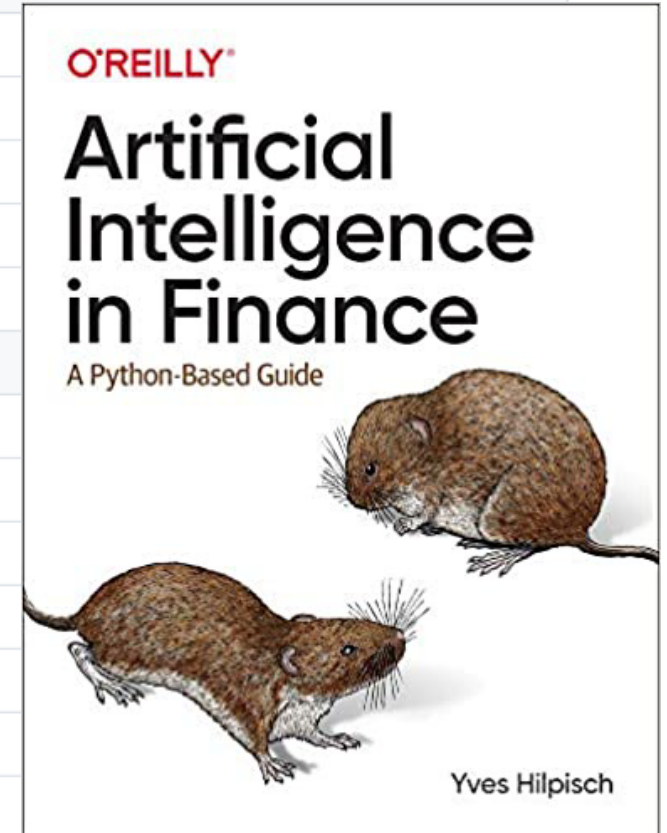
yhilpisch / aiif Public Notifications Star 98 Fork 77

<> 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

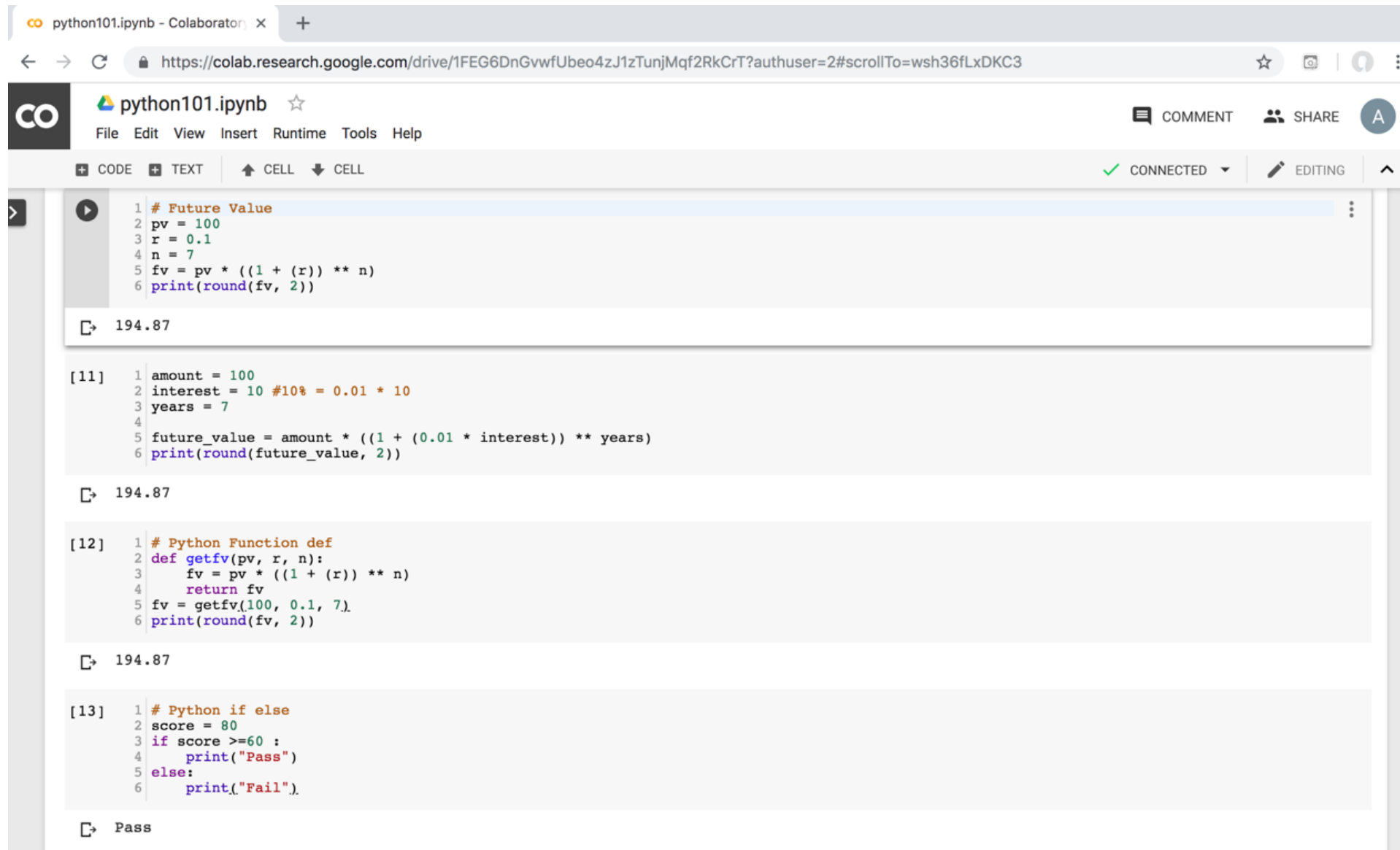
..	
oanda	Code updates for TF 2.3.
01_artificial_intelligence.ipynb	Code updates for TF 2.3.
02_superintelligence.ipynb	Code updates for TF 2.3.
03_normative_finance.ipynb	Code updates for TF 2.3.
04_data_driven_finance_a.ipynb	Initial commit.
04_data_driven_finance_b.ipynb	Initial commit.
05_machine_learning.ipynb	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>



The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: <https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT?authuser=2#scrollTo=wsh36fLxDKC3>. The notebook title is "python101.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with options for CODE, TEXT, CELL, and a status indicator showing "CONNECTED" and "EDITING".

The notebook contains four code cells, each followed by its output:

```
1 # Future Value
2 pv = 100
3 r = 0.1
4 n = 7
5 fv = pv * ((1 + (r)) ** n)
6 print(round(fv, 2))
```

194.87

```
[11] 1 amount = 100
     2 interest = 10 #10% = 0.01 * 10
     3 years = 7
     4
     5 future_value = amount * ((1 + (0.01 * interest)) ** years)
     6 print(round(future_value, 2))
```

194.87

```
[12] 1 # Python Function def
     2 def getfv(pv, r, n):
     3     fv = pv * ((1 + (r)) ** n)
     4     return fv
     5 fv = getfv(100, 0.1, 7)
     6 print(round(fv, 2))
```

194.87

```
[13] 1 # Python if else
     2 score = 80
     3 if score >=60 :
     4     print("Pass")
     5 else:
     6     print("Fail").
```

Pass

<https://tinyurl.com/aintpupython101>

# Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook interface. At the top, the notebook is titled "python101.ipynb" and has a star icon. The top navigation bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", along with "All changes saved". On the right side of the top bar, there are icons for "Comment", "Share", "Settings", and a user profile icon labeled "A".

The left sidebar contains a "Table of contents" panel with a search icon and navigation arrows. The contents are organized into sections:

- AI in Finance
  - Normative Finance and Financial Theories
    - Uncertainty and Risk**
    - Expected Utility Theory (EUT)
    - Mean-Variance Portfolio Theory (MVPT)
    - Capital Asset Pricing Model (CAPM)
    - Arbitrage Pricing Theory (APT)
  - Deep Learning for Financial Time Series Forecasting
  - Portfolio Optimization and Algorithmic Trading
    - Investment Portfolio Optimisation with Python
    - Efficient Frontier Portfolio Optimisation in Python
    - Investment Portfolio Optimization

The main content area shows a code editor with the following Python code:

```
1 import numpy as np
2
3 #The prices of the stock and bond today.
4 S0 = 10
5 B0 = 10
6 print('S0', S0)
7 print('B0', B0)
8
9 #The uncertain payoff of the stock and bond tomorrow.
10 S1 = np.array((20, 5))
11 B1 = np.array((11, 11))
12 print('S1', S1)
13 print('B1', B1)
14
15 #The market price vector
16 M0 = np.array((S0, B0))
```

<https://tinyurl.com/aintpuython101>

# Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share Settings A

RAM Disk Editing ^

**Table of contents**

- Data Driven Finance
  - 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
- Deep Learning for Financial Time Series Forecasting
- Portfolio Optimization and Algorithmic Trading
  - Investment Portfolio Optimisation with Python
  - Efficient Frontier Portfolio Optimisation in Python
  - Investment Portfolio Optimization

**Data Driven Finance**

**Financial Econometrics and Regression**

```
[18] 1 import numpy as np
      2
      3 def f(x):
      4     return 2 + 1 / 2 * x
      5
      6 x = np.arange(-4, 5)
      7 x

array([-4, -3, -2, -1, 0, 1, 2, 3, 4])
```

```
1 y = f(x)
2 y

array([ 0.00,  0.50,  1.00,  1.50,  2.00,  2.50,  3.00,  3.50,  4.00])
```

```
1 print('x', x)
2
3 print('y', y)
4
5 beta = np.cov(x, y, ddof=0)[0, 1] / x.var()
6 print('beta', beta)
```

# Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings A

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()
```

# Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share Settings A

RAM Disk Editing

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

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>



14  
12  
10  
8  
6  
4  
2  
0

2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020

Legend:  
— AAPL.O  
— MSFT.O  
— INTC.O  
— AMZN.O  
— GS.N  
— SPY  
— .SPX  
— .VIX  
— EUR=  
— XAU=  
— GDZ  
— GLD

# Python in Google Colab (Python101)



python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment

Share



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/aintpuython101>



# 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
```

# Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share ⚙️ A

RAM  Disk  Editing ^

+ Code + Text

## 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

## Algorithmic Trading

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

## Vectorized Backtesting

```
1 import os
2 import math
3 import numpy as np
4 import pandas as pd
5 from pylab import plt, mpl
6 plt.style.use('seaborn')
7 mpl.rcParams['savefig.dpi'] = 300
8 mpl.rcParams['font.family'] = 'serif'
9 pd.set_option('mode.chained_assignment', None)
10 pd.set_option('display.float_format', '{:.4f}'.format)
11 np.set_printoptions(suppress=True, precision=4)
12 os.environ['PYTHONHASHSEED'] = '0'
```

## Backtesting an SMA-Based Strategy

# Python in Google Colab (Python101)

The screenshot shows a Google Colab notebook titled 'python101.ipynb'. The interface includes a top navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' menus, along with 'Comment', 'Share', and 'Settings' icons. A 'Table of contents' sidebar on the left lists various topics under 'Algorithmic Trading', 'Risk Management', and 'Backtesting Cryptocurrency'. The main workspace displays two code cells. The first cell, titled 'Vectorized Backtesting', contains Python code for setting up the environment with imports for os, math, numpy, pandas, and matplotlib, and configuring plot styles. The second cell, titled 'Backtesting an SMA-Based Strategy', contains code to fetch data from a CSV file and display its information.

python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings A

RAM Disk Editing

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

## Vectorized Backtesting

```
1 import os
2 import math
3 import numpy as np
4 import pandas as pd
5 from pylab import plt, mpl
6 plt.style.use('seaborn')
7 mpl.rcParams['savefig.dpi'] = 300
8 mpl.rcParams['font.family'] = 'serif'
9 pd.set_option('mode.chained_assignment', None)
10 pd.set_option('display.float_format', '{:.4f}'.format)
11 np.set_printoptions(suppress=True, precision=4)
12 os.environ['PYTHONHASHSEED'] = '0'
```

## Backtesting an SMA-Based Strategy

```
[ ] 1 url = 'http://hilpisch.com/aiif_eikon_eod_data.csv'
     2 symbol = 'EUR='
     3 data = pd.DataFrame(pd.read_csv(url, index_col=0,
     4                               parse_dates=True).dropna()[symbol])
     5 data.info()
```

# Python in Google Colab (Python101)



python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment

Share



- 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>



# Python in Google Colab (Python101)

CO python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙️ A

RAM ✓ Disk

Editing ^

+ Code + Text

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

```
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>

Date	r	s	s_
2018-01	1.00	1.00	1.00
2018-04	1.03	1.08	1.07
2018-07	0.98	1.15	1.14
2018-10	0.95	1.22	1.21
2019-01	0.96	1.30	1.28
2019-04	0.94	1.25	1.24
2019-07	0.93	1.28	1.27
2019-10	0.92	1.30	1.28
2020-01	0.93	1.28	1.27

# Python in Google Colab (Python101)

The screenshot shows a Google Colab notebook interface. At the top left, the notebook is titled 'python101.ipynb' with a star icon. Below the title is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', followed by the text 'All changes saved'. On the top right, there are icons for 'Comment', 'Share', a settings gear, and a user profile icon with the letter 'A'. Below the menu bar, there are two progress indicators: 'RAM' and 'Disk', both with green checkmarks and progress bars. To the right of these are 'Editing' and a caret icon. The main area is divided into a left sidebar and a main content area. The sidebar, titled 'Table of contents', lists several sections: 'Algorithmic Trading', 'Vectorized Backtesting', 'Backtesting an SMA-Based Strategy', 'Backtesting a Daily DNN-Based Strategy', 'Backtesting an Intraday DNN-Based Strategy', 'Risk Management' (highlighted with a yellow bar), 'Trading Bot', 'Vectorized Backtesting', 'Event-Based Backtesting', 'Assessing Risk', 'Backtesting Risk Measures', 'Stop Loss', 'Trailing Stop Loss', 'Take Profit', 'Combinations', and 'Backtesting Cryptocurrency Bitcoin'. The main content area shows two code cells. The first cell is titled 'Risk Management' and contains the following code:

```
[ ] 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('mode.chained_assignment', None)
9 pd.set_option('display.float_format', '{:.4f}'.format)
10 np.set_printoptions(suppress=True, precision=4)
11 os.environ['PYTHONHASHSEED'] = '0'
```

The second cell is titled 'Trading Bot' and contains the following code:

```
[ ] 1 # import finance
2 # finance.py
3 # Finance Environment
4 #
5 # (c) Dr. Yves J. Hilpisch
6 # Artificial Intelligence in Finance
7 #
```

# Python in Google Colab (Python101)

The screenshot shows a Google Colab notebook interface. At the top left, the notebook is titled 'python101.ipynb' with a star icon. Below the title is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', followed by the text 'All changes saved'. On the right side, there are icons for 'Comment', 'Share', a settings gear, and a user profile icon with the letter 'A'. Below the menu bar, there are two tabs: '+ Code' and '+ Text'. On the right side of the notebook, there are indicators for 'RAM' and 'Disk' usage, a status 'Editing', and a vertical scroll bar. A 'Table of contents' sidebar is open on the left, listing various topics under 'Algorithmic Trading' and 'Risk Management'. The 'Event-Based Backtesting' item is highlighted. The main code cell contains the following Python code:

```
1 #import backtesting as bt
2
3 # backtesting.py
4 # Event-Based Backtesting
5 # --Base Class (1)
6 #
7 # (c) Dr. Yves J. Hilpisch
8 # Artificial Intelligence in Finance
9 #
10
11 class BacktestingBase:
12     def __init__(self, env, model, amount, ptc, ftc, verbose=False):
13         self.env = env
14         self.model = model
15         self.initial_amount = amount
16         self.current_balance = amount
17         self.ptc = ptc
18         self.ftc = ftc
19         self.verbose = verbose
20         self.units = 0
21         self.trades = 0
22
23     def get_date_price(self, bar):
24         ''' Returns date and price for a given bar.
25         ...
```

# Python in Google Colab (Python101)

The screenshot shows a Google Colab notebook titled "python101.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", along with a "Comment" button and a "Share" button. On the left, a "Table of contents" sidebar lists various sections: Algorithmic Trading, Risk Management, and Backtesting Cryptocurrency Bitcoin. The "Combinations" section is highlighted. The main code cell contains a single line of Python code: `1 tb.backtest_strategy(sl=0.015, tsl=None, 2 tp=0.0185, wait=5)`. The output of this code is a series of text-based logs for a backtest, starting with "2018-01-17 | \*\*\* START BACKTEST \*\*\*" and "2018-01-17 | current balance = 10000.00". The logs show several trade entries with their respective stop loss and take profit values, such as "\*\*\* STOP LOSS (SHORT | -0.0203) \*\*\*" and "\*\*\* TAKE PROFIT (SHORT | 0.0189) \*\*\*".

python101.ipynb ☆  
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

RAM Disk Editing

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

Combinations

```
1 tb.backtest_strategy(sl=0.015, tsl=None,
2 tp=0.0185, wait=5)
```

```
=====
2018-01-17 | *** START BACKTEST ***
2018-01-17 | current balance = 10000.00
=====
-----
*** STOP LOSS (SHORT | -0.0203) ***
-----
*** STOP LOSS (SHORT | -0.0152) ***
-----
*** TAKE PROFIT (SHORT | 0.0189) ***
-----
*** TAKE PROFIT (SHORT | 0.0219) ***
-----
*** TAKE PROFIT (SHORT | 0.0192) ***
-----
*** STOP LOSS (LONG | -0.0154) ***
-----
*** TAKE PROFIT (SHORT | 0.0214) ***
-----
*** STOP LOSS (SHORT | -0.0158) ***
-----
*** TAKE PROFIT (SHORT | 0.0223) ***
-----
*** STOP LOSS (SHORT | -0.0162) ***
=====
```



# Python in Google Colab (Python101)



python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings A

RAM Disk Editing

Navigation icons: up, down, refresh, comment, edit, copy, trash, menu

## 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

## Backtesting Cryptocurrency Bitcoin

- Financial Functions (ffn): <https://pmorrisette.github.io/ffn/>
- backtesting.py: <https://kernc.github.io/backtesting.py/>

```
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))
```

# Python in Google Colab (Python101)

CO python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings A

RAM  Disk  Editing ^

+ Code + Text

↑ ↓ ↻ ⌨ ⚙ 📄 🗑 ⋮

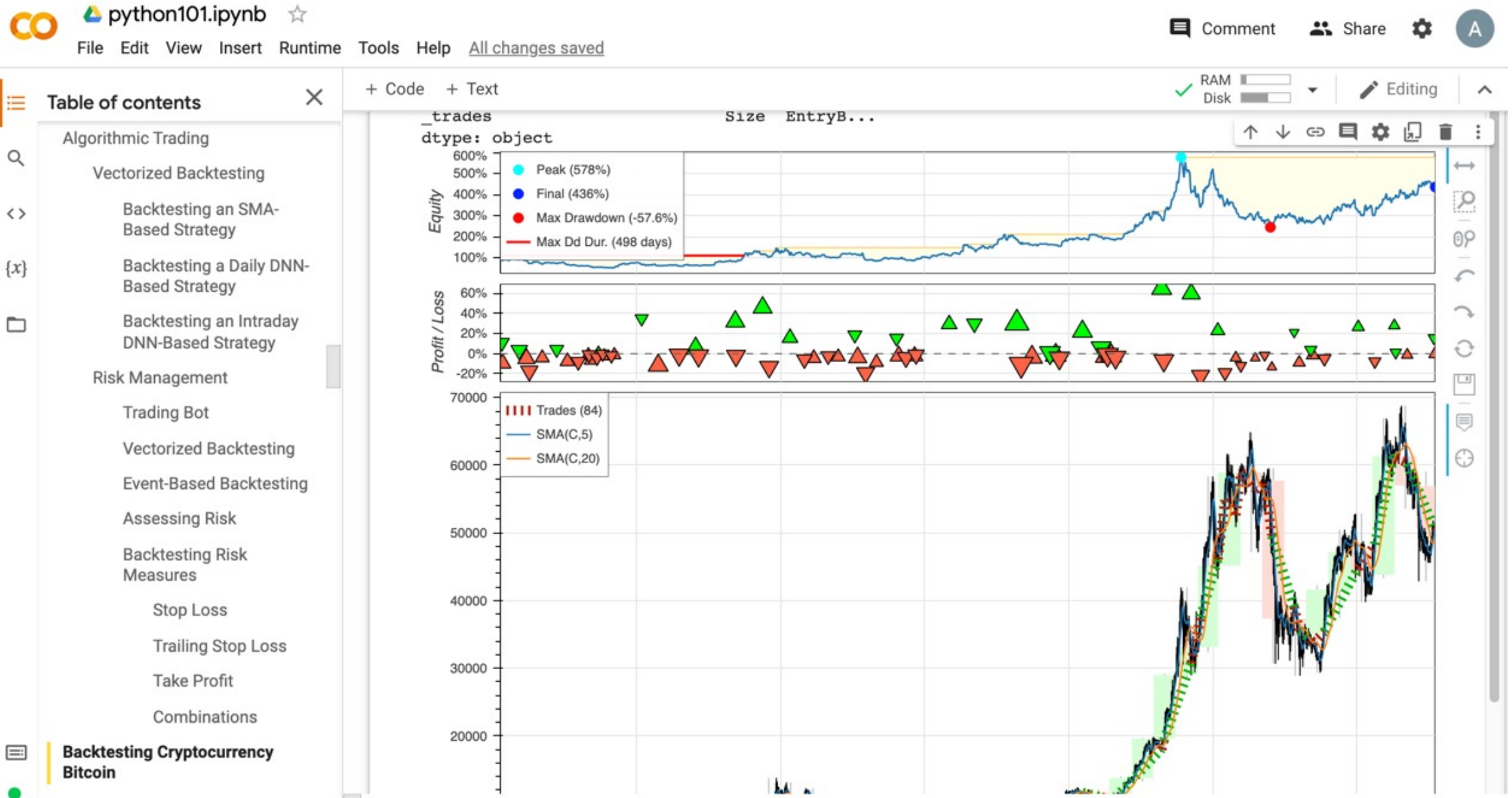
**Table of contents** X

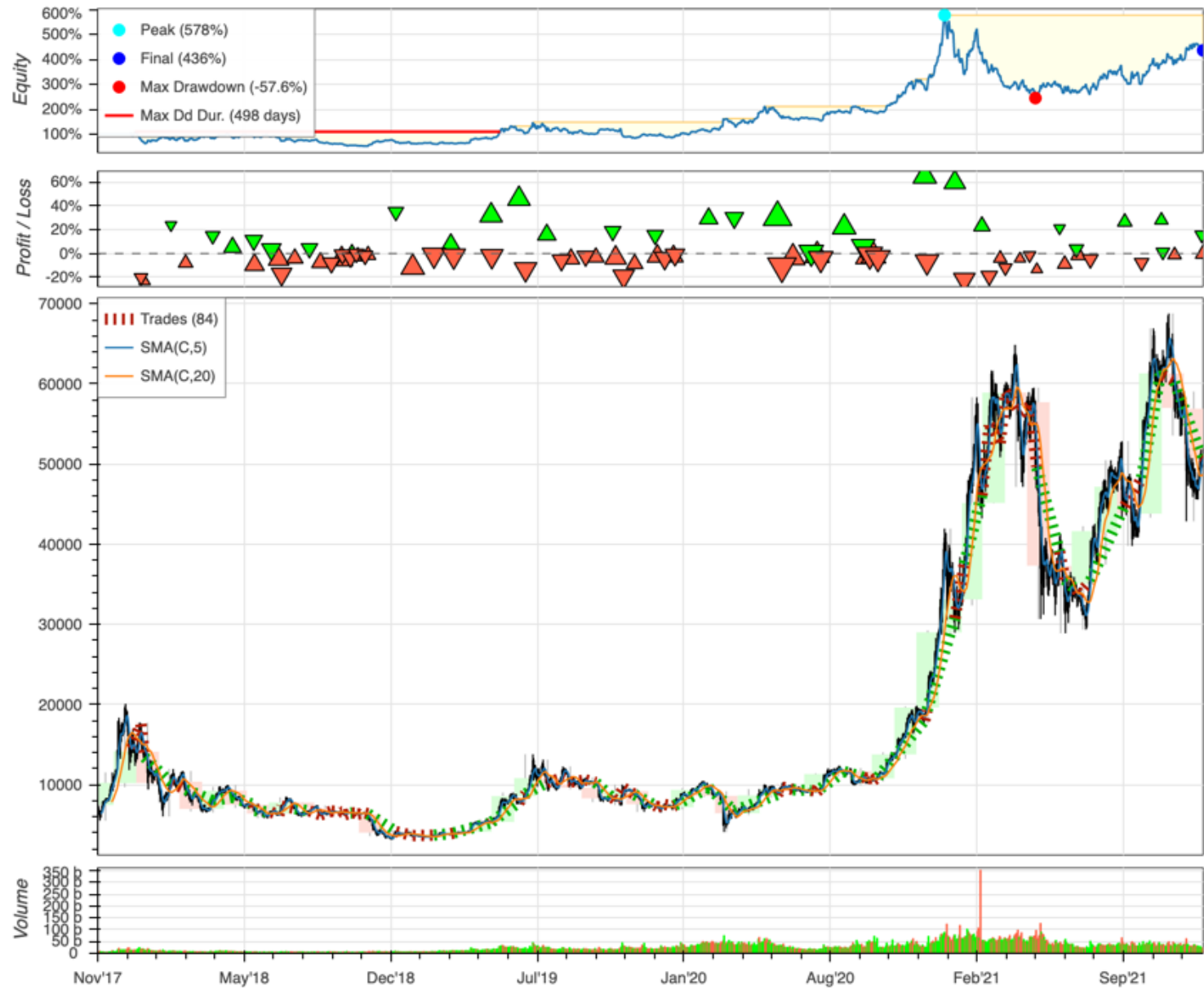
- Algorithmic Trading
  - Vectorized Backtesting
    - Backtesting an SMA-Based Strategy
    - Backtesting a Daily DNN-Based Strategy
    - Backtesting an Intraday DNN-Based Strategy
  - Risk Management
    - Trading Bot
    - Vectorized Backtesting
    - Event-Based Backtesting
    - Assessing Risk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

**btcusd returns box**

The figure is a box plot titled "btcusd returns box". The vertical axis is labeled "btcusd" and has tick marks at -0.4, -0.3, -0.2, -0.1, 0, 0.1, and 0.2. The plot shows a distribution of data points represented by blue dots. A central box is drawn around the median, which is slightly above 0. Whiskers extend from the box to the minimum and maximum values of the data, which are approximately -0.08 and 0.08. There are several outliers, with the most prominent ones at approximately 0.25 and -0.35.

# Python in Google Colab (Python101)





# Summary

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

# 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 (2019), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd 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.
- Min-Yuh Day (2022), Python 101, <https://tinyurl.com/aintpupython101>