VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTATIONAL AND DATA MODELING

Computer Modeling 2026 Master's Thesis

Integrating sentiment analysis into time series forecasting for Bitcoin prices

Sentimentų analizės pritaikymas Bitkoino kainų prognozavimui laiko eilutėse

Done by:
Povilas Mažeika signature

Supervisor:
dr. Ernestas Filatovas

Vilnius
2026

Contents

Abstract

Financial markets are often perceived as unpredictable due to their chaotic nature and the overwhelming volume of information influencing them. However, with the advancement of data processing capabilities, leveraging this information has become more feasible. In this study, we address two often-overlooked aspects of market prediction using news sentiment analysis.

First, we treat sentiment not as a classification problem but as a quantifiable regressor, enabling more nuanced modeling of market impact.

Second, we explore the contextual interpretation of sentiment, for example - a headline such as "$120 million in long positions liquidated in the last hour" may appear negative but could imply a market reversal, representing a potential positive opportunity. To investigate this, we firstly try to define our custom headline-to-price impact scores from Bitcoin price to our headlines dataset and select the best one for further proceeding. Later, we employ regression-based machine learning models, as well as try to adapt fine-tuned financial language models from classification to regression tasks and fine-tuning to our weights.

After choosing viable sentiment analysis approach, we integrate the derived sentiment scores with market data and Bitcoin technical indicators to integrate it to Transformer-based time series forecasting model. Using model, we evaluate the performance in predicting Bitcoin during different market conditions through backtesting with various trading strategies.

Our findings indicate that sentiment analysis is less effective with minimal inputs (price and volume only), but performs better when integrated into more advanced datasets that combine price data with Bitcoin technical indicators and broader market health signals.

Santrauka

Sentimentų analizės pritaikymas laiko eilučių kainos prognozei kriptovaliutų rinkoje

Finansų rinka yra apibūdinama kaip nenuspėjama dėl savo chaotiškumo ir milžiniško informacijos kiekio kuris įtakoja ją. Tačiau su šių laikų informacijos apdorojimo galimybėmis tampa daug lengviau šia informaciją suvaldyti ir ja pasinaudoti. Šiame darbe fokusuojames į naujienų informacijos sentimentą, ypač į du aspektus kuriuos gana retai sutinkami pastarojo meto tyrimuose:

Pirmasis - teksto sentimento naudojimas ne kaip klasifikacijos problema, tačiau kaip regresija, tai leistu tikslingesni rinkos modeliavimą;

Antrasis - tikslingesnė teksto interpretacija rinkoje, kaip pavyzdžiui - "100 milijonų dolerių maržos pirkimo pozicija buvo likviduota" turi neigiamą konotaciją, tačiau rinkoje ji gali reikšti, kad yra pasiektas "dugnas" ir rinka turėtu elgtis pozityviau artimiausiu metu.

Tyrimui visu pirma mes sukūrėme antraštės poveikio kainai balus, kurie yra priklausomas nuo Bitkoino kainos ir apyvartos ir bandėme išsrinkti geriausią su kuriuo tęsti darbus. Toliau mes naudojome regresijos mašininio mokymosi modelius kartu su finansų terminams pritaikytus kalbos modeliais juos pritaikant iš klasifikacijos į regresiją ir derinant pagal mūsų išvestus naujus svorius.

Po to pasirinkę tinkamą sentimentų analizės metodiką, mes integravome išvestus sentimentų balus su rinkos duomenimis ir Bitkoino techniniais indikatoriais ir įtraukėme juos į transformacijų pagrindu sukurtą laiko eilučių prognozavimo modelį. Naudodami šį modelį, mes įvertinome Bitkoino prognozavimą skirtingomis rinkos sąlygomis per prekybos testavimą su įvairiomis prekybos strategijomis.

Rezultatai parodė, kad sentimentų analizė veikia prasčiau, kai modeliai remiasi tik minimaliais duomenimis (kaina ir apyvarta), tačiau veikia geriau, kai ji įtraukiama į išsamesnius duomenų rinkinius, kurie apjungia kainos duomenis su Bitkoino techniniais indikatoriais ir platesnės rinkos sveikatos signalais.

Introduction

The predictability of financial markets has long been debated. The Efficient Market Hypothesis (EMH) argues that asset prices fully reflect all available information, making consistent profits through short-term market predictions impossible [21]. However, critics argue that markets exhibit anomalies and behavioral biases that challenge this notion. Empirical evidence suggests that stock prices do not always reflect all available information instantly, and psychological factors like overconfidence and herd behavior lead to systematic deviations from fundamental values [20]. Behavioral finance has emerged as an alternative framework, demonstrating that incorporating psychological factors and sentiment can better explain market phenomena such as momentum and reversal effects, paving the way for data-driven approaches like sentiment analysis [26, 10].

Khalil and Pipa research has shown that controlling and analyzing information can significantly enhance market prediction accuracy. A study utilizing Natural Language Processing (NLP) and Long Short-Term Memory (LSTM) models found that sentiment analysis of various news sources - including mainstream media, social media, expert opinions, and company announcements has a measurable impact on stock direction within 3-4 hours [13]. By constructing a sentiment index through NLP or other language processing, the study demonstrated that models incorporating textual data significantly outperformed those without, proving that it takes some time to incorporate new information into the market price and it is one of studies which shows that EMH is not really viable in the todays current market conditions while using proper information analysis tools.

As discussed in section 1.3, existing research on market forecasting using sentiment analysis often relies on prebuilt sentiment models or custom feature extraction methods. However, these approaches rarely explore fine-tuning for better adaptation and, more critically, do not incorporate real market data (price impact, volume, or volatility) directly into sentiment analysis. This creates a significant gap: positive news does not always lead to positive price impact and vice versa. For example, the headline "150 million dollars of long positions have been liquidated in the market over the past 24 hours" carries negative sentiment but could signal a buying opportunity after a liquidation cascade.

Research Gap and Problem Statement

Despite the rapid growth of sentiment analysis in natural language processing, limited research has approached it as a regression task that directly measures real-world market impact. Most existing models treat sentiment categorically (positive, negative, neutral) and ignore the influence of actual financial data in model development. This study addresses this gap by developing and fine-tuning sentiment analysis models using cryptocurrency news headlines paired with measured market impact, then integrating the best-performing model into time series forecasting to evaluate whether sentiment features improve prediction accuracy when combined with traditional technical indicators.

Overall Goal and Research Questions

The overall **goal** of this thesis is to create a price change prediction model for the cryptocurrency market by integrating sentiment analysis of news headlines with traditional technical indicators, aiming to enhance forecasting accuracy and market understanding.

To achieve this goal, we formulate the following research questions:

- **RQ1:** How can a news impact score be derived from price and volume data to quantify the real market effect of cryptocurrency-related headlines?
- **RQ2:** Can fine tuned sentiment models using custom cryptocurrency datasets outperform pre-trained models and traditional NLP approaches in predicting market impact?
- **RQ3:** Does integrating sentiment analysis with traditional technical indicators improve the accuracy of cryptocurrency price forecasting models?

Research Objectives and Tasks

To address these research questions, we define two main research objectives:

- **Objective 1 (RQ1, RQ2):** Develop and evaluate fine tuned sentiment analysis models capable of measuring the market impact of cryptocurrency news headlines.
- **Objective 2 (RQ3):** Integrate the most effective sentiment model into time series forecasting models to evaluate improvements in cryptocurrency price prediction accuracy.

To achieve **Objective 1**, we define the following specific tasks:

- Derive a method to calculate price impact weights from market data (price and volume) following news announcements.
- Fine-tune existing language models for sentiment analysis using a custom dataset of cryptocurrency related headlines paired with custom derived impact scores.
- Develop baseline sentiment analysis models using machine learning techniques and NLP approaches such as `TF-IDF`.
- Evaluate and compare all models using error metrics and correlation with actual market movements.
- Select the most effective sentiment model for integration into forecasting applications.

To achieve **Objective 2**, we define the following specific tasks:

- Prepare datasets combining sentiment scores from the selected model with traditional technical indicators (hashrate, transaction count, volume, general market health indicators).
- Develop time series forecasting models that incorporate both sentiment scores and technical indicators as input features.
- Train and validate forecasting models using historical cryptocurrency price data.
- Evaluate forecasting accuracy through backtesting with real market data and compare results against baseline models without sentiment features.

1 Literature analysis

The intersection of market forecasting and sentiment analysis has advanced significantly with improved data and computing power. This section reviews research relevant to our task, focusing on sentiment and prediction possibilities.

First, we outline the factors that drive cryptocurrency prices and volatility. We then examine recent time-series forecasting models, such as xLSTM and iTransformer, and how they handle multivariate data. We also summarize common workflows that integrate sentiment into end-to-end prediction pipelines.

Next, we review finance-specific sentiment models, including FinBERT, CryptoBERT, and FinLlama. We discuss their limitations and highlight regression-based approaches that treat sentiment as a continuous signal rather than simple categories.

Also, we cover backtesting strategies like volatility scaling and momentum which are required for backtesting finally we conclude by connecting required research areas. This should highlight the gaps in existing research and show motivation for the need of such work.

1.1 Factors affecting cryptocurrency price and volatility

Internal Factors		
Supply & Demand		
Transaction Cost (PoW / PoS) Reward System Mining Difficulty (Hash Rate) Coins Circulation Forks (Rule Changes)		
External Factors		
Cryptomarket	**Macro-financial**	**Political**
Attractiveness (Popularity) Market Trend Speculations	Stock Markets Exchange Rate Gold Price Interest Rate Others	Legalization (Adaptation) Restrictions Others

Table 1. Internal and External Factors Affecting Cryptocurrency

The study by Y. Sovbetov [25] utilizes the ARDL (Autoregressive Distributed Lag) technique to analyze key determinants of cryptocurrency prices. It finds that cryptomarket-related factors such as - market beta, trading volume, and volatility play a crucial role in price determination for five major cryptocurrencies in both short and long-run. Also, market volatility is a statistically significant factor affecting cryptocurrency prices in both short and long-term.

López-Cabarcos et. al. [27] investigation results suggested that Bitcoin volatility is more unstable in speculative periods. In stable periods, S&P 500 returns, VIX returns, and sentiment influence Bitcoin volatility. To complete their analysis S&P500 Index, VIX Index, Bitcoin prices, and the sentiment about S&P500 Index were used. And sentiment data was extracted from Stocktwits messages using the Standford CoreNLP. Later, GARCH and EGARCH models were proposed to test the influence of investor sentiment, S&P 500 Index and VIX, respectively on Bitcoin volatility.

Both of them showed positive results. The full set of factors identified by the study can be seen in the Table 1.

Focusing on Bitcoin's 24/7 trading nature, Shen et. al [24] finds that returns in the first half-hour of trading can predict returns in the last half-hour. The study shows that the first trading sessions with the highest volume or volatility are associated with the greatest predictability for intraday time series momentum.

One of the most popular indicators showing current market health is a propietary CNN fear and greed index [7] which derived from parameters:

- **Stock Price Momentum:** S&P 500 relative to its 125-day moving average.
- **Stock Price Strength:** Number of stocks making 52-week highs vs. lows on the NYSE.
- **Stock Price Breadth:** Trading volume in advancing vs. declining stocks.
- **Put and Call Options:** Balance between put and call option activity.
- **Junk Bond Demand:** Yield spread between investment-grade and junk bonds.
- **Market Volatility:** CBOE VIX compared to its 50-day moving average.
- **Safe Haven Demand:** Return difference between stocks and treasury bonds.

Similiarly, some companies and products like "CoinGecko", "CoinMarketCap", "Alternative.me", "Binance" proposed their own proprietary "Fear and Greed" indexes for the public. Most of them are not disclosing how they are calculating such metric.

1.2 Time series forecasting methods

Most classical approach to the timeseries is LSTM (Long Short-Term Memory) networks [12] which are a type of recurrent neural network (RNN) designed to handle sequences and capture long-term patterns. LSTMs solve the disappearing gradient problem of standard RNNs by using memory cells and gates (input, forget, output) that control how information is stored and updated. They are widely used in time series forecasting, natural language processing, and speech recognition.

A recent extension is xLSTM (Extended Long Short-Term Memory) [5], which improves standard LSTMs in two main ways. First, it replaces sigmoid gates with exponential gates to allow better memory updates and gradient flow. Second, it adds two memory types: sLSTM with enhanced scalar memory and mLSTM with matrix memory states ($C \in \mathbb{R}^{d \times d}$). These changes increase memory capacity, allow full parallelization during training, and keep computation linear in sequence length, while achieving performance close to Transformer models.

Also, recently, the iTransformer [19] has been introduced. iTransformer is an inverted Transformer design for multivariate time series forecasting. Instead of embedding entire time steps, it represents each variate as a token and applies attention across variates while modeling temporal dynamics with feed-forward layers. This inversion enables the model to capture cross-variate dependencies more effectively and handle long lookback windows. Across multiple benchmark datasets, iTransformer achieves one of the best forecasting accuracy, ranking among the best approaches in recent literature.

1.3 Time series prediction workflows using sentiment analysis

As we are planning to use sentiment analysis as a parameter for borader time series forecasting in future, we started off as looking how sentiment analysis are used in wider prediction systems.

Khan et al. study explored the use of machine learning classifiers to predict stock market trends by analyzing financial news and social media data. [14] The authors collected data from Twitter, Business Insider, and Yahoo Finance, integrating sentiment analysis using Stanford NLP to classify news and tweets on a scale from highly negative to highly positive. They extracted key stock trend features, such as price fluctuations and future trends over a 10-day period and prepared volatility and influence system as shown in Figure 1. Later, study examined different prediction models based on social media, financial news, and a combination of both, aiming to identify stocks that are difficult to predict and markets more influenced by external sentiment. During this study, various machine learning classifiers, including Gradient Boosting Machine (GBM), Extra Trees (ET), Classification and Regression Trees (CART), and Random Forest (RF), were tested. The results showed that when using only social media data, GBM achieved the highest accuracy at 76.50%, however study concluded that the RF classifier provides the most consistent results for stock market prediction due to the following reasons:

- RF achieves the highest prediction accuracy of 80.53% when using social media data.
- It also demonstrates a strong prediction accuracy of 75.16% when utilizing financial news.
- The accuracy of RF improves after applying feature selection techniques, with increases of 3.31% using SelectKBest and 1.98% using PCA.
- The classifier's accuracy further improves by 1.32% after reducing spam data.
- RF outperforms other classifiers in terms of classification accuracy, precision, recall, and F-measure.

While Khan et al. demonstrate the value of sentiment features, their approach has notable weakness - the scores are derived from Stanford NLP, a general-purpose tool not specifically trained for financial contexts or cryptocurrency markets with binary sentiment classification (positive/negative).

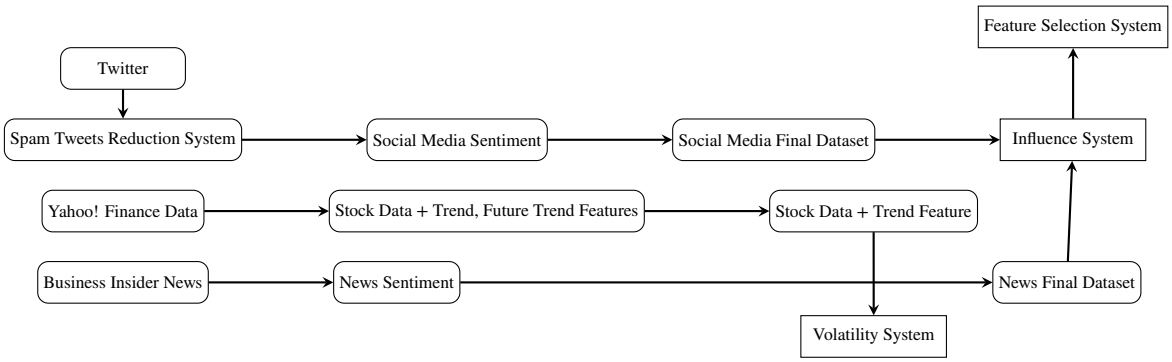


Figure 1. Approximate workflow proposed by Khan et al.

Study by Li et al. investigated the impact of news sentiment on forecasting oil futures returns and volatility using a deep learning approach. [17] Authors utilize two datasets separately: Brent crude oil futures price data and financial news headlines. The news data undergoes preprocessing,

including stop-word removal and vectorization using the Term Frequency-Inverse Document Frequency (TF-IDF) method. To enhance prediction accuracy, the study employs Variational Mode Decomposition (VMD) to extract intrinsic patterns from daily returns and 7-day volatility time series. The cumulative sentiment score from news headlines is integrated with decomposed intrinsic modes to create a hybrid input dataset. A BiLSTM neural network model, comprising multiple layers including forward and backward hidden layers, is then trained to predict daily logarithmic returns and 7-day volatility. Additionally, the study introduces the Senti-GARCH (p, q, m) model, which incorporates news sentiment into variance equations to assess its impact on oil futures volatility. Full process visually can be consulted with Figure 2. The results indicate that news sentiment significantly affects volatility, with positive news reducing volatility and negative news increasing it. These findings highlight the asymmetric effects of news sentiment on market fluctuations. The proposed VMD-BiLSTM model demonstrates superior forecasting performance compared to other models, suggesting that integrating sentiment analysis with deep learning can enhance financial market predictions.

While Li et al. effectively integrate sentiment with time series decomposition, their sentiment extraction relies on TF-IDF vectorization - a bag-of-words approach that captures word frequencies but not semantic context or domain-specific nuances. The sentiment scores are not fine-tuned or validated against actual price impact, and the study does not explore whether using transformer-based models for sentiment extraction could improve forecasting accuracy.

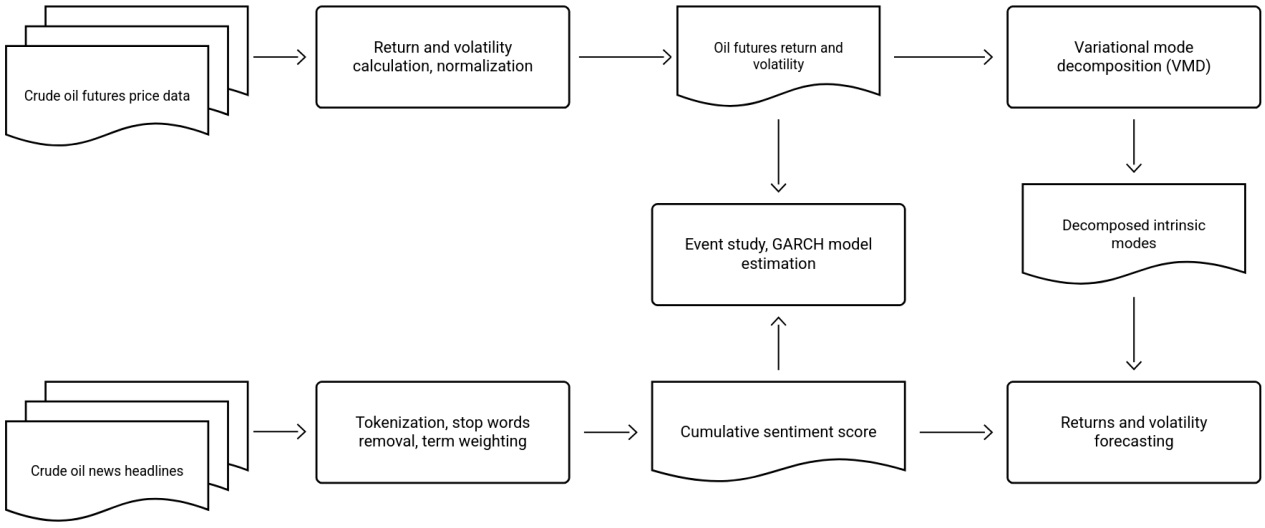


Figure 2. Process in research by Li et. al

1.4 Sentiment analysis for market analysis

FinBERT [3] was fine-tuned by further pre-training a BERT model on a financial text collection (TRC2-financial) using masked language modeling. This adaptation aimed to capture domain-specific terminology and contextual nuances. For sentiment classification, a dense layer was added on top of BERT's token representations, and the model was fine-tuned using the Financial PhraseBank and FiQA datasets. Techniques such as slanted triangular learning rates, discriminative fine-tuning, and gradual unfreezing were applied to prevent catastrophic forgetting and improve training efficiency.

CryptoBERT, for instance, was trained on StockTwits data using the authors used StockTwits data (labeled as bullish or bearish) for the masked language modeling (MLM) task, based on the RoBERTa (Robustly Optimized BERT Pretraining Approach) architecture. Their test results showed a slight improvement (up to 5% better accuracy) over standard fine-tuned sentiment analysis models such as FinBERT, BERTweet, or vanilla non-fine-tuned BERT for financial information classification tasks. [16]

The FinLlama [15] model was fine-tuned on supervised financial sentiment analysis data to better handle the complexities of financial terminology and context. It is equipped with a neural network-based decision mechanism. FinLlama was trained using labeled data with three sentiment classes: positive, neutral, and negative. In this study, the authors also proposed constructing daily stock portfolios based on sentiment-driven rankings. Companies lacking sentiment data for a particular day were excluded from the rankings. This method yielded the best performance in their experiments, however, it seems that the authors did not address the potential issue of "learned future" bias - where the model might inadvertently learn future outcomes.

From this analysis, it is evident that most works successfully adapt language models for finance-related tasks. However, sentiment analysis is predominantly treated as a classification problem, rather than as a regression task that could measure the strength of sentiment. While these models perform well in sentiment classification, they often ignore the gradual nature and intensity of sentiment which could be more appropriately captured by regression modeling - especially in the context of continuous market signals. Also, *these models rely on human-annotated sentiment labels that may not correlate with actual market impact*. A headline labeled as "positive" by human annotators does not necessarily lead to price increases, yet this disconnect is rarely addressed in model training or evaluation

1.5 Sentiment in regression-based models

While sentiment analysis is predominantly applied in classification tasks (e.g., labeling text as positive or negative), a few studies have explored its use in regression settings, treating sentiment as a continuous variable. These efforts aim to capture the intensity or nuance of sentiment rather than categorizing it discretely.

Khalil and Pipa [13] proposed a financial forecasting approach that integrates deep learning with natural language processing. They introduced an *Hourly Sentiment Index*, derived by classifying financial news headlines into positive, negative, or neutral categories using a Naive Bayes classifier. The resulting index, ranging from -1 to +1, quantifies market sentiment on an hourly basis. This continuous sentiment measure was combined with historical stock prices and used as input to a Long Short-Term Memory (LSTM) model to predict short-term price movements. Their results showed that sentiment significantly impacts price direction, especially within a 3-4 hour lag, demonstrating the potential of real-time sentiment analytics in financial modeling.

Similarly, Lin et al. [18] presented a regression-based sentiment analysis framework by fine-tuning LLaMA 3 using Low-Rank Adaptation (LoRA). Starting with a RoBERTa model enhanced with additive attention and an adaptive Huber loss, they generated real-valued sentiment scores and enriched the SemEval 2017 and 2018 datasets - benchmarks that include sentiment annotations on a continuous scale from -1 to 1. They applied several data augmentation strategies (e.g., prompt engineering, synonym replacement, AEDA) to improve generalization. The fine-tuned LLaMA 3 model achieved superior performance in sentiment intensity regression, emphasizing the importance of custom data curation and tailored loss functions when adapting language models to re-

gression tasks. We incorporate a variant of their proposed loss *the Adaptive Weighted Huber Loss* (which can be seen in Figure 3) in our own experiments in section 4.2.2 and section 4.2.1. This loss function dynamically balances the benefits of Mean Squared Error and Mean Absolute Error, while adaptively weighting each sample based on prediction difficulty or noise. Its robustness to outliers and ability to focus learning on informative samples make it particularly well-suited for financial sentiment regression. By reusing this technique, we aim to evaluate its effectiveness in conjunction with transformer-based architectures in a domain-specific regression context.

$$\mathcal{L}_{\text{ScaledHuber}}(y, \hat{y}) = \begin{cases} \frac{1}{2}\epsilon^2 & \text{if } |\epsilon| \leq \delta' \\ \delta' \cdot (|\epsilon| - \frac{1}{2}\delta') & \text{otherwise} \end{cases}$$

Figure 3. Adaptive Weighted Huber Loss

These studies highlight early efforts to model sentiment as a continuous signal rather than a discrete label. However, several key limitations persist:

- Khalil and Pipa's approach still relies on categorical classification (positive/negative/neutral) before aggregation, meaning the underlying sentiment signal remains discretized rather than truly continuous.
- The sentiment scores used are typically derived from external lexicons or pretrained classifiers trained on human-annotated data, which may not reflect actual market dynamics or price impact.
- Lin et al., while employing regression, use SemEval datasets where sentiment labels represent perceived emotional intensity rather thanb observed market impact.

The fundamental gap remains - no existing work derives sentiment supervision signals directly from price behavior (e.g., percentage price changes following news events), which would better align model training with the actual prediction target - a challenge our work aims to address.

1.6 Strategies using time series methods

Besides simple trading strategies using time series methods like longing on positive prediction and shorting on negative, there are more complex strategies proposed in literature.

To avoid impact of sudden market changes, most of algorithmic traders utilizes volatility scaling to adjust their position sizes based on recent market volatility. This helps maintain consistent risk levels despite changing market conditions [22]. The concept of volatility is quite simple:

$$w_t = \frac{1}{\hat{\sigma}_t^2} \tag{1.1}$$

Where w_t is the weight (or position size) at time t, and $\hat{\sigma}_t^2$ is the estimated conditional variance (volatility) of returns at time t. This means that when volatility is high (i.e., $\hat{\sigma}_t^2$ is large), the weight w_t decreases, leading to smaller position sizes. Volatility is usually calculated using by averaging past squared returns r over the last N periods:

$$\hat{\sigma}_t^2 = \frac{1}{N} \sum_{i=1}^{N} r_{t-i}^2 \tag{1.2}$$

One of time series trading strategies is time series momentum which was researched by Moskowitz et al. [23] and they found that time series momentum is a pervasive phenomenon that exists across all asset classes and geographies. They tested this strategy on 58 different futures markets, including commodities, currencies, stock indices, and bonds, over a 40-year period from 1960 to 2009. The results showed that a simple time series momentum strategy, which involves buying assets that have performed well in the past and selling those that have performed poorly, generated significant excess returns. The strategy usually outperformed traditional buy-and-hold strategies. Usually this strategy is used without conjunction with predictive models.

1.7 Comparative summary of related literature

To summarize the various related literature, we present a comparison tables that highlights of the:

- Sentiment analysis approaches and their key characteristics
- Time Series forecasting possibilities with sentiment integration
- Forecasting methods and trading strategies

Most of the reviewed studies focus on sentiment classification using predefined datasets, with binary or multiclass outputs (e.g., positive, negative, neutral). Models such as FinBERT, CryptoBERT, and FinLlama demonstrate strong performance in classifying financial sentiment but do not directly link their outputs to financial prediction tasks. Table 2 summarizes the key sentiment analysis models and their performance characteristics.

Study	**Model/Method**	**Data Source**	**Sentiment Approach**	**Key Performance**
FinBERT [3]	Fine-tuned BERT	Financial PhraseBank, FiQA, TRC2-financial	Classification (3-class: pos/neu/neg)	SOTA for financial sentiment classification
CryptoBERT [16]	Fine-tuned RoBERTa	StockTwits	Binary (bullish/bearish)	5% improvement over FinBERT baseline
FinLlama [15]	Fine-tuned LLaMA	Multiple financial datasets	Classification (3-class)	Best portfolio construction performance
Prottasha et al. [4]	BERT + CNN-BiLSTM	Bengali sentiment datasets	Binary classification	SOTA for Bengali sentiment analysis
Lin et al. [18]	LLaMA 3 + LoRA	SemEval 2017/2018 enriched	Regression (-1 to 1)	Superior intensity prediction with Huber loss

Table 2. Sentiment Analysis Models and Performance Comparison

Only a few studies attempt sentiment regression or use continuous sentiment values to model financial indicators like price returns or volatility. Works such as Li et al. [17] and Khalil and Pipa [13] represent this direction, employing models like LSTM or BiLSTM with sentiment as a predictive input. However, even these approaches rely on existing datasets and sentiment labels, without deriving custom weight scores based on market response or fine-tuning for task-specific forecasting. Table 3 illustrates how sentiment has been integrated into time series forecasting across different studies.

Study	Target Asset	Forecasting Model	Sentiment Integration	Key Results
Khan et al. [14]	Stock trends	RF, GBM, CART, ET	Stanford NLP classification	80.53% accuracy with RF on social data
Li et al. [17]	Oil futures	BiLSTM + VMD	TF-IDF sentiment scores	VMD-BiLSTM outperforms baselines; asymmetric volatility impact
Khalil & Pipa [13]	Stock prices	LSTM	Hourly sentiment index (-1 to 1)	News has 3-4 hour lag significance for price direction
López-Cabarcos et al. [27]	Bitcoin volatility	GARCH, EGARCH	StockTwits sentiment via CoreNLP	S&P 500, VIX, sentiment influence BTC volatility

Table 3. Time Series Forecasting with Sentiment Integration

Beyond sentiment analysis, recent advances in time series forecasting methods and trading strategies have introduced new architectures and approaches that could potentially enhance financial prediction models. Table 4 summarizes these methodological innovations, from architectural improvements like xLSTM and iTransformer to trading strategy developments such as volatility scaling and momentum-based approaches.

Study	Method/Innovation	Key Contribution	Performance/Impact
Beck et al. [5]	xLSTM architecture	Exponential gating, matrix memory states	Competitive with Transformers, linear complexity
Liu et al. [19]	iTransformer - multivariate time series	Inverted attention across variates	Best forecasting accuracy on benchmark datasets
Moskowitz et al. [23]	Time series momentum	Cross-asset momentum persistence	Significant excess returns across 58 futures markets
Moreira & Muir [22]	Volatility scaling	Dynamic position sizing based on volatility	Consistent risk levels, improved risk-adjusted returns
Shen et al. [24]	Intraday momentum	Half-hour return predictability	High volume/volatility sessions show greatest predictability

Table 4. Recent Advances in Time Series Methods and Trading Strategies

This literature review reveals several critical gaps that our work addresses:

1. **Lack of price-derived sentiment labels.** As shown in Table 2, existing sentiment models (FinBERT, CryptoBERT, FinLlama) rely on human-annotated datasets where sentiment is labeled based on perceived emotional tone rather than actual market impact. No work derives regression labels directly from observed price changes following news events.

   Our approach: We calculate sentiment labels from actual cryptocurrency market reactions following news events, measuring real price impact rather than relying on human-annotated emotional tone.

2. **Limited regression approaches.** While classification dominates the field, the few regression studies (Khalil & Pipa, Lin et al.) still use human-annotated sentiment intensity scores from general-domain datasets (SemEval), not market-impact-based labels.

   Our approach: We fine-tune transformer-based models (FinBERT, Qwen) specifically for sentiment regression rather than classification, using our price-derived labels as supervision signals.

3. **Missing comprehensive baseline comparison.** Most studies do not systematically compare fine-tuned models against traditional approaches or validate whether domain-specific fine-tuning provides measurable improvements.

Our approach: We conduct comprehensive evaluation against traditional NLP baselines (TF-IDF + ML) and pretrained models to rigorously validate the effectiveness of fine-tuning approaches.

4. **Missing end-to-end validation.** As Table 3 shows, studies integrate sentiment into forecasting but do not evaluate whether fine-tuned sentiment models outperform baseline approaches in actual trading scenarios through rigorous backtesting on cryptocurrency markets.

   Our approach: We integrate and validate sentiment models through backtesting on cryptocurrency price forecasting with modern time series architectures, demonstrating real-world trading performance.

2 Data Retrieval and preparation

2.1 Bitcoin historical price dataset

Initially, price and volume data were obtained from the Bybit public trading repository [6]. The choice of this dataset was straightforward, as Bybit is one of the few high-volume cryptocurrency exchanges that publicly provides time series data on trading volume and prices. Unlike other exchanges that tend to avoid releasing such data publicly, and alternative sources that lack high-resolution granularity (e.g., 1 minute intervals or lower), Bybit offers reliable and detailed access suitable for this type of analysis.

The data is in 1-minute intervals and includes the following columns: timestamp, open, close, high, low, total volume, buy volume, sell volume, and trade count. The sample can be seen in Table 5 .

timestamp	open	high	low	close	vol_buy	vol_sell	activity
1743334260	83149.78	83184.65	83132.43	83184.65	236173.30	304700.0	1811
1743334320	83184.65	83184.66	83141.85	83141.85	60844.87	190254.5	1300
1743334380	83141.85	83141.86	83100.00	83104.94	164623.56	290815.1	1411
1743334440	83106.00	83110.76	83038.02	83055.34	231423.30	1341206.0	4368
1743334500	83055.34	83055.34	83055.33	83055.33	215.94	7234.119	10

Table 5. Sample Bitcoin price and volume data

2.2 News dataset

Most popular newsfeed datasets, such as Bloomberg or Reuters, are proprietary and come at a high cost. Other options, such as open datasets, often contain noisy, unfiltered, and unrelated information, especially when it comes to social media sources like Twitter or not properly labeled news information. While such data may be useful for obtaining the overall sentiment surrounding of a given asset, our focus in this work is on extracting specific keywords and word combinations that could have a direct impact on price movements. Due to those reasons - three options were considered for the news headlines dataset:

1. The *"Papers with Backtests"* daily news headlines dataset

2. The *Cryptopanic.com* headlines dataset

3. Telegram channels that aggregate news related to financial markets and cryptocurrencies

The *"Papers with Backtests"* dataset contained a significant amount of valuable information that could potentially impact the market. However, it was discarded because much of the content was unrelated to cryptocurrencies, focusing instead on stock markets and other sectors, making it difficult to effectively filter out irrelevant news. Also, there was missing important macroeconomic news in this dataset.

The *Cryptopanic* headlines dataset is cryptocurrency-focused and contains valuable information about the crypto space. However, it lacks market-wide coverage, particularly missing important macroeconomic and legislative news that significantly affect the markets. Attempts to improve relevance by filtering based on headline tags still left the dataset too noisy and poorly correlated with market movements.

The third option, *Telegram news aggregator channels*, provided more market-relevant content like important macroeconomic events, cryptocurrency news like legislation, large transfers and aquisitions, political responses and many more. While it also included unrelated data, such as price alerts (e.g., Bitcoin or Ethereum hitting certain price levels) and advertisements, this dataset was comparatively more focused and useful for market-related news than the previous two.

For this dataset infromation has been obtained from 3 channels - "Watcher Guru", "Crypto News", "unfolded.". Before using this data there text filtering has beed done by removing usual keywords like - "JUST IN:" or "BREAKING:", removing emojis and removing data rows with unrelated information like price announcments and advertisements whenever posible. Also, for each post we needed to remove the Chinese, Japanese, and Korean letters; crypto wallet addresses; URLs, ticker tags ($); hashtags (#); usernames (@).

The small sample of filtered dataset can be seen in Table 6.

unixtime	text	from
1648692637	Two addresses containing 11,325 Bitcoin possibly ...	Watcher Guru
1648713317	Washington has passed a bill aimed at increasing ...	Watcher Guru
1648717971	Terra is now the third-largest disclosed Bitcoin ...	Watcher Guru
1648723160	The United States' IRS tax form is inquiring about ...	Watcher Guru
1648726347	El Salvador has announced the World Blockchain ...	Watcher Guru

Table 6. Cryptocurrency headlines dataset sample

2.3 Bitcoin blockchain data

To further enhance the dataset, Bitcoin blockchain data was also incorporated. This data includes metrics such as hash rate, transaction count, and other on-chain indicators that can provide additional context for market movements. While the blockchain data itself is publicly available, it requires significant processing to extract meaningful features. For this purpose, we utilized publicly available Google Cloud BigQuery datasets [11] that offer pre-processed blockchain data, making it easier to integrate with our existing datasets. We employed a straightforward SQL-like query to retrieve 10 years of hourly Bitcoin blockchain data, including transaction count, blocks mined, and estimated hash rate. A sample of the retrieved data is shown in Table 7.

Hour (UTC)	Transaction Count	Blocks Mined	Estimate hashrate
2015-10-01 00:00:00	4,115	6	9256395
2015-10-01 01:00:00	3,889	2	11243369
2015-10-01 02:00:00	5,112	4	8130558
2015-10-01 03:00:00	7,549	6	7844689

Table 7. Bitcoin Transaction Count by Hour

These specific metrics were selected based on the factors discussed in Section 1.1. Transaction count serves as a measure of network activity and adoption, while hash rate functions as an indicator of network security and miner confidence. By incorporating these blockchain metrics into our dataset, we aim to capture additional dimensions of market behavior that may influence price movements.

2.4 General market data

To complement the cryptocurrency-specific datasets, we also incorporated general market data that includes key economic indicators. This information helps provide a broader context for understanding the factors influencing cryptocurrency prices and market trends.

The general market data was sourced from Tiingo[1], a financial data platform that offers comprehensive datasets on various asset classes, with most of its data originating from IEX Exchange. We were able to retrieve 5 years of hourly data for several important market indicators, including:

- Gold price - often seen as a safe-haven asset during times of economic uncertainty, providing insight into risk-off sentiment that may correlate with Bitcoin demand
- S&P 500 index - represents broader equity market performance which often influences cryptocurrency price.
- VXX (Volatility Index) - measures market fear and uncertainty, as periods of high volatility typically affect Bitcoin's correlation with traditional assets

Data has been retrieved using Tiingo's RESTful API, which allows for easy access to historical price and volume data. A sample of the S&P 500 price data is shown in Table 8.

Timestamp	Open	High	Low	Close	Volume
1569852000	272.33	272.48	272.07	272.44	2,001
1569855600	272.44	272.81	272.41	272.77	4,361
1569859200	272.81	273.15	272.81	273.07	4,751
1569862800	273.07	273.13	273.03	273.08	500

Table 8. S&P 500 Price Data by Timestamp

Problematic aspect of this data that it utilizes traditional market hours and does not provide data for weekends or holidays. To address this, we implemented a forward-fill strategy to populate missing values during non-trading hours, ensuring a continuous time series that aligns with the cryptocurrency market data.

[1] https://www.tiingo.com/ (Accessed: 2025-10-26)

3 Headline-to-price impact parameter

One of the targets of this research is to find an effective score for weighting the impact of headlines on price movements after connecting the headline dataset to corresponding price data.

The complete workflow for deriving and validating headline impact parameters follows three main stages, as illustrated in Figure 4:

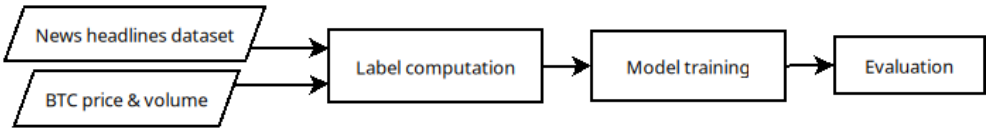


Figure 4. Headline-to-price impact parameter derivation workflow

The methodology for identifying the optimal weighting score was as follows:

1. Deriving eight different possible impact weights by linking price data to corresponding headlines.
2. Testing these weights by training a TF-IDF-based Support Vector Machine (SVM) model.
3. Performing cross-validation and comparing the results using metrics such as Mean Squared Error (MSE), Relative Squared Error (RSE), Symmetric Mean Absolute Percentage Error (SMAPE), and correlation on the test dataset.

3.1 Selecting price impact parameters

In this section - our goal is to build meaningful price impact weights that capture various dimensions of market behavior - volatility, momentum, and trading activity - in order to evaluate how news events affect market prices. Before proceeding, we derived several additional features from the price data. These derived parameters are required for estimating price impact in later steps and as discussed in Section 1.1.

1. **30-day Volatility:** First, compute the minute log return:

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right),$$

where P_t is the close price at time t. Log returns add over time, treat up/down moves symmetrically, and fit volatility-based models.

Then, compute the 30-day rolling volatility using a window of N minutes:

$$\sigma_{30d}(t) = \sqrt{N} \times \mathrm{std}\big(r_{t-N+1}, \ldots, r_t\big),$$

where N is the number of minutes in 30 days (a constant). The $\sqrt{N}$ factor rescales per-minute volatility to a 30-day horizon, making signals comparable across time and sampling frequencies [24].

2. **Total Volume:** This is a direct sum of buy and sell volumes:

$$V_{\text{total},i} = V_{\text{buy},i} + V_{\text{sell},i}$$

Trading volume is widely accepted as a liquidity and sentiment indicator, influencing price elasticity to new information [25].

We then derive eight variations (weights) from price and volume data. These variations aim to reflect different aspects of market response to news events:

- P_t: close price at time t
- $V_t^{buy}, V_t^{sell}, V_t^{total}$: buy, sell, and total trade volumes at time t
- $\sigma_{30d}(t)$: 30-day rolling volatility at time t
- EMA_t: exponential moving average with a 60-minute span
- σ_{max}: maximum observed 30-day volatility in the dataset

1. Relative price momentum (ΔP_t^{smooth})

$$\Delta P_t^{smooth} = \tanh\left(\Delta P_t^{4h}\right) \tag{3.1}$$

$$\Delta P_t^{4h} = \frac{P_{t+4h} - P_{t-5min}}{P_t} \tag{3.2}$$

This weight captures 4-hour price momentum adjusted with a hyperbolic tangent to stabilize outliers. The 5-minute shift accounts for potential lags in news lag via Telegram news aggregators. Khalil and Pipa [13] argue that medium term momentum (3 - 5 hours) best captures price response to news in markets. The hyperbolic tangent (tanh) function is applied to bound outputs to $[-1, 1]$ and reduce the influence of extreme outliers. As in our results we usually get few very high values and few low values - tanh preserves the sign of the change while compressing large absolute values, preventing extreme price movements from dominating the training signal.

2. Volatility scaled momentum

$$\Delta P_t^{smoothVolatility} = \tanh\left(\Delta P_t^{4h} \cdot (\sigma_{max} - \sigma_{30d}(t))\right) \tag{3.3}$$

Here, momentum is weighted by decreasing volatility. This scaling addresses the non-stationarity of crypto assets where earlier time periods (for example - until 2017) had higher absolute volatility. Sovbetov [25] emphasizes volatility normalization for time period comparability in crypto market behavior. Same as previously described, the hyperbolic tangent (tanh) function is applied to bound outputs to $[-1, 1]$ and reduce the influence of extreme outliers.

3. EMA smoothed momentum

$$\Delta P_t^{EMA} = \text{EMA}\left(\Delta P_t^{smooth},\ 60\right) \tag{3.4}$$

Applies exponential moving average (EMA) with a 1 hour span to smooth out high-frequency noise in price momentum. This reduces the impact of very short-term market microstructure effects while preserving the overall trend signal. EMA gives higher weight to more recent observations, making it responsive to developing trends while filtering random fluctuations [8].

4. EMA of volatility scaled momentum

$$\Delta P_t^{EMAVolatility} = \text{EMA}\left(\Delta P_t^{smoothVolatility},\ 60\right) \tag{3.5}$$

Combines both volatility normalization and temporal smoothing. This provides more stable signals by addressing both cross-period comparability (via volatility scaling) and intra-period noise (via EMA), resulting in labels that capture sustained directional moves adjusted for market regime.

5. Volume surge momentum

$$V_t^{surge} = \tanh\left(\frac{\text{EMA}_t^{buy}}{\overline{V_t^{buy,7d}}} - \frac{\text{EMA}_t^{sell}}{\overline{V_t^{sell,7d}}}\right) \tag{3.6}$$

Focuses purely on trading volume imbalance rather than price. The difference between normalized buy and sell volumes captures directional trading pressure. A surge in buy volume relative to recent baseline suggests strong bullish sentiment, while sell volume dominance indicates bearish pressure. Shen et al. [24] show that such volume surges often precede price movements, making them a leading indicator of news impact. The 7-day normalization removes the effect of gradually changing market liquidity over time.

6. Volume surge with volatility adjustment

$$V_t^{adjustedVolatility} = \tanh\left(\sqrt{V_t^{surge} \cdot \left(\sigma_{max} - \sigma_{30d}(t)\right)}\right) \tag{3.7}$$

Combines volume imbalance with volatility scaling. This down-weights volume spikes that occur during already-volatile periods (e.g., market crashes), where high volume is normal. Instead, it emphasizes volume surges during calm periods, which more likely signal news-driven sentiment shifts rather than general market panic.

7. Volume surge with price and volatility adjustment (composite weight)

$$V_t^{adjustedPrice} = \tanh\left(\sqrt{V_t^{surge} \cdot \left(\sigma_{max} - \sigma_{30d}(t)\right) \cdot \Delta P_t^{4h}}\right) \tag{3.8}$$

This composite weight integrates all three dimensions: price momentum, volume imbalance, and volatility regime. The hypothesis is that truly impactful news creates a *coordinated* market response: directional price movement (ΔP_t^{4h}), unusual trading volume (V_t^{surge}), and occurs in a stable enough environment to be news-driven rather than noise ($\sigma_{max} - \sigma_{30d}(t)$). By multiplying these factors, we isolate headlines that trigger comprehensive market reactions across multiple indicators simultaneously.

3.2 Experimental evaluation of the best parameter

Authors used calculated weights on simple Support Vector Machine TF-IDF model which extracts unigrams to 7-grams that appear in at least one document but in no more than 10% of documents. SVM model used a 6th-degree polynomial kernel with a regularization strength of ‘C=0.25‘, an epsilon margin of 0.001 for tolerance around predictions, and automatically scaled gamma for controlling the kernels influence. All those parameters has been found using optimization methods

Weight type	RMSE	MAE	SMAPE	4h price correlation
V_t^{surge}	0.091947	0.056364	85.232580	0.087853
$V_t^{adjustedVolatility}$	0.140443	0.106330	**48.391515**	0.102572
$V_t^{adjustedPrice}$	**0.080502**	**0.050406**	75.093273	**0.148571**
ΔP_t^{smooth}	0.116827	0.074555	123.814772	0.125477
$\Delta P_t^{smoothVolatility}$	0.134681	0.087192	122.898807	0.125240
ΔP_t^{EMA}	0.141806	0.090266	120.972093	0.131476
$\Delta P_t^{EMAVolatility}$	0.147940	0.095203	120.264589	0.131098

Table 9. Weight comparison result with SVM model

by trying different parameters. Values has been calculated using k-fold cross validation method spliting 1:7 test and training data.

From the results in Table 9, we can see that the $V_t^{adjustedPrice}$ weight type achieved the highest correlation with 4-hour price changes, reaching 0.1486. It outperforms other weighting approaches, indicating the best alignment between word vectors and price data. It also records the lowest RMSE and MAE values, confirming stronger consistency compared to other methods. Interestingly, the simpler ΔP_t^{EMA} and $\Delta P_t^{EMAVolatility}$ weight formulas also show relatively good correlation with price and could be useful alternatives when a volume dataset is not available.

4 Selection of sentiment model

In this section, we focus on building and comparing models designed to assign our generated price impact prameter to news headlines, aiming to identify those that most significantly impact Bitcoin's price. We explore a variety of machine learning techniques, including traditional approaches and fine-tuning of large language models. After evaluating different methods, we will summarize and compare their performance in the conclusion.

In all final evaluations to prevent time leakage and ensure realistic evaluation, all models use a strict training/evaluation split. The dataset is ordered chronologically by headline timestamp, and we use the last year (from 2024-04 till 2025-04) as the test set, with all earlier data (2021-08 till 2024-04) for training. This simulates a scenario where the model is trained on historical data and evaluated on future, unseen data. *No random shuffling was performed* - all training examples occur strictly before all test examples in time. Thats gets us to the train-to-test ratio is approximately 7:1, resulting in roughly 17,500 training examples and about 2,500 test examples. For comparison with the time-series model training and backtesting test splits, we include a short summary table in Appendix C.

Also, for convinience all hyperparameter tuning results described in appendix A.

4.1 Evaluating machine learning methods using TF-IDF representations

Here, we aim to identify an effective model for predicting sentiment scores based on headlines, utilizing adapted term weights described in Section 3. Most of experiments were conducted using the high-performance computing infrastructure available at the Vilnius University Faculty of Mathematics and Informatics, using virtual machine equipped with 8 CPUs, 28GB RAM and Python 3.8 Jupyter notebook.

4.1.1 Support vector machine approach

We constructed a sentiment prediction model by applying a TF-IDF vectorizer, including the adapted headline weights introduced in Section 3, followed by a Support Vector Regression (SVR) model using the `scikit-learn` Python library.

To optimize model performance, we conducted an iterative hyperparameter tuning process using `GridSearchCV` with five-fold cross-validation on a reduced training set. Initially, we began with a broad search space, testing coarse parameter ranges. Based on the most promising results, we progressively refined the search space by "zooming in" around higher performing configurations, in this case C, ϵ, Max DF and SVR Degree values. In total, this tuning process involved approximately 7 stages of narrowing down parameters, each time focusing on increasingly finer intervals to best performance.

The final set of tested parameters and the best-performing configuration are summarized in Figure 5.

Parameter	Values Tested	Best Parameters
N-grams range	(1,1), (1,2), (2,2), (2,3), (1,4)	(1,1)
Max DF	0.001, 0.01, 0.1, 0.15, 0.2, 0.3, 0.5, 1, 2	0.15
Min DF	1, 2	1
Sublinear TF	False, True	False
SVR C	0.01, 0.025, 0.05, 0.075, 0.08, 0.09, 0.1, 0.2, 1	0.075
SVR ϵ	0.01, 0.025, 0.05, 0.075, 0.08, 0.09, 0.1, 0.2, 0.5	0.075
SVR Kernel	linear, poly	poly
SVR Gamma	scale	scale
SVR Degree	3, 4, 5, 6, 7, 8, 9	7

Figure 5. Hyperparameter grid search results for TF-IDF-based SVR models

For model evaluation, we split the dataset using a 7:1 train-to-test ratio (1 last year is a test data). After training the SVR model with the best-found parameters, we achieved the following results on the test dataset:

- Mean Squared Error (MSE): 0.0046
- Root Mean Squared Error (RMSE): 0.0678
- Mean Absolute Error (MAE): 0.0474
- 4-hour Price Correlation: 0.0031
- Pearson Correlation P value: 0.9412

4.1.2 Gradient boosting

Similarly to the approach described in Subsection 4.1.1, for this experiment we constructed a sentiment prediction model by reusing the same TF-IDF vectorizer with the adapted weights, followed

by a Gradient Boosting Regressor (GBR) model from the Python `scikit-learn` library. To optimize the model's performance, we performed a hyperparameter search using `GridSearchCV` with five-fold cross-validation on a reduced training set.

As previously - we started with higher values and progressively refined the search space by "zooming in" around higher performing configurations, in this case - GBR Learning Rate, GBR Max Depth, GBR Subsample and Max DF values. For N-grams range we took values (1,1), (1,2), (2,1), (2,1), (3,1), (3,2) and after finding best performing configurations, started testing with different N-grams range. In total, this tuning process involved 12 stages of narrowing down parameters, each time focusing on increasingly finer intervals to best performance.

The tested parameter combinations and their corresponding performance are summarized in Figure 6.

Parameter	Values Tested	Best Parameters
N-grams range	(1,1), (1,2), (1,3), (1,4), (1,5), (1,6), (1,7), (2,a), (2,2), (2,3), (2,4), (2,5), (2,7), (3,1), (3,2), (3,3), (3,4), (3,5)	(2,7)
Max DF	0.001, 0.01, 0.1, 0.2, 0.075, 0.125, 0.15, 0.175	0.125
Min DF	1, 2	1
Sublinear TF	True, False	False
GBR Learning Rate	0.05, 0.06, 0.07, 0.08, 0.09, 0.1	0.06
GBR Max Depth	2, 3, 4, 5, 7	4
GBR Subsample	0.8, 0.95, 1.0, 1.1	1.0

Figure 6. Different TF-IDF Support vector machine parameters

For model evaluation, as mentioned - we split the dataset using a 7:1 train-to-test ratio (1 last year is a test data). After training the GBR model with the best-found parameters, we achieved the following results on the test dataset:

- Mean Squared Error (MSE): 0.0003
- Root Mean Squared Error (RMSE): 0.0159
- Mean Absolute Error (MAE): 0.0111
- 4-hour Price Correlation: 0.0543
- Pearson Correlation P value: 0.0049

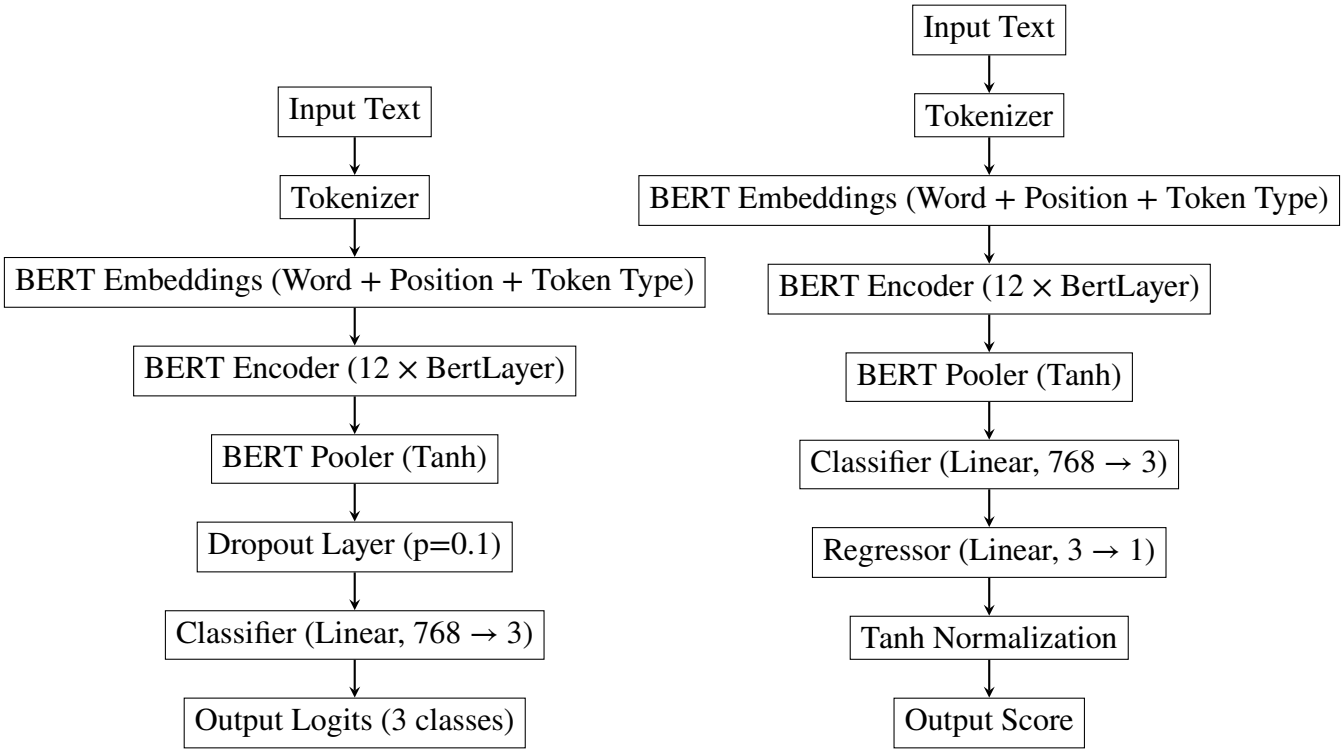
4.2 Evaluating plain and fine-tuned language models

In this subsection, we explore fine-tuning pre-trained language models using our headline dataset combined with the derived sentiment-price impact parameter. All experiments were conducted using Python 3.10 and the PyTorch framework. Training was primarily performed on a "Google Colab" environment equipped with an Nvidia T4 GPU. For the larger Qwen model, due to its higher number of parameters, an Nvidia A100 GPU was utilized.

4.2.1 Finetuning "FinBERT"

In this part of the study, we fine-tune the FinBERT language model using our custom-derived sentiment weights. Figure 7 compares the original FinBERT classification architecture with our modified version adapted for regression tasks.

In the modified model, the output layer was replaced with a regression head that compresses the three sentiment logits into a single continuous value. To stabilize the outputs within the $[-1, 1]$ range, a parabolic `tanh` activation function was applied after the regression head. This adjustment allows the model to predict not only sentiment polarity but also the sentiment strength more precisely, aligning with our custom weighting scheme.



(a) Modified FinBERT architecture for regression (b) Original FinBERT architecture for classification

Figure 7. Comparison of original FinBERT classifier and modified regression architecture

As our dataset is relatively small (approximately 15,000 headlines) and to prevent overfitting and catastrophic forgetting, we decided to freeze all layers of the model except for the last two encoder layers during fine-tuning. For headline tokenization, we used the default tokenizer provided by the BERT framework.

The decision to unlock only the last two encoder layers was based on experimental results, where different configurations were compared by training for a few epochs and evaluating performance. Table 10 summarizes the metrics for unlocking one, two, or three encoder layers.

Important note: These preliminary layer-selection experiments intentionally used a data split with approximately 20% overlap between training and test sets. This was done solely to accelerate the search for optimal architecture configuration by allowing faster adaptation signals. Once the optimal configuration (2 layers) was identified, all subsequent training and final evaluation used the strict temporal split described earlier with zero overlap.

Configuration	MSE	MAE	Corr (4h)
1 Layer	0.0030	0.0327	0.1930
2 Layers	**0.0028**	**0.0313**	**0.2738**
3 Layers	0.0144	0.1116	0.0528

Table 10. Evaluation of Fine-Tuned FinBERT with different numbers of unlocked encoder layers.

To enhance the performance of our regression model, we employed hyperparameter optimization with Optuna framework [2]. Our objective was to minimize the mean squared error (MSE) on a held-out validation set using a reduced training corpus of 5000 examples, sampled randomly from the full dataset to accelerate the tuning process.

The search space included several key parameters: learning rate (1×10^{-6} to 5×10^{-3}, log-uniform), weight decay (0.01 to 0.1), optimizer choice (`AdamW`, `SGD`, `RMSprop`) and momentum (for SGD optimizer) - from 0.1 to 0.9. Each trial reinitialized the model and fine-tuned it for a single epoch using an Adaptive Weighted Huber Loss, which provided robustness to outliers in the regression targets.

The training setup was as follows:

- Optimizer: AdamW
- Loss function: Adaptive Weighted Huber Loss
- Number of epochs: 25 (with early stopping triggered after stable loss)
- Learning rate: 7.35×10^{-5}
- Weight decay: 0.094
- Training/testing split ratio: about 7:1 (taking one year of headlines as testing data)

With the last two encoder layers unlocked and optimized hyperparameters, the fine-tuned FinBERT model achieved the following test set performance:

- **Mean Squared Error (MSE):** 0.0061
- **Mean Absolute Error (MAE):** 0.0282
- **Pearson 4-hour Price Correlation:** 0.0641
- **P-value:** 0.0009

4.2.2 Fine-tuning "Finance Qwen 1.5B"

Finance Qwen is a model developed by WiroAI [1], representing a financial-domain adaptation of the Qwen model family. It has been fine-tuned on over 500,000 high-quality finance-specific texts taken from more than 30 datasets. These datasets span a wide range of financial tasks, including question answering, reasoning, sentiment analysis, topic classification, multilingual named entity recognition (NER), and conversational AI [9]. Like many other sentiment models, its original training objective was supervised classification into three categories: positive, neutral, and negative.

As in Section 4.2.1, we adapted the model architecture for a regression task by replacing the classification head with a single-output regression layer. A hyperbolic tangent (`tanh`) activation

function was used to constrain the predicted sentiment scores to the range $[-1, 1]$. The adapted architecture is illustrated in Figure 8.

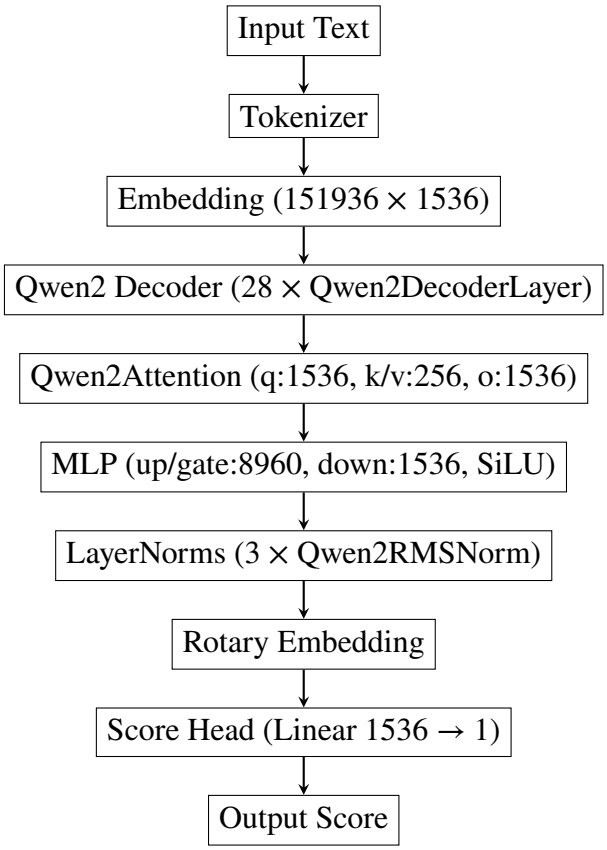


Figure 8. Adapted "Finance Qwen" architecture for regression

Due to the small size of our dataset (approximately 15,000 headlines), we aimed to minimize overfitting and catastrophic forgetting by freezing most of the model parameters. As in the FinBERT experiments, we selectively unfroze only the last few decoder layers for fine-tuning. Based on initial tests across a small number of epochs, we found the optimal configuration to be unlocking the last six decoder layers. Table 11 presents the performance metrics for different layer-unlock configurations.

Important note: As with FinBERT, these preliminary layer-selection experiments used approximately 20% data overlap to accelerate architecture search. All subsequent training and final evaluation used the strict temporal split with zero overlap.

Configuration	**MSE**	**MAE**	**Corr (4h)**
2 Layers	0.0062	0.0689	0.1568
6 Layers	**0.0022**	**0.0243**	**0.2745**
8 Layers	0.0024	0.0282	0.2081

Table 11. Evaluation of Fine-Tuned Qwen with different numbers of unlocked decoder layers.

To enhance the performance of our regression model, we employed Optuna [2] - a hyperparameter optimization framework based on Bayesian optimization. Our objective was to minimize the mean squared error (MSE) on a held-out validation set using a reduced training corpus of 5000 examples, sampled randomly from the full dataset to accelerate the tuning process.

The search space included several key parameters: learning rate (1×10^{-6} to 5×10^{-3}, log-uniform), weight decay (0.0 to 0.1), optimizer choice (`AdamW`, `SGD`, `RMSprop`) and momentum (for SGD optimizer) - from 0.1 to 0.9. Each trial reinitialized the model and fine-tuned it for a single epoch using an Adaptive Weighted Huber Loss, which provided robustness to outliers in the regression targets.

Over the course of 50 optimization trials, the study converged on a set of hyperparameters. The best-performing trial employed the SGD optimizer with a learning rate of 6×10^{-3} and weight decay of 0.012 achieving a validation MSE of 0.4. These results guided our choice of training configuration for full-scale fine-tuning.

The final training setup was as follows:

- Framework: PyTorch
- Optimizer: SGD
- Loss function: Adaptive Weighted Huber Loss
- Number of epochs: 15
- Learning rate: 6×10^{-3}
- Training/testing split ratio: about 7:1 (taking last year headlines as testing data)

With the optimal configuration (last 6 decoder layers unlocked), the fine-tuned Finance Qwen model achieved the following results on the test set (using strict temporal split with no data leakage):

- **Mean Squared Error (MSE):** 0.0024
- **Mean Absolute Error (MAE):** 0.0282
- **Pearson 4-hour Price Correlation:** 0.0512
- **P-value:** 0.0141

The correlation is slightly lower compared to FinBERT and higher p-value indicates weaker statistical significance than fine-tuned FinBERT.

4.3 Model comparison results

In this section we evaluated various regression based sentiment analysis models with our custom *headline-to-price impact parameter*. We conducted a series of regression experiments using both traditional machine learning models (with TF-IDF features) and fine-tuned financial language models. As mentioned before, our sentiment scores are in the normalized range [-1, 1], and the data was split using around 7:1 train-to-test ratio, where the last year serves as the test set. Evaluation metrics include Mean Squared Error (MSE), Mean Absolute Error (MAE), and Pearson correlation with the 4-hour future Bitcoin price movement.

Table 12. Model Performance Comparison

Model	MSE	MAE	Correlation	P-value
TF-IDF + SVR	0.0046	0.0474	0.0031	0.9412
TF-IDF + GBR	**0.0003**	**0.0159**	0.0543	0.0049
FinBERT (fine-tuned)	0.0061	0.0282	**0.0641**	**0.0009**
Finance Qwen (fine-tuned)	0.0024	0.0283	0.0512	0.0141

Among the models tested, the TF-IDF and Gradient Boosting Regressor (GBR) achieved the lowest error metrics (MSE: 0.0003, MAE: 0.0111), indicating precise regression performance. However, in terms of correlation with actual 4-hour price changes, fine-tuned FinBERT demonstrated stronger alignment (Pearson correlation = 0.0641, p = 0.0009), suggesting their better contextual understanding of financial text and/or beign more adaptive to fine-tuning with custom parameter. After consulting with the results we decided to proceed further with fine-tuned FinBERT model.

5 Timeseries predictions using multi-dimensional data

In past sections we discussed about predicting sentiment from headlines information. However, the ultimate goal is to use this information to improve timeseries predictions of asset prices. In this section we will discuss how we are combining different data sources and using them for multi-dimensional timeseries predictions. In later section 6 and 8 we will discuss how well our predictions would perform in real life trading scenarios by implementing backtesting procedure.

Figure 9 illustrates our multi-dimensional timeseries prediction pipeline. We combine multiple data sources (detailed in Section 5.1.1) into unified datasets for model training. Data spans from October 2019 to June 2025 (hourly frequency), split into:

- **Training set**: October 2019 to April 2024
- **Validation set 1**: June 2024 to January 2025 - uptrend market
- **Validation set 2**: January 2025 to June 2025 - sideways market

For comparison with the sentiment training and test splits, we include a short summary table in Appendix C.

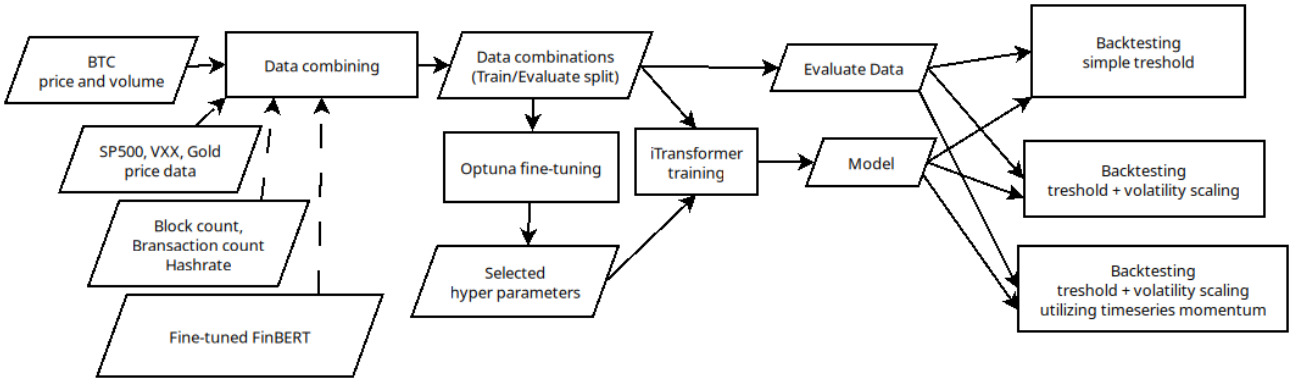


Figure 9. Process utilized for multi-dimensional timeseries predictions

We evaluate four distinct feature combinations through hyperparameter tuning (Section 5.2), train models with optimal hyperparameters, and validate performance through backtesting.

After training, we obtain models that predict the **next hour's price change** (not the absolute price) based on historical data. We then assess their real-world trading performance via backtesting (detailed in Sections 6 and 8).

5.1 Time series prediction setup

We formulate the forecasting problem as a supervised learning task where the model predicts the next hour's price change based on a fixed window of historical data. The experimental setup is defined by the following components:

- **Lookback window**: a historical window of $L = 168$ hours (7 days).
- **Target**: the price change for the next hour (ΔP_{t+1}).
- **Features**: we investigate four distinct feature sets, ranging from basic market data to comprehensive multi-dimensional inputs as discussed in Section 5.1.1.
- **Sentiment Aggregation**: to align asynchronous news headlines with hourly market data, we employ an exponential decay aggregation method as defined in Section 5.1.2.

5.1.1 Feature combinations

We train models on four feature combinations to assess how each data source affects prediction accuracy:

1. Price and volume only
2. Price, volume and sentiment
3. Price, volume, bitcoin network features and general market health indicators
4. Price, volume, bitcoin network features, general market health indicators and sentiment

By bitcoin network features we use data that is directly related to Bitcoin network activity, such as:

- **Hashrate** - total computational power of the network
- **Transaction count** - number of transactions processed in a given time period
- **Block size** - size of blocks added to the blockchain

These features can provide insights into the overall health and activity of the Bitcoin network, which may influence its price movements.

By general market health indicators we use data that reflects the broader financial market conditions, such as:

- **S&P500 price** - a stock market index that measures the stock performance of 500 large companies listed on stock exchanges in the United States

- **Gold price** - the price of gold, often considered a safe-haven asset during times of market uncertainty
- **VXX price** - the ticker symbol for the iPath Series B S&P 500 VIX Short-Term Futures ETN, which provides exposure to market volatility

These indicators can help capture the overall market sentiment and risk appetite, which may impact Bitcoin's price movements.

5.1.2 Sentiment aggregation

To incorporate sentiment data into our multi-dimensional dataset, we utilized in previous sections fine-tuned FinBERT model to predict sentiment scores for each headline in our cryptocurrency headlines dataset. So, we passed all our collected headlines through the model to obtain the sentiment scores. The sentiment scores, ranging from -1 (very negative) to 1 (very positive), were then aggregated on an hourly basis to align with the other time series data. So we received dataset that looks like Table 13.

Timestamp	**Title**	**Predicted Sentiment**
1629741303	Visa Makes Their Grand Entrance Into the NFT ...	0.00835
1630005294	Record breaking sale for Bored Ape	0.00352
1645216870	Binance has resumed deposits and withdrawals via bank transfer in UK.	-0.00661
1645268447	Canada court froze $20 million in cash that links back to Bitcoin ...	0.00789
1645278196	Europe is open to crypto, says EU Commissioner ...	0.00687
1645357315	JUST IN: El Salvador's GDP grew 10.3% ...	0.01391

Table 13. Sample of Predicted Sentiment Scores for Financial News Headlines

As our news headlines timing are not consistent (sometimes there are more headlines in a single hour and sometimes there are none) - we decided to implement a mechanism that aggregates sentiment over time.

To model the persistence of sentiment impact, we implement an exponential decay mechanism that propagates current sentiment into future 4 hours as it was recommended by Khalil & Pipa [13]. For each hour h with sentiment $S_{h,\text{mean}}$, we calculate its contribution to future hours - decayFactor $= \lambda^k$ where $\lambda = 0.5$ is the decay rate and k represents hours into the future ($k \in \{0, 1, 2, 3\}$).

The sentiment contribution from hour h to future hour $h + k$ is calculated as:

$$\text{contribution}_{h \to h+k} = S_{h,\text{mean}} \times \lambda^k \tag{5.1}$$

The final 4-hour decay sentiment score for any given hour j combines all overlapping contributions from the previous 4 hours:

$$S_{j,\text{4h-decay}} = \sum_{i=\max(0,j-3)}^{j} S_{i,\text{mean}} \times \lambda^{j-i} \tag{5.2}$$

This formulation ensures that:

- Current hour sentiment has full impact ($\lambda^0 = 1.0$)
- 1 hour later retains 50% impact ($\lambda^1 = 0.5$)
- 2 hours later retains 25% impact ($\lambda^2 = 0.25$)
- 3 hours later retains 12.5% impact ($\lambda^3 = 0.125$)

Multiple sentiment predictions affecting the same hour are combined additively, allowing the model to capture the cumulative effect of overlapping news sentiment on future price movements.

5.2 iTransformer model hyperparameters tuning

As previously - we utilized hyperparameter optimization with Optuna framework [2]. Our objective was to minimize the Huber loss on a held-out validation set using a reduced training corpus of 5000 examples, sampled randomly from the full dataset to accelerate the tuning process.

Table 14 outlines the hyperparameter search space for the iTransformer model. We experimented with various configurations of model dimension, depth, number of attention heads, and learning rate.

Table 14. iTransformer Hyperparameter Search Space

Parameter	Search Space	Optuna Suggestion API
dim	{32, 64, 128, 256, 512}	categorical
depth	2 to 6	integer
heads	{4, 8, 16, 32, 64}	categorical
learning rate	1e-5 to 1e-2 (log scale)	float

Table 15 summarizes the best hyperparameters identified for each feature set combination after conducting 50 Optuna trials per configuration.

Table 15. Best Hyperparameters Across Feature Set Experiments

Experiment	dim	depth	heads	lr
Price and volume only	64	2	16	0.009526
Price, volume and sentiment	64	2	16	0.007061
Price, volume and technicals	64	2	4	0.008929
Price, volume, sentiment and technicals	128	2	32	0.005194

5.3 iTransformer training process

The iTransformer model was trained using the best hyperparameters identified through the Optuna optimization process for each feature set combination.

Models were trained with best hyperparameters from Table 15 using:

- Batch size: 1024 - used for make training faster
- Number of epochs: 1000

- Optimizer: AdamW

Hardware: Larger feature sets (which involves price, volume, Bitcoin technicals and general market health) trained on Google Colab with L4 GPUs. Smaller sets trained on local machine with NVIDIA RTX 3060 (12GB) + Ryzen 7 3700X CPU.

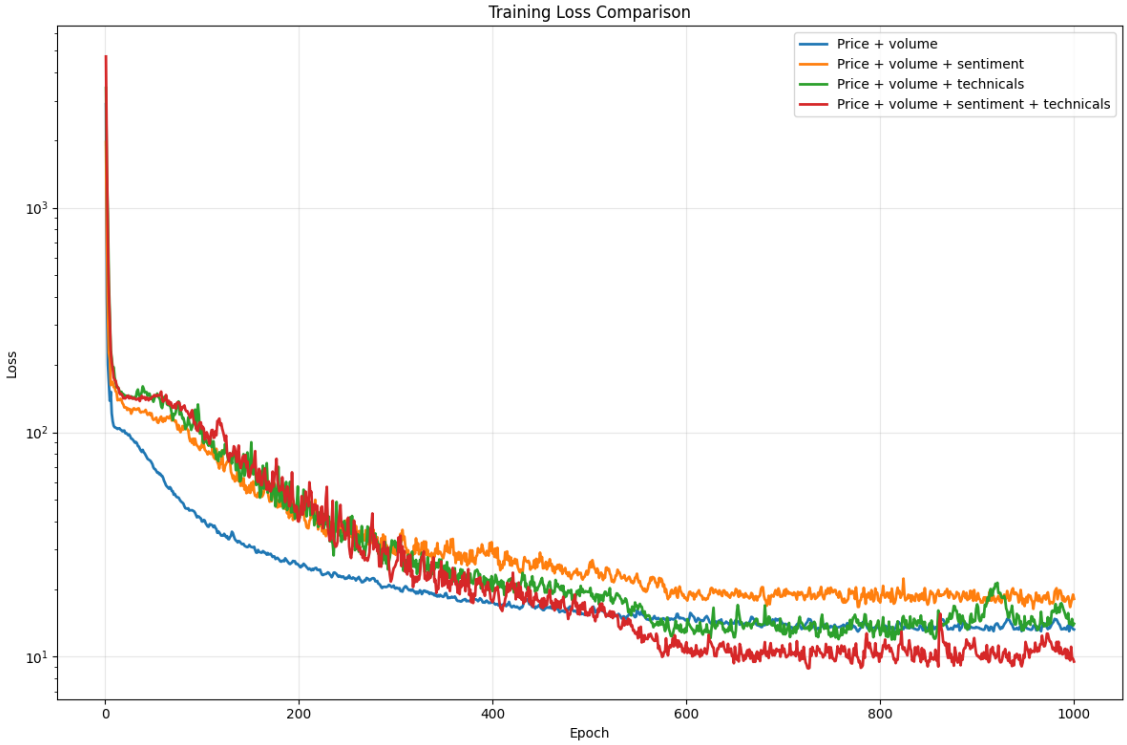


Figure 10. iTransformer training epochs for different data combinations

Figure 10 shows training convergence across feature combinations. As we can see from results - simple price and volume dataset model fitted fastest compared with other, more chaotic datasets. However, it converged to similiar loss as price, volume and Bitcoin technical value models, most likely meaning that introducing technical parameters of Bitcoin has not significantly improved model performance. However, when we introduce sentiment dataset into the models we can see that it introduced a bit chaos into simpler price & volume dataset, however when we used full dataset - it fitted the best.

6 Backtesting methodology and strategies

After training we obtained models that predict **next hour price changes** from historical inputs. All backtests used a simulated initial capital of 10,000$ and the last year of data as the validation period. We applied a conservative fixed taker fee of 0.0275% per executed trade:

- Bybit taker fee varies from 0.0270% till 0.0550%[2]
- Binance futures taker fees varies from 0.0084% till 0.0275%[3]

[2]https://www.bybit.com/en/announcement-info/fee-rate/ (Accessed: 2025-11-01)
[3]https://www.binance.com/en/fee/futureFee (Accessed: 2025-11-01)

- Hyperliquid fees varies from 0.0144% till 0.0450%[4]

We split validation into two regimes - an uptrend (2024-06-01 to 2024-12-01) and a sideways market (2025-01-01 to 2025-07-01) - to see how models behave under materially different conditions. Rolling-window (walk-forward) validation is often preferred, but it was not used here for time consuming and methodological reasons:

- Models should be updated with new information at every step we get, which is costly for transformer-based forecasting and sentiment pipelines;
- Most likely re-optimization of hyperparameters (f.e. thresholds) across many windows would be required. That adds complexity and additional time consumption. Also it lead to hyperparameter instability and accidental leakage if windows are misaligned with feature generation;

By using two fixed periods, we keep the setup simple, ensure clean separation between regimes, focus on comparative behavior of feature sets and not so time-consuming.

To evaluate realistic trading performance we implemented backtesting with three strategies designed to test different aspects of the model predictive capabilities:

- Simple threshold strategy: open a long/short when the model's predicted relative change exceeds a tuned threshold τ, and hold the position for at most H hours.
- Volatility scaled strategy: same entry rule as above, but with position size scaled to recent realized volatility and the same maximum holding period H.
- Time series momentum strategy: combine the model prediction with a time series momentum signal (comparison to price L hours ago) and open positions when the combined signal exceeds a threshold, again holding at most H hours.

6.1 Simple threshold strategy

The first backtesting approach employs a straightforward threshold-based trading strategy. At each hour we compare the predicted percentage price change r_{pred} with a threshold τ. If $r_{pred} > \tau$ we open or maintain a long position, if $r_{pred} < -\tau$ we open or maintain a short position, otherwise we stay flat. Once a position is opened, it is held for at most H hours, after which it is closed even if the threshold condition still holds. The threshold τ (and H when optimized) are selected by grid search over the validation period to maximize final portfolio value. For a compact pseudocode description see Appendix B.

6.2 Volatility-adjusted risk management strategy

The Volatility-Adjusted Risk Management Strategy uses the same entry and exit rules as the simple threshold strategy (including the maximum holding period H), but adjusts position size based on recent realized volatility:

$$\text{positionSize}_t = \text{clip}\left(\frac{\text{targetRisk}}{\sigma_t}, \text{minSize}, \text{maxSize}\right),$$

where σ_t is the rolling standard deviation of hourly returns over a window of week. This makes position size smaller in more volatile periods and larger in calmer markets, while still respecting the same threshold and holding-period rules.

[4]https://hyperliquid.gitbook.io/hyperliquid-docs/trading/fees (Accessed: 2025-11-01)

6.3 Time series momentum strategy

The third strategy combines model predictions with time series momentum signals to generate trading decisions:

$$\text{momentumSignal}_t = \frac{P_{t-1}}{P_{t-1-L}} - 1 \tag{6.1}$$

$$\text{predictionSignal}_t = \frac{P_{t-1} + \Delta P_{pred}}{P_{t-1-L}} - 1 \tag{6.2}$$

where L is the lookback window. In the implementation we use a volatility-based position sizing rule as in the previous subsection, and enter long/short positions when the prediction-based signal exceeds a tuned threshold. Positions are closed when the signal flips direction or when the maximum holding period H is reached. We optimize lookback L, holding period H and signal threshold over the validation year.

7 Hyperparameter search for trading strategies

To select trading parameters we performed a simple cross-validation based search on the training dataset only. The goal was to obtain a consistent set of thresholds and risk levels for all models while avoiding direct usage of the test period.

For each strategy type (simple threshold, volatility-adjusted, and momentum) we used k-fold cross-validation with Optuna to maximize the Sharpe ratio of the validation portfolio. The procedure is shown below in Algorithm 1, together with the main CV and Optuna settings used in our experiments.

Data: Training data, trained model, number of folds k, search space for parameters
Split training sequences into k chronological folds (no shuffling);
for *each fold* $f = 1 \dots k$ **do**
    Use fold f as validation, all earlier data as training context;
    Run time-series prediction model on inputs to obtain predicted price changes;
    Define Optuna objective function:
        Sample parameters (e.g. τ, H) from predefined ranges;
        Run the corresponding backtest on the validation fold;
        Compute Sharpe ratio from the portfolio value path;
        Return Sharpe ratio to Optuna (to be maximized);
    Run Optuna study for this fold with a fixed number of trials and time limit;
    Store the best parameters and best Sharpe for this fold;
end
Average the best parameters across folds to obtain a single configuration for each strategy;

Algorithm 1: K-fold CV parameter search with Optuna

In practice we used $k = 10$ folds without shuffling (timeseries order preserved) and ran Optuna in maximization mode with parallel workers on CPU. The concrete search spaces and optimization settings were:

- **Cross-validation**: $k = 10$ folds, no shuffling; only the training dataset is used for parameter search.
- **Threshold strategy**: $\tau \in [0.01, 2.0]$ (percent), $H \in [1, 168]$ hours; Optuna study with up to 200 trials.
- **Volatility-adjusted strategy**: $\tau \in [0.01, 2.0]$ (percent), per-trade risk target in $[0.01, 2.5]$ percent, $H \in [1, 168]$ hours; Optuna study with up to 500 trials.
- **Momentum strategy**: lookback window $L \in [1, 168]$ hours, holding period $H \in [1, 168]$ hours and signal threshold in a small positive range similar to the threshold strategy. Optuna study with up to 1000 trials.

In tables 16 we summarize the final optimized hyperparameters obtained via this CV and Optuna procedure for all datasets and strategies used in backtesting. These parameters were then fixed and used in the final evaluation on the test dataset as described in Section 6.

Dataset	**Threshold (τ)**	**Hold (H)**
Technicals Only	**1.19**%	**96** h
Sentiment Only	**1.33**%	**73** h
Price Only	**1.42**%	**88** h
Full Dataset	**1.12**%	**69** h

(a) Threshold strategy

Dataset	**Threshold (τ)**	**Target Risk**	**Hold (H)**
Technicals Only	**1.16**%	**1.10**	**95** h
Sentiment Only	**1.46**%	**0.79**	**65** h
Price Only	**1.36**%	**1.23**	**80** h
Full Dataset	**0.92**%	**0.81**	**49** h

(b) Risk-based (volatility-adjusted)

Dataset	**Lookback (L)**	**Threshold (τ)**	**Target Risk**	**Hold (H)**
Technicals Only	**54**	**0.913**%	**0.418**	**73** h
Sentiment Only	**107**	**1.067**%	**0.474**	**80** h
Price Only	**67**	**1.252**%	**0.685**	**79** h
Full Dataset	**71**	**1.05**%	**0.74**	**44** h

(c) Momentum

Table 16. Optimized hyperparameters via CV+Optuna. Values shown per dataset and strategy.

8 Experimental backtesting results

We conducted backtesting experiments with simulated 10,000$ portfolio across four different model configurations to assess the impact of various feature sets on trading performance those results are detailed in further sections.

8.1 Methodological limitations

Important note: the performance metrics presented in this section should not be interpreted as realistic out-of-sample expectations. These results has a few methodological limitations:

- Limited validation split: we used only two time periods within a single year rather than multiple folds or a walk-forward scheme. This small, single-year split does not allow assessment of parameter stability across different market regimes; a parameter set that used in 2024-2025 may act differently in other years.

- Idealized execution: only fixed taker fees (0.0275%) were modelled. Real trading would face additional slippage, market impact, partial fills, and liquidity constraints that could reduce returns.
- Low trade count: some strategies executed very few trades (e.g., 8, 12 or even 0), making metrics like Sharpe ratio and Win Rate unstable. Most likely it happens due to quite short testing periods and the threshold values used.

The purpose of presenting these metrics is comparative analysis - to evaluate the relative performance of different feature sets (price, sentiment, technicals) and strategies under controlled conditions, not to project realistic trading returns.

8.2 Simple threshold strategy

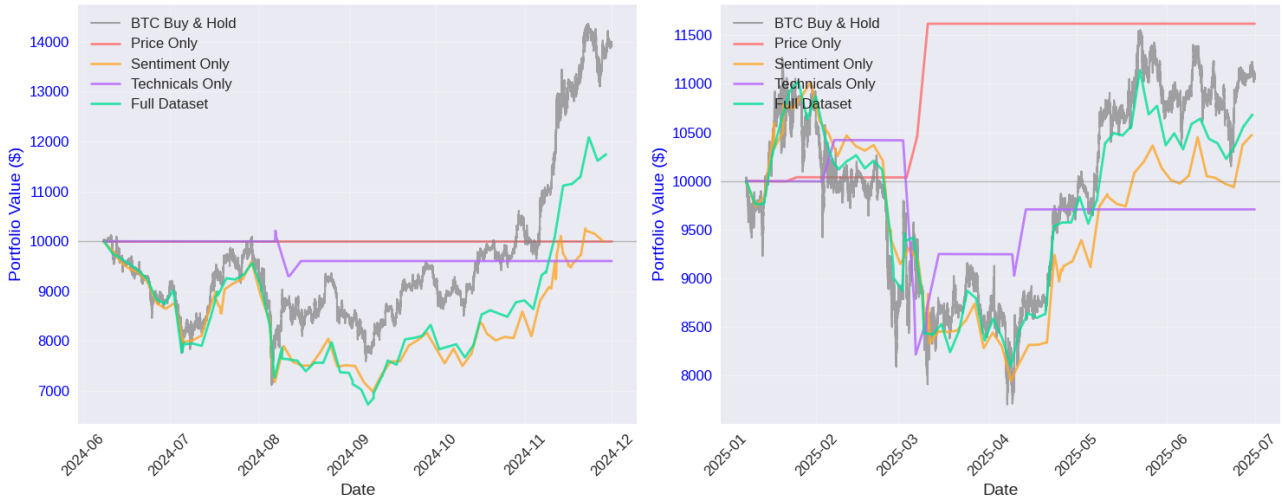


Figure 11. Simple threshold strategy backtesting results with different models comparison

First, we tested all four models using the simple threshold strategy described in Section 6.1. The plots in Figure 11 show the equity curves for both market periods. For the text, we focus on the final portfolio values from the backtests.

Dataset	Trades	Profitable	WR (%)	Max DD (%)	Sharpe	Final Value ($)
Uptrend Market Period						
Price Only	*0*	*0*	*0.00*	*0.00*	*0*	10,000.00
Technicals Only	12	2	18.18	**8.91**	-0.125	9,604.93
Sentiment Only	159	37	23.42	30.12	0.014	9,993.44
Full Dataset	161	41	**25.62**	32.70	**0.050**	**11,740.16**
Buy-and-hold	-	-	-	-	-	*13,800.61*
Sideways Market Period						
Price Only	8	3	**42.86**	**0.03**	**0.573**	**11,618.81**
Technicals Only	22	5	23.81	21.16	-0.016	9,708.53
Sentiment Only	153	39	25.66	27.90	0.025	10,471.80
Full Dataset	131	34	26.15	26.54	0.034	10,679.50
Buy-and-hold	-	-	-	-	-	*11,202.43*

Table 17. Threshold strategy detailed metrics for both market periods. Best performers in each market regime highlighted in bold.

In uptrend markets, the threshold strategy gave small gains or even small losses, depending on the dataset. The best final value in the uptrend period was reached by the "Full Dataset" model (11,740 $); the "Sentiment Only" and "Technicals Only" models finished close to or below the starting value, while the "Price Only" model did not trade at all with the given thresholds.

In sideways markets, the "Price Only" model did best in terms of final value (11,618 $) and Sharpe ratio (0.573), but it was based on only eight trades. The "Full Dataset" and "Sentiment Only" models also ended above the initial 10,000 $, but with lower returns and Sharpe values.

None of the models generated significant returns in this market regime and have not beaten simple buy-and-hold.

8.3 Volatility-adjusted risk-based strategy

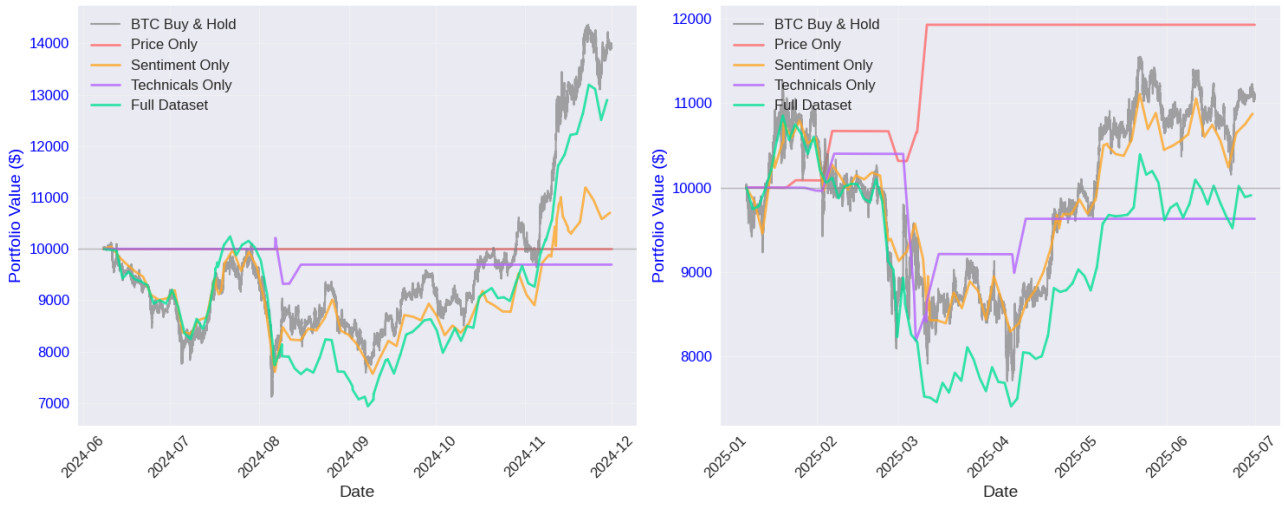


Figure 12. Risk-based strategy backtesting results comparison

Next, we tested the risk-based strategy, where position size is adjusted by volatility. Again, Figure 12 shows the curves, and Table 18 reports the main metrics.

Dataset	Trades	Profitable	WR (%)	Max DD (%)	Sharpe	Final Value ($)
Uptrend Market Period						
Price Only	*0*	*0*	*0.00*	*0.00*	*0*	10,000.00
Technicals Only	14	3	23.08	**8.77**	-0.084	9,693.21
Sentiment Only	167	37	22.29	24.66	0.029	10,700.39
Full Dataset	209	54	**25.96**	32.29	**0.064**	**12,895.57**
Buy-and-hold	-	-	-	-	-	*13,800.61*
Sideways Market Period						
Price Only	12	4	36.36	**3.36**	**0.431**	**11,931.57**
Technicals Only	24	5	21.74	21.35	-0.025	9,628.73
Sentiment Only	161	42	**26.25**	23.19	0.036	10,872.02
Full Dataset	179	39	21.91	31.80	0.008	9,905.34
Buy-and-hold	-	-	-	-	-	*11,202.43*

Table 18. Risk-based strategy detailed metrics for both market periods. Best performers in each market regime highlighted in bold.

Compared to the simple threshold approach, the risk-based strategy changes the results to a positive direction. In the uptrend market, the best final value is again from the "Full Dataset" model (12,895 $). In the sideways market, the "Price Only" model finishes highest (11,931 $) with a Sharpe of 0.431, but based on 12 trades.

Most configurations have win rates close to 20-30% and relatively large drawdowns, especially for the "Full Dataset" in both markets.

8.4 Momentum strategy results

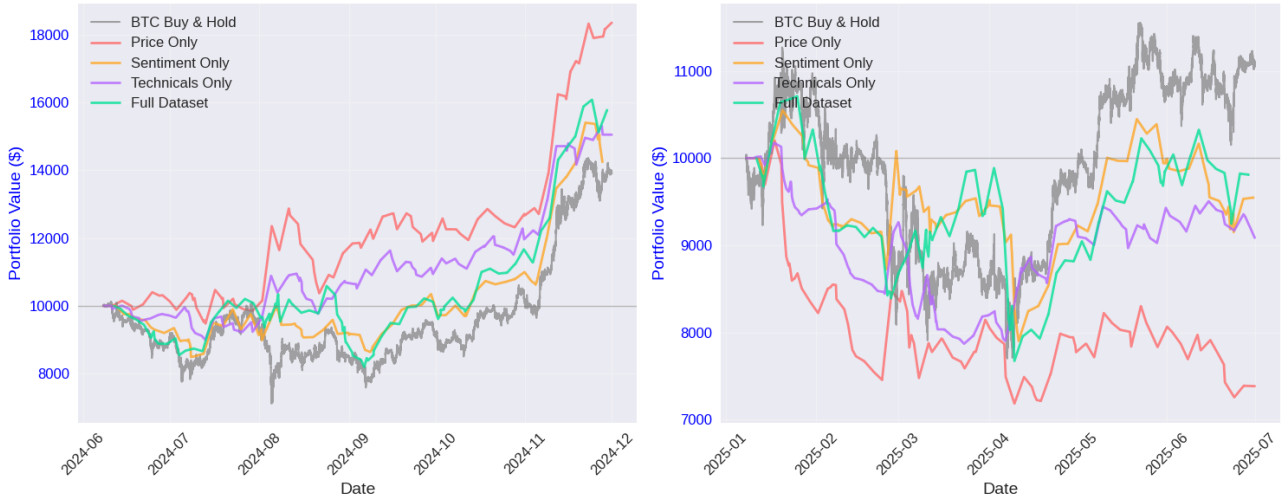


Figure 13. Momentum strategy backtesting results comparison

Finally, we tested the momentum strategy on the same four datasets. Figure 13 shows the equity curves, and Table 19 lists the detailed metrics.

Dataset	Trades	Profitable	WR (%)	Max DD (%)	Sharpe	Final Value ($)
Uptrend Market Period						
Price Only	173	38	22.09	19.42	**0.130**	**18,354.58**
Technicals Only	209	38	18.27	**10.80**	0.106	15,048.60
Sentiment Only	203	51	25.25	15.07	0.089	14,245.89
Full Dataset	223	57	**25.68**	22.55	0.103	15,771.70
Buy-and-hold	-	-	-	-	-	*13,800.61*
Sideways Market Period						
Price Only	176	25	14.29	29.58	-0.075	7,383.85
Technicals Only	239	36	15.13	**22.65**	-0.024	9,085.56
Sentiment Only	201	41	20.50	25.08	-0.004	9,544.51
Full Dataset	181	46	**25.56**	28.37	**0.006**	**9,807.57**
Buy-and-hold	-	-	-	-	-	*11,202.43*

Table 19. Momentum strategy detailed metrics for both market periods. Best performers in each market regime highlighted in bold.

In the uptrend market, all momentum models grow the 10,000 $ starting portfolio, with final values between about 14,200 $ and 18,400 $. The best result comes from the "Price Only" dataset (18,354 $), followed by the "Full Dataset" (15,772 $) and "Technicals Only" (15,049 $). The "Sentiment Only" model also gives a solid gain (14,246 $).

In the sideways market, all momentum models struggle and end below or close to the starting value. The "Full Dataset" model finishes closest to break-even (9,808 $), while the "Price Only" model performs worst (7,384 $). Win rates are low (mostly between 14% and 26%), and drawdowns are larger than in the threshold tests.

8.5 Backtesting results comparison

Table 17, Table 18 and Table 19 summarize the results for all three strategies and all datasets.

In the uptrend market, the highest final value is reached by the "Price Only" momentum model (18,354 $). The best threshold and risk-based results in this period are lower, with the "Full Dataset" risk-based model reaching about 12,896 $ and the "Full Dataset" threshold model about 11,740 $. This shows that, in a strong uptrend, the momentum approach with simple price and volume input works best in this setup.

In the sideways market, none of the strategies beat buy-and-hold. The best final values come from the "Price Only" and "Full Dataset" versions of the threshold and risk-based strategies (around 11,600 - 11,900 $), but these results are based on a small number of trades. The momentum models mostly lose value in this regime, with the "Full Dataset" version doing least badly (9,808 $).

Looking across datasets: "Price Only" performs best in uptrend markets with momentum (18,354 $) but generates no trades with threshold/risk-based strategies. "Full Dataset" is most consistent across all strategies and conditions (11,740 - 15,772 $ in uptrend, 9,807 - 10,679 $ in sideways). "Sentiment Only" is moderate (9,993 - 14,246 $ in uptrend, 9,544 - 10,472 $ in sideways) but underperforms in sideways markets. "Technicals Only" is the weakest overall (9,604 - 15,049 $ in uptrend, 9,085-9,708 $ in sideways). In summary: **"Price Only" excels in uptrends with momentum, "Full Dataset" provides most reliable and stable results, while "Sentiment Only" and "Technicals Only" add no meaningful improvement**.

Conclusions and recommendations

This thesis investigated whether sentiment analysis of cryptocurrency news headlines can enhance price forecasting accuracy when integrated with traditional technical indicators. Below findings are presented organized by research questions.

RQ1: News Impact Score Derivation

How can a news impact score be derived from price and volume data to quantify the real market effect of cryptocurrency-related headlines?

We developed a headline-to-price impact parameter by evaluating various price and volume-derived weight methods. Through TF-IDF SVM comparison, we identified $V_t^{\text{adjustedPrice}}$ (Equation 3.8) as most effective, integrating volatility, volume, and price into a unified metric. ΔP_t^{EMA} and $\Delta P_t^{\text{EMAVolatility}}$ (Equations 3.4, 3.5) proved viable alternatives when volume data is unavailable.

Conclusion: Market impact can be quantified using composite metrics combining price, volatility, and volume—providing a foundation for training sentiment models on real market effects rather than subjective labels.

RQ2: Fine-Tuned Sentiment Model Performance

Can fine-tuned sentiment models using custom cryptocurrency datasets outperform pre-trained models and traditional NLP approaches in predicting market impact?

We compared traditional machine learning models (SVR, GBR with TF-IDF) against fine-tuned transformer models (FinBERT, Finance Qwen) adapted for regression. Fine-tuned FinBERT with 2 layers unlocked achieved best performance in outperforming with "sentiment-price" correlation both traditional NLP approaches and pre-trained models.

Conclusion: Domain-specific fine-tuning on cryptocurrency headlines paired with market-impact labels substantially outperforms generic approaches. Regression-based sentiment captures nuanced market responses that categorical classification (positive/negative/neutral) cannot represent.

RQ3: Sentiment Integration in Price Forecasting

Does integrating sentiment analysis with traditional technical indicators improve the accuracy of cryptocurrency price forecasting models?

We integrated fine-tuned FinBERT outputs into iTransformer based forecasting models and evaluated performance through backtesting across three strategies (threshold, volatility-adjusted, momentum) in uptrend and sideways markets. Results shows mixed results for sentiment's contribution.

Conclusion: Sentiment improves forecasting, however, its benefits are shown only while incorporated with other features and should be used as an additional input rather than a standalone predictor.

Summary and Practical Implications

Both research objectives were achieved:

Objective 1: Developed market-impact-based sentiment models quantifying real price effects. Fine-tuned FinBERT demonstrated more consistent performance than traditional approaches.

Objective 2: Integrated sentiment into forecasting models and conducted systematic backtesting. Established that sentiment provides prediction improvements while in combination with various technical Bitcoin and market health indicators.

Key practical insights:

- Sentiment analysis for finance could be regression-based on actual market outcomes, not only subjective labels.
- Sentiment can be used not only as a binary/ternary classifier but as a continuous variable reflecting magnitude of market impact.
- Sentiment alone is insufficient; requires combination with technical indicators and proper risk management.
- Sentiment analysis should be used incorporate with other more advanced models due to the noisy and chaotic nature of headline sentiment, which can overwhelm simple models yet provides useful contextual signal when integrated with many complementary predictors.

Final assessment: Sentiment analysis can enhance cryptocurrency price forecasting when carefully derived and integrated, but should be used as a complementary input into forecasting models. Future work should explore more sophisticated fusion techniques and broader datasets to further unlock sentiment's potential in financial modeling.

Future work plan

In this work we touched everything just a bit, so there are many possible future directions and improvements to be made.

Sentiment analysis and modelling

- Systematically evaluate additional vectorization methods (e.g., `Word2Vec`) in combination with different machine learning algorithms and compare them against the currently used approaches. Other methods may yield better sentiment representations. Also, simpler models may work better in real-time applications and may reduce unescessary computational time.
- Explore larger and more specialised language models and investigate autoencoder-based adaptations to the target sentiment score. Other models may better capture financial context.
- Use better dataset for sentiment model training. The current dataset is taken from telegram channels, however good datasets from financial news (f.e. Reuters, Bloomberg) are often are expensive or not publicly available. This should improve the quality of sentiment predictions.

Overall, these improvements would help to find best sentiment score prediction with the reduced noise in the sentiment data extracted from news.

Time series forecasting and data

- Investigate alternative architectures for integrating sentiment features into time series models. Other advanced architectures could better capture long-term dependencies. F.e. in our work we were planning to try xLSTM models and benchmark it against current Transformer solution but time constraints prevented us from doing so.
- Incorporate more diverse datasets (other cryptocurrencies, traditional assets, macroeconomic indicators) to test the robustness and generalisability of the results across different markets and regimes. For instance, the analysis could be extended to include Ethereum and macroeconomic indicators like the precious metals price or M2 money supply.

In summary, more sophisticated models and diverse data are needed to find the best forecasting approach that effectively incorporates noisy sentiment information.

Backtesting and trading strategy evaluation

- Implement and compare more advanced trading rules, stop-loss mechanisms and basic portfolio diversification, in order to better approximate real trading scenarios.
- Extend transaction cost modelling beyond a fixed taker fee to include slippage, liquidity constraints, partial fills and approximate market impact.
- Use rolling-window (walk-forward) validation for model evaluation. This mitigates overfitting risks. Specifically, a walk-forward validation with expanding windows should be employed to assess performance stability over time.

Ultimately, these steps would reduce the gap between theoretical backtesting results and practical trading viability.

References

[1] Cengiz Asmazoğlu Abdullah Bezir, Furkan Burhan Türkay. Wiroai/wiroai-finance-qwen-1.5b, 2025.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[3] Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models. *CoRR*, abs/1908.10063, 2019.

[4] Mohammed Baz, Abdullah Sami, Nusrat Prottasha, Md Kowsher, Saydul Murad, Anupam Bairagi, and Mehedi Masud. Transfer learning for sentiment analysis using bert based supervised fine-tuning. *Sensors*, 22, 05 2022.

[5] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael K Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended long short-term memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[6] Bybit. Bybit public market data. https://public.bybit.com/trading/, 2024. Accessed: 2025-05-31.

[7] CNN Business. Fear & greed index. https://edition.cnn.com/markets/fear-and-greed, 2025. Accessed: 2025-05-07.

[8] Eugene F. Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383, May 1970.

[9] Joseph G. Flowers. Finance-instruct-500k, 2025.

[10] Markus Glaser, Markus Nth, and Martin Weber. Behavioral finance. In *Blackwell Handbook of Judgment and Decision Making*, pages 525–546. Blackwell Publishing Ltd, Malden, MA, USA, 2008.

[11] Google Cloud. Bitcoin in bigquery: blockchain analytics on public data. https://cloud.google.com/blog/topics/public-datasets/bitcoin-in-bigquery-blockchain-analytics-on-public-data, 2017. Accessed: 2025-10-12.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 11 1997.

[13] Faisal Khalil and Gordon Pipa. Is deep-learning and natural language processing transcending the financial forecasting? investigation through lens of news analytic process. *Comput. Econ.*, 60(1):147–171, June 2022.

[14] Wasiat Khan, Mustansar Ali Ghazanfar, Muhammad Awais Azam, Amin Karami, Khaled H Alyoubi, and Ahmed S Alfakeeh. Stock market prediction using machine learning classifiers and social media, news. *J. Ambient Intell. Humaniz. Comput.*, 13(7):3433–3456, July 2022.

[15] Thanos Konstantinidis, Giorgos Iacovides, Mingxue Xu, Tony G. Constantinides, and Danilo Mandic. Finllama: Financial sentiment classification for algorithmic trading applications, 2024.

[16] Mikolaj Kulakowski and Flavius Frasincar. Sentiment classification of cryptocurrency-related social media posts. *IEEE Intell. Syst.*, 38(4):5–9, July 2023.

[17] Yuze Li, Shangrong Jiang, Xuerong Li, and Shouyang Wang. The role of news sentiment in oil futures returns and volatility forecasting: Data-decomposition based deep learning approach. *Energy Economics*, 95:105140, 2021.

[18] Diefan Lin, Yi Wen, Weishi Wang, and Yan Su. Enhanced sentiment intensity regression through lora fine-tuning on llama 3. *IEEE Access*, 12:108072–108087, 2024.

[19] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting, 2024.

[20] Burton G Malkiel. The efficient market hypothesis and its critics. *J. Econ. Perspect.*, 17(1):59–82, February 2003.

[21] Burton G Malkiel. *A random walk down Wall Street*. WW Norton, New York, NY, 9 edition, February 2007.

[22] ALAN MOREIRA and TYLER MUIR. Volatility-managed portfolios. *The Journal of Finance*, 72, 04 2017.

[23] Tobias J. Moskowitz, Yao Hua Ooi, and Lasse Heje Pedersen. Time series momentum. *Journal of Financial Economics*, 104(2):228–250, 2012. Special Issue on Investor Sentiment.

[24] Dehua Shen, Andrew Urquhart, and Pengfei Wang. Bitcoin intraday time series momentum. *Financial Review*, 57, 10 2021.

[25] Yhlas Sovbetov. Factors influencing cryptocurrency prices: Evidence from bitcoin, ethereum, dash, litcoin, and monero. *Journal of Economics and Financial Analysis*, 2:1–27, 02 2018.

[26] Richard H. Thaler. Richard h. thaler, misbehaving: The making of behavioral economics. *The Review of Austrian Economics*, 30, 09 2015.

[27] M. Ángeles López-Cabarcos, Ada M. Pérez-Pico, Juan Piñeiro-Chousa, and Aleksandar Šević. Bitcoin volatility, stock market and investor sentiment. are they connected? *Finance Research Letters*, 38:101399, 2021.

Appendices

There are 2 appendices in total, in appendix A the hyperparameters for all sentiment models are detailed, and in appendix B the Simple Threshold Trading Strategy algorithm is described.

A Hyperparameters for all sentiment models

Model	Parameter	Value
TF-IDF + SVR	Vectorizer N-gram Range	(1, 7)
	Max DF	0.1
	Min DF	1
	SVR Kernel	Polynomial (degree 6)
	Regularization (C)	0.25
	Epsilon	0.001
TF-IDF + GBR	Vectorizer N-gram Range	(1, 7)
	Max DF	0.1
	Learning Rate	0.1
	Max Depth	5
	Subsample	0.8
	N Estimators	200
FinBERT	Unlocked Layers	Last 2 encoder layers
	Optimizer	AdamW
	Learning Rate	7.35×10^{-5}
	Weight Decay	0.094
	Loss Function	Adaptive Weighted Huber
	Epochs	25 (early stopping)
	Batch Size	16
Finance Qwen	Unlocked Layers	Last 6 decoder layers
	Optimizer	SGD
	Learning Rate	6×10^{-3}
	Weight Decay	0.012
	Momentum	0.9
	Loss Function	Adaptive Weighted Huber
	Epochs	15

B Backtesting algorithms

All three strategies are implemented using the same generic backtesting loop, with different prediction and position-sizing functions.

Generic backtest loop

Result: Final portfolio value C_T
Data: Prices P_t, predictions x_t, threshold τ, transaction cost c, max holding period H, size function sizeFunc(t), prediction function predFunc(t)

Initialize capital $C = 10{,}000$, position $p = 0$, position size $s = 0$, barsInPosition = 0;
for *each hour t from 1 to T* **do**
  if $p \neq 0$ **then**
    $r_{actual} = (P_t - P_{t-1})/P_{t-1}$;
    $C = C \times (1 + p \cdot s \cdot r_{actual})$ `// update capital`
    barsInPosition ← barsInPosition +1;
  end
  r_{pred} = predFunc(t) `// predicted percentage change / signal`
  Set desired direction d (1 if $r_{pred} > \tau$, -1 if $r_{pred} < -\tau$, else 0);
  if $p = 0$ *and* $d \neq 0$ **then**
    $p = d$; s = sizeFunc($t - 1$);
    $C = C \times (1 - s \cdot c)$;
    barsInPosition = 0;
  end
  else if $p \neq 0$ **then**
    if $d \neq 0$ *and* $d \neq p$ **then**
      `// flip`
      $p = d$; s = sizeFunc($t - 1$); $C = C \times (1 - s \cdot c)$; barsInPosition = 0;
    end
    else if *barsInPosition* $\geq H$ **then**
      `// max holding period`
      $p = 0$; $s = 0$; barsInPosition = 0;
    end
  end
end
Return final capital C_T;

Algorithm 2: Generic backtesting loop with holding period

Simple threshold strategy

For the simple threshold strategy we use:

- sizeFunc(t) = 0.99 (constant fraction of capital),
- predFunc(t) $= \dfrac{P_{t-1} + \Delta P_t^{pred} - P_{t-1}}{P_{t-1}}$ (predicted percentage change).

Volatility-adjusted threshold strategy

For the volatility-adjusted strategy we use the same predFunc(t) as above, but a volatility-based size function:

- Compute rolling volatility σ_t of hourly returns over a window of length volWindow,
- Raw size $= \dfrac{\text{targetRiskPerTrade}}{\sigma_t}$,
- sizeFunc(t) = clip(raw size, minSize, maxSize).

Time series momentum strategy

For the time series momentum strategy we again use a volatility-based sizeFunc(t) and define the prediction function using a lookback window L:

- predFunc(t) $= \dfrac{P_{t-1} + \Delta P_t^{pred} - P_{t-1-L}}{P_{t-1-L}}$,
- the loop starts at $t = 1 + L$ so that P_{t-1-L} is defined.

The same thresholding and holding-period rules from Algorithm 2 are then applied to this momentum-style signal.

C Data splits used in the work

The table below summarizes the date ranges used for the different stages of the study. All splits are chronological (no shuffling) so that training data strictly precedes validation and testing data to avoid time leakage.

Table 20. Data splits

Stage	Split	Date range	Notes
Sentiment	Train	2021-08 - 2024-04 (minute)	Chronological training; approx. 17,500 examples
	Test	2024-04 - 2025-04	Last-year held-out test; approx. 2,500 examples
Forecasting	Train	2019-10 - 2024-04 (hourly)	Main training period using multi-dimensional inputs
Backtesting	Uptrend regime	2024-06 - 2025-01	Backtest uptrend window
Backtesting	Sideways regime	2025-01 - 2025-07	Backtest sideways window