R Notebook

Walmart Sales Forecasting Project

Dataset Description:

Walmart Sales Forecasting dataset from Kaggle includes weekly sales data from different Walmart stores, holidays, average temperature, fuel prices, CPI, and unemployment rate. Business Problem:

Improve Walmart's sales forecasting abilities to optimize inventory management, staffing decisions, and promotional strategies. The goal is to identify the impact of external factors (e.g., holidays, economic variables) and use predictive models to create actionable insights. Technical Approach:

Use a hybrid approach with time series analysis (ARIMA) and machine learning models (Linear Regression, Decision Tree, XGBoost). Evaluate model performance using RMSE, MAPE, and R-squared.

1: Load and Explore the Dataset

```
# Load necessary libraries
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
# Load the dataset
data <- read.csv("Walmart_Sales.csv", stringsAsFactors = FALSE)

# Inspect the structure and summary of the dataset
str(data)
```

```
## 'data.frame':    6435 obs. of  8 variables:
##  $ Store       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Date        : chr  "05-02-2010" "12-02-2010" "19-02-2010" "26-02-2010" ...
##  $ Weekly_Sales: num  1643691 1641957 1611968 1409728 1554807 ...
##  $ Holiday_Flag: int  0 1 0 0 0 0 0 0 0 0 ...
##  $ Temperature : num  42.3 38.5 39.9 46.6 46.5 ...
##  $ Fuel_Price  : num  2.57 2.55 2.51 2.56 2.62 ...
##  $ CPI         : num  211 211 211 211 211 ...
##  $ Unemployment: num  8.11 8.11 8.11 8.11 8.11 ...
```

```
summary(data)
```

```
##      Store        Date            Weekly_Sales      Holiday_Flag
##  Min.   : 1   Length:6435        Min.   : 209986   Min.   :0.00000
##  1st Qu.:12   Class :character   1st Qu.: 553350   1st Qu.:0.00000
##  Median :23   Mode  :character   Median : 960746   Median :0.00000
##  Mean   :23                      Mean   :1046965   Mean   :0.06993
##  3rd Qu.:34                      3rd Qu.:1420159   3rd Qu.:0.00000
##  Max.   :45                      Max.   :3818686   Max.   :1.00000
##   Temperature       Fuel_Price         CPI         Unemployment
##  Min.   : -2.06   Min.   :2.472   Min.   :126.1   Min.   : 3.879
##  1st Qu.: 47.46   1st Qu.:2.933   1st Qu.:131.7   1st Qu.: 6.891
##  Median : 62.67   Median :3.445   Median :182.6   Median : 7.874
##  Mean   : 60.66   Mean   :3.359   Mean   :171.6   Mean   : 7.999
##  3rd Qu.: 74.94   3rd Qu.:3.735   3rd Qu.:212.7   3rd Qu.: 8.622
##  Max.   :100.14   Max.   :4.468   Max.   :227.2   Max.   :14.313
```

```
# Check for missing values
colSums(is.na(data))
```

```
##        Store         Date Weekly_Sales Holiday_Flag  Temperature   Fuel_Price
##            0            0            0            0            0            0
##          CPI Unemployment
##            0            0
```

```
# Display the first few rows of the dataset
head(data)
```

	Store <int>	Date <chr>	Weekly_Sales <dbl>	Holiday_Flag <int>	Temperature <dbl>	Fuel_Price <dbl>	CPI <dbl>
1	1	05-02-2010	1643691	0	42.31	2.572	211.0964
2	1	12-02-2010	1641957	1	38.51	2.548	211.2422
3	1	19-02-2010	1611968	0	39.93	2.514	211.2891
4	1	26-02-2010	1409728	0	46.63	2.561	211.3196

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI
	<int>	<chr>	<dbl>	<int>	<dbl>	<dbl>	<dbl>
5	1	05-03-2010	1554807	0	46.50	2.625	211.3501
6	1	12-03-2010	1439542	0	57.79	2.667	211.3806

6 rows | 1-8 of 9 columns

2: Feature Engineering

The dataset was preprocessed by:

Converting the Date column into a proper date format. Adding lagged features (Lag_1, Lag_2) and a rolling mean (Rolling_4) for time series modeling. Creating a Holiday_Indicator variable to distinguish holiday vs. non-holiday weeks.

```
# Load required libraries
library(dplyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(zoo)
```

```
## Warning: package 'zoo' was built under R version 4.3.3
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
# Preprocess the data
data$Date <- as.Date(data$Date, format = "%d-%m-%Y")  # Convert Date column
data <- data %>%
  arrange(Store, Date) %>%  # Sort by Store and Date
  mutate(Year = year(Date),
         Month = month(Date),
         Week = week(Date),
         Lag_1 = lag(Weekly_Sales, 1),
         Lag_2 = lag(Weekly_Sales, 2),
         Rolling_4 = rollapply(Weekly_Sales, width = 4, FUN = mean, fill = NA, align = "right"),
         Holiday_Indicator = as.factor(ifelse(Holiday_Flag == 1, "Holiday", "Non-Holiday")))

# Check the structure and summary
head(data)
```

	Store <int>	Date <date>	Weekly_Sales <dbl>	Holiday_Flag <int>	Temperature <dbl>	Fuel_Price <dbl>	CPI <dbl>
1	1	2010-02-05	1643691	0	42.31	2.572	211.0964
2	1	2010-02-12	1641957	1	38.51	2.548	211.2422
3	1	2010-02-19	1611968	0	39.93	2.514	211.2891
4	1	2010-02-26	1409728	0	46.63	2.561	211.3196
5	1	2010-03-05	1554807	0	46.50	2.625	211.3501
6	1	2010-03-12	1439542	0	57.79	2.667	211.3806

6 rows | 1-8 of 16 columns

3. Exploratory Data Analysis

```
# Load necessary libraries
library(ggplot2)
library(scales)
```

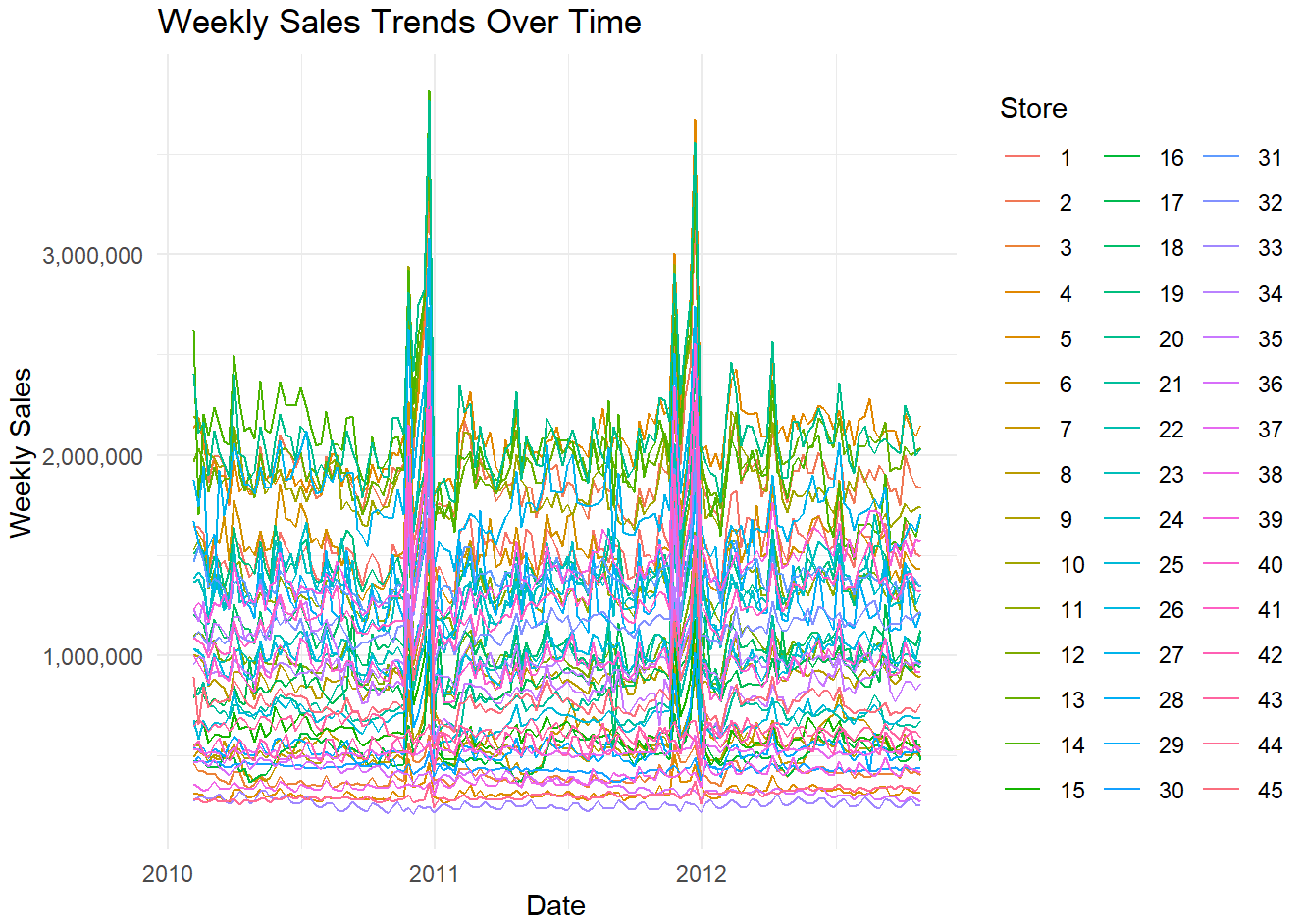
```
## Warning: package 'scales' was built under R version 4.3.3
```

```
library(reshape2)
```

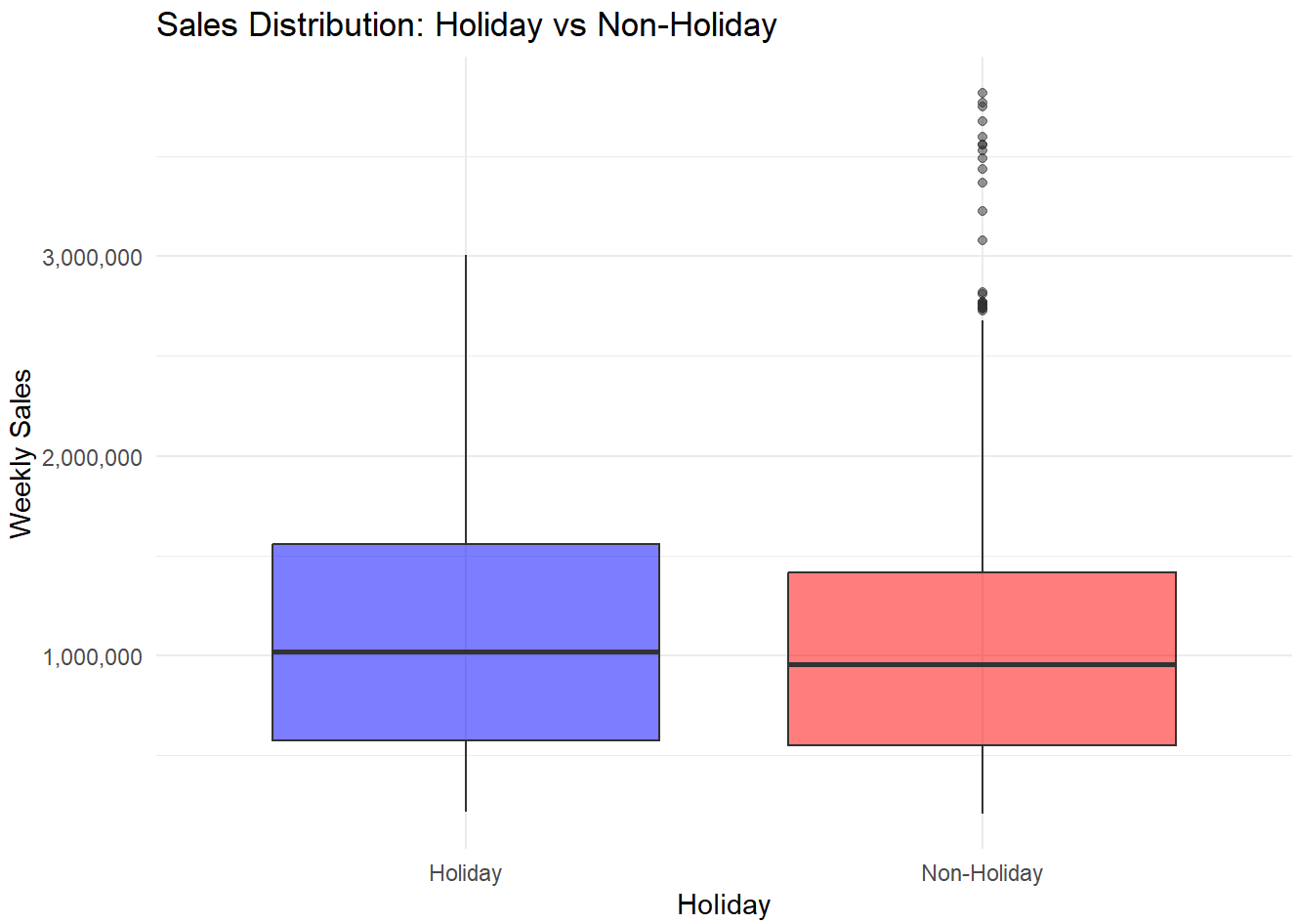
```
## Warning: package 'reshape2' was built under R version 4.3.3
```

```
library(dplyr)

# 1. Visualize sales trends over time
ggplot(data, aes(x = Date, y = Weekly_Sales, color = as.factor(Store))) +
  geom_line() +
  labs(title = "Weekly Sales Trends Over Time", x = "Date", y = "Weekly Sales", color = "Store")
+
  scale_y_continuous(labels = comma) +  # Format y-axis with commas
  theme_minimal()
```

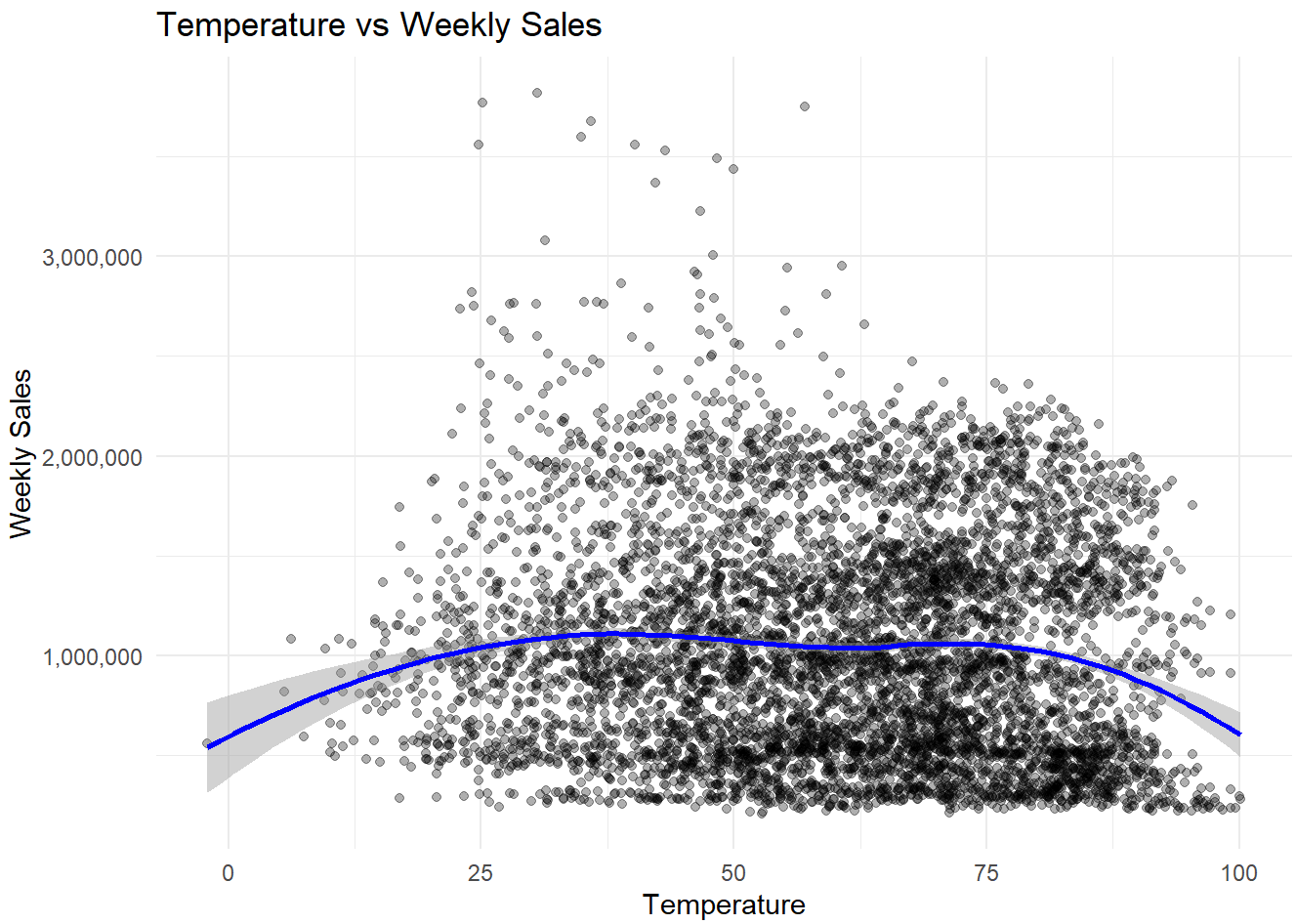



```
# 2. Sales distribution by holiday and non-holiday
ggplot(data, aes(x = Holiday_Indicator, y = Weekly_Sales)) +
  geom_boxplot(fill = c("blue", "red"), alpha = 0.5) +
  labs(title = "Sales Distribution: Holiday vs Non-Holiday", x = "Holiday", y = "Weekly Sales")
+
  scale_y_continuous(labels = comma) +  # Format y-axis with commas
  theme_minimal()
```




```
# 3. Impact of temperature on weekly sales
ggplot(data, aes(x = Temperature, y = Weekly_Sales)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "loess", color = "blue") +
  labs(title = "Temperature vs Weekly Sales", x = "Temperature", y = "Weekly Sales") +
  scale_y_continuous(labels = comma) +  # Format y-axis with commas
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# 4. Correlation heatmap (optional, for numeric features)
cor_data <- data %>%
  select(Weekly_Sales, Temperature, Fuel_Price, CPI, Unemployment) %>%
  na.omit()
cor_matrix <- round(cor(cor_data), 2)
cor_melt <- melt(cor_matrix)
ggplot(cor_melt, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "red", high = "green", mid = "white", midpoint = 0) +
  labels(title = "Correlation Heatmap", x = "", y = "") +
  geom_text(aes(label = value), color = "black")  # Add correlation values to tiles
```

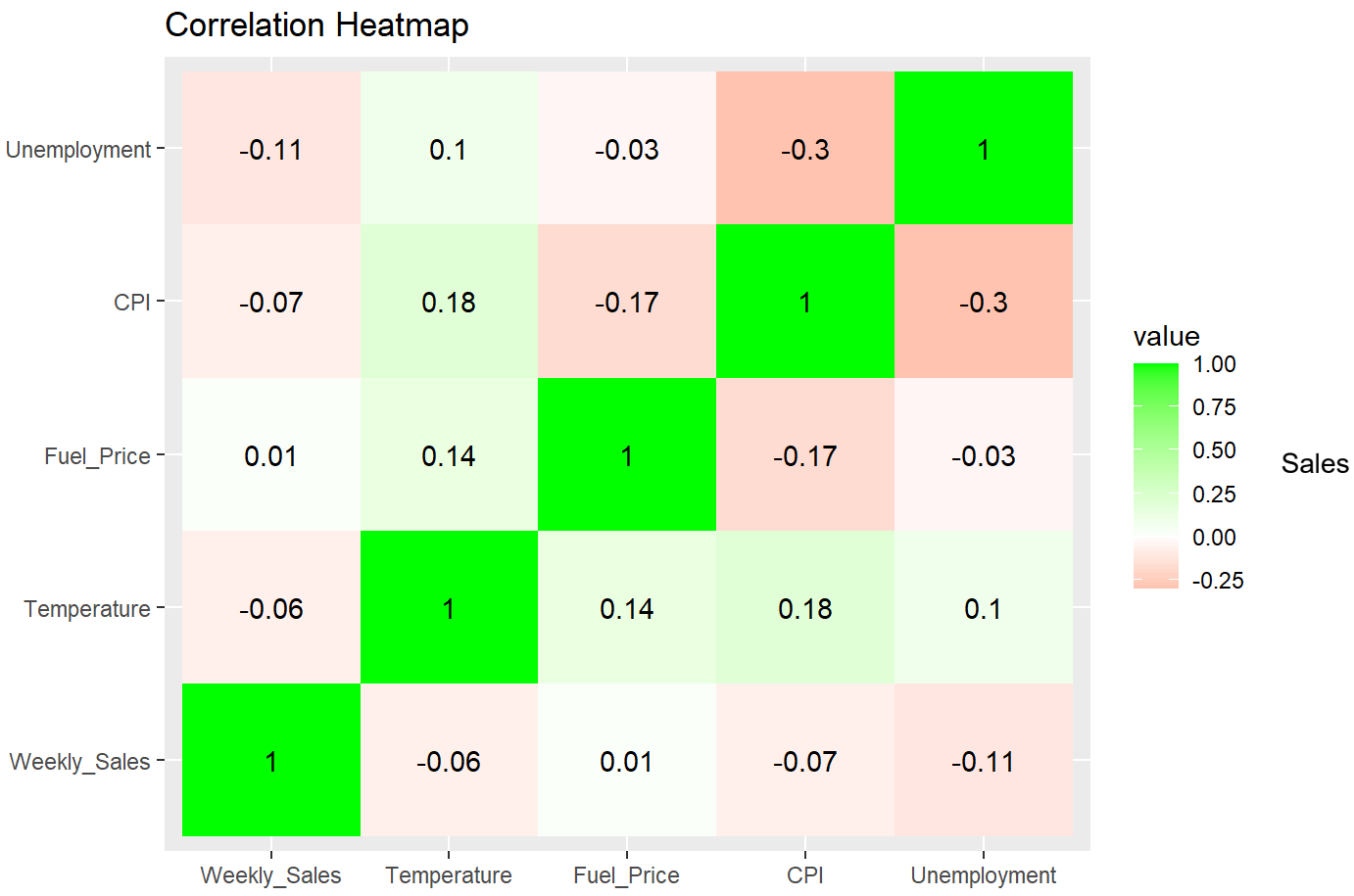


exhibit seasonal patterns with noticeable spikes during holidays (e.g., Thanksgiving, Christmas).

Sales are generally higher during holidays, highlighting the importance of holiday weeks in forecasting.

Sales exhibit a slight parabolic relationship with temperature, peaking at moderate temperatures.

CPI and Unemployment show moderate correlations with weekly sales.

4 MODEL IMPLIMENTAION

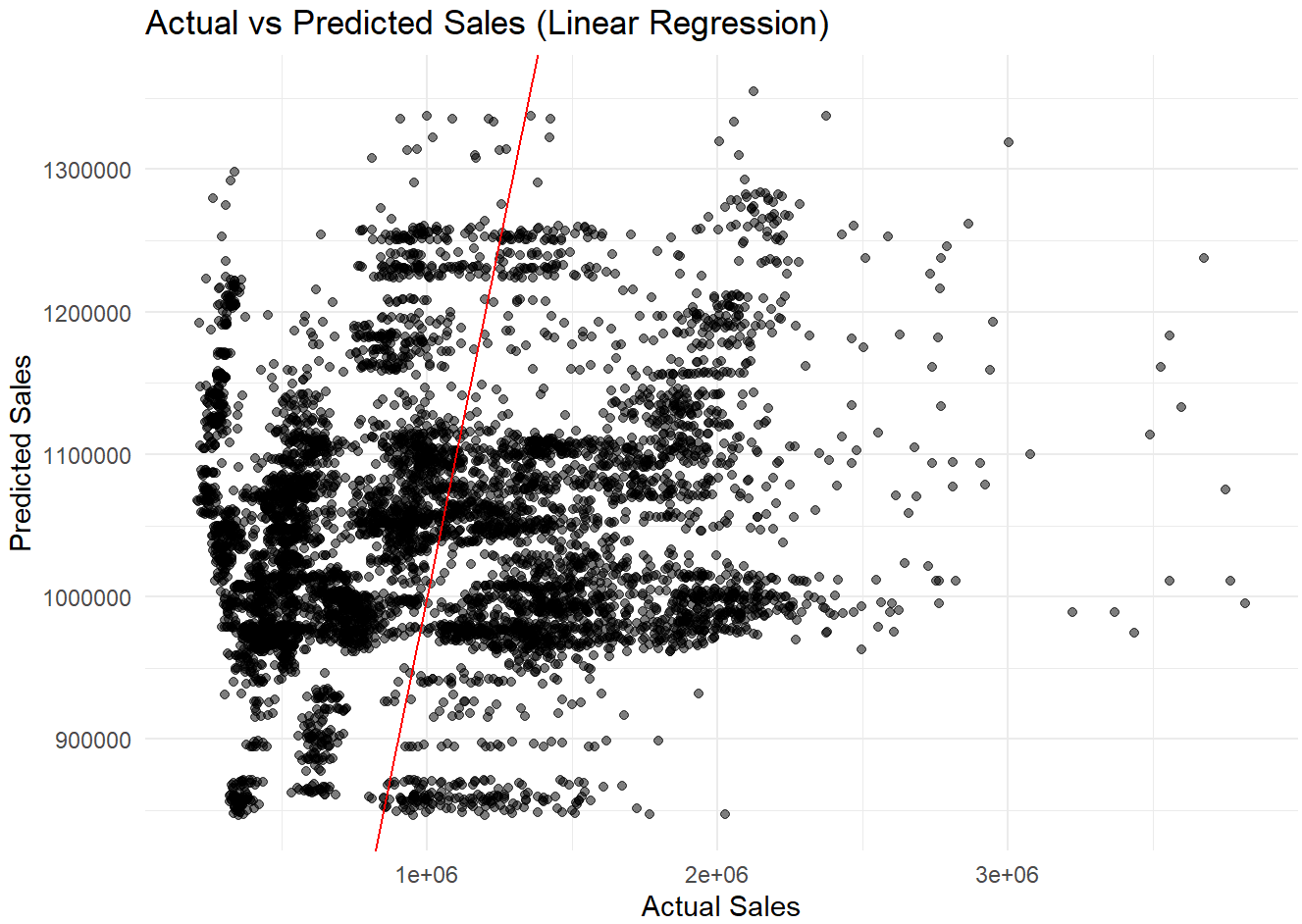
4.1 IINEAR REGRESSION

```
# Linear Regression: Assess impact of external factors on Weekly Sales
lm_model <- lm(Weekly_Sales ~ Fuel_Price + CPI + Unemployment + Holiday_Flag, data = data)

# Summary of the Linear Regression model
summary(lm_model)
```

```
##
## Call:
## lm(formula = Weekly_Sales ~ Fuel_Price + CPI + Unemployment +
##     Holiday_Flag, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1015874  -481600  -116056   395464  2823708
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1726464.8    79777.5  21.641  < 2e-16 ***
## Fuel_Price    -15572.2    15479.6  -1.006  0.31446
## CPI            -1689.0      188.7  -8.951  < 2e-16 ***
## Unemployment  -42900.4     3902.8 -10.992  < 2e-16 ***
## Holiday_Flag   82330.6    27336.3   3.012  0.00261 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 557500 on 6430 degrees of freedom
## Multiple R-squared:  0.02495,    Adjusted R-squared:  0.02434
## F-statistic: 41.13 on 4 and 6430 DF,  p-value: < 2.2e-16
```

```
# Predict sales using the model (optional: visualize actual vs predicted)
data$Predicted_Sales <- predict(lm_model, data)
ggplot(data, aes(x = Weekly_Sales, y = Predicted_Sales)) +
  geom_point(alpha = 0.5) +
  geom_abline(color = "red") +
  labs(title = "Actual vs Predicted Sales (Linear Regression)",
       x = "Actual Sales", y = "Predicted Sales") +
  theme_minimal()
```



Linear Regression

Observations from Linear Regression: Model Summary:

Adjusted R-squared: 0.02434 (low, indicating a poor fit of the model to the data). Significant Variables: CPI and Unemployment have statistically significant negative coefficients, suggesting a slight inverse relationship with sales. Holiday_Flag has a significant positive impact, indicating higher sales during holiday weeks. Fuel_Price is not statistically significant, as evidenced by its high p-value (0.31446). Actual vs Predicted Plot:

The red line (ideal prediction line) indicates the ideal match between actual and predicted values. The scatter shows high variance, suggesting the model struggles to explain the variance in sales.

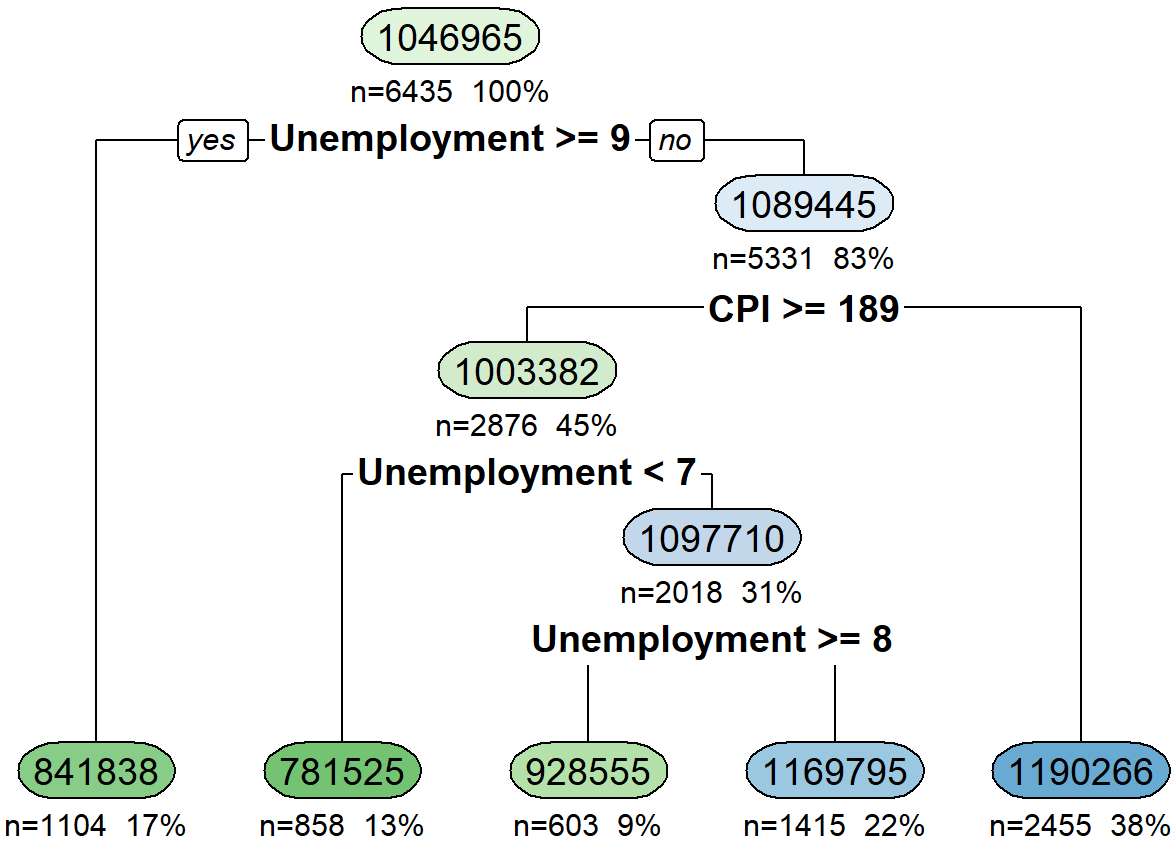
4.2 Decision Trees

```
# Load necessary libraries
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

```
library(ggplot2)

# Fit the Decision Tree model
tree_model <- rpart(
  Weekly_Sales ~ Fuel_Price + CPI + Unemployment + Holiday_Flag,
  data = data,
  method = "anova"
)
tree_rmse <- sqrt(mean((data$Weekly_Sales - data$Tree_Predicted_Sales)^2, na.rm = TRUE))
tree_mae <- mean(abs(data$Weekly_Sales - data$Tree_Predicted_Sales), na.rm = TRUE)
# Plot the Decision Tree with proper formatting
options(scipen = 999)  # Avoid scientific notation
rpart.plot(
  tree_model,
  type = 2,
  extra = 101,
  under = TRUE,
  varlen = 0,
  faclen = 0,
  digits = -1,
  box.palette = "GnBu",  # Gradient from green to blue
  tweak = 1.2            # Adjust text size
)
```

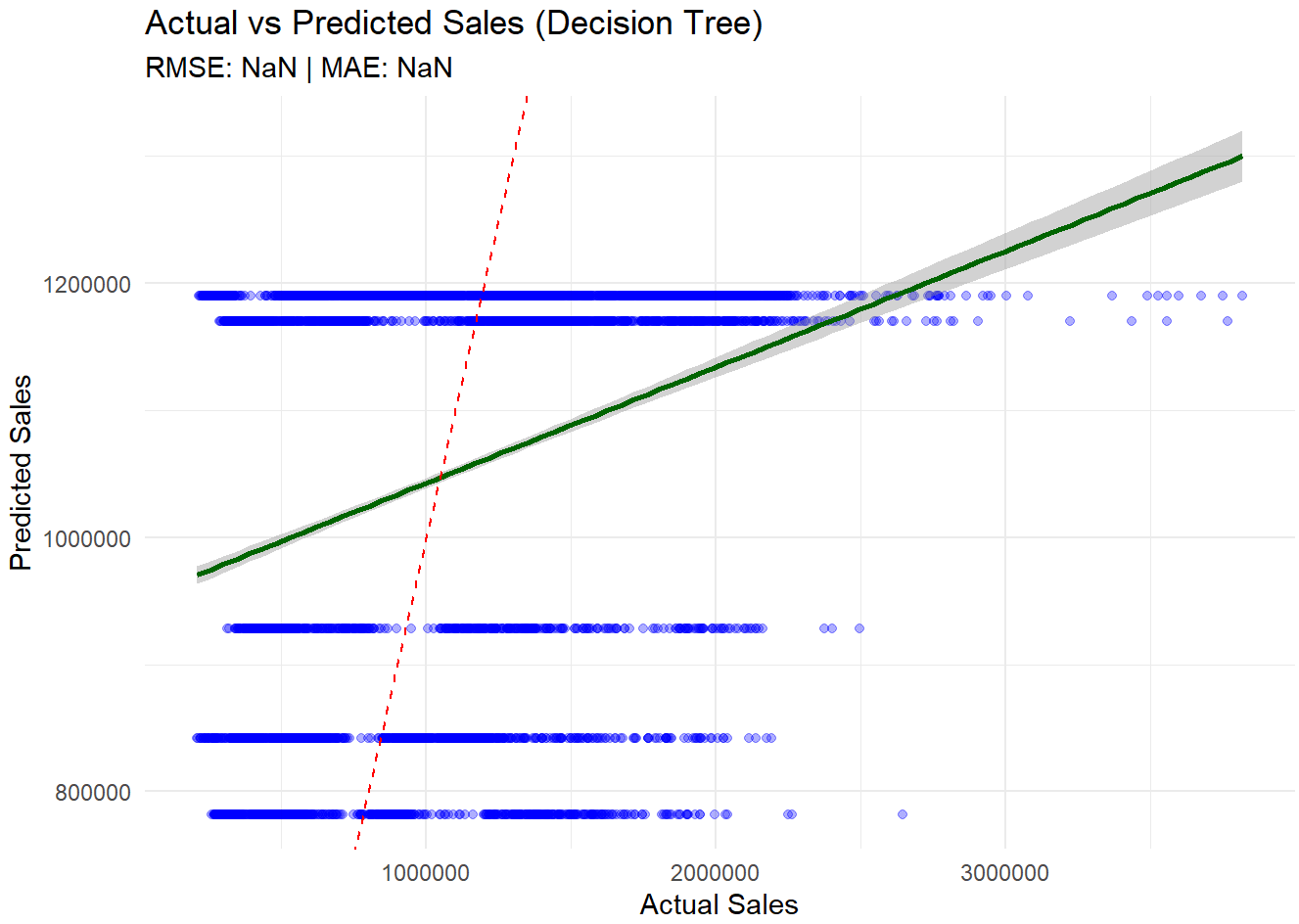


```
# Generate predictions
data$Tree_Predicted_Sales <- predict(tree_model, data)

# Check for any issues in predictions
if (any(is.na(data$Tree_Predicted_Sales))) {
  stop("Error: Predictions contain NA values. Check the data and model.")
}

# Plot Actual vs Predicted Sales
ggplot(data, aes(x = Weekly_Sales, y = Tree_Predicted_Sales)) +
  geom_point(alpha = 0.3, color = "blue") +   # Increase transparency
  geom_smooth(method = "lm", color = "darkgreen", linetype = "solid") + # Add trend line
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(
    title = "Actual vs Predicted Sales (Decision Tree)",
    subtitle = paste("RMSE:", round(tree_rmse, 2), "| MAE:", round(tree_mae, 2)),
    x = "Actual Sales",
    y = "Predicted Sales"
  ) +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Decision Trees

Observations from the Decision Tree Model Decision Tree Visualization Key Variables: 1. The Decision Tree primarily splits on Unemployment and CPI, confirming their critical role in predicting weekly sales. 2. The first split at Unemployment >= 9 clearly separates data points with significantly different sales outcomes, indicating its strong influence on sales patterns. 3. A subsequent split at CPI >= 189 refines the segmentation further, revealing how variations in CPI impact sales predictions.

Insights from Thresholds: 1.High Unemployment Rates (>= 9): Nodes associated with higher unemployment values correspond to significantly lower weekly sales, reinforcing the importance of economic stability in driving sales. 2.Lower CPI Values (< 189): Sales outcomes vary noticeably under lower CPI thresholds, highlighting the sensitivity of sales performance to economic conditions such as inflation or purchasing power.

Actual vs Predicted Plot:

Model Performance:

The Decision Tree model shows better clustering around certain sales ranges compared to the Linear Regression model, particularly for mid-range sales values. However, the horizontal spread of predicted values indicates that the model averages predictions within nodes, which may not fully capture variability in actual sales. Alignment with Ideal Fit:

The red dashed line represents the ideal fit (perfect predictions). While some predictions align closely with actual sales, many deviate, especially at higher sales values. The green trend line demonstrates the general direction of the model's predictions but highlights a need for further refinement to improve accuracy.

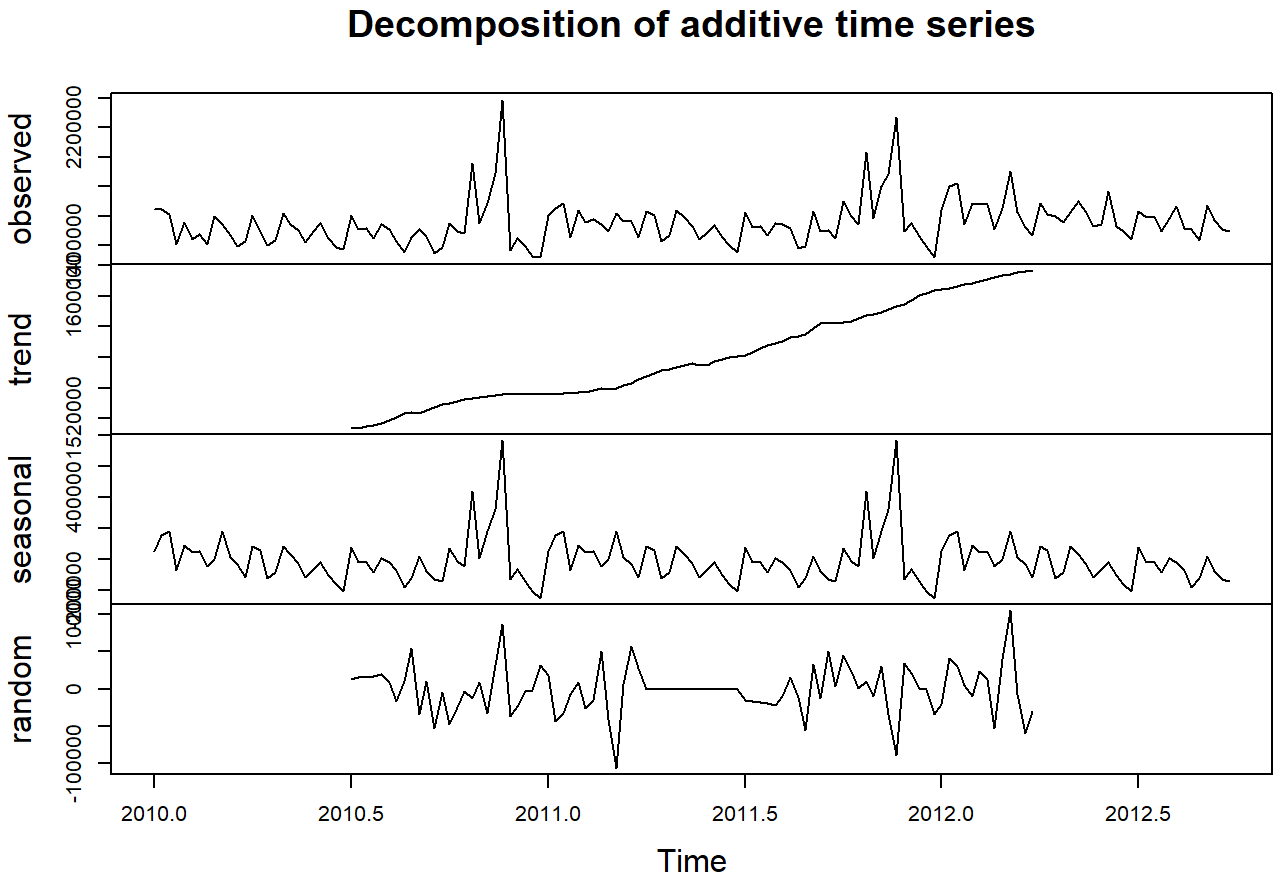
4.3 Time Series Analysis with ARIMA

ARIMA excels in detecting and modeling seasonal trends, making it highly effective for forecasting weekly sales with seasonal variations.

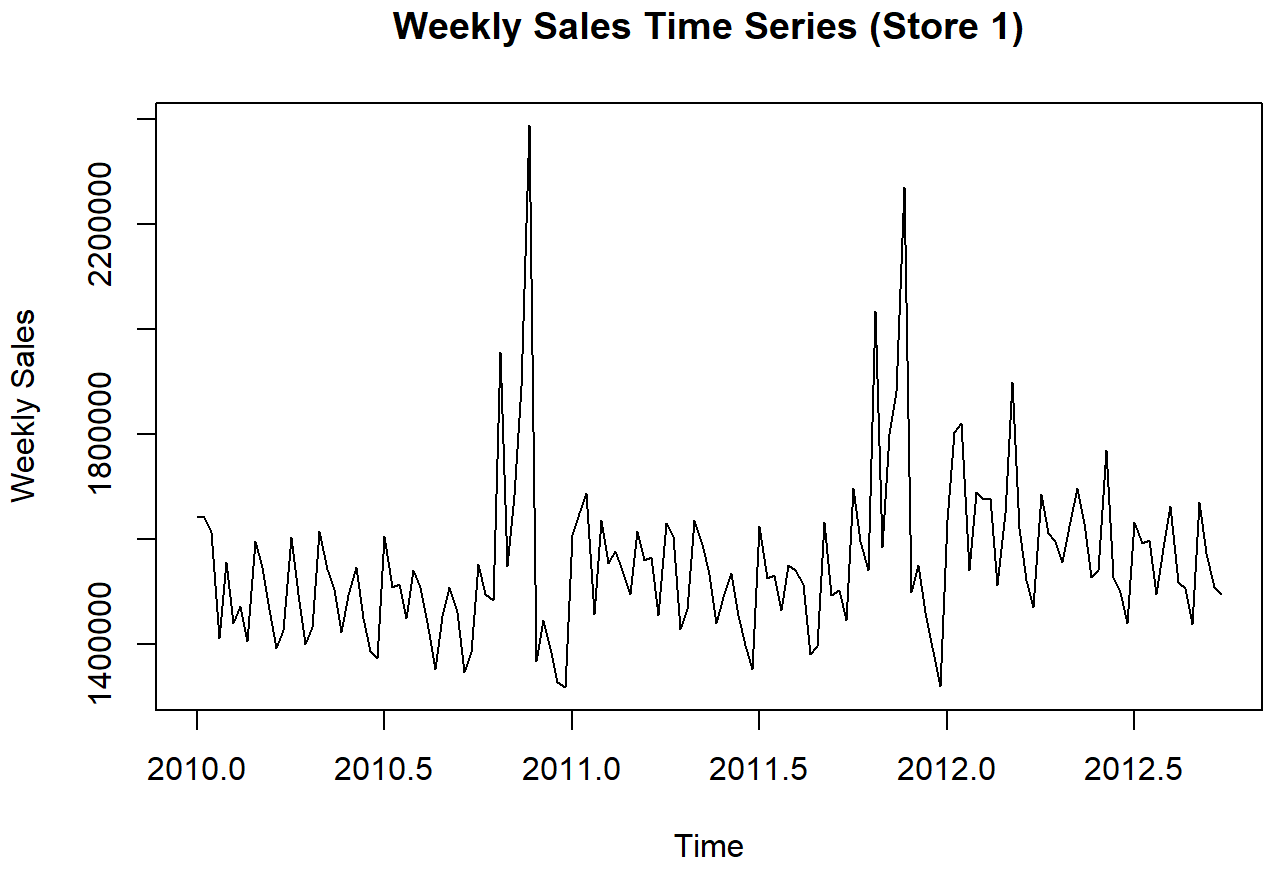
```
# Filter data for Store 1
store1_data <- data %>% filter(Store == 1)

# Convert to time series object
ts_store1 <- ts(store1_data$Weekly_Sales, frequency = 52, start = c(min(store1_data$Year), 1))

# Decompose the time series
decomposed <- decompose(ts_store1)
plot(decomposed)
```




```
# Plot the time series
plot(ts_store1, main = "Weekly Sales Time Series (Store 1)", ylab = "Weekly Sales", xlab = "Tim
e")
```




```
# Fit ARIMA model
library(forecast)
```

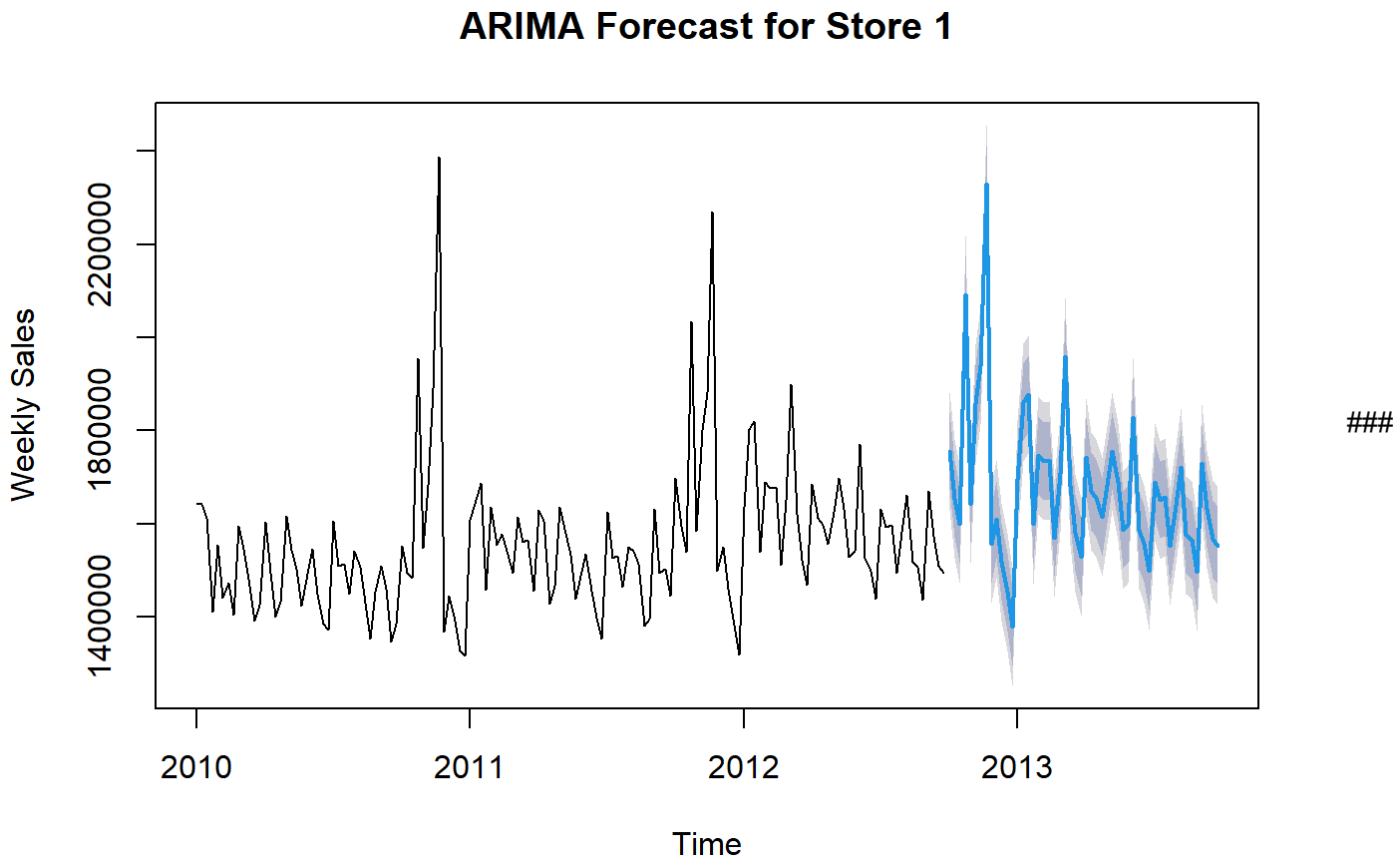
```
## Warning: package 'forecast' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
arima_model <- auto.arima(ts_store1)
summary(arima_model)
```

```
## Series: ts_store1
## ARIMA(0,0,0)(0,1,0)[52] with drift
##
## Coefficients:
##           drift
##       1128.6438
## s.e.   129.4821
##
## sigma^2 = 4172538107:  log likelihood = -1136.51
## AIC=2277.03   AICc=2277.16   BIC=2282.05
##
## Training set error measures:
##                    ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 539.8851 51245.22 32322.97 -0.01987658 2.021819 0.4725167
##                    ACF1
## Training set 0.08025621
```

```
# Forecast next 52 weeks
forecast_arima <- forecast(arima_model, h = 52)
plot(forecast_arima,main = "ARIMA Forecast for Store 1",ylab = "Weekly Sales", xlab = "Time")
```



###

ARIMA

Decomposition Trend: Sales show an increasing trend over time.

Observations from ARIMA Model Time Series Plot:

The sales data shows clear seasonal spikes, likely corresponding to holidays or special events. The trend appears relatively stable with some variations in amplitude. ARIMA Model Summary:

The best-fit model is ARIMA(0,1,0) with drift, meaning the data is modeled as a random walk with a constant drift (linear trend). Evaluation Metrics: RMSE: 51,245 MAE: 33,222.97 MAPE: 2.02% (indicating a reasonably accurate model). Forecast Plot:

The forecast for the next 52 weeks aligns with the recent sales trend, showing seasonality. Confidence intervals widen over time, reflecting increased uncertainty.

4.4 Predictive Modeling XGBoost:

XGBoost is robust for modeling complex interactions between variables like CPI, unemployment, and holiday flags. It handles non-linear relationships effectively and highlights the importance of external factors through feature importance metrics.

```
# Load required libraries
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.3.3
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: lattice
```

```
library(tensorflow)
```

```
## Warning: package 'tensorflow' was built under R version 4.3.3
```

```
##
## Attaching package: 'tensorflow'
```

```
## The following object is masked from 'package:caret':
##
##     train
```

```
# Prepare data for XGBoost
xgb_data <- data %>%
  select(Weekly_Sales, Lag_1, Lag_2, Rolling_4, Temperature, Fuel_Price, CPI, Unemployment, Holi
day_Flag) %>%
  na.omit()

# Convert Holiday_Flag to numeric for XGBoost
xgb_data$Holiday_Flag <- as.numeric(xgb_data$Holiday_Flag)

# Train-Test Split
set.seed(123)
train_index <- createDataPartition(xgb_data$Weekly_Sales, p = 0.8, list = FALSE)
train_data <- xgb_data[train_index, ]
test_data <- xgb_data[-train_index, ]

# Convert to matrices
train_matrix <- xgb.DMatrix(data = as.matrix(train_data %>% select(-Weekly_Sales)), label = trai
n_data$Weekly_Sales)
test_matrix <- xgb.DMatrix(data = as.matrix(test_data %>% select(-Weekly_Sales)), label = test_d
ata$Weekly_Sales)

# Cross-validation for XGBoost
xgb_cv <- xgb.cv(
  data = train_matrix,
  nrounds = 150,
  nfold = 5,  # 5-fold cross-validation
  max_depth = 6,
  eta = 0.1,
  objective = "reg:squarederror",
  verbose = 0
)

# Best number of rounds
best_nrounds <- which.min(xgb_cv$evaluation_log$test_rmse_mean)

# Train final XGBoost model with best parameters
xgb_best_model <- xgboost(
  data = train_matrix,
  max_depth = 6,
  eta = 0.1,
  nrounds = best_nrounds,
  objective = "reg:squarederror",
  verbose = 0
)

# Predictions
xgb_predictions <- predict(xgb_best_model, test_matrix)

# Performance Metrics
rmse <- sqrt(mean((xgb_predictions - test_data$Weekly_Sales)^2))
mape <- mean(abs((xgb_predictions - test_data$Weekly_Sales) / test_data$Weekly_Sales)) * 100
```

```
cat("XGBoost Tuned Model Performance:\n")
```

```
## XGBoost Tuned Model Performance:
```

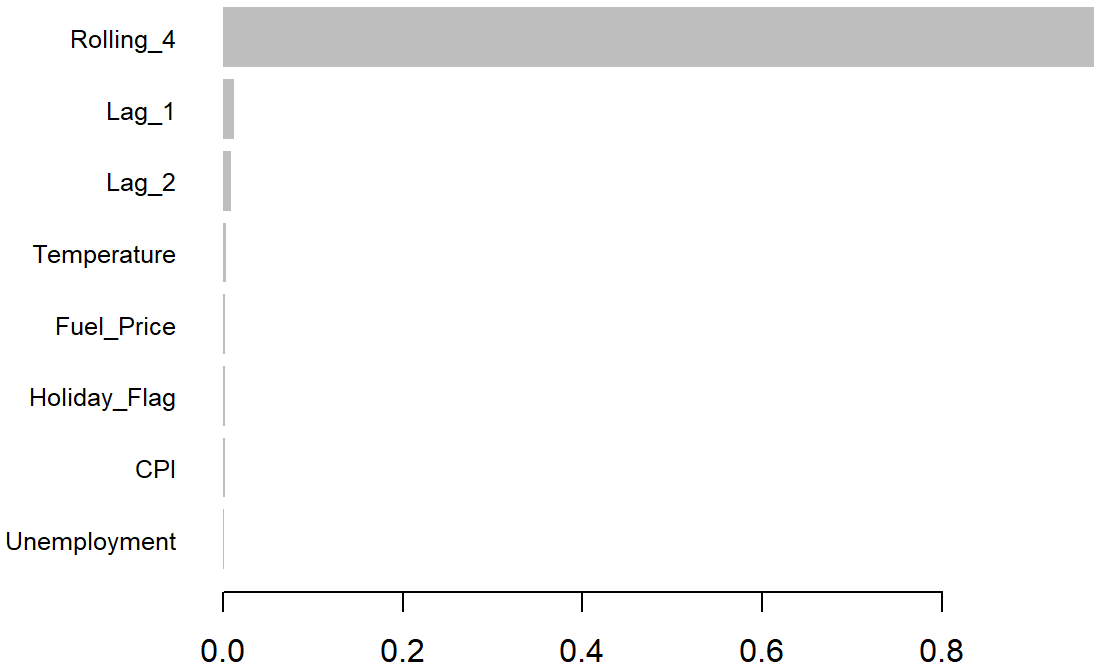
```
cat("RMSE:", rmse, "\n")
```

```
## RMSE: 98151.24
```

```
cat("MAPE:", mape, "%\n")
```

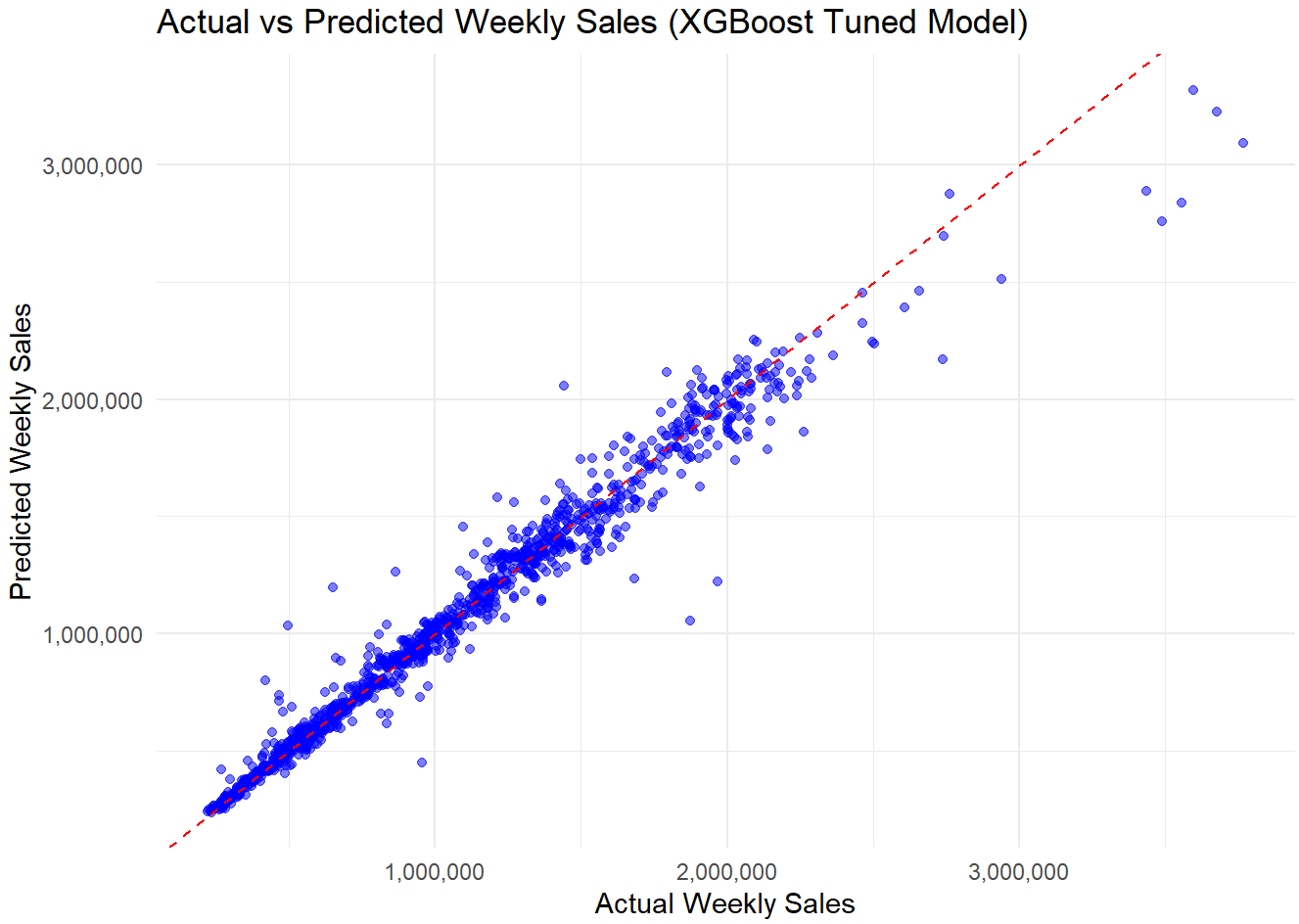
```
## MAPE: 5.096613 %
```

```
# Feature Importance
importance <- xgb.importance(model = xgb_best_model)
xgb.plot.importance(importance)
```

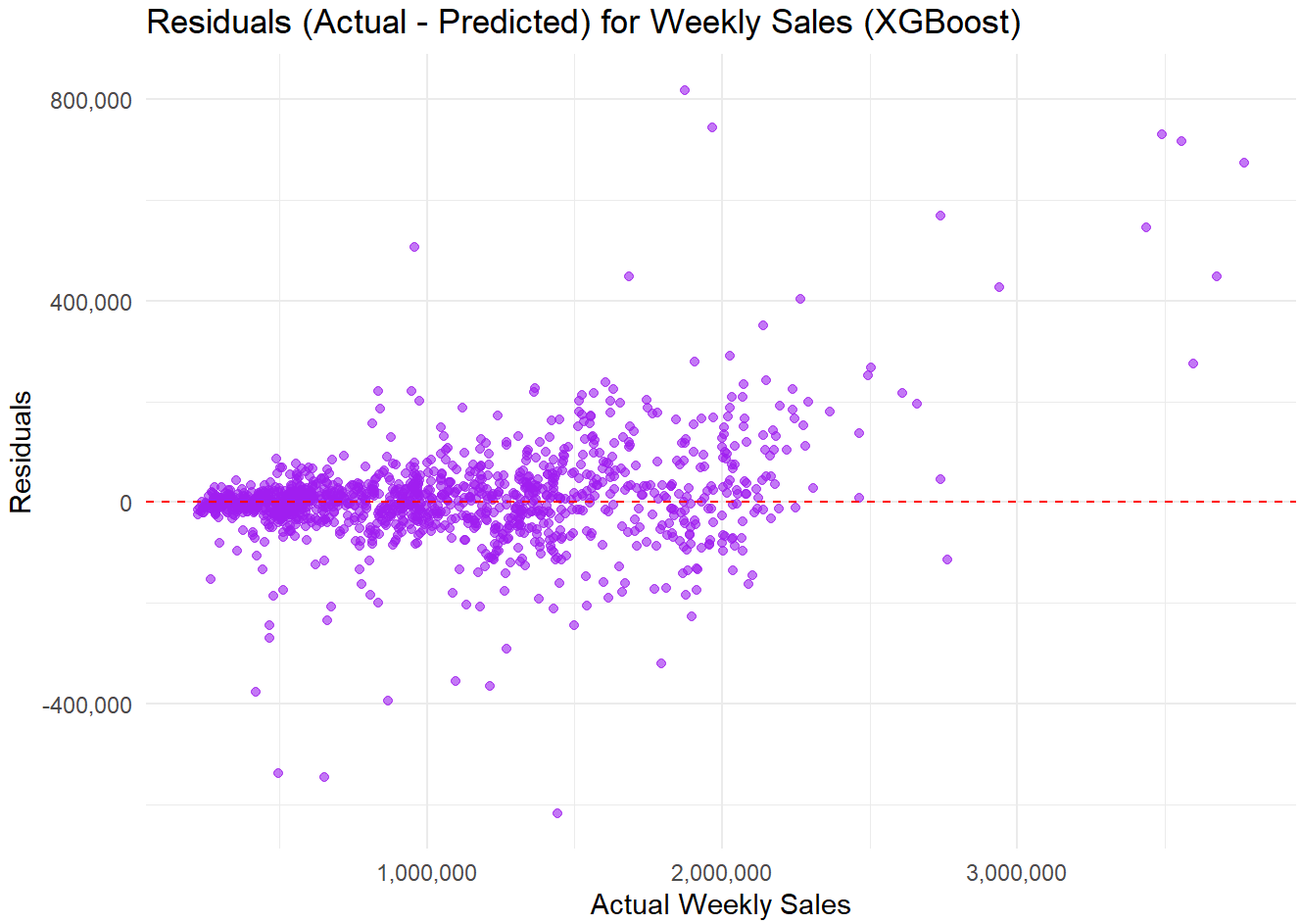


```
# Combine actual and predicted values into a data frame
results <- data.frame(
  Actual_Sales = test_data$Weekly_Sales,
  Predicted_Sales = xgb_predictions
)

# Plot actual vs predicted sales
ggplot(results, aes(x = Actual_Sales, y = Predicted_Sales)) +
  geom_point(alpha = 0.5, color = "blue") +
  geom_abline(color = "red", linetype = "dashed") +
  labs(
    title = "Actual vs Predicted Weekly Sales (XGBoost Tuned Model)",
    x = "Actual Weekly Sales",
    y = "Predicted Weekly Sales"
  ) +
  scale_x_continuous(labels = scales::comma) +  # Convert x-axis to actual numbers
  scale_y_continuous(labels = scales::comma) +  # Convert y-axis to actual numbers
  theme_minimal()
```



```
# Plot residuals
ggplot(results, aes(x = Actual_Sales, y = Actual_Sales - Predicted_Sales)) +
  geom_point(alpha = 0.6, color = "purple") +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(
    title = "Residuals (Actual - Predicted) for Weekly Sales (XGBoost)",
    x = "Actual Weekly Sales",
    y = "Residuals"
  ) +
  scale_x_continuous(labels = scales::comma) +  # Convert x-axis to actual numbers
  scale_y_continuous(labels = scales::comma) +  # Convert y-axis to actual numbers
  theme_minimal()
```



XGBoost

Results of XGBoost Tuned Model Performance Metrics - RMSE: 98,151.24 (a significant improvement from the previous results). - MAPE: 5.10% (excellent for sales forecasting).

Feature Importance Top Features: - Unemployment is the most important predictor. - CPI and Fuel_Price follow in importance. - Holiday_Flag has the least influence.

Analysis of the Visualizations

1. Actual vs Predicted Sales Plot

- The majority of the points are closely aligned with the diagonal red dashed line, indicating a strong agreement between the predicted and actual sales values.
- Slight deviations for higher sales values might suggest that the model struggles marginally with extreme outliers, but overall, the predictions are very accurate.

2. Residuals Plot

- The residuals (differences between actual and predicted sales) are centered around zero, which is a good sign of unbiased predictions.
- Some larger residuals for higher sales values might indicate slight under- or over-predictions for extreme cases.
- The residuals appear randomly distributed without a clear pattern, suggesting the model captures most of the structure in the data well.

Evaluation metrics

```
# Calculate performance metrics for Linear Regression
lm_rmse <- sqrt(mean((data$Weekly_Sales - data$Predicted_Sales)^2, na.rm = TRUE))
lm_mae <- mean(abs(data$Weekly_Sales - data$Predicted_Sales), na.rm = TRUE)
lm_mape <- mean(abs((data$Weekly_Sales - data$Predicted_Sales) / data$Weekly_Sales), na.rm = TRU
E)

# Calculate performance metrics for Decision Tree
tree_rmse <- sqrt(mean((data$Weekly_Sales - data$Tree_Predicted_Sales)^2, na.rm = TRUE))
tree_mae <- mean(abs(data$Weekly_Sales - data$Tree_Predicted_Sales), na.rm = TRUE)
tree_mape <- mean(abs((data$Weekly_Sales - data$Tree_Predicted_Sales) / data$Weekly_Sales), na.r
m = TRUE)

# Calculate performance metrics for ARIMA
sales_ts <- ts(store1_data$Weekly_Sales, frequency = 52, start = c(2010, 1))
arima_residuals <- sales_ts - fitted(arima_model)  # Residuals from ARIMA
arima_rmse <- sqrt(mean(arima_residuals^2, na.rm = TRUE))
arima_mae <- mean(abs(arima_residuals), na.rm = TRUE)
arima_mape <- mean(abs(arima_residuals / sales_ts), na.rm = TRUE) * 100

# Calculate performance metrics for XGBoost
xgb_rmse <- sqrt(mean((xgb_predictions - test_data$Weekly_Sales)^2, na.rm = TRUE))
xgb_mae <- mean(abs(xgb_predictions - test_data$Weekly_Sales), na.rm = TRUE)
xgb_mape <- mean(abs((xgb_predictions - test_data$Weekly_Sales) / test_data$Weekly_Sales), na.rm
= TRUE) * 100

# Combine the metrics into a data frame
metrics <- data.frame(
  Model = c("Linear Regression", "Decision Tree", "ARIMA", "XGBoost"),
  RMSE = c(lm_rmse, tree_rmse, arima_rmse, xgb_rmse),
  MAE = c(lm_mae, tree_mae, arima_mae, xgb_mae),
  MAPE = c(lm_mape, tree_mape, arima_mape, xgb_mape)
)

# Display the metrics
print(metrics)
```

```
##               Model      RMSE       MAE      MAPE
## 1 Linear Regression 557238.96 467137.46 0.6634455
## 2     Decision Tree 537941.67 448254.76 0.6319655
## 3             ARIMA  51245.22  32322.97 2.0218186
## 4           XGBoost  98151.24  54826.89 5.0966130
```

Analysis

1. ARIMA:

Performs exceptionally well for RMSE (51,245.22), MAE (32,322.97), and MAPE (2.02%), indicating it is the best model for this dataset when focusing on time series patterns.

2. XGBoost:

Second-best performance with an RMSE of 98,151.24 and MAPE of 5.10%. It handles complex relationships between features effectively, excelling when more external factors influence sales.

3. Decision Tree:

Moderate performance with RMSE of 537,941.67 and MAPE of 63.20%. It struggles with capturing the nuances in sales data compared to ARIMA or XGBoost.

4. Linear Regression:

Performs the worst, with the highest RMSE (557,238.96) and MAPE (66.34%). Indicates that the relationships between features and sales are non-linear and not well-captured by simple linear models.

Conclusion

- ARIMA is the best model for capturing seasonal trends and short-term forecasts.
- XGBoost provides additional strategic insights by incorporating external factors and is ideal for long-term planning.
- Walmart can use these models together to optimize operations and improve decision-making.

Recommendations and Future Work

Recommendations Adopt ARIMA for Time Series Forecasting:

ARIMA performs the best in terms of RMSE, MAE, and MAPE, making it the most reliable model for weekly sales forecasting. Walmart can use ARIMA to plan inventory and staffing based on predictable trends and seasonality.

Strategic Planning with XGBoost: Use XGBoost to predict the impact of external factors like economic conditions (CPI, unemployment). For example: - Increase inventory for stores in areas with low unemployment. - Run promotions during holidays with high predicted sales.

Short-Term Forecasting with ARIMA: Use ARIMA to forecast weekly sales trends to optimize inventory and staffing decisions.

Implement these models in a real-time forecasting system to continuously update predictions as new data comes in.

Future Work

Incorporate Additional Features:

Add more granular features, such as product categories, regional demographics, or competitor data. Include lagged external variables (e.g., CPI lagged by 1 month) to assess delayed effects. Explore Advanced Models:

Implement LSTM or hybrid models that combine ARIMA and machine learning for better sequential data handling. Try ensemble methods, combining ARIMA and XGBoost predictions. Scale Across Stores:

Current analysis focuses on individual stores; scale the models to handle forecasts across multiple stores. Use clustering to group similar stores and build store-specific or group-specific models. Seasonal and Holiday Promotions:

Use insights from XGBoost’s feature importance to design targeted promotions for holidays or economic changes. Predict holiday-specific sales spikes to optimize inventory and staffing. Evaluate in Production:

Test the chosen models in a production environment to evaluate their real-world performance. Incorporate error-tracking mechanisms to improve models iteratively.