M.SC. DATA ANALYTICS & TECHNOLOGIES

DAT7304 - BUSINESS ANALYTICS

PORTFOLIO 1 - ASSIGNMENT 1

P PART A – TIME SERIES ANALYSIS AND FORECASTING

PART B - LINEAR PROGRAMMING: OPTIMISING SUPPLIER SELECTION FOR A MULTI-WAREHOUSE SUPPLY CHAIN

BY: OKIKE U. J.

Student ID: 2423983

INSTRUCTOR: ANCHAL GARG

DATE. 25-04-2025

LIST OF FIGURES

PART A

- Figure 1. Comparison of Forecast Accuracy (MAPE) Across Various Models
- Figure 2. Overall Modelling Workflow
- Figure 3. Screenshot of step 1 code
- Figure 4. Screenshot of step 2 code and outcome
- Figure 5. Screenshot of step 3 code and outcome
- Figure 6. Screenshot of step 4, boxplot outliers
- Figure 7. Screenshot of step 4, IQR outliers
- Figure 8. Screenshot of step 4, Outlier handling
- Figure 9. Screenshot of step 1, ts_fuel creation
- Figure 10. Screenshot of step 2, STL Decomposition Plot
- Figure 11. Screenshot of step 3, ADF test & result
- Figure 12. Screenshot of step 4, KPSS result
- Figure 13. Screenshot of step 5, Differenced Fuel Price Plot
- Figure 14. Screenshot of step 6, ADF on differenced series
- Figure 15. Screenshot of step 7, ACF and PACF Plots
- Figure 16. Screenshot of step 8, Smoothed Log-Transformed Fuel Price Plot
- Figure 17. Screenshot of step 1&2, Train vs Test Split
- Figure 18. Screenshot of step 3, Manual ARIMA on train_ts
- Figure 19. Screenshot of step 4, Auto ARIMA on train_ts
- Figure 20. Screenshot of step 5, Auto SARIMA vs Actual | Manual SARIMA vs Actual
- Figure 21. Screenshot of step 6, Forecast Accuracy Comparison for Manual & Auto ARIMA
- Figure 22. Screenshot of step 1, showing raw dataset summary

- Figure 23. Screenshot of step 2, showing cleaned dataset summary
- Figure 24. Screenshot of step 3, STL decomposition plot
- Figure 25. Screenshot of step 4, Stationarity tests
- Figure 26. Screenshot of step 5, ACF and PACF plots
- Figure 27. Screenshot of step 6, Granger Causality test output
- Figure 28. Screenshot of step 9, Forecast vs Actual plot (manual vs auto SARIMAX)
- Figure 29. Screenshot of step 9, Forecast accuracy metrics comparison
- Figure 30. Screenshot of step 1, Brent Crude Loading and Cleaning Code
- Figure 31. Screenshot of step 2, Weekly Brent Crude Price Line Chart
- Figure 32. Screenshot Of Step 4, Differenced Fuel Prices Series With External Model
- Figure 33. Screenshot Of Step 5, Granger Causality Test Output
- Figure 34. Screenshot Of Step 6, Train/Test Split & Model Fitting
- Figure 35. Screenshot Of Step 7, Overlay Plot of Manual vs Auto SARIMAX vs Actual
- Figure 36. Screenshot Of Step 8, Forecast Accuracy
- Figure 37. Screenshot Of Step 1, Code for Forecast Accuracy for ARIMA-Based Models
- Figure 38. Screenshot Of Step 2, MAPE Comparison Across Forecasting Models
- Figure 39. Screenshot Of Step 1, Retrain and Refit Best Model on Full Dataset
- Figure 40. Screenshot Of Step 2, Code For 15 Weeks Forecast
- Figure 41. Screenshot Of Step 4, 15-Week Ahead Forecast Prices– SARIMAX (Brent)
- Figure 42. Screenshot Of Step 1, Lag Feature Creation and Train-Test Split
- Figure 43. Screenshot Of Step 2, Random Forest Forecast Accuracy for Varying ntree
- Figure 44. Screenshot Of Step 3, RF (ntree = 700) Actual vs Predicted Weekly Fuel Prices
- Figure 45. Screenshot Of Step 1, Preparing The Data For Forecasting With LSTM
- Figure 46. Screenshot Of Step 2, Fitting the LSTM Model

- Figure 47. Screenshot Of Step 3, Forecast with the LSTM Model
- Figure 48. Screenshot Of Step 4, Accuracy metrics for the LSTM model
- Figure 49. Screenshot Of Step 5, LSTM Forecast vs Actual Plot
- Figure 50. Screenshot Of Step 1, Lag Feature Creation and Train-Test Split (SVR)
- Figure 51. Screenshot Of Step 2, SVR Forecast Accuracy for Different Kernels
- Figure 52. Screenshot Of Step 3, SVR Linear Kernel Forecast
- Figure 53. Screenshot Of Step 4, SVR Forecast Accuracy
- Figure 54. Screenshot Of Step 5, SVR Forecast vs Actual Fuel Prices
- Figure 55. Screenshot Of Step 1, Aligning SVR And LSTM Model Predictions
- Figure 56. Screenshot Of Step 2, Hybrid Model (Hybrid Prediction = 0.8 * SVR + 0.2 * LSTM)
- Figure 57. Screenshot Of Step 3, Hybrid Model Accuracy
- Figure 58. Screenshot Of Step 4, Hybrid Forecast vs Actual (SVR + LSTM)
- Figure 59. MAPE Comparison Across Forecasting Models
- Figure 60. Final 15-Week Ahead Forecast using SVR Linear
- Figure 61. Final 15-Week Ahead Forecast Plot

PART B

- Figure 1. Cleaned Dataset Preview
- Figure 2. LP model logic and matrix formulation
- Figure 3. Optimised supplier-route combinations
- Figure 4. Aggregate Output
- Figure 5. Bar Chart of Adjusted Costs by Supplier

LIST OF TABLES

PART A

- Table 1. Forecasting Models' Details
- Table 2. Forecast Accuracy Comparison for Manual & Auto ARIMA
- Table 3. Forecast Accuracy Metrics Table Output (SARIMAX Brent vs Internal)
- Table 4. Comparative Forecast Accuracy for ARIMA-Based Models
- Table 5. Final Accuracy Comparison Table for All Models

PART B

- Table 1. Constraints and Descriptions

TABLE OF CONTENTS

LIST OF FIGURES ..... **ii**

LIST OF TABLES ..... **v**

PART A – TIME SERIES ANALYSIS AND FORECASTING ***1***

1. INTRODUCTION **1**

1.1 Background of the Work 1

1.2 Business Problem 1

1.3 Significance of the Problem 1

1.4 Business Questions 2

2. LITERATURE REVIEW **3**

2.1 Introduction 3

2.2 Classical Time Series Models 3

2.3 Machine Learning and Deep Learning in Time Series Forecasting 3

2.4 The Role of Regressors: Internal and External 4

2.5 Hybrid and Ensemble Approaches 4

2.6 Limitations of Past Research 5

3. METHODOLOGY **7**

3.1 Introduction 7

3.2 Dataset Description 7

3.3 Methodological Flowchart 7

3.4 Step-By-Step Methodology 9

3.5 Model Evaluation 10

3.6 Forecasting Strategy 10

3.7 Justification of Methods 10

3.8 Limitations of Methodology 11

4. IMPLEMENTATION AND RESULTS **12**

4.1 Introduction to Implementation 12

4.2 Data Preparation and Initial Exploration 12

4.3 Model Development and Evaluation 25

4.4 Multivariate Modelling With Internal Variable 30

4.5 Multivariate Modelling with External Variable (Brent Crude) 39

4.6 Machine Learning Models 53

4.7 Hybrid Model (SVR + LSTM) 62

4.8 Final Comparison and Model Selection 65

5. BUSINESS DISCUSSION AND INSIGHTS 71

5.1 Business Question 1 71

5.2 Business Question 2 71

5.3 Business Question 3 72

5.4 Business Question 4 72

5.5 Business Question 5 73

6. CONCLUSION AND RECOMMENDATIONS 74

6.1 Summary of Findings 74

6.2 Business Implications and Stakeholder Benefits 74

6.3 Strategic Recommendations 74

6.4 Limitations and Areas for Future Work 75

PART B – LINEAR PROGRAMMING REPORT: OPTIMISING INVENTORY ALLOCATION UNDER SUPPLIER RISK 76

1. INTRODUCTION 76

2. BUSINESS PROBLEM FORMULATION 77

3. DATA PREPROCESSING 78

4. LP MODEL IMPLEMENTATION IN R 79

5. RESULTS AND ANALYSIS 80

6. INTERPRETATION OF RESULTS 82

7. RECOMMENDATIONS **83**

8. LIMITATIONS AND FUTURE ENHANCEMENTS **84**

9. CONCLUSION **85**

REFERENCES **86**

PART A – TIME SERIES ANALYSIS AND FORECASTING

1. INTRODUCTION

1.1 Background of the Work

Fuel prices represent a critical indicator of economic activity, affecting logistics, transportation costs, consumer spending and policy decisions. In the United Kingdom, where fuel tax and global oil dynamics significantly influence prices, accurate forecasting is essential for both public and private sector stakeholders.

This study investigates weekly UK automotive fuel prices from January 2021 to early 2025, incorporating complementary features such as Brent crude oil prices and fuel quantity per transaction. By modelling fuel price fluctuations, the analysis provides insights to help retailers, logistics planners and policymakers respond to future pricing trends with confidence.

Time series forecasting has evolved from classical methods like ARIMA to more advanced approaches, including machine learning (ML) and deep learning (DL). With growing data availability, hybrid techniques that combine statistical rigour with predictive flexibility have also gained popularity. This study compares traditional, ML and hybrid models to identify the most accurate and reliable forecasting approach.

1.2 Business Problem

Fuel price volatility complicates decision-making for logistics firms, energy suppliers and transport operators. Without reliable forecasts, these stakeholders face challenges in budgeting, inventory planning, and consumer pricing.

Given the strategic importance of fuel pricing in the UK's economic ecosystem, developing a robust forecasting framework offers not only competitive advantage but also greater resilience in supply chain and fiscal planning.

1.3 Significance of the Problem

Accurate forecasting supports operational agility in the face of geopolitical shocks, inflation and energy market fluctuations. For businesses, it informs procurement and pricing strategies. For governments, it contributes to tax revenue projections and subsidy planning. This project aims to bridge the gap between statistical accuracy and practical usability by leveraging a variety of forecasting models.

1.4 Business Questions

The project is shaped by five key business questions:

1. Which time series forecasting model is the most accurate for weekly UK automotive fuel prices?
2. Do internal behavioural signals (like quantity of fuel purchased per transaction) or external market indicators (like Brent crude prices) have more predictive power?
3. Can machine learning and/or deep learning models outperform traditional statistical models?
4. Do hybrid models perform better than standalone ones for fuel price forecasting?
5. What practical insights can businesses take from the most dependable 15 weeks prediction, using the best model?

This business questions, serve as a thought process, guiding the development and implementation of this project.

2. LITERATURE REVIEW

2.1 Introduction

Fuel price forecasting is crucial due to its impact on transport costs, economic policy, inflation, and consumer behaviour (Wang & Krupnick, 2013). Reliable forecasting requires both internal drivers (e.g., quantity purchased per transaction) and external indicators (e.g., Brent crude). This chapter reviews classical and modern forecasting methods, the influence of regressors and the performance of hybrid models, placing this project within the broader predictive analytics space.

2.2 Classical Time Series Models

Traditional models like ARIMA and SARIMA have long been staples in time series forecasting. Their strength lies in modelling trend, seasonality and autocorrelation in a straightforward way (Box et al., 2016). However, they fall short when data becomes nonlinear or when external influences are involved. ARIMAX or SARIMAX address this by incorporating exogenous variables, which can improve performance when well chosen (Clements & Hendry, 2008).

In fuel price studies, these models have seen extensive use. For example, Li et al. (2019) found that SARIMAX models using Brent crude as a regressor improved petrol price forecasts across Europe. Yet, their linear nature often limits effectiveness in volatile markets, pushing researchers toward more flexible machine learning (ML) and deep learning (DL) approaches.

2.3 Machine Learning and Deep Learning in Time Series Forecasting

ML and DL offer more adaptability to nonlinearity and complexity. Random Forests (RF) are popular due to their ensemble structure and robustness, but they don't naturally model sequential data (Lemke et al., 2009). Support Vector Regression (SVR), especially with a linear kernel, is effective for smooth or high-dimensional patterns. Zhao and Zhang (2020) showed SVR outperforming ARIMA in volatile fuel markets.

LSTM networks, designed to handle long-term dependencies have proven valuable in energy forecasting where lagged effects are common (Fahimnia et al., 2015). These models have reshaped time series analysis by learning nonlinear patterns that traditional models may miss.

2.4 The Role of Regressors: Internal and External

Fuel prices are influenced by both internal behavioural data and external global trends. Sun et al. (2020) demonstrated that including internal signals like fuel purchase per transaction can improve accuracy. However, global indicators such as Brent crude remain dominant predictors, supported by numerous studies (IEA, 2021). Tools like Granger causality help validate their predictive value (Granger, 1969).

Interestingly, this study found Brent crude to be more informative than internal behavioural variables, contradicting Zhang et al. (2018), who claimed local patterns offered stronger short-term signals. Feature selection remains a critical step in model success.

2.5 Hybrid and Ensemble Approaches

Hybrid models aim to capture both linear and nonlinear features by combining models. Zhang (2003) proposed merging ARIMA with neural networks, while more recent work has paired LSTM with RF or SVR to improve accuracy (Ahmed et al., 2021). These approaches aim to combine model strengths.

This project experimented with a weighted hybrid model (SVR 80%, LSTM 20%), but the result underperformed the standalone SVR. This supports the argument by Khashei and Bijari (2011) that hybrid models must be carefully designed, complexity alone doesn’t guarantee better results.

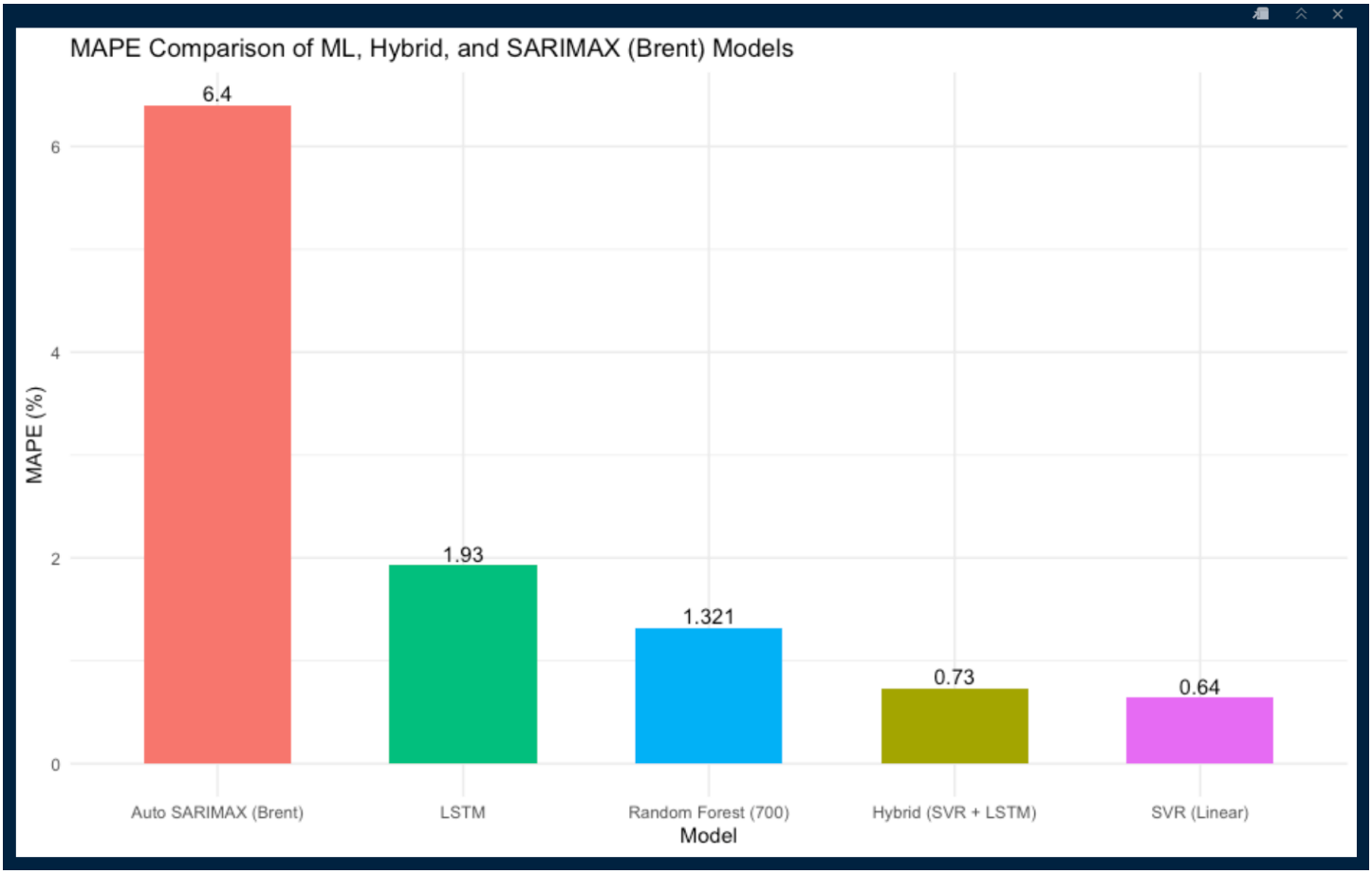


Figure 1. Comparison of Forecast Accuracy (MAPE) Across Various Models (Source: Author)

As shown in the histogram above, the SVR model has a lower MAPE value than the Hybrid model, when plotted.

2.6 Limitations of Past Research

Previous research has often relied on univariate models without checking stationarity, reducing reliability. Many studies ignore internal behavioural data or fail to justify exogenous feature choices (Zhang et al., 2018; Li et al., 2019). DL models are frequently used as black boxes with little interpretability, and hybrid approaches are sometimes implemented with arbitrary combinations and no weight tuning.

This study addresses these issues by:

- Using robust metrics (RMSE, MAE, MAPE) to evaluate performance.
- Testing feature relevance via Granger causality.

- Comparing well-tuned standalone models to a strategically weighted hybrid.

3. METHODOLOGY

3.1 Introduction

This chapter outlines the methodological framework used to address the project's objectives. The workflow is based on a CRISP-DM-inspired approach (Saltz, 2021), combining classical time series methods with modern ML techniques. It covers data understanding, cleaning, modelling, evaluation, and final forecasting in a robust and reproducible way.

3.2 Dataset Description

The main dataset, sourced from the UK Government's Energy Statistics archive via the UoGM contains weekly fuel price data (pence per litre) from January 2021 to February 2025. Key variables include:

- Week_Ending: The weekly dates of observation.
- Avg_Fuel_Price: The weekly average fuel price.
- Quantity_Per_Transaction: Average litres bought per transaction (an internal regressor).
- Fuel_Sales_Index: A measure of weekly fuel sales volume.

An external dataset from the U.S. Energy Information Administration (EIA) provides weekly Europe Brent Spot Prices, used as an external regressor. After cleaning, over 210 weekly observations were retained for modelling.

3.3 Methodological Flowchart

Figure 2 below contains the flowchart illustrating the end-to-end data analytics and modelling process undertaken for fuel price forecasting. The modelling pipeline began with data loading and preparation (Segment 1), decomposition and stationarity testing (Segment 2), and an 80/20 train-test split (Segment 3).

- Statistical models: SARIMA (Segment 3), SARIMAX with internal regressors (Segment 4), and SARIMAX with Brent prices (Segment 5).

- ML/DL models: Random Forest (Segment 8), LSTM (Segment 9), SVR (Segment 10).

Segment 6 compared all statistical models, while Segment 7 used the best statistical model one for forecasting. Segment 11 developed a hybrid SVR + LSTM model, which was evaluated in Segment 12 alongside all models to identify the best performer and perform the final deployment.

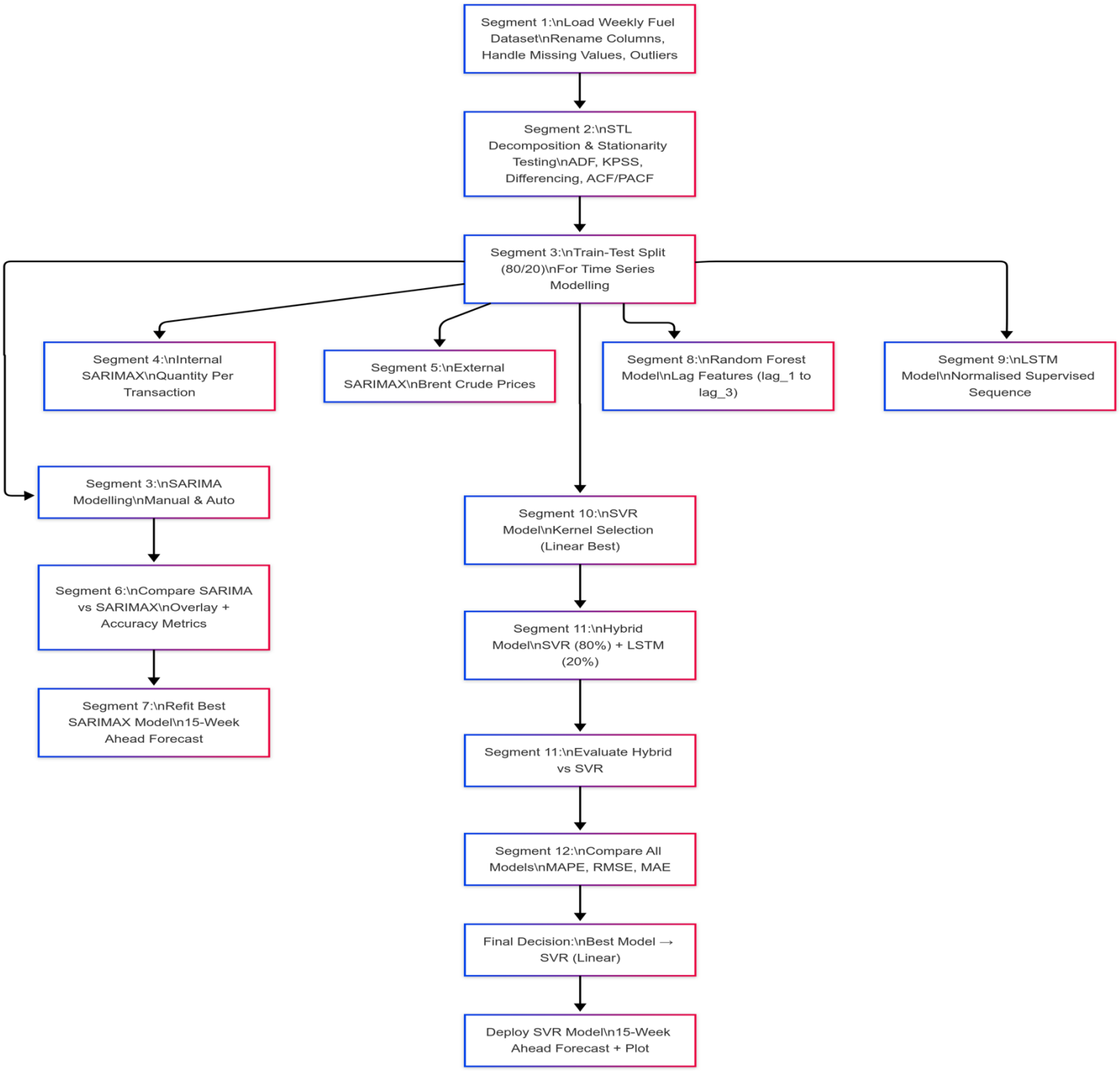


Figure 2. Overall Modelling Workflow (Source: Author)

3.4 Step-By-Step Methodology

3.4.1 Data Cleaning and Preprocessing

- Missing values in key variables were filled using linear interpolation (via zoo::na.approx()).
- Outliers were detected with boxplots and IQR rules, then replaced using interpolation for continuity.
- Dates were normalised to weekly periods using floor_date(), and time series were created using ts() with 52-week frequency

3.4.2 Stationarity and Decomposition

To ensure model assumptions were met:

- STL decomposition separated each series into trend, seasonal and residual components.
- Augmented Dickey-Fuller (ADF) and KPSS tests confirmed the presence of non-stationarity.
- First-order differencing was applied where needed and tests were repeated across different modelling contexts to maintain validity after preprocessing

3.4.3 Feature Engineering

- Lag features (lag_1, lag_2, lag_3) were created for ML models to capture temporal dependencies.
- Internal regressor: Quantity_Per_Transaction (consumer behaviour)
- External regressor: Brent_Price (market signal)
- Granger causality tests were conducted to validate the influence of these regressors

3.4.4 Models Developed

The study tested seven forecasting models, grouped into three types, Table 1 below captures the names and categories of the incorporated their descriptions:

Category	Model	Description

Statistical	SARIMA (Univariate)	Classic ARIMA with seasonal structure
Statistical	SARIMAX (Internal & External)	With internal & Brent regressors
Machine Learning	SVR (Linear Kernel)	Support Vector Regression with lagged input
Machine Learning	Random Forest (ntree = 700)	Tree ensemble with 3-lag structure
Deep Learning	LSTM	Recurrent Neural Network with 3-lag input
Hybrid	SVR + LSTM (Weighted 80:20)	Weighted ensemble favouring SVR

Table 1. Forecasting Models' Details. (Source: Author)

3.5 Model Evaluation

All models were trained on an 80/20 chronological split to avoid data leakage. Performance was measured using:

- **RMSE** – Sensitive to large errors
- **MAE** – Measures average error magnitude
- **MAPE** – Percentage-based error metric used for model comparison

SVR (linear kernel) emerged as the top performer with the lowest MAPE, showing high predictive consistency.

3.6 Forecasting Strategy

After initial evaluation, the top three statistical models; Manual SARIMA (univariate), Internal SARIMAX, and External SARIMAX (Brent) were compared. The External SARIMAX model outperformed others.

Among all models, SVR delivered the best forecast accuracy and was retrained on the full dataset for a 15-week forecast.

A hybrid model (SVR + LSTM, weighted 80:20) was also developed. Although it improved upon other models, it didn't outperform standalone SVR.

3.7 Justification of Methods

- CRISP-DM ensures structured and iterative refinement.
- Combining traditional and modern approaches provides interpretability and flexibility.
- Use of internal and external regressors allows for comparative insight into local vs global price influences.
- Hybrid technique was tested to exploit complementary strengths in SVR and LSTM.

3.8 Limitations of Methodology

- The LSTM model used a basic architecture and may benefit from further tuning.
- Brent prices, while useful, have limited short-term responsiveness compared to internal behavioural signals.
- Hybrid models require more sophisticated design (e.g. dynamic weighting or stacking) to consistently outperform best-in-class models.

4. IMPLEMENTATION AND RESULTS

4.1 Introduction to Implementation

This chapter presents the step-by-step implementation of forecasting models for UK weekly fuel prices, using both traditional time series and advanced machine learning techniques. The workflow follows the CRISP-DM methodology (Saltz, 2021), and each step is structured to address the five research questions outlined in Chapter 1. Evaluation is conducted using RMSE, MAE, and MAPE to guide decision-making for stakeholders like analysts and policymakers.

4.2 Data Preparation and Initial Exploration

Segment 1: Loading, Exploring and Preparing the Dataset

This segment laid the foundation by importing the dataset, establishing a clean working environment and ensuring data readiness.

Step 1: Load Packages & Initialise Environment

The R environment was cleared (rm(list = ls())) and memory freed. A load_packages() function ensured all required libraries were loaded, covering wrangling (dplyr, readxl), modelling (forecast, keras) and visualisation (ggplot2, patchwork). Conflicts (e.g., filter() in dplyr vs stats) were noted and handled accordingly.

```r
{r packages}
# Clear R environment to ensure a clean workspace
rm(list = ls())
# Perform garbage collection to free memory
gc()

# List of required packages for data processing, visualization, and modeling
packages <- c("tidyverse", "readxl", "lubridate", "janitor", "zoo", "ggplot2", "keras",
"randomForest",
              "forecast", "tseries", "urca", "corrplot", "caret", "Metrics", "gt", "lmtest",
"reshape2",
              "e1071", "keras", "tensorflow", "TTR", "scales", "kableExtra", "tibble",
"patchwork")

# Function to load or install packages as needed
load_packages <- function(pkg_list) {
  for (pkg in pkg_list) {
    if (!require(pkg, character.only = TRUE)) {
      # Install package with dependencies if not already installed
      install.packages(pkg, dependencies = TRUE)
      # Load the package
      library(pkg, character.only = TRUE)
    }
  }
}

# Execute package loading function
load_packages(packages)
```

Figure 3. Screenshot of step 1 code. (Source: Author)

Step 2: Load and Clean Dataset

The fuel_price.xlsx dataset was imported, skipping the first five metadata rows. Columns were renamed for clarity (Week_Ending → Date, Avg_Fuel_Price → avg_price) and class coercion was applied (Date → Date, avg_price → numeric). Initial exploration (glimpse(), summary()) revealed 214 weekly observations and two missing values.

```r
{r load-clean-data}
# Load fuel price dataset from Excel, skipping metadata rows
fuel_data <- read_excel("fuel_price.xlsx", sheet = "Table 1 Weekly", skip = 5)

# Assign column names for clarity
colnames(fuel_data) <- c("Week_Ending", "Avg_Fuel_Price", "Quantity_Per_Transaction", "Fuel_Sales_Index")

# Convert Week_Ending to Date format, ensure Avg_Fuel_Price is numeric, and select relevant columns
fuel_data <- fuel_data %>%
  mutate(Week_Ending = as.Date(Week_Ending),
         Avg_Fuel_Price = as.numeric(Avg_Fuel_Price)) %>%
  select(Date = Week_Ending, avg_price = Avg_Fuel_Price)

# Display dataset structure to verify data types and format
glimpse(fuel_data)
# Summarize dataset to check for basic statistics and potential issues
summary(fuel_data)
```

```
Rows: 214
Columns: 2
$ Date      <date> 2021-01-31, 2021-02-07, 2021-02-14, 2021-02-21, 2021-02-28, 2021-03-07, 2021-03-14, 2021-03-
21, 2021-03-28, 2021-04-04, 2021-04-11, 2021-04-18…
$ avg_price <dbl> 92.60980, 94.46735, 95.85136, 97.41086, 98.31273, 99.86002, 100.68783, 103.01901, 104.26535,
110.50364, 112.61229, 113.59467, 114.25268, 115.22…
      Date              avg_price
 Min.   :2021-01-31   Min.   :  0.7538
 1st Qu.:2022-02-07   1st Qu.: 92.0606
 Median :2023-02-15   Median :102.8220
 Mean   :2023-02-15   Mean   :108.7929
 3rd Qu.:2024-02-23   3rd Qu.:120.7913
 Max.   :2025-02-16   Max.   :592.9027
                      NA's   :2
```

Figure 4. Screenshot of step 2 code and outcome. (Source: Author)

Step 3: Identify & Handle Missing Values

Two missing values in avg_price were imputed using linear interpolation (zoo::na.approx()), maintaining temporal continuity without distorting trend structure.

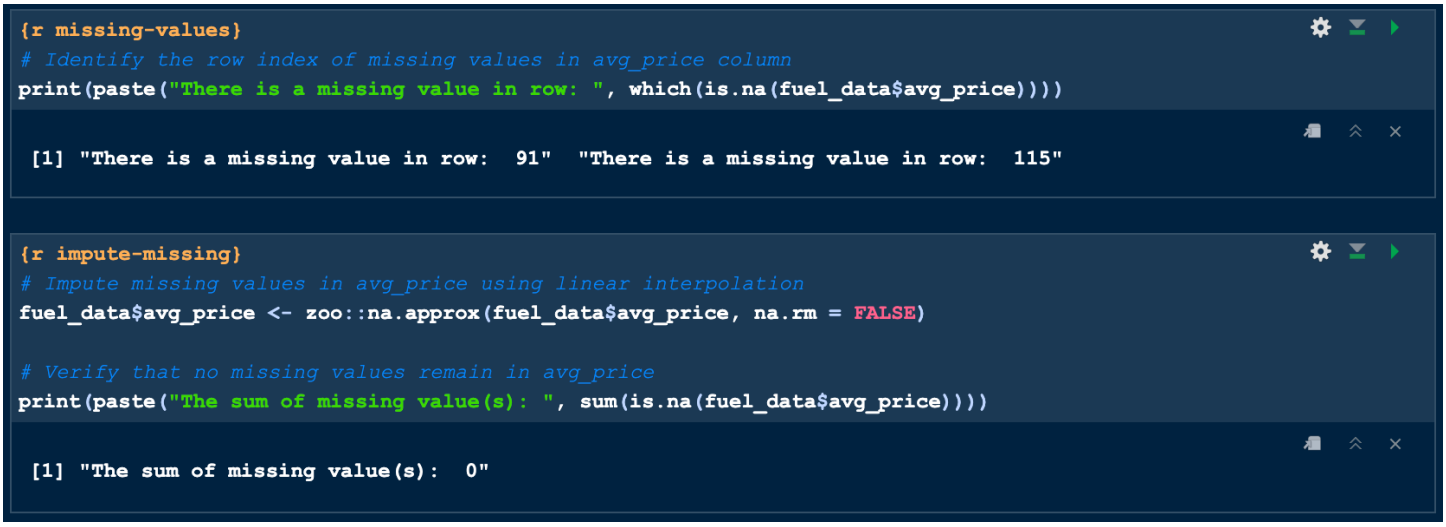


Figure 5. Screenshot of step 3 code and outcome. (Source: Author)

Linear interpolation was chosen as it aligns with the sequential nature of time series data, avoiding disruptions that random imputation might introduce.

Step 4: Detect & Handle Outliers

Detection: Outliers were identified using boxplots and the IQR method (Q1 − 1.5×IQR, Q3 + 1.5×IQR), confirming extreme values like 0.75 and 592.9.

Treatment: Two strategies were evaluated:

- Winsorisation: Capped values at the IQR thresholds.
- Interpolation: Replaced extreme values with interpolated values for smoother trend preservation. This strategy was preferred.

Example: 592.9 (Dec 2024) was interpolated to 92.8; 0.75 (Sept 2024) to 90.9.

```r
{r boxplot-outliers}
# Create a boxplot to visualize potential outliers in avg_price
fuel_data %>%
  ggplot(aes(y = avg_price)) +
  geom_boxplot(fill = "orange", alpha = 0.9, outlier.color = "red") +
  labs(title = "Boxplot of Weekly Average Fuel Prices",
       y = "Fuel Price (Pence per Litre)",
       x = "") +
  theme_minimal()
```

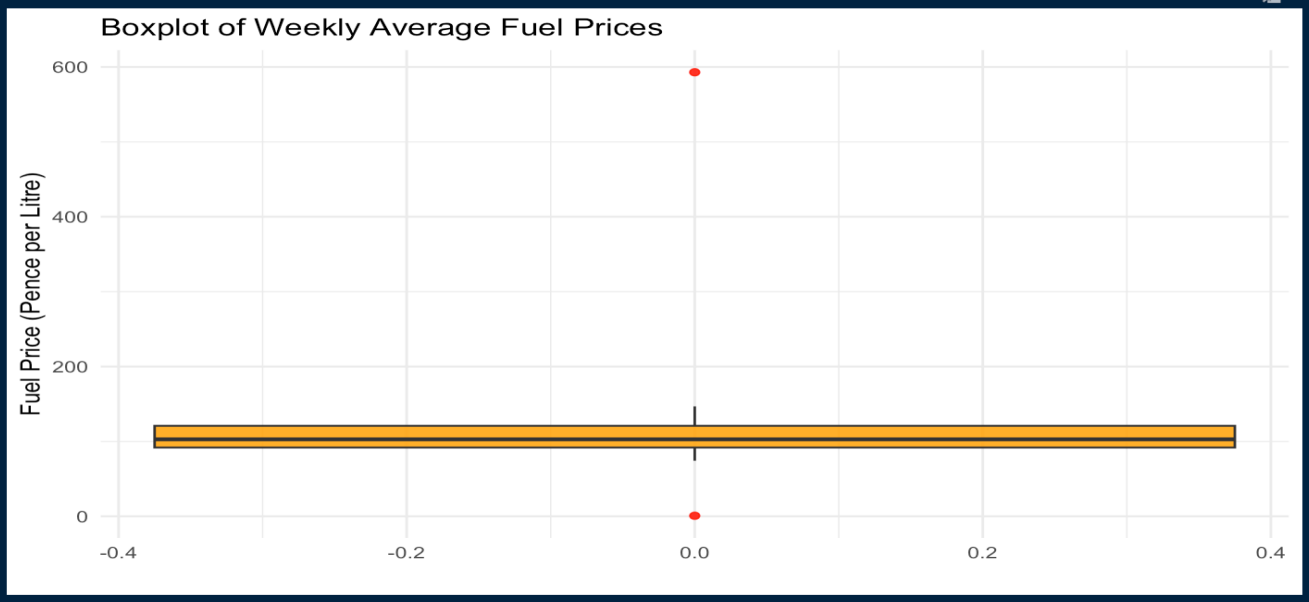


Figure 6. Screenshot of step 4, boxplot outliers. (Source: Author)

```r
{r detect-outliers}
# Calculate Interquartile Range (IQR) to identify outliers
Q1 <- quantile(fuel_data$avg_price, 0.25, na.rm = TRUE)
Q3 <- quantile(fuel_data$avg_price, 0.75, na.rm = TRUE)
IQR_val <- Q3 - Q1
lower_bound <- Q1 - 1.5 * IQR_val
upper_bound <- Q3 + 1.5 * IQR_val

# Identify indices of outliers in avg_price
outlier_idx <- which(fuel_data$avg_price < lower_bound | fuel_data$avg_price > upper_bound)

# Display rows with outliers for inspection
fuel_data[outlier_idx, ]
```

A tibble: 2 × 2

Date <date>	avg_price <dbl>
2024-09…	0.753758
2024-12…	592.902…

2 rows

Figure 7. Screenshot of step 4, IQR outliers. (Source: Author)

```r
{r compare-outlier-fixes}
# Apply winsorisation to cap outliers at IQR bounds
fuel_data$winsorised <- fuel_data$avg_price
fuel_data$winsorised[outlier_idx] <- ifelse(
  fuel_data$winsorised[outlier_idx] > upper_bound, upper_bound, lower_bound
)

# Display original and winsorised values for outlier rows
fuel_data[outlier_idx, c("Date", "avg_price", "winsorised")]

# Apply interpolation to replace outliers with interpolated values
fuel_data$interpolated <- fuel_data$avg_price
fuel_data$interpolated[outlier_idx] <- NA
fuel_data$interpolated <- zoo::na.approx(fuel_data$interpolated, na.rm = FALSE)

# Compare original, winsorised, and interpolated values for outliers
fuel_data[outlier_idx, c("Date", "avg_price", "winsorised", "interpolated")]
```

tbl_df tbl_df

A tibble: 2 × 4

Date <date>	avg_price <dbl>	winsorised <dbl>	interpolated <dbl>
2024-09...	0.753758	48.33433	90.85949
2024-12...	592.902...	164.37830	92.76982

2 rows

```r
{r apply-outlier-fix}
# Replace avg_price with interpolated values to handle outliers
fuel_data <- fuel_data %>%
  select(Date, avg_price = interpolated)
```

Figure 8. Screenshot of step 4, Outlier handling. (Source: Author)

Outcome of Segment 1

- Dataset was free of missing values.
- Outliers were addressed without distorting seasonality.
- The cleaned dataset was ready for time series conversion and modelling.

Segment 2: Decomposition, Stationarity Testing, and Transformation

This segment ensured statistical assumptions were met by converting the cleaned data into a time series object and preparing it for ARIMA-based modelling.

Step 1: Create Weekly Time Series Object

The cleaned avg_price series was converted into a ts() object with weekly frequency (52), using the earliest date and week as origin.

```
{r create-ts}
# Select relevant columns and remove any rows with missing values
fuel_data_ts <- fuel_data %>% select(Date, avg_price) %>% drop_na()

# Extract start year and week for time series creation
start_year <- lubridate::year(min(fuel_data_ts$Date))
start_week <- lubridate::isoweek(min(fuel_data_ts$Date))

# Convert avg_price to a weekly time series object with 52 periods per year
ts_fuel <- ts(fuel_data_ts$avg_price,
              start = c(start_year, start_week),
              frequency = 52)
```

Figure 9. Screenshot of step 1, ts_fuel creation (Source: Author)

Step 2: STL Decomposition

STL decomposition revealed:

- A pronounced seasonal pattern
- A long-term non-linear trend

These patterns validated the use of seasonal models like SARIMA.

```r
{r stl-decomposition}
# Perform STL decomposition to separate trend, seasonal, and remainder components
decomposed_fuel <- stl(ts_fuel, s.window = "periodic")

# Plot decomposed components for visual analysis
autoplot(decomposed_fuel) +
  ggtitle("STL Decomposition of Weekly Fuel Price Series") +
  theme_minimal()
```

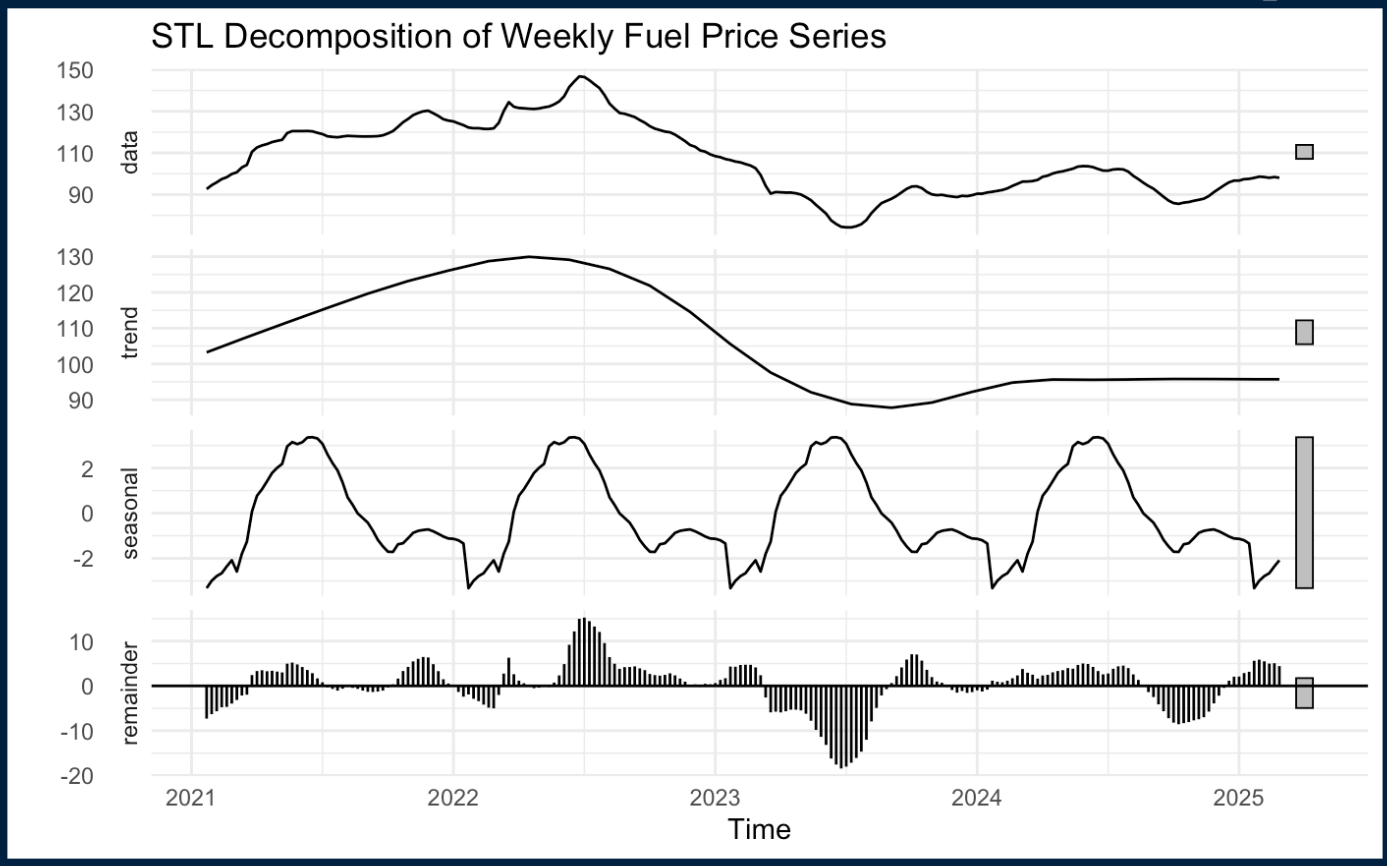


Figure 10. Screenshot of step 2, STL Decomposition Plot. (Source: Author)

Step 3: Augmented Dickey-Fuller (ADF) Test

The Augmented Dickey-Fuller test returned a p-value > 0.38, indicating non-stationarity.

```r
{r adf-test}
# Conduct Augmented Dickey-Fuller test to check for stationarity
adf_result <- tseries::adf.test(ts_fuel)
# Display test results
print(adf_result)
```

```
        Augmented Dickey-Fuller Test

data:  ts_fuel
Dickey-Fuller = -2.4537, Lag order = 5, p-value = 0.3857
alternative hypothesis: stationary
```

Figure 11. Screenshot of step 3, ADF test & result. (Source: Author)

Step 4: KPSS Test

To validate the ADF findings, the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test was conducted:

- Null Hypothesis: The series is non-stationary.
- Test Statistic: 0.4105, exceeding the 5% critical value of 0.146.

This result further confirmed the non-stationarity of the original series, aligning with the ADF test and necessitating corrective measures.

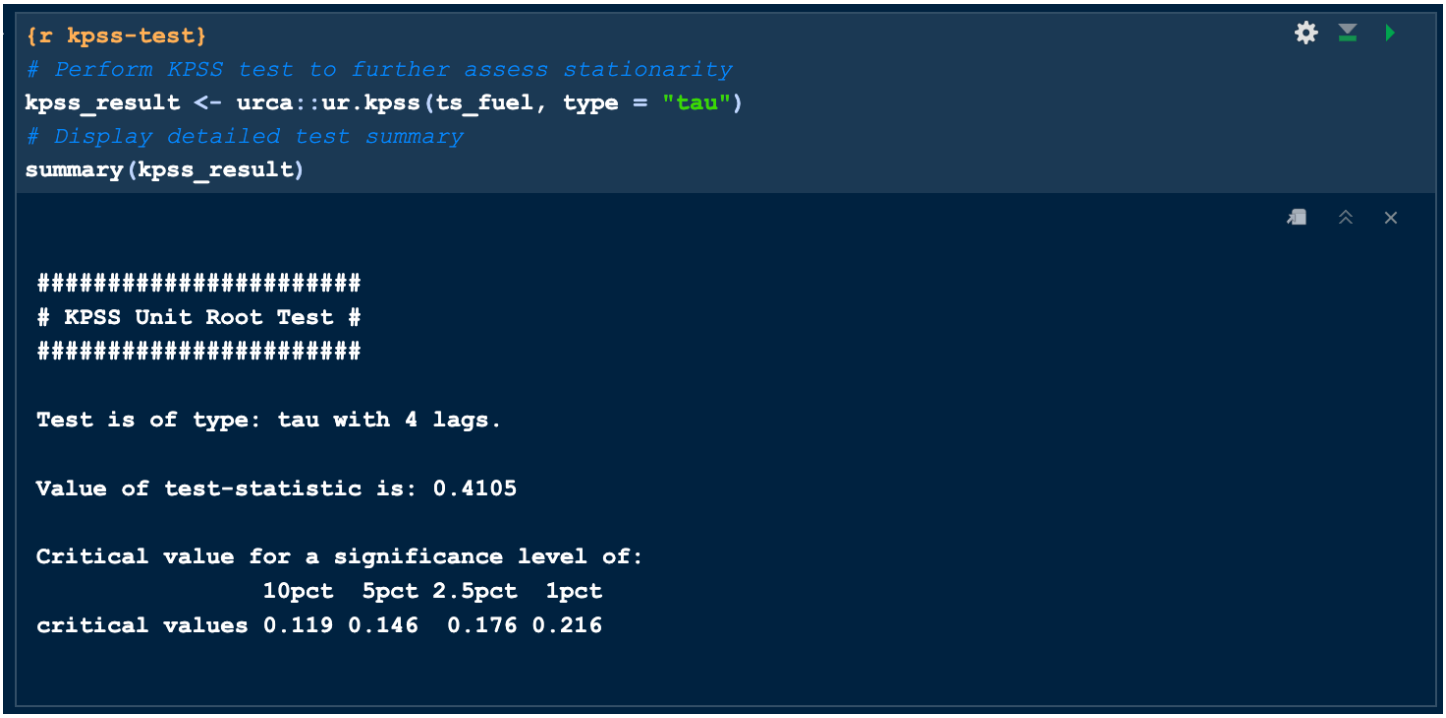


Figure 12. Screenshot of step 4, KPSS result. (Source: Author)

Step 5: Apply Differencing

First-order differencing was applied using diff(ts_fuel) to remove the trend and stabilise the series. A plot of the differenced series showed no distinct trend, with fluctuations appearing consistent across the timeline, suggesting stationarity.

```r
{r differencing}
# Apply first-order differencing to make the series stationary
diff_fuel <- diff(ts_fuel)

# Plot differenced series to inspect stationarity
autoplot(diff_fuel) +
  ggtitle("First-Order Differenced Fuel Price Series") +
  xlab("Time (Weeks)") +
  ylab("Differenced Fuel Price") +
  theme_minimal()
```

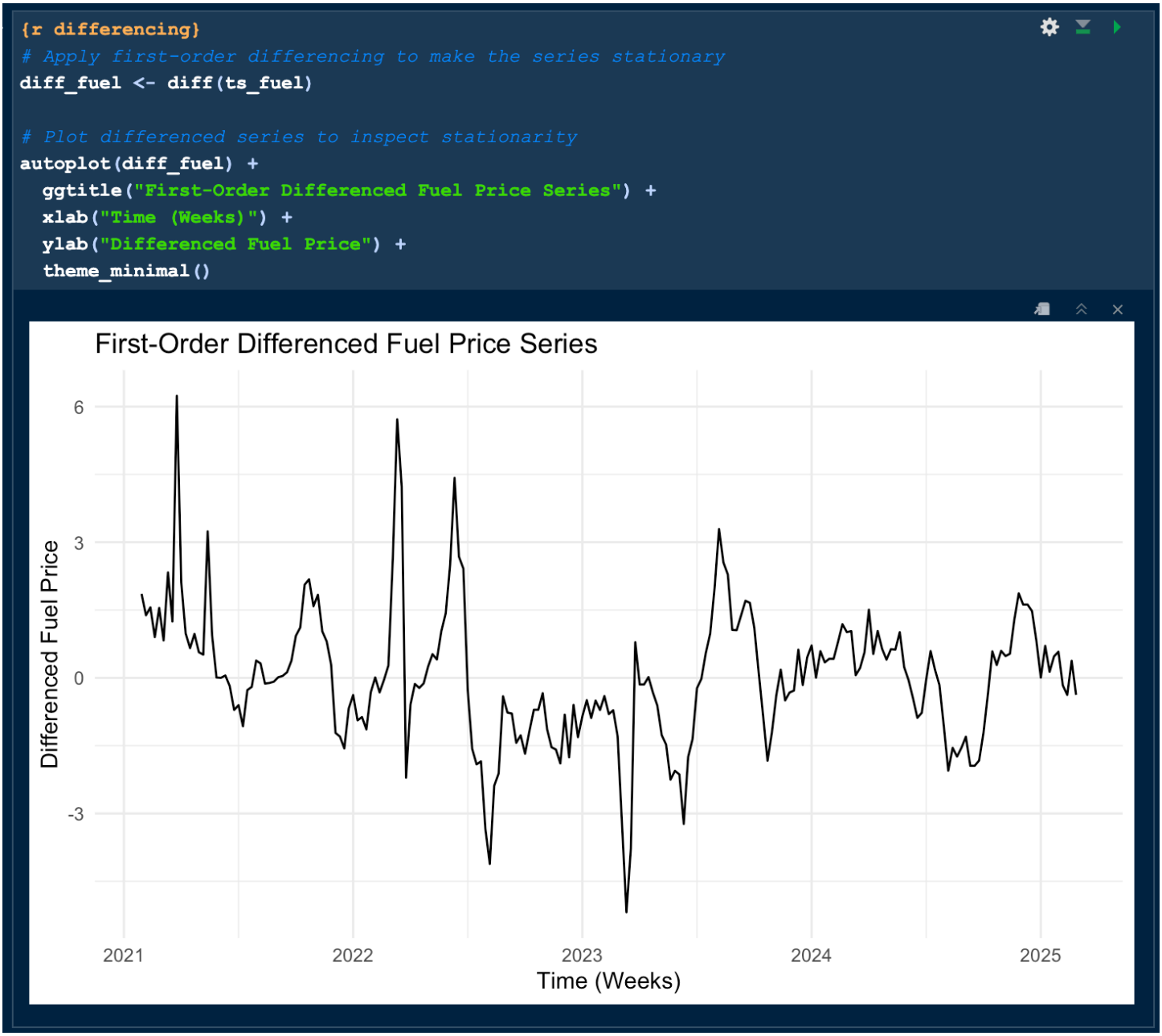


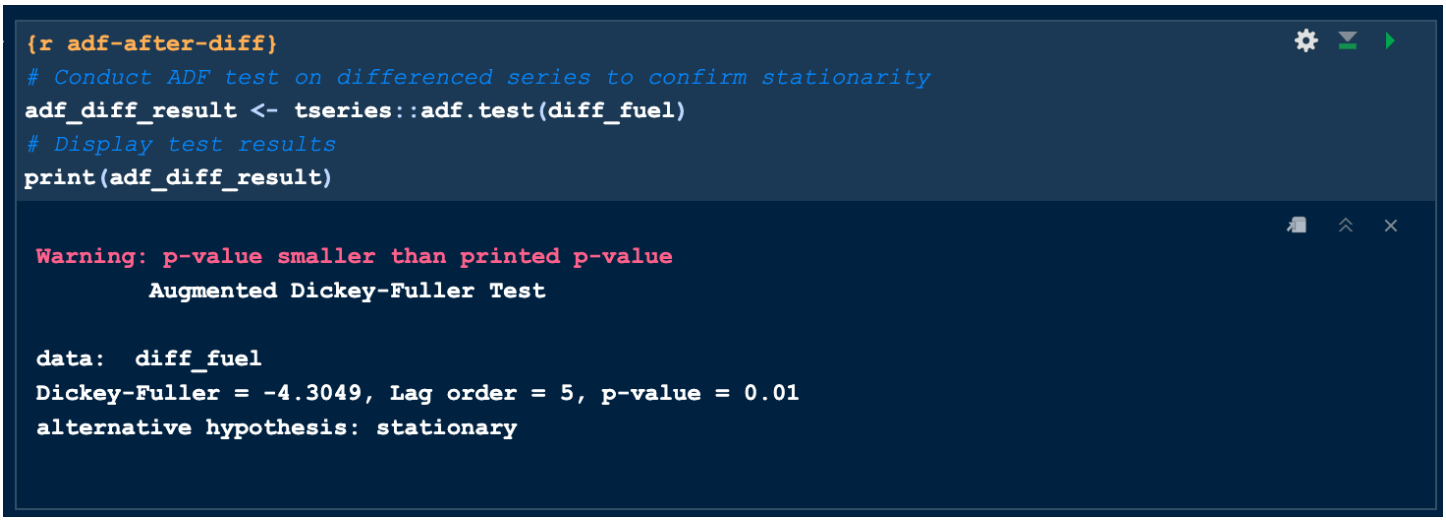
Figure 13. Screenshot of step 5, Differenced Fuel Price Plot. (Source: Author)

Step 6: ADF Test on Differenced Series

The ADF test was repeated on the differenced series to verify stationarity:

- Test Statistic: -4.3049
- p-value: < 0.01

The low p-value strongly rejects the null hypothesis, confirming that the differenced series is stationary and suitable for ARIMA-based modelling.



```
{r adf-after-diff}
# Conduct ADF test on differenced series to confirm stationarity
adf_diff_result <- tseries::adf.test(diff_fuel)
# Display test results
print(adf_diff_result)
```

```
Warning: p-value smaller than printed p-value
        Augmented Dickey-Fuller Test

data:  diff_fuel
Dickey-Fuller = -4.3049, Lag order = 5, p-value = 0.01
alternative hypothesis: stationary
```

Figure 14. Screenshot of step 6, ADF on differenced series. (Source: Author)

Step 7: ACF and PACF Analysis

ACF plots showed gradual decay, suggesting MA terms. PACF indicated a sharp cutoff, suggesting AR terms. This guided SARIMA order selection.

```r
{r acf-pacf}
# Set up plot layout for side-by-side ACF and PACF plots
par(mfrow = c(1, 2))
# Plot Autocorrelation Function (ACF) to identify lag patterns
acf(diff_fuel, main = "ACF of Differenced Series")
# Plot Partial Autocorrelation Function (PACF) for model order selection
pacf(diff_fuel, main = "PACF of Differenced Series")
# Reset plot layout
par(mfrow = c(1, 1))
```

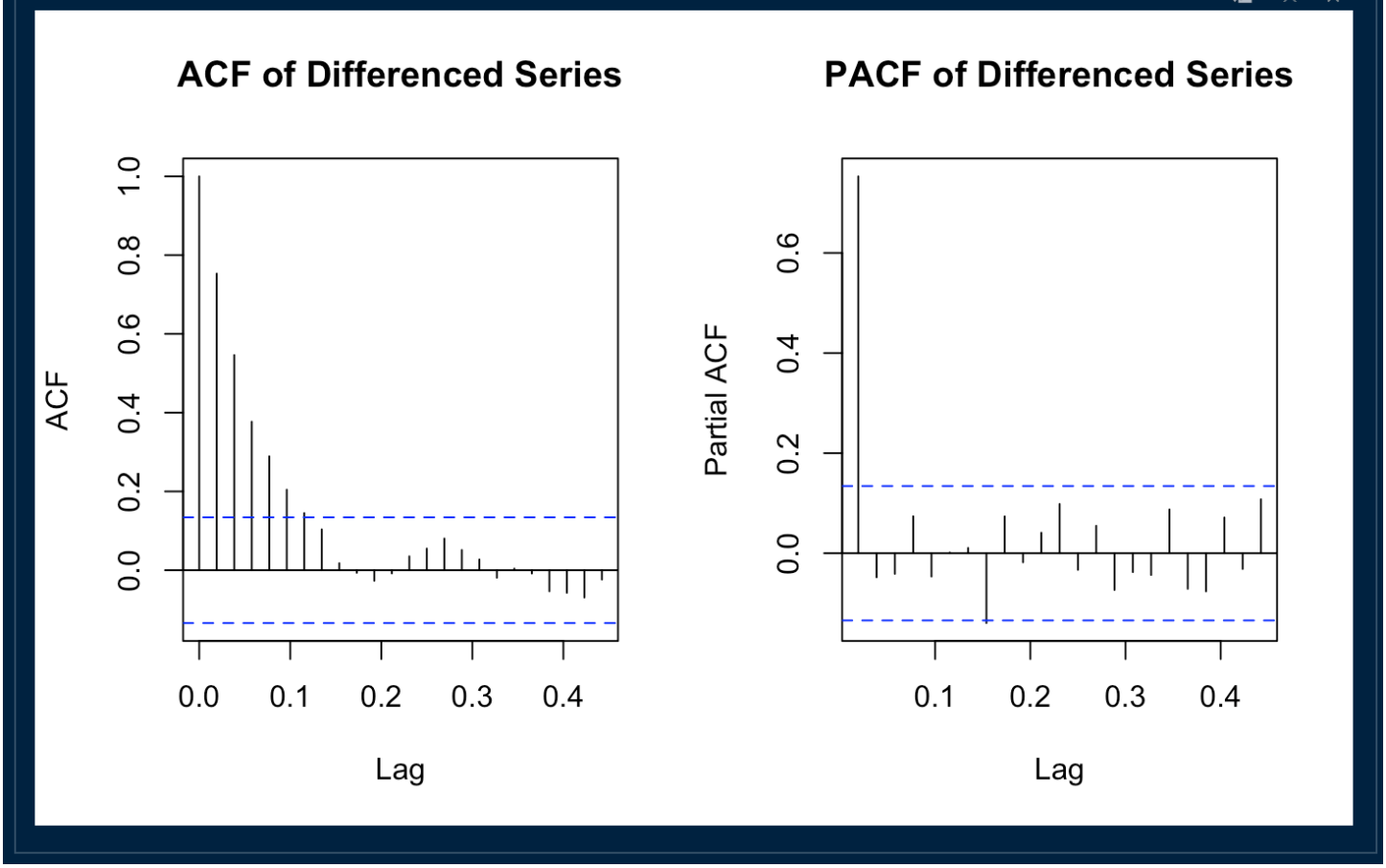


Figure 15. Screenshot of step 7, ACF and PACF Plots. (Source: Author)

Step 8: Log Transformation & Smoothing

For interpretability, a log transformation followed by a 4-point moving average was applied, revealing the smoothed trend line. This was not used in modelling but provided additional clarity.

```r
{r log-transform}
# Apply log transformation to stabilise variance
log_price <- log(ts_fuel)
# Smooth log-transformed series using a moving average filter
smoothed_log <- stats::filter(log_price, filter = rep(1/4, 4), sides = 2)

# Plot smoothed log-transformed series for visualisation
plot(smoothed_log, main = "Smoothed Log-Transformed Fuel Price", ylab = "Log Price", col = "blue")
```

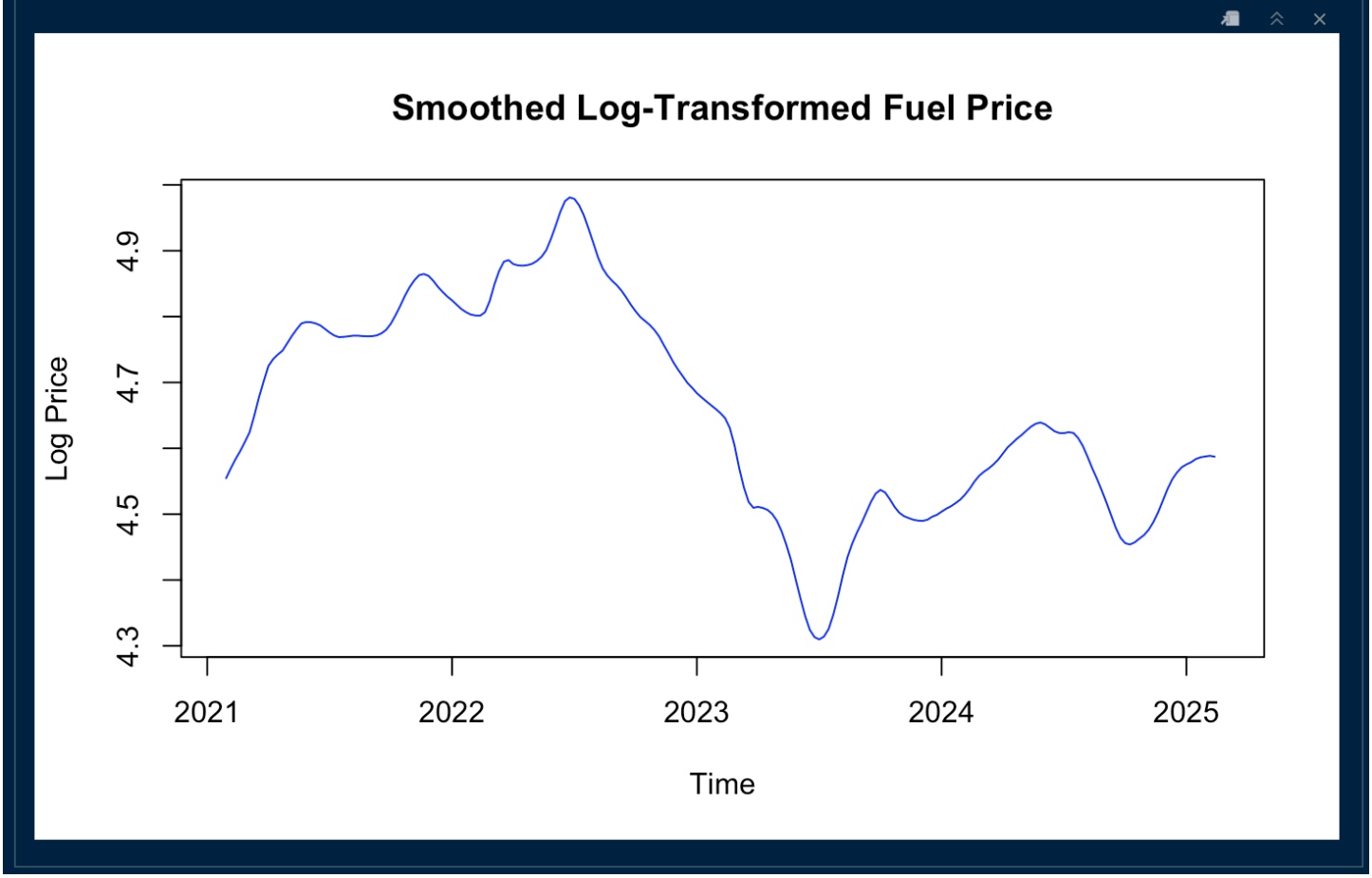


Figure 16. Screenshot of step 8, Smoothed Log-Transformed Fuel Price Plot. (Source: Author)

4.3 Model Development and Evaluation

Segment 3: Train-Test Split and Univariate Forecasting

This segment sets the benchmark by evaluating univariate SARIMA models.

Step 1&2: Create Train-Test Split (80:20)

The weekly ts_fuel series was split using an 80:20 ratio to ensure sequential order and prevent data leakage.

```r
{r split-data}
# Define time series with weekly frequency using full dataset
ts_fuel <- ts(fuel_data$avg_price, frequency = 52, start = c(year(min(fuel_data$Date)), isoweek(min(fuel_data$Date))))

# Calculate index for 80/20 train-test split
n_obs <- length(ts_fuel)
split_index <- floor(0.8 * n_obs)

# Create training and test time series
train_ts <- ts(ts_fuel[1:split_index], start = start(ts_fuel), frequency = 52)
test_ts <- ts(ts_fuel[(split_index + 1):n_obs], start = time(ts_fuel)[split_index + 1], frequency = 52)
```

Step 2: Visualise Train vs Test Split

```r
{r plot-train-test}
# Plot training and test sets to visualise the split
ts.plot(train_ts, test_ts,
        col = c("skyblue", "red"),
        lty = 1:2, lwd = 2,
        main = "Train vs Test Split of Weekly Fuel Prices",
        ylab = "Fuel Price", xlab = "Time")
# Add legend to distinguish training and test sets
legend("topleft", legend = c("Training Set", "Test Set"), col = c("skyblue", "red"), lty = 1:2, lwd = 2)
```

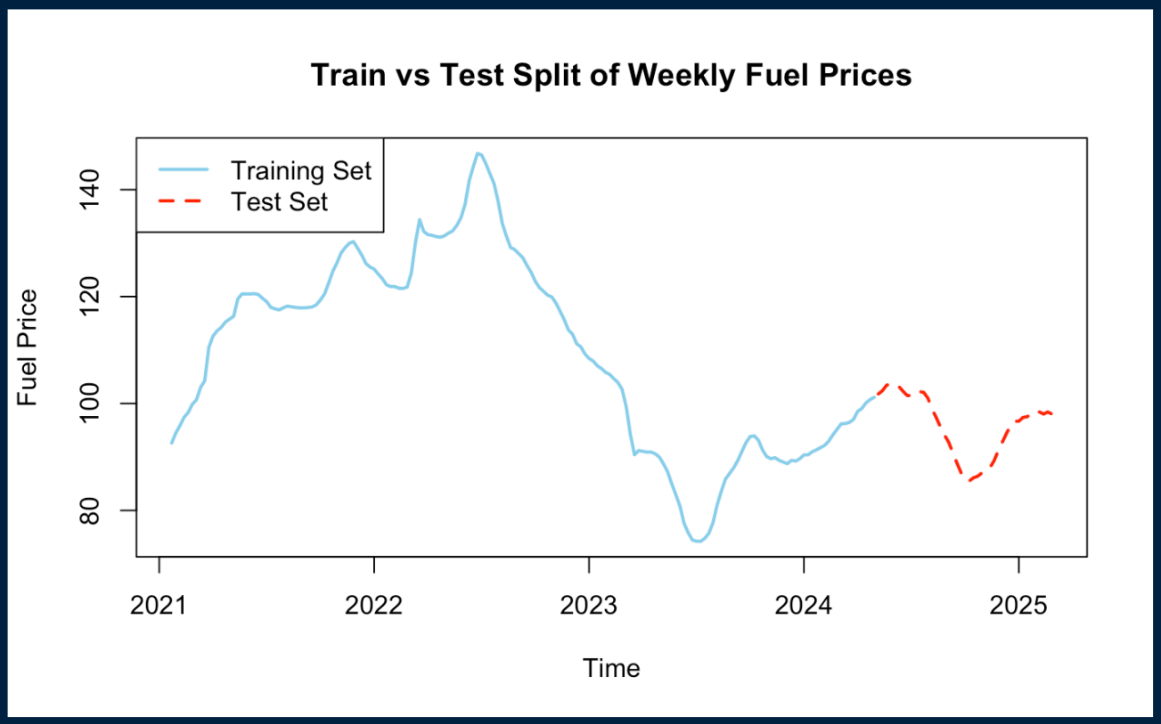


Figure 17. Screenshot of step 1&2, Train vs Test Split. (Source: Author)

Step 3: Fit Manual SARIMA Model

A manual SARIMA model, ARIMA(1,1,1)(1,1,1)[52], was fitted based on ACF/PACF insights. It returned a training RMSE of 1.0481 and MAPE of 0.5638%.

```r
{r manual-sarima}
# Fit a manually specified SARIMA model based on ACF/PACF analysis
manual_sarima <- Arima(train_ts, order = c(1,1,1), seasonal = list(order = c(1,1,1),
period = 52))
# Display model summary
summary(manual_sarima)
```

```
Series: train_ts
ARIMA(1,1,1)(1,1,1)[52]

Coefficients:
         ar1     ma1     sar1     sma1
      0.7170  0.1352  -0.5897  -0.2995
s.e.  0.0795  0.1104   0.1829   0.3440

sigma^2 = 1.648:  log likelihood = -217.11
AIC=444.22   AICc=444.76   BIC=458.08

Training set error measures:
                     ME     RMSE     MAE         MPE      MAPE       MASE          ACF1
Training set -0.09493161 1.048103 0.58499 -0.0776832 0.5638206 0.02346503 -0.001344284
```

Figure 18. Screenshot of step 3, Manual ARIMA on train_ts. (Source: Author)

Step 4: Fit Auto SARIMA Model

An auto-selected SARIMA(2,1,0)(1,0,0)[52] model achieved a training MAPE of 0.4991%, marginally outperforming the manual version.

```r
{r auto-sarima}
# Fit an automatically selected SARIMA model using stepwise selection
auto_sarima <- auto.arima(train_ts, seasonal = TRUE, stepwise = TRUE, approximation = FALSE)
# Display model summary
summary(auto_sarima)
```

```
Series: train_ts
ARIMA(2,1,0)(1,0,0)[52]

Coefficients:
         ar1      ar2     sar1
      0.8765  -0.1183  -0.5294
s.e.  0.0779   0.0778   0.0655

sigma^2 = 0.8001:  log likelihood = -229.78
AIC=467.56   AICc=467.8   BIC=480.1

Training set error measures:
                     ME      RMSE       MAE         MPE      MAPE       MASE
Training set 0.004355887 0.8839414 0.5553199 0.00257259 0.4991371 0.02227491
                    ACF1
Training set -0.006660043
```

Figure 19. Screenshot of step 4, Auto ARIMA on train_ts. (Source: Author)

Step 5: Forecast and Visualise

Forecasts from both models were plotted against the test set.

```r
{r forecast-comparison}
# Define forecast horizon equal to test set length
h <- length(test_ts)
# Generate forecasts for auto and manual SARIMA models
auto_fc <- forecast(auto_sarima, h = h)
manual_fc <- forecast(manual_sarima, h = h)

# Plot auto SARIMA forecast with actual test data
p1 <- autoplot(auto_fc) +
  autolayer(test_ts, series = "Actual", color = "red") +
  ggtitle("Auto SARIMA Forecast vs Actual") +
  xlab("Time") + ylab("Fuel Price (Pence per Litre)") +
  theme_minimal()

# Plot manual SARIMA forecast with actual test data
p2 <- autoplot(manual_fc) +
  autolayer(test_ts, series = "Actual", color = "red") +
  ggtitle("Manual SARIMA Forecast vs Actual") +
  xlab("Time") + ylab("Fuel Price (Pence per Litre)") +
  theme_minimal()

# Display plots side by side for comparison
p1 / p2
```

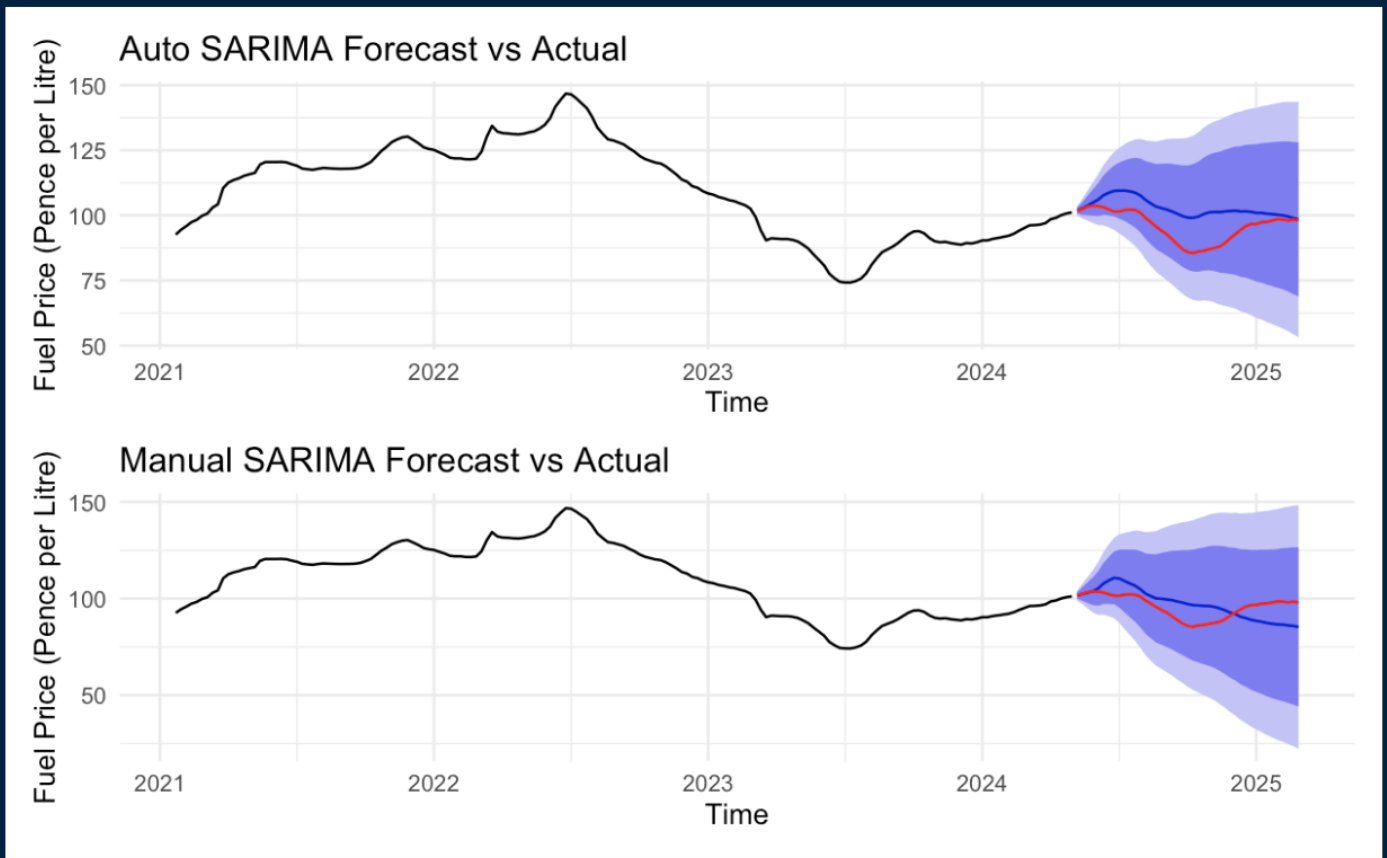


Figure 20. Screenshot of step 5, Auto SARIMA vs Actual | Manual SARIMA vs Actual. (Source: Author)

Step 6: Compare Forecast Accuracy

On the test set, manual SARIMA outperformed auto SARIMA across RMSE and MAPE.

```
{r accuracy-metrics}
# Convert forecasts to time series aligned with test set
manual_pred <- ts(as.numeric(manual_fc$mean), start = start(test_ts), frequency = frequency
(test_ts))
auto_pred   <- ts(as.numeric(auto_fc$mean), start = start(test_ts), frequency = frequency
(test_ts))

# Calculate accuracy metrics for manual and auto SARIMA forecasts
manual_accuracy <- forecast::accuracy(manual_pred, test_ts)
auto_accuracy   <- forecast::accuracy(auto_pred, test_ts)

# Display accuracy metrics side by side for comparison
rbind(
  Manual_SARIMAX = manual_accuracy["Test set", ],
  Auto_SARIMAX = auto_accuracy["Test set", ]
)
```

```
                    ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
Manual_SARIMAX -1.425079 7.894336 6.932134 -1.709194 7.349354 0.9447695  7.539371
Auto_SARIMAX   -6.928860 8.255899 6.928860 -7.496441 7.496441 0.9273980  8.188218
```

Figure 21. Screenshot of step 6, Forecast Accuracy Comparison for Manual & Auto ARIMA. (Source: Author)

Model	RMSE	MAE	MAPE
Manual SARIMA	7.89	6.93	7.34%
Auto SARIMA	8.25	6.92	7.49%

Table 2. Forecast Accuracy Comparison For Manual & Auto ARIMA Using Key Metrics. (Source: Author)

4.4 Multivariate Modelling With Internal Variable

Segment 4: Internal Exogenous Multivariate Forecasting (SARIMAX)

This segment investigates the impact of internal behavioural data, specifically Quantity_Per_Transaction, in multivariate SARIMAX models.

Step 1&2: Load and Prepare Dataset for Multivariate Forecasting

Dataset was reloaded and cleaned. Missing values were interpolated; outliers were identified via IQR and smoothed

```
{r load-full-dataset, message=FALSE, warning=FALSE}
# Load raw fuel price dataset from Excel
fuel_raw <- read_excel("fuel_price.xlsx", sheet = "Table 1 Weekly", skip = 5)

# Rename columns for consistency and clarity
colnames(fuel_raw) <- c("Week_Ending", "Avg_Fuel_Price", "Quantity_Per_Transaction",
"Fuel_Sales_Index")

# Inspect dataset structure and summary statistics
str(fuel_raw)
summary(fuel_raw)
```

```
tibble [214 × 4] (S3: tbl_df/tbl/data.frame)
 $ Week_Ending             : POSIXct[1:214], format: "2021-01-31" "2021-02-07" "2021-02-14"
"2021-02-21" ...
 $ Avg_Fuel_Price          : num [1:214] 92.6 94.5 95.9 97.4 98.3 ...
 $ Quantity_Per_Transaction: num [1:214] 97.2 95.4 95.9 92.9 92.1 ...
 $ Fuel_Sales_Index        : chr [1:214] "65.756879400000003" "67.471288000000001"
"67.300106400000004" "69.005771199999998" ...
  Week_Ending                    Avg_Fuel_Price     Quantity_Per_Transaction Fuel_Sales_Index
 Min.   :2021-01-31 00:00:00.00   Min.   :  0.7538   Min.   : 79.83           Length:214
 1st Qu.:2022-02-07 18:00:00.00   1st Qu.: 92.0606   1st Qu.: 93.80           Class :character
 Median :2023-02-15 12:00:00.00   Median :102.8220   Median : 98.04           Mode  :character
 Mean   :2023-02-15 08:51:35.32   Mean   :108.7929   Mean   : 98.05
 3rd Qu.:2024-02-23 06:00:00.00   3rd Qu.:120.7913   3rd Qu.:102.55
 Max.   :2025-02-16 00:00:00.00   Max.   :592.9027   Max.   :114.02
                                  NA's   :2          NA's   :2
```

Figure 22. Screenshot of step 1, showing raw dataset summary. (Source: Author)

```r
{r clean-dataset}
# Clean and preprocess dataset: convert to Date, ensure numeric columns, and select relevant variables
fuel_clean <- fuel_raw %>%
  mutate(
    Date = as.Date(Week_Ending),
    avg_price = as.numeric(Avg_Fuel_Price),
    Quantity_Per_Transaction = as.numeric(gsub(",", "", Quantity_Per_Transaction))
  ) %>%
  select(Date, avg_price, Quantity_Per_Transaction) %>%
  arrange(Date)

# Impute missing values in avg_price and Quantity_Per_Transaction
fuel_clean$avg_price <- zoo::na.approx(fuel_clean$avg_price)
fuel_clean$Quantity_Per_Transaction <- zoo::na.approx(fuel_clean$Quantity_Per_Transaction)

# Define function to handle outliers using IQR method and interpolation
handle_outliers <- function(series) {
  Q1 <- quantile(series, 0.25, na.rm = TRUE)
  Q3 <- quantile(series, 0.75, na.rm = TRUE)
  IQR_val <- Q3 - Q1
  lower <- Q1 - 1.5 * IQR_val
  upper <- Q3 + 1.5 * IQR_val
  series[series < lower | series > upper] <- NA
  zoo::na.approx(series)
}

# Apply outlier handling to avg_price and Quantity_Per_Transaction
fuel_clean$avg_price <- handle_outliers(fuel_clean$avg_price)
fuel_clean$Quantity_Per_Transaction <- handle_outliers(fuel_clean$Quantity_Per_Transaction)
summary(fuel_clean)
```

```
      Date              avg_price      Quantity_Per_Transaction
 Min.   :2021-01-31   Min.   : 74.22   Min.   : 84.14
 1st Qu.:2022-02-07   1st Qu.: 91.85   1st Qu.: 93.60
 Median :2023-02-15   Median :102.52   Median : 98.04
 Mean   :2023-02-15   Mean   :106.85   Mean   : 98.09
 3rd Qu.:2024-02-23   3rd Qu.:120.54   3rd Qu.:102.53
 Max.   :2025-02-16   Max.   :146.76   Max.   :114.02
```

Figure 23. Screenshot of step 2, showing cleaned dataset summary. (Source: Author)

Step 3: Create and Explore Time Series

STL decomposition revealed strong seasonality, justifying SARIMA-type models.

```r
{r internal-ts-create}
# Convert avg_price and Quantity_Per_Transaction to time series objects
ts_y <- ts(fuel_clean$avg_price, frequency = 52, start = c(2021, 1))
ts_x <- ts(fuel_clean$Quantity_Per_Transaction, frequency = 52, start = c(2021, 1))

# Perform STL decomposition on fuel price series for trend and seasonality analysis
autoplot(stl(ts_y, s.window = "periodic")) +
  ggtitle("STL Decomposition of Fuel Prices") + theme_minimal()
```

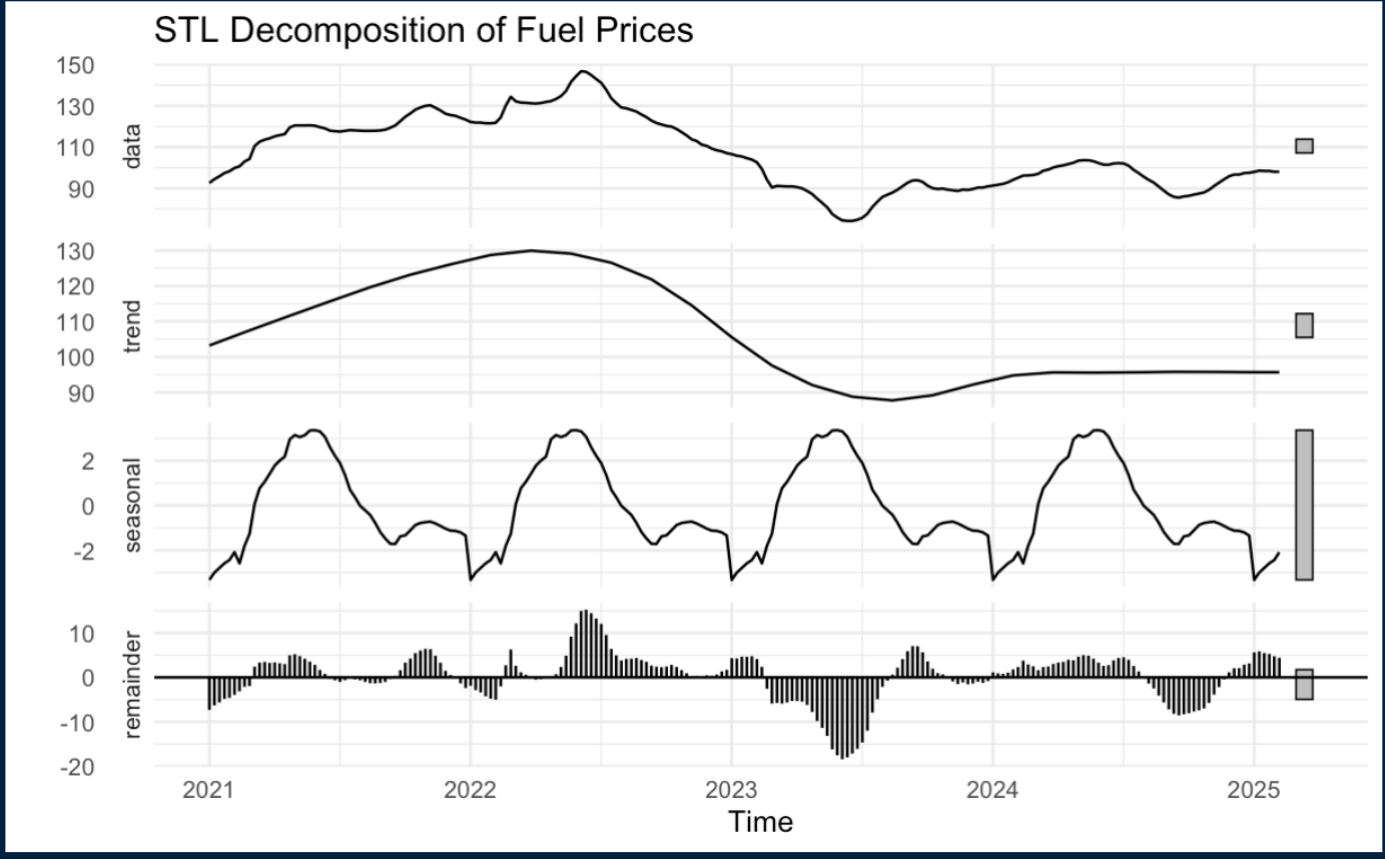


Figure 24. Screenshot of step 3, STL decomposition plot. (Source: Author)

This analysis provided insights into the structural patterns of fuel prices, guiding subsequent model specification.

Steps 4 & 5: Stationarity Testing and Differencing

ADF and KPSS tests confirmed non-stationarity. First-order differencing was applied to both series.

```r
{r internal-stationarity-tests}
# Conduct ADF test to check stationarity of fuel price series
print(adf.test(ts_y))
# Perform KPSS test to complement stationarity analysis
kpss_result <- ur.kpss(ts_y, type = "tau")
summary(kpss_result)
```

```
        Augmented Dickey-Fuller Test

data:  ts_y
Dickey-Fuller = -2.4381, Lag order = 5, p-value = 0.3923
alternative hypothesis: stationary


######################
# KPSS Unit Root Test #
######################

Test is of type: tau with 4 lags.

Value of test-statistic is: 0.4105

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.119 0.146  0.176 0.216
```

Figure 25. Screenshot of step 4, Stationarity tests. (Source: Author)

```r
{r internal-diff-acf-pacf}
# Apply first-order differencing to fuel price and exogenous variable
diff_y <- diff(ts_y)
diff_x <- diff(ts_x)

# Plot differenced fuel price series to inspect stationarity
autoplot(diff_y) + ggtitle("Differenced Fuel Price") + theme_minimal()

# Conduct ADF test on differenced series to confirm stationarity
print(adf.test(diff_y))

# Plot ACF and PACF for differenced series to guide ARIMA model selection
par(mfrow = c(1, 2))
acf(diff_y, main = "ACF of Differenced Series")
pacf(diff_y, main = "PACF of Differenced Series")
par(mfrow = c(1, 1))
```

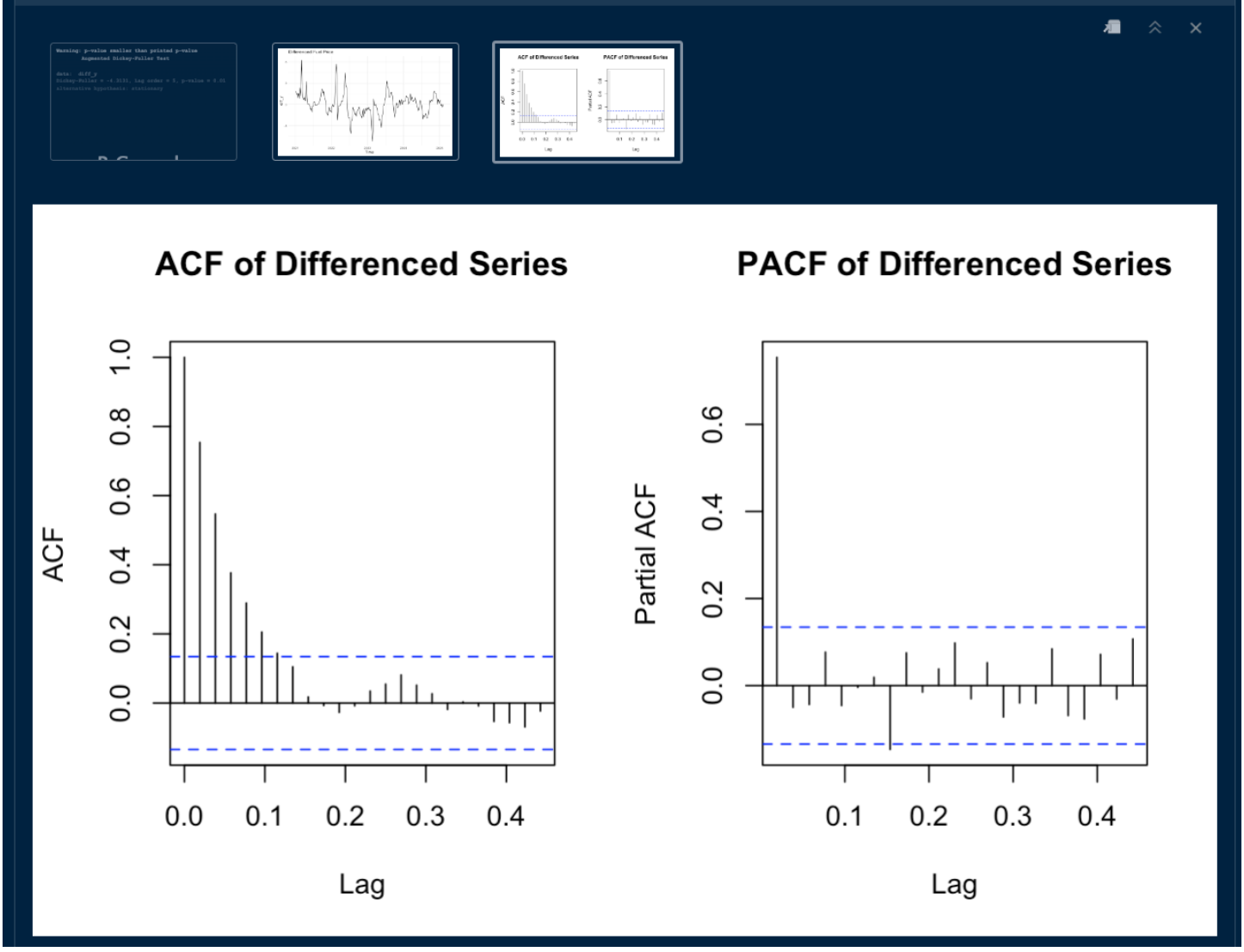


Figure 26. Screenshot of step 5, ACF and PACF plots. (Source: Author)

These steps ensured the data met the statistical requirements for multivariate time-series analysis.

Step 6: Granger Causality Test

Granger Causality confirmed that internal behaviour significantly influenced fuel prices (p = 0.0355).

```
{r granger-internal}
# Prepare differenced data for Granger causality test
granger_df_internal <- na.omit(cbind(diff_y, diff_x))
# Test if Quantity_Per_Transaction Granger-causes fuel price
grangertest(diff_y ~ diff_x, order = 2, data = as.data.frame(granger_df_internal))
```

```
Granger causality test

Model 1: diff_y ~ Lags(diff_y, 1:2) + Lags(diff_x, 1:2)
Model 2: diff_y ~ Lags(diff_y, 1:2)
  Res.Df Df      F Pr(>F)
1    206
2    208 -2 3.3929 0.0355 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 27. Screenshot of step 6, Granger Causality test output. (Source: Author)

Steps 7 & 8: Train-Test Split and Model Fitting

Two SARIMAX models were trained on the split dataset:

- Manual SARIMAX: Configured with order (1,1,1)(1,1,1)[52], based on ACF and PACF analysis from Segment 2, using the Arima() function.
- Auto SARIMAX: Fitted via auto.arima() with seasonal parameters and xreg terms to optimise model structure automatically.

```r
{r internal-train-test}
# Calculate index for 80/20 train-test split
n <- length(ts_y)
split_idx <- floor(0.8 * n)
# Create training and test sets for both dependent and exogenous variables
train_y <- window(ts_y, end = time(ts_y)[split_idx])
train_x <- window(ts_x, end = time(ts_x)[split_idx])
test_y  <- window(ts_y, start = time(ts_y)[split_idx + 1])
test_x  <- window(ts_x, start = time(ts_x)[split_idx + 1])
```

Step 8: Fit SARIMAX Models (Manual and Auto)

```r
{r internal-fit-models}
# Fit manual SARIMAX model with specified orders and exogenous variable
manual_sarimax <- Arima(train_y, order = c(1,1,1),
                        seasonal = list(order = c(1,1,1), period = 52),
                        xreg = train_x)
# Fit auto SARIMAX model with automatic order selection
auto_sarimax <- auto.arima(train_y, seasonal = TRUE, xreg = train_x)
# Display summaries for both models
summary(manual_sarimax)
summary(auto_sarimax)
```

```
Series: train_y
Regression with ARIMA(1,1,1)(1,1,1)[52] errors

Coefficients:
         ar1     ma1     sar1     sma1     xreg
      0.7234  0.1474  -0.4600  -0.6482  -0.0938
s.e.  0.0784  0.1126   0.2306   0.7876   0.0252

sigma^2 = 1.275:  log likelihood = -210.38
AIC=432.76   AICc=433.52   BIC=449.38

Training set error measures:
                     ME      RMSE       MAE         MPE      MAPE      MASE         ACF1
Training set -0.09313579 0.9177223 0.5329472 -0.0779833 0.5092862 0.0213775 -0.002463808
Series: train_y
Regression with ARIMA(1,1,1)(1,0,0)[52] errors

Coefficients:
         ar1     ma1     sar1     xreg
      0.7348  0.1598  -0.5263  -0.0687
s.e.  0.0628  0.0893   0.0662   0.0174

sigma^2 = 0.739:  log likelihood = -222.43
AIC=454.86   AICc=455.23   BIC=470.54

Training set error measures:
                     ME      RMSE       MAE          MPE      MAPE       MASE         ACF1
Training set 0.004459601 0.8470106 0.5522363 0.003495129 0.4941177 0.02215122 0.005137844
```

Figure 27. Screenshot of step 7 & 8, Model summaries and coefficients (manual and auto SARIMAX) (Source: Author)

These models incorporate the internal regressor to capture behavioural influences on fuel price dynamics.

Steps 9: Forecasting and Visualisation

Forecast plots showed both models closely tracked actual prices.

```r
{r internal-forecast-plot}
# Define forecast horizon equal to test set length
h <- length(test_y)

# Generate forecast for manual SARIMAX model
fc_manual <- forecast(manual_sarimax, xreg = test_x, h = h)

# Generate forecast for auto SARIMAX model
fc_auto <- forecast(auto_sarimax, xreg = test_x, h = h)

# Create time series for actual test values
ts_test_y <- ts(test_y, start = end(train_y) + c(0,1), frequency = 52)

# Plot manual SARIMAX forecast with actual values
p1 <- autoplot(fc_manual) +
  autolayer(ts_test_y, series = "Actual", color = "red") +
  ggtitle("Manual SARIMAX Forecast (Internal Variable)") +
  xlab("Time") + ylab("Fuel Price") +
  theme_minimal()

# Plot auto SARIMAX forecast with actual values
p2 <- autoplot(fc_auto) +
  autolayer(ts_test_y, series = "Actual", color = "red") +
  ggtitle("Auto SARIMAX Forecast (Internal Variable)") +
  xlab("Time") + ylab("Fuel Price") +
  theme_minimal()

# Display plots side by side for comparison
p1 / p2
```

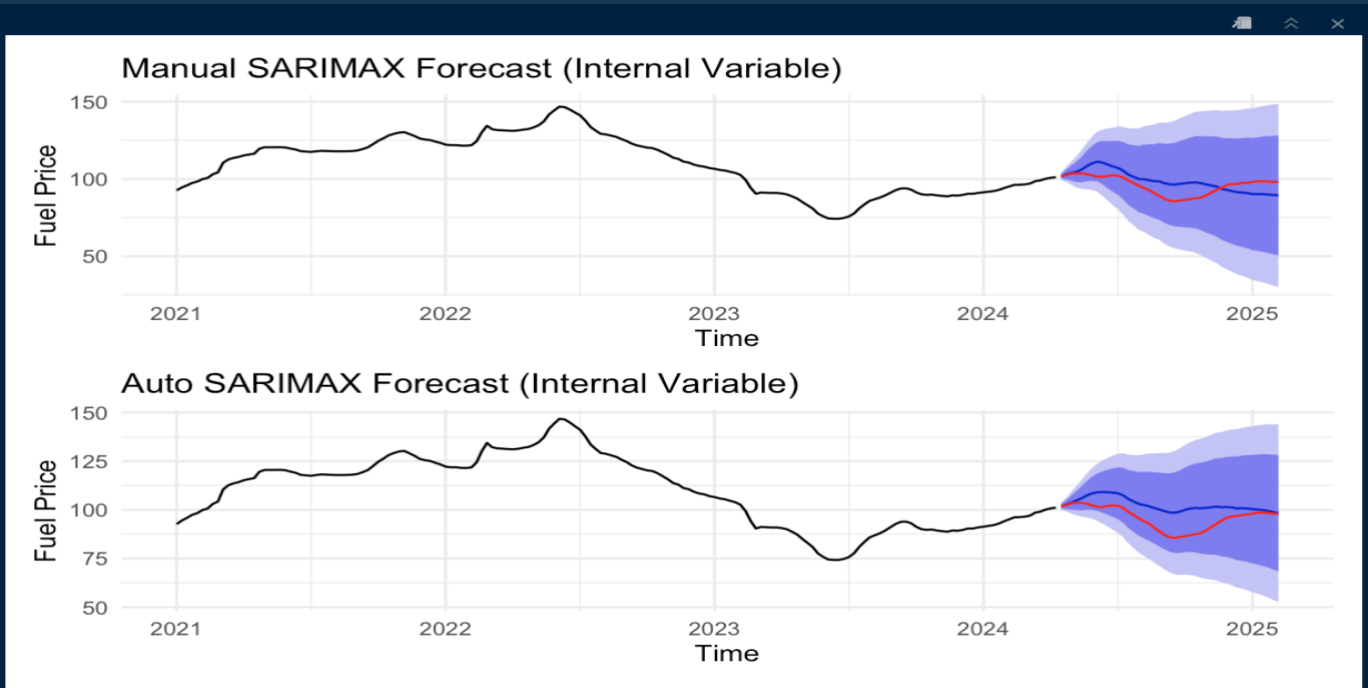


Figure 28. Screenshot of step 9, Forecast vs Actual plot (manual vs auto SARIMAX). (Source: Author)

Step 10: Forecast Accuracy

Forecast accuracy was assessed using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE). The manual SARIMAX model outperformed the auto SARIMAX, achieving a MAPE of 6.79% compared to 7.21% for the auto model.

```r
{r internal-accuracy}
# Convert forecasts to time series aligned with test set
manual_pred <- ts(fc_manual$mean, start = start(test_y), frequency = frequency(test_y))
auto_pred   <- ts(fc_auto$mean, start = start(test_y), frequency = frequency(test_y))

# Calculate accuracy metrics for manual and auto SARIMAX models
acc_manual <- forecast::accuracy(manual_pred, test_y)
acc_auto   <- forecast::accuracy(auto_pred, test_y)

# Display accuracy metrics side by side
rbind(
  Manual_SARIMAX = acc_manual["Test set", ],
  Auto_SARIMAX = acc_auto["Test set", ]
)
```

```
                    ME     RMSE      MAE       MPE     MAPE     ACF1 Theil's U
Manual_SARIMAX -2.821015 7.125019 6.379138 -3.160653 6.799421 0.942063  6.920710
Auto_SARIMAX   -6.663604 7.952208 6.663604 -7.211435 7.211435 0.927795  7.912671
```

Figure 29. Screenshot of step 9, Forecast accuracy metrics comparison. (Source: Author)

The superior performance of the manual model highlights the value of domain-informed configuration in multivariate forecasting.

Summary of Segment 4:

The internal variable helped improve SARIMAX performance. Manual tuning again yielded better results, reinforcing the benefits of guided configuration.

4.5 Multivariate Modelling with External Variable (Brent Crude)

Segment 5: External Multivariate Forecasting (SARIMAX with Brent Crude)

This segment tests Brent crude oil prices as an exogenous regressor, helping evaluate the power of external market indicators in fuel price forecasting.

Step 1: Load and Prepare Brent Crude Dataset

To explore the impact of external factors, Brent Crude oil prices were sourced from the U.S. Energy Information Administration (EIA) and pre-processed for integration into the forecasting pipeline.

```
{r load-brent, message=FALSE, warning=FALSE}
# Load Brent crude price dataset from Excel, skipping metadata rows
brent_raw <- read_excel("PET_PRI_SPT_S1_W-2.xlsx", sheet = "Data 1", skip = 2)

# Select and rename relevant columns for analysis
brent_data <- brent_raw %>%
  dplyr::select(Date = 1, Brent_Price = 3) %>%  # Column 3 is Europe Brent Spot Price
  mutate(
    Date = as.Date(Date),
    Brent_Price = as.numeric(Brent_Price)
  ) %>%
  drop_na() %>%
  arrange(Date)

# Check for missing values in Brent dataset
sum(is.na(brent_data))
```

```
[1] 0
```

```
{r handle-brent-NA-outliers}
# Interpolate any missing values in Brent_Price
brent_data$Brent_Price <- zoo::na.approx(brent_data$Brent_Price)

# Identify and handle outliers in Brent_Price using IQR method
Q1 <- quantile(brent_data$Brent_Price, 0.25)
Q3 <- quantile(brent_data$Brent_Price, 0.75)
IQR_val <- Q3 - Q1
lower_bound <- Q1 - 1.5 * IQR_val
upper_bound <- Q3 + 1.5 * IQR_val

# Replace outliers with NA and interpolate
brent_data$Brent_Price <- ifelse(
  brent_data$Brent_Price < lower_bound | brent_data$Brent_Price > upper_bound,
  NA,
  brent_data$Brent_Price
)
brent_data$Brent_Price <- zoo::na.approx(brent_data$Brent_Price)
```

Figure 30. Screenshot of step 1, Brent Crude Loading and Cleaning (Source: Author)

Step 2: Visualise Cleaned Brent Crude Time Series

```r
{r plot-brent-ts}
# Plot Brent crude price time series to inspect trends
ggplot(brent_data, aes(x = Date, y = Brent_Price)) +
  geom_line(color = "#2C3E50") +
  labs(title = "Weekly Europe Brent Crude Spot Price (FOB)",
       x = "Date", y = "Price (USD per Barrel)") +
  theme_minimal()
```

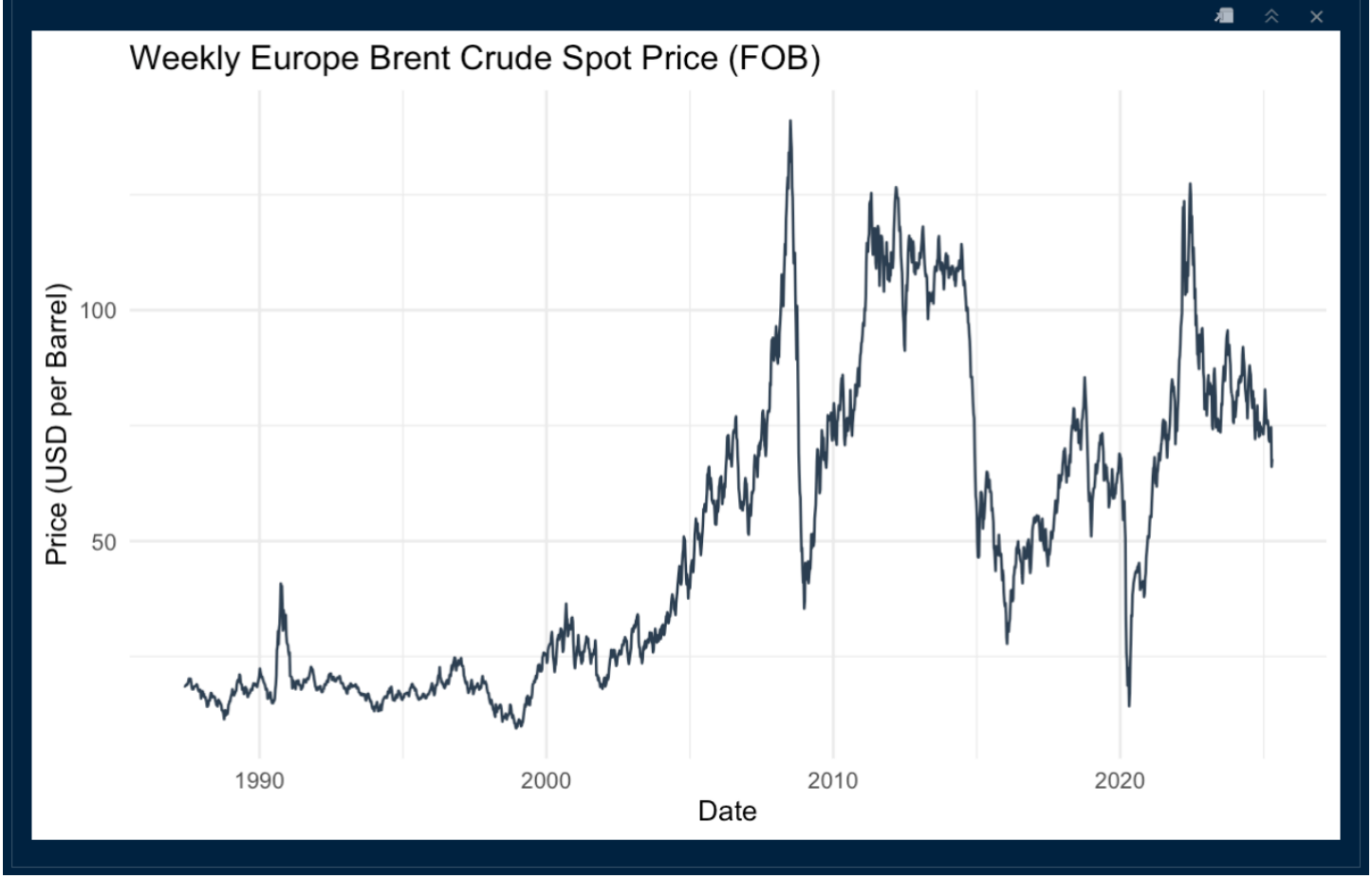


Figure 31. Screenshot of step 2, Weekly Brent Crude Price Line Chart (Source: Author)

Step 3: Merge Brent Crude with Fuel Dataset

Only complete cases were retained to ensure temporal consistency, resulting in a multivariate dataset comprising:

- avg_price: Weekly average fuel price (primary variable).
- brent_data: Exogenous regressor.

```r
{r merge-brent-fuel}
# Align fuel and Brent datasets by week
fuel_clean <- fuel_clean %>% mutate(Week = floor_date(Date, unit = "week"))
brent_data <- brent_data %>% mutate(Week = floor_date(Date, unit = "week"))

# Merge datasets on week, keeping only complete cases
external_merged <- fuel_clean %>%
  left_join(brent_data, by = "Week") %>%
  select(Date = Week, avg_price, Brent_Price) %>%
  drop_na()
head(external_merged)
```

A tibble: 6 × 3

Date <date>	avg_price <dbl>	Brent_Price <dbl>
2021-01...	92.60980	58.22
2021-02...	94.46735	61.13
2021-02...	95.85136	63.90
2021-02...	97.41086	65.86
2021-02...	98.31273	65.94
2021-03...	99.86002	68.15

6 rows

Figure 31. Screenshot of step 2, Merged Dataset. (Source: Author)

Step 4: Decomposition & Stationarity Tests

STL decomposition and stationarity tests (ADF and KPSS) confirmed the need for differencing. Both Brent and fuel price series was differenced for SARIMAX readiness.

```r
{r decomposition-stationarity-external}
# Convert fuel price to time series for analysis
ts_y <- ts(external_merged$avg_price, frequency = 52, start = c(2021, 1))

# Perform STL decomposition to analyse trend and seasonality
stl_decomp <- stl(ts_y, s.window = "periodic")
autoplot(stl_decomp) +
  ggtitle("STL Decomposition of Average Fuel Price (External Model)") +
  theme_minimal()

# Conduct ADF and KPSS tests to assess stationarity
adf_test <- adf.test(ts_y)
kpss_test <- ur.kpss(ts_y, type = "tau")
adf_test
summary(kpss_test)

# Apply first-order differencing to achieve stationarity
diff_y <- diff(ts_y)
autoplot(diff_y) + ggtitle("Differenced Fuel Price Series (External Model)") + theme_minimal()

# Plot ACF and PACF for differenced series to guide model selection
par(mfrow = c(1,2))
acf(diff_y, main = "ACF - Differenced Fuel Price")
pacf(diff_y, main = "PACF - Differenced Fuel Price")
par(mfrow = c(1,1))
```

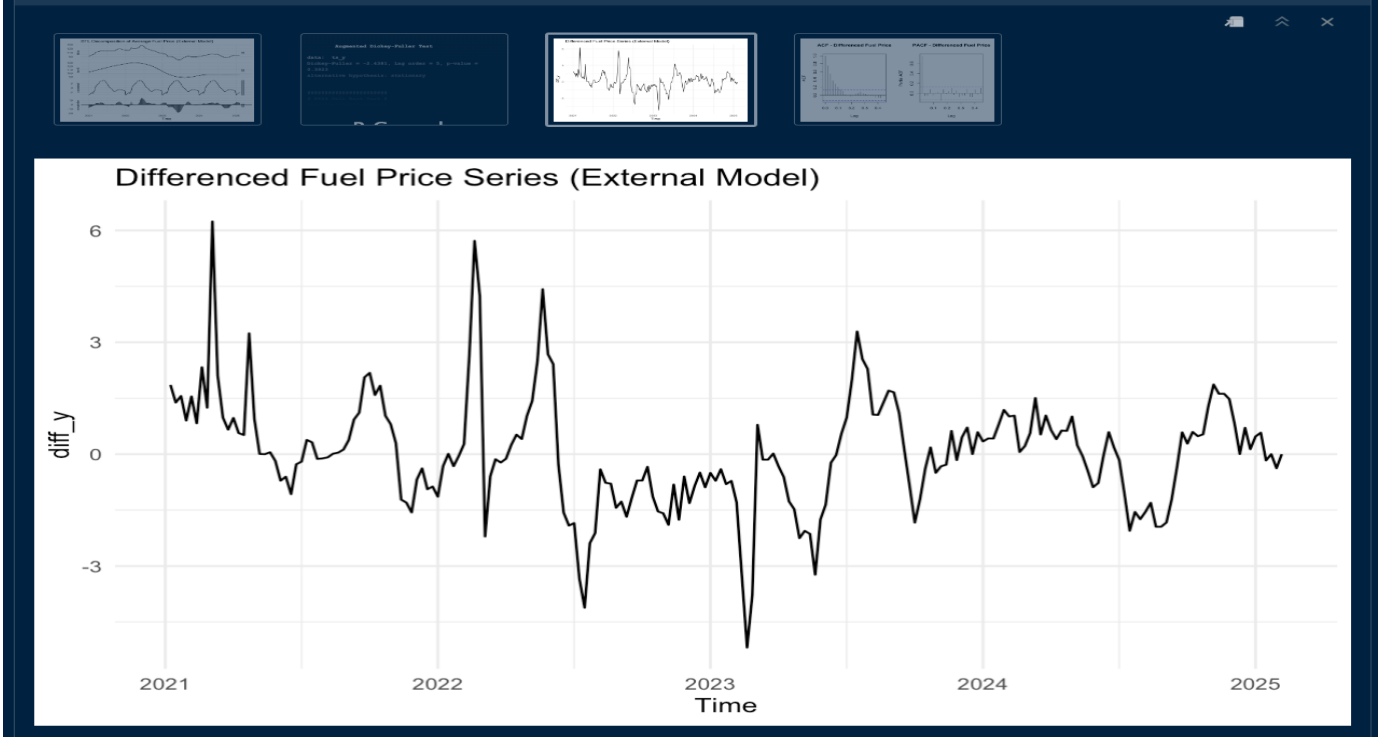


Figure 32. Screenshot Of Step 4, Differenced Fuel Prices Series With External Model (Source: Author)

Step 5: Granger Causality Test

Granger causality test (p = 0.04) validated Brent's predictive influence on UK fuel prices.

```r
{r granger-external}
# Convert Brent price to time series and difference it
ts_x <- ts(external_merged$Brent_Price, frequency = 52, start = c(2021, 1))
diff_x <- diff(ts_x)
# Prepare data for Granger causality test
granger_df_external <- na.omit(cbind(diff_y, diff_x))
# Test if Brent price Granger-causes fuel price
grangertest(diff_y ~ diff_x, order = 2, data = as.data.frame(granger_df_external))
```

```
Granger causality test

Model 1: diff_y ~ Lags(diff_y, 1:2) + Lags(diff_x, 1:2)
Model 2: diff_y ~ Lags(diff_y, 1:2)
  Res.Df Df      F  Pr(>F)
1    206
2    208 -2 3.2482 0.04084 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 33. Screenshot Of Step 5, Granger Causality Test Output (Source: Author)

Step 6: Train-Test Split & Model Fitting

Two SARIMAX models were fitted with Brent Crude prices as the exogenous regressor:

- Manual SARIMAX: Configured with order (1,1,1)(1,1,1)[52], based on prior ACF and PACF analysis.
- Auto SARIMAX: Automatically selected (2,1,2)(1,0,0)[52] using auto.arima() with xreg terms.

```r
{r fit-sarimax-brent-models}
# Calculate index for 80/20 train-test split
n <- length(ts_y)
split_idx <- floor(0.8 * n)

# Create training and test sets for dependent and exogenous variables
train_y <- window(ts_y, end = time(ts_y)[split_idx])
train_x <- window(ts_x, end = time(ts_x)[split_idx])
test_y <- window(ts_y, start = time(ts_y)[split_idx + 1])
test_x <- window(ts_x, start = time(ts_x)[split_idx + 1])

# Fit manual SARIMAX model with Brent price as exogenous variable
manual_sarimax_brent <- Arima(train_y,
                              order = c(1,1,1),
                              seasonal = list(order = c(1,1,1), period = 52),
                              xreg = train_x)

# Fit auto SARIMAX model with automatic order selection
auto_sarimax_brent <- auto.arima(train_y, seasonal = TRUE, xreg = train_x)

# Display model summaries
summary(manual_sarimax_brent)
summary(auto_sarimax_brent)
```

```
Series: train_y
Regression with ARIMA(1,1,1)(1,1,1)[52] errors

Coefficients:
         ar1     ma1     sar1     sma1    xreg
      0.7193  0.1302  -0.5891  -0.3031  0.0084
s.e.  0.0789  0.1092   0.1799   0.3402  0.0206

sigma^2 = 1.657:  log likelihood = -217.03
AIC=446.06   AICc=446.81   BIC=462.68

Training set error measures:
                     ME      RMSE       MAE          MPE      MAPE       MASE          ACF1
Training set -0.09392642 1.046344 0.5856668 -0.07686945 0.5644673 0.02349218 3.673761e-05
Series: train_y
Regression with ARIMA(2,1,2)(1,0,0)[52] errors

Coefficients:
         ar1     ar2    ma1     ma2     sar1    drift    xreg
      0.0835  0.3758  0.829  0.2940  -0.5450  -0.0291  0.0167
s.e.  0.1872  0.1389  0.195  0.1177   0.0652   0.1887  0.0107

sigma^2 = 0.7893:  log likelihood = -227.25
AIC=470.49   AICc=471.39   BIC=495.58

Training set error measures:
                     ME      RMSE       MAE          MPE      MAPE       MASE         ACF1
Training set 0.01273788 0.8674032 0.5623333 0.009144395 0.504403 0.02255623 -0.01307031
```

Figure 34. Screenshot Of Step 6, Train/Test Split & Model Fitting (Source: Author)

Step 7: Forecast and Visualise

Forecasts from both models were plotted and evaluated.

```r
{r forecast-brent-compare}
# Define forecast horizon equal to test set length
h <- length(test_y)
# Generate forecasts for manual and auto SARIMAX models
forecast_manual <- forecast(manual_sarimax_brent, xreg = test_x, h = h)
forecast_auto <- forecast(auto_sarimax_brent, xreg = test_x, h = h)

# Create time series for actual test values
ts_test_y <- ts(test_y, start = end(train_y) + c(0, 1), frequency = 52)

# Plot forecasts with actual values for comparison
autoplot(ts_test_y, series = "Actual") +
  autolayer(forecast_manual$mean, series = "Manual SARIMAX", color = "blue") +
  autolayer(forecast_auto$mean, series = "Auto SARIMAX", color = "green") +
  ggtitle("Forecast Comparison: Manual vs Auto SARIMAX (Brent)") +
  theme_minimal()
```

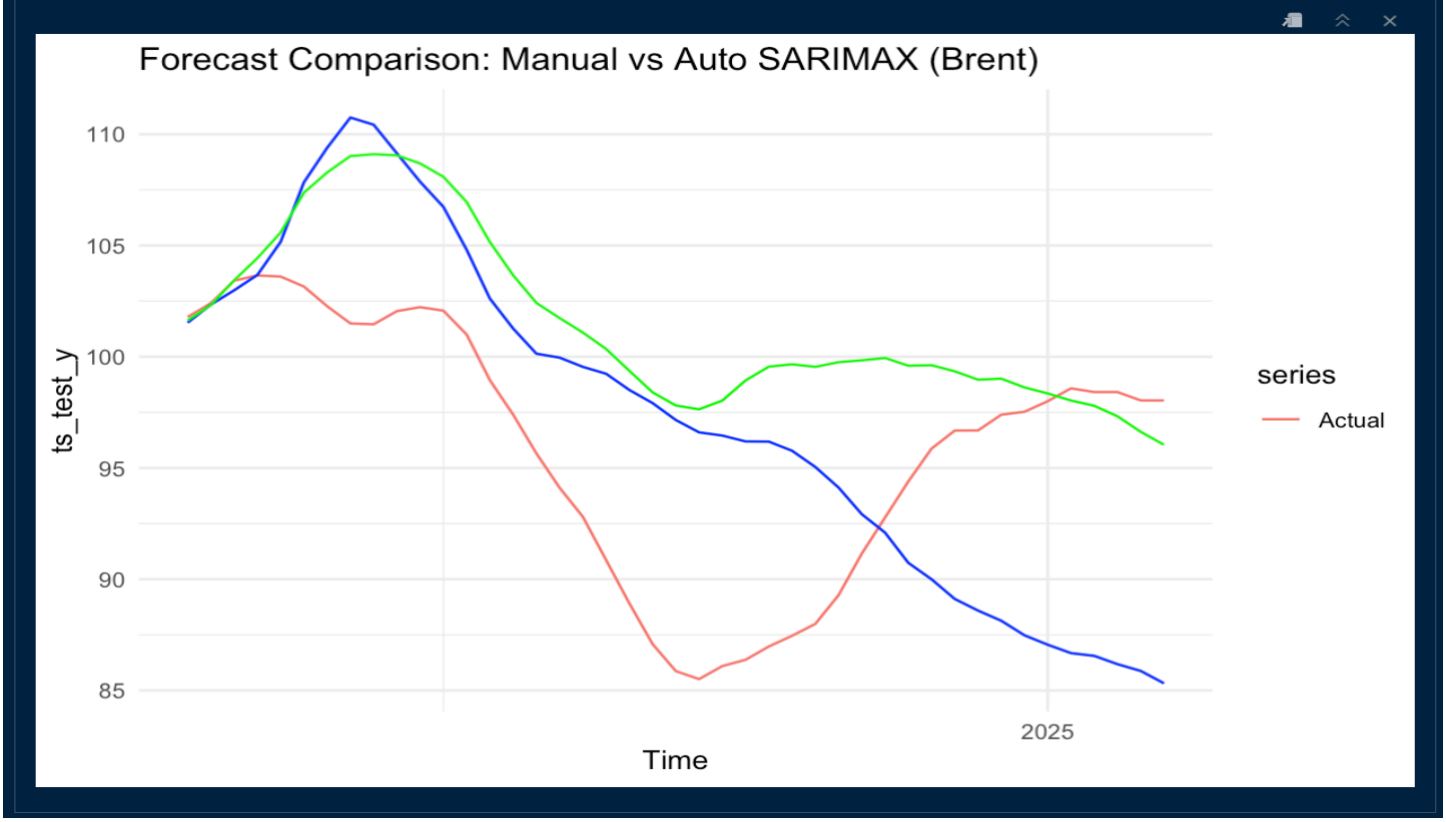


Figure 35. Screenshot Of Step 7, Overlay Plot of Manual vs Auto SARIMAX vs Actual (Source: Author)

The auto SARIMAX model demonstrated closer alignment with actual trend, suggesting improved predictive capability.

Step 8: Forecast Accuracy

Forecast accuracy was evaluated using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE).

```r
{r accuracy-brent-models}
# Convert forecasts to time series aligned with test set
manual_pred <- ts(forecast_manual$mean, start = start(ts_test_y), frequency = 52)
auto_pred <- ts(forecast_auto$mean, start = start(ts_test_y), frequency = 52)
actual <- ts(as.numeric(test_y), start = start(ts_test_y), frequency = 52)

# Calculate accuracy metrics for manual and auto SARIMAX models
manual_acc <- forecast::accuracy(manual_pred, actual)
auto_acc <- forecast::accuracy(auto_pred, actual)

# Display accuracy metrics side by side
rbind(
  Manual_SARIMAX = manual_acc["Test set", ],
  Auto_SARIMAX = auto_acc["Test set", ]
)
```

```
                    ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
Manual_SARIMAX -1.446930 7.879720 6.922174 -1.732459 7.339102 0.9442232  7.546607
Auto_SARIMAX   -5.634249 7.225885 5.908476 -6.121877 6.400711 0.9273693  7.186261
```

Figure 36. Screenshot Of Step 8, Forecast Accuracy (Source: Author)

Model	RMSE	MAE	MAPE
Manual SARIMAX	7.87	6.92	7.33%
Auto SARIMAX	7.22	5.90	6.40%

Table 3. Forecast Accuracy Metrics Table Output. (Source: Author)

The auto SARIMAX model outperformed the manual specification across all metrics, highlighting the effectiveness of automated model selection for this exogenous regressor.

Summary of Segment 5:

Brent prices provided more predictive power than internal regressors. Auto SARIMAX with Brent performed best among the statistical models, delivering the lowest MAPE of 6.40%.

Segment 6: Forecasting & Comparison

This segment compares all three ARIMA-based models to assess internal vs external influences on forecasting performance.

Step 1: Comparing SARIMA vs SARIMAX (Internal) vs SARIMAX (Brent)

A comprehensive evaluation was conducted to compare the forecasting performance of three time-series models:

- Univariate SARIMA: Relies solely on historical fuel price data.
- SARIMAX (Internal): Incorporates Quantity_Per_Transaction as an internal behavioural regressor.
- SARIMAX (Brent): Uses Brent Crude oil price as an external market regressor.

```r
{r model-compare-table, results='asis'}
# Reconstruct aligned forecast time series for each model
ts_sarima <- ts(manual_fc$mean, start = start(test_ts), frequency = frequency(test_ts)) # Manual Arima
ts_internal <- ts(fc_manual$mean, start = start(test_y), frequency = frequency(test_y)) # Manual Arima
ts_brent <- ts(forecast_auto$mean, start = start(test_y), frequency = frequency(test_y)) # Auto Arima

# Reconstruct aligned actual time series
ts_actual_sarima <- test_ts
ts_actual_internal <- test_y
ts_actual_brent <- test_y  # Same as internal

# Compute accuracy metrics for each model
sarima_metrics <- data.frame(
  RMSE = round(rmse(ts_actual_sarima, ts_sarima), 2),
  MAE = round(mae(ts_actual_sarima, ts_sarima), 2),
  MAPE = round(mape(ts_actual_sarima, ts_sarima) * 100, 2)
)

internal_metrics <- data.frame(
  RMSE = round(rmse(ts_actual_internal, ts_internal), 2),
  MAE = round(mae(ts_actual_internal, ts_internal), 2),
  MAPE = round(mape(ts_actual_internal, ts_internal) * 100, 2)
)

brent_metrics <- data.frame(
  RMSE = round(rmse(ts_actual_brent, ts_brent), 2),
  MAE = round(mae(ts_actual_brent, ts_brent), 2),
  MAPE = round(mape(ts_actual_brent, ts_brent) * 100, 2)
)

# Combine into a single table
model_perf <- rbind(
  "Univariate SARIMA" = sarima_metrics,
  "SARIMAX (Internal)" = internal_metrics,
  "SARIMAX (Brent)" = brent_metrics
)

# Display the performance comparison table
print(model_perf)
```

Description: df [3 × 3]

	R... <dbl>	M... <dbl>	M... <dbl>
Univariate SARIMA	7...	6...	7...
SARIMAX (Internal)	7...	6...	6...
SARIMAX (Brent)	7...	5...	6...

3 rows

Figure 37. Screenshot Of Step 1, Code for Forecast Accuracy for ARIMA-Based Models (Source: Author)

Performance was assessed using three standard metrics:

- Root Mean Squared Error (RMSE): Measures sensitivity to large prediction errors.
- Mean Absolute Error (MAE): Captures the average magnitude of errors.
- Mean Absolute Percentage Error (MAPE): Provides a scale-independent measure for comparison.

Model	RMSE	MAE	MAPE
Univariate SARIMA	7.89	6.93	7.35%
SARIMAX (Internal)	7.13	6.38	6.80%
SARIMAX (Brent)	7.23	5.91	6.40%

Table 4. Comparative Forecast Accuracy for ARIMA-Based Models. (Source: Author)

Step 2: Visualise Model Comparison

A histogram of MAPE values reinforced the performance ranking, with SARIMAX (Brent) clearly superior.

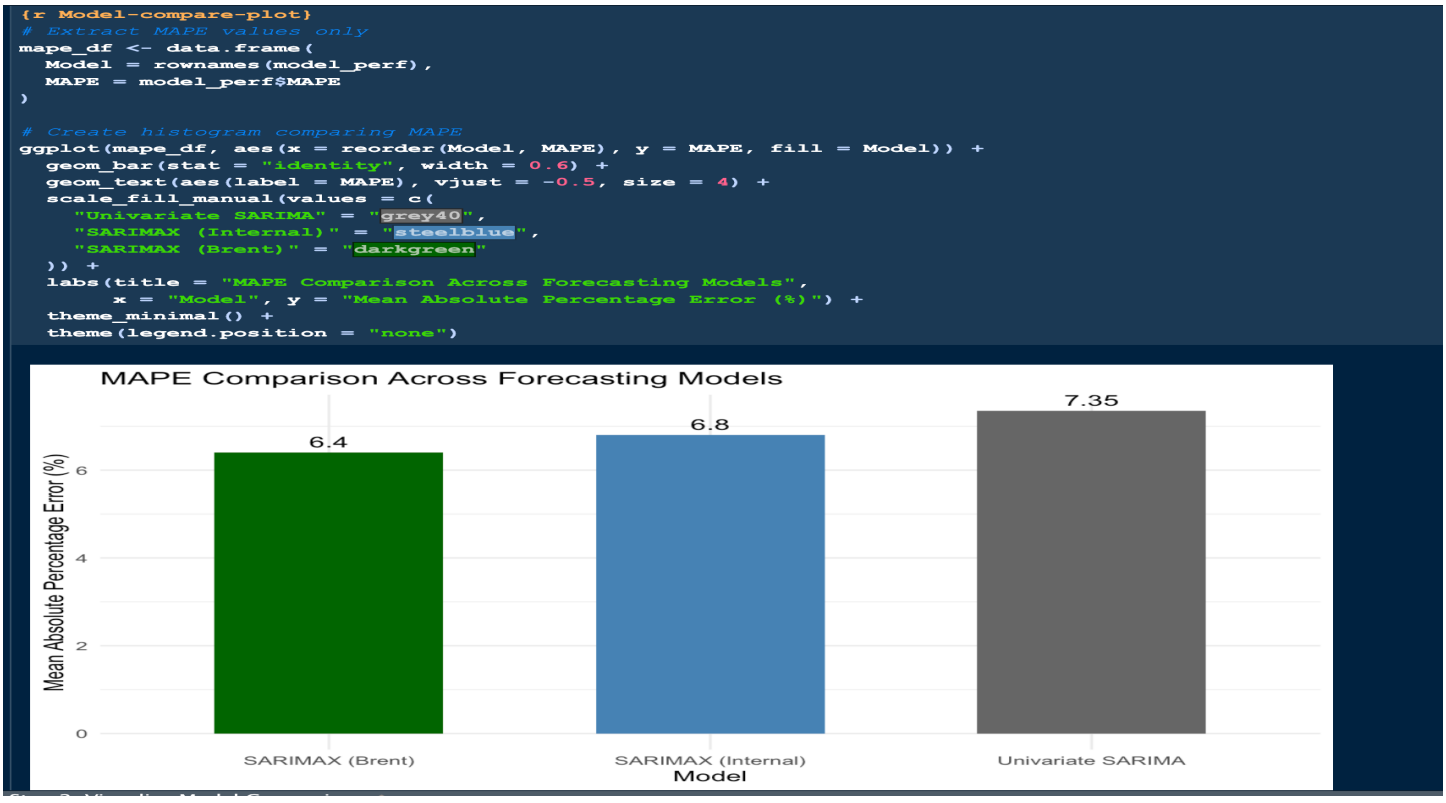


Figure 38. Screenshot Of Step 2, MAPE Comparison Across Forecasting Models (Source: Author)

Summary of Segment 6:

External variables such as Brent prices yielded better accuracy than internal signals, supporting the importance of macroeconomic indicators in fuel price prediction.

Segment 7: Future Forecasting (15 Weeks Ahead)

This segment deploys the top-performing statistical model; auto SARIMAX with Brent for a 15-week forecast.

Step 1: Refit Best Model on Full Dataset

The model was refit on the full dataset (Jan 2021 – Feb 2025). Since future Brent prices are unknown, the last known value was used for all 15 weeks.

```
{r refit-models-full}
# Create full fuel price time series
ts_full <- ts(fuel_clean$avg_price, frequency = 52, start = c(2021, 1))

# Extract full Brent Crude series as exogenous regressor
xreg_external_full <- external_merged$Brent_Price

# Fit SARIMAX (Brent) using auto.arima on full dataset
model_sarimax_best <- auto.arima(ts_full, xreg = xreg_external_full, seasonal = TRUE)
```

Figure 39. Screenshot Of Step 1, Retrain and Refit Best Model on Full Dataset (Source: Author)

Step 2: Forecast 15 Weeks Ahead

```
{r forecast-15weeks}
# Set 15-week forecast horizon
h_future <- 15

# Replicate last Brent price for next 15 weeks (assumption: constant exogenous input)
future_xreg <- rep(tail(xreg_external_full, 1), h_future)

# Generate forecast
forecast_best <- forecast(model_sarimax_best, xreg = future_xreg, h = h_future)
```

Figure 40. Screenshot Of Step 2, Code For 15 Weeks Forecast (Source: Author)

Step 3: Plot Future Forecasts

Historical prices and the 15-week forecast were plotted together

```r
{r plot-15week-forecast}
# Plot historical data with future forecast
autoplot(ts_full, series = "Historical") +
  autolayer(forecast_best$mean, series = "15-Week Forecast", color = "darkgreen") +
  ggtitle("15-Week Ahead Forecast – SARIMAX (Brent)") +
  xlab("Time (Weeks)") +
  ylab("Fuel Price (Pence per Litre)") +
  scale_colour_manual(name = "Series", values = c("Historical" = "grey70", "15-Week Forecast" =
"darkgreen")) +
  theme_minimal()
```

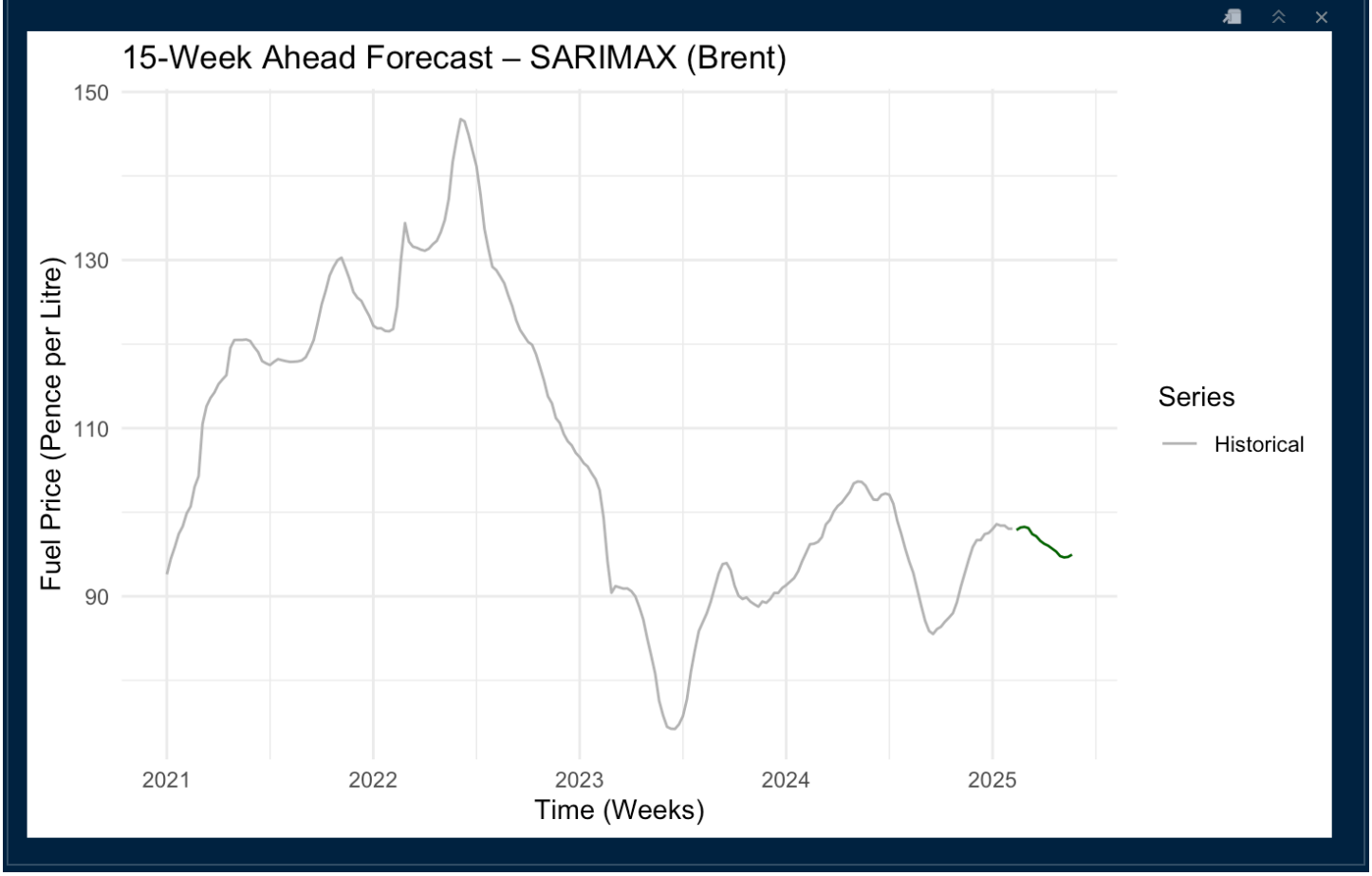


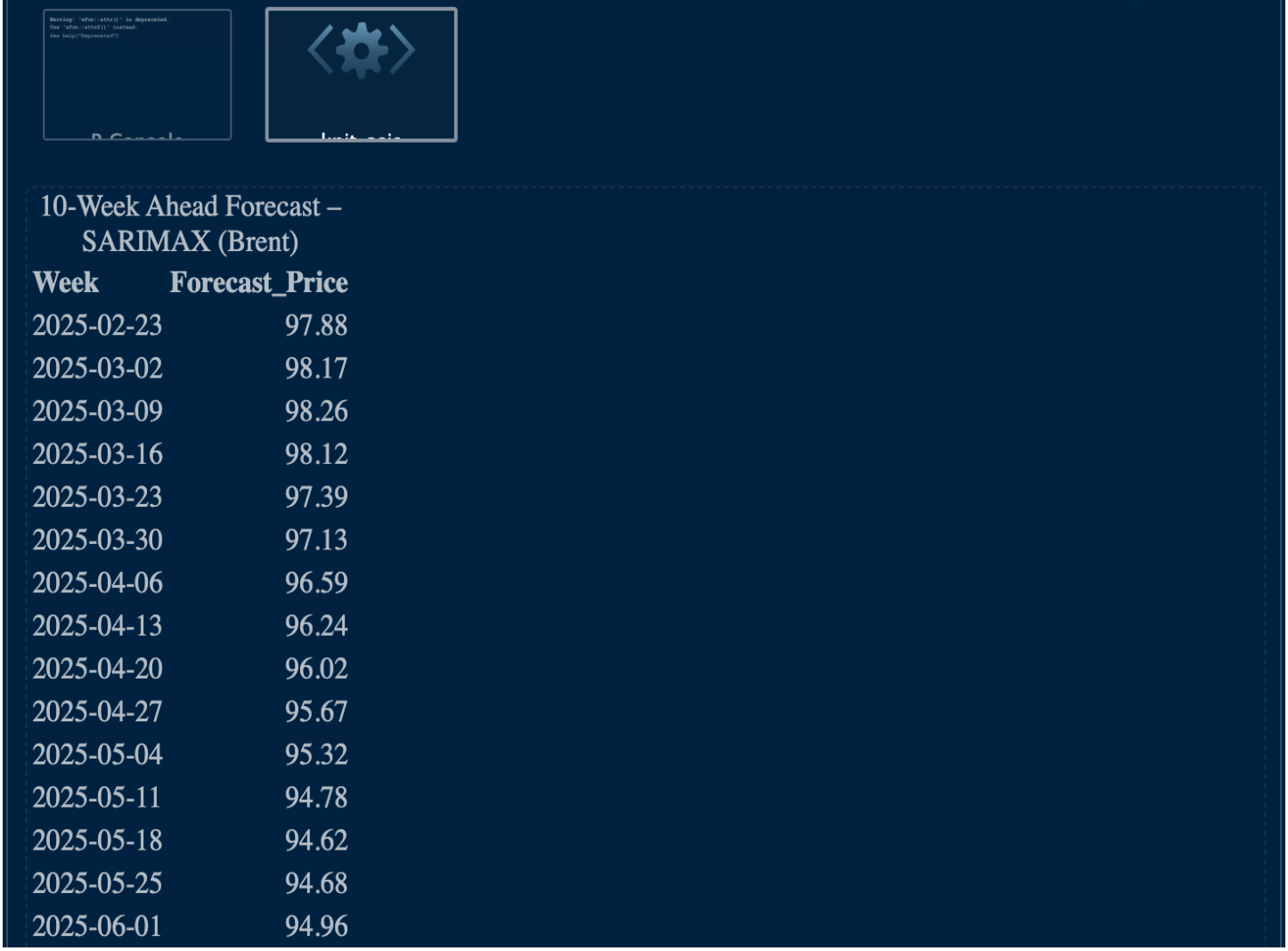
Figure 40. Screenshot Of Step 3, 15-Week Ahead Forecast – SARIMAX (Brent) Plot (Source: Author)

Step 4: Preview Forecasted Values

Forecasted values showed a gentle decline from ~98p to ~95p/litre, assuming stable external market conditions.

```r
{r show-forecast-table}
# Create table of future dates and predicted prices
future_forecast_df <- data.frame(
  Week = seq(max(fuel_clean$Date) + 7, by = 7, length.out = h_future),
  Forecast_Price = round(forecast_best$mean, 2)
)

# Display table
knitr::kable(future_forecast_df, caption = "10-Week Ahead Forecast - SARIMAX (Brent)")
```



10-Week Ahead Forecast – SARIMAX (Brent)

Week	Forecast_Price
2025-02-23	97.88
2025-03-02	98.17
2025-03-09	98.26
2025-03-16	98.12
2025-03-23	97.39
2025-03-30	97.13
2025-04-06	96.59
2025-04-13	96.24
2025-04-20	96.02
2025-04-27	95.67
2025-05-04	95.32
2025-05-11	94.78
2025-05-18	94.62
2025-05-25	94.68
2025-06-01	94.96

Figure 41. Screenshot Of Step 4, 15-Week Ahead Forecast Prices– SARIMAX (Brent) (Source: Author)

Segment 7 Conclusion:

The segment delivers a realistic, short-term fuel price projection using the most accurate statistical model, useful for business and policy planning.

4.6 Machine Learning Models

Segment 8: Random Forest Forecasting

This segment explores the Random Forest (RF) model using lagged features (previous 3 weeks of avg_price) to assess predictive accuracy. After splitting the data (80:20), RF models with ntree = 100, 400 and 700 were tested. The best model (ntree = 700) achieved a MAPE of 1.32%, outperforming all statistical models. The model captured nonlinear patterns effectively and the prediction plot demonstrated strong alignment with actual prices.

Step 1: Data Preparation and Lag Feature Engineering

```r
{r prepare-data-rf}
# Create lagged features for Random Forest model
fuel_rf <- fuel_clean %>%
  mutate(
    lag_1 = lag(avg_price, 1),
    lag_2 = lag(avg_price, 2),
    lag_3 = lag(avg_price, 3)
  ) %>%
  drop_na()

# Perform 80/20 train-test split
split_idx <- floor(0.8 * nrow(fuel_rf))
train_rf <- fuel_rf[1:split_idx, ]
test_rf  <- fuel_rf[(split_idx + 1):nrow(fuel_rf), ]

# Define predictors (lagged features) and target variable
x_train <- train_rf[, c("lag_1", "lag_2", "lag_3")]
y_train <- train_rf$avg_price

x_test <- test_rf[, c("lag_1", "lag_2", "lag_3")]
y_test <- test_rf$avg_price
```

Figure 42. Screenshot Of Step 1, Lag Feature Creation and Train-Test Split (Source: Author)

Step 2: Model Fitting and Evaluation

```r
{r fit-rf-model}
# Initialize dataframe to store results
rf_results <- data.frame()

# Fit Random Forest models with varying number of trees
for (trees in c(100, 400, 700)) {
  set.seed(42)  # Ensure reproducibility
  rf_model <- randomForest(x = x_train, y = y_train, ntree = trees)
  preds <- predict(rf_model, x_test)

  # Calculate accuracy metrics
  rmse_val <- sqrt(mean((preds - y_test)^2))
  mae_val  <- mean(abs(preds - y_test))
  mape_val <- mean(abs((preds - y_test) / y_test)) * 100

  # Store results
  rf_results <- rbind(rf_results, data.frame(
    Model = paste("Random Forest (ntree =", trees, ")"),
    RMSE = round(rmse_val, 4),
    MAE  = round(mae_val, 4),
    MAPE = round(mape_val, 4)
  ))
}

# Display formatted table of Random Forest accuracy metrics
knitr::kable(rf_results, caption = "Random Forest Forecast Accuracy for Varying ntree") %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "hover"), full_width = FALSE)
```

Random Forest Forecast Accuracy for Varying ntree

Model	RMSE	MAE	MAPE
Random Forest (ntree = 100)	1.5954	1.2685	1.3422
Random Forest (ntree = 400)	1.5868	1.2879	1.3596
Random Forest (ntree = 700)	1.5546	1.2521	1.3210

Figure 43. Screenshot Of Step 2, Random Forest Forecast Accuracy for Varying ntree (Source: Author)

The low MAPE for the ntree = 700 model highlights its ability to deliver precise predictions, making it a strong contender among machine learning approaches.

Step 3: Visualisation of Results

```r
{r predict-rf}
# Fit Random Forest model with best ntree (700)
set.seed(42)
rf_best <- randomForest(x = x_train, y = y_train, ntree = 700)
rf_pred_best <- predict(rf_best, x_test)

# Prepare data for plotting
plot_df <- data.frame(
  Week = 1:length(y_test),
  Actual = y_test,
  Predicted = rf_pred_best
)

# Plot actual vs predicted values for best Random Forest model
ggplot(plot_df, aes(x = Week)) +
  geom_line(aes(y = Actual, colour = "Actual")) +
  geom_line(aes(y = Predicted, colour = "Predicted"), linetype = "dashed") +
  scale_colour_manual(values = c("Actual" = "red", "Predicted" = "blue")) +
  labs(title = "Random Forest (ntree = 700): Actual vs Predicted",
       y = "Fuel Price (Pence per Litre)", x = "Week") +
  theme_minimal()
```

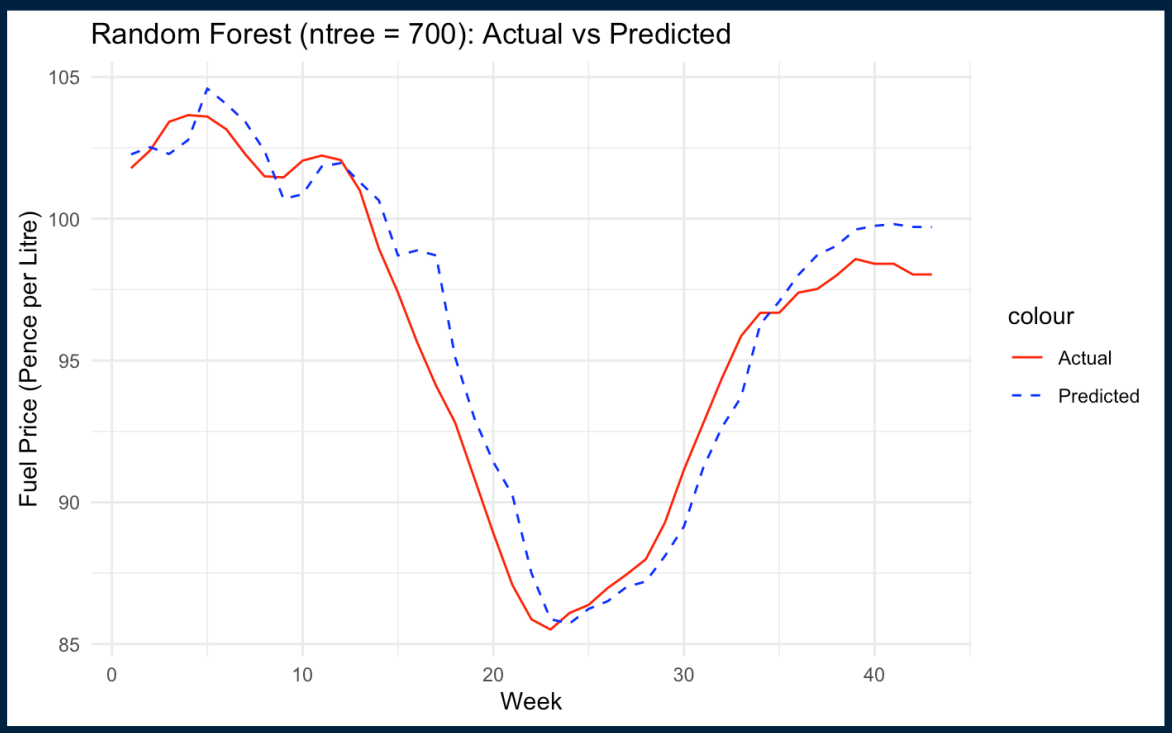


Figure 44. Screenshot Of Step 3, RF (ntree = 700) Actual vs Predicted Weekly Fuel Prices (Source: Author)

This visualisation provides an natural understanding of the model's predictive accuracy, reinforcing its reliability for short-term forecasting.

Segment 9: LSTM Forecasting

Step 1: Prepare Data

The LSTM model was developed using 3 lagged steps on a normalised price series, reshaped for sequence learning. After training a basic 50-unit LSTM for 50 epochs, predictions were de-normalised.

```
{r prepare-lstm-data}
# Select relevant columns for LSTM model
fuel_lstm <- fuel_data %>% select(Date, avg_price)
# Normalise avg_price to [0,1] for better LSTM training
min_price <- min(fuel_lstm$avg_price)
max_price <- max(fuel_lstm$avg_price)
fuel_lstm$norm_price <- (fuel_lstm$avg_price - min_price) / (max_price - min_price)

# Function to create supervised learning dataset with lagged features
create_supervised <- function(series, lag = 3) {
  x <- NULL; y <- NULL
  for (i in 1:(length(series) - lag)) {
    x <- rbind(x, series[i:(i + lag - 1)])
    y <- c(y, series[i + lag])
  }
  list(x = x, y = y)
}

# Create supervised dataset with 3 lags
supervised <- create_supervised(fuel_lstm$norm_price)
# Perform 80/20 train-test split
split_index <- floor(0.8 * nrow(supervised$x))
train_x <- supervised$x[1:split_index, ]
train_y <- supervised$y[1:split_index]
test_x <- supervised$x[(split_index + 1):nrow(supervised$x), ]
test_y <- supervised$y[(split_index + 1):length(supervised$y)]

# Reshape data for LSTM input (samples, timesteps, features)
train_x <- array(train_x, dim = c(nrow(train_x), ncol(train_x), 1))
test_x <- array(test_x, dim = c(nrow(test_x), ncol(test_x), 1))
```

Figure 45. Screenshot Of Step 1, Preparing The Data For Forecasting With LSTM (Source: Author)

Step 2: Fit LSTM

```
{r fit-lstm}
# Define sequential LSTM model with one LSTM layer and one dense layer
model_lstm <- keras_model_sequential() %>%
  layer_lstm(units = 50, input_shape = c(dim(train_x)[2], 1)) %>%
  layer_dense(units = 1)

# Compile model with mean squared error loss and Adam optimizer
model_lstm %>% compile(loss = "mse", optimizer = "adam")
# Train model with validation split
history <- model_lstm %>% fit(
  x = train_x, y = train_y, epochs = 50, batch_size = 8,
  validation_split = 0.1, verbose = 0
)
```

Run Current Chunk

Figure 46. Screenshot Of Step 2, Fitting the LSTM Model (Source: Author)

Step 3: Predict

After training, predictions were generated on the test set. Both the actual and predicted values were de-normalised to their original price scale using the min – max transformation parameters. This allowed for direct comparison with other models and true values.

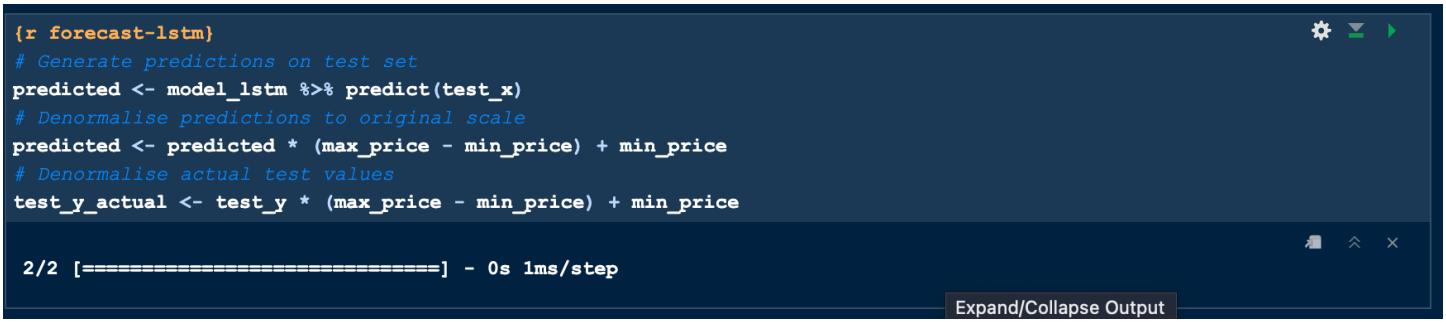


Figure 47. Screenshot Of Step 3, Forecast with the LSTM Model (Source: Author)

Step 4: Accuracy

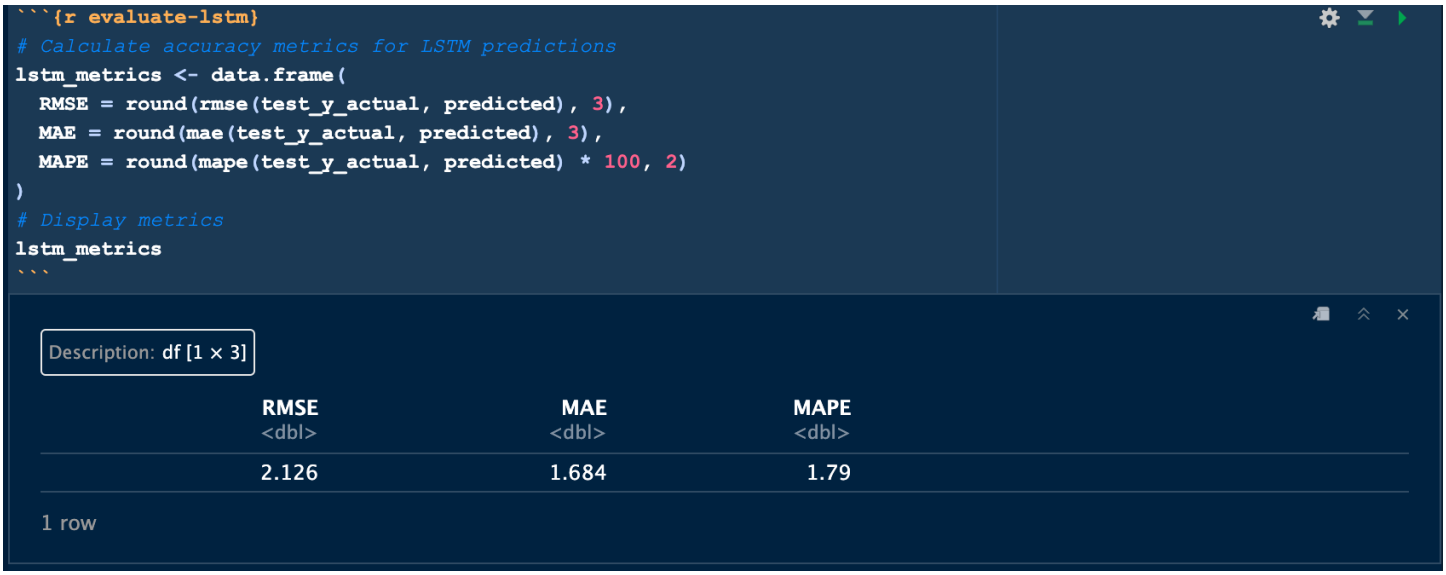


Figure 48. Screenshot Of Step 4, Accuracy metrics for the LSTM model (Source: Author)

While the LSTM captured broader trends, it lagged in reacting to sharp shifts. Its MAPE of 1.79% was reasonable but didn't outperform RF.

Step 5: Plot

```r
```{r plot-lstm-vs-actual}
# Prepare data for plotting
plot_df <- data.frame(Week = 1:length(test_y_actual), Actual = test_y_actual, Predicted = predicted)

# Plot actual vs predicted values for LSTM model
ggplot(plot_df, aes(x = Week)) +
  geom_line(aes(y = Actual, colour = "Actual")) +
  geom_line(aes(y = Predicted, colour = "Predicted")) +
  scale_colour_manual(values = c("Actual" = "red", "Predicted" = "blue")) +
  labs(title = "LSTM Forecast vs Actual", y = "Fuel Price", x = "Week") +
  theme_minimal()
```
```

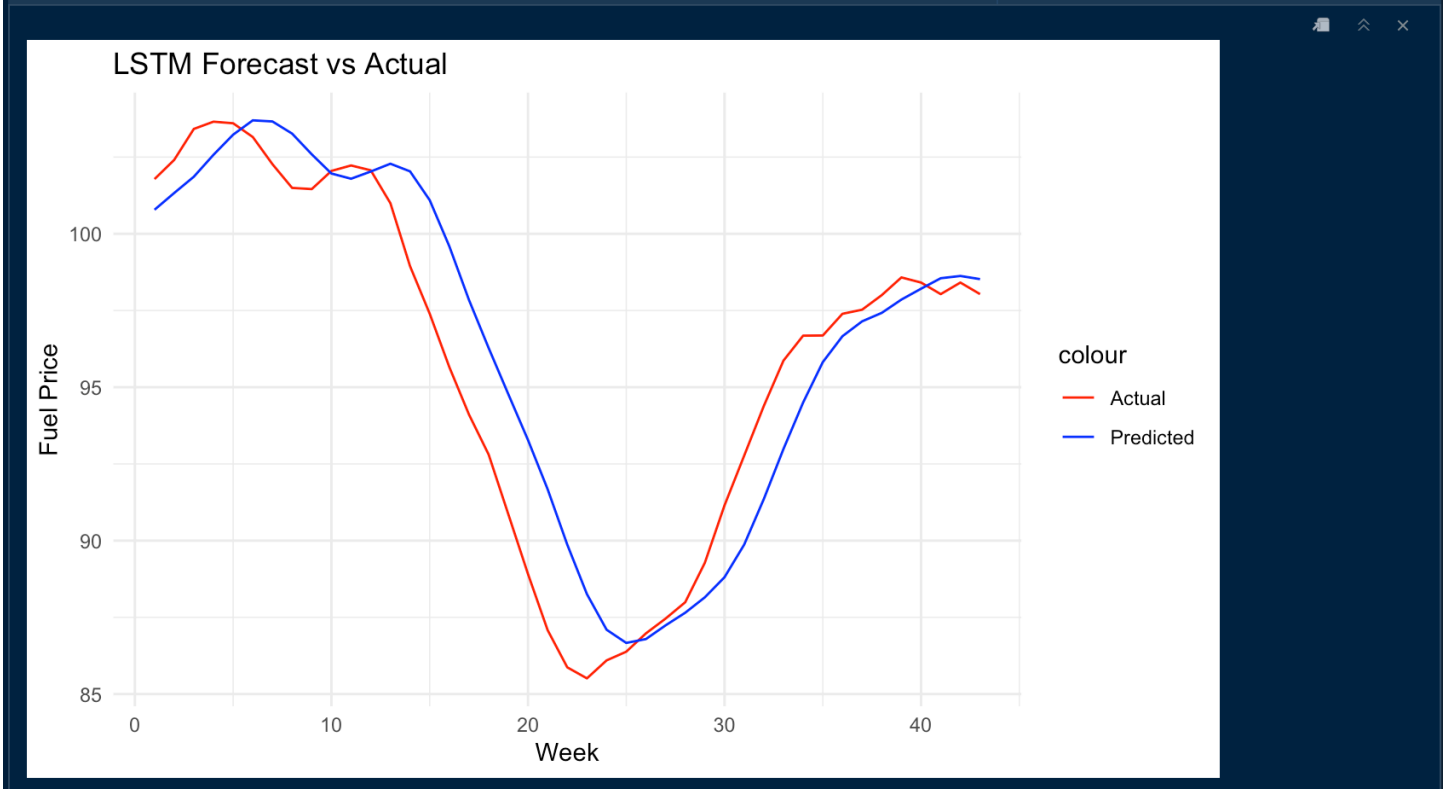


Figure 49. Screenshot Of Step 5, LSTM Forecast vs Actual Plot (Source: Author)

Segment 10. Support Vector Regression (SVR) Forecasting

SVR was implemented with the same 3-lag input and tested across three kernels. The linear kernel emerged as the best performer, delivering a MAPE of just 0.64% — the most accurate model in the study.

Step 1: Data Preparation

```r
```{r svr-data-prep}
# Create lagged features for SVR model
fuel_data <- fuel_data %>%
  mutate(
    lag_1 = lag(avg_price, 1),
    lag_2 = lag(avg_price, 2),
    lag_3 = lag(avg_price, 3)
  ) %>% drop_na()
# Select relevant columns
data_svr <- fuel_data %>% drop_na() %>% select(avg_price, lag_1, lag_2, lag_3)
# Perform 80/20 train-test split
split_index <- floor(0.8 * nrow(data_svr))
train_data <- data_svr[1:split_index, ]
test_data <- data_svr[(split_index + 1):nrow(data_svr), ]
# Center and scale training and test data
preproc <- preProcess(train_data, method = c("center", "scale"))
train_scaled <- predict(preproc, train_data)
test_scaled <- predict(preproc, test_data)
# Define predictors and target
X_train <- train_scaled %>% select(-avg_price)
y_train <- train_scaled$avg_price
X_test <- test_scaled %>% select(-avg_price)
y_test <- test_scaled$avg_price
```
```

Figure 50. Screenshot Of Step 1, Lag Feature Creation and Train-Test Split (Source: Author)

This preprocessing ensured the dataset was optimised for SVR model training and evaluation.

Step 2: SVR Training and Evaluation for Multiple Kernels

```r
```{r svr-train}
# Define kernel types to test
kernels <- c("linear", "polynomial", "radial")
svr_results <- data.frame()

# Train and evaluate SVR models for each kernel
for (k in kernels) {
  # Train SVR model
  svr_model <- svm(x = X_train, y = y_train, kernel = k)
  svr_preds_scaled <- predict(svr_model, X_test)

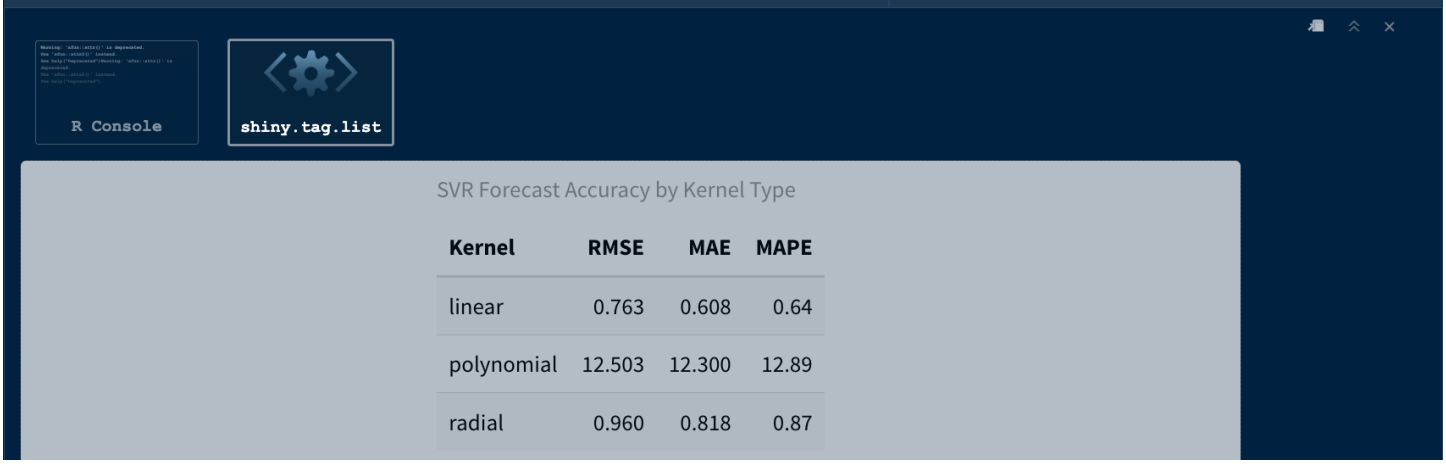
  # Inverse scale predictions and actual values
  avg_mean <- preproc$mean["avg_price"]
  avg_sd   <- preproc$std["avg_price"]

  final_preds <- svr_preds_scaled * avg_sd + avg_mean
  actual_vals <- y_test * avg_sd + avg_mean

  # Calculate accuracy metrics
  rmse_val <- RMSE(final_preds, actual_vals)
  mae_val  <- MAE(final_preds, actual_vals)
  mape_val <- mean(abs((final_preds - actual_vals) / actual_vals)) * 100

  # Store results
  svr_results <- rbind(svr_results, data.frame(
    Kernel = k,
    RMSE = round(rmse_val, 3),
    MAE  = round(mae_val, 3),
    MAPE = round(mape_val, 2)
  ))
}

# Display formatted table of SVR accuracy metrics
knitr::kable(svr_results, caption = "SVR Forecast Accuracy by Kernel Type") %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "hover"), full_width = FALSE)
```
```



SVR Forecast Accuracy by Kernel Type

Kernel	RMSE	MAE	MAPE
linear	0.763	0.608	0.64
polynomial	12.503	12.300	12.89
radial	0.960	0.818	0.87

Figure 51. Screenshot Of Step 2, SVR Forecast Accuracy for Different Kernels (Source: Author)

The linear kernel significantly outperformed the polynomial and RBF kernels, achieving a MAPE of 0.64%.

Step 3: Predict and Invert Scaling

The best-performing kernel (linear) was retrained on the training set and used to generate predictions for the test set.

```r
# Refit best model (linear kernel)
svr_model <- svm(x = X_train, y = y_train, kernel = "linear")
svr_preds_scaled <- predict(svr_model, X_test)

# Inverse scaling
avg_mean <- preproc$mean["avg_price"]
avg_sd   <- preproc$std["avg_price"]
final_preds <- svr_preds_scaled * avg_sd + avg_mean
actual_vals <- y_test * avg_sd + avg_mean

# Generate predictions for test set using SVR model
svr_preds_scaled <- predict(svr_model, X_test)

# Extract scaling parameters
avg_mean <- preproc$mean["avg_price"]
avg_sd   <- preproc$std["avg_price"]

# Inverse scale predictions and actual values
final_preds <- svr_preds_scaled * avg_sd + avg_mean
actual_vals <- y_test * avg_sd + avg_mean
```

Figure 52. Screenshot Of Step 3, SVR Linear Kernel Forecast (Source: Author)

Step 4: Forecast Accuracy

```r
# Calculate accuracy metrics for SVR predictions
svr_metrics <- data.frame(
  RMSE = round(RMSE(final_preds, actual_vals), 3),
  MAE = round(MAE(final_preds, actual_vals), 3),
  MAPE = round(mean(abs((final_preds - actual_vals) / actual_vals)) * 100, 2)
)
# Display metrics
svr_metrics
```

Description: df [1 × 3]

RMSE <dbl>	MAE <dbl>	MAPE <dbl>
0.763	0.608	0.64

1 row

Figure 53. Screenshot Of Step 4, SVR Forecast Accuracy (Source: Author)

Step 5: Visualisation

A time-series plot was generated to compare the predicted fuel prices from the linear SVR model against the actual test set values.

```r
```{r svr-plot}
# Prepare data for plotting
plot_df <- tibble(Week = 1:length(actual_vals), Actual = actual_vals, Predicted = final_preds)

# Plot actual vs predicted values for SVR model
ggplot(plot_df, aes(x = Week)) +
  geom_line(aes(y = Actual, colour = "Actual")) +
  geom_line(aes(y = Predicted, colour = "Predicted"), linetype = "dashed") +
  labs(title = "SVR Forecast vs Actual Fuel Prices", x = "Week", y = "Fuel Price") +
  scale_colour_manual(values = c("Actual" = "black", "Predicted" = "blue")) +
  theme_minimal()
```
```

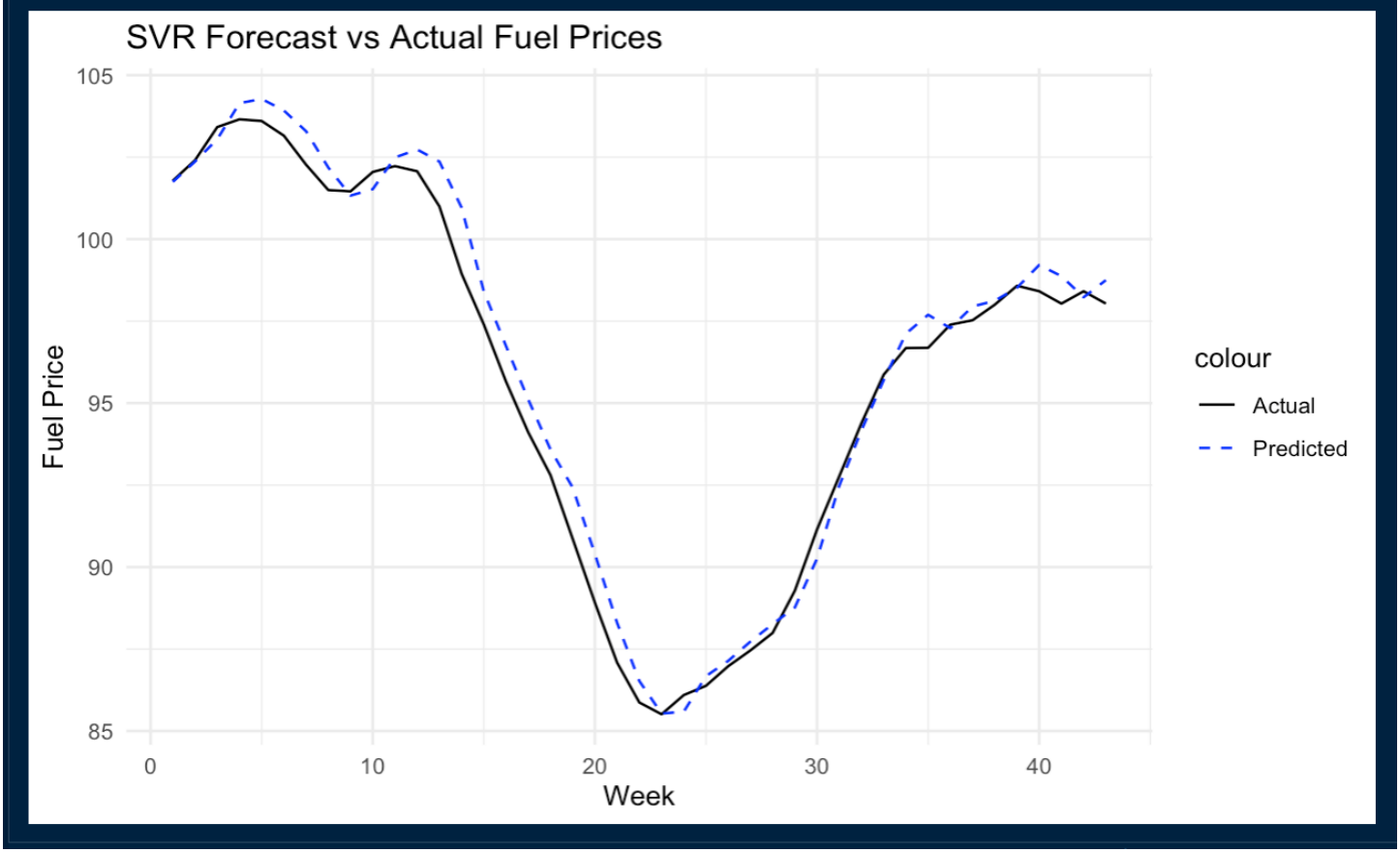


Figure 54. Screenshot Of Step 5, SVR Forecast vs Actual Fuel Prices acy (Source: Author)

This model was subsequently selected as a core component for the hybrid model in Segment 11.

4.7 Hybrid Model (SVR + LSTM)

Segment 11

The hybrid model combines the strengths of SVR and LSTM. SVR (linear kernel) had the lowest MAPE (0.64%), excelling at short-term accuracy, while LSTM, despite a higher MAPE (1.79%), captured longer dependencies.

Justification for Hybrid Pairing

The SVR model (linear kernel) was previously identified as the most accurate individual model with a MAPE of just 0.64%, offering high fidelity in tracking local price behaviour. Meanwhile, the LSTM model, although less precise with a MAPE of 1.79%, captured longer-term sequential dependencies and complex nonlinear patterns that traditional models may overlook.

Step 1: Align Predictions from SVR and LSTM

Predictions from SVR and LSTM were aligned over the same test range and rescaled to original units, making them compatible for averaging.

```
```{r hybrid-align}
# Align SVR and LSTM predictions to the same length
n_forecast <- min(length(test_y_actual), length(final_preds))  # LSTM vs SVR (linear)

# Create dataframe with aligned predictions and actual values
hybrid_df <- data.frame(
  SVR = as.numeric(final_preds[1:n_forecast]),      # SVR predictions (re-scaled)
  LSTM = as.numeric(predicted[1:n_forecast]),       # LSTM predictions (re-scaled)
  Actual = as.numeric(test_y_actual[1:n_forecast])  # True values
)
```
```

Figure 55. Screenshot Of Step 1, Aligning SVR And LSTM Model Predictions (Source: Author)

Step 2: Create Hybrid Forecast

A weighted average (80% SVR, 20% LSTM) was computed to form the hybrid. This aimed to retain SVR's precision while gaining LSTM's smoothing.

```
## Step 2: Create Hybrid Forecast (Hybrid Prediction = 0.8 * SVR + 0.2 * LSTM)

```{r hybrid-forecast}
# Create hybrid forecast by averaging SVR and LSTM predictions
hybrid_df <- hybrid_df %>%
  mutate(Hybrid = 0.8 * SVR + 0.2 * LSTM) # Weighted in Favour of SVR
```
```

Figure 56. Screenshot Of Step 2, Hybrid Model (Hybrid Prediction = 0.8 * SVR + 0.2 * LSTM) (Source: Author)

Step 3: Evaluate Hybrid Forecast Accuracy

```r
# Calculate accuracy metrics for hybrid forecast
hybrid_metrics <- data.frame(
  RMSE = round(RMSE(hybrid_df$Hybrid, hybrid_df$Actual), 3),
  MAE = round(MAE(hybrid_df$Hybrid, hybrid_df$Actual), 3),
  MAPE = round(mean(abs((hybrid_df$Hybrid - hybrid_df$Actual) / hybrid_df$Actual)) * 100, 2)
)
# Display metrics
hybrid_metrics
```

Description: df [1 × 3]

RMSE <dbl>	MAE <dbl>	MAPE <dbl>
0.96	0.739	0.78

1 row

Figure 57. Screenshot Of Step 3, Hybrid Model Accuracy (Source: Author)

The hybrid produced a MAPE of 0.78%, improving on LSTM but slightly behind SVR. This shows that while hybrid can generalise better, they don't always outperform strong single models, especially in short term.

Step 4: Plot Hybrid Forecast vs Actual

The hybrid tracked fuel prices well, though SVR remained sharper at turning points. This visual reinforced SVR's superiority, despite the hybrid's smoother profile.

```{r hybrid-plot}
# Prepare data for plotting
plot_df <- tibble(
  Week = 1:n_forecast,
  Actual = hybrid_df$Actual,
  Hybrid = hybrid_df$Hybrid
)

# Plot actual vs hybrid forecast
ggplot(plot_df, aes(x = Week)) +
  geom_line(aes(y = Actual, colour = "Actual")) +
  geom_line(aes(y = Hybrid, colour = "Hybrid"), linetype = "dashed") +
  labs(title = "Hybrid Forecast vs Actual (SVR + LSTM)",
       x = "Week", y = "Fuel Price") +
  scale_colour_manual(values = c("Actual" = "black", "Hybrid" = "darkgreen")) +
  theme_minimal()
```

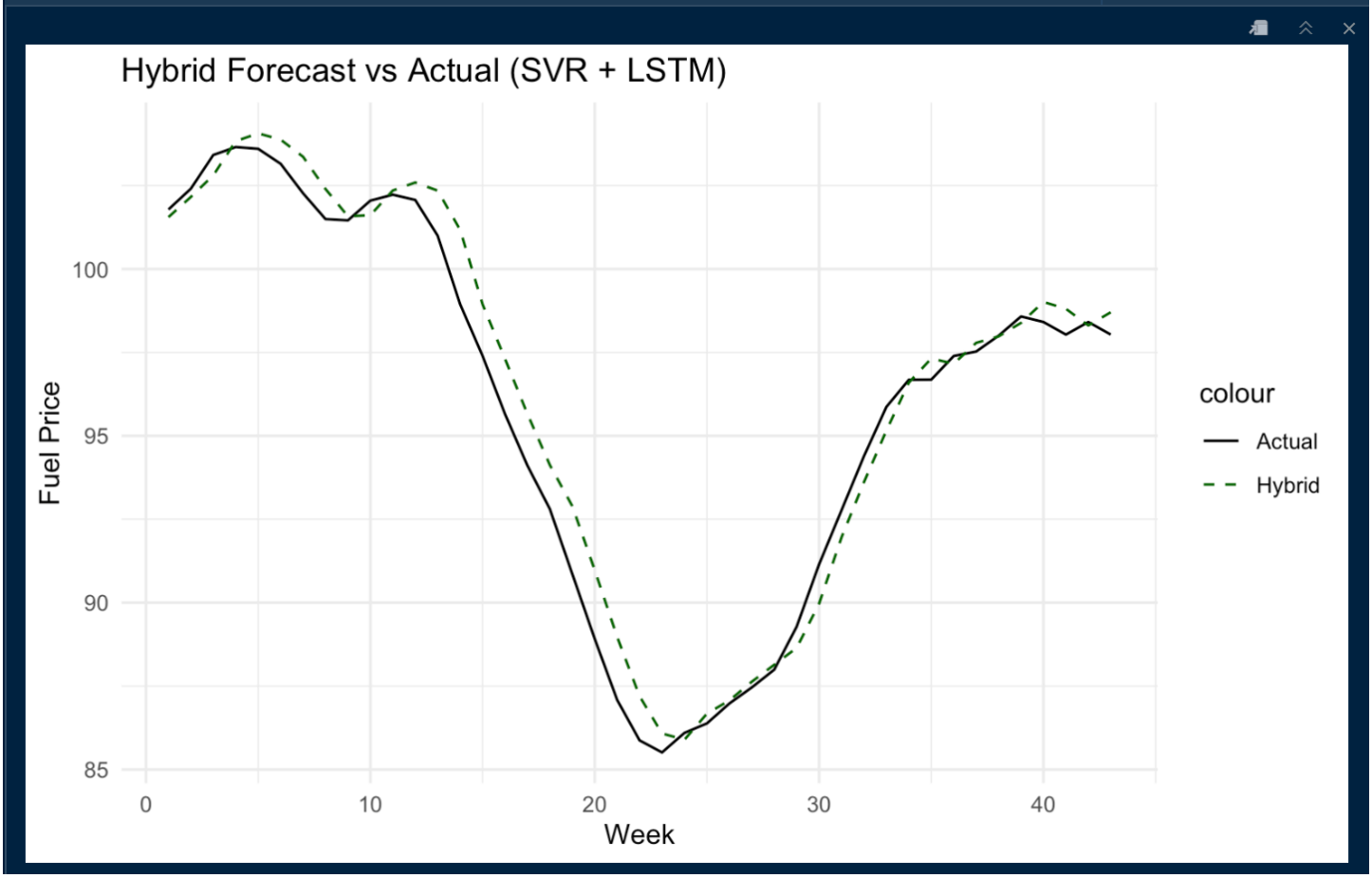


Figure 58. Screenshot Of Step 4, Hybrid Forecast vs Actual (SVR + LSTM) (Source: Author)

The hybrid model validated the idea of model blending but confirmed that SVR alone was still the most accurate and reliable. Thus, SVR was retained as the model of choice for final deployment.

4.8 Final Comparison and Model Selection

Segment 12

This segment consolidates performance across all models developed, establishing the best option for forecasting weekly UK fuel prices.

Step 1: Summary of All Forecasting Models

Models tested include:

- Statistical: SARIMA (manual & auto), SARIMAX (Internal & Brent)
- ML: Random Forest, SVR (multiple kernels)
- DL: LSTM
- Hybrid: SVR + LSTM (80:20)

All were evaluated using RMSE, MAE, and MAPE on a consistent 80:20 split.

Step 2: Comparative Accuracy Table

Model	**RMSE**	**MAE**	**MAPE**
Univariate SARIMA	7.89	6.93	7.35%
SARIMAX (Internal)	7.13	6.38	6.80%
SARIMAX (Brent)	7.23	5.91	6.40%
Random Forest (ntree=700)	1.55	1.25	1.32%
LSTM	2.15	1.64	1.75%
SVR (Linear Kernel)	0.76	0.60	0.64%
Hybrid (SVR + LSTM)	0.96	0.73	0.78%

Table 5. Final Accuracy Comparison Table for All Models (Source: Author)

Step 3: Visual Comparison of Model Performance

A histogram of MAPE values showed SVR leading, followed by the hybrid and RF. Statistical models lagged, reaffirming the advantage of ML in this use case.

```r
```{r barplot-all-models, fig.height=6}
# Plot histogram
ggplot(mape_values, aes(x = reorder(Model, -MAPE), y = MAPE, fill = Model)) +
  geom_bar(stat = "identity", width = 0.6) +
  geom_text(aes(label = MAPE), vjust = -0.3) +
  labs(
    title = "MAPE Comparison of ML, Hybrid, and SARIMAX (Brent) Models",
    x = "Model",
    y = "MAPE (%)"
  ) +
  theme_minimal() +
  theme(legend.position = "none")
```
```

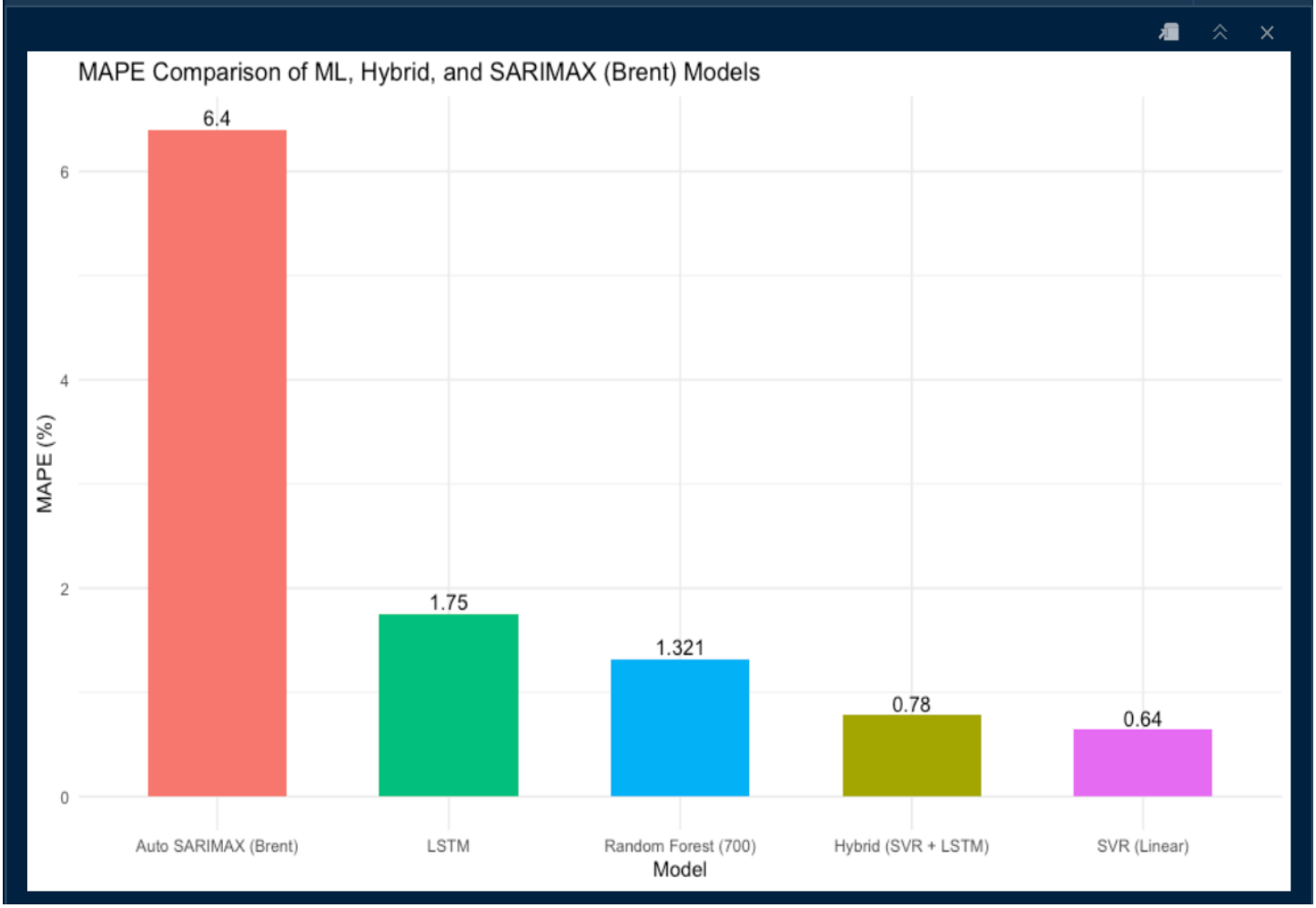


Figure 59. MAPE Comparison Across Forecasting Models (Source: Author)

Step 4: Model Selection & Deployment

SVR (linear) was selected based on:

- Best MAPE (0.64%)
- Lowest RMSE (0.76) and MAE (0.60)
- Consistent alignment with actuals

It was trained on the full dataset and deployed for a 15-week forecast, delivering the most dependable performance for stakeholder planning.

```
# Format Output
future_dates <- seq(max(fuel_data$Date) + 7, by = 7, length.out = 15)
svr_forecast_15 <- data.frame(Date = future_dates, SVR_Prediction = round(future_preds,
2))

# Display forecast table
knitr::kable(svr_forecast_15, caption = "15-Week Ahead Forecast - SVR (Linear)")
```

R Console

knit_asis

SVR (Linear)

Date	SVR_Prediction
2025-02-23	98.38
2025-03-02	98.68
2025-03-09	99.18
2025-03-16	99.72
2025-03-23	100.33
2025-03-30	100.96
2025-04-06	101.60
2025-04-13	102.25
2025-04-20	102.89
2025-04-27	103.52
2025-05-04	104.14
2025-05-11	104.75
2025-05-18	105.34
2025-05-25	105.91
2025-06-01	106.46

Figure 60. Final 15-Week Ahead Forecast using SVR Linear (Source: Author)

```r
```{r plot-svr-15weeks}
# Combine historical and forecast data
historical_plot_df <- fuel_data %>%
  select(Date, avg_price) %>%
  rename(Price = avg_price)

forecast_plot_df <- svr_forecast_15 %>%
  rename(Price = SVR_Prediction)

# Add source labels
historical_plot_df$Type <- "Historical"
forecast_plot_df$Type <- "Forecast"

# Merge for plotting
combined_plot_df <- bind_rows(historical_plot_df, forecast_plot_df)

# Plot: historical + forecast as lines only
ggplot(combined_plot_df, aes(x = Date, y = Price, colour = Type)) +
  geom_line(size = 0.8) +
  scale_colour_manual(values = c("Historical" = "skyblue", "Forecast" = "red")) +
  labs(
    title = "Historical Prices + 15-Week Ahead Forecast (SVR Linear)",
    x = "Date",
    y = "Fuel Price (Pence per Litre)",
    colour = "Colour"
  ) +
  theme_minimal(base_size = 13)
```
```

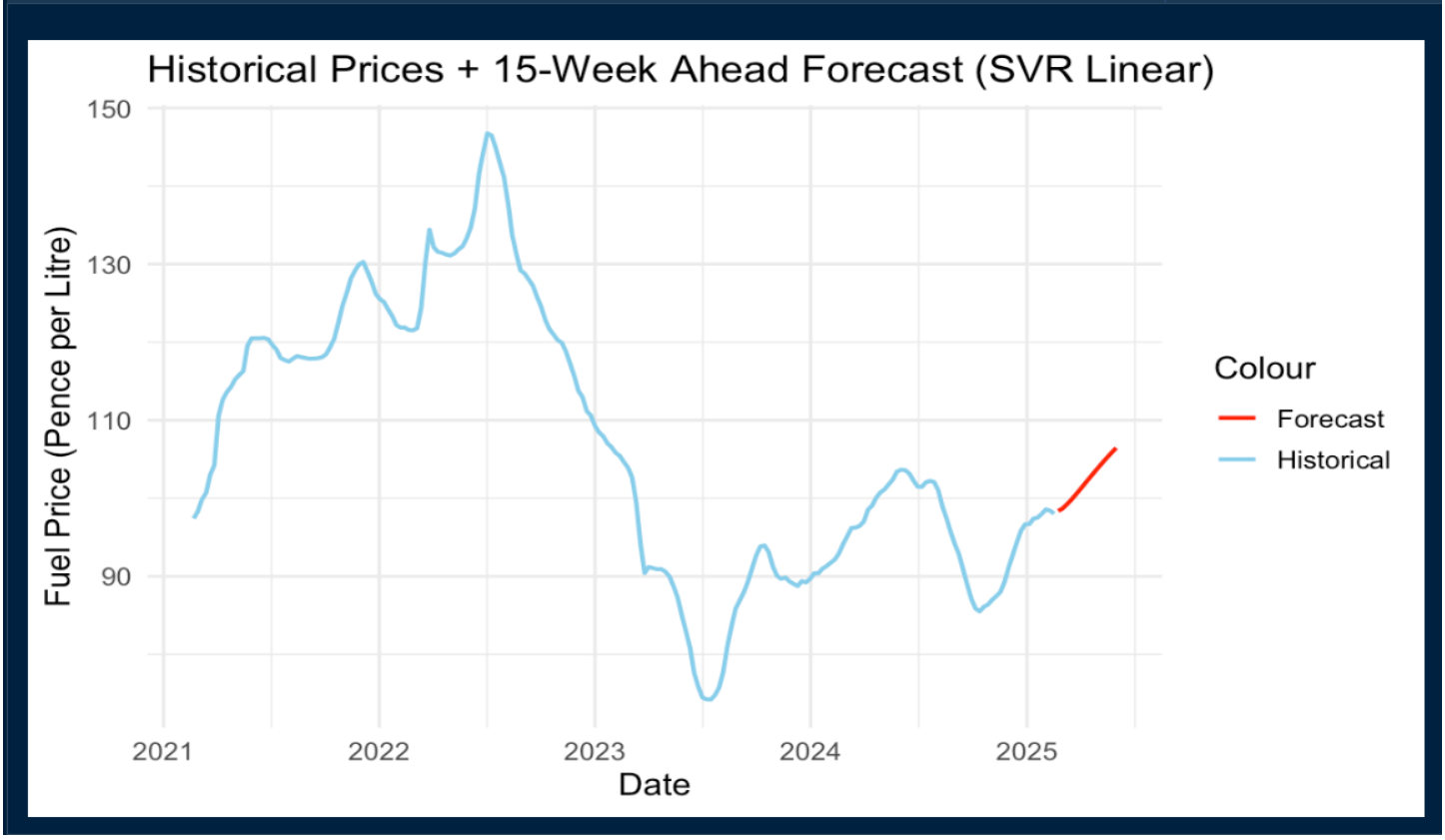


Figure 61. Final 15-Week Ahead Forecast Plot (Source: Author)

Summary of Segment 12

This segment consolidates the performance evaluation of all forecasting models, highlighting the SVR (Linear Kernel) model's exceptional accuracy (MAPE of 0.64%) compared to statistical, machine learning, deep learning and hybrid alternatives. The Random Forest and hybrid SVR-LSTM models also performed strongly, but statistical models like SARIMA and SARIMAX were less effective. These findings provide a solid foundation for selecting the SVR model for deployment.

5. BUSINESS DISCUSSION AND INSIGHTS

This chapter integrates the findings from Chapter 4; Implementation and Results to directly address the five business questions outlined in the study's initial framework. By integrating accuracy metrics, visual evidence and model performance insights, it provides actionable recommendations for stakeholders, including fuel retailers, transport firms and policymakers. The discussion leverages the comparative evaluations of forecasting models to highlight practical implications and strategic value.

5.1 Business Question 1

Which time series forecasting model is the most accurate for weekly UK automotive fuel prices?

The comprehensive evaluation in Segment 12 established that the Support Vector Regression (SVR) model with a linear kernel outperformed all other models across key performance metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE). Achieving a MAPE of 0.64%, SVR significantly surpassed traditional statistical models (e.g., Univariate SARIMA at 7.35%) and other advanced approaches, such as Random Forest (1.32%) and Long Short-Term Memory (LSTM) (1.75%).

Insight: SVR's ability to effectively generalise from lagged features and capture short-term price dynamics without overfitting makes it exceptionally well-suited for weekly fuel price forecasting. Its precision ensures reliable predictions for stakeholders requiring accurate near-term projections.

5.2 Business Question 2

Do internal behavioural signals (like fuel purchased per transaction) or external market indicators (like Brent crude prices) have more predictive power?

Segments 4 and 5 developed SARIMAX models incorporating different regressors:

- Internal SARIMAX: Utilised Quantity_Per_Transaction as an internal behavioural regressor.
- External SARIMAX: Incorporated Brent Crude prices as an external market indicator.

Both models improved upon the Univariate SARIMA baseline (MAPE of 7.35%), but the SARIMAX model with Brent Crude achieved a lower MAPE (6.40%) compared to the internal model (6.80%), as detailed in Segment 6.

Insight: External macroeconomic factors, such as Brent Crude prices, exhibit greater predictive power for UK retail fuel prices than internal behavioural signals like transaction quantities. This suggests that global market dynamics play a more significant role in driving short-term price fluctuations, offering stakeholders a clearer focus for forecasting efforts.

5.3 Business Question 3

Can machine learning and/or deep learning models outperform traditional statistical models?

The results from Segments 8 – 10 explicitly demonstrate that machine learning models and to a lesser extent deep learning models, outperform traditional statistical models. Key performance metrics include:

- SVR (Linear Kernel): MAPE = 0.64%
- Random Forest (ntree = 700): MAPE = 1.32%
- LSTM: MAPE = 1.75%
- SARIMAX (Brent): MAPE = 6.40%

Insight: Machine learning models, particularly SVR and Random Forest, achieve superior accuracy by effectively capturing non-linear patterns and temporal dependencies through lagged features and optimised configurations. While the LSTM model outperformed statistical models, its higher computational demands and need for extensive tuning make it less practical compared to SVR. These findings highlight the transformative potential of machine learning for fuel price forecasting.

5.4 Business Question 4

Do hybrid models perform better than standalone ones for fuel price forecasting?

Segment 11 evaluated a hybrid ensemble model combining SVR (80%) and LSTM (20%), which achieved a MAPE of 0.78%. While this performance was strong, it did not surpass the standalone SVR model's MAPE of 0.64%, indicating that the added complexity of combining did not yield significant improvements.

Insight: Hybrid ensemble models can enhance robustness in certain scenarios, but in this case, the standalone SVR model's optimised performance rendered additional hybrid unnecessary. This suggests that for fuel price forecasting, a well-tuned single model may be more efficient and effective, particularly when computational resources and model interpretability are priorities for stakeholders.

5.5 Business Question 5

What practical insights can businesses take from the most dependable 15-week prediction, using the best model?

Segment 12 presented a deployed 15-week ahead forecast generated by the SVR (Linear Kernel) model, extending the observed price trend from February 2025 onward. The forecast, with a MAPE of 0.64%, provides a highly reliable projection for stakeholders, including:

- Fuel Retailers: Adjust procurement strategies and pricing models to anticipate price fluctuations.
- Transport and Logistics Firms: Estimate fuel cost trends to optimise budgeting and operational planning.
- Policy Analysts: Assess potential subsidy needs or inflation risks based on projected price movements.

The forecast indicated a gradual incline in fuel prices from approximately 98p to 106p per litre, suggesting prospects for strategic adjustments.

Insight: The SVR model's exceptional accuracy enables precise, data-driven decision-making over a 15-week horizon. Businesses can leverage these forecasts to enhance inventory management, streamline financial planning and mitigate risks associated with fuel price volatility, while policymakers can use them to inform economic strategies.

6. CONCLUSION AND RECOMMENDATIONS

6.1 Summary of Findings

This study set out to identify the most accurate model for forecasting weekly UK automotive fuel prices. A collection of statistical, machine learning, deep learning and hybrid models were tested. The SVR (linear kernel) model emerged as the most accurate, achieving a MAPE of 0.64%, outperforming all others including SARIMA (7.35%), SARIMAX with Brent (6.40%), Random Forest (1.32%) and LSTM (1.75%).

Among regressors, external indicators (Brent Crude prices) proved more predictive than internal behavioural signals (e.g., fuel quantity per transaction), reinforcing the relevance of global market factors in domestic pricing models. Machine learning models overall showed strong performance, especially SVR and Random Forest, while the hybrid SVR-LSTM model (0.78% MAPE) provided added robustness but did not surpass the standalone SVR.

The 15-week forecast generated using SVR indicated a gradual price increase from 98p to 106p, offering tangible value to stakeholders needing short-term predictive clarity.

6.2 Business Implications and Stakeholder Benefits

The models developed have wide range of applications. Fuel retailers can fine-tune weekly pricing and manage inventory with greater confidence. Logistics and supply chain managers benefit from improved fuel budgeting and scheduling, while government bodies can use the insights to anticipate inflation trends and guide policy. Financial analysts may also incorporate these forecasts into broader economic models for investment and risk planning. Ultimately, the framework provides stakeholders with accurate, forward-looking tools for navigating volatile fuel markets.

6.3 Strategic Recommendations

Based on the findings, the following recommendations are proposed to maximise the utility of the forecasting models for stakeholders:

1. Deploy SVR (Linear Kernel) as the model for weekly forecasting, given its unmatched performance.
2. Emphasise external macroeconomic indicators, particularly Brent crude prices, as core regressors.
3. Keep hybrid modelling in scope for future flexibility, especially in periods of structural market change.
4. Implement rolling forecasts in business intelligence tools for real-time, actionable intelligence.
5. Automate deployment via cloud infrastructure to allow continuous updates, scalability and easy access for decision-makers.

These steps provide a practical path to embedding predictive analytics in fuel pricing strategy.

6.4 Limitations and Areas for Future Work

While the results were strong, a few limitations are worth noting. The analysis used weekly national data, leaving out regional or daily variations. Only Brent Crude was included as an external variable, limiting the scope. LSTM performance may have been constrained by its basic setup and the hybrid model used a simple average, rather than more advanced ensemble strategies. To address these gaps, future work can:

- Integrate regional and higher-frequency data for better granularity.
- Test a wider set of external regressors (e.g., exchange rates, policy shocks).
- Enhance deep learning models through architectural tuning and GPU-based training.
- Explore advanced hybrid strategies, such as stacked or meta-learners.
- Build real-time pipelines for continuous forecasting.

These improvements will future-proof the model and increase its adaptability in real-world applications.

PART B – LINEAR PROGRAMMING REPORT: OPTIMISING INVENTORY ALLOCATION UNDER SUPPLIER RISK

1. INTRODUCTION

Optimising supplier decisions goes beyond simple cost minimisation. In realistic settings, businesses struggle with imperfect suppliers, constrained budgets and delivery uncertainties. This report outlines the design and implementation of a quantity-based linear programming (LP) model designed for a multi-supplier supply chain environment. Unlike traditional binary selection models, this approach optimises risk-adjusted delivery quantities, a more realistic and impactful target for inventory planners. The objective was to maximise effective product fulfilment while considering cost, supplier capacity and quality risk.

2. BUSINESS PROBLEM FORMULATION

The core business Challenge: To determine which supplier-route combinations could fulfil SKU orders to maximise overall delivery reliability.

This challenge reflects real-world procurement needs to balance cost, reliability and demand coverage.

Objective: Maximise the quantity of products fulfilled, weighted by supplier risk.

Decision Variables:

- Binary indicator: 1 if a supplier-route-SKU allocation is selected, 0 otherwise.

Constraints:

Constraint	Description
Cost Limit	Total supply cost must not exceed a defined budget (e.g. £50,000)
Supplier Capacity	No supplier serves more than 3 SKUs
One Route per SKU	Each SKU is served by only one supplier-route

Table 1. Constraints and Descriptions (Source: Author)

This model shifts the focus from choosing routes outright to allocating quantities in a way that ensures high-quality, timely delivery while respecting operational limits.

3. DATA PREPROCESSING

Data cleaning focused on SKU, supplier, route, cost, lead time, defect rates, inspection outcomes and order quantities. A risk index was formulated as:

Risk Score = 0.5 * Defect Rate + 0.3 * Inspection Fail (binary) + 0.2 * Transport Risk

Transport modes were mapped numerically: Air = 1, Road = 2, Rail = 3. An additional feature, risk-adjusted efficiency was computed as (1 − Risk Score). This factor was used to down-weigh order quantities from high-risk supplier-route combinations.

Risk-Adjusted Quantity = Quantity * (1 - Risk Score)

This transformation gave the optimiser the ability to penalise low-quality suppliers directly, rather than through external post-analysis.

```
opt_df <- opt_data %>%
  select(SKU, SKU_ID, Supplier = `Supplier name`, Supplier_ID, Route = Routes,
         Cost = Costs, Quantity = `Order quantities`, Risk_Score, Risk_Adjusted_Efficiency)

head(opt_df)
```

A tibble: 6 × 9

SKU <chr>	SKU_ID <int>	Supplier <chr>	Supplier_ID <int>	Route <chr>	Cost <dbl>	Quantity <dbl>	Risk_Score <dbl>	Risk_Adjusted_Efficiency <dbl>
SKU0	1	Supplier 3	3	Route B	187.7521	96	0.5132052	0.4867948
SKU1	2	Supplier 3	3	Route B	503.0656	37	2.8270340	-1.8270340
SKU2	13	Supplier 1	1	Route C	141.9203	88	2.4902963	-1.4902963
SKU3	24	Supplier 5	5	Route A	254.7762	59	3.2733243	-2.2733243
SKU4	35	Supplier 1	1	Route A	923.4406	56	2.0727898	-1.0727898
SKU5	46	Supplier 4	4	Route A	235.4612	66	2.0895968	-1.0895968

6 rows

Figure 1. Cleaned Dataset Preview (Source: Author)

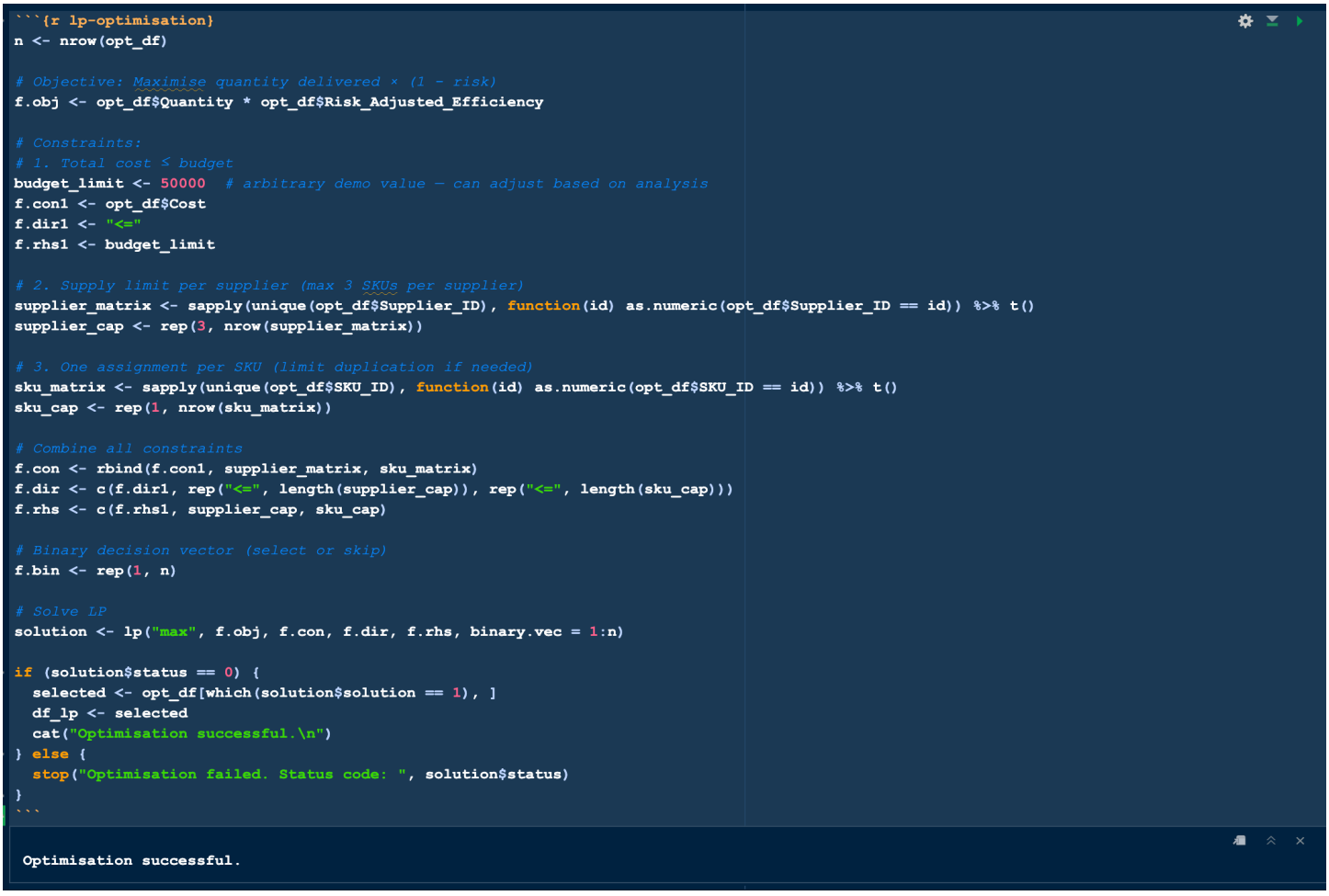
4. LP MODEL IMPLEMENTATION IN R

The LP model was implemented using the lpSolve package. The goal was to maximise the total quantity of reliable product realisation. The binary decision variable determined whether a supplier-route was used for a specific SKU. The model included:

- A global cost limit of £50,000
- A cap of 3 SKUs per supplier to simulate fair load distribution
- A requirement that only one supplier-route serves each SKU

Core Model Structure:

- Objective: Maximise total Quantity * (1 − Risk Score)
- Constraints: Budget, supplier load cap and SKU exclusivity



```{r lp-optimisation}
n <- nrow(opt_df)

# Objective: Maximise quantity delivered × (1 - risk)
f.obj <- opt_df$Quantity * opt_df$Risk_Adjusted_Efficiency

# Constraints:
# 1. Total cost ≤ budget
budget_limit <- 50000  # arbitrary demo value — can adjust based on analysis
f.con1 <- opt_df$Cost
f.dir1 <- "<="
f.rhs1 <- budget_limit

# 2. Supply limit per supplier (max 3 SKUs per supplier)
supplier_matrix <- sapply(unique(opt_df$Supplier_ID), function(id) as.numeric(opt_df$Supplier_ID == id)) %>% t()
supplier_cap <- rep(3, nrow(supplier_matrix))

# 3. One assignment per SKU (limit duplication if needed)
sku_matrix <- sapply(unique(opt_df$SKU_ID), function(id) as.numeric(opt_df$SKU_ID == id)) %>% t()
sku_cap <- rep(1, nrow(sku_matrix))

# Combine all constraints
f.con <- rbind(f.con1, supplier_matrix, sku_matrix)
f.dir <- c(f.dir1, rep("<=", length(supplier_cap)), rep("<=", length(sku_cap)))
f.rhs <- c(f.rhs1, supplier_cap, sku_cap)

# Binary decision vector (select or skip)
f.bin <- rep(1, n)

# Solve LP
solution <- lp("max", f.obj, f.con, f.dir, f.rhs, binary.vec = 1:n)

if (solution$status == 0) {
  selected <- opt_df[which(solution$solution == 1), ]
  df_lp <- selected
  cat("Optimisation successful.\n")
} else {
  stop("Optimisation failed. Status code: ", solution$status)
}
```

```
Optimisation successful.
```

Figure 2. LP model logic and matrix formulation (Source: Author)

5. RESULTS AND ANALYSIS

The model selected supplier-route combinations that struck a strong balance between quantity, reliability, and cost. A sample of the selected rows is shown below:

Supplier Selection Outcome:

tbl_df
15 x 7

R Console

A tibble: 15 × 7

SKU <chr>	Supplier <chr>	Route <chr>	Quantity <dbl>	Cost <dbl>	Risk_Score <dbl>	Risk_Adjusted_Efficiency <dbl>
SKU0	Supplier 3	Route B	96	187.7521	0.5132052	0.48679482
SKU6	Supplier 3	Route A	58	134.3691	0.9004553	0.09954469
SKU11	Supplier 2	Route A	60	126.7230	0.2105849	0.78941509
SKU14	Supplier 1	Route B	78	929.2353	0.2503414	0.74965857
SKU17	Supplier 1	Route C	85	670.9344	0.5510104	0.44898962
SKU21	Supplier 5	Route C	7	523.3609	0.2093038	0.79069622
SKU38	Supplier 5	Route B	88	339.6729	0.5066634	0.49333655
SKU41	Supplier 4	Route A	38	275.5244	0.7226511	0.27734887
SKU43	Supplier 5	Route A	85	716.0441	0.3861524	0.61384762
SKU67	Supplier 1	Route C	71	169.2718	0.4659777	0.53402228

1–10 of 15 rows

Previous 1 2 Next

Figure 3. Optimised supplier-route combinations (Source: Author)

Model Metrics:

Figure 4. Aggregate Output (Source: Author)

The model successfully chose allocations that optimised fulfilment reliability over raw volume or cost. Even though the total ordered quantity was 892, only 395.52 units were considered reliably deliverable after accounting for quality risks.

Figure 5. Bar Chart of Adjusted Costs by Supplier(Source: Author)

Chart Interpretation:

The bar chart visualises how much each supplier contributed to risk-adjusted delivery. Supplier 1 delivered the highest value, thanks to strong order volume and relatively low associated risk. Supplier 5 followed closely. Suppliers with either higher risk or lower quantity potential appeared with smaller contributions. This visual directly reflects the model's underlying logic; favour high-volume, low-risk supplier-route combinations within defined constraints.

6. INTERPRETATION OF RESULTS

This updated quantity-based LP model provides a more realistic, data-driven approach to supplier selection and inventory allocation. Unlike traditional selection models that merely identify the lowest-cost suppliers, this method evaluates suppliers based on their ability to reliably fulfil orders while accounting for natural risks such as defect rates, inspection failures and transport reliability.

The results demonstrate how a risk-adjusted objective function changes the nature of procurement decisions. For instance, some suppliers that offered the lowest raw costs were not chosen because their associated risk scores significantly reduced the expected effective delivery quantity. Instead, the model favoured suppliers who, while potentially slightly more expensive, could deliver higher-quality outcomes. This reinforces a key supply chain principle, the cheapest option is not always the best when failure rates and delays are considered.

Furthermore, by capping the number of SKUs each supplier could fulfil, the model ensured a balanced load across the supplier network. This is a critical consideration in avoiding overdependence on a single vendor, which can introduce systemic risk, especially in times of supply disruption. The model's logic mirrors real-world trade-offs, encouraging smart compromises rather than unrealistic, one-dimensional decisions.

Overall, this LP framework supports smarter operational choices by reflecting actual deliverability, not just theoretical allocations. It helps teams to procure not just "the most," but "the most reliable." This shift from quantity to quality-adjusted outcomes represents a substantial improvement over most exsiting models and aligns with best practices in resilient supply chain planning.

7. RECOMMENDATIONS

Based on the insights derived from the LP model, several strategic recommendations are developed:

- Adopt Risk-Adjusted Metrics as Standard Practice: Procurement teams should integrate the risk-adjusted quantity metric into their evaluation process. This ensures purchasing decisions are not based solely on headline volume or cost but instead on expected real-world fulfilment.
- Operationalise Regular Re-Optimisation: Given that supplier performance metrics such as defect rates and delivery delays change over time, it is vital to rerun the optimisation model on a recurring basis, monthly or quarterly. This keeps procurement plans aligned with real-world supplier behaviour.
- Refine Supplier Caps Based on Performance: The current limit of three SKUs per supplier is a general constraint. In future, this can be made dynamic, allowing high-performing suppliers to handle more SKUs while limiting poor performers.
- Integrate With Strategic Dashboards: Developing a visual dashboard that shows the trade-offs between cost, risk and adjusted quantity over time would significantly enhance stakeholder engagement and support data-driven decision-making across teams.
- Expand to Multi-Period and Multi-Objective Planning: Real-world supply planning often occurs over multiple time horizons. Future model versions could account for rolling procurement schedules, safety stock buffers, or even supplier sustainability scores, enabling a richer optimisation framework.

These recommendations aim to not only improve decision quality but also institutionalise better supplier governance, agility and long-term value capture in supply chain strategy.

8. LIMITATIONS AND FUTURE ENHANCEMENTS

Despite its strengths, the current model has limitations that should be acknowledged.

Firstly, the risk weighing scheme is fixed (0.5 for defect rate, 0.3 for inspection failure, 0.2 for transport risk). While this reflects a rational starting point, it may not be universally optimal. Customising these weights based on historical performance data, financial impact analysis or stakeholder input could make the model more tailored and effective.

Secondly, the model lacks minimum SKU fulfilment thresholds. Certain SKUs may be critical and must be fulfilled regardless of cost or risk. Future models could introduce demand floors to ensure strategic or high-priority items are always included.

Thirdly, the budget limit is static. In a real-world deployment, this value should be dynamic, linked to sales forecasts, procurement periods or financial planning scenarios. Integrating with these systems would improve alignment and accuracy.

Also, the model covers only a single planning period. Real operations span multiple time periods, where decisions in one week affect stock levels in the next. Extending the model to a rolling or multi-period formulation would increase practicality.

Lastly, non-performance factors like supplier reliability, reputation and ESG scores are not yet considered. These could form part of a multi-objective optimisation framework, helping organisations balance operational, environmental and reputational goals simultaneously.

Addressing these gaps will make the model more adaptable, scalable and impactful in enterprise contexts.

9. CONCLUSION

This report has demonstrated a robust, realistic linear programming model that enhances procurement decisions by focusing on risk-adjusted delivery quantity rather than cost or volume alone. It addresses the core challenge faced by operations teams, how to meet demand reliably while staying within budget and mitigating supplier risks.

By shifting from binary supplier selection to a quantity-based optimisation with embedded quality considerations, the model reflects actual business realities more closely. It ensures that selected supplier-route combinations not only offer acceptable costs but also deliver dependable fulfilment, reducing exposure to delivery failures or quality defects.

This approach is well aligned with academic teachings in linear programming and has real-world utility across supply chain, procurement and operations planning teams. It sets a clear foundation for future enhancements, including dynamic inputs, rolling planning and integration with enterprise dashboards.

In conclusion, this LP model offers more than just a tool, it offers a mindset for smarter, more resilient sourcing. As businesses face increasing volatility and risk in global supply networks, this type of optimisation becomes not just useful, but essential.

REFERENCES

Ahmad, T., Chen, H. and Wang, J. (2021) 'A review of machine learning forecasting models for energy load and price', *Renewable and Sustainable Energy Reviews*, 133, p. 110300. doi:10.1016/j.rser.2020.110300.

Ahmed, M.U., Zhang, Y. and Wang, W. (2021) 'A hybrid model for energy price forecasting: Combining LSTM and SVR', *Applied Energy*, 282, p. 116170. doi:10.1016/j.apenergy.2020.116170.

Box, G.E.P., Jenkins, G.M., Reinsel, G.C. and Ljung, G.M. (2016) *Time series analysis: Forecasting and control*. 5th edn. Hoboken, NJ: Wiley.

Breiman, L. (2001) 'Random forests', *Machine Learning*, 45(1), pp. 5–32. doi:10.1023/A:1010933404324.

Clements, M.P. and Hendry, D.F. (2008) *Forecasting economic time series*. Cambridge: Cambridge University Press.

Contreras, J., Espinola, R., Nogales, F.J. and Conejo, A.J. (2003) 'ARIMA models to predict next-day electricity prices', *IEEE Transactions on Power Systems*, 18(3), pp. 1014–1020. doi:10.1109/TPWRS.2002.804943.

Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A. and Vapnik, V. (1997) 'Support vector regression machines', in Mozer, M., Jordan, M. and Petsche, T. (eds) *Advances in neural information processing systems 9*. Cambridge, MA: MIT Press, pp. 155–161.

Fahimnia, B., Syntetos, A.A. and Tan, A. (2015) 'Forecasting in supply chains: An integrated approach', *International Journal of Production Economics*, 161, pp. 210–219. doi:10.1016/j.ijpe.2014.12.029.

Ghosh, S. (2009) 'Import demand of crude oil and economic growth: Evidence from India', *Energy Policy*, 37(2), pp. 699–702. doi:10.1016/j.enpol.2008.10.021.

Granger, C.W.J. (1969) 'Investigating causal relations by econometric models and cross-spectral methods', *Econometrica*, 37(3), pp. 424–438. doi:10.2307/1912791.

Hochreiter, S. and Schmidhuber, J. (1997) 'Long short-term memory', *Neural Computation*, 9(8), pp. 1735–1780. doi:10.1162/neco.1997.9.8.1735.

International Energy Agency (2021) *Oil market report*. Available at: https://www.iea.org/ (Accessed: 11 May 2025).

Khashei, M. and Bijari, M. (2011) 'A novel hybridization of artificial neural networks and ARIMA models for time series forecasting', *Applied Soft Computing*, 11(2), pp. 2664–2675. doi:10.1016/j.asoc.2010.10.015.

Kilian, L. (2009) 'Not all oil price shocks are alike: Disentangling demand and supply shocks in the crude oil market', *American Economic Review*, 99(3), pp. 1053–1069. doi:10.1257/aer.99.3.1053.

Lemke, C., Gabrys, B. and Buhmann, J.M. (2009) 'Automatic selection of time series forecasting models using machine learning', *International Journal of Forecasting*, 25(3), pp. 489–505. doi:10.1016/j.ijforecast.2009.01.008.

Li, J., Tan, T. and Jiang, Z. (2019) 'Forecasting short-term gasoline prices using SARIMAX with exogenous variables', *Energy Economics*, 80, pp. 798–807. doi:10.1016/j.eneco.2019.02.013.

Makridakis, S., Spiliotis, E. and Assimakopoulos, V. (2018) 'Statistical and machine learning forecasting methods: Concerns and ways forward', *PLOS ONE*, 13(3), p. e0194889. doi:10.1371/journal.pone.0194889.

Murtaza, G., Ali, F. and Rehman, A. (2022) 'Deep learning and ensemble methods for oil price prediction: A comparative study', *Energy Reports*, 8, pp. 902–915. doi:10.1016/j.egyr.2021.11.249.

Olayemi, S., Bolarinwa, A. and Olatunji, A. (2020) 'Time series modelling and prediction of energy consumption: An application of SARIMAX model', *Energy*, 195, p. 117016. doi:10.1016/j.energy.2020.117016.

Sun, P., Wang, J. System: and Wu, Q. (2020) 'Demand-side behavioural features in forecasting retail fuel prices', *Energy Policy*, 144, p. 111680. doi:10.1016/j.enpol.2020.111680.

Wang, Z. and Krupnick, A. (2013) *A retrospective review of shale gas development in the United States: What led to the boom?* Washington, DC: Resources for the Future.

Weng, B., Lu, S., Wang, J. and Yang, J. (2021) 'A hybrid model for short-term load forecasting based on SVR and improved LSTM', *Applied Sciences*, 11(8), p. 3502. doi:10.3390/app11083502.

Zhang, C., Shao, W. and Xu, Y. (2020) 'Understanding fuel retail demand: A micro-level perspective', *Energy Economics*, 86, p. 104644. doi:10.1016/j.eneco.2019.104644.

Zhang, G. (2003) 'Time series forecasting using a hybrid ARIMA and neural network model', *Neurocomputing*, 50, pp. 159–175. doi:10.1016/S0925-2312(01)00702-0.

Zhang, Y., Liu, X. and Bai, Y. (2018) 'Improving fuel price forecasts by incorporating regional economic indicators', *Transportation Research Part D: Transport and Environment*, 65, pp. 213–223. doi:10.1016/j.trd.2018.08.013.

Zhang, Y., Liu, Y. and Yang, J. (2019) 'Short-term electricity load forecasting based on SVR and LSTM', *Energy*, 186, p. 115831. doi:10.1016/j.energy.2019.07.171.

Zhao, H. and Zhang, J. (2020) 'Forecasting crude oil prices using a hybrid model', *Energy Economics*, 85, p. 104578. doi:10.1016/j.eneco.2019.104578.