School of Pure and Applied Sciences
Department of Mathematics & Statistical Sciences

Analyzing the Impact of Cross-Asset Data on Price Prediction

by

Kabo Letsogo
Reg. No: 21001206
BSc Statistics, BIUST, Botswana

A Project Submitted to the School of Pure and Applied Sciences in Partial Fulfillment of the Requirements for the Award of the BSc Statistics of BIUST

Supervisor:

Dr M. Lekgari
Department of Mathematics & Statistical Sciences,
School of Pure and Applied Sciences, BIUST
E-mail: lekgarim@biust.ac.bw

May 29, 2025

1. Signatures

I Kabo Letsogo of student ID 21001206 hereby declare that this project titled 'Analyzing the Impact of Cross-Asset Data on Price Prediction' is my original work. All sources are properly acknowledged.

Student Name: Kabo Letsogo

Signature: _____ Date: _______________

Supervisor Name: Dr Mokaedi Lekgari

Signature: _____ Date: _______________

2. Abstract

Normally, to create a predictive model for a financial asset X, one would fit the model using the historical data of asset X, this study then aims to find out if historical data of other assets was used as well to fit the model, would this significantly improve predictions. Daily data from 2021 to 2024 from 18 financial assets was used. The study uses ElasticNet Regression models both linear and logistic to fit predictive models. Root Mean Squared Error (RSME) and the adjusted coeffecient of determination R^2 were prediction performance metrics for linear regression models. Accuracy, specificity and precision were performance metrics for logistic regression models. For each asset, a model using only its historical data was fitted, we called this a base model, then a model using historical data from all the available assets was fitted, called a full model. Then sign tests were conducted to compare metrics from base models to full models. The tests showed there to be a significant difference between base models and full models which showed that full models generally performed better. It is recommended the research be repeated with data from more assets and at a different timeframe like hourly data.

3. Acknowledgements

I would like to express my deepest appreciation to Dr M. Lekgari for his guidance, support and giving me feedback on how to create and improve this body of work. Beyond him being my supervisor, learning from him for three semesters he is a great role model who shows mastery of the subject he instructs on and goes to great lengths to ensure his pupils understand and excel academically.

Contents

List of Figures

List of Tables

4. Introduction

4.1 Background of the Study

Financial markets are a common target of both traditional and sophisticated modelling techniques in an attempt to predict where the markets will move. In order to make a predictive model for a particular financial asset one usually uses the historical data of that target asset to train or fit the model (Mozaffari & Zhang, 2024). This study challenges the norm by investigating where fitting models with additional data that is not from the target asset to see if the new models with additional information will lead to statistically significant improvements compared to models that use data from the target asset only.

4.2 Research Objectives

The primary objective of this study is to determine whether cross-asset technical indicators enhance prediction accuracy compared to traditional single-asset models. Specifically, the research aims to:

- Develop baseline prediction models using only an asset's own technical indicators
- Construct enhanced models incorporating technical indicators from related assets
- Quantify the improvement in prediction accuracy across different asset classes (FX pairs vs. equities)
- Test if the prediction accuracy results of baseline models are significantly different from the results of enhanced models

The study uses data from 9 FX pairs and 9 US stocks from January 2021 to November 2024. The data was sourced from Yahoo Finance. The research process begins with data scraping, followed by data preparation and feature engineering of technical indicators for each asset. Both linear regression (for continuous price movement prediction) and logistic regression (for directional movement classification) models will be developed and compared.

4.3 Research Questions

This study addresses the following key questions:

1. Does the inclusion of cross-asset technical indicators significantly improve prediction accuracy compared to single-asset models?
2. Is information from correlated assets better at enhancing prediction performance than non correlated assets?
3. Between linear and logistic regression which has better performance with cross asset data?

4.4 Significance of the Study

This research carries important implications for both academic finance and practical trading applications. From a theoretical perspective, it contributes to the ongoing debate about market efficiency by testing whether cross-asset relationships contain predictable patterns not captured by single-asset models. For practitioners, the findings could inform the development of more robust algorithmic trading strategies and portfolio management tools.

The potential benefits extend beyond pure prediction accuracy. Successful cross-asset models could:

1. Reduce overfitting risks by incorporating broader market signals
2. Provide earlier warning signals for regime changes in market conditions
3. Serve as a form of pseudo-portfolio diversification.

4.5 Scope and Limitations

The study focuses exclusively on technical indicators rather than fundamental data (news), analyzing price-based relationships across assets. While this provides a controlled experimental framework, it necessarily excludes macroeconomic factors that influence asset prices. The research is limited to daily frequency data for FX pairs and US equities, which may not capture intraday relationships that could be valuable for higher-frequency trading strategies.

Key limitations include:

- The study period covers only recent market conditions (January 2021 to November 2024)
- Results may not generalize to less liquid assets or emerging markets

- Computational constraints limit the number of assets and indicators that can be simultaneously analyzed

5. Literature Review

In this chapter there were 5 research studies considered as literature review, The summaries below generally follow a format of discussing the aims, methodology, findings, conclusions, and also simillarities & differences between the paper being discussed and this paper.

5.1 Cross-Asset Signals and Time Series Momentum

A research study by (Pitkäjärvi, Suominen, & Vaittinen, 2019) investigated the phenomenon of cross-asset time series momentum in bond and equity markets across 20 developed countries. The authors aim to demonstrate how past bond market returns positively predict future equity returns, while past equity returns negatively predict future bond returns. Building on Moskowitz et al.'s (2012) work on single-asset time series momentum, the paper seeks to construct diversified portfolios that leverage these cross-asset predictive relationships. Additionally, the research explores the economic mechanisms behind these effects, particularly the role of slow-moving capital and its connection to real economic activity. The study ultimately aims to provide both empirical evidence and theoretical explanations for this novel cross-asset momentum phenomenon.

The research by Pitkajarvi et al. utilized monthly bond and equity index returns from 20 developed countries spanning January 1980 to December 2016. Unlike previous studies that relied on futures data, this study uses index returns to enable broader cross-country analysis. The methodology involves constructing both traditional time series momentum (TSMOM) and cross-asset time series momentum (XTSMOM) portfolios with various lookback and holding periods. Statistical analyses include pooled panel regressions, Sharpe ratio comparisons, and vector autoregressions to examine return predictability. The study also incorporates US data on mutual fund flows, margin debt, and economic indicators to investigate the slow-moving capital hypothesis. Robustness checks are performed using alternative specifications and subsamples.

According to Pitkajarvi et al.'s research, there is strong evidence of cross-asset predictability, with past bond returns positively predicting equity returns and past equity returns negatively predicting bond returns. The XTSMOM portfolio yields a Sharpe ratio 45% higher than traditional TSMOM strategies. Analysis of mutual fund flows reveals that capital moves particularly slowly across asset classes, creating persistent return predictability. The study also finds that combined bond and equity momentum regimes

contain valuable information about future economic activity, with simultaneously positive regimes predicting higher industrial production and investment growth. Speculators' trading patterns in futures markets further corroborate the cross-asset momentum effects.

In conclusion, the study states that cross-asset time series momentum represents a significant and persistent market phenomenon that outperforms traditional momentum strategies. The effects are driven by slow-moving capital across partially segmented bond and equity markets, with mutual fund flows playing a particularly important role. The findings suggest that investors can improve portfolio performance by incorporating cross-asset signals into momentum strategies. Moreover, the connection between momentum regimes and future economic activity indicates these patterns reflect fundamental economic changes rather than mere financial market anomalies. The paper contributes to both the academic understanding of asset pricing dynamics and practical portfolio construction techniques.

Both studies use cross-asset prediction. Pitkäjärvi et al. uses bonds and equities while this study uses foreign exchange (FX) pairs and equities. Both studies use regression to quantify improvement, but Pitkäjärvi et al. use pooled panel regressions, while this study applies ElasticNet regularization. This study uses daily price information from 2021 to 2024, while Pitkäjärvi et al. use monthly data over decades from 1980 to 2016. Both papers compare base models to cross-asset models.

5.2 Cross-Asset Speculation in Stock Markets

A research study by (Bernhardt & Taub, 2008) investigated how heterogeneously informed speculators combine private information about multiple stocks with information in prices, while accounting for the impact of their trades on market prices and the inferences of other traders. The authors aim to characterize equilibrium outcomes in this setting, focusing on the correlation structures of prices and order flows, and how these are driven by asset fundamentals and liquidity trade. Additionally, the study explores how speculator profits vary with the distributions of information and liquidity trade across assets and traders.

The research by Bernhardt and Taub developed a multi-asset model of speculative trade where risk-neutral speculators and exogenous noise traders interact in a market with competitive, uninformed market makers. The model extends Admati's (1985) noisy rational expectations framework to incorporate strategic behavior. The authors use an iterative best-response mapping to solve for equilibrium trading strategies and pricing, proving the existence of a unique linear equilibrium. Numerical analyses complement theoretical results, examining symmetric and asymmetric environments.

According to their research, prices are more correlated than asset fundamentals, espe-

cially in integrated markets where market makers use cross-asset order flow information. The covariance structure of prices is driven solely by asset fundamentals, while order flow covariance depends only on liquidity trade. Observable prices significantly reduce speculator profits by intensifying competition, as traders internalize how their orders influence others via price impacts. The model also quantitatively matches empirical findings on the relationship between order flow and return commonality.

In conclusion, the study states that market transparency (observable prices) amplifies competition and alters trading strategies, leading to higher price volatility and stronger cross-asset correlations. The findings provide testable predictions about how modern market structures affect price formation. The paper highlights the importance of strategic interactions in multi-asset markets and offers insights for understanding the factor structure of returns and order flows observed empirically.

Bernhardt and Taub use simulated data to analyze order flows and liquidity, while this study uses real-world data with technical indicators. Both studies use correlation analysis of multiple assets. They use equilibrium analysis, while this study uses regression analysis.

5.3 Financial Multi-Asset Portfolio Risk Prediction Using Convolutional Neural Networks and Image Processing

A research study by (Hu, Lei, Shi, & Li, 2024) aimed to address the limitations of traditional risk assessment methods in multi-asset portfolios by proposing a novel risk prediction model based on Convolutional Neural Networks (CNNs) and image processing techniques. The authors sought to leverage CNNs' feature extraction capabilities to capture complex, nonlinear relationships among assets and improve prediction accuracy under dynamic market conditions. The research also explored the effectiveness of converting financial time-series data into two-dimensional images for CNN processing.

The research by Hu et al. utilized data from stocks, bonds, commodities, and foreign exchange markets spanning 2000-2022. Financial time-series data were preprocessed, normalized, and converted into 32×32 heatmaps for CNN input. The model architecture included five convolutional layers, pooling layers, and fully connected layers, optimized using Bayesian methods. Performance was evaluated against traditional models (e.g., Linear Regression, SVM) using metrics like Root Mean Squared Error (RSME) and R^2, with additional validation under extreme market conditions (e.g., the 2008 financial crisis).

Their CNN model outperformed traditional methods, achieving an RMSE of 0.043 and R^2 of 0.88. It demonstrated superior robustness during market shocks, with RMSE values

of 0.048 (2008 crisis) and 0.052 (2020 pandemic). The model excelled across asset classes, particularly in stocks ($R^2 = 0.90$) and bonds ($R^2 = 0.88$). Statistical tests (t-tests, K-S tests) confirmed the significance of its accuracy improvements. Visualization techniques, such as heatmaps and radar charts, effectively highlighted risk-weight relationships.

In conclusion, the study states that CNNs combined with image processing offer a transformative approach to multi-asset risk prediction, especially in volatile markets. Limitations include reliance on historical data and delayed responsiveness to sudden events. Future directions include integrating real-time systems, expanding datasets, and testing hybrid models (e.g., RNNs, GNNs). The research provides a foundation for advancing AI-driven financial risk management.

Both studies used multi-asset data to compare one type of model to another, both used R^2 and RSME to measure model performance. The study by Hu et al. compared performance of CNN to traditional models like linear regression while this study compared performance of ElasticNet Regression models with different amounts of data.

5.4 Exploring High-Frequency Lead-Lag Effects and Cross-Asset Linkages

A research study by (Buccheri, Corsi, & Peluso, 2020) aimed to investigate high-frequency lead-lag effects and cross-asset linkages in financial markets by introducing a Multi-Asset Lagged Adjustment (MLA) model. They sought to generalize univariate microstructure models to a multivariate framework, capturing lagged price adjustments and cross-asset dependencies. Their goals included developing a unified statistical test for lead-lag correlations, estimating contemporaneous and lagged dependencies separately, and providing an unbiased estimator of the integrated covariance of efficient prices robust to microstructure noise and asynchronous trading.

The research proposed the MLA model, which combines a vector autoregression of order 1 (VAR(1)) process for latent returns with an additive noise term to account for microstructure effects. The model was cast into a state-space representation and estimated using a Kalman-EM algorithm, treating asynchronous trading as a missing data problem. Simulations were conducted to validate the model's robustness under misspecified data-generating processes, including scenarios with stochastic volatility and varying levels of noise. Empirical analysis was performed on high-frequency NYSE transaction data from 2006 to 2014, focusing on 10 liquid stocks and an ETF.

According to Buccheri et al.'s research, the MLA model successfully identified significant lead-lag correlations, with non-diagonal elements in the lagged adjustment matrix indicating cross-asset price formation mechanisms. The estimator outperformed alternatives

like the Hayashi-Yoshida method, avoiding spurious correlations caused by asynchronous trading. Empirical results showed that lead-lag effects were more pronounced during periods of high volatility, suggesting a link to high-frequency trading activity. Notably, less liquid assets sometimes led more liquid ones, contradicting conventional assumptions about liquidity and informativeness.

In conclusion, the study states that the MLA model provides a robust framework for analyzing high-frequency lead-lag effects and cross-asset linkages. It offers advantages over existing methods by handling microstructure noise, asynchronous trading, and time-varying covariances. The findings highlight the importance of cross-asset trading in price discovery and suggest that traditional liquidity-based assumptions about market leadership may need revision. The authors recommend further research to explore the implications of their model for high-frequency trading strategies and market microstructure theory.

Both papers look at how different assets influence each other's price. Buccheri's research focuses on how some assets indicate information in their price movements faster than others while this research paper explores if using data from multiple assets can lead to statistically significant increase in prediction performance.

5.5 VWAP Execution with Signature-Enhanced Transformers: A Multi-Asset Learning Approach

A research study by (Genet, 2025) addresses two critical challenges in Volume Weighted Average Price (VWAP) execution: the inefficiency of asset-specific model training and the difficulty in capturing complex temporal dependencies in market data. Genet proposes a unified neural network architecture that can generalize across multiple assets while incorporating advanced geometric features of price-volume trajectories. The research aims to demonstrate that this globally-fitted model can outperform traditional approaches on both seen and unseen assets, particularly in cryptocurrency markets. The study also seeks to validate the model's practical applicability through real-time trading simulations.

The research by Genet processes input sequences at multiple temporal scales using a transformer-based design with causal attention masking and signature feature integration. Experiments were conducted on hourly cryptocurrency trading data from 80 Binance pairs, split into training and test sets. Four model variants were compared: asset-fitted Dynamic (AFD) VWAP, globally-fitted Dynamic VWAP (GFD), transformer-based GFT, and signature-enhanced GFT-Sig. The models were evaluated using absolute and quadratic VWAP loss metrics against a naive benchmark. Additional real-world validation was performed through deployment in Aplo's trading simulation environment with

four major cryptocurrency pairs.

According to Genet's research, the GFT-Sig model achieved superior performance, reducing absolute VWAP loss by 21.87% and quadratic loss by 35.96% on test assets compared to the benchmark. Globally-fitted models consistently outperformed asset-specific approaches, demonstrating effective cross-asset generalization. Transformer architectures showed particular strength in capturing long-range dependencies, while signature features enhanced robustness to diverse market conditions. Real-time trading experiments confirmed the framework's practical viability, with improvements scaling positively with order duration. Notably, the model showed reduced effectiveness on outlier assets like privacy coin XMR, highlighting limitations in handling highly idiosyncratic markets.

In conclusion, the study states that combining global parameter sharing, transformer architectures, and signature-based features creates a powerful framework for VWAP execution. The approach offers significant operational advantages by eliminating the need for asset-specific models while maintaining or improving execution quality. Practical deployment considerations are addressed through techniques like recursive bin refinement and multi-frequency training. The work contributes substantially to algorithmic trading literature by showing how modern ML techniques can enhance execution strategies. Future directions may include applications to other asset classes and integration of additional market microstructure features.

Both research papers use data from multiple assets however they use cryptocurrency data while this research uses data from stocks and foreign exchange pairs. Genet's research produces a single model that generalizes across many assets while this research paper produces multiple models each for a specific asset. The results of Genet's research are more geared towards optimal trade placement and execution while this paper is geared more towards price prediction.

6. Data and Methodology

6.1 Data Source

The study uses data from 18 assets comprising 9 FX pairs and 9 US stocks. The data is hourly and spans from January 2021 to December 2024, sourced from Yahoo Finance (*Yahoo! Finance*, 2020) using the python package 'yFinance' (Aroussi, 2023). The FX pairs contain 1,042 observations, while the stocks have 1,000 observations due to market closures during US holidays.

Originally, each asset includes 5 columns: Open, High, Low, Close (OHLC), and Volume. The Volume column for FX pairs is typically "null" or 0, as global FX trading volume cannot be accurately tracked due to decentralized trading across banks and currency exchanges.

From these 5 raw columns, the data is processed to generate approximately 40+ explanatory variables and 2 response variables per asset, these are shown in section 3.2.

6.2 Variables of Interest

6.2.1 Response Variables

1. **NextCandleDirectionR (Continuous):** Price difference between current close and next day's close.
2. **NextCandleDirectionC (Binary):** Directional label (1 = next close is greater than current close price, 0 otherwise).

6.2.2 Explanatory Variables

- **Derived Price Metrics:**
  - Weighted Price Average (`weighted`)
  - Open-Close Delta (`ocDelta`)
  - High-Low Delta (`hlDelta`)
  - High-Low to Open-Close Ratio (`hl_oc`)
  - Percent Change (`Percent_Change`)

  - Candle Direction (`CandleDirection`)

- **Trend Indicators:**
  - Simple Moving Average (`SMA`, 14-period)
  - Exponential Moving Average (`EMA`, 14-period)
  - Double Exponential Moving Average (`DEMA`, 14-period)
  - Triple Exponential Moving Average (`TEMA`, 14-period)
  - Rate of Change (`ROC`, 14-period)
  - Average True Range (`ATR`, 14-period)
  - Normalized ATR (`NATR`, 14-period)
  - Bollinger Bands (`BBANDS_UPPER/MIDDLE/LOWER`, 14-period)
- **Momentum & Volatility Indicators:**
  - Aroon Up/Down (`Aroon_Up/Down`, 14-period)
  - Aroon Oscillator (`Aroon_Osc`, 14-period)
  - Average Directional Index (`ADX`, 14-period)
  - Commodity Channel Index (`CCI`, 14-period)
  - Rate of Change Percentage (`ROCP`, 14-period)
  - Williams %R (`WilliamsR`, 14-period)
  - Chaikin Oscillator (`ADOSC`, fast=14, slow=28)
  - Parabolic SAR (`SAR`)
- **Candlestick Patterns:**
  - Hammer (`CDLHAMMER`)
  - Inverted Hammer (`CDLINVERTEDHAMMER`)
  - Hanging Man (`CDLHANGINGMAN`)
  - Shooting Star (`CDLSHOOTINGSTAR`)
  - Engulfing Pattern (`CDLENGULFING`)
  - Harami Pattern (`CDLHARAMI`)
  - Piercing Line (`CDLPIERCING`)
  - Dark Cloud Cover (`CDLDARKCLOUDCOVER`)

  - Morning/Evening Star (`CDLMORNINGSTAR/CDLEVENINGSTAR`)

- **Deviations:**
  - Weighted Price - SMA (`w-SMA`)
  - Weighted Price - Bollinger Upper (`w-UPPER`)
  - Weighted Price - Bollinger Lower (`w-LOWER`)

Python code was used for accessing, downloading the data and generataing the above features using the Technical Analysis Library "TA-Lib" (Fortier, 2023).

6.3 Statistical Methods

6.3.1 Descriptive Statistics

Measures of central tendency such as mean, median and mode are quantities calculated on a dataset of a variable that show the central or typical value for a distribution.

Measures of dispersion such as range, Inter Quartile Range(IQR), variance, skewness and kurtosis are used to quantify the spread of data from a central value (Sanders & Smidt, 1999).

Measures of central tendency and dispersion were used to see how the weighted price is distributed for all the assets.

Correlation quantifies the magnitude and strength of a linear relationship between two variables (Walpole, Myers, Myers, & Ye, 2012).

Correlation was measured between the assets to see if there are any linear relationships between the assets so that later on we could observe if data from correlated assets improved predictions.

6.3.2 Data Cleaning

The data was checked for missing values, and none were found. The data was then tested for multivariate normality using the Henze-Zirkler test, with results indicating rejection of the null hypothesis ($p < 0.05$), confirming non-normal distribution. Given the presence of negative values in some columns (e.g., `ocDelta`, `hlDelta`), neither min-max normalization nor Box-Cox transformation could be uniformly applied. The continuous variables were instead scaled using the **Robust Scaler**, defined as:

$$x_{\text{scaled}} = \frac{x_i - \text{Median}(X)}{\text{IQR}(X)} \tag{6.1}$$

,

where Median(X) is the 50th percentile of feature X and IQR$(X) = Q_3(X) - Q_1(X)$ is the interquartile range (75th - 25th percentile)

Research by (Suthiponpisal & Tancharoen, 2024) showed that the robust scaler is the best data normalization technique for stock market data used for predition. The main advantages of this normalization technique being it is not affected by outliers, it maintains the distribution shape of the original data and it makes data have comparable scales across all features which accelerates regression optimization convergence.

6.3.3 Inferential Statistics

Regression Method: ElasticNet

This project uses financial data for which most features exhibit high multicollinearity. Thus, Ordinary Least Squares (OLS) would not be suitable for modeling because OLS assumes independent predictors (no perfect multicollinearity), and its estimates become unstable when this assumption is violated (Bain & Engelhardt, 1992).

Instead, a modified version of OLS is called ElasticNet Regularized Regression is used. Regularization is a technique used to improve generalization of a model such as by adding constraints to the objective function. ElasticNet is a combination of two regularization methods L1 and L2: (Zou & Hastie, 2005)

1. **L1 least absolute shrinkage and selection operator(LASSO) regularization**, which serves to perform feature selection by forcing weak/noisy coefficients to exactly zero. It has the mathematical form:

$$\min_{\beta} \left(\|y - X\beta\|_2^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right).$$

The solution for each coefficient β_j is determined by:

$$\beta_j = \begin{cases} \rho_j - \lambda & \text{if } \rho_j > \lambda, \\ 0 & \text{if } |\rho_j| \leq \lambda, \\ \rho_j + \lambda & \text{if } \rho_j < -\lambda, \end{cases}$$

where $\rho_j = \langle x_j, y - X_{-j}\beta_{-j} \rangle$ is the partial correlation of x_j with the residuals and λ defines the threshold for sparsity. If $|\rho_j| \leq \lambda$, $\beta_j = 0$. By setting $\beta_j = 0$ the j-th feature/coeficient would then have effectively been removed.

2. **L2 (Ridge) Regression** Rigde Regression also known as Tikhonov regularization, is a way of estimating coefficients of a regression problem where the independent variables are highly correlated (Hilt & Seegrist, 1977). Ridge regression's regularization parameter λ induces proportional shrinkage across all coefficients while ensuring none are driven exactly to zero. It has the mathematical form

$$\min_{\beta} \left(||y - X\beta||_2^2 + \lambda ||\beta||_2^2 \right).$$

The closed-form solution is:

$$\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y.$$

ElasticNet Linear Regression

ElasticNet combines L1 and L2 to have the form

$$\min_{\beta} \left(\|\boldsymbol{y} - \mathbf{X}\beta\|_2^2 + \lambda \left(\alpha \|\boldsymbol{\beta}\|_1 + \frac{(1-\alpha)}{2} \|\boldsymbol{\beta}\|_2^2 \right) \right).$$

The parameter α is specific to ElasticNet and determines the relative contribution of L1 versus L2 penalties, with $\alpha \in [0, 1]$ defining the spectrum between pure L1 ($\alpha = 1$, LASSO) and pure L2 ($\alpha = 0$, Ridge) regularization, where intermediate values $0 < \alpha < 1$ create a hybrid ElasticNet solution. Mathematically, the term $\alpha\|\beta\|_1$ governs the sparsity-inducing L1 effect, while $\frac{1-\alpha}{2}\|\beta\|_2^2$ controls the shrinkage behavior characteristic of L2 regularization, with the $\frac{1}{2}$ factor ensuring consistency with standard Ridge regression when $\alpha = 0$. Together with λ which determines the overall strength of regularization, α specifically modulates the balance between L1 and L2 penalty dominance in the model.

By balancing both methods ElasticNet provides stable feature selection while preserving grouped effects. This combination compensates for the weaknesses of each method since L1 can ignore weak but consistent signals while L2 tends to retain all features including noise. This is critical for our project because we need to Identify the most predictive cross-asset features without overselecting and losing predictors with weak consistent signals.Account for correlated predictors (L2's shrinkage)

Performance Metrics

1. Root Mean Square Error (RMSE): is a metric used to measure the difference between the observed and the predicted values. It is the average squared difference, shown in the formula below. It is in the same units as the variable being predicted (Hyndman & Koehler, 2006).

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

where n is the number of observations, y_i is the actual (observed) value and $\hat{y}_i$ is the predicted value.

2. Adjusted Coefficient of Determination R^2: R^2 is a measure of how well the independent variables explain the variability of the dependent variable. The adjusted R^2 accounts for the number of independent variables in the model. (Walpole et al., 2012) The formulas for the two are shown below:

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}},$$

where $\text{SS}_{\text{res}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ is the Sum of Squared Residuals, $\text{SS}_{\text{tot}} = \sum_{i=1}^{n}(y_i - \bar{y})^2$ is the Total Sum of Squares , y_i is the actual (observed) value, $\hat{y}_i$ is the predicted value, $\bar{y}$ = is the Mean of observed values, n is the number of observations and k is the number of independent variables.

The Adjusted R^2 is calculated as:

$$\text{Adjusted } R^2 = 1 - \left(\frac{(1-R^2)(n-1)}{n-k-1}\right).$$

ElasticNet Logistic Regression

For binary classification of binary response variable for each asset, the objective function combines logistic loss with Elastic Net regularization:

$$\min_{\beta} \underbrace{-\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\sigma(x_i^T\beta)) + (1-y_i)\log(1-\sigma(x_i^T\beta))\right]}_{\text{Logistic Loss}} + \underbrace{\lambda\left(\alpha\|\beta\|_1 + \frac{1-\alpha}{2}\|\beta\|_2^2\right)}_{\text{Elastic Net}} \quad (6.2)$$

.

The logistic regression model uses the sigmoid function $\sigma(z) = 1/(1+e^{-z})$ to transform linear predictions into probabilities, where the regularization strength is controlled by $\lambda = 1/(C \cdot n)$. The ElasticNet mixing parameter $\alpha = 0.5$ provides equal weighting between L1 and L2 penalties when both regularization types are active. The L1 penalty induces feature selection through its characteristic soft-thresholding behavior, which operates on

each coefficient β_j according to the piecewise rule: $\beta_j = \begin{cases} \tilde{\beta}_j - \lambda\alpha & \text{if } \tilde{\beta}_j > \lambda\alpha, \\ 0 & \text{if } |\tilde{\beta}_j| \leq \lambda\alpha, \\ \tilde{\beta}_j + \lambda\alpha & \text{if } \tilde{\beta}_j < -\lambda\alpha, \end{cases}$, where $\tilde{\beta}_j$ represents the unregularized coefficient estimates.

Performance Metrics

1. Accuracy measures the overall correctness of the model's predictions by calculating the proportion of correctly classified instances out of all predictions (Olson & Delen, 2008). The formula is given by:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN},$$

where TP represents True Positives (correctly predicted positive instances), TN represents True Negatives (correctly predicted negative instances), FP represents False Positives (negative instances incorrectly predicted as positive), and FN represents False Negatives (positive instances incorrectly predicted as negative).

2. **Specificity** measures the model's ability to correctly identify negative instances, calculated as the proportion of true negatives out of all actual negative instances. (Yerushalmy, 1947) The formula is:

$$\text{Specificity} = \frac{TN}{TN + FP},$$

3. Precision evaluates the model's ability to correctly identify positive instances out of all predicted positive instances, focusing on minimizing false positives. (Olson & Delen, 2008) The formula is:

$$\text{Precision} = \frac{TP}{TP + FP},$$

Sign Test

The Sign Test is a non-parametric statistical method used to determine if there is a significant difference between paired observations by comparing their medians. In this study, it was employed to compare the performance metrics between base and full models.

- **Null Hypothesis ($\mathbf{H_0}$)**: The median difference between pairs is zero
- **Alternative Hypothesis ($\mathbf{H_1}$)**: The median difference is non-zero

The Sign Test was chosen for its robustness to non-normal distributions and outliers.(Hogg, Mckean, & Craig, 2020).

Spearman's Rank Correlation Test

Spearman's Rank Correlation Test measures the strength and direction of monotonic association between two ranked variables. In this research, it was used to examine the relationship between feature importance and cross-asset correlations.

- **Null Hypothesis(H_0)**: There is no monotonic correlation between the variables ($\rho = 0$)
- **Alternative Hypothesis(H_1)**: There is a monotonic correlation between variables
- **Rejection Criteria**: Reject H_0 if the p-value < 0.05

This non-parametric test was selected because it does not assume linearity or normality in the relationship between variables, making it appropriate for analyzing financial data where these assumptions often fail. It assesses whether highly correlated assets tend to produce more important features in the predictive models (Hogg et al., 2020).

7. Descriptive Analysis

In this chapter, we present the key characteristics of the dataset through descriptive statistics.

The tables below shows the measures of central tendency and dispersion for the assets. All of the symbols shown below are those whose price was denoted in USD terms in order for the values to be comparable across the columns. Some of the FX pairs are not quoted in USD and thus are not in the tables but will be in the next section of the descriptive analysis.

Table 7.1: Descriptive Statistics of USD Quoted FX Pairs and Equities (January 2021–November 2024)

Statistic	Currency Pairs			Equities (USD)			
	EURUSD	GBPUSD	AUDUSD	META	AAPL	AMZN	NFLX
Count	1039.0000	1039.0000	1039.0000	1000.0000	1000.0000	1000.0000	1000.0000
Mean	1.0999	1.2836	0.6926	316.8851	167.1022	149.7926	475.8696
Median	1.0880	1.2713	0.6775	309.2388	164.6338	155.7021	489.0387
Std	0.0579	0.0703	0.0425	133.1461	31.2614	32.3037	170.2546
Var	0.0034	0.0049	0.0018	17727.8949	977.2781	1043.5266	28986.6297
Min	0.9605	1.0709	0.6219	89.0453	116.0438	82.4475	168.4075
Max	1.2318	1.4214	0.7977	630.4701	258.4508	231.0425	931.5250
Range	0.2713	0.3505	0.1758	541.4248	142.4070	148.5950	763.1175
5th %ile	1.0024	1.1691	0.6391	127.3680	124.3523	94.4196	196.8929
95th %ile	1.2112	1.3916	0.7741	569.8273	227.9020	194.9253	729.8416
IQR	0.0651	0.1068	0.0654	166.7535	40.4839	48.1450	263.7875
Skew	0.3600	-0.0488	0.5951	0.4611	0.7045	-0.1745	0.2088
Kurtosis	-0.2678	-0.4646	-0.7898	-0.6349	-0.1056	-0.6783	-0.3900
CV	0.0527	0.0548	0.0614	0.4202	0.1871	0.2157	0.3578

Table 7.2: Descriptive Statistics of Additional Equities (January 2021–November 2024)

Statistic	Equities (USD)				
	GOOGL	JPM	GS	C	AXP
Count	1,000	1,000	1,000	1,000	1,000
Mean	129.95	150.40	356.43	52.46	177.17
Median	130.09	140.51	333.23	52.63	161.82
Std Dev	25.44	34.39	74.36	9.19	42.98
Variance	647.27	1,182.88	5,529.93	84.36	1,847.09
Minimum	84.50	97.05	246.48	36.49	108.89
Maximum	197.04	249.52	603.42	71.87	305.38
Range	112.54	152.48	356.94	35.39	196.49
5th %ile	92.38	107.16	276.94	39.23	133.95
95th %ile	175.51	221.56	514.53	66.85	270.76
IQR	34.42	36.09	66.28	16.90	34.81
Skewness	0.371	1.037	1.478	0.086	1.296
Kurtosis	-0.523	0.334	1.730	-1.289	0.831

Key takeaways from these tables are firstly that the variance of FX pairs is far lower than that of equities such as EURUSD: 0.0034, GB- PUSD: 0.0049 compared to (META: 17,727.90 and NFLX: 28,986.60). Variance can be used as a measure of risk and volatility of an asset. FX pairs are known to have lower volatility and returns compared to equities because the price of currencies are controlled/stabilised by central banks.

Another key takeaway is that most assets have negative kurtosis which says that their distributions have thinner tails than the normal distribution. This implies prices are less likely reach extreme deviations away from the mean when compared to the normal distribution.

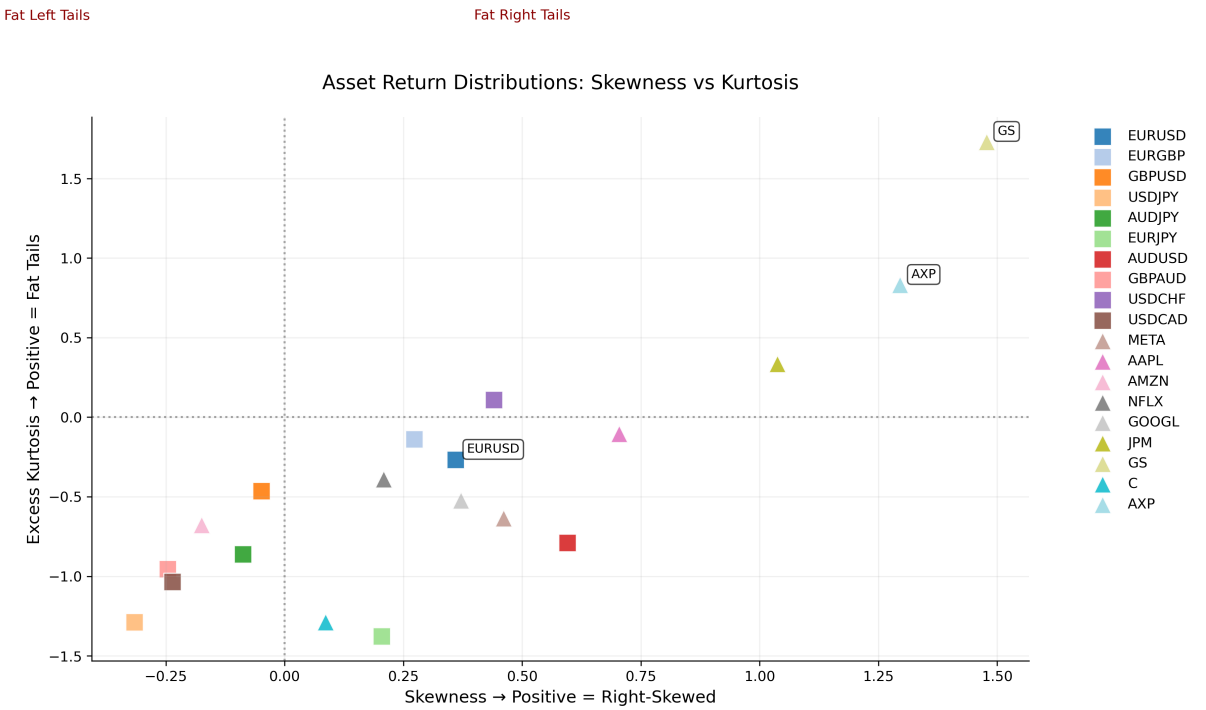

Figure 7.1: **Scatterplot of skewness and kurtosis of assets**: FX pairs with boxes and equities with triangles.

The scatterplot in figure 7.1 shows us that most assets have negative kurtosis as discussed above.

The plot also shows us that most equities are positively skewed which means their tail is fat on the right . This implies that for equities , their extreme values are typically higher than the mean which is due to the equities generally increased in value over the observed time.

7.1 Correlation Analysis

In this section of descriptive statistics we look at the correlations between different assets. This can help us guage the existance of cross asset relationships which we can test our research questions with.

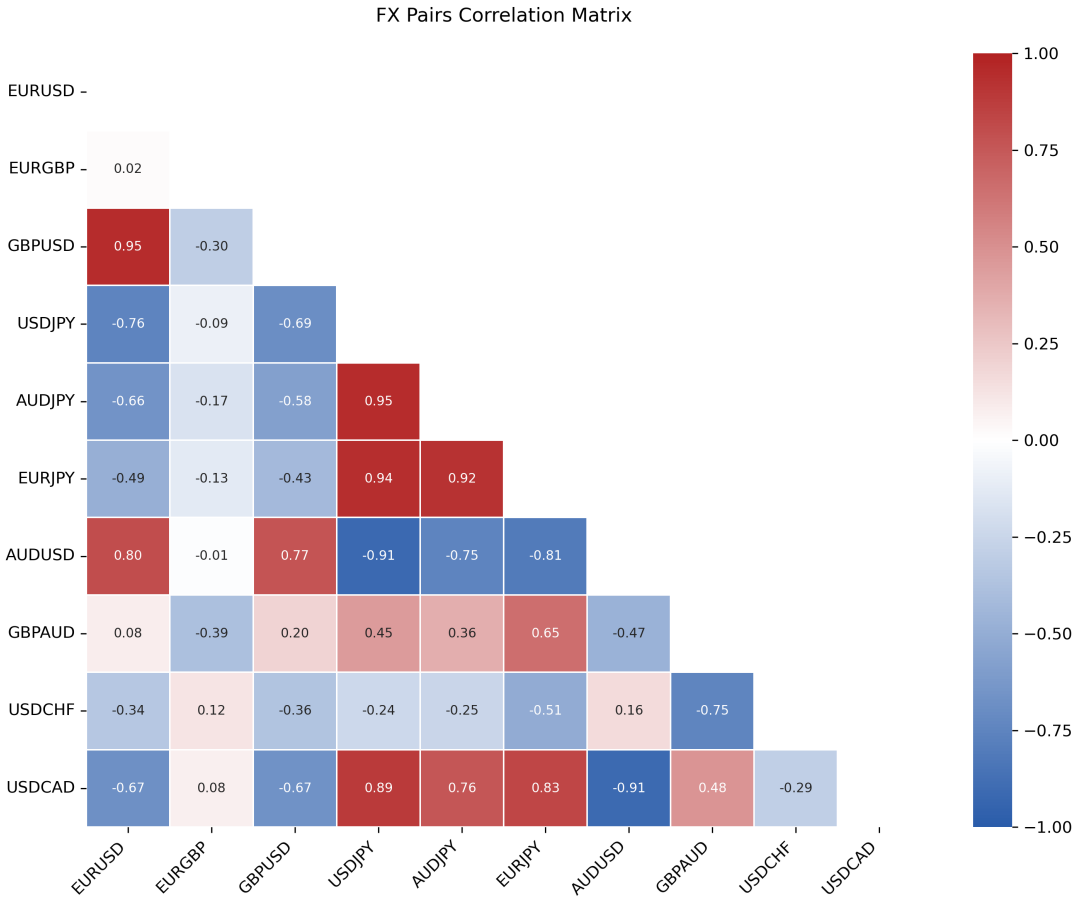

Figure 7.2: FX Pairs Correlation Matrix

FX pair correlations in figure 7.2 show that there is high correlation between 2 FX pairs which share a common currency. This is to be expected naturally because they have a common factor. There are still FX pairs which do not share a common currency but are still correlated such as EURUSD and AUDJPY.

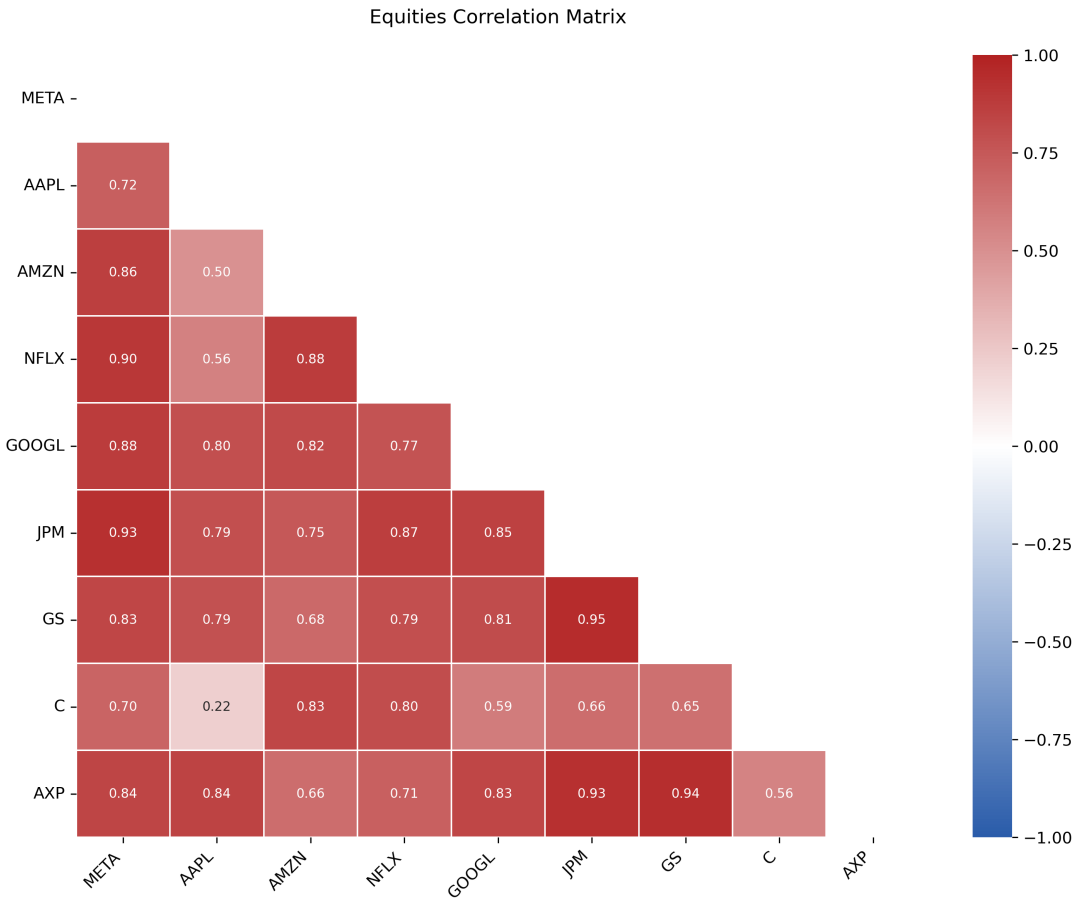

Figure 7.3: Equities Correlation Matrix

The figure 7.3 show that United States stocks are highly positively correlated. Even stocks in different industries such as Facebook-Meta(META) a social media company is highly correlated with a bank JP Morgan(JPM).

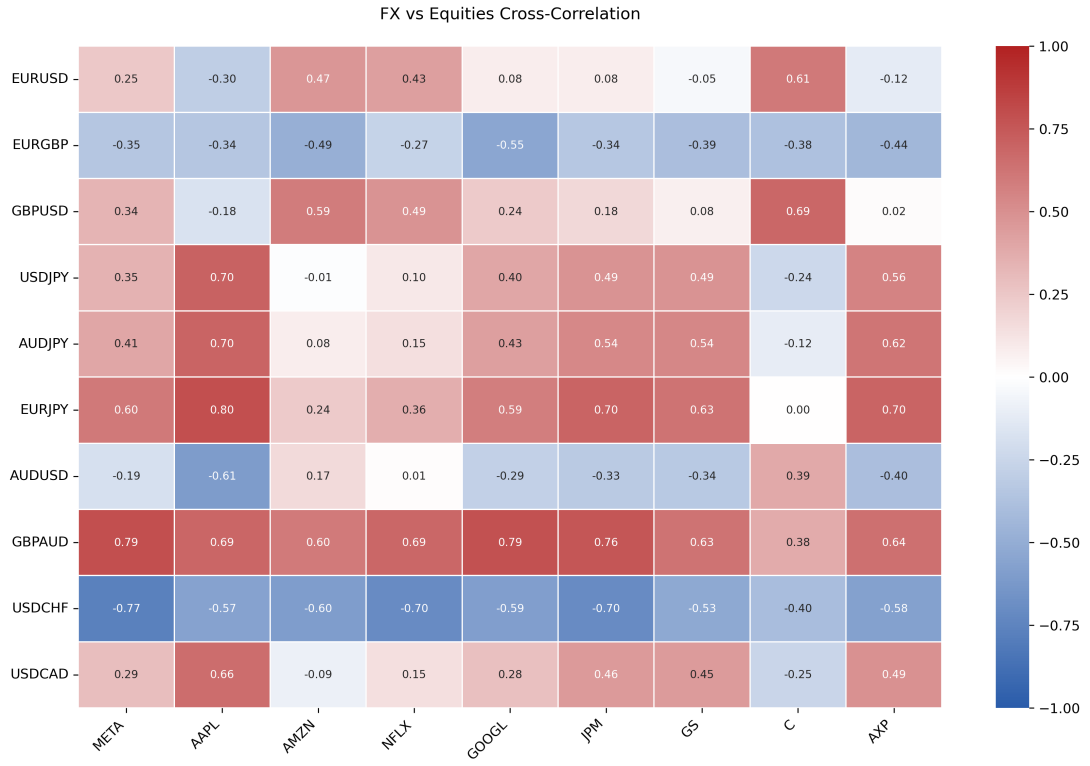

Figure 7.4: FX vs Equities Cross-Correlation

The correlation matrix in figure 7.4 shows are that some FX pairs are correlated with some USA stocks. A noteworthy observation was that GBPAUD and USDCHF are moderate to strongly correlated with all USA stocks positively and negatively respectively.

All in all, the correlation matrices show us that relationships can indeed exist across assets and now we will test if these relationships can aid in predictive modeling in the next chapter.

8. Inferential Analysis

8.1 Regression Model Results

Each asset has two target variables. The first target variable was 'NextCandleDirectionR' which has continuous numeric data upon which we fitted two ElasticNet linear regression models on. One model called the base model which used explanatory variables derived from that asset's data only and another model called a full model which used explanatory variables from its own asset and other assets. The second target variable 'NextCandleDirectionC' is a binary categorical variable. Simillarly, we fitted two ElasticNet logistic regression models on it being a base model and a full model.

The tables below show the performance metrics of the models.

	RMSE		Adj. R^2		AIC	
Asset	Base	Full	Base	Full	Base	Full
EURUSD	0.0040	0.0020	0.4310	0.6070	-11,123.2060	-11,229.9630
EURGBP	0.0040	0.0030	0.4980	0.6160	-11,067.1700	-11,069.3750
GBPUSD	0.0050	0.0030	0.4500	0.6610	-10,560.3190	-10,781.2850
USDJPY	0.6230	0.3520	0.4180	0.5450	-877.3140	-878.9420
AUDJPY	0.4760	0.2640	0.4120	0.6070	-1,414.4180	-1,553.1990
EURJPY	0.6310	0.3580	0.4230	0.5430	-849.3680	-839.4180
AUDUSD	0.0030	0.0020	0.4180	0.6400	-11,320.7770	-11,541.8770
GBPAUD	0.0060	0.0040	0.4320	0.6240	-10,052.9420	-10,198.8000
USDCHF	0.0030	0.0020	0.4720	0.6430	-11,424.6100	-11,558.8930
USDCAD	0.0040	0.0030	0.4330	0.6310	-11,000.2720	-11,165.5390
META	4.8340	3.4860	0.6140	0.6280	3,237.5410	3,465.7120
AAPL	1.3590	0.8570	0.7330	0.7430	697.9700	900.9380
AMZN	1.7110	1.2720	0.6870	0.6960	1,157.8350	1,390.9410
NFLX	6.1010	5.4710	0.7200	0.6960	3,700.7660	3,980.9460
GOOGL	1.3700	1.0320	0.6770	0.7150	713.2010	829.0420
JPM	1.0570	0.7440	0.7790	0.8110	194.4200	298.5630
GS	2.7790	1.9690	0.7660	0.7950	2,223.9230	2,310.7780
C	0.3430	0.2740	0.7830	0.7970	-1,473.0460	-1,502.1690
AXP	1.4440	1.0430	0.7810	0.8160	821.2840	894.7250

Table 8.1: Performance Metrics of Linear Regression Models

Table 8.1 shows us a general trend of improvement from Adjusted. R^2_{base} on base models to Adjusted. R^2_{full} in full models. This shows that the full models were better at explaining the variability of the target variable.

There is also a general decrease in root mean squared error (RSME) from base models to full models which indicates that full models had less prediction errors.

For FX pairs we observed a general decrease in Aikake Information Criterion (AIC) from base to full models which showed us that full models were a better balance of fit and model complexity compared to base models. The opposite was observed for stocks as base models tended to have the lower AIC score.

	Accuracy		Specificity		Precision		AIC	
Asset	Base	Full	Base	Full	Base	Full	Base	Full
EURUSD	0.677	0.792	0.660	0.806	0.684	0.810	-1280.000	-344.000
EURGBP	0.720	0.768	0.704	0.776	0.721	0.779	-1368.000	-88.000
GBPUSD	0.753	0.804	0.754	0.817	0.756	0.814	-1448.000	-378.000
USDJPY	0.713	0.784	0.730	0.778	0.664	0.734	-1360.000	-138.000
AUDJPY	0.689	0.815	0.722	0.819	0.666	0.792	-1358.000	-464.000
EURJPY	0.711	0.789	0.726	0.775	0.686	0.755	-1360.000	-144.000
AUDUSD	0.664	0.816	0.649	0.840	0.660	0.832	-1254.000	-380.000
GBPAUD	0.546	0.774	1.000	0.768	1.000	0.767	-1084.000	-476.000
USDCHF	0.538	0.780	1.000	0.770	1.000	0.759	-1070.000	-110.000
USDCAD	0.554	0.796	0.883	0.795	0.627	0.787	-1096.000	-166.000
META	0.860	0.900	0.948	0.905	0.932	0.898	-1698.000	-368.000
AAPL	0.860	0.905	0.938	0.916	0.916	0.902	-1696.000	-424.000
AMZN	0.892	0.888	0.891	0.903	0.885	0.894	-1718.000	-394.000
NFLX	0.925	0.925	0.928	0.938	0.924	0.933	-1772.000	-504.000
GOOGL	0.862	0.909	0.919	0.911	0.892	0.896	-1700.000	-382.000
JPM	0.897	0.922	0.900	0.926	0.888	0.917	-1714.000	-404.000
GS	0.910	0.908	0.911	0.913	0.906	0.907	-1666.000	-738.000
C	0.924	0.913	0.935	0.919	0.932	0.916	-1166.000	-388.000
AXP	0.848	0.904	0.947	0.915	0.925	0.903	-1670.000	-404.000

Table 8.2: Performance Metrics of Logistic Regression Models

Table 8.2 shows a general improvement in accuracy, specificity and precision from base to full models. This shows that full models are better at predicting the categorical target variable.

All base models had a smaller AIC value than full models, which implies that base models

had a better balance of complexity and fit compared to the full models.

8.2 Sign Tests

From these results we want to know if there is a significant difference in the performance of the base models versus the full models. To that end we, use the Sign test to test the pairs of results for each metric to determine if full models were better, based on that metric. While the Wilcoxon test is also a relevant test, the test assumes symmetric distribution of differences which is not the case with our data as it usually has a general trend one side and so the Sign test was preferred.

8.2.1 Hypothesis Testing

For each performance metric, we conducted a Sign test with the following hypotheses:

- **Null Hypothesis (H_0)**: There is no significant difference between full (cross-asset data) and base (single-asset data) model performance.
- **Alternative Hypothesis (H_1)**: There is a significant difference between full and base model performance.

Rejection Criteria: We reject the null hypothesis at significance level $\alpha = 0.05$ if the p-value < 0.05.

8.2.2 Results

Metric	p-value	Conclusion
Logistic Regression		
Accuracy	0.0075	Reject H_0
Precision	0.1671	Fail to reject H_0
Specificity	0.6476	Fail to reject H_0
Logistic AIC	0.0000	Reject H_0
Linear Regression		
RMSE	0.0000	Reject H_0
Adjusted R^2	0.0001	Reject H_0
Linear AIC	1.0000	Fail to reject H_0

Table 8.3: Sign Test Results Comparing Model Performance.

For the logistic regression results, the main metric being accuracy has been shown to be significantly different between full and base models. Regression results showed generally better accuracy in full models. While the precision and specificity results show no significant difference, since these two metrics are two parts contributing towards accuracy

it could be said that the insignificant increases in precision and specificity of full models added up to a whole significant increase in accuracy. AIC value of the Sign test shows that base models were a significantly better in balancing model complexity and performance.

For the linear regression results both root mean squared error (RMSE) and Adjusted R^2 point to there being a significant difference between base and full models which is to say that full models performed significantly better than base models. The AIC result shows that there was no significant chance in the balance of model complexity and performance between base and full models.

8.3 Spearman Rank Tests

First, to explain source and target assets.When we are using features from Amazon.com Inc (AMZN), Netflix Inc (NFLX) and Alphabet Inc (GOOGL) to help make predictions about EURUSD let us call EURUSD the target asset while AMZN,NFLX,GOOGL are the source assets.

In order to find out if features from source assets that are correlated with the target asset are better predictors, we used Spearman's Rank Correlation tests. This specifically measured the correlation between absolute feature importance (regression coefficient) and the correlation of the source asset of the feature to the target asset.

Each cross asset model has an array of absolute feature importance values for features coming from a variety of source assets. Then for each of those models the correlation test was conducted.

We take the array of the feature importance values,then we order them by their absolute values whilst taking note of the source asset that the features are from. From this array we create a corresponding second array which is the correlation of the source symbol and target asset. Then conduct a Spearman rank correlation test of the two arrays.

8.3.1 Hypothesis Testing

For each asset, we conducted a Spearman rank correlation test with the following hypotheses:

- **Null Hypothesis (H_0)**: There is no monotonic relationship between absolute feature importance (regression coefficient) and the correlation of the source asset of the feature to the target asset (Spearman's $\rho = 0$).
- **Alternative Hypothesis (H_1)**: There is a monotonic relationship between absolute feature importance (regression coefficient) and the correlation of the source asset of the feature to the target asset (Spearman's $\rho \neq 0$).

Rejection Criteria: We reject the null hypothesis at a significance level $\alpha = 0.05$ if the p-value < 0.05.

8.3.2 Results

Asset	Spearman's ρ	p-value	Conclusion
EURUSD	-0.038	0.464	Fail to reject H_0
EURGBP	0.075	0.148	Fail to reject H_0
GBPUSD	0.003	0.950	Fail to reject H_0
USDJPY	0.002	0.966	Fail to reject H_0
AUDJPY	-0.017	0.692	Fail to reject H_0
EURJPY	0.030	0.467	Fail to reject H_0
AUDUSD	0.022	0.681	Fail to reject H_0
GBPAUD	-0.121	0.015	Reject H_0
USDCHF	0.112	0.032	Reject H_0
USDCAD	-0.093	0.068	Fail to reject H_0
META	-0.093	0.049	Reject H_0
AAPL	-0.041	0.334	Fail to reject H_0
AMZN	-0.084	0.086	Fail to reject H_0
NFLX	0.012	0.846	Fail to reject H_0
GOOGL	0.032	0.546	Fail to reject H_0
JPM	0.007	0.881	Fail to reject H_0
GS	-0.004	0.943	Fail to reject H_0
C	0.030	0.647	Fail to reject H_0
AXP	-0.002	0.964	Fail to reject H_0

Table 8.4: Spearman Rank Test Results.

Interpretation

Overall, for most assets, we failed to find statistically significant evidence of a monotonic relationship between absolute feature importance (regression coefficient) and the correlation of the source asset of the feature to the target asset, as indicated by p-values greater than 0.05. This suggests that features from source assets with higher correlation to the target asset will not tend to be better predictors.

9. Discussion

9.1 Discussion on Findings

The study set out to, primarily, find out if including cross asset technical indicators would significantly improve the accuracy of prediction compared to when only using indicators from the target asset. Descriptive analysis showed that most assets are correlated with each other and thus using linear and logistic regression was employed to create the prediction models. It was soon realised that features had high multicollinearity which made sense as all the features were variations of calculations or transformations from the same open, high, low and close (OHLC) columns. This then lead to the use of ElasticNet regression as it is robust against multicollinearity. An added advantage to ElasticNet was that it also performed feature selection. This was imperative as our data had a very high dimensionality.

ElasticNet Regression was done on both models with only the target asset's data (base models) and models with data from all assets (full models). The prediction metrics such as Root Mean Square Error(RSME), Adjusted R^2, Accuracy, Specificity and Precision were compared for base models aganist full models using the Sign test in order to finally answer whether using features from other assets improves predictions. The results from the Sign test showed that the use of features from other assets will generally lead to better predictions.

These results are consistent with findings by (Pitkäjärvi et al., 2019) that cross asset data improved prediction performance. The implications of their research is that cross asset data could be used to predict long term macroeconomic trends while this study's results show that cross asset data could be used for short term predictions also.

In order to find out if features from source assets that have high correlation with the target asset lead to better predictions than features with lower correlation, Spearman's Rank Correlation Test was used. The general outcome of the test was that using features from source assets that have high correlation with the target asset did not necessarily mean that those features would be better predictors than assets from sources with lower correlation with target asset.

The research also aimed to find out where the inclusion of features from other symbols would benefit either linear or logistic regression more. Linear regression takes advantage of cross asset information better. This is shown by the Sign test indicating that for linear

regression's prediction metrics both of them showed that there was a significant difference between base and full models, the difference was with the full models showing better prediction results. This is in contrast to the prediction metrics of logistic regression where only accuracy showed significant difference between base and full models while specificity and precision did not. This however is not necessarily a bad thing as both base and full models generally had very high precision and specificity with little room for differentiation.

9.2 Improving this Study

This study trained models on all the data available and then tested the models on that same data. This could be improved by doing a train test split of the data where some portion of the data is reserved for training / fitting the model while another portion is for testing the model. These splits also help reduce overfitting and can be used to see how well model performs on data it has not 'seen' before (Tan, Yang, Wu, Chen, & Zhao, 2021). A model that performs well when having been tested this way is deemed more reliable and can be used in production level environments where the model predictions are actually used to place trades with real money.

This research was limited to only comparing which model between linear and logistic regression performed better with cross asset data. This research can be expanded to different models such as Support Vector Machines (SVM) , K-Nearest Neighbours (KNN), Decision Trees, Machine Learning (ML) and Reinforcement learning. (Hu et al., 2024) showed that convolutional neural networks (CNNs) which are a type of machine learning outperformed linear regression in prediction accuracy of financial assets. While they only showed CNNs, this research can be expanded further to other types of models such as comparing if CNNs are better than Long Short Term Memory (LSTM) models which are more specialized for timeseries data.

The study was limited to 9 stocks and 9 FX pairs, this was reduced from 16 pairs and 12 stocks because of computer power limitations. There may be more hidden interactions or predictive power gains that could be unleashed if more assets from various classes were used such as oil, gold, silver prices or even including stocks and indices from various countries like China, Europe and Japan. Globalization has linked economies and financial markets around the world and thus it should be plausible to test if data from across the globe improves financial price prediction.

The tests in this research can be repeated at lower timeframes such as 1 minute or 1 hour data which should offer more observations and hence increase reliability of test results. since results from (Pitkäjärvi et al., 2019) suggest that the predictive power of cross asset

information is useful in making long term predictions

Research by (Genet, 2025) showed the creation of one model which worked for multiple financial assets whereas this research produced a model for each symbol. This research could be improved by creating a single model that can be used for predictions for all the symbols.

9.3 Challenges to the Study

The study incurred no true research challenges, the inconviniences that arose are an expected part of the research process. Some of these inconveniences include:

- Long model optimization times. The linear regression models took one hour to be trained while the logistic regression models took 9 hours to be trained. This was exhacebated by constant debugging, for example if a certain column was not calculated correctly or a new explanatory variable had to be added, the optimization would have to be repeated constantly.
- Limited reliable data sources. Ideally the research would have used hourly data over 5 years but that size was too much for Yahoo Finance to provide, another possible data provider MetaQuotes had a history of data manipulation (News, 2022) which made their data less reliable.

  Other reliable sources of data would have cost about BWP2700 to get the data needed but that data would also need cloud computing to optimize it which would be another cost. The budget of BWP3000 would not allow for me to both buy the data and pay for cloud computing to optimize it faster.

10. Conclusion

In order to find out if using features from other assets improves predictions; results from base models were compared to full models, the comparison showed full models performed better in both linear and logistic regression. This shows that including features from other assets improved predictions. It was also found out that using features from source assets that have high correlation with the target asset did not necessarily mean that those features would be better predictors than assets from sources with lower correlation with target asset. Linear regression had better performance with cross asset data than logisitic regression as more linear regression metrics showed a statistically significant improvement whereas logistic regression had only one metric that did.

It is recommended for entities that want to act on or improve this research to include train test splits in their evaluation process. Such entities could also incorparate different models such as machine learning models, repeat the research with more assets over a greater duration of time with higher granularity such as one hour time-frame data

References

Aroussi, R. (2023). *yfinance: Yahoo! finance market data downloader.* Retrieved from https://github.com/ranaroussi/yfinance

Bain, L. J., & Engelhardt, M. (1992). *Introduction to probability and mathematical statistics.* Thomson Learning Duxbury.

Bernhardt, D., & Taub, B. (2008, September). Cross-asset speculation in stock markets. *The Journal of Finance*, *63*(5), 2385–2427. doi: 10.1111/j.1540-6261.2008.01400.x

Buccheri, G., Corsi, F., & Peluso, S. (2020, January). High-frequency lead-lag effects and cross-asset linkages: A multi-asset lagged adjustment model. *Journal of Business & Economic Statistics*, 1–22. doi: 10.1080/07350015.2019.1697699

Fortier, M. (2023). *Ta-lib: Technical analysis library.* Retrieved from http://ta-lib.org/

Genet, R. (2025, March). Vwap execution with signature-enhanced transformers: A multi-asset learning approach. *ResearchGate*. Retrieved from https://www.researchgate.net/publication/389581238_VWAP_Execution_with_Signature-Enhanced_Transformers_A_Multi-Asset_Learning_Approach doi: 10.48550/arXiv.2503.02680

Hilt, D. E., & Seegrist, D. W. (1977, Jan). Ridge, a computer program for calculating ridge regression estimates.
doi: https://doi.org/10.5962/bhl.title.68934

Hogg, R. V., Mckean, J. W., & Craig, A. T. (2020). *Introduction to mathematical statistics.* Upper Saddle River: Pearson.

Hu, Z., Lei, F., Shi, G., & Li, Z. (2024, December). Research on financial multi-asset portfolio risk prediction model based on convolutional neural networks and image processing. *International Journal of Innovative Research in Engineering and Management*, *11*(6), 1–127. Retrieved from https://ijirem.org/view_abstract.php?title=Research-on-Financial-Multi-Asset-Portfolio-Risk-Prediction-Model-Based-on-Convolutional-Neural-Networks-and-Image-Processing&year=2024&vol=11&primary=QVJULTE4NDY= doi: 10.55524/ijirem.2024.11.6.13

Hyndman, R. J., & Koehler, A. B. (2006, Oct). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679–688. doi: https://doi.org/10.1016/j.ijforecast.2006.03.001

Mozaffari, L., & Zhang, J. (2024, May). Predictive modeling of stock prices using transformer model. *2024 9th International Conference on Machine Learning Technologies*

(ICMLT), *23*, 41–48. doi: https://doi.org/10.1145/3674029.3674037

News, F. (2022, Sep). *The metatrader case and why it is important to take a closer look! — fintelegram news.* Retrieved from https://fintelegram.com/the-metatrader-case-and-why-it-is-important-to-take-a-closer-look/

Olson, D. L., & Delen, D. (2008). *Advanced data mining techniques*. Berlin: Springer.

Pitkäjärvi, A., Suominen, M., & Vaittinen, L. (2019). Cross-asset signals and time series momentum. *Journal of Financial Economics*, *134*, 428–451. doi: 10.1016/j.jfineco.2019.02.011

Sanders, D. R., & Smidt, R. (1999). *Statistics : A first course* (8th ed.). London: Mcgraw-Hill.

Suthiponpisal, V., & Tancharoen, D. (2024, Nov). A comparative evaluation of noise reduction versus data normalization techniques in stock market prediction using transformer models. *2024 8th International Conference on Information Technology (InCIT)*, 775–780. doi: https://doi.org/10.1109/incit63192.2024.10810587

Tan, J., Yang, J., Wu, S., Chen, G., & Zhao, J. (2021, Jun). A critical look at the current train/test split in machine learning. *arXiv:2106.04525 [cs]*. Retrieved from https://arxiv.org/abs/2106.04525

Walpole, R. E., Myers, R. H., Myers, S. L., & Ye, K. (2012). *Probability statistics for engineers scientists*. Boston: Prentice Hall.

Yahoo! finance. (2020). Retrieved from https://finance.yahoo.com (Historical market data)

Yerushalmy, J. (1947). Statistical problems in assessing methods of medical diagnosis, with special reference to x-ray techniques. *Public Health Reports (1896-1970)*, *62*(40), 1432–1449. Retrieved from https://www.jstor.org/stable/4586294?Search=yes&resultItemClick=true&searchText=jacob+yerushalmy&searchUri=%2Faction%2FdoBasicSearch%3FQuery%3Djacob%2Byerushalmy&ab_segments=0%2Fbasic_search_gsv2%2Fcontrol&refreqid=fastly-default%3A635899aa2f4399b24944f966be04299a&seq=1#metadata_info_tab_contents doi: https://doi.org/10.2307/4586294

Zou, H., & Hastie, T. (2005, Apr). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(2), 301–320. doi: https://doi.org/10.1111/j.1467-9868.2005.00503.x

11. Appendix 1 - Python Code Used

```
#Imports
import pandas as pd
import numpy as np
import os
import datetime
from datetime import timedelta
import yfinance as yf
import pyarrow.parquet as pq
import pyarrow as pa
from tqdm import tqdm
import json
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.model_selection import train_test_split

from sklearn.decomposition import PCA, KernelPCA, IncrementalPCA, FastICA
from sklearn.manifold import Isomap, LocallyLinearEmbedding, MDS, TSNE
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.stats.outliers_influence import variance_inflation_factor

import talib as ta
#I used a on wheels install of talib specific to
#the current python version i am using.
#  Do so to if necessary in future
# pip install talib will typically install the
#32bit version of the package which is
#incompatible with
# most 64 bit computers. So expect an error. The
#website below should have a command/release for
# whatever python version you have. NOTE you
```

```
#must tailor the command to your version
# https://github.com/cgohlke/talib-build?#tab=readme-ov-file

# Set pandas display options
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)

# Define asset lists
FX = [
    "EURUSD=X", "EURGBP=X", "GBPUSD=X", "USDJPY=X",
    "AUDJPY=X", "EURJPY=X", "AUDUSD=X", "GBPAUD=X",
    "USDCHF=X", "USDCAD=X"
]

STOCKS = [
    "META", "AAPL", "AMZN", "NFLX", "GOOGL",
    "JPM", "GS", "C", "AXP"
]

SYMBOLS = FX + STOCKS

# Parameters for data download
START_DATE = "2020-12-17"
END_DATE = "2025-01-01"
INTERVAL = "1d"

# Directory to save parquet files
SAVE_DIR = "new_yahoo_finance_data"
os.makedirs(SAVE_DIR, exist_ok=True)

#Data Download and pre processing
def process_and_save_data(ticker):
    print(f"Fetching data for {ticker}...")
    df = yf.download(ticker, start=START_DATE, end=END_DATE, interval=INTERVAL)

    if df.empty:
        print(f"Data for {ticker} is empty. Skipping...")
        return
```

```
# Flattens the data, Keeps only Price Type
df.columns = [col[0] for col in df.columns]

#Change index from date to numbers and retain a date column
df.reset_index(inplace=True)
df["Date"] = pd.to_datetime(df["Date"], utc=True)

# Create derived columns
df["weighted"] = (df['Open'] + df['High'] + df['Low'] + df['Close']) / 4
df['ocDelta'] = df['Close'] - df['Open'] #open close delta
df['hlDelta'] = df['High'] - df['Low']
df['hl_oc'] = df['hlDelta'] / df['ocDelta']
df['Percent_Change'] = (df['ocDelta']/df["Open"])*100
df['CandleDirection'] = (df['Close'] > df['Open']).astype(int)

# Replace infinite or NaN values
df.replace([np.inf, -np.inf], 0, inplace=True)

## FEATURE CREATION OF INDICATORS AND FORMATIONS

df[f'NextCandleDirectionR'] = df['Close'].shift(1) - df['Close']
df['NextCandleDirectionC'] = (df['NextCandleDirectionR'] > 0).astype(int)

#Averages
df[f"SMA"] = ta.SMA(df['weighted'], timeperiod=14)
df[f"EMA"] = ta.EMA(df['weighted'], timeperiod=14)
df[f"DEMA"] = ta.DEMA(df['weighted'], timeperiod=14)
df[f"TEMA"] = ta.TEMA(df['weighted'], timeperiod=14)
df[f"ROC"] = ta.ROC(df['weighted'], timeperiod=14)
df[f"ATR"] = ta.ATR(df['High'], df['Low'], df['Close'], timeperiod=14)
df[f"NATR"] = ta.NATR(df['High'], df['Low'], df['Close'], timeperiod=14)
upper, middle, lower = ta.BBANDS(df['weighted'], timeperiod=14)
df[f"BBANDS_UPPER"] = upper
df[f"BBANDS_MIDDLE"] = middle
df[f"BBANDS_LOWER"] = lower
#Deviations
df["w_SMA"] = df["weighted"]-df[f"SMA"]
df["w_UPPER"] = df["weighted"]-df[f"BBANDS_UPPER"]
```

```
df["w_LOWER"] = df["weighted"]-df[f"BBANDS_LOWER"]
df[f"Aroon_Up"], df[f"Aroon_Down"] = ta.AROON(df['High'], df['Low'], timeperiod=14
df[f"Aroon_Osc"] = ta.AROONOSC(df['High'], df['Low'], timeperiod=14)
df[f"ADX"] = ta.ADX(df['High'], df['Low'], df['Close'], timeperiod=14)
df[f"CCI"] = ta.CCI(df['High'], df['Low'], df['Close'], timeperiod=14)
df[f"ROCP"] = ta.ROCP(df['weighted'], timeperiod=14)
df[f"WilliamsR"] = ta.WILLR(df['High'], df['Low'], df['Close'],
timeperiod=14)
df[f"ADOSC"] = ta.ADOSC(df['High'], df['Low'], df['Close'], df['Volume'], fastperi
df['SAR'] = ta.SAR(df['High'].values, df['Low'].values)

df = df.replace([np.inf, -np.inf,np.nan], 0)

df.drop(df.head(14).index,inplace=True)
df.drop(df.tail(1).index,inplace=True)

df.reset_index(inplace=True, drop=True)
df = df.replace([np.inf, -np.inf,np.nan], 0)
#Patterns
# Hammer
df["CDLHAMMER"] = ta.CDLHAMMER(df["Open"], df["High"], df["Low"],
df["Close"])

# Inverted Hammer
df["CDLINVERTEDHAMMER"] = ta.CDLINVERTEDHAMMER(df["Open"], df["High"],
df["Low"], df["Close"])

# Hanging Man
df["CDLHANGINGMAN"] = ta.CDLHANGINGMAN(df["Open"], df["High"], df["Low"],
df["Close"])

# Shooting Star
df["CDLSHOOTINGSTAR"] = ta.CDLSHOOTINGSTAR(df["Open"], df["High"],
df["Low"], df["Close"])

# Engulfing Pattern
df["CDLENGULFING"] = ta.CDLENGULFING(df["Open"], df["High"], df["Low"],
df["Close"])
```

```
    # Harami Pattern
    df["CDLHARAMI"] = ta.CDLHARAMI(df["Open"], df["High"], df["Low"],
    df["Close"])

    # Piercing Line
    df["CDLPIERCING"] = ta.CDLPIERCING(df["Open"], df["High"], df["Low"],
    df["Close"])

    # Dark Cloud Cover
    df["CDLDARKCLOUDCOVER"] = ta.CDLDARKCLOUDCOVER(df["Open"], df["High"],
    df["Low"], df["Close"], penetration=0)

    # Morning Star
    df["CDLMORNINGSTAR"] = ta.CDLMORNINGSTAR(df["Open"], df["High"],
    df["Low"], df["Close"], penetration=0)

    # Evening Star
    df["CDLEVENINGSTAR"] = ta.CDLEVENINGSTAR(df["Open"], df["High"],
    df["Low"], df["Close"], penetration=0)



    # Remove '=X' from ticker symbol for cleaner column names
    clean_ticker = ticker.replace("=X", "")

    # Rename columns to include ticker name
    df.columns = [f"{col}_{clean_ticker}" if col != "Date" else "Date" for col
    in df.columns]

    # Save as Parquet
    file_path = os.path.join(SAVE_DIR, f"{clean_ticker}.parquet")
    df.to_parquet(file_path, engine="pyarrow")
    print(f"Saved {ticker} as {file_path}")

# Download and process all assets
for asset in tqdm(FX + STOCKS, desc="Downloading & Processing Data"):
    process_and_save_data(asset)

#Merging data
```

```
# Load FX data
fx_dataframes = []
for fx_ticker in FX:
    clean_ticker = fx_ticker.replace("=X", "")
    file_path = os.path.join(SAVE_DIR, f"{clean_ticker}.parquet")
    df = pd.read_parquet(file_path)
    fx_dataframes.append(df)

# Load Stock data
stock_dataframes = []
for stock_ticker in STOCKS:
    file_path = os.path.join(SAVE_DIR, f"{stock_ticker}.parquet")
    df = pd.read_parquet(file_path)
    stock_dataframes.append(df)

# Function to merge dataframes on the 'Date' column
def merge_dataframes(dataframes):
    merged_df = dataframes[0]  # Start with the first dataframe
    for df in dataframes[1:]:
        merged_df = pd.merge(merged_df, df, on="Date", how="inner")
    return merged_df

# Merge FX dataframes
fx_merged_df = merge_dataframes(fx_dataframes)

# Merge Stock dataframes
stock_merged_df = merge_dataframes(stock_dataframes)

# Merge the FX and Stock dataframes on the 'Date' column
combined_df = pd.merge(fx_merged_df, stock_merged_df, on="Date", how="inner")

# Save the merged dataframes to Parquet files (optional)
fx_merged_df.to_parquet(os.path.join(SAVE_DIR, "fx_merged.parquet"),
engine="pyarrow")
stock_merged_df.to_parquet(os.path.join(SAVE_DIR, "stock_merged.parquet"),
engine="pyarrow")
combined_df.to_parquet(os.path.join(SAVE_DIR, "combined_fx_stock.parquet"),
engine="pyarrow")
```

```
# import merged data
fx_df = pd.read_parquet(os.path.join(SAVE_DIR, "fx_merged.parquet"))
stock_df = pd.read_parquet(os.path.join(SAVE_DIR, "stock_merged.parquet"))
combined_df = pd.read_parquet(os.path.join(SAVE_DIR,
"combined_fx_stock.parquet"))

def FilterColumn(partial: list,df: (pd.core.frame.DataFrame),compliment: bool=False):
  """
  Provides filtered list of columns

  Uses list of keywords provided to filter columns of the df provided.
  If compliment is on, the filter excludes the columns having the input
  list of keywords.

  Parameters
  ----------
  partial (list):list containing keywords/phrases/sections of column names
  df  (pandas.core.frame.DataFrame): dataframe in which columns are to be
  filtered
  compliment (bool):False to include the columns containing the keywords, True
  to exclude such columns
  """

  hasIt = []
  hazIt = list(df.columns)

  for p in partial:
    for column in df.columns:
      if(compliment==False):
        if(column.find(p)!=-1):
          hasIt.append(column)

      if(compliment==True):
        if(column.find(p)!=-1):
          hazIt.remove(column)

  if(compliment==False):
    return hasIt
  if(compliment==True):
```

```
    return hazIt

# Check for missing values
missing_counts = combined_df.isnull().sum()
columns_with_missing = missing_counts[missing_counts > 0].index.tolist()
rows_with_missing = combined_df[combined_df.isnull().any(axis=1)].shape[0]

print(f"Columns with missing values: {columns_with_missing}")
print(f"Total rows with missing data: {rows_with_missing}/{len(combined_df)}")

def weighted_price_stats(df, assets):
    """
    Generate descriptive statistics for weighted price across assets.

    Parameters:
    - df: Merged DataFrame containing all assets
    - assets: List of asset symbols (e.g., ['EURUSD', 'AAPL'])

    Returns:
    - DataFrame with assets as index and statistics as columns
    """
    # Initialize stats dictionary
    stats_list = []

    # Calculate statistics for each asset
    for asset in assets:
        col_name = f'weighted_{asset}'
        if col_name not in df.columns:
            continue

        series = df[col_name].dropna()
        if len(series) == 0:
            continue

        stats = {
            'Asset': asset,
            'Mean': series.mean(),
            'Median': series.median(),
            'Min': series.min(),
```

```
            'Max': series.max(),
            'Std': series.std(),
            'Var': series.var(),
            'IQR': series.quantile(0.75) - series.quantile(0.25),
            'Skew': series.skew(),
            'Kurtosis': series.kurtosis(),
            'Count': len(series),
            '5th %ile': series.quantile(0.05),
            '95th %ile': series.quantile(0.95),
            'Range': series.max() - series.min(),
            'CV': series.std() / series.mean()  # Coefficient of variation
        }
        stats_list.append(stats)

    # Create and format DataFrame
    stats_df = pd.DataFrame(stats_list)
    stats_df.set_index('Asset', inplace=True)

    # Reorder columns logically
    col_order = [
        'Count', 'Mean', 'Median', 'Std', 'Var', 'Min', 'Max', 'Range',
        '5th %ile', '95th %ile', 'IQR', 'Skew', 'Kurtosis', 'CV'
    ]
    return stats_df[col_order]

# Example usage:
fx_symbols = [x.replace("=X", "") for x in FX]
stock_symbols = STOCKS

# Get stats for FX and Stocks separately
fx_stats = weighted_price_stats(fx_df, fx_symbols)
stock_stats = weighted_price_stats(stock_df, stock_symbols)

# Combine and add asset type marker
fx_stats['Type'] = 'FX'
stock_stats['Type'] = 'Stock'
all_stats = pd.concat([fx_stats, stock_stats])

# Format numeric columns
```

```
float_cols = all_stats.select_dtypes(include=[float]).columns
all_stats[float_cols] = all_stats[float_cols].round(4)

# Display
print("Weighted Price Descriptive Statistics")
print("="*60)
display(all_stats.sort_values('Type'))

def plot_skew_kurt(stats_df, save_path='figures/skew_kurt_plot.png'):
    """
    Enhanced scatterplot of skewness vs kurtosis with:
    - Boxes () for FX pairs
    - Triangles () for equities
    - Unique colors per asset
    - Professional formatting
    - Saves as high-res PNG
    """
    # Create directory if needed
    os.makedirs(os.path.dirname(save_path), exist_ok=True)

    plt.figure(figsize=(12, 7), dpi=300)  # High DPI for quality

    # Create style mappings
    marker_map = {'FX': 's', 'Stock': '^'}  #  for FX,  for stocks
    unique_assets = stats_df['Asset'].unique()
    colors = plt.cm.tab20(np.linspace(0, 1, len(unique_assets)))

    # Plot each asset
    for i, asset in enumerate(unique_assets):
        asset_data = stats_df[stats_df['Asset'] == asset]
        plt.scatter(
            x=asset_data['Skew'],
            y=asset_data['Kurtosis'],
            s=150,  # Marker size
            marker=marker_map[asset_data['Type'].iloc[0]],
            color=colors[i],
            edgecolor='white',
            linewidth=0.8,
            alpha=0.9,
```

```
            label=asset
        )

    # Reference lines
    plt.axvline(0, color='gray', linestyle=':', alpha=0.7)
    plt.axhline(0, color='gray', linestyle=':', alpha=0.7)

    # Annotations
    plt.text(0.5, 2.5, "Fat Right Tails", ha='center', fontsize=10,
    color='darkred')
    plt.text(-0.5, 2.5, "Fat Left Tails", ha='center', fontsize=10,
    color='darkred')

    # Highlight special cases
    for asset in ['GS', 'AXP', 'EURUSD']:  # Example highlights
        if asset in stats_df['Asset'].values:
            idx = stats_df[stats_df['Asset'] == asset].index[0]
            plt.annotate(
                asset,
                (stats_df.loc[idx, 'Skew'], stats_df.loc[idx, 'Kurtosis']),
                textcoords="offset points",
                xytext=(8,5),
                ha='left',
                fontsize=9,
                bbox=dict(boxstyle='round,pad=0.3', fc='white', alpha=0.7)
            )

    # Styling
    plt.title('Asset Return Distributions: Skewness vs Kurtosis', pad=20,
    fontsize=14)
    plt.xlabel('Skewness → Positive = Right-Skewed', fontsize=12)
    plt.ylabel('Excess Kurtosis → Positive = Fat Tails', fontsize=12)
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', frameon=False)
    plt.grid(alpha=0.2)
    sns.despine()

    # Save as high-quality PNG
    plt.savefig(save_path, bbox_inches='tight', dpi=300, transparent=False)
    plt.close()
```

```
    print(f"Saved to {save_path}")

# Usage
plot_skew_kurt(all_stats.reset_index())

# Custom color map
corr_cmap = LinearSegmentedColormap.from_list(
    'corr_cmap', ['#2A5CAA', 'white', '#B22222'])

def preprocess_data(df):
    """Filter to keep only continuous weighted price columns"""
    return df[[col for col in df.columns if 'weighted_' in col]]

def plot_correlation_matrix(df, title, figsize=(10,8)):
    """Plot cleaned correlation matrix"""
    # Clean column names
    df.columns = [col.replace('weighted_', '') for col in df.columns]
    corr = df.corr()

    # Create mask for upper triangle
    mask = np.triu(np.ones_like(corr, dtype=bool))

    plt.figure(figsize=figsize)
    sns.heatmap(
        corr,
        mask=mask,
        cmap=corr_cmap,
        center=0,
        vmin=-1,
        vmax=1,
        annot=True,
        fmt='.2f',
        annot_kws={'size':8},
        linewidths=0.5
    )
    plt.title(title, pad=20)
    plt.xticks(rotation=45, ha='right')
    plt.yticks(rotation=0)
    plt.tight_layout()
```

```
    plt.savefig(f'figures/{title.lower().replace(" ", "_")}.png',
                dpi=300, bbox_inches='tight')
    plt.close()
    return corr

def get_correlation_tiers(corr_matrix):
    """Identify correlation strength tiers"""
    corr_pairs = corr_matrix.stack()
    corr_pairs = corr_pairs[corr_pairs.index.get_level_values(0) !=
                    corr_pairs.index.get_level_values(1)]  # Remove diagonal

    strong = corr_pairs[abs(corr_pairs) > 0.6].sort_values(ascending=False)
    moderate = corr_pairs[(abs(corr_pairs) >= 0.3) &
                          (abs(corr_pairs) <= 0.6)].sort_values(ascending=False)

    return strong, moderate

# 1. FX vs FX correlations
fx_cont = preprocess_data(fx_df)
fx_corr = plot_correlation_matrix(fx_cont, "FX Pairs Correlation Matrix")
fx_strong, fx_moderate = get_correlation_tiers(fx_corr)

# 2. Stocks vs Stocks correlations
stock_cont = preprocess_data(stock_df)
stock_corr = plot_correlation_matrix(stock_cont, "Equities Correlation Matrix")
stock_strong, stock_moderate = get_correlation_tiers(stock_corr)

# 3. FX vs Stocks correlations
combined = pd.concat([fx_cont, stock_cont], axis=1)
cross_corr = combined.corr().loc[fx_cont.columns, stock_cont.columns]
cross_corr.columns = [col.replace('weighted_', '') for col in
cross_corr.columns]
cross_corr.index = [col.replace('weighted_', '') for col in cross_corr.index]

plt.figure(figsize=(12,8))
sns.heatmap(
    cross_corr,
    cmap=corr_cmap,
    center=0,
```

```
    vmin=-1,
    vmax=1,
    annot=True,
    fmt='.2f',
    annot_kws={'size':8},
    linewidths=0.5
)
plt.title("FX vs Equities Cross-Correlation", pad=20)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig('figures/fx_vs_equities_correlation.png', dpi=300,
bbox_inches='tight')
plt.close()

# Exclude discrete/categorical columns (e.g., candlestick patterns)
categorical_keywords = ['Date','CDL', 'CandleDirection','Next']
non_contin_cols = FilterColumn(categorical_keywords, combined_df,
compliment=False)

# Get continuous columns (numeric with >4 distinct values)
continuous_cols = [
    col for col in combined_df.columns
    if col not in discrete_cols
    and np.issubdtype(combined_df[col].dtype, np.number)
    and len(combined_df[col].unique()) > 4
]

continuous_cols

#perform the Henze-Zirkler Multivariate Normality Test (H0: data is
multivariate normal)
from pingouin import multivariate_normality
X_continuous = combined_df[continuous_cols]
multivariate_normality(X_continuous, alpha=.05)

#data isnt multivariate continous

# Scale all continuous features uniformly
```

```
scaler = RobustScaler()
combined_df[continuous_cols] =
scaler.fit_transform(combined_df[continuous_cols])



# Get clean symbol names
symbols = [s.replace('=X', '') for s in SYMBOLS]

# Define model parameters
l1_ratioA = 0.5  # Fixed L1 ratio
full_cols = FilterColumn(['Date','Next'], combined_df, compliment=True)

# Define alpha search space (wider range with more steps)
alphas = np.logspace(-5, -1, 30)  # From 1e-5 to 1e-1

# Initialize results dataframe with all needed columns
linear_results = pd.DataFrame(
    index=symbols,
    columns=[
        'RMSE_base','RMSE_full', 'R2_base','R2_full',
        'Adj_R2_base','Adj_R2_full',
        'AIC_base','AIC_full', 'Features_used_base', 'Features_used_full',
        'Cross_asset_features_used', 'Cross_asset_pctage', 'Alpha_base',
        'Alpha_full'
    ],
    data=151515
)

# Initialize feature importance storage
feature_importance = {
    "linear": {
        sym: {
            "base": {},
            "full": {}
        } for sym in symbols
    }
}
```

```
for symb in tqdm(symbols, desc="Processing assets"):
    # Prepare data for current symbol
    base_cols = [s for s in full_cols if symb in s]
    X_base = combined_df[base_cols]
    X_full = combined_df[full_cols]
    y = combined_df[f"NextCandleDirectionR_{symb}"]

    # ===== ALPHA OPTIMIZATION WITH CONVERGENCE HANDLING =====
    def find_best_alpha(X, y, model_type='base'):
        best_alpha = None
        best_score = -np.inf
        best_model = None

        for alpha in alphas:
            try:
                model = ElasticNet(
                    alpha=alpha,
                    l1_ratio=l1_ratioA,
                    random_state=42,
                    max_iter=100000,
                    selection='random'  # Helps with convergence
                )
                model.fit(X, y)

                # Calculate score (using R² adjusted for fair comparison)
                n = X.shape[0]
                p = np.sum(model.coef_ != 0)
                r2 = model.score(X, y)
                adj_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)

                if adj_r2 > best_score:
                    best_score = adj_r2
                    best_alpha = alpha
                    best_model = model

            except Exception as e:
                print(f"Warning: Failed for {symb} {model_type} with alpha
                {alpha:.2e}: {str(e)}")
                continue
```

```
    return best_alpha, best_model

# Find best alpha for baseline and full models
best_alpha_base, base_model = find_best_alpha(X_base, y, 'base')
best_alpha_full, full_model = find_best_alpha(X_full, y, 'full')

# ===== CALCULATE ALL METRICS AND STORE FEATURE IMPORTANCE =====
def calculate_metrics(model, X, y, alpha, model_type):
    # Basic metrics
    pred = model.predict(X)
    rmse = np.sqrt(mean_squared_error(y, pred))
    r2 = model.score(X, y)

    # Adjusted R²
    n = X.shape[0]
    p = np.sum(model.coef_ != 0)
    adj_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)

    # AIC
    RSS = np.sum((y - pred) ** 2)
    aic = n * np.log(RSS / n) + 2 * p

    # Feature usage
    features_used = np.sum(model.coef_ != 0)

    # Store feature importance
    feature_importance["linear"][symb][model_type] = {
        col: float(coef)  # Convert numpy to native Python float
        for col, coef in zip(X.columns, model.coef_)
        if coef != 0  # Only store non-zero coefficients
    }

    return {
        'RMSE': rmse,
        'R2': r2,
        'Adj_R2': adj_r2,
        'AIC': aic,
        'Features_used': features_used,
```

```
            'Alpha': alpha
        }

    # Calculate and store baseline metrics
    base_metrics = calculate_metrics(base_model, X_base, y, best_alpha_base,
    'base')
    linear_results.loc[symb, 'RMSE_base'] = base_metrics['RMSE']
    linear_results.loc[symb, 'R2_base'] = base_metrics['R2']
    linear_results.loc[symb, 'Adj_R2_base'] = base_metrics['Adj_R2']
    linear_results.loc[symb, 'AIC_base'] = base_metrics['AIC']
    linear_results.loc[symb, 'Features_used_base'] =
    base_metrics['Features_used']
    linear_results.loc[symb, 'Alpha_base'] = base_metrics['Alpha']

    # Calculate and store full model metrics
    full_metrics = calculate_metrics(full_model, X_full, y, best_alpha_full,
    'full')
    linear_results.loc[symb, 'RMSE_full'] = full_metrics['RMSE']
    linear_results.loc[symb, 'R2_full'] = full_metrics['R2']
    linear_results.loc[symb, 'Adj_R2_full'] = full_metrics['Adj_R2']
    linear_results.loc[symb, 'AIC_full'] = full_metrics['AIC']
    linear_results.loc[symb, 'Features_used_full'] =
    full_metrics['Features_used']
    linear_results.loc[symb, 'Alpha_full'] = full_metrics['Alpha']

    # Cross-asset features
    cross_features = sum(1 for col in X_full.columns[full_model.coef_ != 0] if
    symb not in col)
    linear_results.loc[symb, 'Cross_asset_features_used'] = cross_features
    linear_results.loc[symb, 'Cross_asset_pctage'] = (cross_features /
    full_metrics['Features_used'] * 100
                                                      if
                                                  full_metrics['Features_used']
                                                  > 0 else 0)

# Display results
print("\nFinal Linear Regression Results:")
display(linear_results)
```

```
# Save results
linear_results.to_csv('linear_regression_results_optimized.csv')

# Save feature importance
with open('linear_feature_importance.json', 'w') as f:
    json.dump(feature_importance, f, indent=2)

print("\nFeature importance saved to linear_feature_importance.json")

# Initialize symbols and results structures
symbols = [s.replace('=X', '') for s in SYMBOLS]
full_cols = FilterColumn(['Date','Next'], combined_df, compliment=True)
l1_ratioA = 0.5

# Initialize results DataFrame
logistic_results = pd.DataFrame(
    index=symbols,
    columns=[
        'Accuracy_base', 'Precision_base', 'Specificity_base', 'AIC_base',
        'Features_used_base', 'Alpha_base',
        'Accuracy_full', 'Precision_full', 'Specificity_full', 'AIC_full',
        'Features_used_full', 'Alpha_full',
        'Cross_asset_features_used', 'Cross_asset_pct'
    ],
    data=151515
)

# Initialize feature storage dictionary
feature_storage = {
    "logistic": {
        sym: {"base": {}, "full": {}} for sym in symbols
    }
}

# Define alpha search space
alphas = np.logspace(-3, 1, 20)  # Wider range for logistic regression

for symb in tqdm(symbols, desc="Processing assets"):
    # Prepare data
```

```
base_cols = [s for s in full_cols if symb in s]
X_base = combined_df[base_cols]
X_full = combined_df[full_cols]
y = combined_df[f"NextCandleDirectionC_{symb}"]

# ===== ALPHA OPTIMIZATION =====
def find_best_logistic_alpha(X, y, model_type):
    best_alpha = None
    best_score = -np.inf
    best_model = None

    for alpha in alphas:
        try:
            model = LogisticRegression(
                penalty='elasticnet',
                solver='saga',
                l1_ratio=l1_ratioA,
                C=1/(alpha * len(y)),
                random_state=42,
                max_iter=10000,
                class_weight='balanced'
            )
            model.fit(X, y)

            # Use balanced accuracy as selection criteria
            pred = model.predict(X)
            tn, fp, fn, tp = confusion_matrix(y, pred).ravel()
            specificity = tn / (tn + fp)
            precision = precision_score(y, pred)
            score = (accuracy_score(y, pred) + specificity + precision) /
            3  # Combined metric

            if score > best_score:
                best_score = score
                best_alpha = alpha
                best_model = model

        except Exception as e:
            continue
```

```
    return best_alpha, best_model

# Find optimal alphas
best_alpha_base, base_model = find_best_logistic_alpha(X_base, y, 'base')
best_alpha_full, full_model = find_best_logistic_alpha(X_full, y, 'full')

# ===== METRIC CALCULATION =====
def calculate_logistic_metrics(model, X, y, alpha):
    # Predictions
    pred = model.predict(X)
    proba = model.predict_proba(X)[:, 1]

    # Basic metrics
    tn, fp, fn, tp = confusion_matrix(y, pred).ravel()
    metrics = {
        'Accuracy': accuracy_score(y, pred),
        'Precision': precision_score(y, pred),
        'Specificity': tn / (tn + fp),
        'Features_used': np.sum(model.coef_ != 0),
        'Alpha': alpha
    }

    # AIC calculation
    n = len(y)
    k = np.sum(model.coef_ != 0) + 1
    ll = model.score(X, y) * n  # Approximate log-likelihood
    metrics['AIC'] = 2 * k - 2 * ll

    # Feature importance
    features = {
        col: float(coef)
        for col, coef in zip(X.columns, model.coef_[0])
        if coef != 0
    }

    return metrics, features

# Calculate and store baseline metrics
```

```
    base_metrics, base_features = calculate_logistic_metrics(base_model,
    X_base, y, best_alpha_base)
    for metric, value in base_metrics.items():
        logistic_results.loc[symb, f'{metric}_base'] = value
    feature_storage["logistic"][symb]["base"] = base_features

    # Calculate and store full model metrics
    full_metrics, full_features = calculate_logistic_metrics(full_model,
    X_full, y, best_alpha_full)
    for metric, value in full_metrics.items():
        logistic_results.loc[symb, f'{metric}_full'] = value
    feature_storage["logistic"][symb]["full"] = full_features

    # Cross-asset features
    cross_features = sum(1 for col in full_features if symb not in col)
    logistic_results.loc[symb, 'Cross_asset_features_used'] = cross_features
    logistic_results.loc[symb, 'Cross_asset_pct'] = (cross_features /
    full_metrics['Features_used'] * 100
                                                    if
                                                full_metrics['Features_used']
                                                > 0 else 0)

# Save results
logistic_results.to_csv('logistic_regression_results_optimized.csv')

# Save feature importance
with open('logistic_feature_importance.json', 'w') as f:
    json.dump(feature_storage, f, indent=2)

print("\nLogistic Regression Results:")
display(logistic_results)
print("\nFeature importance saved to logistic_feature_importance.json")


# Load your results (replace with your path)
df = pd.read_csv("linear_regression_results_optimized.csv")

# Compute differences (Full - Base)
df["RMSE_diff"] = df["RMSE_full"] - df["RMSE_base"]  # Lower RMSE = better →
```

```
Negative diff = improvement
df["Adj_R2_diff"] = df["Adj_R2_full"] - df["Adj_R2_base"]        # Higher R² =
better → Positive diff = improvement
df["AIC_diff"] = df["AIC_full"] - df["AIC_base"]     # Lower AIC = better →
Negative diff = improvement

n_better_rmse = (df["RMSE_diff"] < 0).sum()  # Count where Full model has
lower RMSE
n_total_rmse = (df["RMSE_diff"] != 0).sum()  # Ignore ties (if any)

# One-sided test: Is Full model better more often than chance?
result_rmse = binomtest(n_better_rmse, n_total_rmse, p=0.5, alternative='two-
sided')
print(f"RMSE: p-value = {result_rmse.pvalue:.4f}")

n_better_Adj_R2 = (df["Adj_R2_diff"] > 0).sum()  # Count where Full model has
higher R²
n_total_Adj_R2 = (df["Adj_R2_diff"] != 0).sum()  # Ignore ties

result_Adj_R2 = binomtest(n_better_Adj_R2, n_total_Adj_R2, p=0.5,
alternative='two-sided')
print(f"Adj_R²: p-value = {result_Adj_R2.pvalue:.4f}")

n_better_aic = (df["AIC_diff"] < 0).sum()  # Count where Full model has lower
AIC
n_total_aic = (df["AIC_diff"] != 0).sum()  # Ignore ties

result_aic = binomtest(n_better_aic, n_total_aic, p=0.5, alternative='two-sided')
print(f"AIC: p-value = {result_aic.pvalue:.4f}")



df = pd.read_csv("logistic_regression_results_optimized.csv")
df.columns

df["Accuracy_diff"] = df["Accuracy_full"] - df["Accuracy_base"]
n_better_Accuracy = (df["Accuracy_diff"] < 0).sum()  # Count where Full model
has lower Accuracy
n_total_Accuracy = (df["Accuracy_diff"] != 0).sum()  # Ignore ties (if any)
result_Accuracy = binomtest(n_better_Accuracy, n_total_Accuracy, p=0.5,
```

```
alternative='two-sided')
print(f"Accuracy: p-value = {result_Accuracy.pvalue:.4f}")

df["Precision_diff"] = df["Precision_full"] - df["Precision_base"]
n_better_Precision = (df["Precision_diff"] < 0).sum()  # Count where Full
model has lower Precision
n_total_Precision = (df["Precision_diff"] != 0).sum()  # Ignore ties (if any)
result_Precision = binomtest(n_better_Precision, n_total_Precision, p=0.5,
alternative='two-sided')
print(f"Precision: p-value = {result_Precision.pvalue:.4f}")

df["Specificity_diff"] = df["Specificity_full"] - df["Specificity_base"]
n_better_Specificity = (df["Specificity_diff"] < 0).sum()  # Count where Full
model has lower Specificity
n_total_Specificity = (df["Specificity_diff"] != 0).sum()  # Ignore ties (if
any)
result_Specificity = binomtest(n_better_Specificity, n_total_Specificity,
p=0.5, alternative='two-sided')
print(f"Specificity: p-value = {result_Specificity.pvalue:.4f}")

df["AIC_diff"] = df["AIC_full"] - df["AIC_base"]
n_better_AIC = (df["AIC_diff"] < 0).sum()  # Count where Full model has lower
AIC
n_total_AIC = (df["AIC_diff"] != 0).sum()  # Ignore ties (if any)
result_AIC = binomtest(n_better_AIC, n_total_AIC, p=0.5, alternative='two-
sided')
print(f"AIC: p-value = {result_AIC.pvalue:.4f}")
```