STA457 Final Project

Abtin Kit Shirvani, Siyu Li, Sungmin Lee, Yuxin Zhong

March, 2025

1 Introduction

Cocoa is one of the world's most important agricultural products, and its market value plays an important role in the economies of major producers. For example, Ghana contributes about 17 percent of the world's cocoa production, so accurate price predictions are crucial in national planning, policy development, and trade strategies.[1] However, cocoa prices are characterized by a combination of market volatility and external shocks such as climate change. Our study focuses on modelling and forecasting future cocoa prices using classical time series techniques, particularly the ARIMA and SARIMA models. We work with a detailed dataset from the International Cocoa Organization (ICCO), which contains 5,903 records of daily cocoa prices. Over the covered period, prices varied between \$720 and \$3,775 per tonne, with the average price settling around \$2,365. To improve the reliability of our forecasts and account for potential environmental influences, we also incorporate daily climate data from Ghana obtained through the National Centers for Environmental Information (NCEI). Rainfall and temperature significantly impact the growth of the cacao plant; therefore, we will treat these two factors as external variables and use them to predict cacao prices. In selecting the optimal prediction model, comparing the features of each model will be considered crucial, as it is unlikely to find a model that perfectly satisfies all assumptions based solely on our data. Our objective is to evaluate the predictive power of ARIMA, SARIMA, and other potential models, as well as to assess how climate variables influence cocoa price forecasts. Ultimately, the results of this study aim to aid in predicting market fluctuations and further contribute to developing an effective pricing strategy.

2 Literature Review

Forecasting commodity prices is a topic that has been widely studied in the field of time series modeling. Sukiyono et al. (2018) compared three classical univariate forecasting techniques- ARIMA, Exponential Smoothing, and Decomposition- to model cocoa prices in Indonesia.[2]. According to their research, they concluded that the ARIMA model consistently showed the best results in terms of prediction accuracy based on MAPE, MAD, and MSD. This means that the ARIMA model is capable of effectively identifying and predicting unique features that appear in commodity price time series data. Kamu et al. (2010) explored four univariate models- exponential Smoothing, ARIMA, GARCH, and a hybrid ARIMA-GARCH - for predicting cocoa prices in Malaysia cite CocoaBeanPricesKamu. They documented that the GARCH model showed a slight advantage in short-term prediction, but the ARIMA(3,1,2) model showed superior results in terms of overall predictive power and interpretive power. They also noted the importance of first differencing to ensure the stationarity. Building on these studies, our analysis applies ARIMA, SARIMA and GARCH models to predict future cocoa prices. The final model will be selected based on a combination of forecast accuracy metrics and diagnostic tests, reflecting a careful adaptation of the techniques supported by the literature. Zhang et al. (2023) analyzed the systemic interdependence of the U.S. commodity markets and demonstrated that climate factors such as precipitation, temperature, and solar radiation play a significant role in influencing commodity price dynamics.[3] Using wavelet coherence analysis, this study identified a time-varying association between weather conditions and variability in commodity markets. It provides the basis for incorporating exogenous weather variables, specifically precipitation and temperature regression variables, into the model to improve forecasting performance.

3 Methodology

This section outlines the complete process of preparing the data for forecasting, including data cleaning, merging, aggregation, and final transformation into a time series format. We also describe the selection and configuration

of ARIMA, SARIMA, and GARCH models used in this study.

3.1 Data pre-processing

To prepare the data for forecasting, we conducted several preprocessing steps involving cleaning, transformation, merging, and aggregation. These steps ensured that the input to our time series models was consistent, interpretable, and aligned with our forecasting goals.

3.1.1 Cleaning and Structuring Raw Data

We began with two primary datasets:

- **ICCO Cocoa Prices** (`Daily Prices_ICCO.csv`): This dataset contains daily international cocoa prices in US dollars per tonne.
- **Ghana Climate Data** (`Ghana_data.csv`): This includes daily climate observations from multiple stations in Ghana, such as precipitation (PRCP), average temperature (TAVG), maximum temperature (TMAX), and minimum temperature (TMIN).

In the ICCO dataset, the date column was converted from character to `Date` format using the `dmy()` function from the `lubridate` package. Price values, initially formatted as character strings with embedded commas, were converted to numeric after the formatting symbols were removed. After conversion, the dataset was sorted chronologically. For the Ghana climate data, the date column was similarly parsed using `ymd()`. All climate variables were explicitly converted to numeric type to prevent downstream errors in aggregation or modeling. Based on the dataset documentation, blank precipitation values were interpreted as no rainfall and replaced with zeros. Other missing climate values were left as `NA` to allow for controlled interpolation during forecasting. Because the Ghana dataset included multiple stations per day, we averaged all station values by date to obtain a single observation per variable.

3.1.2 Merging Datasets

After cleaning both datasets, we merged them using a `left_join()` on the `Date` field. This ensured that all cocoa price observations were preserved and matched with any available climate data. Dates where climate observations were unavailable remained as `NA` in the merged dataset, which we retained intentionally to control how missing values were treated for modelling.

3.1.3 Monthly Aggregation and Forecast Range Handling

The merged dataset was divided into two distinct periods:

- **Historical (Training) Period:** October 1994 to November 2024
- **Forecast Period:** December 2024 to February 2025

For the historical period, we dropped all rows containing `NA` values to ensure that monthly averages were calculated only using complete daily records. These clean observations were then aggregated into monthly averages using the `floor_date()` function to group by month and the `mean()` function to compute average values for each variable. For the forecast period, where daily Ghana climate data was not available, we applied **seasonal average interpolation**. Specifically, we calculated the historical average for each day of the year (1 to 366) using the pre-2024 data. These seasonal averages were then mapped onto the future dates from November 29, 2024, to February 27, 2025, by matching the day of the year. Cocoa prices for these dates were included where available from the ICCO dataset. Both the historical and forecast period monthly averages were then combined into a single dataset. To prevent overlapping months (e.g., November 2024 appearing twice), we applied trimming to the last month from the historical section before appending the forecasted values.

3.1.4 Handling Missing Months in Monthly Aggregation

During the monthly aggregation process, we observed that some months were entirely missing from the dataset. This occurred when no valid daily climate data was available in a given month, resulting in those months being excluded during the group-wise averaging step. To preserve the continuity of the time series, we constructed a complete sequence of monthly dates from the first available observation (October 1994) to the last (February 2025). We then performed a left join between the full monthly date sequence and the aggregated data, ensuring

that all months were present in the dataset, even if their values were initially missing. Missing monthly values were subsequently imputed using linear interpolation. This approach helped maintain a continuous, equally spaced time series that could be used directly for modelling with ARIMA and ETS methods.

3.1.5 Final Time Series Construction

The final dataset contained one row per month from October 1994 to February 2025. The primary target variable, the monthly average cocoa price, was converted into a time series object using R's `ts()` function with a frequency of 12 to reflect monthly seasonality. This object served as the input to both the ETS and ARIMA models.

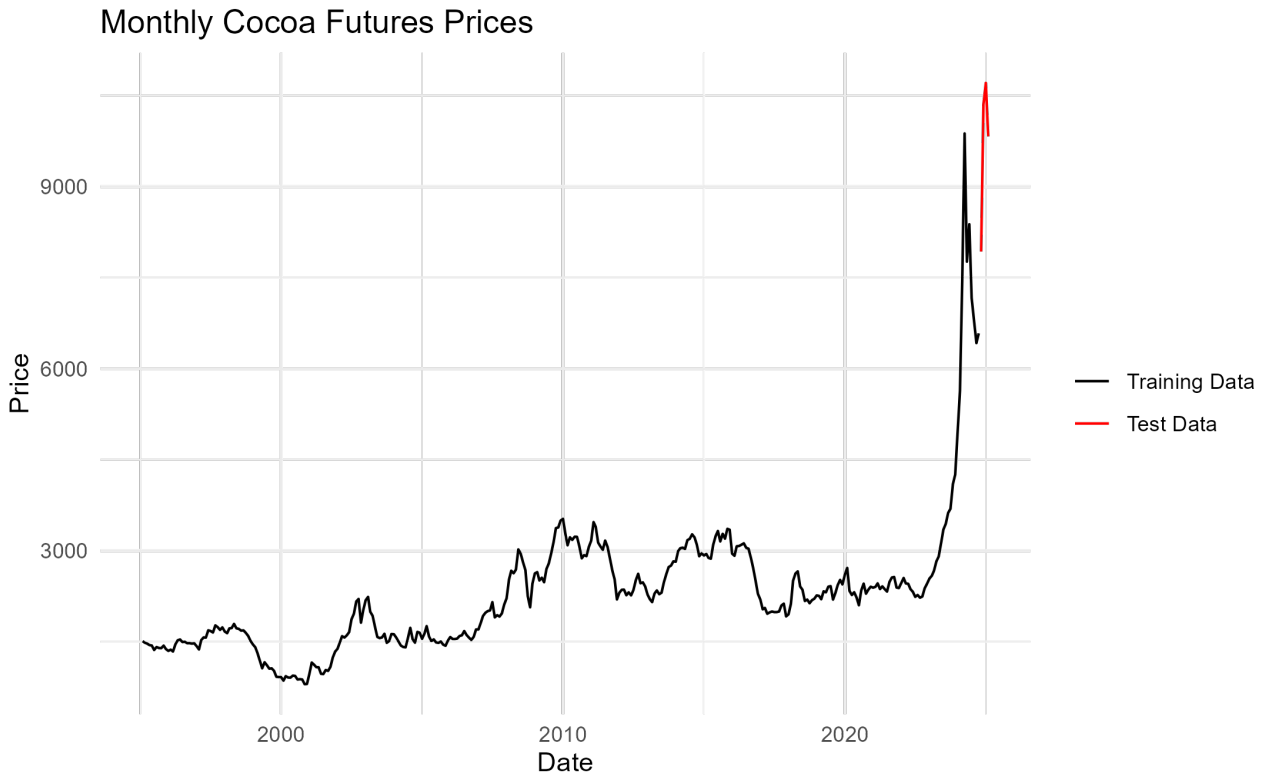


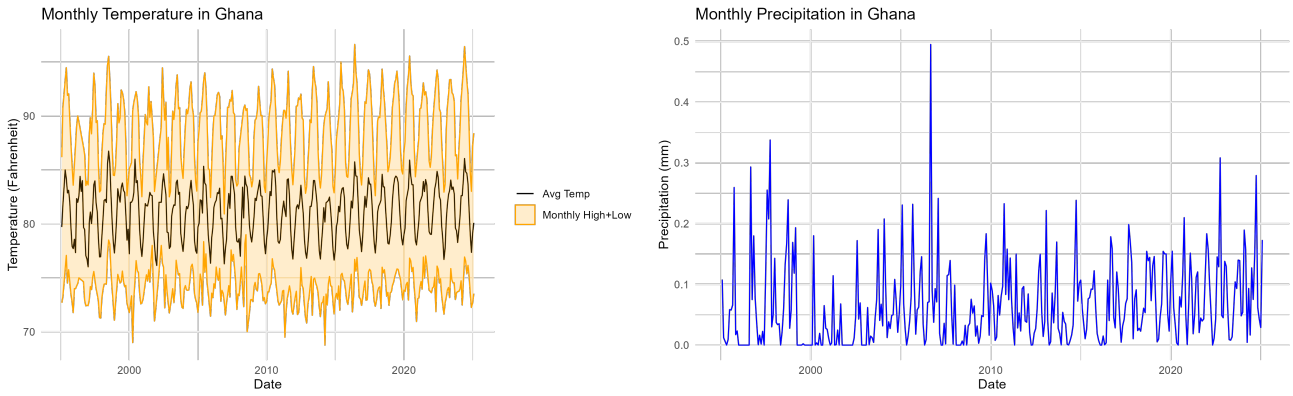
Figure 1: Monthly Cocoa Future Prices split into train and test data.

3.2 Modelling

We used univariate time series models (i.e. ARIMA, SARIMA, GARCH) to forecast future cocoa prices as they have been shown to yield adequate predictions.[4;2] ARIMA parameters (p,d,q) were evaluated based on patterns in the ACF and PACF plots, along with the Augmented Dickey-Fuller (ADF) Test for stationarity. The residuals of the candidate ARIMA models were checked to follow $IID\ N(0, \sigma_w^2)$ using standardized residual plots over time, QQ plots, ACF plots, and the Ljung-Box test. The seasonality parameters (P, D, Q) of the SARIMA models were derived similarly from the patterns in the ACF and PACF plots and the seasonal unit root tests using the `nsdiffs()` function from the `forecast` package. Seasonality (S) was set to 12 based on past literature as well as the fact that, intuitively, one would assume similar patterns occur year over year.[2] Various combinations of exogenous regressors from the Ghana climate dataset were incorporated into the candidate models. Exogenous regressors were lagged by 4 months to avoid data leakage and allow for up to 4-month forecasts. Models were assessed on prediction accuracy using out-of-sample RMSE, MAE, and MAPE. We explored a log transformation of the price to stabilize variance and satisfy residual assumptions. GARCH was applied to the residuals of the best candidate SARIMA model due to heteroskedasticity. GARCH parameters (p,q) as well as extensions of Standard GARCH (i.e. apARCH, eGARCH, fGARCH) were considered and compared using the Ljung-Box Test on standardized residuals and sign bias to assess how well the residual assumptions were satisfied.

4 Data

In this study, we exclusively use data from the ICCO Cocoa Price and Ghana Climate Data, with no additional external sources. It provides daily data from October 1994 to Feburary 2025. For ICCO Cocoa price, the maximum recorded value is 11984.66 USD, and the minimum price is 774.1 USD, resulting in a wide fluctuation of over 11,000 USD. To simplify the data and facilitate forecasting, we computed monthly average prices, which have a maximum value of 10709.31 USD and a minimum price of 792.765 USD. Ghana's climate dataset contains records of daily temperature and precipitation. Instead of listing all temperature-related variables one by one, graphs were used to represent the average daily temperature and the range together. Through this, we visualized the central trend and variability over time at a glance. As observed in Figure 2a, the average temperature shows a strong seasonal pattern. It remains relatively stable between 75°F and 87°F throughout the year but tends to fluctuate significantly over time. The precipitation pattern shows sharp fluctuations and irregular flows, as shown in Figure 2b. This suggests high-frequency variability and intermittent outliers. These features are important factors to consider carefully when modelling such a framework. There are null values in the climate data. We observed 19,018 missing values for minimum temperature and 18,368 missing values for maximum temperature. These visualizations will help understand more clearly the temporal variability and variability of climate variables associated with cocoa price prediction models.



(a) Average Temperature w/ highs and lows

(b) Average Precipitation

Figure 2: Monthly Ghana Climate Data

5 Forecasting and Results

This section presents the process and results of forecasting monthly cocoa prices using various time series models. We begin by outlining the model training procedure, followed by performance evaluation based on standard metrics. We then analyze forecast patterns and assess model accuracy by comparing predictions to actual observed prices.

5.1 Model Training and Validation

After cleaning and merging the datasets, we split the data into training and testing sets. As shown in Figure 1, there is a sharp increase in cocoa prices after 2020, rising from approximately 2250 to nearly 10000 in US dollars per tonne. Therefore, a standard 80 : 20 split would place this steep price surge entirely in the testing set, causing the training set to lack information about the recent market shift and potentially leading to poor model performance. To address this, we designated the last four observations as the testing dataset. This approach ensures that the training set includes recent trends, while the testing set provides a meaningful basis for evaluating model performance using the metrics discussed in the next section.

5.2 Model Evaluation and Performance Metrics

The metrics we chose for evaluating the performance of predicting are RMSE, MAE, and MAPE. RMSE measures the root mean square errors, MAE measures the mean absolute errors, and MAPE measures the mean

Table 1: Out-of-Sample Prediction Accuracy of Top Candidate Models

Model	**RMSE**	**MAE**	**MAPE**
SARIMAX$(2,1,2)(1,2,0)_{12}$ w/ Precipitation	1872.2	1566.9	15.48%
ARIMAX$(2,1,2)$ w/ Precipitation	3292.4	3107.8	31.07%
ARIMAX$(2,1,2)$ w/ Avg Temp	3388.0	3205.4	32.08%
ARIMAX$(2,2,2)$ w/ Precipitation	3225.6	3050.8	30.52%
ARIMAX$(2,2,2)$ w/ Avg Temp	3314.6	3142.9	31.48%
ARIMAX$(7,2,2)$ w/ Precipitation	2632.9	2368.1	23.30%
ARIMAX$(7,2,2)$ w/ Avg Temp	2668.12	2397.508	23.59%
SARIMAX$(1,1,1)(1,2,0)_{12}$ w/ All Features	1814.4	1492.3	14.68%
SARIMAX$(1,1,1)(1,2,0)_{12}$ w/ All Features + Log	2360.4	1840.3	19.09%

absolute percentage errors. Their formulas are listed below:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|$$

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \times 100\%$$

The three metrics all contain the term $|y - \hat{y}|$, which is the absolute value between the predicted price and actual price. As all of the three metrics measure the overall prediction error, we want them to be as small as possible, indicating the model is giving a relatively accurate prediction.
According to Table 1 above, we can find that the model with the lowest RMSE, MAE, and MAPE is SARIMAX$(1,1,1)(1,2,0)_1$ w/ All Features. Therefore, purely from the point of view of the three metrics chosen, the best model should be SARIMAX$(1,1,1)(1,2,0)_{12}$ w/ All Features.

5.3 Forecast Results and Pattern Analysis

For the SARIMAX$(1,1,1)(1,2,0)_{12}$ with All Features, we have forecasted values that exhibit an upward trend. The shaded grey area represents the confidence intervals, indicating uncertainty around the predictions. The forecast suggests that prices will continue rising, aligning with the recent upward movement but with a smoother rate. Before 2022, prices remained relatively stable pattern with minor fluctuations. However, between 2023–2024, there was a sharp rise in prices, peaking around April 2024, followed by a slight decline. Compared to the historical trend, the forecast prices appear more conservative and may underestimate the price changes.

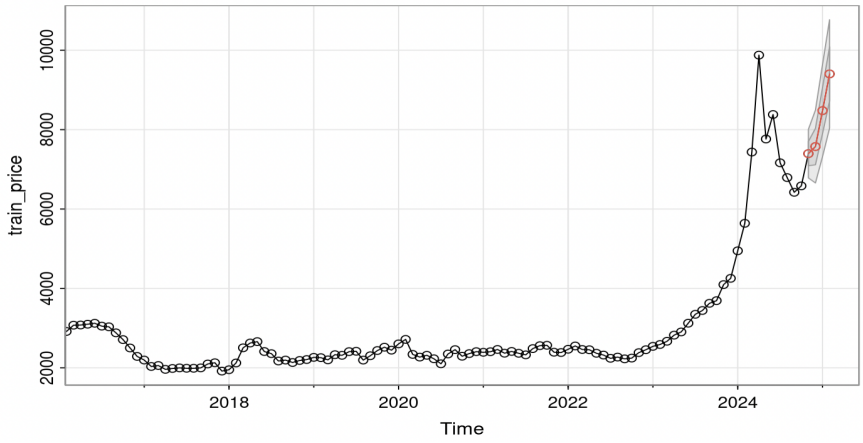


Figure 3: SARIMAX$(1,1,1)(1,2,0)_{12}$ 4-month Forecast

5.4 Forecast Accuracy: Predicted vs. Actual

The model we chose is the SARIMAX$(1,1,1)(1,2,0)_{12}$ 4-month Forecast with GARCH$(1,1)$ applied to the residuals. The forecasted prices exhibit an increasing trend from November 2024 to February 2025, while actual price data follow a different pattern. It follows an increasing trend and spikes in January 2025, then drops in February 2025. For the confidence intervals, we have the 95% CI (gray) and 90% CI (blue), both becoming slightly narrower over time as the GARCH$(1,1)$ is applied to the residuals. The GARCH$(1,1)$ captures time-varying volatility, adjusting the confidence intervals based on the conditional variance of past errors. Actual prices from December 2024 and January 2025 lie outside the upper confidence interval, suggesting that the model underestimates the main trend of the true price. By February 2025, the Confidence interval of the model captures the true price. Overall, the model provides a reasonable long-term trend forecast in price, but it does not fully capture short-term market change. Even though the general trend is accurate, it highlights the need for alternative or enhanced modelling approaches.

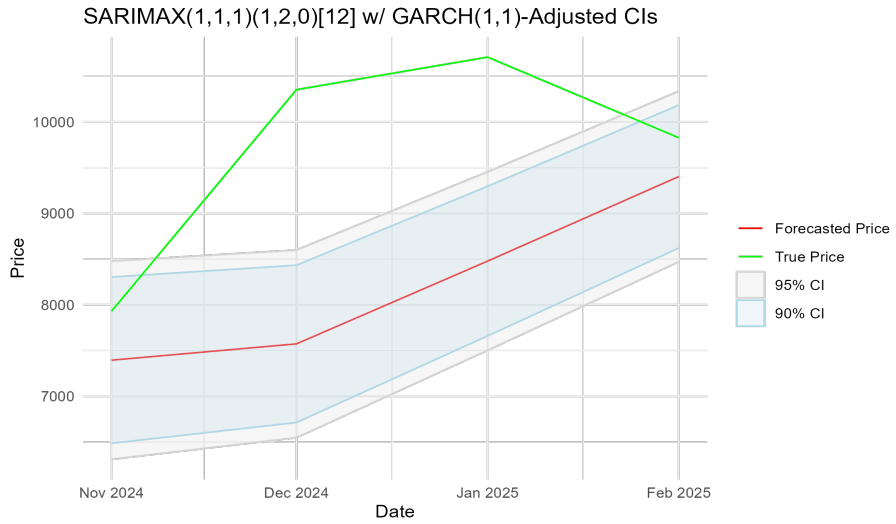


Figure 4: SARIMAX$(1,1,1)(1,2,0)_{12}$ 4-month Forecast and Observed Prices with GARCH(1,1)-adjusted Confidence Intervals

6 Discussion and Conclusion

Although a variety of models were considered, the final model chosen did not adequately forecast the four-month period, as shown in Figure 4. This was largely due to the sporadic nature of the test data - around 2024 leading into 2025, there is a large spike and an increase in volatility, which makes cocoa prices difficult to forecast, especially for traditional time series models.[5]. For these reasons, ML methods would likely be worth exploring in the future. Forecasts were restricted to 4 months to strike a balance between having a reasonable forecast period and having to reduce the training data to lag the exogenous regressors, which limits the utility of the model. Additionally, significant portions of the climate data used as exogenous regressors were imputed using historical seasonal averages, which were likely to be inaccurate, introducing bias into the model. Model parameters were largely informed by patterns in diagnostic plots as well as trial-and-error, which inherently leaves out many potentially well-performing models. Using a more robust, holistic method of testing models, such as a grid search to explore the full sample space of parameters, would likely have found a better-performing model.

7 Appendix: Code

Appendix: Code

```r
library(tidyverse)   # for data manipulation and plotting
library(lubridate)   # for date formatting
library(zoo)         # for time series interpolation
library(forecast)    # for ARIMA, ETS, and other time series models
library(tseries)     # for stationarity tests
library(astsa)       # for sarima
library(Hmisc)       # for lag features
library(rugarch)     # for GARCH
library(ggplot2)     # for plots

# Load datasets
icco_raw <- read.csv("Daily Prices_ICCO.csv", stringsAsFactors = FALSE)
ghana <- read_csv("Ghana_data.csv")
```

```r
# --- Clean ICCO data (keep NAs) ---
icco_clean <- read_csv("Daily Prices_ICCO.csv") %>%
  rename(Date = 1, `Daily Price` = 2) %>%
  mutate(Date = dmy(Date)) %>%
  mutate(`Monthly Average Price` = as.numeric(gsub(",", "", `Daily Price`))) %>%
  arrange(Date)

# --- Clean Ghana data (keep NAs) ---
ghana <- read_csv("Ghana_data.csv") %>%
  mutate(DATE = ymd(DATE)) %>%
  arrange(DATE) %>%
  mutate(across(c(PRCP, TAVG, TMAX, TMIN), as.numeric)) %>%
  mutate(PRCP = ifelse(is.na(PRCP), 0, PRCP))  # Based on documentation

# Aggregate by date (avg across stations)
ghana_daily <- ghana %>%
  group_by(DATE) %>%
  summarise(across(c(PRCP, TAVG, TMAX, TMIN), mean, na.rm = TRUE)) %>%
  ungroup()
# Merge ICCO and Ghana (full merged data with NAs preserved)
merged_data <- icco_clean %>%
  left_join(ghana_daily, by = c("Date" = "DATE"))

pre_forecast_data <- merged_data %>%
  filter(Date >= as.Date("1994-10-03") & Date <= as.Date("2024-11-28")) %>%
  drop_na() %>%  # remove rows with any NA
  mutate(month = floor_date(Date, unit = "month")) %>%
  group_by(month) %>%
  summarise(across(where(is.numeric), mean)) %>%
  ungroup()
```

```r
# Slice only the forecasting range
post_forecast_data <- merged_data %>%
  filter(Date >= as.Date("2024-11-29") & Date <= as.Date("2025-02-27"))

# Interpolate missing values for this range only
post_forecast_filled <- post_forecast_data %>%
  mutate(across(c(`Daily Price`, PRCP, TAVG, TMAX, TMIN), ~ na.approx(., na.rm = FALSE))) %>%
  mutate(month = floor_date(Date, unit = "month")) %>%
  group_by(month) %>%
  summarise(across(where(is.numeric), mean)) %>%
  ungroup()

monthly_data <- bind_rows(pre_forecast_data, post_forecast_filled)
```

```r
ghana_seasonal <- merged_data %>%
  filter(Date <= as.Date("2024-11-28")) %>%  # Only use historical data
  mutate(doy = yday(Date)) %>%
  group_by(doy) %>%
  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)
  )
# Create future date range
future_dates <- tibble(
  Date = seq.Date(from = as.Date("2024-11-29"), to = as.Date("2025-02-27"), by = "day")
) %>%
  mutate(doy = yday(Date))

# Join with seasonal averages
future_filled <- future_dates %>%
  left_join(ghana_seasonal, by = "doy") %>%
  left_join(icco_clean, by = "Date")  # bring in Daily Price

# Monthly average of forecast period (now filled!)
post_forecast_filled <- future_filled %>%
  mutate(month = floor_date(Date, "month")) %>%
  group_by(month) %>%
  summarise(across(where(is.numeric), mean, na.rm = TRUE)) %>%
  ungroup()
pre_forecast_trimmed <- pre_forecast_data %>%
  filter(month < as.Date("2024-11-01"))  # exclude Nov 2024 to avoid duplication
monthly_data <- bind_rows(pre_forecast_trimmed, post_forecast_filled)
monthly_data <- monthly_data %>%
  select(-doy)
```

```r
# Step 1: Aggregate daily merged data by month (without filtering NAs)
merged_monthly <- merged_data %>%
  mutate(month = floor_date(Date, "month")) %>%
  group_by(month) %>%
  summarise(across(where(is.numeric), ~ mean(.x, na.rm = TRUE))) %>%
  ungroup()
```

```r
# Step 2: Create full sequence of months from first to last available date
all_months <- tibble(
  month = seq(from = floor_date(min(merged_data$Date), "month"),
              to = floor_date(max(merged_data$Date), "month"),
              by = "1 month")
)

# Step 3: Left join to preserve all months (even if missing from aggregated data)
monthly_complete <- all_months %>%
  left_join(merged_monthly, by = "month")

# Step 4: Interpolate NA values (linear)
monthly_imputed <- monthly_complete %>%
  mutate(across(where(is.numeric), ~ na.approx(.x, na.rm = FALSE)))
```

```r
# Implement 4 month lag on Exogenous Features to avoid data leakage
lag <- 4

monthly_data <- monthly_imputed |>
  mutate(PRCP_lag = Lag(PRCP,lag)) |>
  mutate(TAVG_lag = Lag(TAVG,lag)) |>
  mutate(TMAX_lag = Lag(TMAX,lag)) |>
  mutate(TMIN_lag = Lag(TMIN,lag)) |>
  select(-c(PRCP, TAVG, TMIN, TMAX)) |>
  na.exclude()

monthly_ts <- ts(monthly_data$`Monthly Average Price`,
                 start = c(1995, 2), # starting year and month
                 frequency = 12)
```

```r
# Split data into training set and test set

train_size <- length(monthly_ts) - 4

train_price <- head(monthly_ts, train_size)
test_price <- tail(monthly_ts, length(monthly_ts) - length(train_price))

train_xreg <- monthly_data[1:train_size,4:7]
test_xreg <- monthly_data[(train_size+1):nrow(monthly_data),4:7]

autoplot(train_price) + autolayer(test_price)
```

EDA

```r
train_dates <- as.Date.yearmon(time(train_price))
test_dates <- as.Date.yearmon(time(test_price))

ggplot() +
  geom_line(aes(x = train_dates, y = train_price, color = "Training Data")) +
  geom_line(aes(x = test_dates, y = test_price, color = "Test Data")) +
```

```
  labs(title = "Monthly Cocoa Futures Prices", x = "Date", y = "Price") +
  scale_colour_manual("",
                      breaks = c("Training Data", "Test Data"),
                      values = c("black", "red")) +
  theme_minimal()
```

```
# Data visualization
plot(monthly_data$month, monthly_data$`Monthly Average Price`,main="Monthly Cocoa Price", xlab="Time", 

plot(monthly_data$month, monthly_data$PRCP_lag, type="l", col="blue", lwd=2, ylim=c(0, 120),
     xlab="Date", ylab="Values", main="Other Impact Variables Over Time")
lines(monthly_data$month, monthly_data$TAVG_lag, col="red", lwd=2)
lines(monthly_data$month, monthly_data$TMAX_lag, col="green", lwd=2)
lines(monthly_data$month, monthly_data$TMIN_lag, col="purple", lwd=2)
legend("bottomright", legend=c("Daily precipitation", "Daily average temperature", "Maximum daily tempe

ggplot(monthly_data, aes(x = `Monthly Average Price`)) +
  geom_histogram(fill = "blue", bins = 30, alpha = 0.7) +
  labs(x = "Cocoa Price", y = "Frequency") +
  ggtitle("Histogram of Monthly Average Price") +
  theme(plot.title = element_text(hjust = 0.5))
  theme_minimal()
```

Model

```
# ETS Model Without Log-Transformation
ets_model <- ets(train_price, model = "AAA")

# Print model summary
summary(ets_model)

# Draw ETS Model
plot(ets_model)
```

```
# ETS Model with Log-Transformation
ets_model_log <- ets(log(train_price), model = "AAN")

# Print model summary
summary(ets_model_log)

# Draw ETS Model
plot(ets_model_log)
```

AIC and BIC for log-transformation model is much smaller. Compare with not transformation, the log transformation's AIC and BIC value around 30 times smaller. The value of MAPE is smaller for log transformed model which is only 0.6257 and for original one is 5.37 74. Thus, for ETS model, use log-transformation is better.

```
# Residual Check
Box.test(residuals(ets_model), type="Ljung-Box")
```

```
acf2(resid(ets_model))
qqnorm(residuals(ets_model))
qqline(residuals(ets_model), col="red")
```

```
Box.test(residuals(ets_model_log), type="Ljung-Box")
acf2(resid(ets_model_log))
qqnorm(residuals(ets_model_log))
qqline(residuals(ets_model_log), col="red")
```

For both model have p value greater than 0.05. Residual normality graph show both models not fit the straight line and at beginning and end of line not fit the straight line. However, the log transformation model have more data doesn't fit. This refers both model doesn't follow the residual assumption.

```
# ARIMA Model
# Check Stationary
autoplot(train_price)
acf2(train_price)
```

It is clear to show that the ACF doesn't cutoff and decay slowly which need difference to make data stationary.

```
# Check by Difference
one_difference <- diff(train_price)
autoplot(one_difference)
acf2(one_difference)
```

The ACF/PACF perform better and cutoff at lag 2 and 2. For first difference, p=2, q=2. It can form a ARIMA(2,1,2) model.

```
# Check Second Difference
two_difference <- diff(train_price,2)
autoplot(two_difference)
acf2(two_difference)
```

By figures, it shows for second difference the lag is worse as the lag cutoff at 2 and 7 for ACF and PACF. This make a ARIMA(7,2,2). The first order difference is enough for the stationary so d = 1. ACF and PACF are both 2. This means the q=2 and p=2. Now, by the difference, the ARIMA model is clear to show which is ARIMA(2,1,2)

```
# Check stationary
adf.test(train_price)
```

By checking the p value, it shows greater than 0.05 which is non-stationary. This need to difference the data to make it stationary.

```
# Check stationary by difference data
adf.test(one_difference)
```

By difference data, now it become stationary as p value is smaller than 0.05.

```r
# ARIMAX(2,1,2) with precipitation
arimax_model2 <- sarima(train_price, 2,1,2, xreg = train_xreg$PRCP_lag)
```

```r
# ARIMAX(2,1,2) with Average Temperature
arimax_model3 <- sarima(train_price,2,1,2, xreg = train_xreg$TAVG_lag)
```

```r
# ARIMAX(2,2,2) with precipitation
arimax_model4 <- sarima(train_price, 2,2,2, xreg = train_xreg$PRCP_lag)
```

```r
# ARIMAX(2,2,2) with average temperature
arimax_model5 <- sarima(train_price, 2,2,2, xreg = train_xreg$TAVG_lag)
```

```r
# ARIMAX(7,2,2) with precipitation
arimax_model6 <- sarima(train_price, 7,2,2, xreg = train_xreg$PRCP_lag)
```

```r
# ARIMAX(7,2,2) with average temperature
arimax_model7 <- sarima(train_price, 7,2,2, xreg = train_xreg$TAVG_lag)
```

By compare AIC and BIC, ARIMA(2,1,2) with difference data for precipitation have the smallest number. From this aspect, this is the best model. For all models, the residual normality have most of points is lay on the line but for the beginning and ending part. They same the similar performance. All the graph have outliers at beginning. ARIMA(7,2,2) for second difference with precipitation have the smallest

$$\sigma^2$$

which is 67074.81.

```r
# SARIMA Model
adf_result <- adf.test(train_price, k = 12)
print(adf_result)

# Extract the p-value
p_value <- adf_result$p.value
print(p_value)
```

```r
adf.test(diff(train_price,lag=12))
adf.test(diff(train_price,lag=12, differences=2))
```

```r
# Check Seasonality
nsdiffs(train_price) # return 0 means no seasonal pattern and not need difference, so D=0
```

By above, it clear to show that D = 0.

```r
acf2(diff(train_price, differences=2))    # Plot ACF
```

As p value smaller than 0.05 by second difference the seasonal difference become D=2. By above time series graph, it shows there is not a strong repeat. The ACF and PACF cutoff both happen at lag = 2. Thus, p=q=2.

By the graph, it clear to show at lag = 12, ACF is positive Thus, P = 1 and Q = 0. Therefore, the model is SARIMA(2,1,2)x(1,2,0,12)

```
sarimax_model1 <- sarima(train_price,2,1,2,1,2,0,12,xreg=train_xreg$PRCP_lag)
```

Try p = 1 and q = 1 for the SARIMA model.

```
sarimax_model2 <- sarima(train_price,1,1,1,1,2,0,12,xreg=train_xreg$PRCP_lag)
```

AIC and BIC for SARIMA model is bigger than ARIMA model, This is possible to happen. Notice that the p values for the Ljung-Box statistic are all smaller than 0.05.

##Test Set

```
#Use the test set to do this, first plot the actual data

ggplot(tail(monthly_data,4),aes(x = month, y = `Monthly Average Price`)) +
  geom_line(color = "blue") +
  geom_point(color = "red") +  # Points on the line
  labs(title = "Monthly Price",
       x = "Month",
       y = "Price") +
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.title = element_text(size = 12),
        axis.text.x = element_text(angle = 45, hjust = 1),
        legend.text = element_text(size = 12))
```

We use the last 4 observations for the test data.

###Arima and SARIMA

Assume that all ARIMA and SARIMA models above satisfied the residual assumptions.

```
sarima.for(train_price,
           n.ahead = 4,
           p = 2, d = 1, q = 2,
           P = 1, D = 2, Q = 0, S = 12,  # add the PDQ
           xreg =  train_xreg$PRCP_lag,        # historical xreg, Use trained data.
           newxreg = test_xreg$PRCP_lag)
```

```
sarima.for(train_price,
           n.ahead = 4,
           p = 1, d = 1, q = 1,
           P = 1, D = 2, Q = 0, S = 12,  # add the PDQ
           xreg =  train_xreg$PRCP_lag,        # historical xreg, Use trained data.
           newxreg = test_xreg$PRCP_lag)
```

Then check the ARIMA models

```
#For model ARIMA(2,1,2)
sarima.for(train_price,
           n.ahead = 4,
           p = 2, d = 1, q = 2,
           xreg =  train_xreg$PRCP_lag,        # historical xreg
           newxreg = test_xreg$PRCP_lag)
```

```
#For ARIMA(2,1,2)
sarima.for(train_price,
           n.ahead = 4,
           p = 2, d = 1, q = 2,
           xreg =  train_xreg$TAVG_lag,        # historical xreg
           newxreg = test_xreg$TAVG_lag)
```

As we are checking the model ARIMA(2,1,2) with difference, it is equivalent to the ARIMA(2,2,2) without difference.

```
#For model ARIMA(2,2,2)
sarima.for(train_price,
           n.ahead = 4,
           p = 2, d = 2, q = 2,
           xreg =  train_xreg$PRCP_lag,        # historical xreg
           newxreg = test_xreg$PRCP_lag)
```

```
#For model ARIMA(2,2,2)
sarima.for(train_price,
           n.ahead = 4,
           p = 2, d = 2, q = 2,
           xreg =  train_xreg$TAVG_lag,        # historical xreg
           newxreg = test_xreg$TAVG_lag)
```

```
#For model ARIMA(7,2,2)
sarima.for(train_price,
           n.ahead = 4,
           p = 7, d = 2, q = 2,
           xreg =  train_xreg$PRCP_lag,        # historical xreg
           newxreg = test_xreg$PRCP_lag)
```

```
#For model ARIMA(7,2,2)
sarima.for(train_price,
           n.ahead = 4,
           p = 7, d = 2, q = 2,
           xreg =  train_xreg$TAVG_lag,        # historical xreg
           newxreg = test_xreg$TAVG_lag)
```

For the plots above, we can see that the model with smaller AIC would have relatively smooth forecast, and the model with larger AIC would have relatively fluctuating forecast. To compare the performance of the models, we can use three different evaluation metrics to check the accuracy of SARIMA and ARIMA forecasting models.

Check and compare RMSE, MAE, and MAPE for the models:

1. SARIMAX(2,1,2)(1,2,0)[12] w/ Precipitation

```
forecast_result_1 <- sarima.for(train_price,
           n.ahead = 4,
           p = 2, d = 1, q = 2,
           P = 1, D = 2, Q = 0, S = 12,
```

```
            xreg =  train_xreg$PRCP_lag,
            newxreg = test_xreg$PRCP_lag)
predicted_values_1 <- forecast_result_1$pred

#MAE
# Calculate errors
errors_1 <- test_price - predicted_values_1

# RMSE (Root Mean Squared Error)
rmse_1 <- sqrt(mean(errors_1^2))

# MAE (Mean Absolute Error)
mae_1 <- mean(abs(errors_1))

# MAPE (Mean Absolute Percentage Error)
mape_1 <- mean(abs(errors_1 / test_price)) * 100  # in percentage

# Print results
cat("RMSE:", rmse_1, "\n")
cat("MAE:", mae_1, "\n")
cat("MAPE:", mape_1, "%\n")
```

2. ARIMAX(2,1,2) w/ Precipitation

```
forecast_result_2 <- sarima.for(train_price,
            n.ahead = 4,
            p = 2, d = 1, q = 2,
            xreg =  train_xreg$PRCP_lag,        # historical xreg
            newxreg = test_xreg$PRCP_lag)

predicted_values_2 <- forecast_result_2$pred

#MAE
# Calculate errors
errors_2 <- test_price - predicted_values_2

# RMSE (Root Mean Squared Error)
rmse_2 <- sqrt(mean(errors_2^2))

# MAE (Mean Absolute Error)
mae_2 <- mean(abs(errors_2))

# MAPE (Mean Absolute Percentage Error)
mape_2 <- mean(abs(errors_2 / test_price)) * 100  # in percentage

# Print results
cat("RMSE:", rmse_2, "\n")
cat("MAE:", mae_2, "\n")
cat("MAPE:", mape_2, "%\n")
```

3. ARIMAX(2,1,2) w/ Avg Temperature

```r
forecast_result_3 <- sarima.for(train_price,
           n.ahead = 4,
           p = 2, d = 1, q = 2,
           xreg =  train_xreg$TAVG_lag,        # historical xreg
           newxreg = test_xreg$TAVG_lag)

predicted_values_3 <- forecast_result_3$pred

#MAE
# Calculate errors
errors_3 <- test_price - predicted_values_3

# RMSE (Root Mean Squared Error)
rmse_3 <- sqrt(mean(errors_3^2))

# MAE (Mean Absolute Error)
mae_3 <- mean(abs(errors_3))

# MAPE (Mean Absolute Percentage Error)
mape_3 <- mean(abs(errors_3 / test_price)) * 100  # in percentage

# Print results
cat("RMSE:", rmse_3, "\n")
cat("MAE:", mae_3, "\n")
cat("MAPE:", mape_3, "%\n")
```

4. ARIMAX(2,2,2) w/ Precipitation

```r
forecast_result_4 <- sarima.for(train_price,
           n.ahead = 4,
           p = 2, d = 2, q = 2,
           xreg =  train_xreg$PRCP_lag,        # historical xreg
           newxreg = test_xreg$PRCP_lag)

predicted_values_4 <- forecast_result_4$pred

#MAE
# Calculate errors
errors_4 <- test_price - predicted_values_4

# RMSE (Root Mean Squared Error)
rmse_4 <- sqrt(mean(errors_4^2))

# MAE (Mean Absolute Error)
mae_4 <- mean(abs(errors_4))

# MAPE (Mean Absolute Percentage Error)
mape_4 <- mean(abs(errors_4 / test_price)) * 100  # in percentage

# Print results
cat("RMSE:", rmse_4, "\n")
cat("MAE:", mae_4, "\n")
cat("MAPE:", mape_4, "%\n")
```

5. ARIMAX(2,2,2) w/ Avg Temperature

```
forecast_result_5 <- sarima.for(train_price,
           n.ahead = 4,
           p = 2, d = 2, q = 2,
           xreg =  train_xreg$TAVG_lag,        # historical xreg
           newxreg = test_xreg$TAVG_lag)

predicted_values_5 <- forecast_result_5$pred

#MAE
# Calculate errors
errors_5 <- test_price - predicted_values_5

# RMSE (Root Mean Squared Error)
rmse_5 <- sqrt(mean(errors_5^2))

# MAE (Mean Absolute Error)
mae_5 <- mean(abs(errors_5))

# MAPE (Mean Absolute Percentage Error)
mape_5 <- mean(abs(errors_5 / test_price)) * 100  # in percentage

# Print results
cat("RMSE:", rmse_5, "\n")
cat("MAE:", mae_5, "\n")
cat("MAPE:", mape_5, "%\n")
```

6. ARIMAX(7,2,2) w/ Precipitation

```
forecast_result_6 <- sarima.for(train_price,
           n.ahead = 4,
           p = 7, d = 2, q = 2,
           xreg =  train_xreg$PRCP_lag,        # historical xreg
           newxreg = test_xreg$PRCP_lag)

predicted_values_6 <- forecast_result_6$pred

#MAE
# Calculate errors
errors_6 <- test_price - predicted_values_6

# RMSE (Root Mean Squared Error)
rmse_6 <- sqrt(mean(errors_6^2))

# MAE (Mean Absolute Error)
mae_6 <- mean(abs(errors_6))

# MAPE (Mean Absolute Percentage Error)
mape_6 <- mean(abs(errors_6 / test_price)) * 100  # in percentage

# Print results
cat("RMSE:", rmse_6, "\n")
```

```r
cat("MAE:", mae_6, "\n")
cat("MAPE:", mape_6, "%\n")
```

7. ARIMAX(7,2,2) w/ Avg Temperature

```r
forecast_result_7 <- sarima.for(train_price,
           n.ahead = 4,
           p = 7, d = 2, q = 2,
           xreg =  train_xreg$TAVG_lag,        # historical xreg
          newxreg = test_xreg$TAVG_lag)

predicted_values_7 <- forecast_result_7$pred

#MAE
# Calculate errors
errors_7 <- test_price - predicted_values_7

# RMSE (Root Mean Squared Error)
rmse_7 <- sqrt(mean(errors_7^2))

# MAE (Mean Absolute Error)
mae_7 <- mean(abs(errors_7))

# MAPE (Mean Absolute Percentage Error)
mape_7 <- mean(abs(errors_7 / test_price)) * 100  # in percentage

# Print results
cat("RMSE:", rmse_7, "\n")
cat("MAE:", mae_7, "\n")
cat("MAPE:", mape_7, "%\n")
```

8. SARIMAX(1,1,1)(1,2,0)[12] w/ all features

```r
forecast_result_8 <-sarima.for(train_price,
           n.ahead = 4,
           p = 1, d = 1, q = 1,
           P = 1, D = 2, Q = 0, S = 12,  # add the PDQ
           xreg =  train_xreg,        # historical xreg, Use trained data.
           newxreg = test_xreg)

predicted_values_8 <- forecast_result_8$pred

#MAE
# Calculate errors
errors_8 <- test_price - predicted_values_8

# RMSE (Root Mean Squared Error)
rmse_8 <- sqrt(mean(errors_8^2))

# MAE (Mean Absolute Error)
mae_8 <- mean(abs(errors_8))
```

```r
# MAPE (Mean Absolute Percentage Error)
mape_8 <- mean(abs(errors_8 / test_price)) * 100  # in percentage

# Print results
cat("RMSE:", rmse_8, "\n")
cat("MAE:", mae_8, "\n")
cat("MAPE:", mape_8, "%\n")
```

RMSE measures the average magnitude of errors. MAE measures average absolute errors. MAPE measures percentage errors. So we want smaller values for RMSE, MAE, and MAPE because they indicate lower prediction errors and better accuracy. Comparing the values for these three performance indicators, we can find that SARIMA(1,1,1)x(1,2,0,12) with average perception is the best model for prediction.

###Forecast versus Actual test price Use SARIMAX(1,1,1)x(1,2,0)[12] with all features to visualize Forecast vs. Actual See that with the 90% and 95% confidence intervals, we have:

```r
lower_95 <- forecast_result_8$pred - 1.96 * forecast_result_8$se
upper_95 <- forecast_result_8$pred + 1.96 * forecast_result_8$se
lower_90 <- forecast_result_8$pred - 1.645 * forecast_result_8$se
upper_90 <- forecast_result_8$pred + 1.645 * forecast_result_8$se

date <- as.Date.yearmon(time(test_price))

ggplot() +
  geom_line(aes(x = date, y = forecast_result_8$pred, color = "Forecasted Price")) +
  geom_ribbon(aes(x = date, ymin = lower_95, ymax = upper_95, color = "95% CI"),
              fill = "lightgrey", alpha = 0.2) +
  geom_ribbon(aes(x = date, ymin = lower_90, ymax = upper_90, color = "90% CI"),
              fill = "lightblue", alpha = 0.2) +
  geom_line(aes(x = date, y = test_price, color = "True Price")) +
  labs(title = "SARIMAX(1,1,1)(1,2,0)[12] w/ all features",
       x = "Date", y = "Price",) +
  scale_colour_manual("",
                      breaks = c("Forecasted Price", "True Price", "95% CI",
                                 "90% CI"),
                      values = c("red", "green", "lightgrey", "lightblue")) +
  scale_x_date(date_labels = "%b %Y") +
  theme_minimal()
```

Even though we choose the best ARIMA model, the 90% CI still does not capture all actual values of the price. But the last value is within the 95% CI.

Apply Log to price

9. SARIMAX(1,1,1)(1,2,0)[12] w/ all features + log transformation

```r
forecast_result_9 <-sarima.for(log(train_price),
           n.ahead = 4,
           p = 1, d = 1, q = 1,
           P = 1, D = 2, Q = 0, S = 12,  # add the PDQ
           xreg =  train_xreg,        # historical xreg, Use trained data.
```

```
           newxreg = test_xreg)

predicted_values_9 <- exp(forecast_result_9$pred)*(1+forecast_result_9$se/2)

#MAE
# Calculate errors
errors_9 <- test_price - predicted_values_9

# RMSE (Root Mean Squared Error)
rmse_9 <- sqrt(mean(errors_9^2))

# MAE (Mean Absolute Error)
mae_9 <- mean(abs(errors_9))

# MAPE (Mean Absolute Percentage Error)
mape_9 <- mean(abs(errors_9 / test_price)) * 100  # in percentage

# Print results
cat("RMSE:", rmse_9, "\n")
cat("MAE:", mae_9, "\n")
cat("MAPE:", mape_9, "%\n")
```

```
lower_95 <- forecast_result_9$pred - 1.96 * forecast_result_9$se
upper_95 <- forecast_result_9$pred + 1.96 * forecast_result_9$se
lower_90 <- forecast_result_9$pred - 1.645 * forecast_result_9$se
upper_90 <- forecast_result_9$pred + 1.645 * forecast_result_9$se

date <- as.Date.yearmon(time(test_price))

ggplot() +
  geom_line(aes(x = date, y = forecast_result_9$pred, color = "Forecasted Log Price")) +
  geom_ribbon(aes(x = date, ymin = lower_95, ymax = upper_95, color = "95% CI"),
              fill = "lightgrey", alpha = 0.2) +
  geom_ribbon(aes(x = date, ymin = lower_90, ymax = upper_90, color = "90% CI"),
              fill = "lightblue", alpha = 0.2) +
  geom_line(aes(x = date, y = log(test_price), color = "True Log Price")) +
  labs(title = "SARIMAX(1,1,1)(1,2,0)[12] w/ all features + Log transformation",
       x = "Date", y = "Log Price") +
  scale_colour_manual("",
                      breaks = c("Forecasted Log Price", "True Log Price", "95% CI",
                                 "90% CI"),
                      values = c("red", "green", "lightgrey", "lightblue")) +
  scale_x_date(date_labels = "%b %Y") +
  theme_minimal()
```

- Check Residuals for SARIMAX(1,1,1)(1,2,0)[12]
- Fit GARCH(1,1) to Residuals due to heteroskedasticity

```
sarimax_mod9 <- sarima(train_price,1,1,1,1,2,0,12, xreg = train_xreg)
residuals <- resid(sarimax_mod9$fit)
```

```
garch_spec <- ugarchspec(variance.model = list(model = "apARCH", garchOrder = c(1,1)),
                         mean.model = list(armaOrder = c(0,0), include.mean = FALSE),
                         distribution.model = "norm")

garch_fit <- ugarchfit(garch_spec, residuals)
print(garch_fit)
```

- Standard GARCH (sGARCH) showed significant Negative Sign Bias
- Asymmetric Power ARCH (apARCH) relaxed the assumption of positive and negative shocks contributing equal effects to volatility
- apARCH: No significant sign biases + lower AIC, but increases volatility by a lot
- (p=1,q=1) showed lower AIC than (0,1) and (1,0) failed to converge
- Opt for sGARCH

```
garch_forecast <- ugarchforecast(garch_fit, n.ahead = 4)
garch_volatility <- sigma(garch_forecast)
plot(garch_volatility, type = "l", main = "Forecasted Volatility")
```

```
lower_95 <- forecast_result_8$pred - 1.96 * garch_volatility
upper_95 <- forecast_result_8$pred + 1.96 * garch_volatility
lower_90 <- forecast_result_8$pred - 1.645 * garch_volatility
upper_90 <- forecast_result_8$pred + 1.645 * garch_volatility

date <- as.Date.yearmon(time(test_price))

ggplot() +
  geom_line(aes(x = date, y = forecast_result_8$pred, color = "Forecasted Price")) +
  geom_ribbon(aes(x = date, ymin = lower_95, ymax = upper_95, color = "95% CI"),
              fill = "lightgrey", alpha = 0.2) +
  geom_ribbon(aes(x = date, ymin = lower_90, ymax = upper_90, color = "90% CI"),
              fill = "lightblue", alpha = 0.2) +
  geom_line(aes(x = date, y = test_price, color = "True Price")) +
  labs(title = "SARIMAX(1,1,1)(1,2,0)[12] w/ GARCH(1,1)-Adjusted CIs",
       x = "Date", y = "Price") +
  scale_colour_manual("",
                      breaks = c("Forecasted Price", "True Price", "95% CI",
                                 "90% CI"),
                      values = c("red", "green", "lightgrey", "lightblue")) +
  scale_x_date(date_labels = "%b %Y") +
  theme_minimal()
```

8 References

[1] Emmanuel Bangmarigu and Artan Qineti. Cocoa Production and Export in Ghana. 162nd Seminar, April 26-27, 2018, Budapest, Hungary 272051, European Association of Agricultural Economists, April 2018.

[2] Ketut Sukiyono, Musriyadi Nabiu, Bambang Sumantri, Ridha Novanda, Nyayu Arianti, Sriyoto, M. Yuliarso, Redy Badrudin, Muhamad Romdhon, and H. Mustamam. Selecting an accurate cacao price forecasting model. *Journal of Physics: Conference Series*, 1114:012116, 11 2018.

[3] Dongna Zhang, Xingyu Dai, Qunwei Wang, and Chi Keung Marco Lau. Impacts of weather conditions on the us commodity markets systemic interdependence across multi-timescales. *Energy Economics*, 123:106732, 7 2023.

[4] Assis Kamu, Amran Ahmed, and Remali Yusoff. Forecasting cocoa bean prices using univariate time series models. *Journal of Arts Science Commerce*, 1:71, 01 2010.

[5] Andreea-Cristina PETRICĂ, Stelian STANCU, and Alexandru TINDECHE. Limitation of ARIMA models in financial and monetary economics. *Theoretical and Applied Economics*, 0(4(609), W):19–42, Winter 2016.