Advanced Target Variable Engineering for Algorithmic Trading Models

Introduction

The efficacy of any supervised machine learning model is fundamentally constrained by the quality and relevance of its target variable. In the domain of quantitative finance, the definition of this target variable—the very outcome the model is trained to predict—is arguably the most critical decision in the entire modeling pipeline.[1] It dictates the nature of the problem being solved, influences the choice of features and algorithms, and ultimately determines whether a model's predictive power can be translated into tangible, risk-adjusted returns. A model trained on a poorly conceived target may exhibit high statistical accuracy yet consistently fail in a live trading environment, a phenomenon that underscores the chasm between abstract prediction and practical profitability.[3] The model's outputs are rendered meaningless if the target itself is not well-understood and precisely aligned with the financial objective.[1]

The current approach, which involves a CatBoost model predicting a fixed 10% price increase over a static 7-day period, is a standard implementation of what is known as the **Fixed-Time Horizon (FTH)** method.[6] This technique is prevalent in academic literature and serves as a common entry point for financial machine learning due to its simplicity. It assigns a label based on the price return over a predetermined number of future bars or days.[7] While straightforward, the FTH method is encumbered by fundamental flaws that severely limit its real-world applicability and profitability. Its primary weakness is a structural blindness to the

path of returns and the practical mechanics of risk management, such as the execution of stop-loss or profit-taking orders.

This report provides a comprehensive and exhaustive analysis of superior, industry-standard alternatives to the FTH method. It will systematically deconstruct the inherent fallacies of fixed-horizon labeling and introduce a more robust paradigm

for target variable engineering. The analysis will begin with the **Triple-Barrier Method (TBM)**, a revolutionary technique that reframes the prediction problem from one of simple forecasting to one of simulating a risk-managed trade. From this foundation, the report will explore more advanced classification targets, including multi-class labels and the sophisticated two-stage process of **Meta-Labeling**. Finally, it will venture into regression-based and alternative paradigms, such as predicting future volatility, directly optimizing financial metrics like the Sharpe Ratio, and the nascent application of Reinforcement Learning. The overarching objective is to provide a clear, logical progression from a simplistic forecasting model to a sophisticated system that learns to identify signals conducive to profitable trading under a realistic risk management framework.

The Triple-Barrier Method: A Paradigm Shift in Financial Labeling

The most significant and foundational improvement over the Fixed-Time Horizon method is the adoption of the Triple-Barrier Method (TBM). This technique, pioneered and popularized by Dr. Marcos Lopez de Prado, represents a paradigm shift in how financial machine learning practitioners define prediction targets.[9] Instead of asking an abstract forecasting question, the TBM frames the problem in a manner that is congruent with the real-world mechanics of executing a trade, incorporating profit-taking objectives, risk-management limits, and time-based capital allocation constraints directly into the label itself.[12]

Deconstructing the Fixed-Time Horizon Fallacy

The FTH method, despite its widespread use, is fundamentally misaligned with the realities of trading because it is path-independent. It evaluates the price only at a single point in the future (t+h) and disregards the sequence of prices that occurred during the holding period.[6] This structural flaw leads to systematic mislabeling of trades, teaching the machine learning model incorrect lessons about profitability and risk.

Two illustrative scenarios highlight the critical deficiencies of the FTH approach:

- Scenario A: The Misclassified Winner (False Negative)
  Consider a trading signal that prompts a long position. The stock price subsequently rallies 12% over the next three days, surpassing a hypothetical 10% profit-take target. A disciplined trader would have exited the position and realized a 10% gain. However, by the end of the fixed 7-day horizon, the stock has reverted and is now only 5% above the entry price. The FTH method would assign this event a label of 0 (failure), as the 10% threshold was not met at the 7-day mark. The model is thus incorrectly penalized for what was, in practice, a successful and profitable trade. It learns to avoid signals that lead to quick, profitable mean-reverting patterns, which is a significant strategic handicap.14
- Scenario B: The Unrecognized Loser (Ignored Risk)
  Now consider a signal where the stock price drops by 5% within the first two days. This decline would trigger the specified 4% stop-loss, forcing a real-world trader to exit with a managed loss. By the end of the 7-day horizon, the stock has recovered to show a 2% gain. The FTH method, looking only at the final price, would again label this a 0 (failure to reach 10%), but it fails to differentiate this outcome from a small, unrealized gain. Critically, it does not label the event as a loss. The model is never taught that the signal led to an outcome where the predefined risk limit was breached. It learns nothing about the path that led to the stop-out, effectively ignoring the most important risk management component of the strategy.

Beyond path-independency, the FTH method suffers from a **volatility mismatch**. It rigidly applies the same time horizon and percentage thresholds to all assets across all market conditions. A 10% target may be a reasonable objective for a volatile technology stock over one week, but it could be an extremely rare event for a stable utility stock, for which a 2% move might be significant.[15] Conversely, during periods of high market volatility, a 10% move might occur in a matter of hours, making a 7-day horizon unnecessarily long and exposing the position to reversal risk. The TBM addresses this by making both the profit/loss thresholds and the time horizon dynamic and responsive to the prevailing market environment.[12]

Ultimately, the FTH method is flawed because it is not grounded in the logic of a trading strategy. It asks, "Where will the price be?" The more financially meaningful and robust question is, "Would a trade initiated from this signal have been profitable, given my specific risk-management rules?".[4] The TBM is designed to answer the latter.

The Three Barriers: A Framework for Realistic Trade Outcomes

The Triple-Barrier Method provides a robust framework for labeling financial time series data by simulating the lifecycle of a trade with predefined exit conditions. It employs three distinct barriers around the trade entry point to determine the outcome.[12]

- **The Upper Barrier (Profit-Take):** This is a horizontal price level set above the entry price. It represents the profit-taking objective for the trade. If the asset's price path touches this barrier before any other, the event is labeled as a success (e.g., +1), and the position is considered closed with a profit.[6]
- **The Lower Barrier (Stop-Loss):** This is a horizontal price level set below the entry price, serving as the risk limit for the trade. If the price path touches this lower barrier first, the event is labeled as a failure (e.g., -1), representing a trade that was closed due to hitting the maximum acceptable loss.[6]
- **The Vertical Barrier (Time-Out):** This barrier is not a price level but a time limit on the trade's duration, defined by a maximum number of bars or a specific time horizon. If the price fails to touch either the upper or lower horizontal barrier within this specified period, the vertical barrier is "hit," and the position is closed. This mechanism ensures that capital is not indefinitely tied up in trades that are not performing.[16] The label for a vertical barrier hit is often 0, or it can be determined by the sign of the return at the moment of exit (e.g., +1 if the return is positive, -1 if negative), providing more granular information.[6]

The defining characteristic of the TBM is the **"First Touch" Principle**. The final label assigned to a trading event is determined exclusively by whichever of the three barriers is hit *first*.[14] This principle is what imbues the method with path-dependency, ensuring that the label accurately reflects the realized profit or loss of a structured trade as it would have occurred in a live setting. This is a profound departure from the FTH method, which is blind to such intraday or intra-period price movements.

Feature	Fixed-Time Horizon (FTH) Method	Triple-Barrier Method (TBM)
Core Question	"Where will the price be at time T?"	"Will my trade succeed given my P/L rules?"
Path Dependency	Ignored. Only the price at the end of the horizon matters.	Central to the method via the "first touch" rule.

Risk Management (Stop-Loss)	Applied post-prediction during backtesting; not part of the label itself.	Integrated directly into the -1 label, teaching the model about risk.
Profit Taking	Not explicitly modeled in the label; a fixed target is checked at a fixed time.	Integrated directly into the +1 label, capturing early exits.
Adaptability to Volatility	Static time horizon and fixed percentage thresholds are unresponsive to market conditions.	Dynamic barriers are set as a function of market volatility, adapting to the asset.
Likelihood of Mislabeling	High. Prone to false negatives and ignoring realized losses.	Low. Labels accurately reflect the P&L of a structured, realistic trade.

Dynamic Barrier Setting: Adapting to Market Volatility

A key sophistication of the TBM is its ability to set profit-take and stop-loss levels that are dynamic and context-aware, rather than fixed and arbitrary. The user's current model with its static 10% profit target and 4% stop-loss is suboptimal because it treats all market environments and all assets identically.[15] A 10% move is a vastly different event for a low-beta utility stock than for a high-beta technology stock.

The professional standard, as advocated by de Prado, is to set the horizontal barriers as a function of the asset's recent volatility.[10] This ensures that the risk and reward targets are scaled appropriately to the instrument's typical price fluctuations.

The implementation of dynamic barriers follows a clear, logical process:

1. **Calculate Daily Volatility:** First, compute a measure of the asset's recent volatility. This is typically done by calculating the standard deviation of daily returns over a rolling lookback period. An Exponentially Weighted Moving Average (EWMA) of the standard deviation is often preferred, as it places greater importance on more recent price action, making the volatility measure more responsive to changing market conditions.[10] A common lookback period is 20, 50, or 100 days.

2. **Set Barriers as Multiples of Volatility:** The upper and lower barriers are then defined as multiples of this calculated daily volatility. This establishes a risk-reward ratio that is consistent in terms of market behavior rather than fixed percentages. For instance, the barriers can be set according to the following formulas:
   Upper Barrier: $pu=p0\times(1+cup\times\sigma t)$
   Lower Barrier: $pl=p0\times(1-cdown\times\sigma t)$
   Where:
   - $p0$ is the entry price.
   - σt is the estimated daily volatility at the time of entry.
   - cup and cdown are user-defined constants that set the risk-reward ratio. For example, setting cup=2 and cdown=1 would establish a 2:1 profit target to stop-loss ratio, scaled by volatility.[14]

A Python function to compute this dynamic volatility, based on de Prado's methodology, can be structured as follows:

Python

```
import pandas as pd

def get_daily_volatility(close_prices: pd.Series, span: int = 100) -> pd.Series:
    """
    Calculates the daily volatility of a time series, typically used for setting
    dynamic profit-take and stop-loss barriers.

    Args:
        close_prices: A pandas Series of asset closing prices.
        span: The span for the exponentially weighted moving average calculation.

    Returns:
        A pandas Series of daily volatility estimates.
    """
    # Use percentage change to calculate daily returns [12]
    returns = close_prices.pct_change()

    # Calculate the EWMA of the standard deviation of returns [12]
    volatility = returns.ewm(span=span, adjust=False).std()
```

```
    return volatility.round(6)
```

This approach allows the trading algorithm to self-regulate, setting wider targets during volatile periods and tighter targets during calm periods, which is a hallmark of a robust and adaptive strategy.[15]

Implementation in Practice: From Signal to Label

Implementing the Triple-Barrier Method correctly involves a structured pipeline that goes beyond simply defining the barriers. It requires careful consideration of how data is sampled and when to trigger a labeling event.

1. Step 1: Data Resampling with Information-Driven Bars
   Financial market activity is not uniformly distributed through time; it exhibits significant seasonality, with activity clustering around the market open and close.19 Using standard time-based bars (e.g., 1-hour or 1-day bars) can introduce statistical biases because a bar during a low-activity period contains far less information than a bar during a high-activity period. A more sophisticated approach, and a prerequisite for high-quality financial machine learning, is to sample data based on information flow rather than the clock. This is achieved using
   information-driven bars.[9] Common types include:
   - **Tick Bars:** A new bar is formed after a fixed number of transactions (ticks).[11]
   - **Volume Bars:** A new bar is formed after a fixed amount of the asset's volume has been traded.[11]
   - **Dollar Bars:** A new bar is formed after a fixed dollar value has been traded. This is often considered the most statistically robust method as it accounts for both volume and price changes.[9]

     By using these bars, each data sample represents a more consistent "packet" of market information, leading to returns that have better statistical properties (e.g., closer to being independent and identically distributed, or IID).11
2. Step 2: Event Sampling with Filters
   Once the data is appropriately resampled, the next step is to determine when to initiate a potential trade and apply the triple barriers. This is known as event-based sampling.9 Instead of labeling every single bar, which would create highly overlapping and redundant samples, we only trigger a labeling event when

something interesting happens. A widely used technique for this is the **Symmetric CUSUM (Cumulative Sum) Filter**.[10] This filter tracks the cumulative sum of price changes and triggers an event when this sum exceeds a certain threshold, which is typically based on the local volatility. This method effectively identifies moments of significant deviation from the recent mean, which may indicate a structural break or the beginning of a new trend.[16]

3. Step 3: Applying the Barriers and Generating Labels
   For each event timestamp generated in Step 2, the TBM is applied. The process involves looking forward in the time series (of information-driven bars) from the event time. The algorithm checks at each subsequent bar whether the high or low price has crossed the pre-calculated upper or lower dynamic barriers. If a horizontal barrier is crossed, the label (+1 or -1) and the time of the touch are recorded. If the vertical barrier (the maximum holding period in terms of number of bars) is reached without a horizontal barrier being touched, the label (0 or based on the sign of the return) and the time of the vertical barrier are recorded.
4. Step 4: Leveraging Python Libraries for Implementation
   This entire pipeline, from bar creation to labeling, is complex to implement from scratch. Fortunately, the quantitative finance community has developed open-source libraries that provide robust implementations. The most notable is mlfinlab, a Python package based directly on the work of Dr. Lopez de Prado, which includes functions for creating information-driven bars, applying CUSUM filters, and executing the Triple-Barrier Method.11 Another specialized library is **triple-barrier**, which focuses specifically on the labeling utility and can be integrated into backtesting pipelines.[22] Using these libraries allows practitioners to focus on strategy development rather than re-implementing these foundational tools. Furthermore, the parameters of the TBM itself, such as the volatility lookback period, the barrier multipliers, and the timeout duration, can be systematically optimized using techniques like grid search or even genetic algorithms to achieve desirable characteristics, such as balanced label proportions across the classes.[17]

The transition from a Fixed-Time Horizon method to a fully-fledged Triple-Barrier Method is not merely an incremental improvement; it is a fundamental reframing of the prediction problem. It moves the model's objective away from abstract forecasting and towards the practical, risk-managed simulation of a trading strategy. By making the labels path-dependent and adaptive to market volatility, the TBM produces a target variable that is a far more realistic and reliable representation of financial success, providing a superior foundation upon which to train any machine learning

algorithm.

Advanced Target Formulations for Classification Models

Once the foundational Triple-Barrier Method is in place, it is possible to build upon its framework to create even more nuanced and powerful target variables for classification models. These advanced formulations allow the model to learn not just the binary outcome of a trade but also the magnitude of the move or the confidence in a signal. This progression represents a move from asking "is it a good trade?" to "how good is the trade?" and ultimately to "how confident am I that this is a good trade?".

Multi-Class Classification: Capturing the Magnitude of Moves

The standard TBM, as described previously, typically yields a binary or ternary outcome: win (+1), loss (-1), or timeout (0). While this is a significant improvement over FTH, it still possesses a limitation: it treats all wins and all losses as equal. A trade that hits the profit-take barrier by a large margin is given the same +1 label as one that barely crosses it. Similarly, the model does not distinguish between a small loss at the vertical barrier and a significant loss at the stop-loss barrier.

To provide the model with more granular information, the target variable can be expanded into a multi-class problem.[25] This involves defining several distinct classes that represent different degrees of success or failure. This allows the model to learn the subtle differences in features that might lead to a modest win versus a strong win, enabling more sophisticated position management and trade selection in a live environment.

A potential set of class definitions could be structured as follows [25]:

- **Class +2 (Strong Win):** The upper barrier (profit-take) is hit first.
- **Class +1 (Modest Win):** The vertical barrier is hit, and the return at exit is positive and above a certain small threshold.
- **Class 0 (Neutral/Hold):** The vertical barrier is hit, and the return at exit is near

zero (within a small dead zone).
- **Class -1 (Modest Loss):** The vertical barrier is hit, and the return at exit is negative and above the stop-loss level.
- **Class -2 (Strong Loss):** The lower barrier (stop-loss) is hit first.

This multi-class approach enriches the information content of the target variable. A model trained on these labels could, for example, learn to prioritize signals that have a high probability of resulting in a +2 outcome, while avoiding those likely to lead to a -2.

However, this approach comes with trade-offs. The primary challenge is the need for a sufficiently large dataset to ensure each class is populated with an adequate number of examples for the model to learn from.[25] Class imbalance can become a more pronounced issue; "strong win" and "strong loss" events are often much rarer than "modest" outcomes. Techniques for handling class imbalance, such as setting class weights in the model (

class_weight='balanced') or using specialized sampling methods, become even more critical.[8] Fortunately, modern gradient boosting libraries like CatBoost and XGBoost, as well as Random Forests, have native support for multi-class classification, making the model implementation straightforward once the labels are generated.[25]

Meta-Labeling: A Two-Stage Approach to Sizing and Selection

Meta-Labeling is a sophisticated, two-stage machine learning technique designed to improve the performance of an existing trading strategy, referred to as the "primary model".[9] Its core purpose is not to generate new trading signals from scratch, but rather to act as an intelligent filter. It addresses the question of

when to trade a signal that has already been identified, effectively decoupling the decision of the trade's *side* (long or short) from the decision of its *size* (including a size of zero, i.e., not trading at all).[29]

This technique is particularly powerful for addressing a common problem in quantitative strategy development: many simple strategies, such as a moving average crossover or a Bollinger Band breakout, tend to have high recall but low precision. That is, they generate a large number of trading signals, capturing most of the real opportunities, but they also produce a high volume of false positives that lead to losses. Meta-labeling aims to systematically identify and filter out these false

positives, thereby increasing the strategy's precision and its overall F1-score (the harmonic mean of precision and recall).[29]

The meta-labeling process works as follows:

1. **Generate Signals with a Primary Model:** First, a primary model is used to generate the initial trading signals (e.g., long, short, or hold). This primary model can be a simple technical rule (like the Bollinger Band strategy described in [11]), a model based on economic theory, or even another, simpler machine learning model designed for high recall.[30]
2. **Determine Ground Truth with TBM:** For every signal generated by the primary model, the Triple-Barrier Method is used to determine the actual outcome of the trade. This generates a binary label: 1 if the trade would have been profitable (hit the upper barrier) and 0 otherwise (hit the lower or vertical barrier). This TBM label serves as the ground truth for the primary model's success on each specific signal.
3. **Train the Secondary "Meta" Model:** A second, more sophisticated machine learning model (the "meta model") is then trained.
    - **Features:** The input features for the meta model are typically the same features that were available to the primary model at the time the signal was generated.[29]
    - **Target (The Meta-Label):** The target variable for this secondary model is the "meta-label." This is a binary label that indicates whether the primary model's prediction was correct. That is, meta_label = 1 if the primary model generated a signal and the TBM outcome was 1 (a true positive). The meta_label is 0 if the primary model's signal resulted in a TBM outcome of 0 (a false positive).[29] The meta model is learning to predict the probability that the primary model's signal is a true positive.
4. **Inference and Bet Sizing:** In a live trading environment, the system operates in two stages. First, the primary model must generate a signal (e.g., "go long"). Second, this signal is fed into the trained meta model. A trade is only executed if the meta model predicts a high probability of success (e.g., > 0.9, as suggested in [30]). The probability output from the meta model is not just a filter; it is a powerful tool for
   bet sizing. A higher predicted probability can justify a larger position size, while a lower probability might warrant a smaller position or no trade at all. This aligns the size of the bet with the model's confidence, a critical component of risk management.[33]

The Meta-Labeling Debate: A Nuanced View

While powerful, meta-labeling is not a "silver bullet" and its application requires a nuanced understanding of its mechanics and limitations.[32] A critical perspective argues that if both the primary and meta models are machine learning algorithms of similar complexity and are trained on the same feature set, there is no logical reason why the secondary model should be able to extract additional information. As one practitioner aptly put it, "You cannot squeeze the same orange twice".[32]

This critique suggests that the value of meta-labeling is not in generating new alpha from the same information, but rather in its intelligent application as a form of structured ensembling or filtering. Meta-labeling tends to be most effective under specific conditions:

1. **Information Asymmetry:** The meta model provides the most value when there is an information or complexity advantage. For instance, if the primary model is a simple, linear, or "white box" model (e.g., a strategy based on fundamental economic theory), a non-linear machine learning model (like a Random Forest or Gradient Boosting model) used as the meta model can capture complex patterns and relationships that the primary model misses.[29]
2. **Improving Low-Precision Strategies:** It is tailor-made for situations where the primary strategy has high recall but suffers from a high rate of false positives. Its primary function is to improve precision.[29]
3. **Additional Features for the Meta Model:** The performance of the meta model can be significantly enhanced by providing it with features that the primary model does not have access to. These could include features describing the broader market state or regime, such as market-wide volatility or correlation metrics. This allows the meta model to learn the specific market conditions under which the primary model is likely to succeed or fail.[21]

Practitioners must also be aware of the implementation challenges. Meta-labeling inherently increases the overall complexity of the trading system, which can elevate the risk of overfitting and make hyperparameter tuning more difficult.[32] Furthermore, it is absolutely critical to use strictly separate or purged datasets for training the primary and secondary models to prevent information leakage, which could lead to a falsely overconfident meta model.[21]

In essence, the progression from binary TBM to multi-class classification and finally to meta-labeling reflects an increasing sophistication in the definition of the trading problem. It moves the model's objective from identifying simple win/loss scenarios to quantifying the potential outcome and, ultimately, to assessing the confidence in the trading signal itself. This hierarchy provides a powerful toolkit for developing more robust, nuanced, and profitable classification-based trading strategies.

Regression-Based and Alternative Target Paradigms

While classification-based targets like the Triple-Barrier Method form the bedrock of modern financial machine learning for directional trading, a separate class of strategies can be built by framing the prediction problem as a regression task or by leveraging entirely different machine learning paradigms. These approaches often target different aspects of market behavior, such as volatility or risk-adjusted returns, and can unlock new and complementary sources of alpha. The choice of target variable in this context is a direct reflection of the specific investment strategy's objective.

Predicting Future Realized Volatility

Instead of predicting the direction of price movement, a model can be trained to predict the magnitude of future price fluctuations—that is, future realized volatility. This is a powerful strategy for several reasons. Volatility is a tradable asset class in its own right through instruments like VIX futures and options. Furthermore, accurate volatility forecasts are critical inputs for a wide range of financial applications, including options pricing (e.g., the Black-Scholes model), dynamic risk management, and risk-parity portfolio allocation strategies, where asset weights are adjusted based on their volatility.[36]

- **Defining the Target Variable:** The target variable, y, for a volatility forecasting model is the **future realized volatility**. This is typically calculated as the standard deviation of logarithmic returns over a specific future period. For example, to predict one-month-ahead volatility, the target for a given day would be the standard deviation of returns over the subsequent 21 trading days.[37] This is a

continuous, real-valued target, making this a regression problem.

- **Features and Model Choice:** The features (predictors) for a volatility model often include historical measures of volatility (a strong predictor due to volatility clustering), lagged returns, moving averages of price, and various technical indicators that capture market momentum and trend, such as the Relative Strength Index (RSI) and Moving Average Convergence Divergence (MACD).[37] While classical econometric models like GARCH (Generalized Autoregressive Conditional Heteroskedasticity) are the traditional benchmark for this task, machine learning models such as Random Forest Regressors, Gradient Boosting Regressors (like CatBoost), and deep learning models like LSTMs are increasingly used due to their ability to capture complex, non-linear relationships in the data.[38]
- **Python Implementation Concept:** A practical implementation would involve the following steps:
  1. Calculate daily returns from the adjusted close prices.
  2. Calculate historical volatility over a rolling window (e.g., 21 days) to use as a feature.
  3. Define the target variable by calculating the rolling standard deviation over a *future* window and then shifting it back in time so that it aligns with the features that would have been available at the time of prediction.
  4. Train a regression model (e.g., RandomForestRegressor) on the historical features to predict the future volatility target.

```
Python
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Assume 'data' is a pandas DataFrame with 'Adj Close' prices
# Step 1: Calculate returns
data = np.log(data['Adj Close'] / data['Adj Close'].shift(1))

# Step 2: Create historical volatility feature
data['Volatility_21d'] = data.rolling(window=21).std() * np.sqrt(252)

# Step 3: Create the future volatility target variable [42]
data['Future_Volatility_21d'] = data['Volatility_21d'].shift(-21)

# Prepare data for ML
data.dropna(inplace=True)
X = data[['Volatility_21d']] # Add other features like RSI, MACD, etc.
y = data['Future_Volatility_21d']
```

```
# Step 4: Train the model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Evaluate the model
predictions = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, predictions))
print(f'RMSE for Future Volatility Prediction: {rmse}')
```

Direct Optimization of Financial Metrics: The Sharpe Ratio Target

A more ambitious and direct approach to portfolio construction is to train a model to optimize a desired financial metric itself. Instead of predicting an intermediate variable like returns, which are notoriously noisy, this method attempts to directly maximize the portfolio's **Sharpe Ratio**, the measure of risk-adjusted return.[44] This is considered a "holy grail" in quantitative finance because it aligns the model's objective function directly with the portfolio manager's end goal.

The primary challenge with this approach is that the Sharpe Ratio is a complex, non-differentiable function of portfolio weights. This makes it unsuitable for standard gradient-based optimization algorithms used in most machine learning models (e.g., deep neural networks). The solution lies in using non-gradient-based optimization techniques, with **Genetic Programming (GP)** being a particularly powerful and flexible choice.[47]

Genetic Programming is an evolutionary algorithm that works by creating a "population" of candidate solutions (in this case, trading strategies or portfolio allocation functions). These solutions are represented as mathematical formulas or tree structures. The algorithm then iteratively applies principles of natural selection: the "fittest" solutions—those that produce the highest Sharpe Ratio on a validation dataset—are more likely to "survive" and "reproduce" (by combining parts of their formulas) to create the next generation of solutions. Over many generations, the population evolves towards highly optimized strategies that directly maximize the Sharpe Ratio.[47] This computationally intensive method is used by sophisticated quantitative funds to discover novel alpha factors and construct portfolios that are

optimized for risk-adjusted performance from the ground up.

Predicting Tail Risk

While most models focus on predicting the expected or average outcome, the most impactful events in finance occur in the tails of the return distribution—the rare but severe market crashes often referred to as "black swans".[48] Predicting tail risk is a specialized subfield of financial machine learning aimed at forecasting these high-impact events.

The target variable for a tail risk model can be formulated in several ways:

- **Binary Classification:** A simple approach is to define a binary target where 1 represents the occurrence of a tail event and 0 represents normal market conditions. A tail event could be defined as a daily or weekly return falling below a certain threshold, such as -3 standard deviations from the mean, or a stock price crash of a certain magnitude (e.g., > 20% drop in a month).[50]
- **Regression:** A more nuanced approach is to predict a continuous measure of tail risk directly. This could involve forecasting metrics like **Value-at-Risk (VaR)**, which estimates the potential maximum loss over a specific time frame at a certain confidence level, or **Expected Shortfall (ES)**, which measures the expected loss given that the loss exceeds the VaR threshold.[51]

Several machine learning methodologies are suited for this task. **Deep Quantile Regression** can be used to directly model and forecast specific quantiles of the return distribution, making it a natural fit for predicting VaR.[51] Anomaly detection algorithms, such as

Isolation Forest, can be trained on market data to identify unusual patterns of behavior that may precede a tail event.[49] Furthermore, models can be trained on fundamental or market-based indicators of financial distress, such as the

distance-to-default (DtD) metric derived from the Merton model, which has been shown to be a strong predictor of stock price crashes.[50]

A Note on Reinforcement Learning: Redefining the "Target"

Finally, it is worth noting the paradigm of **Reinforcement Learning (RL)**, which represents a fundamental departure from supervised learning.[52] In an RL framework, there is no explicitly defined target variable

y. Instead, the system consists of:

- An **Agent:** The trading algorithm.
- An **Environment:** The financial market.
- A set of **Actions:** The decisions the agent can take (e.g., buy, sell, hold, adjust position size).
- A **Reward Signal:** A metric that provides feedback to the agent after each action.

The "target" in RL is implicit: the agent's objective is to learn a **Policy**—a strategy for choosing actions—that maximizes its cumulative future reward.[55] The reward signal can be defined in various ways, such as the profit and loss of a trade, the change in portfolio value, or even the portfolio's Sharpe Ratio over a recent period. The agent learns through a process of trial and error, exploring different actions and observing their impact on the reward, gradually converging on an optimal strategy. This makes RL exceptionally well-suited for dynamic, sequential decision-making problems like optimal trade execution, market making, and continuous portfolio management.[52] While more complex to implement, RL represents a frontier in quantitative finance, moving beyond prediction to direct, optimized action.

Evaluation and Backtesting: The Primacy of Financial Metrics

The development of a sophisticated trading model does not end with the selection of an advanced target variable and the training of a classifier or regressor. The final and most crucial step is a rigorous and realistic evaluation of the resulting strategy's performance. In this domain, standard machine learning evaluation metrics are not only insufficient but can be dangerously misleading. The ultimate arbiter of a strategy's success is not its statistical accuracy but its ability to generate consistent, risk-adjusted profits in a simulated trading environment.

Beyond Accuracy: Why Standard ML Metrics Fail in Finance

For classification problems, metrics like accuracy, precision, recall, and F1-score are the default tools in many machine learning domains. However, they are fundamentally ill-suited for evaluating financial trading models due to several critical blind spots.[3]

- **The Accuracy Trap:** A model's accuracy—the proportion of correct predictions—is perhaps the most deceptive metric. A trading model can achieve very high accuracy and still be unprofitable. Consider a strategy that makes 100 trades. It correctly predicts 99 of them, which are small wins of $1 each, for a total gain of $99. However, the one incorrect prediction results in a single catastrophic loss of $100. The model's accuracy is an impressive 99%, but its net profit is -$1. Standard classification metrics are blind to the *magnitude* of the outcomes; they treat a small win and a large win, or a small loss and a large loss, as equivalent events.[4]
- **The Class Imbalance Problem:** Financial markets are inherently imbalanced. Profitable trading opportunities (the positive class) are often much rarer than periods of sideways or unprofitable movement. In such a scenario, a naive model that always predicts "no trade" (the majority class) can achieve very high accuracy while having zero practical value. Metrics like precision, recall, and the F1-score offer a more nuanced view in the face of imbalance, but they still fail to incorporate the financial consequences of the predictions.[3]
- **Misalignment with the Business Objective:** The fundamental business objective of a trading strategy is not to be "correct" in its predictions but to generate a positive risk-adjusted return on capital.[3] Evaluation metrics must be directly aligned with this financial goal. A model should be judged on its simulated profit and loss (P&L) curve, not on its confusion matrix.

The Quant's Toolkit: Risk-Adjusted Performance Metrics

To properly evaluate a trading strategy, quantitative finance professionals rely on a suite of metrics derived from the strategy's backtested P&L curve. These metrics provide a holistic view of performance by integrating return, risk, and the stability of the strategy over time. The following are essential components of any robust backtesting evaluation.[57]

Metric Name	Formula	Interpretation	Strategic Relevance
Sharpe Ratio	σpRp−Rf	Measures the excess return (Rp−Rf) generated per unit of total risk (volatility, σp).	The primary measure of risk-adjusted performance. A higher ratio indicates a more efficient strategy. Essential for comparing different strategies on a level playing field.[44]
Sortino Ratio	σdRp−Rf	Similar to the Sharpe Ratio, but only considers downside deviation (σd) in the denominator.	Focuses on "bad" volatility (losses) while ignoring "good" volatility (gains). Better reflects an investor's aversion to losses and is crucial for evaluating capital preservation.[57]
Maximum Drawdown (MaxDD)	Peak ValueTrough Value−Peak Value	The largest percentage decline from a portfolio's peak value to a subsequent trough.	A critical measure of the potential "pain" or worst-case loss a strategy can inflict. Informs risk limits and the psychological resilience required to stick with the strategy.[57]
Calmar Ratio	MaxDDAnnualized Return	Measures the annualized return relative to the maximum drawdown.	Provides a direct assessment of return generated per unit of the worst historical loss. A high Calmar Ratio is desirable for strategies focused on smooth returns and drawdown control.[57]
Profit Factor	Gross LossGross Profit	The ratio of total profits from all winning trades to total losses from all	A straightforward measure of overall profitability. A value greater than 1

		losing trades.	indicates a profitable strategy. A high value suggests robust profitability.[57]
Win Rate	Total Number of TradesNumber of Winning Trades	The percentage of trades that were profitable.	A simple measure of the strategy's success rate. Must be analyzed alongside the average profit/loss per trade to be meaningful.[57]
Turnover	Measures the frequency at which assets are bought and sold.	Indicates the cost-efficiency of a strategy. High turnover leads to higher transaction costs (commissions, slippage), which can significantly erode net returns.[57]	

There is a deep and logical connection between the choice of target variable and the appropriate evaluation framework. A simple FTH target might naively lead a developer to focus on classification accuracy. However, the adoption of a TBM target, which is explicitly designed to model the P&L of a trade, naturally compels the developer to evaluate the resulting strategy using metrics derived from a P&L curve, such as the Sharpe and Sortino ratios. The TBM forces the modeler to think like a portfolio manager.

Even when using a regression target, such as predicting future volatility, a two-tiered evaluation is necessary. The regression model itself would be assessed using standard regression metrics like Root Mean Squared Error (RMSE). However, the ultimate success of the *trading strategy that utilizes these volatility predictions* must still be judged by the financial metrics in the toolkit above. This highlights a critical hierarchy in evaluation: model-level statistical metrics are necessary for diagnosing the predictive component, but they are not sufficient. The ultimate arbiter of success is the strategy-level financial performance, which accounts for the integrated system of

signal generation, risk management, and execution. The entire process—from information-driven data sampling, to event-based filtering, to path-dependent TBM labeling, and finally to evaluation with financial metrics—should be viewed as a single, integrated system designed to create a robust and realistic simulation of a real-world trading process.

Conclusion and Recommendations

The journey from a novice to a sophisticated quantitative strategist is marked by a deepening understanding that the definition of the problem is often more important than the choice of algorithm to solve it. For a machine learning-based trading model, the target variable is the definition of the problem. The initial approach of using a Fixed-Time Horizon (FTH) label, while simple to implement, is fundamentally misaligned with the path-dependent and risk-managed nature of financial trading. It teaches the model to solve an abstract forecasting problem that often fails to translate into real-world profitability.

This report has systematically deconstructed the flaws of the FTH method and presented a clear, hierarchical path toward more robust and professionally recognized alternatives. The cornerstone of this advancement is the **Triple-Barrier Method (TBM)**, which reframes the prediction task from one of forecasting to one of simulating a realistic trade. By incorporating profit-taking, stop-loss, and time-based exit conditions directly into the label, the TBM ensures that the machine learning model is trained on outcomes that reflect actual trading mechanics. The use of dynamic, volatility-adjusted barriers further enhances this realism, allowing the strategy to adapt to changing market conditions.

Building upon this foundation, advanced classification techniques like **multi-class labeling** and **meta-labeling** offer further refinements. Multi-class labels provide the model with more granular information about the magnitude of potential outcomes, while meta-labeling introduces a powerful two-stage framework for filtering signals and informing bet size, thereby improving the precision of an underlying strategy. Beyond classification, regression-based targets, such as predicting **future volatility** or **tail risk**, open the door to entirely new strategic applications in areas like options trading and risk management. Finally, frontier paradigms like **Genetic Programming** for direct Sharpe Ratio optimization and **Reinforcement Learning** represent the

future of the field, moving from prediction to direct, optimized action.

Crucially, the choice of a more sophisticated target variable necessitates a corresponding evolution in evaluation methodology. Standard machine learning metrics like accuracy are insufficient and misleading. The ultimate success of any trading strategy must be judged by a rigorous backtest that employs a suite of risk-adjusted financial metrics, including the **Sharpe Ratio, Sortino Ratio, and Maximum Drawdown**.

Based on this comprehensive analysis, the following actionable roadmap is recommended for enhancing the current algorithmic trading module:

1. **Immediate Priority: Implement the Triple-Barrier Method.** This is the single most impactful improvement that can be made. The initial implementation should replace the "10% gain in 7 days" logic with a TBM framework. Start by setting the horizontal barriers as fixed multiples of the entry price (e.g., a 2:1 profit-take to stop-loss ratio) and a reasonable time-out period. The immediate next step should be to implement dynamic, volatility-adjusted barriers, which will make the model significantly more adaptive.
2. **Next Step for Refinement: Explore Meta-Labeling.** Once a robust TBM-based model is operational, the next logical step is to implement meta-labeling. The existing CatBoost model can serve as the "primary model." A second CatBoost model can then be trained as the "meta model" to predict the probability of success for the signals generated by the primary model. This will serve as an intelligent filter to reduce false positives and will provide a principled foundation for dynamic position sizing.
3. **Advanced Exploration for New Strategies:** The regression and alternative targets discussed—such as predicting volatility, tail risk, or the Sharpe Ratio—should be viewed not as direct upgrades to the current directional strategy, but as building blocks for *entirely new and complementary strategies*. For instance, a volatility prediction model could form the basis of an options-based strategy that runs alongside the directional stock-picking model.

By systematically progressing through these stages of target variable sophistication, a quantitative trading model can be transformed from a simple academic exercise into a robust, adaptive, and financially-grounded system. The core principle remains constant: the ultimate target is not just a label, but a consistently profitable, risk-managed strategy that stands up to rigorous financial evaluation.

Works cited

1. What is a Target Variable in Machine Learning? - H2O.ai, accessed June 28, 2025, https://h2o.ai/wiki/target-variable/
2. The Hardest Part: Defining A Target For Classification | by Chris Bruehl - Medium, accessed June 28, 2025, https://medium.com/data-science/the-hardest-part-defining-a-target-for-classification-50c34d37e0b8
3. How to Choose the Right Evaluation Metric for Your ML Model - Dataaspirant, accessed June 28, 2025, https://dataaspirant.com/right-evaluation-metric/
4. Creative target variables for supervised ML? : r/algotrading - Reddit, accessed June 28, 2025, https://www.reddit.com/r/algotrading/comments/1c2jtut/creative_target_variables_for_supervised_ml/
5. Target Variables - Graphite Note, accessed June 28, 2025, https://graphite-note.com/understanding-target-variables-in-machine-learning/
6. The Triple Barrier Method: A New Standard for Investment Labeling and Analysis, accessed June 28, 2025, https://medium.datadriveninvestor.com/the-triple-barrier-method-a-new-standard-for-investment-labeling-and-analysis-1a525a0a2f46
7. Chapter 3 Labeling - Advances in Financial Machine Learning [Book] - O'Reilly Media, accessed June 28, 2025, https://www.oreilly.com/library/view/advances-in-financial/9781119482086/c03.xhtml
8. Labeling Financial Data | RiskLab AI, accessed June 28, 2025, https://www.risklab.ai/research/financial-data-science/labeling
9. Advances in Financial Machine Learning | Wiley, accessed June 28, 2025, https://www.wiley.com/en-us/Advances+in+Financial+Machine+Learning-p-9781119482086
10. Improve Your ML Model With Better Labels - Artificial Intelligence in Plain English, accessed June 28, 2025, https://ai.plainenglish.io/start-using-better-labels-for-financial-machine-learning-6eeac691e660
11. Does Meta Labeling Add to Signal Efficacy? - Hudson & Thames, accessed June 28, 2025, https://hudsonthames.org/does-meta-labeling-add-to-signal-efficacy-triple-barrier-method/
12. Triple Barrier Method: Python | GPU | Nvidia - QuantInsti Blog, accessed June 28, 2025, https://blog.quantinsti.com/triple-barrier-method-gpu-python/
13. The Triple Barrier Method: A Python GPU-based Computation – Part I - Interactive Brokers, accessed June 28, 2025, https://www.interactivebrokers.com/campus/ibkr-quant-news/the-triple-barrier-method-a-python-gpu-based-computation-part-i/
14. The Triple Barrier Method: Labeling Financial Time Series for ML in Elixir | by Yair Oz, accessed June 28, 2025, https://medium.com/@yairoz/the-triple-barrier-method-labeling-financial-time-series-for-ml-in-elixir-e539301b90d6

15. Dynamic Take Profit Stop Loss Based on Volatility - INSIGHTS - Trade View, accessed June 28, 2025, https://insights.tradeview.com.au/algo-trading-talk/dynamic-volatility-take-profit-stop-loss-strategy/
16. Flexible horizon in Triple Barrier Method - Quantitative Finance Stack Exchange, accessed June 28, 2025, https://quant.stackexchange.com/questions/49314/flexible-horizon-in-triple-barrier-method
17. Enhanced Genetic-Algorithm-Driven Triple Barrier Labeling Method ..., accessed June 28, 2025, https://www.mdpi.com/2227-7390/12/5/780
18. Using stop-loss and take-profit orders - Deriv, accessed June 28, 2025, https://deriv.com/academy/lessons/using-stop-loss-and-take-profit-orders
19. Data Labelling - Mlfin.py, accessed June 28, 2025, https://mlfinpy.readthedocs.io/en/latest/Labelling.html
20. python Archives - Hudson & Thames, accessed June 28, 2025, https://hudsonthames.org/tag/python/
21. Meta labeling in Cryptocurrencies Market. | by Quang Khải Nguyễn Hưng | Medium, accessed June 28, 2025, https://medium.com/@liangnguyen612/meta-labeling-in-cryptocurrencies-market-95f761410fac
22. triple-barrier·PyPI, accessed June 28, 2025, https://pypi.org/project/triple-barrier/
23. mchiuminatto/triple_barrier: Semi vectorized trades labeler ... - GitHub, accessed June 28, 2025, https://github.com/mchiuminatto/triple_barrier
24. Stock Price Prediction Using Triple Barrier Labeling and Raw OHLCV Data: Evidence from Korean Markets - arXiv, accessed June 28, 2025, https://arxiv.org/html/2504.02249v2
25. (PDF) MULTICLASS CLASSIFIERS FOR STOCK PRICE PREDICTION: A COMPARISON STUDY - ResearchGate, accessed June 28, 2025, https://www.researchgate.net/publication/361582393_MULTICLASS_CLASSIFIERS_FOR_STOCK_PRICE_PREDICTION_A_COMPARISON_STUDY
26. Machine Learning Classification Methods and Factor Investing - - Alpha Architect, accessed June 28, 2025, https://alphaarchitect.com/machine-learning-classification-methods-and-factor-investing/
27. (PDF) A multiclass classification model for stock news based on structured data, accessed June 28, 2025, https://www.researchgate.net/publication/303772899_A_multiclass_classification_model_for_stock_news_based_on_structured_data
28. How to Solve a Multi Class Classification Problem with Python?, accessed June 28, 2025, https://www.projectpro.io/article/multi-class-classification-python-example/547
29. Meta Labeling (A Toy Example) - Hudson & Thames, accessed June 28, 2025, https://hudsonthames.org/meta-labeling-a-toy-example/
30. dreyhsu/Meta_Labeling: Meta labeling is a method of determining the size of the bet. - GitHub, accessed June 28, 2025, https://github.com/dreyhsu/Meta_Labeling

31. The Journal of Finance and Data Science-SciEngine, accessed June 28, 2025, https://www.sciengine.com/JFDS/issue/8/1
32. Why Meta-Labeling Is Not a Silver Bullet by Francesco Baldisserri - QuantConnect.com, accessed June 28, 2025, https://www.quantconnect.com/forum/discussion/14706/why-meta-labeling-is-not-a-silver-bullet/
33. Meta-Labeling: Solving for Non Stationarity and Position Sizing - YouTube, accessed June 28, 2025, https://www.youtube.com/watch?v=WbgglcXfEzA
34. hudson-and-thames/meta-labeling: Code base for the meta ... - GitHub, accessed June 28, 2025, https://github.com/hudson-and-thames/meta-labeling
35. Meta-Labeling: Calibration and Position Sizing - Portfolio Management Research, accessed June 28, 2025, https://www.pm-research.com/content/iijjfds/5/2/23
36. Top 10 Quantitative Analyst Interview Questions, accessed June 28, 2025, https://www.interviews.chat/questions/quantitative-analyst
37. Machine Learning & Volatility Forecasting: Avoiding the Look-Ahead ..., accessed June 28, 2025, https://medium.com/@contact_9367/machine-learning-volatility-forecasting-avoiding-the-look-ahead-trap-6ff63c8c703c
38. Analysis and Predict the Volatility of Nasdaq-100 through Machine Learning — MSMF, accessed June 28, 2025, https://www.monashmsmf.org/institute/publications/analysis-and-predict-the-volatility-of-nasdaq-100-through-machine-learning
39. Volatility And Measures Of Risk-Adjusted Return With Python - Interactive Brokers, accessed June 28, 2025, https://www.interactivebrokers.com/campus/ibkr-quant-news/volatility-and-measures-of-risk-adjusted-return-with-python/
40. Volatility And Measures Of Risk-Adjusted Return With Python - QuantInsti Blog, accessed June 28, 2025, https://blog.quantinsti.com/volatility-and-measures-of-risk-adjusted-return-based-on-volatility/
41. How to Predict Stock Volatility Using Python | Berkindale Analytics, accessed June 28, 2025, https://berkindale.com/en/newsroom/how-to-predict-stock-volatility-using-python/
42. Python for Machine Learning-Powered Volatility Forecasting | by SR ..., accessed June 28, 2025, https://medium.com/@deepml1818/volatility-forecasting-is-crucial-in-quantitative-finance-as-it-directly-affects-risk-management-2a9fd6387c2b
43. Forecasting Implied Volatility using Machine Learning - Quantra by QuantInsti, accessed June 28, 2025, https://quantra.quantinsti.com/glossary/Forecasting-Implied-Volatility-using-Machine-Learning
44. Sharpe Ratio: Definition, Formula, and Examples - Investopedia, accessed June 28, 2025, https://www.investopedia.com/terms/s/sharperatio.asp
45. TIME-VARYING SHARPE RATIOS AND MARKET TIMING - NYU Stern, accessed

June 28, 2025, https://pages.stern.nyu.edu/~rwhitela/papers/tvsharpe.pdf
46. The Journal of Financial Data Science - Scilit, accessed June 28, 2025, https://www.scilit.com/sources/45916
47. Maximizing the Sharpe Ratio: A Genetic Programming Approach, accessed June 28, 2025, https://afajof.org/management/viewp.php?n=37996
48. Estimating Tail Risk in Neural Networks - Alignment Research Center, accessed June 28, 2025, https://www.alignment.org/blog/estimating-tail-risk-in-neural-networks/
49. AI for Tail Risk Management - InsiderFinance Wire, accessed June 28, 2025, https://wire.insiderfinance.io/ai-for-tail-risk-management-8bd79c4aa7b4
50. Real-life experience: Using ML and distance-to-default to predict distress risk | Robeco USA, accessed June 28, 2025, https://www.robeco.com/en-us/insights/2024/02/real-life-experience-using-ml-and-distance-to-default-to-predict-distress-risk
51. Learning about tail risk: Machine learning and combination with regularization in market risk management | Request PDF - ResearchGate, accessed June 28, 2025, https://www.researchgate.net/publication/386227412_Learning_about_tail_risk_machine_learning_and_combination_with_regularization_in_market_risk_management
52. 7 Applications of Reinforcement Learning in Finance and Trading - Neptune.ai, accessed June 28, 2025, https://neptune.ai/blog/7-applications-of-reinforcement-learning-in-finance-and-trading
53. Reinforcement Learning in Finance - ExtractAlpha, accessed June 28, 2025, https://extractalpha.com/2024/08/22/reinforcement-learning-in-finance/
54. Reinforcement Learning in Finance | Planergy Software, accessed June 28, 2025, https://planergy.com/blog/reinforcement-learning-in-finance/
55. Deep Reinforcement Learning for Trading: Strategy Development & AutoML - MLQ.ai, accessed June 28, 2025, https://blog.mlq.ai/deep-reinforcement-learning-trading-strategies-automl/
56. How to Choose the Right Metric for Your Model - Codefinity, accessed June 28, 2025, https://codefinity.com/blog/How-to-Choose-the-Right-Metric-for-Your-Model
57. Essential Backtesting Metrics in Algo Trading - uTrade Algos, accessed June 28, 2025, https://www.utradealgos.com/blog/what-are-the-key-metrics-to-track-in-algo-trading-backtesting
58. Advanced Backtesting Metrics | TrendSpider Learning Center, accessed June 28, 2025, https://trendspider.com/learning-center/advanced-backtesting-metrics/
59. Key Trading Metrics: Sortino Ratio, Sharpe Ratio, Volatility, and Risk - PineConnector, accessed June 28, 2025, https://www.pineconnector.com/blogs/pico-blog/key-trading-metrics-sortino-ratio-sharpe-ratio-volatility-and-risk-reward-ratio
60. Strategy Evaluation using the Sharpe and Sortino Ratios | FXCM Markets, accessed June 28, 2025,

https://www.fxcm.com/markets/insights/strategy-evaluation-using-the-sharpe-and-sortino-ratios/