# **Architecting a Modern Cryptocurrency Analysis Bot: A Quantitative Approach to Signal Generation on Coinbase**

## **Section 1: The Data Backbone - Sourcing and Managing Coinbase Market Data**

The efficacy of any quantitative analysis system is fundamentally determined by the quality, timeliness, and reliability of its input data. For a cryptocurrency analysis bot focused exclusively on Coinbase, establishing a robust and efficient data pipeline is the foundational step upon which all subsequent strategies and risk management frameworks will be built. The architectural decisions made at this stage—specifically the choice of Application Programming Interface (API) and data handling methodology—will directly influence the bot's performance, accuracy, and capacity for future expansion.

### **1.1. Primary Connection: The Official Coinbase Advanced Trade Python SDK**

The most direct and recommended data entry point for a bot designed to operate solely within the Coinbase ecosystem is the official coinbase-advanced-py Software Development Kit (SDK).1 This library is purpose-built by Coinbase to interact with its modern

**Advanced Trade API**, which has officially superseded the older, now-deprecated Coinbase Pro API.3 Opting for the official SDK provides several distinct advantages. It guarantees direct, unfettered access to the full spectrum of platform-specific features and endpoints, ensures optimal performance by minimizing abstraction layers, and aligns the bot's development with Coinbase's official product roadmap, reducing the risk of future compatibility issues.

The coinbase-advanced-py SDK is architected to support two primary modes of data interaction: a RESTClient for synchronous, request-response communication and a WSClient for asynchronous, real-time data streaming via WebSockets.1 For the initial construction and backtesting of the analysis bot, the

RESTClient is indispensable. It will be used to fetch historical market data, specifically the Open, High, Low, Close, and Volume (OHLCV) data points that form the basis of candlestick charts. The getProductCandles or its public counterpart, get\_public\_candles, will be the primary function for this task, allowing the retrieval of historical data at various granularities (e.g., 1-hour, 4-hour, 1-day candles).2 Once the bot's strategies are developed, the

WSClient can be integrated to receive live market data, enabling the bot to perform real-time analysis and generate timely notifications as new price information becomes available. Authentication for private endpoints is managed through modern Coinbase Developer Platform (CDP) API keys, which offer a more secure and robust mechanism than legacy key systems.1

A critical lesson from the evolution of cryptocurrency platforms is the constant state of flux in their APIs. The sunsetting of the widely used Coinbase Pro API serves as a significant case study.4 Developers searching for tools may also encounter older, seemingly official libraries on platforms like PyPI, such as the

coinbase package, which has not been updated since 2018—an eternity in the fast-moving digital asset space.6 Relying on such unmaintained or deprecated tools introduces a substantial technical risk, as they may cease to function without warning or provide inaccurate data. Therefore, a core principle of development must be to exclusively target the current, actively maintained Coinbase Advanced Trade API and its official

coinbase-advanced-py SDK.

### **1.2. A Versatile Alternative: The CCXT Library**

While the official SDK is the premier choice for a single-platform focus, a more sophisticated and forward-thinking architectural approach involves using the CryptoCurrency eXchange Trading Library (CCXT). CCXT is a powerful, open-source Python library that provides a standardized, unified interface for interacting with over 100 cryptocurrency exchanges, including Coinbase.7

The primary advantage of CCXT is abstraction. By building the bot's analytical logic to interface with CCXT's standardized methods, the system becomes exchange-agnostic. This means that if the need arises to expand the bot's scope to include other exchanges like Binance or Kraken in the future, the transition would require minimal code changes, primarily limited to the initial exchange connection parameters.10 This design choice insulates the core strategic logic from the specifics of the data source, creating a more resilient, portable, and future-proof system.

For implementation, CCXT simplifies data retrieval through its unified API calls. The fetch\_ohlcv method, for example, is used to retrieve historical candlestick data from any supported exchange, including Coinbase, using the same function signature.11 The library internally manages the complexities of each exchange's specific authentication protocols, endpoint nuances, and rate-limiting rules, allowing the developer to focus on strategy implementation.7 The trade-off for this versatility is that

CCXT may not support every niche, Coinbase-specific endpoint that falls outside the common standard implemented across most exchanges. However, for the core requirement of fetching OHLCV and market data, its capabilities are more than sufficient.

The choice between the official Coinbase SDK and CCXT is a critical architectural decision. It represents a trade-off between deep, specialized integration with a single platform versus building a more generalized, adaptable system for the long term. For initial development focused purely on Coinbase, the official SDK offers a lower learning curve. However, structuring the data access layer of the application in a modular way—so that a CCXT-based data provider could be swapped in later—is a professional best practice that aligns with modern software design principles.

### **1.3. Data Handling and Storage**

Regardless of the API used, the raw data returned (typically in JSON format) must be transformed into a structured format suitable for quantitative analysis. The pandas library is the undisputed industry standard in Python for manipulating and analyzing time-series data.15

The standard workflow involves making an API call to fetch the OHLCV data, which is often returned as a list of lists or a list of dictionaries. This raw data is then loaded into a pandas DataFrame. A crucial step is the conversion of the timestamp provided by the API—often a Unix timestamp in milliseconds or seconds—into a pandas datetime object. This datetime column should then be set as the DataFrame's index. This creates a standardized, time-indexed data structure with columns for Open, High, Low, Close, and Volume. This format is the required input for nearly all technical analysis libraries (like TA-Lib) and machine learning frameworks, ensuring seamless integration with the subsequent stages of the analysis pipeline.11

### **1.4. API/Library Comparison for Coinbase Data Access**

To aid in the architectural decision-making process, the following table provides a direct comparison of the key attributes of the official Coinbase Advanced SDK and the CCXT library.

| Feature | Coinbase Advanced SDK (coinbase-advanced-py) | CCXT Library |
| --- | --- | --- |
| **Official Support** | Yes, maintained by Coinbase. | No, community-driven open source. |
| **Access to Coinbase-Specific Features** | Full access to all Advanced Trade API endpoints. | Partial access, focused on standardized features. |
| **Multi-Exchange Capability** | No, designed exclusively for Coinbase. | Yes, supports over 100 exchanges. |
| **Code Portability** | Low, logic is tied to Coinbase's API structure. | High, logic is abstracted from the exchange. |
| **Learning Curve** | Low, straightforward for a single platform. | Medium, requires understanding the abstraction layer. |
| **Long-Term Flexibility** | Lower, locked into the Coinbase ecosystem. | Higher, adaptable to market and exchange changes. |

## **Section 2: Foundational Strategy Development - Technical Analysis**

Technical analysis forms the bedrock of most algorithmic trading and analysis systems. It operates on the principle that historical price and volume data contain patterns that can indicate future price movements. By implementing a suite of classic, time-tested technical indicators, the bot can generate a baseline of clear, interpretable signals about market dynamics. These indicators are reactive, meaning they describe the current or recent state of the market, and are essential for identifying trends, momentum, and potential reversal points.

### **2.1. Trend Following: Moving Average (MA) Crossovers**

One of the most fundamental and widely used technical analysis strategies is the moving average (MA) crossover. This trend-following technique utilizes two moving averages of an asset's price, each calculated over a different time period (lookback window), to identify the prevailing trend direction and signal changes in that trend.17 A common configuration involves a short-term MA (e.g., 50-period) and a long-term MA (e.g., 200-period).

The signals are generated when these two lines intersect:

* **Golden Cross (Bullish Signal):** Occurs when the short-term MA crosses above the long-term MA. This event suggests that short-term momentum is accelerating upwards faster than long-term momentum, indicating the potential start or continuation of an uptrend.19
* **Death Cross (Bearish Signal):** Occurs when the short-term MA crosses below the long-term MA. This indicates that short-term momentum is waning and may signal the beginning of a downtrend.

There are several types of moving averages, with the two most common being the Simple Moving Average (SMA) and the Exponential Moving Average (EMA). The SMA gives equal weight to all prices in the lookback period, while the EMA gives more weight to more recent prices, making it more responsive to new information.18 In Python, SMAs can be calculated directly on a pandas DataFrame's 'Close' price column using the

.rolling(window=...).mean() method.17 EMAs are calculated using the

.ewm(span=..., adjust=False).mean() method.21 For a more comprehensive and optimized implementation, specialized libraries such as

TA-Lib or pandas-ta are highly recommended, as they provide functions for a vast array of technical indicators.15

### **2.2. Momentum Oscillation: Relative Strength Index (RSI)**

The Relative Strength Index (RSI) is a momentum oscillator that measures the speed and magnitude of recent price changes to evaluate overbought or oversold conditions in the price of an asset.22 The RSI is displayed as a line graph that fluctuates between 0 and 100.23

The primary use of the RSI is to identify market extremes:

* **Overbought:** An RSI reading above 70 traditionally indicates that an asset may be overbought, meaning its price has risen too quickly and may be due for a corrective pullback. A potential sell signal is generated when the RSI line crosses back down below the 70 level.24
* **Oversold:** An RSI reading below 30 suggests that an asset may be oversold, having fallen too sharply, and could be poised for a rebound. A potential buy signal occurs when the RSI crosses back up above the 30 level.24

A more advanced use of the RSI involves identifying **divergence**. A bearish divergence occurs when the asset's price reaches a new high, but the RSI fails to reach a new high, suggesting that the upward momentum is weakening and a reversal may be imminent. Conversely, a bullish divergence happens when the price makes a new low, but the RSI makes a higher low, indicating that the downward momentum is fading.24 The

TA-Lib library provides a direct and efficient talib.RSI() function that can be applied to a pandas Series of closing prices to calculate the indicator.15

### **2.3. Trend & Momentum Synergy: Moving Average Convergence Divergence (MACD)**

The Moving Average Convergence Divergence (MACD) indicator is a versatile tool that captures both trend-following and momentum aspects of a market. It is composed of three main components, all derived from the asset's price history 27:

* **MACD Line:** The difference between two Exponential Moving Averages (EMAs), typically the 12-period EMA and the 26-period EMA.
* **Signal Line:** A 9-period EMA of the MACD line itself. This line is smoother than the MACD line and serves to generate the primary trading signals.
* **MACD Histogram:** The difference between the MACD line and the Signal line. The histogram gives a visual representation of the momentum of the trend. When the histogram is positive and growing, bullish momentum is increasing. When it is negative and falling, bearish momentum is increasing.

The core trading signals from the MACD are generated by crossovers:

* **Bullish Crossover:** When the MACD line crosses above the Signal line, it is considered a buy signal, indicating a potential shift to upward momentum.30
* **Bearish Crossover:** When the MACD line crosses below the Signal line, it is a sell signal, suggesting a potential shift to downward momentum.

Like the RSI, the TA-Lib library offers a straightforward function that calculates all three components of the MACD (the MACD line, the signal line, and the histogram) from a series of closing prices.15

A crucial principle in developing a robust analysis bot is that technical indicators should not be used in isolation. A signal generated by a single indicator can often be "market noise." A high-quality, actionable signal is one that is confirmed by a **confluence** of events across multiple, ideally uncorrelated, indicators. For instance, a simple MA crossover can produce numerous false signals in a sideways, non-trending market. Similarly, the RSI can remain in the "overbought" territory for an extended period during a strong uptrend, leading to premature sell signals if acted upon alone.22 Therefore, the bot's logic must be built to identify these moments of confluence. A high-conviction "BUY" notification, for example, could be defined by the simultaneous occurrence of: (1) a Golden Cross (MA Crossover), (2) an RSI reading that is rising but still below the 70 level (indicating upward momentum without being overextended), and (3) a positive and expanding MACD histogram (confirming bullish momentum). This multi-condition logic dramatically improves the signal-to-noise ratio and is a cornerstone of professional strategy design.

Furthermore, the standard parameters for these indicators (e.g., 14 for RSI, 12/26/9 for MACD, 50/200 for MAs) were historically developed for traditional stock markets, which operate on a different schedule and often exhibit lower volatility than the 24/7 cryptocurrency markets. A "good" strategy is one that is demonstrably effective on historical data for the specific asset and timeframe being analyzed. The profitability of these strategies is highly sensitive to their input parameters. Consequently, the development process for the bot must incorporate a rigorous backtesting phase where various parameter combinations are systematically tested to discover the optimal settings for each cryptocurrency. Frameworks like vectorbt or Backtrader are specifically designed for this type of parameter optimization and strategy validation.15 Deploying a strategy with default, un-tested parameters is a naive approach that fails to account for the unique characteristics of the crypto market.

## **Section 3: Advanced Strategy Development - Machine Learning and Quantitative Models**

Moving beyond the reactive nature of traditional technical indicators, a modern analysis bot can incorporate proactive, predictive models based on machine learning (ML) and statistical forecasting. This represents a significant leap in sophistication, aiming not just to describe the current market state but to forecast its future direction. This approach requires a higher degree of statistical rigor, a robust data pipeline, and a keen awareness of potential pitfalls such as overfitting and concept drift.

### **3.1. Statistical Forecasting: ARIMA Models**

The Auto-Regressive Integrated Moving Average (ARIMA) model is a class of statistical models for analyzing and forecasting time-series data.32 It is a powerful tool for capturing linear relationships within a time series and is well-suited for short-term price forecasting. The name reflects its three core components 33:

* **Auto-Regressive (AR):** A regression of the variable against its own past (lagged) values. This component assumes that future values have a linear dependency on past values.
* **Integrated (I):** The use of differencing of the raw observations to make the time series stationary. Stationarity, where statistical properties like mean and variance are constant over time, is a prerequisite for many time-series models.
* **Moving Average (MA):** A regression of the variable against the past forecast errors (residuals). This component helps to model shocks or unexpected events in the time series.

Implementation of an ARIMA model in Python is typically done using the statsmodels library. The process involves several steps: first, testing the price series for stationarity using statistical tests like the Augmented Dickey-Fuller (ADF) test.32 If the series is non-stationary, it must be differenced until stationarity is achieved. Next, the optimal model parameters (p, d, q) are identified, often by analyzing Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots. Finally, the model is fitted to the historical data, and the

forecast() method is used to generate predictions for future time steps.33

### **3.2. Deep Learning for Time-Series: LSTM Networks**

Long Short-Term Memory (LSTM) networks are a specialized type of Recurrent Neural Network (RNN) that are exceptionally well-suited for learning patterns in sequential data, such as financial price time-series.34 Unlike traditional neural networks, LSTMs have internal memory cells that allow them to remember information over long periods. This capability enables them to capture complex, non-linear relationships and long-term dependencies in the data that simpler linear models like ARIMA might miss.36

Building and training an LSTM model for price prediction involves using deep learning frameworks like TensorFlow (often with the high-level Keras API) or PyTorch.15 The typical workflow is as follows:

1. **Data Preparation:** The price data must be preprocessed. This includes scaling the data (e.g., to a range between 0 and 1) to help the model converge, and transforming the data into sequences of a fixed window size (e.g., using the last 60 periods of data to predict the next period).34
2. **Model Architecture:** The LSTM network is constructed by stacking layers. A typical architecture might include one or more LSTM layers, followed by Dropout layers to prevent overfitting, and a final Dense layer to output the price prediction.34
3. **Training:** The model is compiled with an optimizer (e.g., 'adam') and a loss function (e.g., 'mean\_squared\_error'). It is then trained on the historical data sequences for a specified number of epochs.38
4. **Prediction:** Once trained, the model can take the most recent sequence of data as input to predict the price for the next time step.

### **3.3. Critical Considerations for ML Models**

The successful application of machine learning in financial forecasting is fraught with challenges that must be addressed to build a reliable system.

* **Feature Engineering:** The performance of any ML model is heavily dependent on the quality and relevance of its input features. Relying solely on historical closing prices is often insufficient. A more robust approach involves **feature engineering**, where additional data streams are created and fed into the model. These can include trading volume, volatility metrics like the Average True Range (ATR), and even the outputs of traditional technical indicators such as the RSI and MACD values.37 By providing the model with a richer, multi-dimensional view of the market, its predictive power can be significantly enhanced. The bot's architecture must therefore include a dedicated module for feature generation.
* **Overfitting and Validation:** ML models, particularly complex ones like LSTMs, are susceptible to **overfitting**. This occurs when the model learns the noise and random fluctuations in the training data so well that it fails to generalize to new, unseen data, rendering it useless for actual forecasting. To combat this, it is imperative to split the historical data into three distinct sets: a **training set** (to train the model), a **validation set** (to tune model hyperparameters), and a **testing set** (to evaluate the final model's performance on completely unseen data). Techniques like Dropout, which randomly deactivates neurons during training, are also crucial for mitigating overfitting.34
* **Concept Drift:** Financial markets are non-stationary systems; their underlying statistical properties and behavioral dynamics change over time. This phenomenon, known as **concept drift**, means that a model trained on data from a 2021 bull market may perform poorly in a 2023 bear market. A static, "train-once" model is destined to fail. A professional-grade system must account for concept drift by implementing a strategy for periodic retraining on more recent data and continuous performance monitoring to detect when the model's predictive power is degrading.

While technical indicators are reactive, describing what has already happened, ML models offer the potential for prediction. This makes them an incredibly powerful component of an analysis bot, but also a more dangerous one if their output is misinterpreted. The prediction from an ML model should never be treated as a certainty but rather as a **probabilistic forecast**. Instead of a binary "BUY" signal, a more useful and honest notification would be, "LSTM model predicts a 5% price increase in the next 24 hours with 75% confidence." This reframing acknowledges the inherent uncertainty of financial markets and provides the user with a more nuanced piece of information for their decision-making process.

## **Section 4: The Alpha in the Noise - Integrating Sentiment and On-Chain Analysis**

To elevate the bot from a purely price-based analysis tool to a comprehensive market intelligence system, it is essential to incorporate "alternative data." This data provides context that is completely invisible to models that only look at price and volume. For cryptocurrencies, the two most potent forms of alternative data are market sentiment derived from public discourse and on-chain data derived directly from the blockchain ledger. These data sources can often act as leading indicators, providing clues about future price movements before they are reflected in the charts.

### **4.1. Gauging Market Psychology: News and Social Media Sentiment**

Market sentiment—the collective attitude of investors and traders towards a particular asset—is a powerful, often self-fulfilling, driver of price action. By programmatically analyzing the tone and content of news articles, blog posts, and social media conversations related to specific cryptocurrencies, the bot can quantify whether the prevailing narrative is bullish (positive), bearish (negative), or neutral.40

* **Data Sources & APIs:** A variety of APIs can provide the raw text data required for this analysis. Some, like **Finnhub** 42 and  
  **Crypto News API** 43, are specifically designed for financial news and may even offer pre-computed sentiment scores. General news APIs or web scraping tools like  
  **Apify** 44 can also be used to gather relevant articles. For social media, the primary source is the API for X (formerly Twitter).
* **Implementation:** The core of sentiment analysis is Natural Language Processing (NLP). The process involves fetching the text data (e.g., news headlines or tweets), cleaning it to remove irrelevant characters and formatting, and then applying an NLP library to score its sentiment. For financial and social media text, rule-based models often perform very well. Libraries like **VADER** (Valence Aware Dictionary and sEntiment Reasoner) are specifically tuned for social media language, accounting for slang, punctuation, and capitalization.40  
  **TextBlob** is another popular and easy-to-use library for this purpose.41 The individual scores of many articles or posts are then aggregated over a recent period (e.g., the last 24 hours) to produce a single, time-series sentiment score for each asset.

### **4.2. Decoding the Blockchain: On-Chain Analysis**

On-chain analysis is a field of study unique to cryptocurrencies. It involves examining the public, immutable data on the blockchain itself to understand investor behavior, capital flows, and the overall health of the network.45 This provides a ground-truth view of economic activity, distinct from the often speculative price action seen on exchanges.

* **Key Metrics for Trading:** While hundreds of on-chain metrics exist, a few are particularly potent for generating trading signals:
  + **Exchange Flows (Inflows/Outflows):** This metric tracks the amount of a cryptocurrency being moved to and from exchange wallets. A sustained increase in **inflows** (coins moving onto exchanges) often precedes selling pressure, as investors prepare to liquidate their holdings (bearish). Conversely, a high volume of **outflows** (coins moving off exchanges into private wallets) suggests investors are accumulating for long-term holding, reducing the available supply and acting as a bullish signal.47
  + **Market Value to Realized Value (MVRV) Ratio:** This ratio compares the Market Value (current price) to the Realized Value (the average price at which every coin in circulation last moved). A high MVRV ratio indicates that the market is, on average, holding large unrealized profits, which increases the likelihood of profit-taking and can signal a market top. A low MVRV ratio suggests the market is undervalued relative to its cost basis, often identifying market bottoms.45
  + **Net Unrealized Profit/Loss (NUPL):** This metric measures the overall profitability of the market by showing the difference between unrealized profits and unrealized losses. NUPL charts are often color-coded to represent market sentiment stages, from "Capitulation" (extreme loss, potential bottom) to "Euphoria" (extreme profit, potential top).47
  + **Active Addresses:** This tracks the number of unique wallet addresses participating in transactions on a given day. A rising number of active addresses during a price uptrend is a strong confirmation of network growth and user adoption, validating the trend's health. A price increase with declining active addresses can be a bearish divergence, suggesting the rally is not supported by fundamental network activity.47
* **Data Sources & APIs:** Accessing and processing raw blockchain data is a complex data engineering task. Fortunately, specialized data providers offer clean, aggregated on-chain metrics through APIs. **Glassnode** 50 and  
  **Santiment** 54 are the leading platforms in this space, both providing extensive on-chain data and offering official Python client libraries to facilitate easy integration.

The integration of these alternative data sources marks a fundamental shift in the bot's capabilities. While technical analysis is largely based on lagging indicators that confirm what has already occurred, sentiment and on-chain data can often serve as **leading indicators**. For example, a sudden, massive spike in exchange inflows (an on-chain event) can precede the large-scale selling that will eventually push the price down and trigger a bearish MA crossover. A bot that monitors these on-chain flows can anticipate the move rather than just reacting to it. This provides a crucial analytical edge.

Furthermore, these data sources provide the "why" behind the "what." A price chart simply shows *what* is happening—the price is falling. On-chain analysis can help explain *why* it is falling. Is it due to long-term holders finally taking profits after a long bull run (indicated by a high MVRV and an increase in "Coin Days Destroyed"), or is it a panic-driven sell-off by newer market participants (indicated by a low Spent Output Profit Ratio)? The strategic response to these two scenarios is vastly different. This contextual layer allows the bot to generate far more nuanced and intelligent signals.

## **Section 5: A Unified Signal - Synthesizing Strategies for Robust Notifications**

With multiple analytical engines running in parallel—technical analysis, machine learning, sentiment analysis, and on-chain analysis—the bot will inevitably face the challenge of conflicting signals. On any given day, the moving average crossover might be bullish, while the RSI is overbought, news sentiment is turning negative, and on-chain data shows whale accumulation. A naive bot that acts on any single signal without considering the broader context will be whipsawed by the market and perform poorly. The central task of the bot's core logic is to synthesize these disparate inputs into a single, high-conviction notification, effectively increasing the signal-to-noise ratio.

### **5.1. The Challenge of Conflicting Signals**

The various analytical methods employed by the bot view the market through different lenses. Technical indicators look at price and volume patterns. Machine learning models look for complex, non-linear relationships in historical data. Sentiment analysis gauges human emotion and narrative. On-chain analysis measures the fundamental economic activity of the blockchain network. Because these sources are largely uncorrelated, they will frequently disagree. This disagreement is not a flaw in the system; it is a feature. It reflects the complex, multi-faceted nature of financial markets. The goal is not to eliminate disagreement but to build a system that can weigh the evidence and only act when there is a clear consensus.

### **5.2. A Multi-Factor Model for Signal Generation**

A practical and highly effective method for synthesizing signals is to implement a multi-factor scoring model. In this framework, each analytical component contributes to an overall "bullishness" or "bearishness" score for a given asset. A final "BUY" or "SELL" notification is only triggered when this aggregate score surpasses a predefined threshold, indicating that the weight of the evidence from multiple, diverse sources points in the same direction.

An example implementation of such a scoring system could be as follows:

1. **Technical Analysis Score (Range: -3 to +3):**
   * MA Crossover: Assign +1 if a Golden Cross occurred within the last N periods. Assign -1 for a Death Cross.
   * RSI: Assign +1 if the RSI has crossed up from the oversold region (<30) recently. Assign -1 if it has crossed down from the overbought region (>70).
   * MACD: Assign +1 for a recent bullish crossover (MACD line > Signal line). Assign -1 for a bearish crossover.
   * **Sum these values to get the TA Score.**
2. **Sentiment Score (Range: -2 to +2):**
   * Calculate the net sentiment from news and social media over the past 24 hours (e.g., % positive articles - % negative articles).
   * Scale this value to a normalized range, for example, from -2 (overwhelmingly negative) to +2 (overwhelmingly positive).
3. **On-Chain Score (Range: 0 to +3 for a buy signal):**
   * Exchange Flows: Assign +1 if there have been significant net outflows from exchanges over the past week.
   * MVRV Ratio: Assign +1 if the MVRV Z-score is in a low "opportunity" zone (historically undervalued).
   * Active Addresses: Assign +1 if the 30-day moving average of active addresses is in a clear uptrend.
   * **Sum these values to get the On-Chain Score.**
4. **Final Signal Logic:**
   * A rule is then defined to trigger a notification only when a high level of confluence is achieved. For example:  
     IF (TA Score >= 2) AND (Sentiment Score >= 0) AND (On-Chain Score >= 2) THEN issue 'STRONG BUY' notification.
   * This rule ensures that a buy signal is only generated when there is strong bullish technical momentum, non-negative sentiment, and confirming on-chain fundamentals.

This scoring model's primary purpose is to filter out low-conviction signals. By requiring agreement from multiple, uncorrelated analytical dimensions, the system effectively separates a true, robust market signal from the random noise generated by any single indicator. This approach is analogous to how a physician makes a diagnosis not from a single symptom but from a combination of lab tests, physical examination, and patient history. The result is a dramatic increase in the reliability and robustness of the notifications sent to the user.

Furthermore, the weights assigned to each component in the scoring system need not be static. A more advanced implementation could feature dynamic weighting based on the prevailing market regime. For instance, in a strongly trending bull market, trend-following technical indicators like moving averages might be assigned a higher weight. In a choppy, sideways market, mean-reversion oscillators like the RSI and on-chain metrics indicating accumulation by long-term holders might be more relevant. This adaptability allows the bot to adjust its analytical focus to match the current market character.

### **5.3. Strategy and Signal Synthesis Matrix**

The following table provides a blueprint for the bot's analytical engine, summarizing the various strategies and their characteristics within the multi-factor model.

| Strategy Family | Specific Indicator/Model | Signal Type | Data Source | Example "Bullish" Signal | Proposed Score Weight |
| --- | --- | --- | --- | --- | --- |
| **Technical Analysis** | MA Crossover (50/200) | Lagging | Price (OHLCV) | 50-period MA crosses above 200-period MA. | 1 |
|  | RSI (14) | Coincident | Price (Close) | RSI crosses above 30 from oversold territory. | 1 |
|  | MACD (12,26,9) | Lagging | Price (Close) | MACD line crosses above the signal line. | 1 |
| **Machine Learning** | LSTM Prediction | Leading | Price, Volume, etc. | Model forecasts a price increase > X% in N periods. | 2 |
| **Sentiment Analysis** | News/Social Sentiment | Leading | Text Data | Net sentiment score > 0.5 over 24 hours. | 1 |
| **On-Chain Analysis** | Exchange Net Flow | Leading | On-Chain API | Sustained net outflows from exchanges. | 2 |
|  | MVRV Z-Score | Coincident | On-Chain API | MVRV Z-Score enters the "undervalued" zone (<0). | 1 |
|  | Active Addresses | Coincident | On-Chain API | 30-day MA of active addresses is trending up. | 1 |

## **Section 6: Capital Preservation and Volatility Management**

Generating entry and exit signals is only half the battle. In the hyper-volatile cryptocurrency markets, long-term success is dictated not by picking winners, but by effectively managing risk and preserving capital. An analysis bot that only provides entry signals without any risk context is incomplete and potentially dangerous. This section details how to integrate a robust risk management framework into the bot, transforming it from a simple signal generator into a sophisticated, risk-aware advisory tool.

### **6.1. Dynamic Risk Controls: Stop-Loss and Take-Profit**

A **stop-loss** is a pre-determined price level at which a losing trade is exited to prevent further losses. A **take-profit** is the price at which a profitable trade is closed to realize the gains. The implementation of these risk controls is paramount for disciplined trading.

In cryptocurrency markets, using static, percentage-based stop-losses (e.g., "sell if the price drops 5% from my entry") is a common but flawed approach. Such a rule fails to account for the market's changing volatility. A 5% drop might be catastrophic in a calm market but could be normal, insignificant noise during a volatile period. A stop-loss that is too tight will result in being prematurely "stopped out" of otherwise good trades, while a stop that is too loose exposes the trader to excessive risk.

A far superior, modern approach is to use a volatility-adjusted stop-loss based on the **Average True Range (ATR)**. The ATR is a technical indicator that measures market volatility over a given period.58 Instead of a fixed percentage, the stop-loss is placed at a multiple of the ATR below the entry price (for a long position). A common setting is

Entry Price - (2 \* ATR) or Entry Price - (3 \* ATR).58 This creates a dynamic stop-loss that automatically widens during periods of high volatility (giving the trade more room to breathe) and tightens during periods of low volatility (protecting capital more closely). This method adapts to the market's character, preventing premature exits caused by normal price fluctuations.

### **6.2. Volatility-Adjusted Position Sizing**

This is arguably the single most important risk management technique for any trader in a volatile market. Instead of risking a fixed amount of currency (e.g., trading $1,000 worth of Bitcoin per signal), the size of each position is calculated based on a fixed percentage of the total portfolio that the trader is willing to risk on that single trade.60 This calculation directly incorporates the stop-loss distance.

The formula is:

$$\text{Position Size (in currency)} = \frac{(\text{Total Account Capital} \times \text{Risk Percentage})}{(\text{Percentage Distance to Stop-Loss})} $$Or, more practically for execution:$$ \text{Position Size (in units of crypto)} = \frac{(\text{Total Account Capital} \times \text{Risk Percentage})}{(\text{Entry Price} - \text{Stop-Loss Price})}$$

The bot's logic should integrate this calculation seamlessly. When a "BUY" signal is generated, the bot must also perform the following steps:

1. Calculate the current ATR value for the asset's chart.
2. Determine the suggested stop-loss price based on a multiple of that ATR (e.g., Entry Price - 2.5 \* ATR).
3. Using this stop-loss, calculate the appropriate position size for a pre-defined risk parameter (e.g., risk 1% of the total portfolio).

The final notification sent to the user's console should be a complete trade plan, not just an entry signal. For example:

STRONG BUY SIGNAL: BTC-USD | Suggested Entry: $65,000 | Suggested Stop-Loss: $62,500 (based on 2.5 \* ATR) | Position Size for 1% risk on $100,000 portfolio: $4,000 (0.0615 BTC)

This system embodies the "holy trinity" of risk management: it defines (1) where to get in, (2) where to get out if wrong (stop-loss), and (3) how much to trade (position size). These three elements are inextricably linked; the stop-loss distance directly determines the position size. The core principle is to make the amount of capital risked per trade **constant** (e.g., always 1% of the portfolio), even as the market's volatility and the required position size fluctuate. This transforms trading from an emotional gamble into a disciplined process with a clearly defined and consistent risk profile, which is essential for long-term survival and profitability.

### **6.3. Portfolio-Level Risk: Diversification**

If the bot is configured to monitor and generate signals for multiple cryptocurrencies, an additional layer of risk management is required at the portfolio level. Many cryptocurrencies, especially those with large market capitalizations like Bitcoin and Ethereum, are highly correlated, meaning they tend to move in the same direction. Issuing simultaneous buy signals for two highly correlated assets is not true diversification; it is a concentrated bet on the direction of the entire market.

To manage this, the bot can implement a correlation check. It should periodically calculate and maintain a correlation matrix for the assets it monitors. When a buy signal is generated for an asset (e.g., Ethereum), the bot's logic can check the correlation of that asset with any other assets that are also close to generating a buy signal. If the correlation is above a certain threshold (e.g., >0.9), the bot could suppress the signal for the second asset or issue a notification with a warning about the concentrated risk exposure. This prevents the user from inadvertently taking on excessive, undiversified market risk.

## **Section 7: System Architecture and Operational Best Practices**

The final component in architecting a professional-grade analysis bot involves the practical considerations of deployment, operational logic, and maintenance. A brilliant strategy is worthless if the system implementing it is unreliable or poorly structured. This section covers the best practices for ensuring the bot runs effectively, efficiently, and robustly.

### **7.1. The 24/7 Imperative: Deployment on a Virtual Private Server (VPS)**

The single most defining characteristic of the cryptocurrency market is that it never closes. It operates 24 hours a day, 7 days a week, 365 days a year.62 Significant price movements and trading opportunities can occur at any time, including weekends and late-night hours. Consequently, any serious analysis bot must be operational continuously to capture these events.

Attempting to run such a bot on a personal desktop or laptop computer is not a viable long-term solution. Personal machines are subject to a host of reliability issues, including internet service disruptions, power outages, automatic system updates and reboots, and hardware failures.64 Any of these events would cause the bot to go offline, potentially for hours, during which critical market signals would be missed.

The professional standard and architectural requirement for this type of application is deployment on a cloud-based **Virtual Private Server (VPS)**. A VPS provides a dedicated, managed server environment with guaranteed uptime (often 99.9% or higher), redundant power and network connections, and professional maintenance.64 A lightweight, Linux-based VPS offers a stable, low-resource, and secure environment perfectly suited for running a Python script 24/7. This choice is not merely a matter of convenience; it is a core design requirement. The system's reliability and uptime are as crucial to its success as the quality of the analytical strategies it employs.

### **7.2. Multi-Timeframe Analysis (MTA)**

A powerful technique used by professional traders to increase the probability of their trades is **Multi-Timeframe Analysis (MTA)**. The core concept is to align trades with the dominant, longer-term market trend.65 By making strategic decisions on a longer timeframe and then looking for tactical entry points on a shorter timeframe, a trader can avoid "fighting the trend" and ensure they are trading with the market's primary momentum.

This hierarchical approach should be built into the bot's core logic as a master filter that sits above all other signal generation strategies. The implementation framework is as follows:

1. **Trend/Context Timeframe (e.g., Daily or 4-Hour):** The bot first analyzes a longer-term chart to establish the overall market "regime." A simple yet effective method is to use a long-period moving average, such as the 200-period SMA. If the current price is above the 200 SMA, the long-term trend is classified as **"Bullish."** If the price is below it, the trend is **"Bearish."**
2. **Signal/Execution Timeframe (e.g., 1-Hour or 15-Minute):** After determining the long-term trend, the bot then switches to analyzing the shorter timeframe chart. However, it now operates in a filtered mode:
   * If the long-term trend is **"Bullish,"** the bot will *only* search for and act upon "BUY" signals generated by its multi-factor model. All "SELL" signals on the shorter timeframe are ignored as they are likely to be minor pullbacks within a larger uptrend.
   * If the long-term trend is **"Bearish,"** the bot will *only* search for and act upon "SELL" signals. All "BUY" signals are ignored as they are likely to be temporary bounces in a larger downtrend.65

This MTA structure acts as the highest-level filter in the entire system. It prevents the bot from taking low-probability counter-trend trades and significantly improves the quality of the signals it generates. The choice of which timeframes to pair depends on the desired trading style, from short-term day trading to longer-term position trading.67

### **7.3. Multi-Timeframe Analysis Pairings Guide**

The following table provides a practical guide for selecting appropriate timeframe pairs based on different trading profiles.

| Trading Profile | Trend Timeframe | Signal Timeframe | Typical Holding Period | Example |
| --- | --- | --- | --- | --- |
| **Day Trader** | 1-Hour (H1) | 5-Minute (M5) or 15-Minute (M15) | Minutes to Hours | Check H1 trend; enter on M5 MA crossover. |
| **Short-Term Swing Trader** | Daily (D1) | 4-Hour (H4) | Days to a Week | Check D1 trend; enter on H4 RSI divergence. |
| **Long-Term Swing Trader** | Weekly (W1) | Daily (D1) | Weeks to a Month | Check W1 trend; enter on D1 MACD crossover. |
| **Position Trader** | Monthly (MN1) | Weekly (W1) | Months to Years | Check MN1 trend; enter on W1 support levels. |

### **7.4. Logging and Monitoring**

For any automated system, a robust logging mechanism is not an optional feature but a core requirement for debugging, performance analysis, and maintaining an audit trail. The bot should be configured to log every significant action it takes.

Using Python's built-in logging module, the bot should write timestamped messages to a log file. Key events to log include:

* Application start and stop times.
* Scheduled analysis runs (e.g., "Starting hourly analysis for BTC-USD...").
* Successful and failed API calls for data fetching.
* The calculated values of key indicators and scores from the multi-factor model.
* Signals that were generated but did not meet the final confluence threshold (e.g., "TA score of 2 met, but On-Chain score of 0 did not. No signal issued.").
* The full details of any final notification sent to the console, including the asset, signal type, price, and all risk management parameters.

This detailed log file becomes an invaluable resource for understanding the bot's behavior over time, diagnosing issues, and refining the strategy based on a historical record of its "decisions."

## **Conclusion and Future Directions**

### **Summary of Recommended Architecture**

The architecture outlined in this report provides a comprehensive blueprint for developing a modern, sophisticated cryptocurrency analysis bot tailored for the Coinbase platform. The recommended system is built upon a foundation of professional-grade software engineering and quantitative finance principles. It advocates for a 24/7 deployment on a stable Virtual Private Server (VPS), utilizing a modular data access layer that can leverage either the official Coinbase Advanced SDK for direct integration or the CCXT library for future flexibility.

The analytical core of the bot is a multi-factor synthesis model that moves beyond simplistic, single-indicator strategies. It achieves signal robustness through the principle of confluence, requiring agreement across four distinct analytical dimensions:

1. **Technical Analysis:** Using classic indicators like Moving Averages, RSI, and MACD to gauge price momentum.
2. **Machine Learning:** Employing predictive models like LSTMs to forecast future price movements.
3. **Sentiment Analysis:** Quantifying market psychology by analyzing news and social media data.
4. **On-Chain Analysis:** Leveraging blockchain-native data to understand fundamental network health and investor behavior.

This multi-faceted analytical engine is governed by a hierarchical **Multi-Timeframe Analysis** framework, which ensures that all tactical signals are aligned with the dominant long-term market trend. Crucially, every signal is coupled with a robust, volatility-aware risk management plan, providing the user not just with an entry point, but with a complete trade plan that includes an ATR-based stop-loss and a dynamically calculated position size to maintain consistent risk on every trade.

### **The Crucial Next Step: Backtesting**

Before the signals generated by this bot can be trusted with real capital, the entire strategic framework—from data ingestion and signal synthesis to risk management—must be rigorously validated against historical data. This process, known as **backtesting**, is a non-negotiable step in quantitative strategy development.

Backtesting involves simulating the execution of the bot's strategy on past market data to assess its historical performance. This provides statistical evidence of whether the strategy has a positive expectancy and helps to uncover its characteristics, such as profitability, maximum drawdown (the largest peak-to-trough decline), and win rate. Python offers powerful open-source backtesting libraries like vectorbt and Backtrader that are specifically designed for this purpose.15 These frameworks can rapidly test thousands of parameter combinations for technical indicators and risk models, allowing for the optimization of the strategy for each specific cryptocurrency and timeframe. This validation phase is the only way to move from a theoretical strategy to one with statistical confidence behind it.

### **Path to Live Notifications**

Following a successful and satisfactory backtesting phase, the final step before full deployment is a period of live monitoring, often called "paper trading." In this stage, the bot is deployed on the VPS and runs in real-time, fetching live market data and generating its notifications to the console as if it were live. However, the user does not act on these signals with real capital. Instead, they observe the bot's performance in the current market environment, validating that its real-time behavior matches the backtested expectations. This final validation step builds confidence in the system and allows for any last-minute adjustments before the user begins to incorporate its analytical output into their actual trading decisions. By following this structured path of design, backtesting, and validation, a developer can construct a powerful and reliable tool to navigate the complexities of the cryptocurrency markets.

#### **Works cited**

1. coinbase/coinbase-advanced-py: The Advanced API ... - GitHub, accessed August 19, 2025, <https://github.com/coinbase/coinbase-advanced-py>
2. Coinbase Advanced API Python SDK documentation - GitHub Pages, accessed August 19, 2025, <https://coinbase.github.io/coinbase-advanced-py/>
3. Welcome to Advanced Trade API - Coinbase Developer Documentation, accessed August 19, 2025, <https://docs.cdp.coinbase.com/advanced-trade/docs/welcome>
4. [NEW EXCHANGE]: Coinbase advanced trade · Issue #16281 · ccxt/ccxt - GitHub, accessed August 19, 2025, <https://github.com/ccxt/ccxt/issues/16281>
5. coinbase-advancedtrade-python 0.1.3 - PyPI, accessed August 19, 2025, <https://pypi.org/project/coinbase-advancedtrade-python/0.1.3/>
6. coinbase - PyPI, accessed August 19, 2025, <https://pypi.org/project/coinbase/>
7. ccxt - documentation, accessed August 19, 2025, <https://docs.ccxt.com/>
8. ccxt - PyPI, accessed August 19, 2025, <https://pypi.org/project/ccxt/>
9. Manual · ccxt/ccxt Wiki - GitHub, accessed August 19, 2025, <https://github.com/ccxt/ccxt/wiki/manual>
10. Working with CCXT Python Library - DEV Community, accessed August 19, 2025, <https://dev.to/macanudo527/working-with-ccxt-python-library-2koe>
11. How to get historical price data using ccxt (over 500 and 1000 rows) - Manuel Levi, accessed August 19, 2025, <https://manuellevi.com/how-to-get-more-data-price-data-using-ccxt/>
12. Python Scripts for CCXT Crypto Candlestick (OHLCV) Charting Data - Medium, accessed August 19, 2025, <https://medium.com/coinmonks/python-scripts-for-ccxt-crypto-candlestick-ohlcv-charting-data-83926fa16a13>
13. Getting started with CCXT Crypto Exchange Library and Python | Adnan's Random bytes, accessed August 19, 2025, <https://blog.adnansiddiqi.me/getting-started-with-ccxt-crypto-exchange-library-and-python/>
14. CryptoCurrency Market Data with CCXT: A Technical Guide | by PI | Neural Engineer, accessed August 19, 2025, <https://medium.com/neural-engineer/cryptocurrency-market-data-with-ccxt-a-technical-guide-08a9943f9639>
15. Best Python Libraries for Algorithmic Trading and Financial Analysis - QuantInsti Blog, accessed August 19, 2025, <https://blog.quantinsti.com/python-trading-library/>
16. How to Extract Crypto Price Data with Python | Crawlbase, accessed August 19, 2025, <https://crawlbase.com/blog/how-to-extract-crypto-data-with-python/>
17. Simple Moving Average Crossover Strategy using Python | by PulsePointFX | Medium, accessed August 19, 2025, <https://medium.com/@pulsepointfx/simple-moving-average-crossover-strategy-using-python-45691c9b12c1>
18. Moving Average Crossover Strategies: Types, Calculations, Pros & Cons for Trading, accessed August 19, 2025, <https://blog.quantinsti.com/moving-average-trading-strategies/>
19. Backtesting a Moving Average Crossover in Python with pandas - QuantStart, accessed August 19, 2025, <https://www.quantstart.com/articles/Backtesting-a-Moving-Average-Crossover-in-Python-with-pandas/>
20. boku-wa-abhi/moving\_average\_crossover: Trading Strategy, Algorithmic Trading, S&P500, Python - GitHub, accessed August 19, 2025, <https://github.com/abbi163/moving_average_crossover>
21. How to code an EMA crossover in Python - Medium, accessed August 19, 2025, <https://medium.com/@financial_python/how-to-code-an-ema-crossover-in-python-96ecd22ae252>
22. Relative Strength Index (RSI) Indicator Explained With Formula - Investopedia, accessed August 19, 2025, <https://www.investopedia.com/terms/r/rsi.asp>
23. RSI Trading Strategy | Relative Strength Index - AvaTrade, accessed August 19, 2025, <https://www.avatrade.com/education/technical-analysis-indicators-strategies/rsi-trading-strategies>
24. How to Use the Relative Strength Index (RSI) - Charles Schwab, accessed August 19, 2025, <https://www.schwab.com/learn/story/how-to-use-relative-strength-index-rsi>
25. What is RSI? - Relative Strength Index - Fidelity Investments, accessed August 19, 2025, <https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/RSI>
26. RSI Trading Strategy: How To Use RSI Indicator - Capital.com, accessed August 19, 2025, <https://capital.com/en-int/learn/technical-analysis/relative-strength-index>
27. Python Trading Strategy: Synergizing Stochastic Oscillator and MACD Indicator | EODHD APIs Academy, accessed August 19, 2025, <https://eodhd.com/financial-academy/backtesting-strategies-examples/using-python-to-create-an-innovative-trading-strategy-and-achieve-better-results>
28. Python and MACD Trading Strategy: Backtest, Rules, Code, Setup, Performance, accessed August 19, 2025, <https://www.quantifiedstrategies.com/python-and-macd-trading-strategy/>
29. Building a MACD Indicator in Python - Medium, accessed August 19, 2025, <https://medium.com/@financial_python/building-a-macd-indicator-in-python-190b2a4c1777>
30. nmayur1997/MACD-Strategy - GitHub, accessed August 19, 2025, <https://github.com/nmayur1997/MACD-Strategy>
31. GZotin/RSI\_MACD\_strategy: Python script for trading analysis using RSI and MACD indicators. - GitHub, accessed August 19, 2025, <https://github.com/GZotin/RSI_MACD_strategy>
32. Bitcoin Arima Model | Coinmonks - Medium, accessed August 19, 2025, <https://medium.com/coinmonks/bitcoin-arima-model-f22456bd1fa9>
33. How to Build ARIMA Model in Python for time series forecasting? - ProjectPro, accessed August 19, 2025, <https://www.projectpro.io/article/how-to-build-arima-model-in-python/544>
34. CRYPTO CURRENCY PRICE PREDICTION TOOL USING PYTHON - IRJMETS, accessed August 19, 2025, <https://www.irjmets.com/uploadedfiles/paper/issue_8_august_2022/29069/final/fin_irjmets1663071054.pdf>
35. Bitcoin Price Prediction For 10 Days (LSTM vs GRU) - Kaggle, accessed August 19, 2025, <https://www.kaggle.com/code/mekhdigakhramanian/bitcoin-price-prediction-for-10-days-lstm-vs-gru>
36. Forecasting Cryptocurrency Prices Using LSTM, GRU, and Bi-Directional LSTM: A Deep Learning Approach - MDPI, accessed August 19, 2025, <https://www.mdpi.com/2504-3110/7/2/203>
37. How to predict stock or crypto prices. In 2024. With Python. And PyTorch. | by Nicky Reinert, accessed August 19, 2025, <https://nickyreinert.medium.com/how-to-predict-stock-or-crypto-prices-in-python-with-pytorch-in-2024-do-it-now-21873964be4b>
38. alexanderstahl93/Bitcoin-Price-Prediction-with-LSTM - GitHub, accessed August 19, 2025, <https://github.com/alexanderstahl93/Bitcoin-Price-Prediction-with-LSTM>
39. Predicting Cryptocurrency Prices with Machine Learning Algorithms: A Comparative Analysis - DiVA portal, accessed August 19, 2025, <https://www.diva-portal.org/smash/get/diva2:1778251/FULLTEXT03>
40. rishikonapure/Cryptocurrency-Sentiment-Analysis - GitHub, accessed August 19, 2025, <https://github.com/rishikonapure/Cryptocurrency-Sentiment-Analysis>
41. Bitcoin Sentiment Analysis using Python and X (Formerly Twitter) - DEV Community, accessed August 19, 2025, <https://dev.to/spiff/bitcoin-sentiment-analysis-using-python-and-x-formerly-twitter-359h>
42. News Sentiment - Finnhub, accessed August 19, 2025, <https://finnhub.io/docs/api/news-sentiment>
43. Crypto News API - A JSON API for crypto news, accessed August 19, 2025, <https://cryptonews-api.com/>
44. Crypto News Pro Scraper API in Python - Apify, accessed August 19, 2025, <https://apify.com/buseta/crypto-news/api/python>
45. On-Chain Analysis Explained: How to Read Blockchain Like a Pro - CCN.com, accessed August 19, 2025, <https://www.ccn.com/education/understanding-on-chain-analysis-a-comprehensive-guide/>
46. What is onchain analysis and how to use it as a crypto trader? - Coinbase, accessed August 19, 2025, <https://www.coinbase.com/learn/advanced-trading/what-is-onchain-analysis-and-how-to-use-it-as-a-crypto-trader>
47. How to Read and Interpret On-Chain Metrics Like a Pro - Altrady, accessed August 19, 2025, <https://www.altrady.com/crypto-trading/onchain-blockchain-analytics-for-traders/how-to-read-interpret-onchain-metrics>
48. A Beginner's Guide to On-chain Metrics for Crypto Investing - Arbismart - Trusted Transparent Arbitrage Trading - EU Regulated., accessed August 19, 2025, <https://arbismart.com/blog/beginners-guide-to-on-chain-metrics-for-crypto-investing/>
49. How Are Crypto Investors and Traders Using On-Chain Analysis? - Cointree, accessed August 19, 2025, <https://www.cointree.com/learn/on-chain-analysis/>
50. glassnode-api-python-client/glassnode.py at master - GitHub, accessed August 19, 2025, <https://github.com/glassnode/glassnode-api-python-client/blob/master/glassnode.py>
51. API Key - Glassnode Docs, accessed August 19, 2025, <https://docs.glassnode.com/basic-api/api-key>
52. glassnode-api-python-client - GitHub, accessed August 19, 2025, <https://github.com/glassnode/glassnode-api-python-client>
53. On-Chain Metric Investing Strategy with Glassnode API - Alpaca, accessed August 19, 2025, <https://alpaca.markets/learn/on-chain-metric-investing-strategy-with-glassnodeapi>
54. Getting started using Santiment's API | Santiment Knowledge Base, accessed August 19, 2025, <https://santiment.github.io/articles/intercom-articles/getting-started/sanbase-api/getting-started-using-santiment-s-api/>
55. sanpy · PyPI, accessed August 19, 2025, <https://pypi.org/project/sanpy/0.4.1/>
56. santiment/sanpy: Santiment API Python Client - GitHub, accessed August 19, 2025, <https://github.com/santiment/sanpy>
57. Using Python to Get Crypto Market Data | by Yura Zatsepin - DataDrivenInvestor, accessed August 19, 2025, <https://medium.datadriveninvestor.com/using-python-to-get-crypto-market-data-ba173d93896b>
58. 4 practical methods to set your stop-loss when algo-trading Bitcoin - Jesse Blog, accessed August 19, 2025, <https://jesse.trade/blog/tutorials/4-practical-methods-to-set-your-stop-loss-when-algo-trading-bitcoin>
59. Grid Trading with Python: A Simple and Profitable Algorithmic Strategy | by Ziad Francis, PhD | Medium, accessed August 19, 2025, <https://medium.com/@ziad.francis/grid-trading-with-python-a-simple-and-profitable-algorithmic-strategy-820410698516>
60. Volatility-Based Position Sizing with Python: How to Adjust Your Trades | by SR | Medium, accessed August 19, 2025, <https://medium.com/@deepml1818/volatility-based-position-sizing-with-python-how-to-adjust-your-trades-1f88efc8b228>
61. Position Sizing in a Turtle Trading system - QuantifiedStrategies.com, accessed August 19, 2025, <https://www.quantifiedstrategies.com/position-sizing-in-a-turtle-trading-system/>
62. Crypto Trading Bots: Roles, Types & Use Cases - AvaTrade, accessed August 19, 2025, <https://www.avatrade.com/blog/trading-tools-technologies/understanding-crypto-trading-bots>
63. Understanding Crypto Auto Trading Bots: Benefits and Risks - Rapid Innovation, accessed August 19, 2025, <https://www.rapidinnovation.io/post/crypto-auto-trading-bot-whats-the-buzz-all-about>
64. VPS for Crypto Trading Bots: Unlock 24/7 Uptime, Speed & Reliability in 2025 - Bluehost, accessed August 19, 2025, <https://www.bluehost.com/blog/vps-for-crypto-trading-bots/>
65. How to Use Multi-Time Frame Analysis in Crypto Markets - ZenLedger, accessed August 19, 2025, <https://zenledger.io/blog/how-to-use-multi-timeframe-analysis-in-crypto-markets/>
66. Crypto Time Intervals - altFINS, accessed August 19, 2025, <https://altfins.com/knowledge-base/time-frames/>
67. Time Interval Analysis in Crypto: 1m, 5m, 1H, 1D, 1W Strategies - YouHodler, accessed August 19, 2025, <https://www.youhodler.com/education/time-interval-analysis-1m-5m-15m-1h-4h-1d-1w>
68. How to Pick the Best Time Frames for Crypto Trading - Blueberry Markets, accessed August 19, 2025, <https://blueberrymarkets.com/market-analysis/how-to-choose-optimal-time-frames-for-crypto-trading/>