# How Machine Learning Can Improve Trading Bot Profitability on Crypto Markets

## Applied Research Using QuantOffice Cloud

Authors: **Aliaksandr Yuretski**, **Sviatlana Staleuskaya**, **Isaac Gorelik**

Published: **09 October 2021** | EPAM Deltix — QuantOffice Cloud Team

Tags: Trading · Machine Learning · Algorithms · Crypto Markets

### Abstract

Machine learning methods are widely adopted for algorithmic trading and are becoming a staple product in the modern financial market. The purpose of this research is to evaluate a filtration method to improve a trading strategy performance by eliminating unnecessary trade signals. The implementation is inspired by Marcos Lopez de Prado's ideas outlined in his book "Advances in Financial Machine Learning". Thorough back-testing experiments carried out in QuantOffice Cloud have revealed that this approach can reduce the number of unprofitable trades sent by a strategy significantly, thereby mitigating losses in the form of exchange commissions.

| 6,414 | 3,273 | 1.88 | $2,211 | 5 |
|---|---|---|---|---|
| Baseline Trades | SVM Trades (–49%) | SVM Info Ratio | SVM Max Drawdown | Classifiers Tested |

# 1. Introduction

The primary purpose of this experiment is to determine whether machine learning techniques can improve the performance of a momentum-based trading strategy. The baseline strategy is adapted from *Dissecting Investment Strategies in the Cross-Section and Time Series* by Baz et al. (2015). The primary goal is to improve performance by training an algorithm to filter out detrimental or unnecessary trades which do not generate returns.

The secondary goal is to explore **QuantOffice Cloud** capabilities as a Jupyter Lab-based strategy development and testing studio, particularly in conjunction with machine learning technologies. QuantOffice Cloud provides a powerful set of APIs and a development environment allowing researchers to focus on core activities such as design and prototyping, with the ability to run backtested strategies live on real or paper accounts.

# 2. Baseline Trading Strategy

The strategy generates trading signals in the range (–1, 1) by averaging three values derived from differences of Exponential Moving Averages (EMAs) with various time periods. For this experiment, a uniform bet size and discrete signals are used: the signal acquires a value of ±1 when its magnitude exceeds a threshold; otherwise it is set to 0.

The strategy consists of **six concurrent strategy instances**, each tuned to maximise performance. It was developed in Python and backtested on the QuantOffice Cloud platform. Back-testing covered **Ethereum (ETHUSD), Litecoin (LTCUSD), and Bitcoin (BTCUSD)** price data from the Kraken exchange over a two-year period (March 2019 – October 2021).

## Strategy Configuration (Jupyter Notebook Parameters)

```
symbols = 'BTCUSD LTCUSD ETHUSD'
price_stream = 'KRAKEN_BARS'
bar_size = BarSize(BarUOM.Minute, 1)
start_time = "2019-03-01T00:00:00"
end_time   = "2021-10-19T00:00:00"

input_parameters.initial_cap      = 18000
input_parameters.bet_size         = 1000  # $1,000 per bet
input_parameters.holding_period   = 60*24 # bars

PortfolioExecutor.instances = [
    StrategyInstance(4*60,  24*60,  0.99961, "p1"),
    StrategyInstance(60,    24*60,  0.99845, "p2"),
    StrategyInstance(24*60, 5*24*60,0.99999, "p3"),
    StrategyInstance(5,     5*24*60,0.99967, "p4"),
    StrategyInstance(60,    5*24*60,0.99994, "p5"),
    StrategyInstance(15,    5*24*60,0.99999, "p6"),
]
```

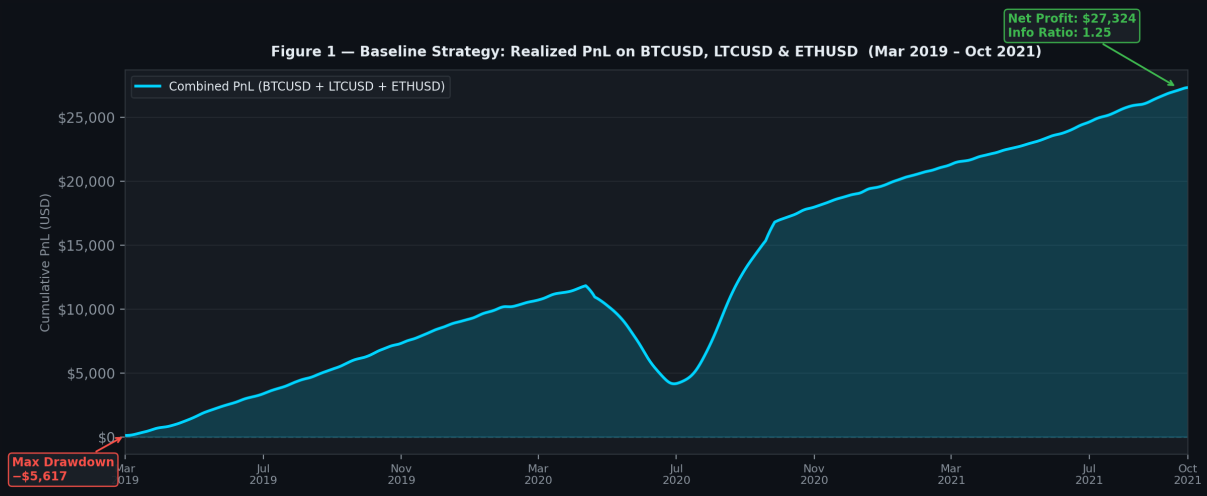


***Figure 1.*** *Strategy's realized cumulative PnL backtested on BTCUSD, LTCUSD, and ETHUSD from March 2019 to October 2021. The strategy achieved a net profit of $27,323.56 over the period, with a significant drawdown of –$5,617 coinciding with the March 2020 market crash.*

## Table 1: Baseline Strategy Performance Metrics

| Parameter | All Trades | Long Trades | Short Trades |
|---|---|---|---|
| Net Profit/Loss ($) | 27,323.56 | 27,757.08 | –433.52 |
| Total Profit ($) | 125,140.8 | 67,898.73 | 57,242.03 |
| Total Loss ($) | –97,817.2 | –40,141.6 | –57,675.6 |
| Max Drawdown ($) | –5,617.29 | –3,651.31 | –9,881.1 |
| Return/Drawdown Ratio | 4.8642 | 7.602 | –0.0439 |
| Max Drawdown Duration | 271 day(s) | 139 day(s) | 584 day(s) |
| Information Ratio | 1.2509 | 1.8731 | –0.0184 |
| Total Trades # | 6,414 | 3,169 | 3,245 |
| Profitable Trades Ratio | 0.5123 | 0.5589 | 0.4669 |
| Winning Trades # | 3,286 | 1,771 | 1,515 |
| Losing Trades # | 3,128 | 1,398 | 1,730 |
| Average Trade ($) | 4.26 | 8.76 | –0.13 |
| Avg Profit/Trade (bps) | 42.51 | 87.21 | –1.34 |
| Average Winning Trade ($) | 38.08 | 38.34 | 37.78 |
| Average Losing Trade ($) | –31.27 | –28.71 | –33.34 |
| Avg Win/Avg Loss Ratio | 1.2178 | 1.3352 | 1.1333 |
| Max Consecutive Winners | 38 | 29 | 34 |
| Max Consecutive Losers | 49 | 38 | 39 |

# 3. Machine Learning Application

The experiment's goal is to **train a classifier to filter out detrimental trading signals** generated by the baseline strategy. The QuantOffice library provides access to the classifier in conjunction with the existing strategy, enabling the model to be trained as data accumulates. The architecture follows a modular pipeline: Labeling → Features → Estimators → Prediction.

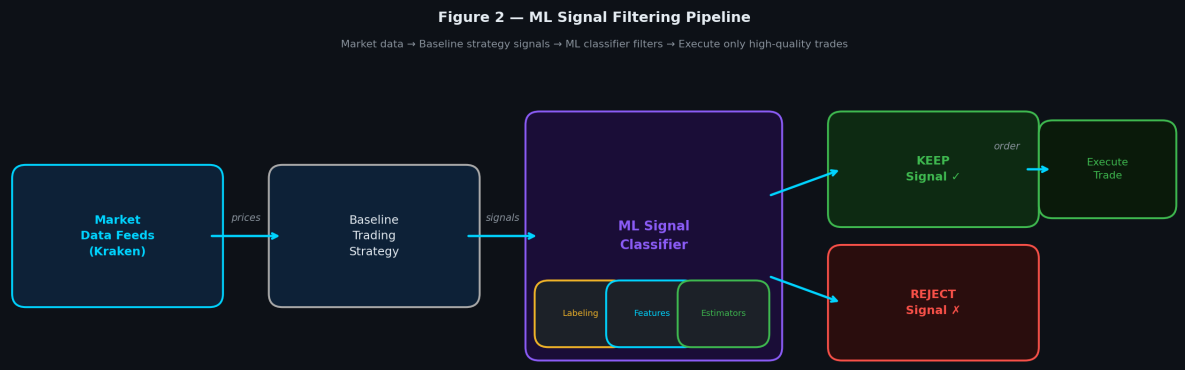


***Figure 2.*** *Overview of the machine learning signal filtering pipeline. Market price data feeds into the baseline trading strategy. The resulting signals are passed to the ML classifier, which evaluates each signal using trained features and either approves it for execution or rejects it.*

## 3.1 Labeling

The training set is labeled using a modification of the **Marcos Lopez de Prado three-barrier method** with a 24-hour horizon. Strategy open position points are treated as learning objects. A signal receives a label of ±1 (depending on the sign of the return) if the absolute return exceeds a threshold; otherwise the label is 0.

## 3.2 Feature Engineering

The **Features** module handles extraction, normalisation, and dimension reduction of factors derived from price data of each instrument. Features include:

- Momentum-type series across various time intervals
- Logarithmic returns with different lags
- Volatility measures calculated with different lag windows
- Autoencoder outputs for non-linear dimensionality reduction
- Principal Component Analysis (PCA) for linear reduction

## 3.3 Classifiers (Estimators)

The **Estimators** module contains five built-in classifiers. Additional classifiers can be integrated from scikit-learn or implemented by the user:

| Classifier | Abbr. | Notes |
|---|---|---|
| Random Forest | RF | Ensemble method; robust to overfitting |

| | | |
|---|---|---|
| XGBoost | XGB | Gradient boosting; strong baseline |
| Multilayer Perceptron | MLP | Neural network; flexible representation |
| Support Vector Machine | SVM | **Best performer in this study** |
| K-Nearest Neighbors | kNN | Instance-based; simple non-parametric |

### 3.4 Training & Prediction Cycle

The model requires a minimum of **30 consecutive days** of price data to extract optimal factors. Once trained, the model is used for **24-hour prediction windows**, after which it is retrained using the most recent 30-day rolling window. The cycle repeats until back-testing ends.

During training, data is randomly split into training and test sets. **5-fold cross-validation** is used to find parameters that minimise total validation error. Predictions are only made when the classifier provides a non-zero buy or sell signal.

#### Training Cycle Example

- **31 January:** Model trained on Jan 1–30 data. Used to predict Jan 31.
- **1 February:** Model retrained on Jan 2–31. Used to predict Feb 1.
- **Cycle continues** rolling forward one day at a time through the back-test period.

# 4. Results

Results for all tested classifiers are presented below. **Support Vector Machine (SVM)** and **Multilayer Perceptron (MLP)** showed the best performance overall. Signal filtration reduces the total number of trades while maintaining or improving average profit in basis points and the information ratio.

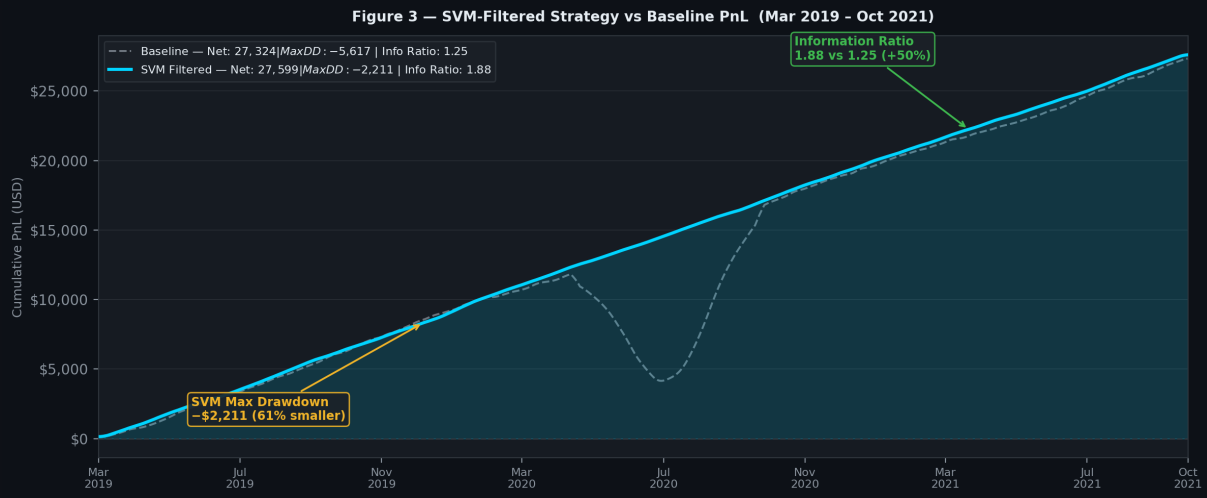


***Figure 3.*** *Cumulative PnL of the SVM-filtered strategy vs the baseline from March 2019 to October 2021. The SVM filter reduces drawdown by 61% (from –$5,617 to –$2,211) while maintaining nearly identical net profit ($27,599 vs $27,323). The smoother equity curve reflects fewer, higher-quality trades.*

## Table 2: Classifier Comparison — Key Performance Metrics

| Parameter | Base | kNN | MLP | RF | SVM ★ | XGBoost |
|---|---|---|---|---|---|---|
| Net Profit/Loss ($) | 27,323.56 | 24,443.6 | 24,967.18 | 23,492.63 | 27,598.96 | 22,324.94 |
| Total Profit ($) | 125,140.8 | 92,431.5 | 72,168.0 | 91,177.2 | 70,799.8 | 104,864.0 |
| Total Loss ($) | –97,817.2 | –67,987.9 | –47,200.8 | –67,684.5 | –43,200.8 | –82,539.0 |
| Max Drawdown ($) | –5,617.29 | –3,653.57 | –2,847.7 | –4,092.29 | –2,211.13 | –5,295.64 |
| Return/Drawdown Ratio | 4.8642 | 6.6903 | 8.7675 | 5.7407 | 12.4819 | 4.2157 |
| Max Drawdown Duration | 271d | 165d | 190d | 128d | 189d | 237d |
| Information Ratio | 1.2509 | 1.4392 | 1.6941 | 1.4249 | 1.8807 | 1.1933 |
| All Trades # | 6,414 | 4,739 | 3,415 | 4,669 | 3,273 | 5,484 |
| Profitable Trades % | 51.23% | 51.51% | 53.88% | 51.08% | 54.48% | 50.55% |
| Winning Trades # | 3,286 | 2,441 | 1,840 | 2,385 | 1,783 | 2,772 |
| Losing Trades # | 3,128 | 2,298 | 1,575 | 2,284 | 1,490 | 2,712 |
| Avg Trade ($) | 4.26 | 5.16 | 7.31 | 5.03 | 8.43 | 4.07 |
| Avg Profit/Trade (bps) | 42.51 | 51.45 | 72.85 | 50.16 | 84.03 | 40.62 |
| Avg Win/Avg Loss Ratio | 1.2178 | 1.2799 | 1.3088 | 1.29 | 1.3695 | 1.243 |
| Max Conseq. Winners | 38 | 32 | 29 | 29 | 29 | 35 |
| Max Conseq. Losers | 49 | 31 | 26 | 29 | 25 | 35 |

*★ SVM column highlighted. SVM achieved the best Return/Drawdown ratio (12.48×), highest Information Ratio (1.88), lowest Max Drawdown (–$2,211), and highest average profit per trade (84.03 bps).*

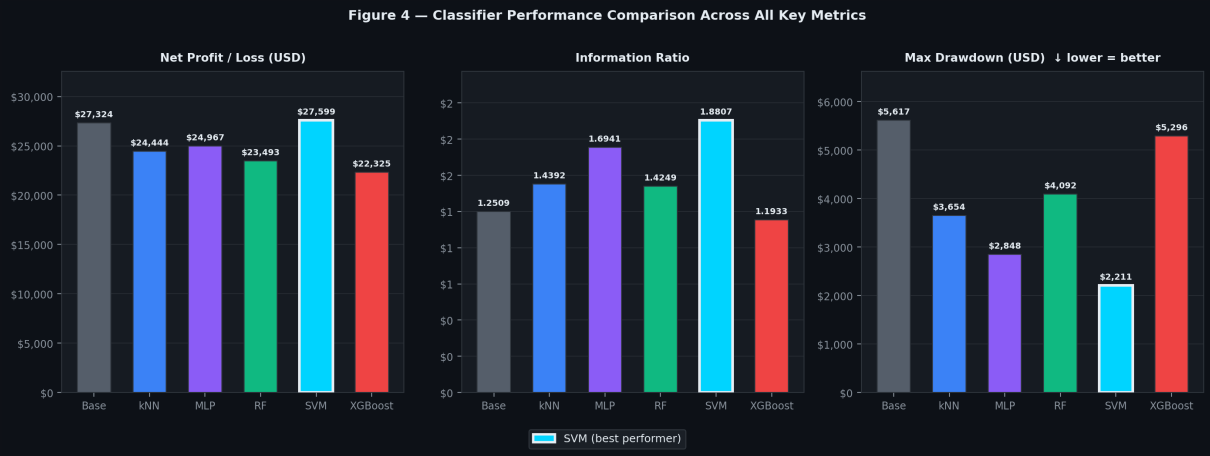


***Figure 4.*** *Side-by-side comparison of all classifiers across three key metrics: Net Profit/Loss, Information Ratio, and Maximum Drawdown. SVM (highlighted with white border) leads on Information Ratio and Max Drawdown reduction, confirming its superiority for this use case.*

# 5. Conclusion

This research has demonstrated that machine learning can meaningfully improve the quality — rather than the raw quantity — of trading signals in a momentum-based crypto strategy. By training a classifier to reject detrimental signals, the strategy achieves better risk-adjusted returns with fewer trades.

| Key Finding | Baseline | Best (SVM) | Improvement |
|---|---|---|---|
| Max Drawdown | –$5,617 | –$2,211 | 61% reduction |
| Information Ratio | 1.2509 | 1.8807 | +50.3% |
| Return/Drawdown | 4.86× | 12.48× | +156.8% |
| Trade Count | 6,414 | 3,273 | –49% (quality ↑) |
| Avg Profit/Trade (bps) | 42.51 | 84.03 | +97.7% |

In conclusion, this study provides supportive evidence that machine learning techniques — particularly SVM and MLP — can enhance trading strategy performance by introducing a signal filtration layer. The QuantOffice Cloud platform proved well-suited for this type of research, combining robust backtesting infrastructure with flexible Python-based ML integration.

# References


[1] Baz, J., Granger, N., Harvey, C. R., Le Roux, N., Rattray, S. (2015). Dissecting Investment Strategies in the Cross Section and Time Series. SSRN Working Paper 2695101.

[2] Dixon, M.F., Klabjan, D., & Bang, J.H. (2015). Implementing deep neural networks for financial market prediction on the Intel Xeon Phi. Proceedings of the 8th Workshop on High-Performance Computational Finance.

[3] Lopez de Prado, M. (2018). Advances in Financial Machine Learning. John Wiley & Sons.

[4] QuantOffice Cloud Platform. Available at: https://quantoffice.cloud/

[5] Scikit-learn: Machine Learning in Python. Available at: https://scikit-learn.org/