UCLouvain

LOUVAIN
School of Management

Louvain School of Management

Analyzing the impact of news on stock price predictions using high and low frequency signals

Author(s): Adrien Payen
Supervisor(s): Corentin Vande Kerckhove
Academic year 2024-2025
Dissertation for the master of
Business Analytics
Daytime schedule

Abstract

Acknowledgments

I express my sincere gratitude to Delphia for the trust they have placed in me throughout this project. Their support has been invaluable and I am especially grateful for the data they provided which formed the backbone of my research.

I would also like to extend my heartfelt thanks to Professor Vande Kerckhove for his guidance, feedback and constant encouragement. His expertise has been instrumental in shaping this research and helping me navigate the complexities of this topic.

Finally, I would like to acknowledge all those who have contributed to this project, either directly or indirectly. Your support, insight and encouragement have been greatly appreciated. I am grateful to each one of you.

Contents

1 Introduction

In a rapidly evolving financial landscape, the role of information in shaping market behavior has become increasingly significant. One of the most intriguing and complex challenges is understanding the relationship between prediction markets and their influence on stock price movements. This thesis addresses the problem of quantifying how predictive signals derived from prediction markets impact stock prices.

1.1 The Current World of Prediction Markets and Stock Prices

In today's financial world, markets are flooded with a wide range of information that can impact investor behavior (Li et al., 2014). Prediction markets, also known as event futures or information markets, aggregate individual opinions and probabilities about future events, offering valuable signals about the direction of various economic, political or social factors (Waitz and Mild, 2013; Wolfers and Zitzewitz, 2004). These markets provide a unique opportunity for investors to gauge future expectations and anticipate market movements.

1.2 Research Question

This thesis focuses on the research question: *How do prediction markets signals affect stock prices?* To answer this, we explore the potential mechanisms through which these signals influence investor behavior and, by extension, the market valuation of assets. By investigating prediction markets signals, this research aims to build a comprehensive model for stock price prediction that accounts for the full spectrum of information available to investors.

To address this question, we adopt a comparative modeling strategy. Specifically, we assess the relevance and performance of a global modeling approach — a single regression model applied across all stocks, incorporating both returns and exogenous variables common to all assets — versus a stock-specific model, where a distinct model is trained for each individual stock. This methodology enables to determine whether a shared structure, enriched by exogenous information from prediction markets, can generalize across different stocks, or whether asset-specific nuances require tailored models.

1.3 Managerial Relevance

This research holds substantial managerial relevance for Delphia, a Canadian fintech specializing in data-driven investment strategies. The implications are outlined below :

1. Enhanced data-driven investment strategies: By analyzing how predictive signals from prediction markets influence stock prices, Delphia can refine its investment models. Integrating these signals into algorithms could enhance decision-making processes and potentially improve portfolio performance (Waitz and Mild, 2013).

2. Improved forecasting accuracy: Prediction markets aggregate public sentiment and probabilities about future events, offering valuable insights to anticipate market trends. By combining these signals with Delphia's existing data models, the accuracy of stock price predictions could be significantly improved, leading to more informed investment decisions and better risk management.

3. Detection of market inefficiencies and early trends: This research can aid Delphia in identifying market inefficiencies and emerging trends by leveraging prediction market data. This proactive approach could allow the company to capitalize on shifts in market dynamics before competitors.

4. Risk management optimization: Incorporating prediction market signals would bolster Delphia's ability to detect potential risks during periods of volatility or uncertainty. This enhanced risk management capability could help protect the company's portfolios from significant losses and reduce exposure to high-risk investments.

5. Innovation in investment products: The findings could inspire the development of new financial products that utilize prediction market signals. For instance, funds tailored to exploit trends from these markets could offer clients innovative, high-return investment options.

6. Competitive advantage through strategic information use: Expanding Delphia's data integration to include prediction markets could provide a distinct competitive edge. By harnessing diverse information sources, the company can position itself as a leader in identifying and act-

ing on emerging trends, attracting clients and increasing its market presence.

Overall, this research aligns closely with Delphia's mission of leveraging data to inform investment decisions. The insights gained can optimize strategies, improve predictive accuracy and strengthen the company's position in the fintech sector.

1.4 Thesis Structure

This thesis is organized into chapters. The second chapter, Literature Review, examines the existing literature on prediction markets, stock price dynamics and recent advancements in global modeling applied to finance. This sets a comprehensive foundation for the research. In the third chapter, Methodology, the focus is on data collection methods, the techniques employed to implement global modeling and the integration of prediction markets to enhance the understanding of stock market movements. This chapter provides a detailed explanation of the methodological framework, including the implementation of prediction market data and global modeling. Chapter four, Analysis of the results, presents the findings of the empirical analysis, showcasing how prediction markets influence stock prices and evaluating the effectiveness of global modeling techniques in interpreting these dynamics. The fifth chapter, Discussion, interprets the results, compares them with existing literature and discusses their implications for both theory and practice. Finally, the Conclusion summarizes the research, highlights its implications for investors and regulators and suggests areas for future research.

2 Literature Review

This section examines previous theories and research on the valuation of information and its influence on financial markets. It reviews key studies, theoretical frameworks and findings relevant to understanding the connection between prediction markets and stock market responses.

2.1 Prediction Markets

Prediction markets are platforms where participants trade contracts that pay out based on the outcome of future events (Wolfers and Zitzewitz, 2004). These markets aggregate the individual predictions of participants, turning them into a collective forecast of the likelihood that an event will occur. The concept of prediction markets is rooted in the idea that the collective wisdom of a diverse group of individuals, each with access to different information and insights, can generate more accurate forecasts than any individual expert (Bossaerts et al., 2022). These markets function similarly to financial markets (Wolfers and Zitzewitz, 2004), where participants buy and sell contracts based on their predictions of future outcomes. For instance, participants might buy a contract that pays out if a specific political event occurs, such as the election of Trump against Kamala Harris during the 2024 presidential race (Mongrain and Stegmaier, 2024), or if a company meets a certain financial target.

One of the key features of prediction markets is their ability to aggregate information. As discussed by Bossaerts et al., 2022, these markets incorporate diverse viewpoints, enabling them to reflect the collective intelligence of participants. This phenomenon is often referred to as the ”wisdom of crowds”. Berg et al., 2008, have shown that prediction markets can outperform individual experts in terms of forecasting accuracy, as they incorporate a wide range of opinions.

These markets are typically structured as winner-take-all markets or vote-share markets. In winner-take-all markets (Dai et al., 2021), participants predict a single outcome and those who predict the correct event share the prize. This market model offers a fixed payout for the correct prediction, which is typically the same for all winners. Vote-share markets involve trading based on the estimated proportion of votes or shares in a specific outcome, often used for more detailed or continuous predictions, such as estimating the exact vote share in an election or market share in a competition (Dai et al., 2021).

Research has shown that prediction markets can effectively predict not only political events but also economic trends, such as the likelihood of a recession or the movement of stock prices (Wolfers and Zitzewitz, 2004). These markets function as a tool to forecast economic indicators, company performance, or even geopolitical events. By integrating real-time information and the collective insights of a diverse set of participants, prediction markets offer a unique and valuable source of predictive signals that can inform decision-making in various domains, including finance. For example, Google estimates its market capitalization using prediction markets to forecast its value prior to an initial public offering (Berg et al., 2009).

2.2 Conceptual and theoretical framework

2.2.1 Efficient Market Hypothesis (EMH)

The Efficient Market Hypothesis (EMH), developed by Eugene Fama, posits that asset prices fully and instantaneously reflect all available information (Naseer, 2016). This theory assumes that markets are highly efficient in processing and integrating information into prices, leaving no opportunity for consistent outperformance based on publicly available data.

In this context, prediction markets can be viewed as a mechanism to enhance market efficiency by aggregating dispersed and heterogeneous information from a diverse set of participants (Downey, 2024). These markets function by incentivizing participants to trade based on their knowledge, insights, or expectations of future outcomes, effectively pooling collective intelligence.

Unlike traditional financial markets, which rely on a combination of fundamental and technical analysis, prediction markets are explicitly designed to reflect probabilities of future events (Nti et al., 2020; Naseer, 2016). By doing so, they may uncover ”hidden” information that would otherwise remain unintegrated into asset prices. This ability to aggregate a wide array of viewpoints and data sources positions prediction markets as a complementary tool for improving the informational efficiency of financial markets.

2.2.2 Investor behavior

Herding : Herding behavior is a significant phenomenon in financial markets, where investors mimic the actions of others rather than relying on their own private information. This tendency often arises during periods of market stress, leading to deviations of asset prices from their intrinsic values. Research highlights that herding is particularly pronounced in emerging markets, where information asymmetry and market inefficiencies are more prevalent (Ah Mand et al., 2023). For example, in the Malaysian stock market, evidence suggests that herding behavior exhibits non-linear characteristics, with variations between up and down market conditions. Shariah-compliant stocks tend to demonstrate herding more significantly during upward market movements, while conventional stocks show limited evidence of herding. This behavior has critical implications for market stability, as it may amplify volatility and hinder market efficiency.

Financial influencers : Financial influencers on social media platforms, particularly those categorized as ”mega influencers” with over one million followers, have a unique ability to shape investor behavior and market dynamics. Studies demonstrate that posts from these influencers can significantly affect investor attention, trading volume and stock price volatility (Keasey et al., 2024). However, their influence on stock returns is limited, requiring posts with extreme sentiment from top influencers to elicit short-lived impacts on returns. This aligns with the ”noise trader” hypothesis, which posits that uninformed trading triggered by such influencers introduces temporary mispricing that reverses over time. By analyzing over 16 million Instagram posts, researchers have highlighted the importance of sentiment and engagement metrics, such as comment volume, in amplifying the visibility and potential market impact of influencer content. These findings underscore the dual-edged role of influencers in promoting market participation while potentially fostering instability through noise trading.

3 Methodological Framework

This section outlines the conceptual and theoretical underpinnings of the research. It begins with a comparative modeling strategy employed, contrasting global versus local models, and explores the theoretical principles, such as the bias-variance tradeoff, guiding this choice. Subsequently, it articulates the testable hypotheses that stem from this framework. The research design is then detailed, covering the overall experimental architecture, the rationale for employing temporal validation methods like walk-forward analysis with specific parameters used in the implementation, and the considerations for its practical execution. Finally, it describes the evaluation methodology, including the selection of Root Mean Squared Error (RMSE) as the primary performance metric and the use of the Diebold-Mariano test for comparing forecasting model efficacy, as implemented in the analysis.

3.1 Comparative Modeling Strategy: Global vs. Local Approaches

In financial time series forecasting, one crucial methodological decision involves the level of aggregation in model training — whether to use a single unified model across multiple assets (global modeling) or to estimate distinct models for each asset (local modeling). This distinction has important implications for model complexity, data efficiency, and forecasting accuracy. The present research implements and compares both approaches in the context of predicting daily stock returns using signals from prediction markets as exogenous inputs.

3.1.1 Global Models

A global model is trained using the pooled data from all stocks in the dataset. In this study, a tree-based ensemble algorithm — specifically XGBoost — was used for global modeling due to its ability to handle high-dimensional, heterogeneous features and capture non-linear interactions. The premise of global modeling lies in the assumption that underlying drivers of stock returns share common structures across assets — such as macroeconomic shocks, investor sentiment, or systemic risk factors — that can be learned jointly (Hartford et al., 2018; Gu et al., 2020).

By training a model across all series simultaneously, the global approach benefits from a larger effective sample size, which typically reduces variance and improves generalizability (Montero-

Manso and Hyndman, 2021). Furthermore, the presence of exogenous variables — such as aggregated sentiment from prediction markets — reinforces the potential effectiveness of a global structure by exploiting shared informational signals. However, global models may suffer from specification bias if individual assets exhibit idiosyncratic behaviors that cannot be adequately captured by a single unified function.

3.1.2 Local Models

In contrast, local models are estimated separately for each stock. Each asset is modeled independently, allowing the estimation process to capture its unique autoregressive patterns and responsiveness to prediction market signals. For this study, we implemented ARIMA and ARIMAX models, which are well-suited to univariate or low-dimensional time series data with clear temporal dependencies. The local approach assumes that stocks respond to signals in a firm-specific manner, which may reflect structural differences in sectors, liquidity, or investor composition (Lopez de Prado, 2018; Tashman, 2000).

Local modeling allows for lower bias by fitting asset-specific dynamics; however, it comes at the cost of higher estimation variance, especially when limited historical data is available per asset. This tradeoff becomes particularly acute in short time series, where overfitting and parameter instability can degrade forecasting performance. From a practical standpoint, maintaining and updating hundreds of separate models also poses operational complexity, especially in high-frequency or live trading environments.

3.1.3 Bias-Variance Tradeoff and Model Evaluation

This dual modeling approach is grounded in the classical bias-variance tradeoff (Geman et al., 1992). Global models, by pooling data, offer stable predictions with lower variance but may introduce systematic bias if heterogeneity across series is substantial. Local models are more flexible and tailored, potentially reducing bias, but are more prone to variance-driven errors. This empirical analysis evaluates these tradeoffs in the context of out-of-sample forecasting error, using Root Mean Squared Error (RMSE) and the Diebold-Mariano test to compare model performance statistically across assets and time periods.

3.2 Testable Hypotheses Derived from the Framework

The following hypotheses guided the empirical investigation:

- **H1:** Prediction markets provide significant incremental information for stock return forecasting. This posits that signals from prediction markets, when incorporated into forecasting models, improve predictive accuracy compared to models using only historical data.
- **H2:** The informational value of prediction market signals for stock return forecasting is more pronounced in local (stock-specific) models than in global (market-wide) models. This explores whether local models better leverage granular prediction market information.
- **H3:** The nature and strength of the relationship between prediction market signals and stock returns exhibit heterogeneity across different stocks. This acknowledges that the impact of prediction market signals may vary by factors like sector or company characteristics (though the implemented code focuses on model comparison rather than deep analysis of this heterogeneity for H3).

3.3 Research Design

The research design delineated the systematic approach to empirically test the hypotheses, ensuring robustness in evaluating model performance.

3.3.1 Overall Experimental Structure

The experiment assessed the forecasting accuracy of global and local models for each individual stock. Two modeling paradigms were compared:

- **Global model:** A single XGBoost model was trained using features including prediction market signals to forecast returns for all stocks collectively. This model captures potential cross-stock patterns and global market dynamics.
- **Local models:** A distinct ARIMA-based model (ARIMA or ARIMAX) was built for each stock. For ARIMAX models, prediction market signals were included as exogenous regressors, but only if they exhibited Granger causality with the target stock (i.e.,

if they were shown to provide predictive power over past values of the stock in a time series sense). If no prediction market signals passed the Granger test for a given stock, a standard ARIMA model was used instead.

The comparison was made on a per-stock basis: for each stock, the performance of its local model was compared against the global model's forecast for that same stock. This design allowed a direct evaluation of whether stock-specific (local) models leveraging causally relevant prediction signals outperformed a global, feature-rich model.
Performance was evaluated on an out-of-sample period using walk-forward validation to ensure temporal robustness.

3.3.2 Temporal Validation (Walk-Forward)

Given the temporal nature of financial time series, a walk-forward validation approach was implemented using the `time_series_cv` function from the `timetk` package in R. This method preserves the chronological order of observations and simulates real-world forecasting scenarios, where models are periodically re-estimated as new data becomes available. Such an approach is essential to avoid look-ahead bias and to assess model stability and adaptability in dynamic financial markets.

3.3.3 Number and Duration of Folds

Walk-forward validation was configured with the following parameters for the `time_series_cv` function:

- `initial = "150 days"`: approximately 5 months of data used for the initial training set.
- `assess = "75 days"`: approximately 2.5 months allocated for each test (assessment) period.
- `skip = "75 days"`: the window advances by 2.5 months after each assessment period.
- `cumulative = TRUE`: the training window expands cumulatively, incorporating previous assessment data.

3.4 Evaluation Methodology

The evaluation methodology provided a comprehensive and statistically sound assessment of model predictive performance.

3.4.1 Choice of RMSE as the Primary Metric

The Root Mean Squared Error (RMSE) was chosen as the primary metric for evaluating forecast accuracy. RMSE penalizes larger errors more heavily and is expressed in the same units as the target variable (stock returns). It was computed using the `modeltime_accuracy` function from the `modeltime` package in R.

3.4.2 Choice of the Diebold-Mariano Test

To statistically compare the predictive accuracy of two competing forecasting models (e.g., global vs. local models), the Diebold-Mariano (DM) test was employed using the `dm.test` function from the `forecast` R package. The null hypothesis of the test is that both models have equal forecast accuracy. The test was conducted on the forecast errors (denoted as $e1_t$).

The `h` parameter (forecast horizon) for the DM test was defined as `max(1, min(1, floor(n_obs / 2)))`, where n_obs represents the number of common observations between the two error series. This heuristic is commonly used when a specific forecast horizon is not pre-defined or is expected to be short. The test was performed using a two-sided alternative hypothesis (`alternative = "two.sided"`).

3.4.3 Criteria for Interpreting Results

Interpretation of the results relied on RMSE values and the statistical significance (p-values) obtained from the Diebold-Mariano tests. A p-value below 0.05 was considered indicative of a statistically significant difference in forecast accuracy. The results were presented using formatted tables generated with the `kable` and `kableExtra` packages in R, which included conditional formatting to highlight the better-performing model for each stock and validation fold. Summary statistics, including the frequency with which local or global models performed better and the average improvement, were also provided.

4 Data and Preprocessing

This section describes the data sources and the preprocessing steps applied to prepare the datasets for empirical modeling, distinguishing between the procedures used for the global and local modeling strategies. The implementation was done in `R`, utilizing a wide range of packages including `tidymodels`, `modeltime`, `timetk`, `tidyverse`, `lubridate`, `KFAS`, `tseries`, `urca`, and others.

4.1 Data Sources

The empirical analysis is based on three primary categories of data, each serving a complementary purpose in the modeling and evaluation process:

- **Prediction Market Data (Kalshi):** Forecast data were collected from Kalshi, a regulated prediction market platform, covering a wide range of macroeconomic and firm-specific events. These included both quarterly contracts (e.g., Tesla vehicle production, Netflix subscriber growth, Meta daily active users, U.S. GDP growth) and annual contracts (e.g., hurricane counts, SpaceX launches, tech layoffs, Apple car announcements, Ethereum price, measles incidence). Data was provided in CSV format (e.g., `kalshi-chart-data-tesla-24-q1.csv`), with time-stamped forecast values or percentages. Custom parsing functions in `R` were used to standardize and convert these values to a daily frequency suitable for time series modeling.

- **Financial Market Data (Delphia):** Stock return data for a curated selection of U.S. equities were provided by Delphia, a data-driven fintech company. These returns, compiled in the file `tilt_stocks_2024.csv`, include daily observations for prominent stocks such as TSLA, NFLX, META, AAPL, NVDA, MSFT, and others. This dataset served as the primary dependent variable for evaluating the predictive power of Kalshi-based signals on individual company returns.

- **ETF Market Data (Yahoo Finance via `yfinance`):** To investigate whether the relationships observed between Kalshi signals and individual stock returns also extend to broader asset classes, exchange-traded funds (ETFs) were incorporated into the analysis. A Python script leveraging the `yfinance` library was used to query and download historical price and return data for four major ETFs:

- CAC 40 (`CAC.PA`)
- S&P 500 (`SPY`)
- MSCI World Index (`IWDA.AS`)
- Nasdaq 100 (`QQQ`)

These ETFs were chosen to represent a range of geographies and sector exposures, allowing the study to explore whether insights derived from Kalshi signals are consistent across both equity-specific and index-based instruments.

4.2 Common Preprocessing Pipeline

For both the global and local models, the following preprocessing steps were systematically applied:

- **Daily Aggregation:** Raw Kalshi prediction market data came with heterogeneous temporal structures. Some contracts were issued on a quarterly basis (e.g., Tesla Q1–Q4 production), others annually (e.g., total hurricanes in the year), and certain signals were updated multiple times per day. To ensure compatibility with daily financial returns, all prediction signals were aggregated to a daily frequency using a custom function (`process_df_daily`). This function computed the mean of all intraday values for each contract and date, producing one representative prediction value per day per event.
- **Missing Value Imputation:** The handling of missing data differed depending on the data type:
  - *Prediction Market Data:* Between prediction periods (e.g., from Q1 to Q2), some dates were missing due to inactive or unpublished forecasts. To reconstruct a complete daily timeline for each contract, a Kalman filter was applied using a local linear trend state-space model (`SSModel`) from the `KFAS` package. The smoothed state estimates filled in the missing values, ensuring temporal continuity across contract boundaries.
  - *Financial Market Data:* Stock and ETF return data naturally omitted weekends and public holidays, during which financial markets are closed. These are not true data gaps but expected non-trading days. As such, no imputation was performed. The

data was simply filtered to exclude these dates using `na.omit()`, ensuring that models were trained only on valid trading sessions (typically five days per week).

- **Data Integration:** Once the prediction signals and financial return series were processed, the two datasets were merged on the basis of the available trading dates from the financial market data (i.e., stock and ETF returns). This approach ensured temporal alignment and eliminated any potential look-ahead bias, by guaranteeing that only information available at or prior to each trading day was used for forecasting. By anchoring the integration to the financial data timeline, we avoided including future prediction signals that would not have been observable at the time of the actual market movement. The final dataset consisted of: (i) daily asset returns as target variables, (ii) corresponding prediction market signals, and (iii) metadata such as date, asset ticker, and event identifiers.

To reinforce temporal causality and prevent future information from influencing model training or evaluation, the integration pipeline was anchored on financial market timestamps, ensuring that only contemporaneous or past Kalshi signals were matched with asset returns. In particular:

- Only prediction signals published prior to or on the same day as the asset return were used.
- Future data—even if technically available in the full dataset—was never leaked into the training or testing process.
- All model evaluations were conducted using walk-forward cross-validation, with strict temporal separation between training and test sets.

This careful design eliminated the risk of contaminating the model with information from the future and ensured the empirical validity of the forecasting exercise.

4.3 Global Modeling Preprocessing

In the global modeling framework (`global_model.R`), the goal was to identify generalizable relationships between Kalshi forecast signals and financial asset returns by training a single unified model across all stocks. Unlike the individual asset approach, a common model architecture was used to highlight predictive features that are valid across companies, contracts, and time periods.

- **Data Structure:** After preprocessing and integration steps, the forecast signals and financial returns were merged into a longitudinal dataset (long format), covering all tickers. Each row represented a unique combination of trading date and stock ticker, with Kalshi signals as explanatory variables and the daily return as the target variable.

- **Feature Engineering:** Kalshi signals were converted from long format to wide format using the `pivot_wider()` function. Each distinct Kalshi contract (e.g., `TSLA_Q1_prod`, `META_DAU`) became a separate explanatory variable column. This transformation led to a high-dimensional feature space. Missing values—often due to contracts being specific to certain companies—were kept and handled directly by the model.

- **Metadata Addition:** In addition to the forecast signals, categorical variables such as the stock ticker (`ticker`) were included as predictors. These were encoded into dummy variables using `step_dummy()` from the `recipes` package. This allowed the model to learn company-specific effects while maintaining a common modeling structure.

- **Temporal Split and Causal Integrity:** The data were sorted chronologically and then split into training and test sets to ensure that the test periods were distinct and subsequent to the training sets. This method preserved the temporal integrity of the data, preventing any future information from contaminating the training phase. All preprocessing steps—including imputation, encoding, and normalization—were applied only to the training sets, maintaining a strict separation between phases.

- **Handling Missing Values:** The XGBoost algorithm has native handling of missing values, which eliminated the need for additional imputation. Missing values in Kalshi signals—usually due to contracts not relevant for certain companies—were left as `NA` and passed directly to the model. This allowed the model to use the absence of information as a signal in itself.

This global approach enabled the exploitation of inter-company variation while maintaining temporal rigor. By training on a unified dataset covering various stocks and types of contracts, the model aimed to capture robust and generalizable latent structures, thus improving predictive performance and transferability.

4.4 Local Modeling Preprocessing

While the global model adopted a unified architecture across assets, the local modeling approach (`local_model.R`) focused on constructing individualized models for each stock, allowing for asset-specific dynamics to be captured more effectively. This section details the additional preprocessing steps and methodological refinements applied in this localized framework.

- **Asset-Specific Subsetting:** For each asset (e.g., TSLA, NFLX, META), the dataset was filtered to isolate the corresponding return series and its associated Kalshi contracts. This ensured that only relevant signals were considered, reducing noise and dimensionality for each model instance.

- **Stationarity Assessment and Differencing:** As local models were more sensitive to temporal properties of the time series, both the return and prediction signal series were subjected to stationarity diagnostics. A custom function (`check_stationarity`) applied the Augmented Dickey-Fuller (ADF) test recursively, with up to two orders of differencing if required. This ensured that the modeling inputs met the assumptions of stationarity-critical techniques, such as vector autoregression and Granger causality.

- **Dynamic Predictor Selection via Granger Causality:** To identify which Kalshi contracts contained statistically significant predictive information for a given asset, pairwise Granger causality tests were conducted between the differenced prediction signals and the asset's return series. Only predictors passing the causality threshold (typically at a 5% significance level) were retained. This procedure filtered out noisy or irrelevant features, yielding a parsimonious and asset-specific feature set for each local model.

- **Customized Feature Engineering:** Unlike the global model, no dummy variables for tickers or cross-sectional metadata were included, as each local model was inherently tied to a specific stock. Instead, lagged features were created on a per-asset basis using

functions from the `timetk` and `recipes` packages, allowing the model to incorporate short-term memory effects and autoregressive structures.

- **Residual Analysis and Iterative Refinement:** After initial model training (e.g., with XGBoost or linear regression), residual diagnostics were performed. This included checks for autocorrelation (e.g., using ACF plots), heteroscedasticity, and structural breaks. When inadequacies were detected, models were refined iteratively—either by adjusting the lag structure, reconsidering differencing depth, or revising the predictor set.

- **Temporal Cross-Validation:** As with the global model, temporal integrity was strictly enforced. Walk-forward cross-validation was applied individually to each asset-specific dataset, ensuring that the validation periods always followed the training sets in time. Preprocessing steps—including stationarity transformation and causality testing—were nested within each fold to avoid information leakage.

This tailored approach enabled a deeper exploration of asset-specific dynamics and improved modeling accuracy in cases where global signals might dilute or overlook localized patterns. While computationally more intensive, the local strategy allowed for greater interpretability and robustness in the presence of heterogeneous signal relevance across assets.

5 Modeling (Implementation)

This section details the practical implementation of the forecasting models. The primary global model implemented was XGBoost, and local models were ARIMA, both evaluated within a walk-forward validation framework.

5.1 Technical specifications of the global model (XGBoost)

The global forecasting model used in this project is based on the XGBoost algorithm, a powerful gradient boosting technique particularly well-suited for structured/tabular data. The model is trained and evaluated using a time-series cross-validation framework with walk-forward validation to preserve the temporal structure of the data. Below are the key steps of the implementation:

Data Splitting and Cross-Validation: To evaluate the model's generalization ability, we use the `time_series_cv()` function to implement a walk-forward validation strategy. This method ensures that each training set contains only observations prior to the corresponding testing period. We use two non-overlapping folds with the following configuration:

```
# Crée deux splits temporels non superposés avec fenêtre cumulative
splits <- data_tbl %>%
  arrange(date) %>%
  time_series_cv(
    date_var    = date,         # Variable de temps pour organiser les observations
    initial     = "150 days",   # Fenêtre d'initialisation: 150 jours de données
        pour l'entraînement
    assess      = "75 days",    # Fenêtre d'évaluation: 75 jours de données pour le
        test
    skip        = "75 days",    # Décalage de 75 jours entre chaque fold
    cumulative = TRUE,          # Cumulatif: chaque fenêtre d'entraînement inclut
        toutes les données précédentes
    slice_limit = 2             # Limitation à 2 folds pour la validation
  )
```

In this section of the code, we create a walk-forward validation strategy using the `time_series_cv()` function. First, we ensure that the data is ordered by date, which is essential for time-series analysis. The function splits the data into two folds with a training window of 150 days and a testing window of 75 days. The training window is cumulative, meaning each new fold includes all previous data. The `skip` argument is set to 75 days, meaning the training set for the next fold

will start 75 days after the previous fold's test set. Finally, the `slice_limit` argument is used to limit the number of folds to 2 for simplicity.

Feature Engineering: Each training fold is processed through a `recipe` object, which includes the following transformations:

- Conversion of categorical features to dummy variables using `step_dummy()`
- Normalization of numeric predictors to have zero mean and unit variance using `step_normalize()`
- Extraction of time-based features using `step_timeseries_signature()`
- Removal of near-zero variance predictors using `step_zv()`

```
# Pipeline de préparation des données pour XGBoost
rec_obj_xgb <- recipe(value ~ ., data = train_data) %>%
  step_dummy(id) %>% # Conversion des variables catégorielles en variables indicatrices (dummy)
  step_normalize(all_numeric(), -all_outcomes()) %>% # Normalisation des prédicteurs numériques (sauf la variable cible)
  step_timeseries_signature(date) %>% # Extraction des caractéristiques temporelles à partir de la variable "date"
  step_rm(date) %>% # Suppression de la variable "date" car elle n'est plus nécessaire après l'extraction des caractéristiques temporelles
  step_zv(all_predictors()) %>% # Suppression des prédicteurs avec une variance proche de zéro
  step_dummy(all_nominal_predictors(), one_hot = TRUE) # Conversion des prédicteurs nominaux en variables "one-hot"
```

The `recipe` function is used to preprocess the data for the XGBoost model. This includes several important steps. First, categorical variables (such as "id") are converted into dummy variables using `step_dummy()`. Numeric predictors are then normalized to have zero mean and unit variance using `step_normalize()`. Time-based features are extracted with `step_timeseries_signature()` to allow the model to learn seasonal patterns. The `step_rm()` function removes the "date" column after the time-based features are extracted. Next, `step_zv()` removes predictors that have near-zero variance, which are unlikely to contribute to the model. Finally, `step_dummy()` is applied to nominal predictors to perform one-hot encoding, which transforms categorical variables into a format suitable for machine learning algorithms.

Model Training: For each fold, an XGBoost regression model is trained within a `workflow` using the previously defined recipe. This workflow combines the data preprocessing steps with the model training.

```
# Création d un workflow combinant le modèle XGBoost avec la recette
wflw_xgb <- workflow() %>%
  add_model(boost_tree("regression") %>%
  set_engine("xgboost")) %>% # Spécification de l'algorithme XGBoost pour la ré
      gression
  add_recipe(rec_obj_xgb) %>% # Ajout de la recette de prétraitement
  fit(train_data) # Entraînement du modèle sur les données d'entraînement
```

In this part of the code, an XGBoost model is created using the `workflow()` function. This workflow integrates the pre-processing steps from the recipe with the XGBoost model. The model is specified as a regression model by using the `boost_tree()` function and setting the engine to `xgboost`. The `add_recipe()` function adds the preprocessing steps defined earlier, ensuring that the data is properly prepared before fitting the model. Finally, the model is trained using the `fit()` function on the training data.

Model Evaluation: The model is evaluated using the `modeltime` framework. For each fold, the calibrated model is assessed on unseen data and its accuracy is reported both globally and by ticker (i.e., for each time series). RMSE is used as the main metric to assess model performance.

```
# Calibrage du modèle sur les données de test
calib_tbl <- model_tbl %>%
  modeltime_calibrate(new_data = test_data, id = "ticker") # Calibrage sur les
      données de test, par ticker

# Calcul des performances (RMSE, etc.) par ticker (id)
calib_tbl %>%
  modeltime_accuracy(acc_by_id = TRUE) # Calcul des métriques de performance par
      ticker
```

Once the model is trained, it is calibrated on the test data using the `modeltime_calibrate()` function. This function evaluates the model's performance on unseen data, and accuracy metrics (like RMSE) are calculated using the `modeltime_accuracy()` function. The evaluation is done both globally and by each ticker to assess the model's performance on individual time series.

Forecast Error Computation: Forecast errors $e_t = y_t - \hat{y}_t$ are computed and stored for further evaluation, combining both folds. This provides a comprehensive understanding of model performance over time, helping to identify patterns in prediction errors.

Final Forecasting and Refit: After validation, the model is refitted on the full dataset to produce future forecasts. A 30-day future frame is generated for each ticker, and forecasts with confidence intervals are produced.

```
# Réentraînement du modèle sur l'ensemble des données (après validation)
refit_tbl <- calib_tbl_1 %>%
  modeltime_refit(data = data_tbl)  # Réentraînement du modèle sur l'ensemble des
      données disponibles

# Génération des prévisions sur 30 jours pour chaque ticker
forecast_results <- refit_tbl %>%
  modeltime_forecast(
    new_data = future_tbl,  # Données futures pour lesquelles générer les pré
        visions
    actual_data = data_tbl,  # Données réelles historiques pour calculer les
        intervalles de confiance
    conf_by_id = TRUE  # Calcul des intervalles de confiance par ticker
  )
```

After model validation, the final model is refitted using the entire dataset with `modeltime_refit()`. This allows the model to incorporate all available data, including the validation data, before making future predictions. A 30-day future frame is generated for each ticker, and predictions are made using the `modeltime_forecast()` function, which also calculates confidence intervals for the forecasts.

Result Storage: Both RMSE metrics and final forecast outputs are saved as RDS files for reproducibility and further analysis. This ensures that results can be easily accessed and shared for future analysis or reporting.

```
# Sauvegarde des résultats de RMSE et des prévisions
saveRDS(global_rmse_results, "data/rds/global_model_rmse.rds")
# Sauvegarde des résultats RMSE pour chaque ticker
saveRDS(forecast_results, "data/rds/global_model_forecasts.rds")
# Sauvegarde des résultats de prévisions
```

The results, including RMSE metrics and forecast outcomes, are saved in RDS files using the `saveRDS()` function. This ensures that the model's results can be reproduced in the future and shared for further analysis or reporting.

5.2 Technical specifications of the local model (ARIMAX)

To study the relationship between stock returns and prediction markets, we have implemented a local modeling approach using ARIMAX, which is an ARIMA model incorporating external explanatory variables. The approach is structured according to the following steps:

5.2.1 Checking the Stationarity of Stock Time Series

Before conducting any time series analysis, it is essential to ensure that the series being analyzed are stationary. To do this, we apply the Augmented Dickey-Fuller (ADF) unit root test. The following code illustrates the procedure applied to each ticker:

```
check_stationarity <- function(df, ticker) {
  df_clean <- df[!is.na(df$returns), ]
  adf_result <- adf.test(df_clean$returns, alternative = "stationary")

  if (adf_result\$p.value > 0.05) {
    df\$returns <- c(NA, diff(df\$returns, differences = 1))
    df_clean <- df[!is.na(df\$returns), ]
    adf_result_diff1 <- adf.test(df_clean\$returns, alternative = "stationary")

    if (adf_result_diff1\$p.value > 0.05) {
      df\$returns <- c(NA, diff(df\$returns, differences = 1))
    }
  }
  return(df)
}
```

5.2.2 Cross-Correlation Analysis

The objective is to examine temporal relationships between stock return series and prediction market series. To achieve this, we use the cross-correlation function (CCF), which helps detect lagged effects between the two series:

```r
compute_cross_correlation <- function(stock_series, pred_series, max_lag = 20) {
  df_combined <- inner_join(stock_series, pred_series, by = "date", suffix = c("_stock", "_pred"))
  df_clean <- df_combined %>% drop_na(pred_daily_stock, pred_daily_pred)
  ccf_result <- ccf(df_clean$pred_daily_stock, df_clean$pred_daily_pred, lag.max = max_lag, plot = FALSE)

  result_df <- data.frame(
    lag = ccf_result\$lag,
    correlation = ccf_result\$acf,
    stock = unique(stock_series\$ticker)[1],
    prediction_market = unique(pred_series\$id)[1]
  )
  return(result_df)
}
```

5.2.3 Time Splitting: Walk-forward CV

We used a *walk-forward cross-validation* time splitting method to simulate a real forecasting situation. The dataset is divided into successive windows, with a training period of 150 days and a test period of 75 days. This procedure is repeated for two folds:

- **Fold 1:** Training data for the first 150 days, testing on the following 75 days.
- **Fold 2:** Cumulative training data (225 days), testing on the following 75 days.

```r
splits <- df_combined %>%
  arrange(date) %>%
  time_series_cv(
    date_var    = date,
    initial     = "150 days",
    assess      = "75 days",
    skip        = "75 days",
    cumulative = TRUE,
    slice_limit = 2
  )
```

```r
first_split <- splits\$splits[[1]]
second_split <- splits\$splits[[2]]

train_data_1 <- analysis(first_split)
test_data_1  <- assessment(first_split)
train_data_2 <- analysis(second_split)
test_data_2  <- assessment(second_split)
```

5.2.4 Selection of Exogenous Variables

For each stock, we apply a Granger causality test to identify which series from the prediction markets are significantly predictive. Only variables with a p-value below 5% and results indicating that they *cause* returns are retained as exogenous candidates.

```
granger_causality_test <- function(pred_series, stock_series, max_lag = 5) {
  df_combined <- inner_join(pred_series, stock_series, by = "date", suffix = c("_pred", "_stock"))
  df_clean <- df_combined %>% drop_na(pred_daily_pred, pred_daily_stock)
  grangertest(df_clean\$pred_daily_stock ~ df_clean\$pred_daily_pred, order = max_lag)
}
```

5.2.5 Fitting the ARIMAX Model

Based on the results of the Granger test, two scenarios are considered:

- **No significant exogenous variables:** A simple ARIMA model is fitted.
- **At least one exogenous variable:** An ARIMAX model is fitted with the corresponding predictions as explanatory variables.

The exogenous variables are time-aligned with the return series and formatted into a matrix. Additional checks are performed to handle missing values and potential misalignments. In case of an error during fitting, the model defaults to a simple ARIMA model.

```
fit_local_model_fold <- function(ticker_data, granger_results, prediction_markets, fold_number) {
  # Extraction du titre courant
  current_ticker <- unique(ticker_data\$ticker)

  # Extraction des variables exogènes significatives
  exog_vars <- granger_results %>%
    filter(Stock == current_ticker, result == "Granger-causes", p_value < 0.05) %>%
    pull(Prediction_Market)

  ts_data <- ts(ticker_data\$returns, frequency = 365)

  if (length(exog_vars) > 0) {
    exog_matrix <- prediction_markets %>%
      filter(id %in% exog_vars) %>%
```

```
      pivot_wider(names_from = id, values_from = pred_daily) %>%
      arrange(date) %>%
      filter(date %in% ticker_data$date) %>%
      select(-date) %>%
      as.matrix()

    if (any(is.na(exog_matrix))) {
      model <- auto.arima(ts_data)
    } else {
      model <- auto.arima(ts_data, xreg = exog_matrix)
    }
  } else {
    model <- auto.arima(ts_data)
  }

  return(model)
}
```

5.2.6 Evaluation via Recursive Forecasting (Rolling Forecast)

For each fitted model, the performance is evaluated by generating the forecasts recursively to avoid any look-ahead bias:

- Forecasts are made point by point, updating the model with the most recent available data.
- At each iteration, the values of the exogenous variables for the target date are used.

The performance criterion chosen is the root mean square error (RMSE) over the test period.

5.2.7 Residual Analysis

The residuals of the models are analyzed across several dimensions:

- **Normality:** Shapiro-Wilk test.
- **Autocorrelation:** Ljung-Box test.
- **Heteroscedasticity:** Breusch-Pagan test.
- **Basic Statistics:** mean, standard deviation, skewness, and kurtosis.

These tests allow for assessing the quality of the fitted model and the validity of its assumptions.

6 Analysis of Results

7 Discussion

This section compares the results obtained with those of previous studies and outlines the practical and theoretical implications of the study. It also explores potential limitations and alternative interpretations of the findings.

8 Conclusion

This final section summarizes the main findings of the study, discusses its limitations and suggests avenues for future research.

Bibliography

Ah Mand, A., Janor, H., Abdul Rahim, R., & Sarmidi, T. (2023). Herding behavior and stock market conditions. *PSU Research Review*, *7*(2), 105–116. https://doi.org/10.1108/PRR-10-2020-0033

Berg, J. E., Nelson, F. D., & Rietz, T. A. (2008). Prediction market accuracy in the long run. *International Journal of Forecasting*, *24*(2), 285–300. https://doi.org/10.1016/j.ijforecast.2008.03.007

Berg, J. E., Neumann, G. R., & Rietz, T. A. (2009). Searching for Google's Value: Using Prediction Markets to Forecast Market Capitalization Prior to an Initial Public Offering [Publisher: INFORMS]. *Management Science*, *55*(3), 348–361. Retrieved January 14, 2025, from https://www.jstor.org.proxy.bib.uclouvain.be:8888/stable/40539152

Bossaerts, F., Yadav, N., Bossaerts, P., Nash, C., Todd, T., Rudolf, T., Hutchins, R., Ponsonby, A.-L., & Mattingly, K. (2022, September). Price Formation in Field Prediction Markets: The Wisdom in the Crowd [arXiv:2209.08778 [q-fin]]. https://doi.org/10.48550/arXiv.2209.08778

Dai, M., Jia, Y., & Kou, S. (2021). The wisdom of the crowd and prediction markets. *Journal of Econometrics*, *222*(1), 561–578. https://doi.org/10.1016/j.jeconom.2020.07.016

Downey, l. (2024, October). Efficient Market Hypothesis (EMH): Definition and Critique. Retrieved January 15, 2025, from https://www.investopedia.com/terms/e/efficientmarkethypothesis.asp

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, *4*(1), 1–58.

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, *33*(5), 2223–2273.

Hartford, J., Leyton-Brown, K., Ravanbakhsh, S., & Poole, D. (2018). Deep learning for predicting asset returns. *arXiv preprint arXiv:1804.09314*.

Keasey, K., Lambrinoudakis, C., Mascia, D. V., & Zhang, Z. (2024). The impact of social media influencers on the financial market performance of firms. *European Financial Management*, eufm.12513. https://doi.org/10.1111/eufm.12513

Li, Q., Wang, T., Li, P., Liu, L., Gong, Q., & Chen, Y. (2014). The effect of news and public mood on stock movements. *Information Sciences*, *278*, 826–840. https://doi.org/10.1016/j.ins.2014.03.096

Lopez de Prado, M. (2018). *Advances in financial machine learning*. John Wiley & Sons.

Mongrain, P., & Stegmaier, M. (2024). Introduction to Forecasting the 2024 US Elections. *PS: Political Science & Politics*, 1–12. https://doi.org/10.1017/S1049096524001008

Montero-Manso, P., & Hyndman, R. J. (2021). Modelling global versus local time series using hierarchical forecasting models. *International Journal of Forecasting*, *37*(1), 117–134.

Naseer, M. (2016). The Efficient Market Hypothesis: A Critical Review of the Literature. *The IUP Journal of Financial Risk Management*, *Vol. XII*.

Nti, I. K., Adekoya, A. F., & Weyori, B. A. (2020). A systematic review of fundamental and technical analysis of stock market predictions. *Artificial Intelligence Review*, *53*(4), 3007–3057. https://doi.org/10.1007/s10462-019-09754-z

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: An analysis and review. *International journal of forecasting*, *16*(4), 437–450.

Waitz, M., & Mild, A. (2013). Corporate prediction markets: A tool for predicting market shares. *Journal of Business Economics*, *83*(3), 193–212. https://doi.org/10.1007/s11573-012-0645-1

Wolfers, J., & Zitzewitz, E. (2004). Prediction Markets. *Journal of Economic Perspectives*, *18*(2), 107–126. https://doi.org/10.1257/0895330041371321

Appendix

Abstract :

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Louvain School of Management

Place des Doyens, 1 bte L2.01.01, 1348 Louvain-la-Neuve
Boulevard Emile Devreux 6, 6000 Charleroi, Belgique
Chaussée de Binche 151, 7000 Mons, Belgique

www.uclouvain.be/lsm