Predicting and Forecasting Cocoa Futures Prices Using Time Series Analysis

Datasets:
Cocoa Futures Price Data from the International Cocoa Organization (ICCO)
Climate Data (Temperature and Precipitation) from Ghana from the National Centers for Environmental Information (NCEI)

Luke Ham
4 April 2025

Abstract

Primarily grown in West Africa, Latin America, and Southeast Asia, Theobroma cacao, more commonly recognized as the cocoa tree, is the source of cocoa and many other products that cocoa produces, such as chocolate. Cocoa is an essential food industry commodity in the global market since it is a primary ingredient in chocolate production. However, it experiences significant market price volatility due to environmental and economic factors (Voora et al., 2019). Consequently, it is important to be able to predict and forecast the price movements of cocoa futures as accurately as possible as it benefits cocoa farmers and chocolate production companies. In this project, we developed a predictive model to forecast cocoa futures prices by comparing the performance of two time series models- ARIMA (0,1,1) and GRACH(1,1)-with two machine learning models: Gradient Boosting Machine(GBM) and Random Forest(RF). Using historical cocoa price data from the International Cocoa Organization (ICCO) and climate data from Ghana (National Centers for Environmental Information, NCEI), we applied time series analysis and machine learning techniques. Following data preprocessing, model selection, and model tuning, the RF model outperformed the others, delivering the best accuracy and reliability in predicting and forecasting cocoa futures prices. These findings contribute to better decision-making for farmers and chocolate production companies.

Introduction

Historically, modeling commodity prices plays a crucial role in agriculture and international trade industries as commodity price models may display signs of seasonality and volatility. These models help forecast price fluctuations and make optimal financial decisions. Among these commodities, with approximately 4 million tonnes produced annually, cocoa is a significant key agricultural commodity globally as it is the primary ingredient in chocolate production as well as other food products—while being easily susceptible to price volatility caused by various factors (Voora et al., 2019). Furthermore, the cocoa industry is experiencing growth in global demand as the demand for cocoa ingredients such as cocoa powder used in cookies, is increasing in developing countries with growing economies. In fact, the cocoa industry is expected to experience a 7.3% Compound Annual Growth Rate from 2019 to 2025. This growth is attributed to the growth of the middle-upper class—particularly in Asian countries such as China and India. According to a global market report for Cocoa, “Asia is expected to become the second largest consumer market of cocoa-based ingredients in the world after Western Europe” (Voora et al., 2019). The cacao tree, or Theobroma cacao, grows in the equatorial belt between 10 degrees north and south of the equator in tropical climates such as West Africa, Latin America, and Southeast Asia. West Africa is said to make up 70% of global cocoa production, with most cocoa coming from Ghana and the Ivory Coast. Due to the area the cacao tree grows in as well as its requirements to grow, it is often affected by many factors and can possibly limit the continuous growth of the cocoa industry. This is partly because while the cacao tree relies on climates with a lot of rain and humidity, it is also sensitive to environmental and economic conditions. Thus, the cocoa market has historically been characterized by significant market price volatility driven by factors such as climate conditions, geopolitical events, supply chain disruptions, and global demand fluctuations. This unpredictability creates challenges for farmers and chocolate production companies, who rely on accurate price forecasts for decision-making. To be more specific, the cacao tree has given cocoa farmers a challenge in meeting these demands due to rising temperatures, aging cocoa trees that give less yield, and systemic poverty that affects the farmers (Voora et al., 2019). Additionally, other weather conditions such as droughts or excessive rainfall, pests such as cocoa pod borers, diseases

such as black pod disease, supply chain and politics, currency exchange rates, and the increasing demand from industries such as the chocolate industry in China and India all influence the cocoa prices, so we must understand these factors to model cocoa futures prices effectively.
As such, despite the growing demand, cocoa supply is limited by these significant farm risks—thus it is crucial to be able to predict and forecast cocoa futures prices using time series analysis by considering numerous factors such as weather conditions. Ultimately, this project and study are motivated by the need for better forecasting methods of cocoa futures prices as traditional methods may struggle to comprehend the complex relationships between these variables. Therefore, this study and project's motivation is to develop and evaluate time series forecasting models to accurately predict cocoa futures price predictions by analyzing historical price data and climate conditions. This project focuses on applying time series analysis techniques to forecast cocoa futures prices, and our key objective is to develop an effective, well-reasoned and data-driven model for forecasting cocoa futures prices—with the model capturing the underlying patterns and improving price predictions.
This study has significance in the real world as it would improve decision-making in the cocoa industry as understanding these patterns, trends, and risks and being able to predict cocoa futures prices is essential not only for farmers and cocoa producers but also for buyers such as chocolate production companies as the exported cocoa beans industry is valued at 8.6 billion USD in 2017 and provides revenue and a source of income for approximately 50 million people in 2012—especially in developing countries (Voora et al., 2019).
The study utilizes datasets from the International Cocoa Organization (ICCO) and climate data from Ghana, the world's leading cocoa producer, collected by the National Centers for Environmental Information (NCEI). The project presents several challenges, including seasonality and volatility, that impact price movements. To address these complexities, we explore multiple methodologies, including classical time series models such as ARIMA(0,1,1) and GRACH(1,1), alongside advanced machine learning techniques like Gradient Boosting Machine (GBM) and Random Forest (RF).

Problem Statement

Predicting cocoa futures prices is essential for both cocoa farmers and chocolate production companies since cocoa—a primary commodity in chocolate production- poses key significant challenges such as market price volatility due to various factors such as weather and climate conditions. Motivated by these issues, this project aims to create a model that accurately predicts and forecasts cocoa futures prices for informed decision-making. This problem has significant real-world relevance since cocoa is a part of a lot of economies—particularly of developing countries with growing economies such as West Africa which exports cocoa and India or China which imports cocoa. To address this issue, the objective of this project is to apply time series analysis techniques to develop an accurate and reliable forecasting model for predicting cocoa futures prices by using historical price data trends from the International Cocoa Organization (ICCO) and climate data from Ghana. By improving price forecasting prediction accuracy, this project supports economic stability in cocoa-producing regions.

Literature Review

Time series forecasting has been a critical area of research, particularly in the context of commodity price modeling. Various methodologies have been proposed to capture the behavior of time-dependent data, ranging from classical statistical models to advanced machine learning techniques.

Box and Jenkins (1970) introduced two distinct models that describe the behavior of time series data: the Autoregressive Integrated Moving Average (ARIMA) model, which has been widely recognized in the field (Kongcharoen and Kruangpradit, 2013). The ARIMA model consists of three main components: autoregressive (AR), integrated (I), and moving average (MA) processes (Gibrilla et al., 2018). The parameters of the model include p (the order of the autoregressive component), d (the degree of differencing for stationarity), and q (the order of the moving average component), all of which are non-negative integers (Bhavani and Singh, 2018).

Since its introduction, ARIMA has been extensively used in various domains, including energy consumption and greenhouse gas emissions (Sen et al., 2016; Yuan et al., 2016), agriculture production and fisheries (Praveen and Sharma, 2019; Stergiou, 1991), and commodity price forecasting, such as rice prices and production (Bhavani and Singh, 2018; Ohyver and Pudjihastuti, 2018). The availability of ARIMA functions in the R forecast package (e.g., `auto.arima()`, `Arima()`, `arima()`, and `ar()`) has facilitated its application in forecasting, ensuring the selection of optimal models using statistical criteria like Akaike Information Criterion (AIC) and Maximum Likelihood Estimation (MLE) (Hyndman and Khandakar, 2008).

On the other hand, GARCH (Generalized Autoregressive Conditional Heteroskedasticity) models have traditionally been used to model the volatility of asset prices, with applications extending to forecasting the volatility of cryptocurrencies (Trucíos, 2019). GARCH processes are particularly effective in predicting time series with heteroscedasticity (variable variance), and their evaluation typically requires fewer computational resources. For time series data with high frequency, GARCH is preferred, whereas ARIMA or wavelet ARIMA models are more commonly applied for time series with lower frequency (Rubio et al., 2023).

Traditional Grey Models (GM), such as GM(1,1), have been utilized for time series forecasting, particularly for data with small samples and limited information (Xie et al., 2013). However, these models assume homogeneous index trend characteristics, which may not always hold in real-world scenarios. The Nonhomogeneous Discrete Grey Model (NDGM) was developed to address this limitation, which follows the principle of approximate non-homogeneous exponential growth (Javed and Liu, 2018; Wu et al., 2014). The NDGM model extends the Discrete Grey Model (DGM) and has been applied in various domains, including tourism forecasting (Pirthee, 2017), research productivity prediction (Javed and Liu, 2018), and ISO certification forecasting (Ikram et al., 2019).

Long-short-term memory (LSTM) networks are a specialized form of Recurrent Neural Networks (RNNs) designed to handle long-term dependencies in sequential data (Cai, 2024). The key innovation in LSTMs lies in their gating mechanisms—the input gate, forget gate, and output gate—which allow the network to retain and discard information across time steps selectively. This enables LSTMs to effectively model long sequences without suffering from the vanishing gradient problem, a common issue in traditional RNNs.

Despite their effectiveness, LSTMs still have limitations in capturing global dependencies in time series data. Researchers have explored hybrid models, such as Attention-based Convolutional Neural Network (ACNN) and LSTM architectures, which improve sequence modeling by integrating local and global dependencies (Sutskever, Vinyals, & Le, 2014). Shi et al. (2022) further enhanced this approach by combining LSTM, CNN, and XGBoost in a hybrid framework, demonstrating improved predictive performance for stock market forecasting. Their study highlighted the effectiveness of

ARIMA preprocessing, which enhances the accuracy of deep learning models by capturing linear trends before feeding the data into complex architectures.

Tree-based ensemble methods, such as RF and XGBoost, have gained popularity in time series forecasting due to their robustness and flexibility. Mei et al. (2014) applied the Random Forest algorithm to the New York electricity market, demonstrating its superiority over traditional Artificial Neural Networks (ANNs) and ARMA models. Similarly, Kara et al. (2021) used Random Forest and Support Vector Regression (SVR) for price and volatility forecasting in the Turkish electricity market, finding that RF outperformed SVR in volatility prediction.

XGBoost, an optimized version of gradient boosting, has been widely adopted for stock price forecasting due to its high prediction accuracy and computational efficiency (Shi & Hu, 2022). Their study introduced the Attention-based CNN- LSTM and XGBoost hybrid model (AttCLX), demonstrating strong predictive capabilities for Bank of China stock prices. However, they only tested this model in the Chinese stock market, leaving open the question of its effectiveness in other financial markets.

The Geometric Brownian Motion (GBM) model is a widely used stochastic process for modeling commodity prices, stock prices, and exchange rates (Kumar et al., 2024). GBM is a non-negative variation of Brownian motion (BM), making it particularly suitable for financial applications, including option pricing, value at risk, and mortgage insurance (Ibe, 2013).

Although the classical GBM model assumes price independence, recent studies suggest that financial time series exhibit memory effects (Han et al., 2020; Rejichi & Aloui, 2012). To account for this, researchers have developed the Geometric Fractional Brownian Motion (GFBM) model, which replaces the traditional BM process with a fractional Brownian motion (FBM) process, incorporating long-term memory properties (Grau-Carles, 2000; Kim et al., 2020). These extensions improve the accuracy of price forecasting models, making them more suitable for real-world financial applications.

In this study, we chose ARIMA(0,1,1) and GARCH(1,1) as classical time series models due to their strong theoretical foundations and widespread use in forecasting economic and commodity prices. We also included two machine learning models—Gradient Boosting Machine (GBM) and Random Forest (RF)—to capture more complex, non-linear relationships in the data that traditional models like ARIMA and GARCH may struggle to address.

Methodology

To capture the different dynamics in cocoa futures prices, we built four models: ARIMA(0,1,1), GARCH(1,1), GAM regression, and Random Forest (RF). Each model was chosen based on its theoretical motivation, empirical features observed in the data, and support from literature.

Data Transformation and Feature Engineering

The log transformation was performed on the cocoa price series to smooth large fluctuations and address the sudden rise in 2023. To meet the stationarity assumption required by models like ARIMA and GARCH, we take the first difference of the log prices. The resulting daily log returns appear stationary based on visual inspection. We handle missing values using Kalman smoothing or remove rows when necessary. We build a complete monthly timeline for the climate data and fill in the missing months using average values. Monthly information is extracted to capture seasonal patterns. In addition, we also create lagged versions of the log prices up to seven days back and use them as features in the machine learning models. For Random Forest, lag selection is tuned using a walk-forward expanding window to reflect real-time forecasting better. Lastly, we use the back-transformation to validate the forecasting, which changes the diff log price back to the actual values.

ARIMA(0,1,1)

To ensure consistency across datasets and minimize daily fluctuations, we aggregated the daily cocoa price data to the monthly level and converted it into a time series format for analysis. We implemented an ARIMA(0,1,1) model to capture short-term linear trends in the log-transformed cocoa price series. The original series exhibited clear non-stationarity, which was confirmed by the Augmented Dickey-Fuller (ADF) test and eye inspection. After first differencing, the series appeared stationary and suitable for modeling.

Both visual diagnostics and theoretical criteria supported the choice of model order. The autocorrelation function (ACF) showed a significant spike at lag 1(see Fig.1), followed by a rapid decline. While the partial autocorrelation function (PACF) displayed no strong patterns beyond the first lag and tailed off. This structure is consistent with a moving average process of order one and supports the ARIMA(0,1,1) specification.

ARIMA is a fundamental technique in time series forecasting, mainly for economic and commodity data. Prior research on rice price forecasting, including Gibrilla et al. (2018) on groundwater levels in Ghana and Ohyver and Pudjihastuti (2018), illustrates its efficacy in capturing linear temporal correlations. Although ARIMA cannot address volatility clustering or nonlinear interactions, it provides a straightforward and interpretable baseline as a benchmark for assessing more sophisticated models.

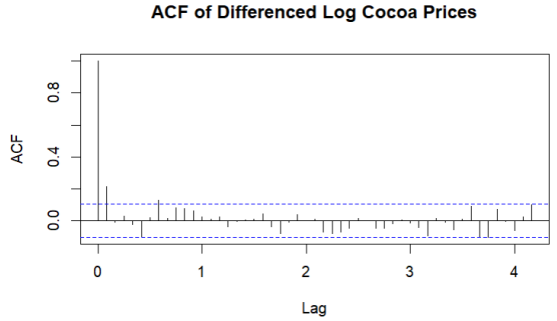

Fig.1

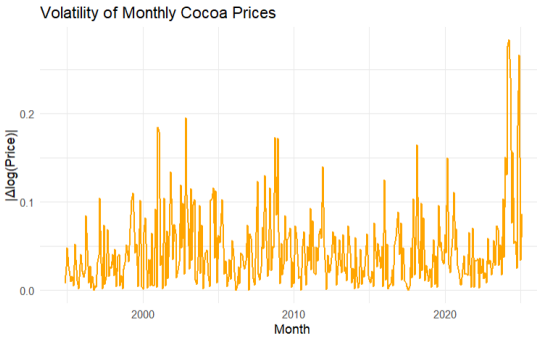

Fig.2

GARCH(1,1)

Although ARIMA effectively models the conditional mean of a time series, it assumes constant variance over time, which is not satisfied in our data. As seen in Fig. 2, the return series shows apparent volatility clustering—periods of stability interrupted by sharp price movements (upward), particularly after 2023. This pattern violates the homoskedasticity assumption and calls for a model that accounts for time-varying conditional variance.

To address this issue, we utilized a GARCH(1,1) model on the first-differenced log prices. The GARCH model was introduced by Bollerslev (1986) as an extension of ARCH. It enables the conditional variance to be influenced by prior squared errors and prior variances. This method enhances the efficacy of capturing sustained volatility. This expansion offers a more adaptable and robust method for simulating volatility across time. Compared to simple ARCH models, GARCH(1,1) requires more lag terms to capture long memory in volatility and often achieves similar or better performance using fewer parameters. The GARCH(1,1) model is specified as:

$$r\_t = \mu + \varepsilon\_t, \qquad \varepsilon\_t = \sigma\_t \times z\_t, \qquad z\_t \sim N(0,1)$$

$$\sigma\_t^2 = \alpha_0 + \alpha_1 \times \varepsilon^2\_\{t-1\} + \beta_1 \times \sigma^2\_\{t-1\}$$

This specification allows volatility to respond dynamically to new market information, making it suitable for financial series where uncertainty varies over time. GARCH models are frequently used in economic and commodity forecasting because they can simulate heteroskedasticity and generate time-varying confidence intervals. In our context, the GARCH(1,1) model captures volatility increases associated with climatic disruptions, rising demand, or political changes. Han et al. (2020) observed that explicitly modeling volatility enhances risk management and improves the interpretability of financial time series.

Generalized Additive Model (GAM)

We used a Generalized Additive Model (GAM) to capture potential nonlinear trends in cocoa prices that a linear model cannot. This approach maintains the interpretability of additive models while introducing flexibility through smooth functions to account for the random effects. Given the observed increase in cocoa price volatility and possible nonlinear time effects after 2023 (see Fig. 4), a GAM is suitable for modeling such complex dynamics. Our GAM includes two lagged log prices, average temperature (TAVG), and maximum temperature (TMAX) as linear terms while incorporating a smooth function over time. This allows the model to account for long-term trends and seasonal variation. The fitted model is expressed as:

$$Log\ (Price\_t) = \beta_0 + \beta_1 \times lag_1\_t + \beta_2 \times lag_2\_t + \beta_3 \times TAVG\_t + \beta_4 \times TMAX\_t + f(time\_t) + \varepsilon\_t$$

Where *f(time_t)* is a smooth function of time that flexibly models long-term price movements, this semi-parametric structure balances fitting accuracy with smoothness through penalized likelihood, avoiding overfitting (Wood, 2017; Hastie & Tibshirani, 1990). Furthermore, the study by Bhavani and Singh (2018) also indicates that regression works well for agricultural price and yield modeling when predictors are well-structured. Although it does not account for nonlinear relationships or time-varying effects, the model serves as a helpful baseline for evaluating the performance of more advanced methods.

Fig. 3

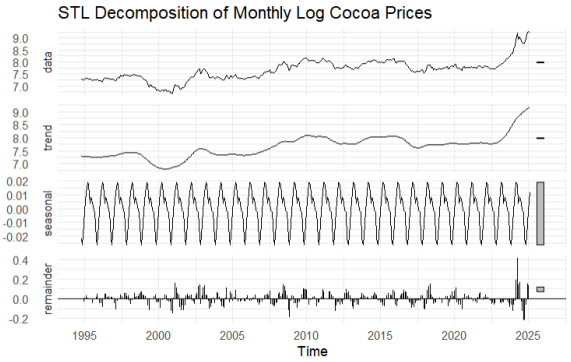

Fig. 4

Random Forest Utilizing Expanding Window and Dynamic Lag Selection

We utilized a Random Forest (RF) model to explain nonlinear correlations and interactions between lagged prices and climatic predictors. Random Forest is efficient at forecasting in noisy, high-dimensional environments without relying on distributional assumptions.

Walk-forward forecasting involves retraining the model on an expanding dataset at each iteration, using the most recent data to predict the next period. This iterative process simulates real-world forecasting by continuously incorporating new information, allowing the model to adapt to changes over time. Since our data changed quickly after 2023, this method is helpful because it helps the model adjust to these changes and make more accurate forecasts.

We employed an expanding window technique for model training to replicate realistic forecasting scenarios. Unlike static training splits, expanding window validation better reflects real-time forecasting conditions. To determine the best lag for the model, we tested different lag values and evaluated the model performance based on RMSE, MAE, and MAPE. Based on the performance of each iteration, we identified the best-performing lag by evaluating different lag lengths and selecting the one with the lowest error metrics through cross-validation. This ensured the model could accurately capture both short- and long-term price movements. In addition, it enables the model to adjust to changing temporal dependencies affected by seasonality or external shocks, as illustrated in Fig. 5–6.

Previous research supports this methodology. Kara et al. (2021) and Mei et al. (2014) demonstrated the efficacy of RF in volatile situations and with walk-forward retraining, consequently confirming its utility for forecasting commodity prices.

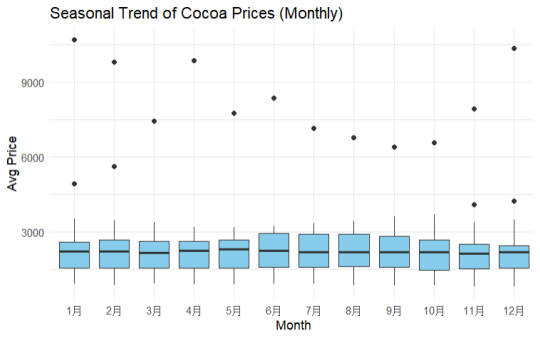

Fig. 5

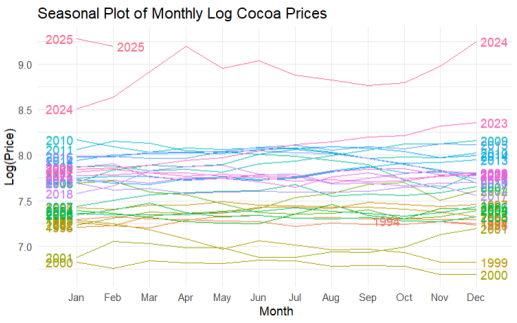

Fig. 6

Forecast Evaluation

We assessed model performance utilizing RMSE, MAE, and MAPE. RMSE emphasizes significant mistakes, MAE reflects overall precision, and MAPE indicates errors in relative terms. Alongside measurements, we employed forecast-versus-actual graphs and back-formation to evaluate prediction quality visually and identify problems such as underfitting or lagging. We also analyzed the residuals of ARIMA and GARCH for autocorrelation and heteroskedasticity. This quantitative and visual evaluation combination comprehensively assesses each model's generalization performance.

Data

This study relies on two principal datasets—daily cocoa futures prices and Ghanaian climate data—merged to create a comprehensive dataset for modelling. Integrating economic and environmental variables provides a robust basis for understanding and forecasting cocoa price movements.

Cocoa Futures Price Data

The cocoa price dataset is sourced from the International Cocoa Organization (ICCO) and contains the daily closing prices of cocoa futures contracts from March 10, 1994, to February 27, 2025. The original dataset includes prices recorded in the "US$/tonne" format and requires parsing in numeric format for modelling. The variables include:

- Date: The date of the transaction.
- Price: Daily closing price of cocoa futures (US$/tonne).

Ghana Climate Data

The Ghana weather dataset, sourced from the National Centres for Environmental Information (NCEI), includes daily meteorological observations from multiple stations across Ghana, a key cocoa-producing region. The dataset spans from January 1, 1990, and includes:

- DATE: Observation date
- PRCP: Daily precipitation (mm)
- TAVG: Daily average temperature (°F)
- TMAX: Daily maximum temperature (°F)
- TMIN: Daily minimum temperature (°F)

Merged Dataset

The two datasets were merged on the Date field using an inner join, resulting in a final dataset suitable for time series forecasting. The merged dataset preserves the alignment between daily cocoa price observations and corresponding climate indicators.

- Final Variables: Date, Price, PRCP, TAVG, TMAX, TMIN

The following two plots show the overall trend of cocoa price over time and the seasonality trend, respectively.

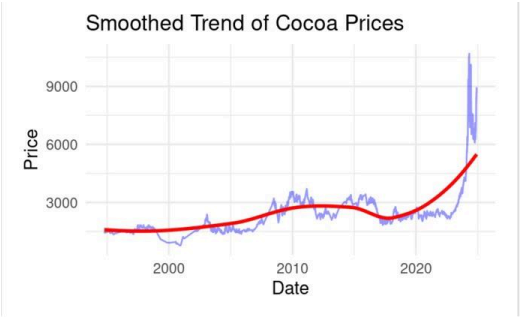

Fig.7

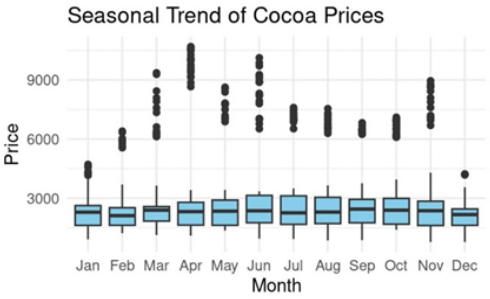

Fig.8

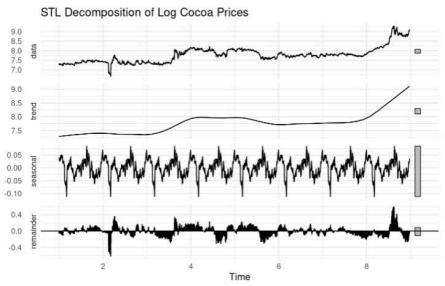

Fig. 9

The cocoa market experienced relatively modest volatility until around 2008, after which more pronounced price fluctuations were observed. Notable upward price movements occurred during the late 2000s and mid-2010s, with a particularly sharp spike starting around 2023–2024. These may correspond to concerns over poor conditions in West Africa and strong demand for Seanad. Prices rose from around $2,500 to over $9,000 per tonne in a relatively short span, warranting further econometric or causal investigation.

Cocoa prices tend to be higher from March to June, with April and May having the greatest median values and the highest dispersion. They also feature numerous outliers with exceptionally high prices. This may align with crop cycles in West Africa, where mid-harvest supply constraints and logistical bottlenecks can drive prices up. The months of September through November appear more stable and slightly lower in median price, likely corresponding with post-harvest periods when supply is more abundant.

As mentioned above, we first used log transformation on price to stabilize the variance. From its decomposition, the trend component highlights a steady increase in log prices post-2020, aligning with recent market volatility. The seasonal component shows a consistent intra-year pattern with annual peaks and troughs, confirming strong seasonality in cocoa prices. The remainder captures irregular fluctuations and short-term shocks, including recent volatility spikes.

The average cocoa price is USD 2,288.24, with a high standard deviation of 1,138.62, indicating large price fluctuations. The log-transformed price has a mean of 7.65 and a lower standard deviation of 0.41, showing reduced skewness. The differenced log price has a near-zero mean and low variability, suggesting stationarity. Precipitation (PRCP) averages 0.23 with a standard deviation of 0.41, indicating most days had little to no rain. Average, maximum, and minimum temperatures (TAVG, TMAX, TMIN) show relatively stable patterns, with means of 81.09°F, 88.94°F, and 74.06°F, respectively, and low standard deviations. (Table and histograms are in the appendix.)

Forecasting and Results

ARIMA(0,1,1)

To model and forecast cocoa prices, we applied an ARIMA model to the log-transformed and monthly-aggregated price series. The original series displayed clear non-stationarity since the ACF plot (Fig.10) revealed a slow decay in autocorrelations. Then, we applied a logarithmic transformation to stabilize the variance, followed by first-order differencing to remove the trend. This transformation addresses both non-stationarity and heteroscedasticity, making the data more suitable for linear time series modeling. Fig. 11 shows that the differenced log-transformed series fluctuates around a constant mean with a stable variance, indicating stationarity. To further assess the autocorrelation structure of the differenced log-transformed series, we examined the ACF(Fig.12) and PACF(Fig. 13) plots. The ACF plot exhibited a significant spike at lag 1, followed by a rapid decline, while the PACF showed no substantial correlations beyond the first lag and gradually tapered off. Given that the series was differenced once (d = 1), this autocorrelation structure is characteristic of a moving average process of order one, supporting the suitability of an ARIMA(0,1,1) model specification.

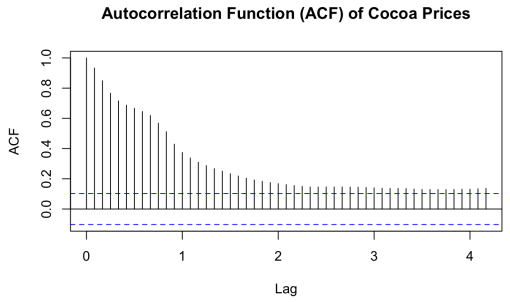

Fig.10

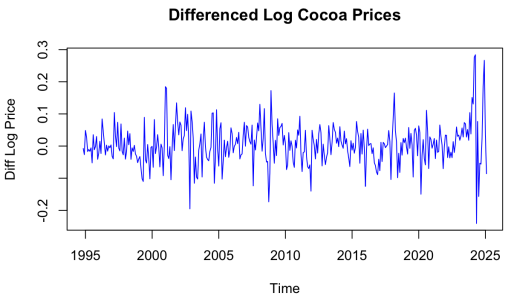

Fig.11

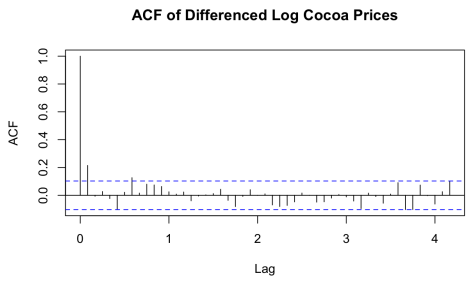

Fig.12

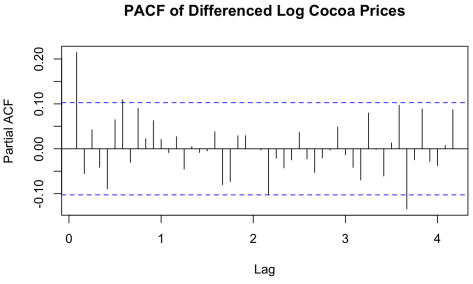

Fig.13

The model training involved fitting an ARIMA(0,1,1) model to the full dataset without splitting, using the Arima() function in R with maximum likelihood estimation. The model included first-order differencing to address non-stationarity and a moving average component to capture short-term error correlations. Forecast performance was evaluated based on in-sample predictions using standard error metrics. The Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) were 285.15, 123.85, and 4.44%, respectively. These results suggest that the model provided a strong fit to the training data and is suitable for short-term forecasting of cocoa prices(see Fig.14). However, when the ARIMA(0,1,1) model is used to forecast cocoa prices for the next 4 months, its performance is poor(see Fig.15). The predicted value is a straight line, and it is difficult to see the trend change, even with the 95% confidence interval.

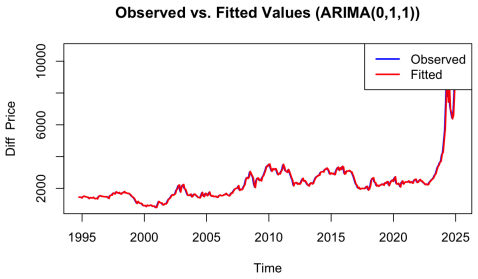

Fig. 14

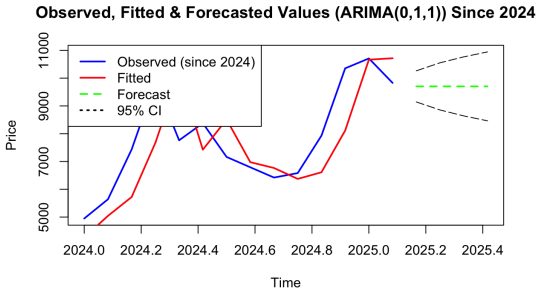

Fig. 15

GARCH(1,1)

To forecast cocoa prices, we implemented a GARCH(1,1) model with a constant mean (ARMA(0,0)) and normally distributed errors. The model was fitted on daily log returns. All estimated coefficients were statistically significant: $\alpha_1 = 0.0416$, $\beta_1 = 0.9491$, and $\alpha + \beta = 0.9907$, indicating strong volatility persistence. The Box-Ljung test results for standardized residuals ($p = 0.1341$) and squared residuals ($p = 1.0$) suggest no significant autocorrelation, confirming a good in-sample fit. In addition, the estimated sum of 1 indicates that volatility shocks are highly persistent. This implies that the impact of a shock to returns does not dissipate quickly but instead carries forward across many future periods. The volatility exhibits long memory, and the conditional variance responds slowly to new information.

The GARCH(1,1) model captures the usual volatility patterns, such as gradual changes and periods of higher risk. However, the model struggles when prices change quickly or unexpectedly. The plot shows that the realized volatility (from actual returns) reacts fast to price spikes in 2023, while the GARCH model adjusts slowly. This happens because GARCH doesn't include outside factors and assumes price changes are primarily stable. So when there's a sudden market event, like after 2023, the model can't keep up and produces a smoother result. The comparative graph demonstrates that the predicted price trajectory is excessively uniform and significantly lags the actual price trend. Consequently, it cannot accommodate external shocks or capture sustained trend shifts, leading to an overly smooth forecast path that fails to reflect the dynamic nature of the actual price.

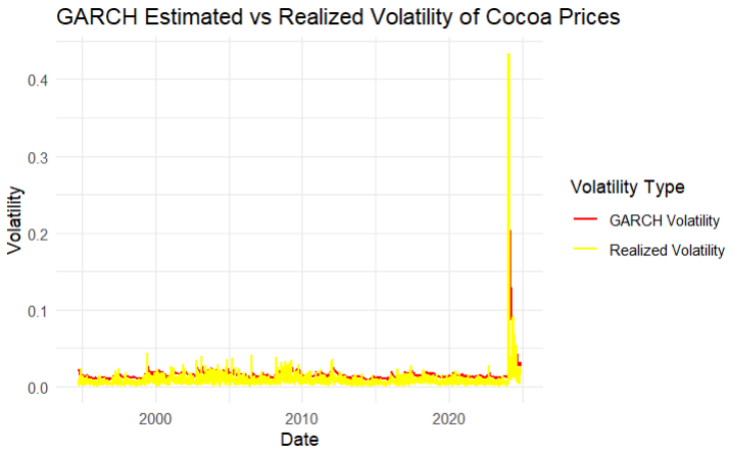

Fig.16

GAM

To forecast cocoa prices, we implemented a Generalized Additive Model (GAM) using lagged log-prices, temperature averages (TAVG), and maximum temperature (TMAX) as predictors. The

model training involved splitting the dataset into 80% training and 20% testing, and we used a walk-forward validation approach over a 30-day horizon. The forecast process started with the last observed values from the training set and iteratively predicted future prices using the GAM model, updating lag values at each step. Forecast performance was evaluated using Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE), which were 2660.79, 1688.72, and 29.15%, respectively. These results indicate relatively poor forecasting accuracy, as the model underestimates sharp price increases and fluctuations after the shock. This discrepancy is clearly illustrated in the graphical comparison of actual versus forecasted prices (Fig.17), where the orange GAM forecast line lags behind the steep rise in black actual prices.

Fig.17

Random Forest (GBM for comparison)

To better forecast the cocoa prices, we implemented two machine learning methods using a walk-forward expanding window: Random Forest (RF) and Gradient Boosting Machine (GBM). Both models were trained on lagged log prices and other relevant features such as temperature averages (TAVG) and maximum temperature (TMAX), which were included to capture seasonal and weather-related effects on cocoa prices. The model was trained on 80% of the data, with walk-forward forecasting retraining the model at each step on an expanding training set to predict the next step. Based on the Lag tuning function, the best lag value with the lowest error matrix for both the RF and GBM models is lag_4. After training and forecasting, we evaluated the models using RMSE, MAE, and MAPE. For the RF model, the RMSE was 234.65, the MAE was 89.46, and the MAPE was 1.82%, indicating a good fit with low error. In comparison, the GBM model showed higher errors across all metrics, with an RMSE of 1206.74, MAE of 568.35, and MAPE of 10.97%.

In the residual plots (Fig. 18), the RF model shows smaller and more stable residuals over time, while the GBM model exhibits more significant deviations, particularly during sharp price increases. The scatter plot (Fig. 19) of actual versus predicted prices further highlights that the RF model aligns more closely with the diagonal line (perfect prediction). This illustrates that RF and GBM are suitable for lower prices, although both models show some divergence at higher price levels. The expanding window forecast plot (Fig. 21) indicates that the RF model tracks price trends more accurately, especially during rapid price fluctuations, indicating its better suitability for this task than the GBM. The RF model generated a 2000-day forecast (Fig. 22) based on past cocoa prices and lagged features. The plot shows that the forecast remains relatively stable and bounded, capturing recent price levels but failing to reproduce the sharp upward trend observed before the forecast period.

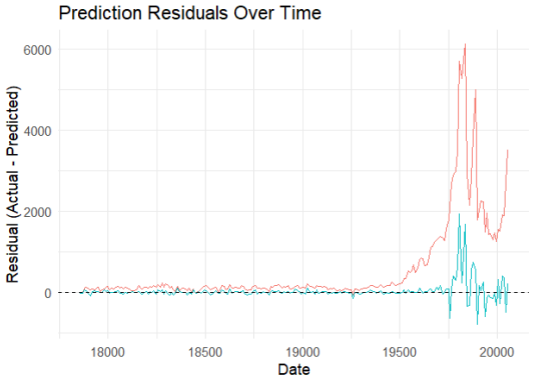

Fig. 18

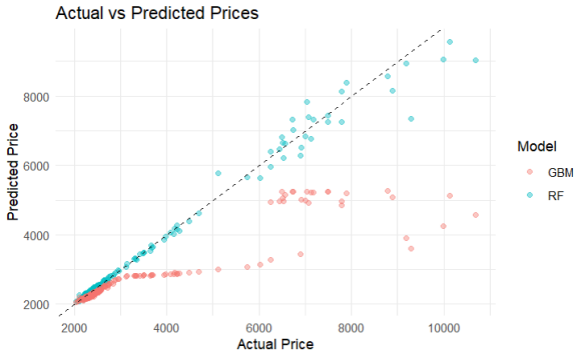

Fig.20

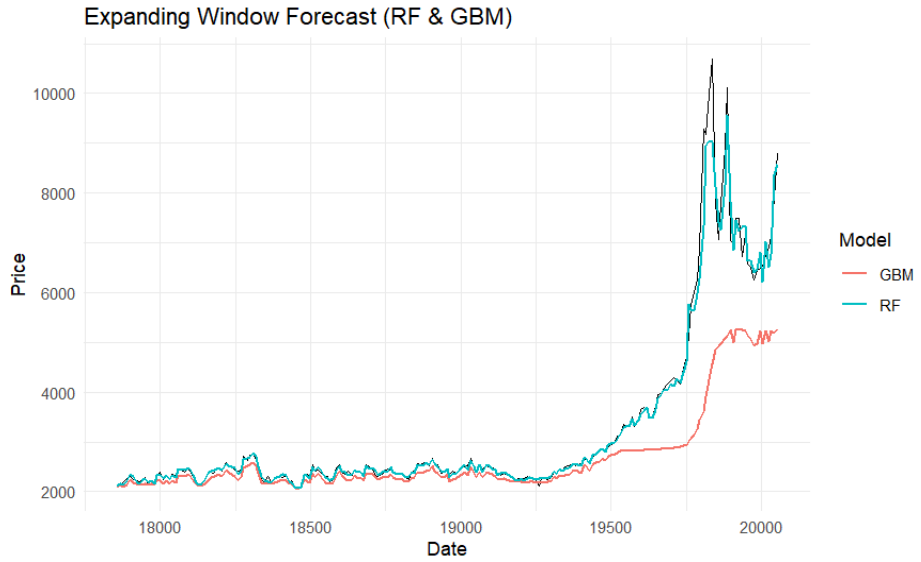

Fig. 21

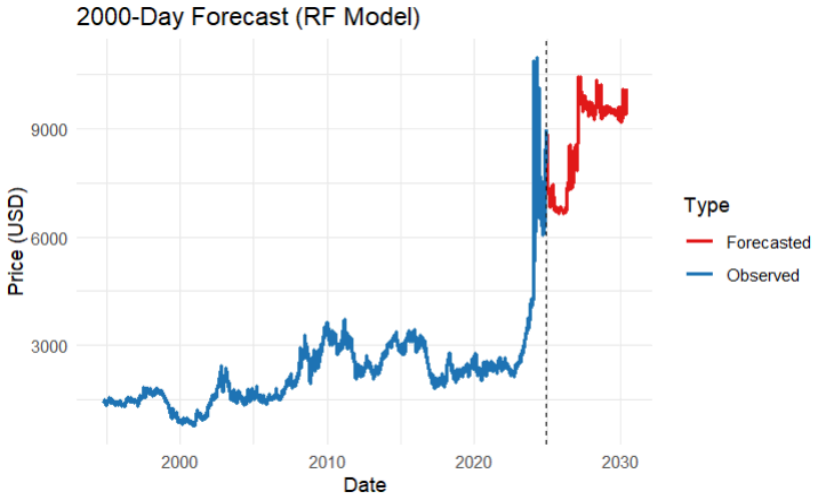

Fig.22

Discussion and Conclusion

We compared the forecasting performance of two time series models—ARIMA(0,1,1) and GARCH(1,1)—with two machine learning models—Gradient Boosting Machine (GBM) and Random Forest (RF)—to forecast cocoa prices. The goal was to determine which model best captures the

complexities of the cocoa market, characterized by volatility and sudden price shifts, and provides the most accurate short-term predictions.

The GARCH(1,1) model is designed to simulate volatility and more effectively captures periods of increased risk and gradual price movements. However, the model cannot respond to sharp, sudden price changes. During the 2023 price spike, the GARCH model was too slow to react and only gradually adjusted to the changes. This slow adjustment is due to the GARCH model's assumption that price changes are steady and excludes external factors that affect the market. The comparison plot clearly shows that the forecasts produced by the GARCH model are too smooth and lag behind actual price trends, failing to capture the dynamic price movements every day in the cocoa market.

We selected two machine learning models to incorporate temperature data into price prediction and better capture price trends: the Gradient Boosting Machine (GBM) and Random Forest (RF). Both models can handle complex, non-linear relationships in time series data. While the GBM model showed some improvement over traditional methods, it still exhibited higher errors across all metrics—RMSE (1206.74), MAE (568.35), and MAPE (10.97%). It particularly struggled during sharp price increases, with residual plots revealing significant deviations, indicating its difficulty in adapting quickly to rapid market changes. In contrast, the RF model outperformed GBM, with significantly lower error metrics—RMSE (234.65), MAE (89.46), and MAPE (1.82%), all of which were better than those of the ARIMA model. The RF model demonstrated superior stability in residuals and better accuracy in tracking price trends, especially during periods of volatility. RF effectively captured seasonal effects by incorporating external features such as temperature data, making it more robust in forecasting cocoa prices. The RF model also performed particularly well in predicting future cocoa prices. Unlike the horizontal line of the ARIMA model, the RF model was able to predict future price changes more effectively, capturing trends and fluctuations more accurately.

The forecasted values from the Random Forest (RF) model successfully capture the general upward trend in cocoa prices observed over the historical period. The model predicts a continued rise in prices after 2025, with a notable increase that aligns with the observed upward trend before the forecast period. The forecast reflects the broader market direction, which is evident in the historical data, including the significant rise in prices around 2023. As we approach the forecast period after 2025, the RF model suggests that prices will steadily increase, maintaining the growth trajectory of previous years. The forecast values are consistent with the overall price dynamics. The model does an excellent job of capturing the long-term growth pattern.

Although the RF model we ultimately selected provides a good fit for the given data, our forecast results are still limited by the constraints of the dataset. Our next step will be to explore additional data on supply chains and political stability, which can offer a better explanation for the sudden increase in cocoa prices around 2023. This will lead to better forecast results.

Appendix

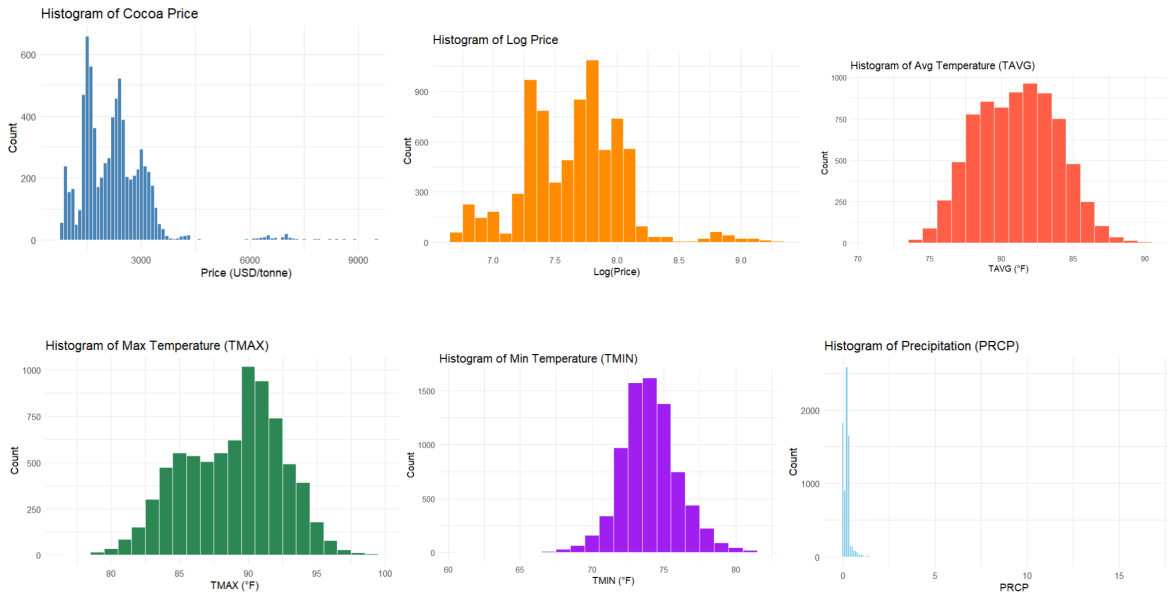

Table: Descriptive data for the variables

	mean	sd
Price	**2288.24**	**2194.60**
log_price	**7.65**	**0.41**
diff_log_price	**0.00**	**0.02**
PRCP	**0.23**	**0.41**
TAVG	**81.09**	**2.82**
TMAX	**88.94**	**3.60**
TMIN	**74.06**	**74.00**

References

Ayvaz, B., & Kusakci, A. O. (2017). Electricity consumption forecasting for Turkey with nonhomogeneous discrete grey model. *Energy Sources, Part B: Economics, Planning and Policy, 12(3)*, 260-267. https://doi.org/10.1080/15567249.2015.1089337.

Bhavani, G., & Singh, A. K. (2018). ARIMA forecast of area, production and productivity of rice in amalgamated Raipur using R. *International Journal of Agricultural Science and Research (IJASR), 8(1)*, 129-136.

Gibrilla, A., Anornu, G., & Adomako, D. (2018). Trend analysis and ARIMA modelling of recent groundwater levels in the White Volta River basin of Ghana. *Groundwater for Sustainable Development, 6*, 150-163. https://doi.org/10.1016/j.gsd.2017.12.006.

González, C., Mira-McWilliams, J., & Juárez, I. (2015). Important variable assessment and electricity price forecasting based on regression tree models: Classification and regression trees, bagging and random forests. *IET Generation, Transmission & Distribution, 9*, 1120-1128.

Han, C., Wang, Y., & Xu, Y. (2020). Nonlinearity and efficiency dynamics of foreign exchange markets: Evidence from multifractality and volatility of major exchange rates. *Economic Research-Ekonomska Istraživanja, 33*, 731-751.

Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software, 27(3)*, 22. Available at: http://www.jstatsoft.org/v27/i03/paper

Ibe, O. C. (2013). *Markov processes for stochastic modeling*. Amsterdam: Elsevier.

Javed, S. A., & Liu, S. (2018). Predicting the research output/growth of selected countries: Application of Even GM(1,1) and NDGM models. *Scientometrics, 115(1)*, 395-413. https://doi.org/10.1007/s11192-017-2586-5

Kara, M., Atici, K. B., & Ulucan, A. (2021). Price and volatility forecasting in electricity with support vector regression and random forest. In *Applied Operations Research and Financial Modelling in Energy* (pp. 101-124). Cham, Switzerland: Springer.

Kongcharoen, C., & Kruangpradit, T. (2013). Autoregressive Integrated Moving Average with Explanatory Variable (ARIMAX) Model for Thailand Export.

Kumar, A., Jamadar, I., Goel, R., Petluri, R. C., & Feng, W. (2024). Mathematically forecasting stock prices with geometric Brownian motion. *The North Carolina Journal of Mathematics and Statistics, 10*, 1-14.

Minocha, S., Makkar, S., Swaminathan, S., Thomas, T., Webb, P., & Kurpad, A. V. (2019). Supply and demand of high-quality protein foods in India: Trends and opportunities. *Global Food Security, 23*, 139-148.

Mei, J., He, D., Harley, R., Habetler, T., & Qu, G. (2014). A random forest method for real-time price forecasting in New York electricity market. *Proc. IEEE PES General Meeting – Conf. Expo.*, 1-5.

Ohyver, M., & Pudjihastuti, H. (2018). ARIMA model for forecasting the price of medium quality rice to anticipate price fluctuations. *Procedia Computer Science, 135*, 707-711. https://doi.org/10.1016/j.procs.2018.08.215

Parag, S., Roy, M., & Pal, P. (2016). Application of ARIMA for forecasting energy consumption and GHG emission: A case study of an Indian pig iron manufacturing organization. *Energy, 116(Part 1)*, 1031-1038. https://doi.org/10.1016/j.energy.2016.10.068

Pirthee, M. (2017). Grey-based model for forecasting Mauritius international tourism from different regions. *Grey Systems: Theory and Application, 7(2)*, 259-271. https://doi.org/10.1108/gs-042017-0008

Praveen, B., & Sharma, P. (2019). Climate variability and its impacts on agriculture production and future prediction using autoregressive integrated moving average method (ARIMA). *Journal of Public Affairs, 20(2)*, e2016. https://doi.org/10.1002/pa.2016

Sen, P., Roy, M., & Pal, P. (2016). Application of ARIMA for forecasting energy consumption and GHG emission: A case study of an Indian pig iron manufacturing organization. *Energy, 116(Part 1)*, 1031-1038. https://doi.org/10.1016/j.energy.2016.10.068

Stergiou, K. I. (1991). Short-term fisheries forecasting: Comparison of smoothing, ARIMA, and regression techniques. *Journal of Applied Ichthyology, 7(4)*, 193-204.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NeurIPS*.

Voora, V., Bermúdez, S., & Larrea, C. (2019). Global Market Report: Cocoa. International Institute for Sustainable Development (IISD). https://www.iisd.org/publications/report/global-market-report-cocoa

Wu, L. F., Liu, S. F., Cui, W., Liu, D. L., & Yao, T. X. (2014). Non-homogeneous discrete grey model with fractional-order accumulation. *Neural Computing and Applications, 25(5)*, 1215-1221. https://doi.org/10.1007/s00521-014-1605-1

Xie, N. M., Liu, S. F., Yang, Y. J., & Yuan, C. Q. (2013). On novel grey forecasting model based on nonhomogeneous index sequence. *Applied Mathematical Modelling, 37(7)*, 5059-5068. https://doi.org/10.1016/j.apm.2012.10.037

Yuan, C., Liu, S., & Fang, Z. (2016). Comparison of China's primary energy consumption forecasting by using ARIMA (the autoregressive integrated moving average) model and GM(1,1) model. *Energy, 100*, 384-390. https://doi.org/10.1016/j.energy.2016.02.001

Z. Shi, Y. Hu, G. Mo, & J. Wu (2022). Attention-based CNN-LSTM and XGBoost hybrid model for stock prediction. *arXiv preprint arXiv:2204.02623*.

Codes

```
```{r setup, include=FALSE}
# install.packages("imputeTS")
# install.packages("randomForest")
# install.packages("imputeTS")
# install.packages("gbm")
# install.packages("scales")
# install.packages("forecast")
# install.packages("imputeTS")
# install.packages("glmnet")
# install.packages("rugarch")
# install.packages("mgcv")
# install.packages("psych")
# install.packages("corrplot")
# install.packages("reshape2")
library(reshape2)
library(corrplot)
library(mgcv)
library(tidyverse)
library(lubridate)
library(randomForest)
library(gbm)
library(ggplot2)
library(scales)
library(forecast)
library(imputeTS)
library(glmnet)
library(rugarch)
library(psych)
library(zoo)
library(Metrics)
```

Data preparation

```{r}
# Load
cocoa_prices <- read.csv("Daily Prices_ICCO.csv", stringsAsFactors = FALSE)
cocoa_prices$Price <- as.numeric(gsub(",", "", cocoa_prices$ICCO.daily.price..US..tonne.))
cocoa_prices$Date <- as.Date(cocoa_prices$Date, format='%d/%m/%Y')

# Clean
cocoa_prices <- cocoa_prices %>%
  select(Date, Price) %>%
  arrange(Date) %>%
  filter(!is.na(Price) & Price > 0)

# Compute log and differenced log
```
```

```
cocoa_prices <- cocoa_prices %>%
  mutate(
    log_price = log(Price),
    diff_log_price = c(NA, diff(log_price))
  )
```

```{r}
# Load
ghana_weather <- read.csv("Ghana_data.csv", stringsAsFactors = FALSE)
ghana_weather$DATE <- as.Date(ghana_weather$DATE)

ghana_weather <- ghana_weather %>%
  group_by(DATE) %>%
  summarise(
    PRCP = mean(PRCP, na.rm = TRUE),
    TAVG = mean(TAVG, na.rm = TRUE),
    TMAX = mean(TMAX, na.rm = TRUE),
    TMIN = mean(TMIN, na.rm = TRUE),
    .groups = "drop"
  )

# Clean
date_seq <- tibble(DATE = seq(min(ghana_weather$DATE), max(ghana_weather$DATE), by =
"day"))

# Join and fill missing days with NA
ghana_weather <- date_seq %>%
  left_join(ghana_weather, by = "DATE")

# Fill NA values using Kalman smoothing for daily series
ghana_weather <- ghana_weather %>%
  mutate(
    PRCP = na_kalman(PRCP, model = "StructTS", smooth = TRUE),
    TAVG = na_kalman(TAVG, model = "StructTS", smooth = TRUE),
    TMAX = na_kalman(TMAX, model = "StructTS", smooth = TRUE),
    TMIN = na_kalman(TMIN, model = "StructTS", smooth = TRUE)
  )

# Aggregate to monthly level and plot
cocoa_prices_monthly <- cocoa_prices %>%
  mutate(YearMonth = floor_date(Date, "month")) %>%
  group_by(YearMonth) %>%
  summarise(
    AvgPrice = mean(Price, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  mutate(
```

```
    log_price = log(AvgPrice),
    diff_log_price = c(NA, diff(log_price))
  )
```

Tables & Descriptive plots
```{r}
# Histogram of cocoa price
ggplot(cocoa_prices, aes(x = Price)) +
  geom_histogram(binwidth = 100, fill = "steelblue", color = "white") +
  labs(title = "Histogram of Cocoa Price", x = "Price (USD/tonne)", y = "Count") +
  theme_minimal()

# Histogram of log-transformed price
ggplot(cocoa_prices, aes(x = log_price)) +
  geom_histogram(binwidth = 0.1, fill = "darkorange", color = "white") +
  labs(title = "Histogram of Log Price", x = "Log(Price)", y = "Count") +
  theme_minimal()

# Histogram of precipitation
ggplot(ghana_weather, aes(x = PRCP)) +
  geom_histogram(binwidth = 0.1, fill = "skyblue", color = "white") +
  labs(title = "Histogram of Precipitation (PRCP)", x = "PRCP", y = "Count") +
  theme_minimal()

# Histogram of average temperature
ggplot(ghana_weather, aes(x = TAVG)) +
  geom_histogram(binwidth = 1, fill = "tomato", color = "white") +
  labs(title = "Histogram of Avg Temperature (TAVG)", x = "TAVG (°F)", y = "Count") +
  theme_minimal()

# Histogram of max temperature
ggplot(ghana_weather, aes(x = TMAX)) +
  geom_histogram(binwidth = 1, fill = "seagreen", color = "white") +
  labs(title = "Histogram of Max Temperature (TMAX)", x = "TMAX (°F)", y = "Count") +
  theme_minimal()

# Histogram of min temperature
ggplot(ghana_weather, aes(x = TMIN)) +
  geom_histogram(binwidth = 1, fill = "purple", color = "white") +
  labs(title = "Histogram of Min Temperature (TMIN)", x = "TMIN (°F)", y = "Count") +
  theme_minimal()
```

Merge
```{r}
```

```
cocoa_data <- left_join(cocoa_prices, ghana_weather, by = c("Date" = "DATE")) %>%
mutate(log_price = log(Price),
       diff_log_price = c(NA, diff(log_price))) %>% na.omit()
```

Plotting

```{r}
diff_log_ts <- diff(log(cocoa_data$Price))

ggplot(cocoa_data, aes(x = Date, y = Price)) +
  geom_line(color = "blue", size = 0.8) +
  scale_y_continuous(labels = dollar_format(prefix = "$")) +
  labs(title = "Daily Cocoa Prices", x = NULL, y = "Price (USD)") +
  theme_minimal(base_size = 13)

ggplot(cocoa_data, aes(x = Date, y = log_price)) +
  geom_line(color = "darkgreen", size = 0.8) +
  labs(title = "Log Transformed Cocoa Prices", x = NULL, y = "Log(Price)") +
  theme_minimal(base_size = 13)

plot(cocoa_prices_monthly$YearMonth, cocoa_prices_monthly$diff_log_price,
     type = "l", col = "blue",
     main = "Differenced Monthly Log Cocoa Prices",
     ylab = "Diff Log(Price)", xlab = "Date")

# ACF plot
acf(diff_log_ts, main = "ACF of Differenced Log Cocoa Prices", col = "blue", lwd = 2)

# PACF plot
pacf(diff_log_ts, main = "PACF of Differenced Log Cocoa Prices", col = "darkgreen", lwd = 2)

# Month-wise price trend
cocoa_prices_monthly$Month <- month(cocoa_prices_monthly$YearMonth, label = TRUE)
ggplot(cocoa_prices_monthly, aes(x = Month, y = AvgPrice)) +
  geom_boxplot(fill = "skyblue") +
  labs(title = "Seasonal Trend of Cocoa Prices (Monthly)", x = "Month", y = "Avg Price") +
  theme_minimal()

# Volatility plot
cocoa_prices_monthly$abs_diff <- abs(cocoa_prices_monthly$diff_log_price)
ggplot(cocoa_prices_monthly, aes(x = YearMonth, y = abs_diff)) +
  geom_line(color = "orange", size = 0.7) +
  labs(title = "Volatility of Monthly Cocoa Prices", y = "|Δlog(Price)|", x = "Month") +
  theme_minimal()

# STL Decomposition
ts_log_price <- ts(cocoa_prices_monthly$log_price, frequency = 12, start =
c(year(min(cocoa_prices_monthly$YearMonth)), month(min(cocoa_prices_monthly$YearMonth))))
```

```
decomp <- stl(ts_log_price, s.window = "periodic")
autoplot(decomp) +
  labs(title = "STL Decomposition of Monthly Log Cocoa Prices") +
  theme_minimal()

# Seasonal plot
ggseasonplot(ts_log_price, year.labels = TRUE, year.labels.left = TRUE) +
  labs(title = "Seasonal Plot of Monthly Log Cocoa Prices", y = "Log(Price)", x = "Month") +
  theme_minimal()

# Subseries plot
ggsubseriesplot(ts_log_price) +
  labs(title = "Subseries Plot of Monthly Log Cocoa Prices", y = "Log(Price)", x = "Month") +
  theme_minimal()
```

Some addtional plots

```{r}
# Correlation Plot
num_vars <- cocoa_data %>% select(where(is.numeric))
corrplot(cor(num_vars, use = "complete.obs"), method = "color", type = "upper", tl.cex = 0.8)

# Missing Value Heatmap
ggplot(cocoa_data, aes(x = Date)) +
  geom_point(aes(y = Price), alpha = 0.3) +
  labs(title = "Price Data Over Time (Missing Check)", y = "Price") +
  theme_minimal()

# Smoothed Trend
ggplot(cocoa_data, aes(x = Date, y = Price)) +
  geom_line(alpha = 0.4) +
  geom_smooth(method = "loess", color = "red") +
  labs(title = "Smoothed Trend of Cocoa Prices", x = "Date", y = "Price") +
  theme_minimal()
```

Split Data into Training and Testing Sets

```{r}
set.seed(6657)
train_size <- floor(0.8 * nrow(cocoa_data))
train_data <- cocoa_data[1:train_size, ]
test_data <- cocoa_data[(train_size + 1):nrow(cocoa_data), ]
```

Choosing predictors

```{r}
```

```
cor(cocoa_data[, c("log_price", "diff_log_price", "TAVG", "PRCP", "TMAX", "TMIN")], use =
"complete.obs")
```
Volatility Check Plot
```{r}
# Compute log returns
log_returns <- diff(log(cocoa_data$Price))
log_returns <- na.omit(log_returns)
dates <- cocoa_data$Date[2:(length(log_returns) + 1)]

# Plot log returns
plot(dates, log_returns, type = "l", col = "darkblue",
    main = "Log Returns of Cocoa Prices",
    xlab = "Date", ylab = "Log Return")

# Plot squared log returns (volatility clustering check)
squared_returns <- log_returns^2
plot(dates, squared_returns, type = "l", col = "firebrick",
    main = "Squared Log Returns (Check for Volatility Clustering)",
    xlab = "Date", ylab = expression(r[t]^2))

# ACF of squared returns
acf(squared_returns, main = "ACF of Squared Log Returns")

# Ljung-Box test on squared returns
ljung_result <- Box.test(squared_returns, lag = 10, type = "Ljung-Box")
print(ljung_result)
```

ARIMA model
```{r}
ts_data <- ts(cocoa_prices_monthly$AvgPrice, start = c(1994, 10), frequency = 12)

arima_model <- Arima(ts_data, order = c(0,1,1), include.mean = TRUE)

# Extract fitted values
fitted_values <- fitted(arima_model)
plot(ts_data, main = "Observed vs. Fitted Values (ARIMA(0,1,1))", col = "blue", type = "l", ylab =
"Diff  Price", xlab = "Time")

# Add the fitted values in red
lines(fitted_values, col = "red", lwd = 2)

# Add legend
legend("topright", legend = c("Observed", "Fitted"), col = c("blue", "red"), lty = 1, lwd = 2)

arima_model <- Arima(ts_data, order = c(0,1,1), include.mean = TRUE)
```

```
# Forecast next 4 months
forecast_values <- forecast(arima_model, h = 4)

# Get time index
time_index <- time(ts_data)

# Filter data from 2024 onward
start_year <- 2024
ts_subset <- window(ts_data, start = c(start_year, 1))
fitted_subset <- window(fitted(arima_model), start = c(start_year, 1))

# Create future time points for forecast (monthly assumed)
future_time <- time(ts_data)[length(ts_data)] + seq(1, 4)/12

# Plot observed data (since 2020)
plot(ts_subset,
     main = "Observed, Fitted & Forecasted Values (ARIMA(0,1,1)) Since 2024",
     col = "blue", lwd = 2, ylab = "Price", xlab = "Time",
     xlim = c(start_year, future_time[4]), ylim = range(c(ts_subset, forecast_values$mean,
forecast_values$lower[,2], forecast_values$upper[,2])))

# Add fitted values
lines(fitted_subset, col = "red", lwd = 2)

# Add forecasted values
lines(future_time, as.numeric(forecast_values$mean), col = "green", lwd = 2, lty = 2)

# Add 95% confidence interval
lines(future_time, forecast_values$lower[,2], col = "black", lty = 5)
lines(future_time, forecast_values$upper[,2], col = "black", lty = 5)

# Add legend
legend("topleft",
       legend = c("Observed (since 2024)", "Fitted", "Forecast", "95% CI"),
       col = c("blue", "red", "green","black"),
       lwd = 2, lty = c(1, 1, 2, 3))

```

GAM model

```{r}
# Create lag features
cocoa_lagged <- cocoa_data |>
  mutate(
    lag_1 = lag(log_price, 1),
    lag_2 = lag(log_price, 2),
    date_int = as.numeric(Date)  # Convert Date to numeric for smoothing
```

```
  ) |>
  filter(
    !is.na(log_price),
    !is.na(lag_1),
    !is.na(lag_2),
    !is.na(TAVG),
    !is.na(TMAX)
  )

# Split the data
set.seed(8848)
split_index <- floor(0.8 * nrow(cocoa_lagged))
data_train <- cocoa_lagged[1:split_index, ]
data_test <- cocoa_lagged[(split_index + 1):nrow(cocoa_lagged), ]

# Fit GAM model
my_gam <- gam(log_price ~ s(date_int) + lag_1 + lag_2 + TAVG + TMAX,
              data = data_train, method = "ML")

h <- 3000
last_row <- tail(data_train, 1)
future_forecast <- tibble(
  Date = seq(last_row$Date + 1, by = "days", length.out = h),
  lag_1 = NA_real_,
  lag_2 = NA_real_,
  TAVG = rep(mean(data_train$TAVG, na.rm = TRUE), h),
  TMAX = rep(mean(data_train$TMAX, na.rm = TRUE), h),
  date_int = as.numeric(seq(last_row$Date + 1, by = "days", length.out = h))
)

# Initialize lags
lag_1 <- last_row$log_price
lag_2 <- last_row$lag_1
log_forecasts <- numeric(h)

# walk-forward
for (i in 1:h) {
  new_data <- tibble(
    date_int = future_forecast$date_int[i],
    lag_1 = lag_1,
    lag_2 = lag_2,
    TAVG = future_forecast$TAVG[i],
    TMAX = future_forecast$TMAX[i]
  )
  pred_log <- predict(my_gam, newdata = new_data)
  log_forecasts[i] <- pred_log
  lag_2 <- lag_1
  lag_1 <- pred_log
```

```
}

future_forecast$Price <- exp(log_forecasts)
future_forecast$Model <- "GAM"

ggplot() +
  geom_line(data = cocoa_data, aes(x = Date, y = Price), color = "black") +
  geom_line(data = future_forecast, aes(x = Date, y = Price, color = Model), linewidth = 1.2) +
  scale_color_manual(values = c("GAM" = "orange")) +
  labs(
    title = "GAM Forecast vs Actual Prices (Walk-Forward Forecast)",
    x = "Date",
    y = "Price",
    color = "Model"
  ) +
  theme_minimal()
```

GARCH and ARCH

```{r}
# Compute log returns
returns <- cocoa_data %>%
  mutate(log_return = log(Price) - lag(log(Price))) %>%
  drop_na()

# Fit GARCH(1,1) model
garch <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "norm"
)

fit_garch <- ugarchfit(spec = garch, data = returns$log_return)

g_1 <- tibble(
  Date = returns$Date,
  g_1 = as.numeric(sigma(fit_garch))
)
r_1 <- rollapply(abs(returns$log_return),
                 width = 5, FUN = sd,
                 align = "right", fill = NA)

r_df <- tibble(
  Date = returns$Date,
  r_1 = r_1
) %>% drop_na()

vol_df <- g_1 %>%
```

```
  inner_join(r_df, by = "Date")

# Plot
ggplot(vol_df, aes(x = Date)) +
  geom_line(aes(y = g_1, color = "GARCH Estimated"), linewidth = 0.8) +
  geom_line(aes(y = r_1, color = "Realized (Rolling SD)"), linewidth = 0.8) +
  scale_color_manual(values = c("GARCH Estimated" = "red", "Realized (Rolling SD)" = "yellow"))
+
  labs(
    title = "GARCH Estimated vs Actual Volatility",
    x = "Date",
    y = "Volatility",
    color = "Type"
  ) +
  theme_minimal()

### Evaluations

# Standardized residuals
resid_std <- residuals(fit_garch, standardize = TRUE)

# Coefficient table
fit_garch@fit$matcoef

# Stationarity check
alpha1 <- fit_garch@fit$coef["alpha1"]
beta1 <- fit_garch@fit$coef["beta1"]
sum_ab <- alpha1 + beta1
cat("alpha + beta =", round(sum_ab, 4), "\n")

if (sum_ab < 1) {
  cat("Stationary: yes\n")
} else {
  cat("Stationary: no\n")
}

# Ljung-Box test
Box.test(resid_std, lag = 10, type = "Ljung-Box")
Box.test(resid_std^2, lag = 10, type = "Ljung-Box")
```

Walk-Forward Lag Selection for RF and GBM Forecasting

```{r}
# This section tunes the number of lag features (1 to 5) for forecasting cocoa prices.
# It uses repeated walk-forward validation on the last 150 observations of the dataset.
# For each lag setting, it fits both a Random Forest and a GBM model to predict
# the difference in log prices (diff_log_price), then transforms predictions back
```

```
# to raw price scale to calculate RMSE. The average RMSE across repetitions is used.

# Creating lag 1-5
create_lags <- function(data, lags = 1:7) {
  set.seed(10086)
  lag_vars <- lapply(lags, function(l) dplyr::lag(data$log_price, l))
  names(lag_vars) <- paste0("lag_", lags)
  lag_df <- as.data.frame(lag_vars)
  dplyr::bind_cols(data, lag_df)
}

# RMSE holder
rmse_results <- data.frame(Lags = integer(), RF_RMSE = numeric(), GBM_RMSE = numeric())

# last 150 rows for tuning
cocoa_data_short <- cocoa_data[(nrow(cocoa_data) - 150):nrow(cocoa_data), ]

N_rep <- 5
for (lag_n in 1:5) {
  rmse_rf_vec <- c()
  rmse_gbm_vec <- c()

  for (rep in 1:N_rep) {
    set.seed(100 + rep)
    cocoa_data_lagged <- create_lags(cocoa_data_short, lags = 1:lag_n)
    cocoa_data_lagged <- cocoa_data_lagged %>%
      mutate(diff_log_price = c(NA, diff(log_price))) %>%
      drop_na()

    model_data <- cocoa_data_lagged
    start_index <- nrow(model_data) - 30
    pred_rf <- c()
    pred_gbm <- c()
    actual <- model_data$Price[(start_index+1):nrow(model_data)]

    for (i in start_index:(nrow(model_data) - 1)) {
      train_data <- model_data[1:i, ]
      test_data <- model_data[i+1, ]

      predictors <- c(paste0("lag_", 1:lag_n), "TAVG", "TMAX")
      formula <- as.formula(paste("diff_log_price ~", paste(predictors, collapse = "+")))

      set.seed(42 + rep)
      rf_model <- randomForest(formula, data = train_data)

      set.seed(42 + rep)
      gbm_model <- gbm(formula, data = train_data, distribution = "gaussian",
                       n.trees = 100, interaction.depth = 3, shrinkage = 0.05,
```

```
                bag.fraction = 1, verbose = FALSE)

    rf_pred <- predict(rf_model, newdata = test_data)
    gbm_pred <- predict(gbm_model, newdata = test_data, n.trees = 100)

    last_log_price <- log(model_data$Price[i])
    rf_back <- exp(last_log_price + rf_pred)
    gbm_back <- exp(last_log_price + gbm_pred)

    pred_rf <- c(pred_rf, rf_back)
    pred_gbm <- c(pred_gbm, gbm_back)
   }
   rmse_rf_vec <- c(rmse_rf_vec, sqrt(mean((pred_rf - actual)^2)))
   rmse_gbm_vec <- c(rmse_gbm_vec, sqrt(mean((pred_gbm - actual)^2)))
  }
  rmse_results <- rbind(rmse_results,
                  data.frame(Lags = lag_n,
                        RF_RMSE = mean(rmse_rf_vec),
                        GBM_RMSE = mean(rmse_gbm_vec)))
}

# Find best lag based on average RMSE
best_rf_lag <- rmse_results$Lags[which.min(rmse_results$RF_RMSE)]
best_gbm_lag <- rmse_results$Lags[which.min(rmse_results$GBM_RMSE)]
```

```{r}
# Plotting results
results_long <- melt(rmse_results, id.vars = "Lags", variable.name = "Model", value.name =
"RMSE")
ggplot(results_long, aes(x = Lags, y = RMSE, color = Model)) +
  geom_line() +
  geom_point() +
  labs(title = "Average RMSE by Number of Lags (Repeated Walk-Forward)",
       x = "Number of Lags", y = "Average RMSE")
```

Expanding Window Cross-Validation Function (RF_model1 and GBM_model1)

```{r}
# This function performs expanding window cross-validation for time series forecasting.
# It takes in lagged features and fits either a Random Forest or GBM model.
# The model is retrained in each iteration as more data becomes available.
# Forecasts are made for one step ahead repeatedly and predictions are returned
# in a tibble with date, actual price, and predicted price.

expanding_window_cv <- function(data, model_type = "rf", lag = 5, step = 7) {
  step <- as.numeric(step)
```

```
  if (nrow(data) < 30) {
    stop("No enough data") # check enough
  }

  preds <- c()
  actuals <- c()
  dates <- c()

  init_size <- floor(0.8 * nrow(data))
  formula <- as.formula(paste("log_price ~", paste0("lag_", 1:lag, collapse = " + "), " + TAVG +
TMAX"))

  for (i in seq(init_size, nrow(data) - 1, by = step)) {
    train <- data[1:i, ]
    test <- data[i + 1, , drop = FALSE]

    if (any(is.na(test)) || any(is.na(train))) next

    set.seed(7727)

    model <- if (model_type == "rf") {
      randomForest(formula, data = train, ntree = 50)
    } else {
      gbm(formula, data = train, distribution = "gaussian",
          n.trees = 30, shrinkage = 0.05, interaction.depth = 3,
          bag.fraction = 1, verbose = FALSE)
    }
    pred_log <- if (model_type == "rf") {
      predict(model, test)
    } else {
      predict(model, test, n.trees = 30)
    }
    preds <- c(preds, exp(pred_log))
    actuals <- c(actuals, test$Price)
    dates <- c(dates, test$Date)
  }

  return(tibble(Date = dates, Actual = actuals, Predicted = preds))
}

final_lagged_data <- create_lags(cocoa_data, lags = 1:max(best_rf_lag, best_gbm_lag)) %>%
  mutate(log_price = log(Price)) %>%
  drop_na()
```

Final RF and GBM Forecasting Using Optimal Lag Variables
*** May take 10 to 15 mins *** :(
```{r}
```

```
# Run RF and GBM forecasts using best lag values(10 mins)
rf_df <- expanding_window_cv(final_lagged_data, model_type = "rf", lag = best_rf_lag, step = 7)
gbm_df <- expanding_window_cv(final_lagged_data, model_type = "gbm", lag = best_gbm_lag, step
= 7)
```

Evaluation
```{r}
# Plot actual vs predicted
plot_df <- bind_rows(
  rf_df %>% mutate(Model = "RF"),
  gbm_df %>% mutate(Model = "GBM")
)
ggplot() +
  geom_line(data = plot_df, aes(x = Date, y = Actual), color = "black") +
  geom_line(data = plot_df, aes(x = Date, y = Predicted, color = Model), linewidth = 0.6) +
  labs(title = "Expanding Window Forecast (RF & GBM)", x = "Date", y = "Price") +
  theme_minimal()

#Residuals
plot_df <- bind_rows(
  rf_df %>% mutate(Model = "RF", Residual = Actual - Predicted),
  gbm_df %>% mutate(Model = "GBM", Residual = Actual - Predicted)
)

ggplot(plot_df, aes(x = Date, y = Residual, color = Model)) +
  geom_line() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(title = "Prediction Residuals Over Time", y = "Residual (Actual - Predicted)", x = "Date") +
  theme_minimal()

ggplot(plot_df, aes(x = Actual, y = Predicted, color = Model)) +
  geom_point(alpha = 0.4) +
  geom_abline(slope = 1, intercept = 0, color = "black", linetype = "dashed") +
  labs(title = "Actual vs Predicted Prices", x = "Actual Price", y = "Predicted Price") +
  theme_minimal()

# RF performance
rf_rmse <- rmse(rf_df$Actual, rf_df$Predicted)
rf_mae <- mae(rf_df$Actual, rf_df$Predicted)
rf_mape <- mape(rf_df$Actual, rf_df$Predicted)

# GBM performance
gbm_rmse <- rmse(gbm_df$Actual, gbm_df$Predicted)
gbm_mae <- mae(gbm_df$Actual, gbm_df$Predicted)
gbm_mape <- mape(gbm_df$Actual, gbm_df$Predicted)

results <- tibble(
```

```
  Model = c("RF", "GBM"),
  RMSE = c(rf_rmse, gbm_rmse),
  MAE = c(rf_mae, gbm_mae),
  MAPE = c(rf_mape, gbm_mape)
)

# Residual Histogram
ggplot(plot_df, aes(x = Residual, fill = Model)) +
  geom_histogram(alpha = 0.6, position = "identity", bins = 30) +
  geom_vline(xintercept = 0, linetype = "dashed") +
  labs(title = "Residual Distribution of RF and GBM Forecasts",
       x = "Residual (Actual - Predicted)",
       y = "Count") +
  theme_minimal()
print(results)

acf(rf_df$Actual - rf_df$Predicted, main = "ACF of RF Forecast Residuals")
```

```{r}
# RMSE plot reliability
rf_results <- rf_df %>%
  mutate(Error = Predicted - Actual,
         SE = Error^2,
         Step = row_number(),
         RMSE = sqrt(SE))

ggplot(rf_results, aes(x = Step, y = RMSE)) +
  geom_line(color = "steelblue") +
  geom_point(size = 1) +
  labs(title = "Walk-forward RMSE of Random Forest Forecasts",
       x = "Forecast Step",
       y = "RMSE") +
  theme_minimal()
```

Forecast Next 2000 Days Using Random Forest (walk-forward)

```{r}
predict_next_days_rf <- function(data, model_lag = 5, days_ahead = 90, ntree = 50) {
  if (nrow(data) < 30 || !"log_price" %in% names(data)) {
    stop("Data must contain log_price and be at least 30 rows.")
  }

  last <- tail(data, 1)

  rf_formula <- as.formula(
    paste("log_price ~", paste0("lag_", 1:model_lag, collapse = " + "), " + TAVG + TMAX")
  )
```

```
  rf_model <- randomForest(
    formula = rf_formula,
    data = data,
    ntree = ntree
  )

  future_preds <- tibble()
  current_input <- last

  for (i in 1:days_ahead) {
    pred_log <- predict(rf_model, newdata = current_input)
    pred_price <- exp(pred_log)

    new_row <- current_input %>%
      mutate(
        Date = current_input$Date + days(1),
        Price = pred_price,
        log_price = pred_log
      )

    for (j in model_lag:2) {
      new_row[[paste0("lag_", j)]] <- current_input[[paste0("lag_", j - 1)]]
    }
    new_row[["lag_1"]] <- pred_log

    new_row$TAVG <- current_input$TAVG + rnorm(1, mean = 0, sd = 0.5)
    new_row$TMAX <- current_input$TMAX + rnorm(1, mean = 0, sd = 0.5)

    future_preds <- bind_rows(future_preds, new_row)
    current_input <- new_row
  }

  return(future_preds)
}
```

```{r}
set.seed(1314)
future_rf_2000 <- predict_next_days_rf(
  data = final_lagged_data,
  model_lag = best_rf_lag,
  days_ahead = 2000
)
```

```{r}
```

```
plot_df <- bind_rows(
  final_lagged_data %>% select(Date, Price) %>% mutate(Type = "Observed"),
  future_rf_2000  %>% select(Date, Price) %>% mutate(Type = "Forecasted")
)

ggplot(plot_df, aes(x = Date, y = Price, color = Type)) +
  geom_line(size = 0.9) +
  geom_vline(xintercept = max(final_lagged_data$Date), linetype = "dashed", color = "black") +
  scale_color_manual(values = c("Observed" = "#1f77b4", "Forecasted" = "#e41a1c")) +
  labs(title = "2000-Day Forecast (RF Model)", x = "Date", y = "Price (USD)") +
  theme_minimal()
```