Developing Time Series Models to Predict Auto Sales

Maxwell Smith, Jabari Rose, Lyan Nhemachena, Tsitsi Njovana

1. Introduction

Goal: The purpose of this study is to develop time series models to accurately predict auto sales using trend(s), seasonality, and other economic factors.

Trend: The auto sales in the US shows a negative trend from 1995 to 2020, but can be separated into three time segments for better forecasting: January 1995 - October 2008, November 2008 - May 2014, and June 2014 - December 2020 (Figure 1.1).

Seasonality: Auto Sales are typically higher in spring and summer months, compared to fall and winter months (Figure 1.2).

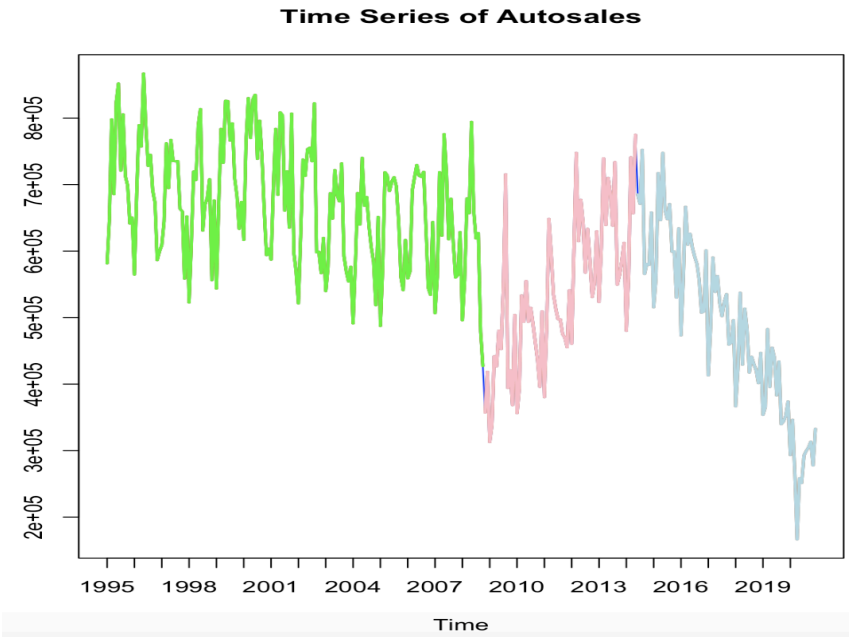


Figure 1.1

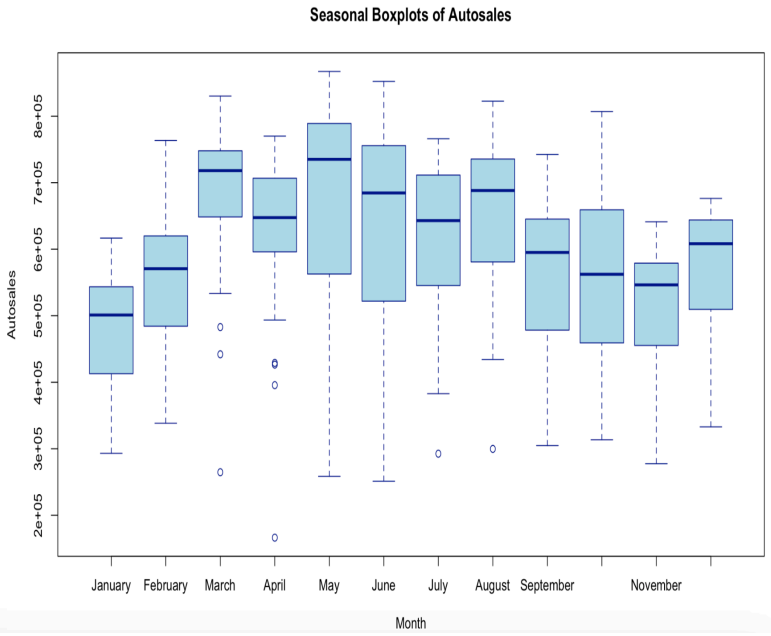


Figure 1.2

Data Background: The US auto sales dataset consists of 312 observations with seven variables. The first 288 observations were used as training (1995 - 2018) and the last 24 as holdout (2019 - 2020). The target variable is sales, with additional variables including year-to-year inflation, monthly unemployment, Producer Price Index (PPI) for iron and steel, and regular gas prices.

2. Univariate Time Series Models

2.1 Deterministic Time Series Models

The models were fit using a holdout size of the last 24 observations. The Cyclical Trend model performed the best on the train data (figure 2.1.1). However, the Trend model performed slightly better on the holdout set based on MAPE (figure 2.1.2), so that was chosen for further analysis.

Train Performance

Model	R-Squared	MAPE	RMSE	MAE
Trend	**.5358**	**10.62%**	**77858.92**	**63529.08**
Monthly + Trend	**.8541**	**5.54%**	**43644.71**	**33351.61**
Cyclical + Trend	**.8807**	**4.87%**	**39472.38**	**29299.77**

Figure 2.1.1

Holdout Performance

Model	MAPE	RMSE	MAE
Trend	**17.97 %**	**63560.05**	**50221.29**
Monthly + Trend	**21.84 %**	**83798.2**	**58009.38**
Cyclical + Trend	**19.48 %**	**69657.28**	**54442.87**

Figure 2.1.2

Trend Model Analysis

The projected auto sales in January 1995 is ~$728K. Until November 2008, it is expected to decrease by $701 a month. In November 2008 the projected auto sales are expected to decrease by an additional ~$327K. However, beginning this month, and lasting until May 2014, the expected auto sales are expected to increase by $4687 a month. Beginning June 2014, auto sales projections decreased by ~50K and an additional $3872 a month until the end of 2020 (figure 2.1.3).

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  728568.3    12269.3  59.381  < 2e-16 ***
GTrend         -701.0      127.4  -5.500 8.50e-08 ***
D1          -327497.7    22990.2 -14.245  < 2e-16 ***
D2           -49547.3    24764.8  -2.001   0.0464 *
GTrend:D1      4687.2      513.1   9.134  < 2e-16 ***
GTrend:D2     -3872.2      680.4  -5.691 3.16e-08 ***
```

Figure 2.1.3

Actual vs. Predicted Values

The predicted values show how predictions change with the three time segments (Figure 2.1.4).

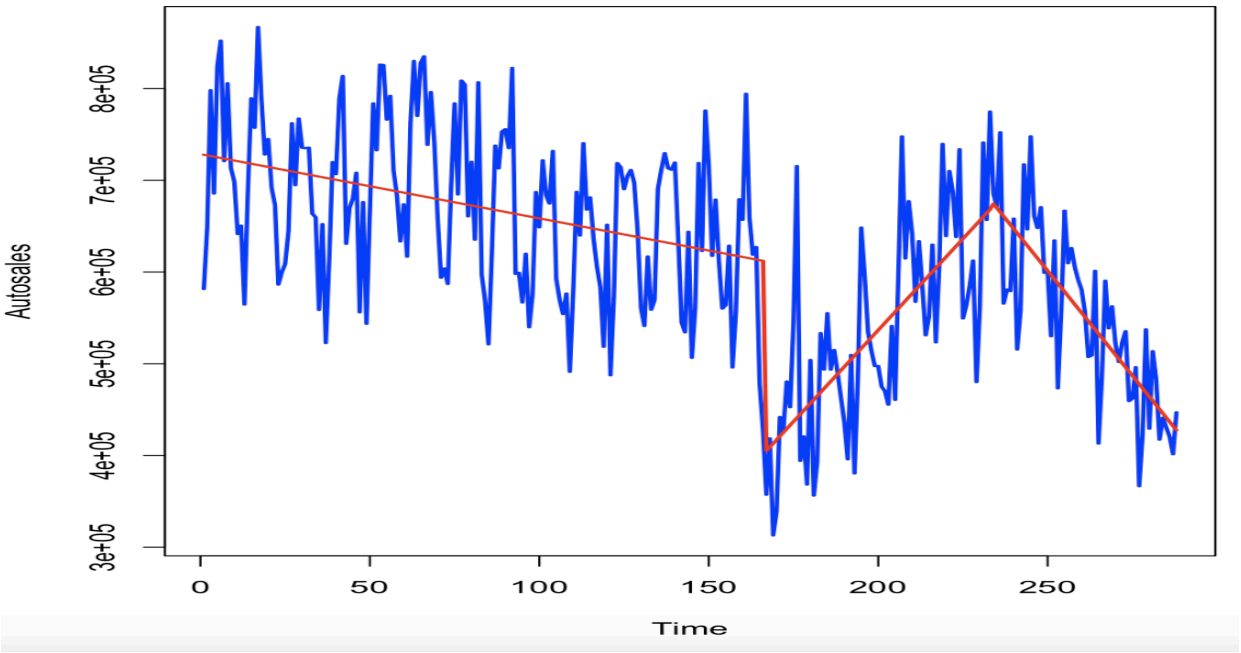


Figure 2.1.4

ACF Chart

The cyclical trends in ACF chart (Figure 2.1.5) and p-value from Box-Pierce test (Figure 2.1.6), we can conclude that the residuals are not white noise and non-stationary.

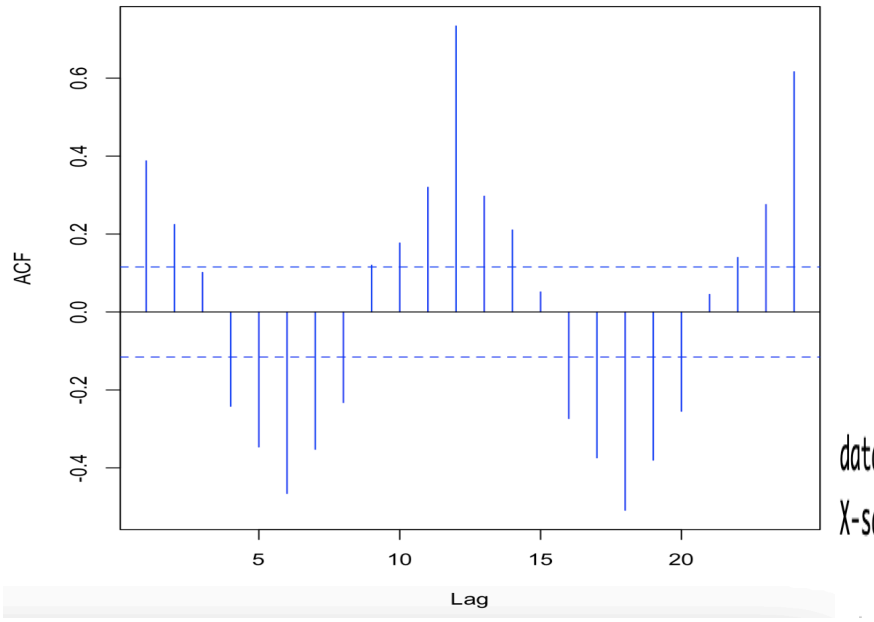


Figure 2.1.5

```
Box-Pierce test

data:  residuals_model
X-squared = 797.02, df = 24, p-value < 2.2e-16
```

Figure 2.1.6

Periodogram (Cyclical + Trend Model)

Highest Amplitude Periods: 12, 3, 2.4, 4, 72, 11.52, 6, 3.95 (Figure 2.1.7) and (Figure 2.1.8).

Period 12 displays by far the highest amplitude demonstrating seasonality and a strong annual cycle (figure 2.1.7). While other seasonality also exists, the trend model is capturing the most important factor in the relationship between auto sales and time/seasonality. The cyclical trend model could potentially be introducing noise to the predictions in the holdout set. While using slightly less periods was tested, the results were not as strong as when all eight were used.

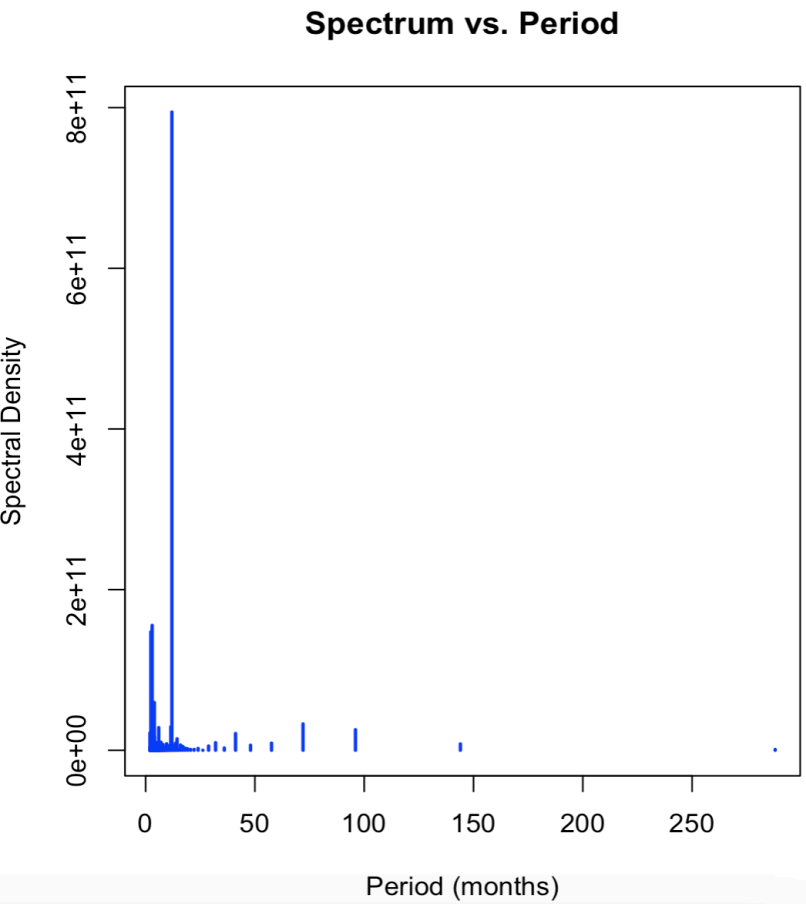


Figure 2.1.7

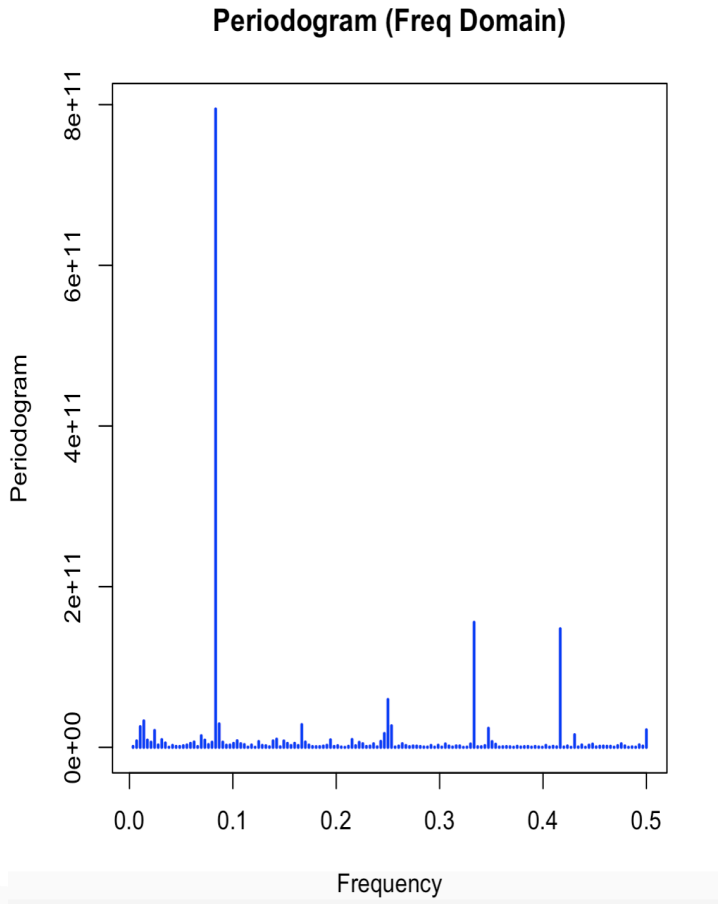


Figure 2.1.8

2.2 Exponential Smoothing Models

The following models were tested on the data with the Holt-Winter having the strongest MAPE across the train data (figure 2.2.1), but it also performed the worst on the holdout data. The Simple Exponential Smoothing (SES) model had the strongest performance on the holdout data (figure 2.2.2).

Train Performance

Model	MAPE	RMSE	MAE
SES	10.54%	79009.2	62819.89
Holt-Linear	10.73%	79878.62	63739.25
Holt-Winter	5.75%	43903.23	33744.48

Figure 2.2.1

Holdout Performance

Model	MAPE	RMSE	MAE
SES	14.33%	53756.38	41946.4
Holt-Linear	14.43%	54062.77	43463.73
Holt-Winter	18.08%	71621.8	50288.78

Figure 2.2.2

SES Model Output and Actual vs. Predicted Values

The SES model (Figure 2.2.4) does a better job than the trend and cyclical trend models by factoring in seasonality and trends without overfitting in the holdout sample (Figure 2.2.3).

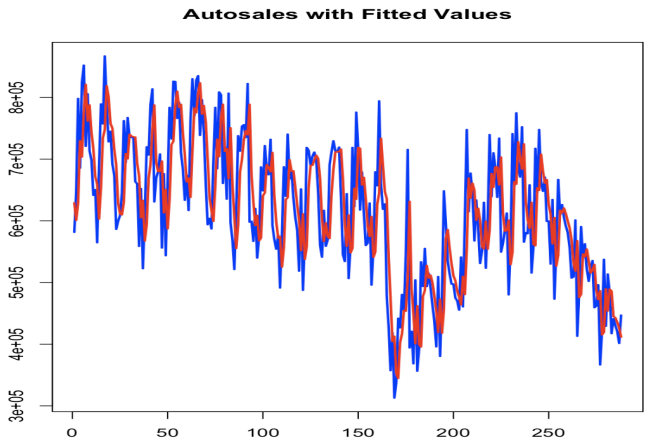


Figure 2.2.3

SES Model Output	
Alpha	.595
Initial State	628921.71
Sigma	79284.98

Figure 2.2.4

3. Time Series Regression Models

3.1 Multiple Linear Regression (MLR) - Adding Economic Variables

Inflation, PPI, Gas prices, Unemployment rate.

Correlation Analysis - Trend Correlation by Segment

The table displays the correlation between trend and autosales for the three segments. The last one sees the strongest correlation at -.901 (figure 3.1.1).

Segment 1 (1995 - 2008)	Segment 2 (2008 - 2014)	Segment 3 (2014 - 2020)
-.374	.698	-.901

Figure 3.1.1

Correlation Analysis - Economic Variables (All Time Segments)

Negative Correlation with Auto Sales: Trend, PPI, Unemployment, Gas Prices (figure 3.1.2).
Positive Correlation with Auto Sales: Inflation (figure 3.1.2).

Correlation Matrix

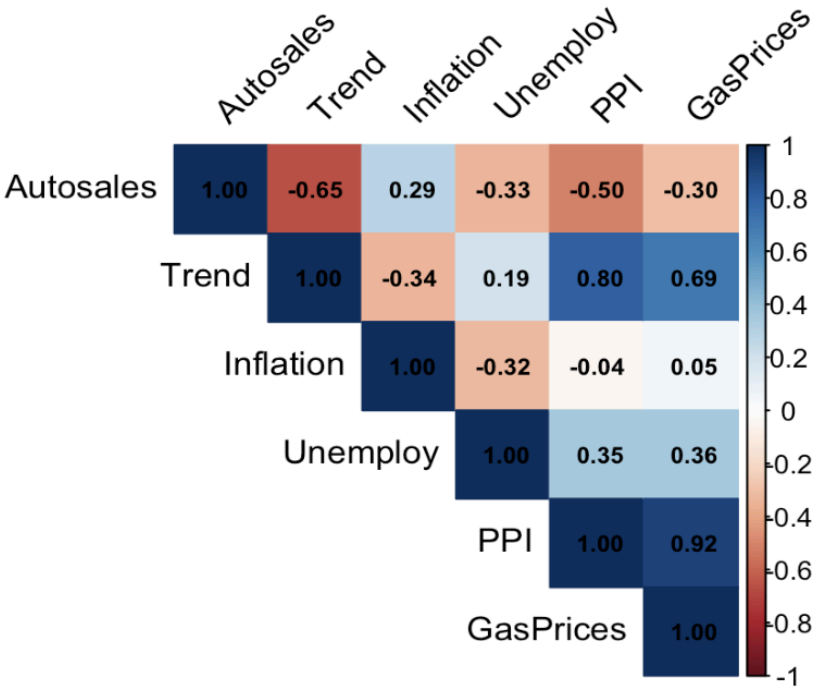


Figure 3.1.2

Scatter Plots (vs. Auto Sales)

The following plots (figure 3.1.3 - figure 3.1.6) shows the relationship between auto sales and each of the economic variables added, as well the trend for the whole period.

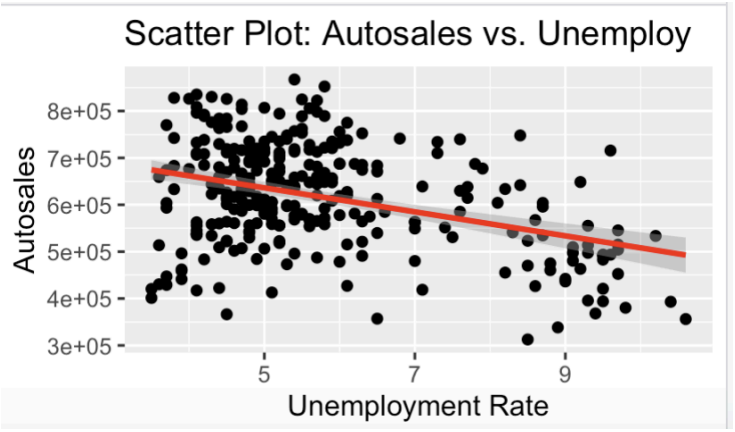


Figure 3.1.3

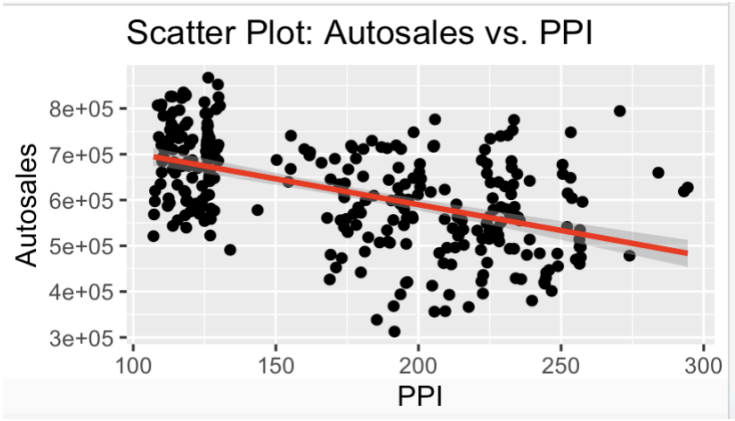


Figure 3.1.4

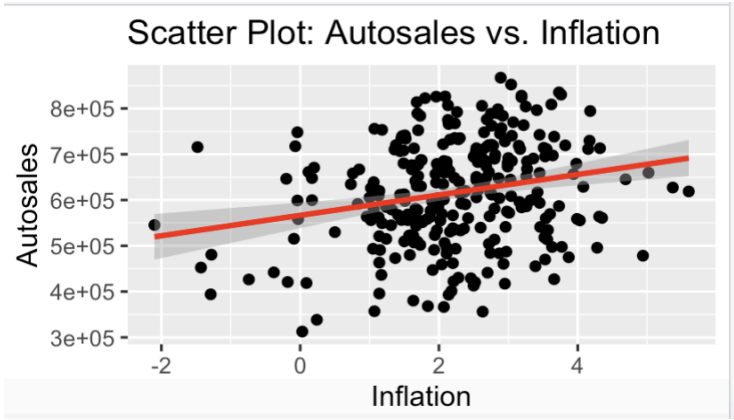


Figure 3.1.5

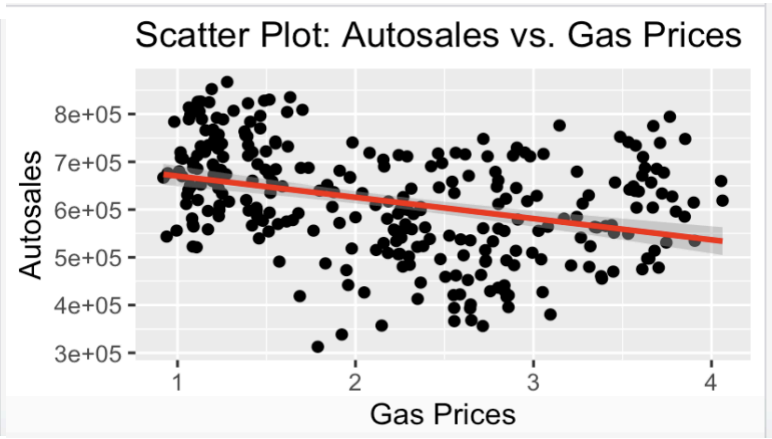


Figure 3.1.6

3.2 Model Comparison (MLR - Economic Variables Added)

The models were now trained adding the economic variables described above. The Trend MLR model used only our trends and the new variables, while the trend + seasonal model included those terms, but also added the top six cyclical trends we identified from the periodogram. The six terms were used here (instead of eight) due to adding the economic variables, which added complexity to the model. Like we saw previously, the seasonal model outperforms the trend model on the training data (figure 3.2.1). However, this time the seasonal model also performed the best on the holdout data, suggesting that the economic factors were important for making predictions on the holdout set (figure 3.2.2).

Training Performance

Model	R-Squared	MAPE	RMSE	MAE
Trend MLR	**.6145**	**9.96%**	**70953.21**	**59426.14**
Trend + Cyclical MLR	**.8817**	**4.81%**	**39300.36**	**28898.97**

Figure 3.2.1

Holdout Performance

Model	MAPE	RMSE	MAE
Trend MLR	**15.44%**	**67076.65**	**51934.29**
Trend + Cyclical MLR	**13.29%**	**55928.77**	**42275.02**

Figure 3.2.2

Actual vs. Predicted Values (Trend + Cyclical MLR Model)

The actual vs predicted plot, with economic variables considered (Figure 3.2.3).

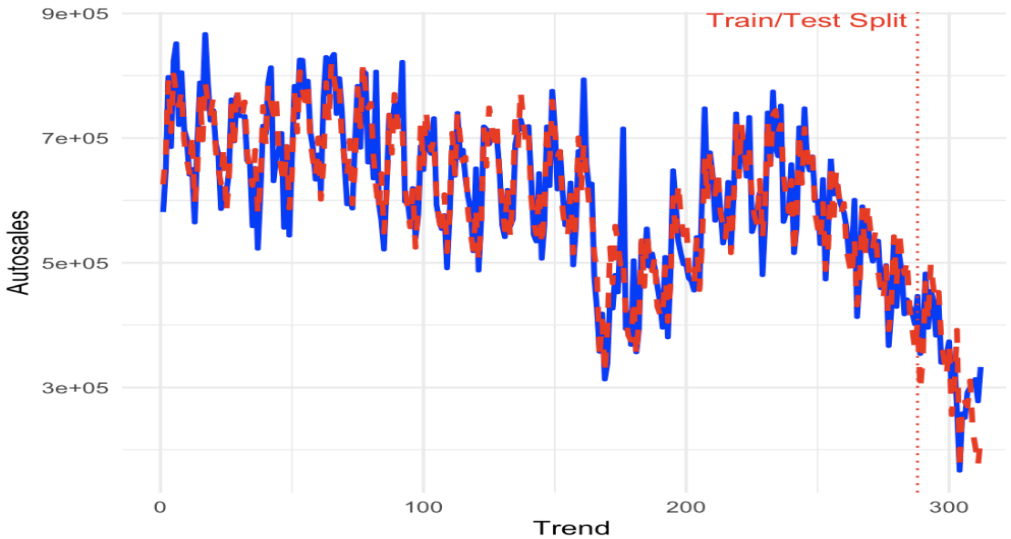


Figure 3.2.3

Trend + Cyclical Model

The model below shows how seasonality and trend remain strong predictors of auto sales, even with economic variables considered. Among the variables added, PPI has the lowest p-value, while Inflation is only one that is insignificant (Figure 3.2.4). As a whole, the trend terms and the three highest periods (12, 3, 2.4) are contributing the most to the model's predictions.

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	851834.3	22735.7	37.467	< 2e-16	***
GTrend	-877.8	141.6	-6.199	2.16e-09	***
D1	-259219.1	32978.6	-7.860	9.57e-14	***
D2	-49124.2	30699.2	-1.600	0.110746	
Inflation	-5524.1	4060.6	-1.360	0.174854	
Unemploy	-13154.8	4179.6	-3.147	0.001835	**
PPI	-813.6	183.7	-4.429	1.38e-05	***
GasPrices	52599.9	15789.3	3.331	0.000987	***
cos_1	-68235.6	3968.6	-17.194	< 2e-16	***
cos_2	24134.6	3410.1	7.077	1.30e-11	***
cos_3	18642.6	3426.9	5.440	1.21e-07	***
cos_4	14686.2	3469.0	4.234	3.17e-05	***
cos_5	14162.1	4964.2	2.853	0.004674	**
cos_6	7770.1	3441.0	2.258	0.024750	*
sin_1	18456.0	3521.9	5.240	3.27e-07	***
sin_2	-20833.1	3419.2	-6.093	3.87e-09	***
sin_3	26317.2	3413.5	7.710	2.52e-13	***
sin_4	-12521.8	3411.7	-3.670	0.000293	***
sin_5	-8219.7	4079.9	-2.015	0.044946	*
sin_6	-11997.4	3488.3	-3.439	0.000677	***
GTrend:D1	3272.0	337.2	9.704	< 2e-16	***
GTrend:D2	-4148.6	534.5	-7.762	1.80e-13	***

Figure 3.2.4

Stochastic Time Series Models

4.1 ACF Charts

Original Auto Sales Time Series

The ACF chart for Auto Sales exhibits a slow decay outside the bounds indicating this series is not stationary, so differencing will need to be applied. The spikes are also a sign of seasonality (Figure 4.1.1).

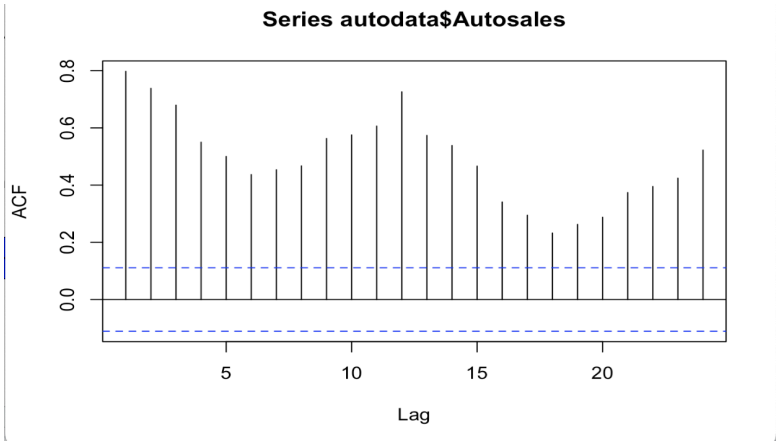


Figure 4.1.1

PACF and ACF of Differentiated Series

The ACF (Figure 4.1.2) of the differenced series shows a gradual decay, which is characteristic of an AR process. However, since it also drops off after lag 1, an MA(1) model was initially considered. The PACF, on the other hand, (Figure 4.1.3) cuts off after lag 1, supporting the possibility of an AR(1) or AR(2) process. In addition, the ACF displays significant spikes at multiples of 12 lags, suggesting the presence of seasonality. As a result, seasonal differencing (lag 12) was applied to the series in an effort to achieve stationarity.

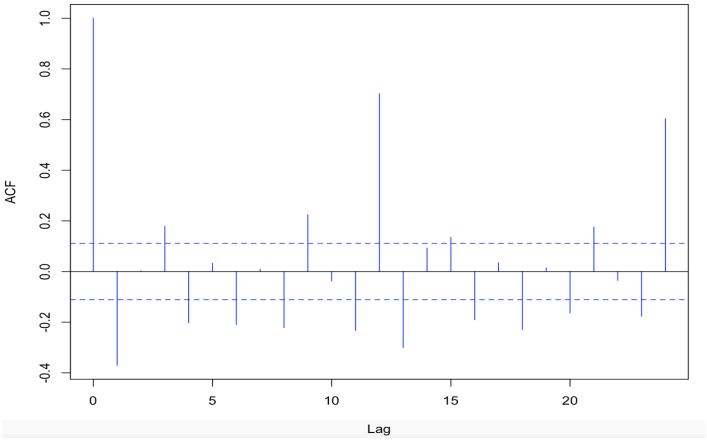


Figure 4.1.2

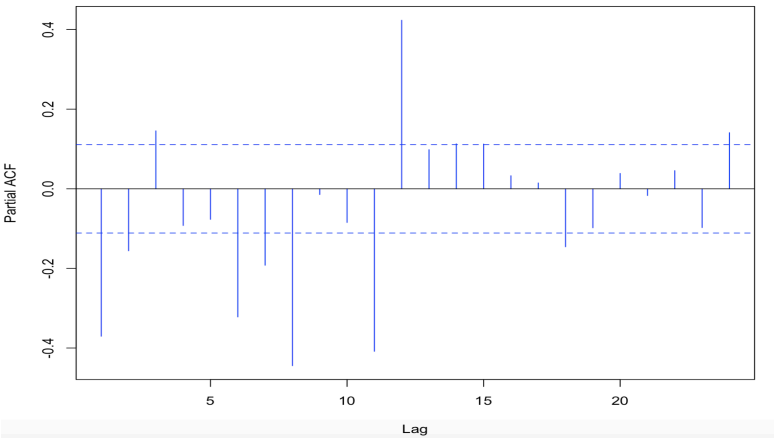


Figure 4.1.3

PACF and ACF of Seasonal Differentiated Series

To account for seasonality and non-stationarity, regular and seasonal differencing was also applied. The ACF has a slow decay (Figure 4.1.4), indicating it's still non-stationary. The PACF also drops to 0 at lag 5, so an AR(5) component was used (Figure 4.1.5).

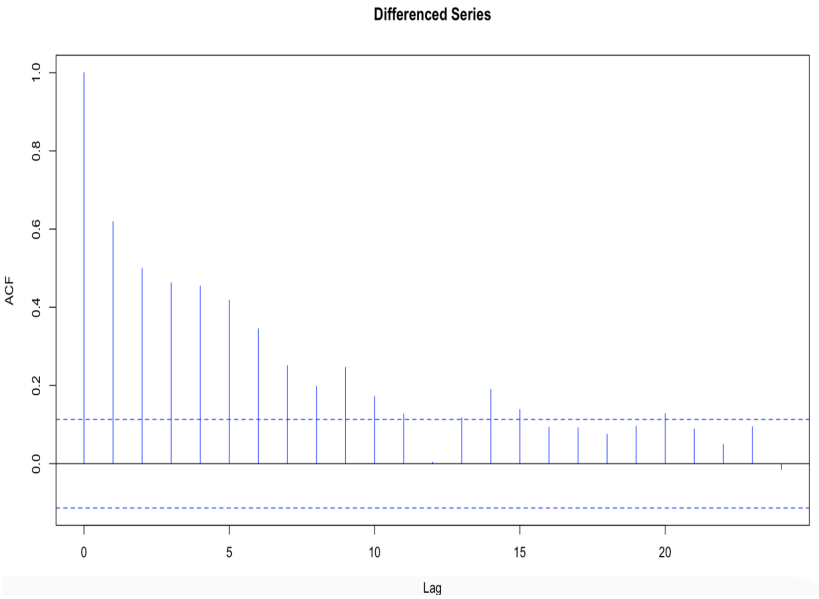


Figure 4.1.4

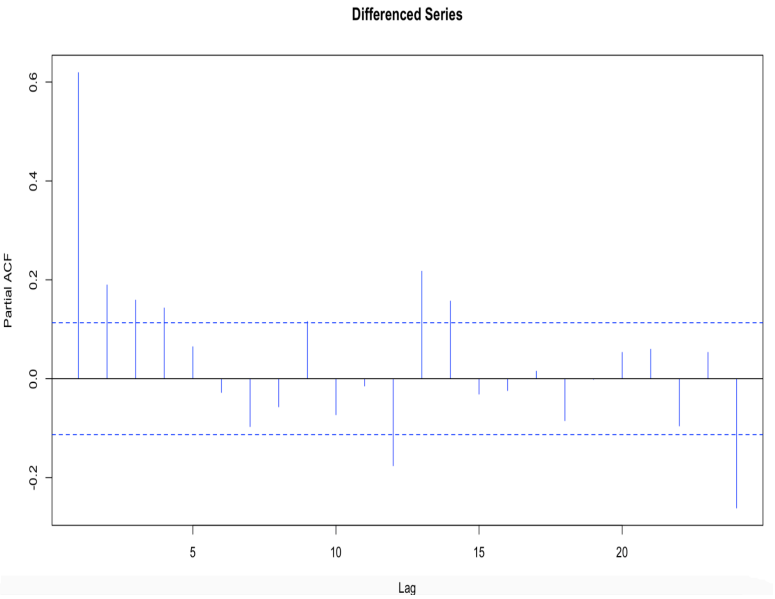


Figure 4.1.5

Model Comparison

ARIMA (0,1,1) (1,1,0) [12]: Captures short-term noise smoothing through a non-seasonal MA(1) component and models yearly seasonal patterns with a seasonal AR(1) term after both non-seasonal and seasonal differencing.

ARIMA (1,1,0) (1,1,0) [12]: Uses a single non-seasonal AR(1) term and seasonal AR(1) term to capture dependencies at lag 1 and lag 12. Appropriate when past values (monthly and yearly) influence the current observation after differencing.

ARIMA (2,1,0) (1,1,0) [12]: Incorporates two non-seasonal AR terms to model more complex short-term dependencies, along with seasonal differencing and a seasonal AR(1) component to handle yearly seasonality.

ARIMA (5,1,1) (1,1,0) [12]: A highly flexible model with five non-seasonal AR terms and one MA term for capturing intricate short-term dynamics, plus seasonal differencing and a seasonal AR(1) term. While powerful, it risks overfitting if the data doesn't justify the complexity.

Train Performance

The charts below display how the models performed on the training (Figure 4.1.6) and holdout (Figure 4.1.7). While all models performed similarly, the ARIMA (2,1,0) (1,1,0) [12] performed the strongest on the holdout data at 9.62%.

Seasonal Model	MAPE	RMSE	MAE
ARIMA (0,1,1)	6.24%	50423.8	36657.53
ARIMA (1,1,0)	6.54%	53351.14	38701.6
ARIMA (2,1,0)	6.28%	51416.97	37096.09
ARIMA (5,1,1)	6.17%	50169.29	36254.13

Figure 4.1.6

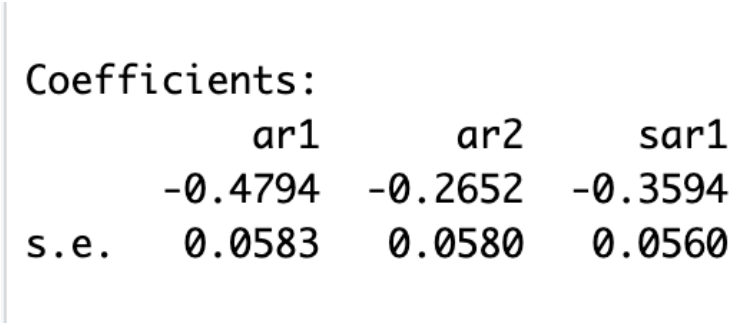
Holdout Performance

Seasonal Model	MAPE	RMSE	MAE
ARIMA (0,1,1)	10.23%	50578.06	26465.62
ARIMA (1,1,0)	9.8%	48660.55	25525.49
ARIMA (2,1,0)	9.62%	49654.18	24854.13
ARIMA (5,1,1)	10.33%	50884.39	27154.81

Figure 4.1.7

ARIMA (2,1,0) (1,1,0) [12] Model Summary

The model output shows thall all three terms are statistically significant (Figure 4.1.8). This suggests that the current value of the series is influenced negatively by its values from one and two months ago (short-term dependence), as well as by its value 12 months ago (seasonal dependence). The negative signs indicate an inverse relationship with those past values.



```
Coefficients:
          ar1      ar2     sar1
      -0.4794  -0.2652  -0.3594
s.e.   0.0583   0.0580   0.0560
```

Figure 4.1.8

4.2 Residual Performance from Trend + Seasonal Regression Model

Since the residuals mostly behave like a stationary series with points inside or below the bounds (figure 4.2.1), an ARIMA model was fit that includes seasonality to account for spikes that are seen, as done in Part 4.1. At first, the MLR - ARIMA models included the variables PPI, Gas Prices, and Unemployment (Inflation excluded since it was insignificant in the original model). However, the only two that were kept for model training were Gas Prices and Unemployment, as using more than two economic variables added too much noise to the model, and lowered accuracy. The trend and seasonality variables were also removed from the original Trend + Cyclical Model, as the ARIMA process accounts for them.

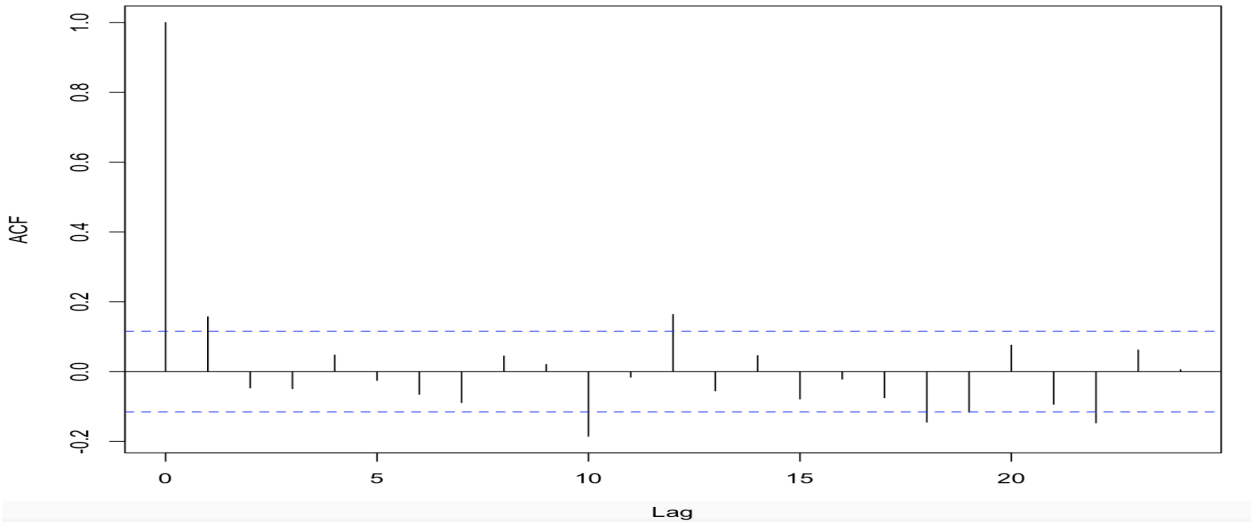


Figure 4.2.1

Model Comparison (MLR + ARIMA)

The models were trained adding gas prices and unemployment, still including seasonal differencing to account for cyclical trends. The ARIMA (0,1,1) model performed the best on the train data (Figure 4.2.2), but was again overfit. The ARIMA (2,1,0) model with economic variables considered performed the strongest on the holdout set at 6.37% (Figure 4.2.3).

Training Performance

Seasonal Model	MAPE	RMSE	MAE
ARIMA (0,1,1) MLR	**5.7%**	**48116.51**	**34055.97**
ARIMA (1,1,0) MLR	**6.33%**	**52230.51**	**37537.04**
ARIMA (2,1,0) MLR	**6.04%**	**49948.36**	**35719.48**
ARIMA (5,1,1) MLR	**5.71%**	**47620.71**	**33943.94**

Figure 4.2.2

Holdout Performance

Seasonal Model	MAPE	RMSE	MAE
ARIMA (0,1,1) MLR	**9.14%**	**42413.2**	**23700.25**
ARIMA (1,1,0) MLR	**8.61%**	**40915.52**	**23278.34**
ARIMA (2,1,0) MLR	**6.37%**	**38302.13**	**18293.73**
ARIMA (5,1,1) MLR	**10.62%**	**48475.04**	**25925.85**

Figure 4.2.3

ARIMA (2,1,0) (1,1,0) [12] MLR Model Summary

The three autoregressive terms (ar1, ar2, sar1) are all statistically significant, as well as gas prices, indicating these are the strongest predictors in the model. The relationship with unemployment rate is captured as expected, as when it increases, the expected auto sales are expected to decrease (Figure 4.2.4).

```
Coefficients:
          ar1      ar2     sar1  GasPrices   Unemploy
      -0.5159  -0.2906  -0.3828   53297.57  -5003.472
s.e.   0.0584   0.0576   0.0561   13750.08  12851.564
```

Figure 4.2.4

5. Conclusion

5.1 Model Comparison (Top Performers)

Across the training data (Figure 5.1.1), the trend and cyclical MLR model performed the strongest, however it was overfit on the holdout data (Figure 5.1.2). The ARIMA (2,1,0) (1,1,0) [12] MLR model performed the best on the holdout data at 6.37%, indicating it was the best fit for the dataset.

Training Performance

Model	MAPE	RMSE	MAE
Trend	10.62%	77858.92	63529.08
SES	10.54%	79009.2	62819.89
Trend + Cyclical MLR	4.81%	39300.36	28898.97
ARIMA (2,1,0 (1,1,0) [12]	6.28%	51416.97	37096.09
ARIMA (2,1,0 (1,1,0) [12] MLR	6.04%	49948.36	35719.48

Figure 5.1.1

Holdout Performance

Model	MAPE	RMSE	MAE
Trend	17.97 %	63560.05	50221.29
SES	14.33%	53756.38	41946.4
Trend + Cyclical MLR	13.29%	55928.77	42275.02
ARIMA (2,1,0 (1,1,0) [12]	9.62%	49654.18	24854.13
ARIMA (2,1,0 (1,1,0) [12] MLR	6.37%	38302.13	18293.73

Figure 5.1.2

Appendix - R Code

```
##PART 1
# Read in the data
autodata <- read.csv("USAutosales.csv", header = TRUE)

# Convert the 'Month' column from numeric (1,2,3,...) to month names
autodata$Month <- month.name[autodata$Month]

# Add the 'Year' column
autodata$Year <- 1995 + (seq_along(autodata$Month) - 1) %/% 12

autodata$Trend <- seq_along(autodata$Month)
autodata$GTrend <- c(
  seq(1, 166),              # First period: 1 to 166
  seq(1, 233 - 166),        # Second period: 1 to (232 - 166) = 66
  seq(1, 312 - 233)         # Third period: 1 to (312 - 232) = 80
)
# Create dummy variables for Period 2 and Period 3
autodata$D1 <- ifelse(autodata$Trend >= 167 & autodata$Trend <= 233, 1, 0)  # Period 2
autodata$D2 <- ifelse(autodata$Trend >= 234, 1, 0)                          # Period 3
train_data <- autodata[1:288, ]  # First 288 rows for training
holdout_data <- autodata[289:312, ]
colnames(autodata)
# Build the model with interaction terms
model <- lm(Autosales ~ GTrend + D1 + D2 + GTrend:D1 + GTrend:D2, data = train_data)

# View summary
summary(model)

train_data$Month <- factor(train_data$Month, levels = month.name)
model2 <- lm(Autosales ~ GTrend + D1 + D2 + GTrend:D1 + GTrend:D2 + Month, data =
train_data)
summary(model2)
# 1. Get predictions from the model
predictions <- predict(model, newdata = train_data)

# 2. Actual values
actuals <- train_data$Autosales

# 3. Calculate metrics

# MAE - Mean Absolute Error
mae <- mean(abs(actuals - predictions))
```

```
# RMSE - Root Mean Squared Error
rmse <- sqrt(mean((actuals - predictions)^2))

# MAPE - Mean Absolute Percentage Error
mape <- mean(abs((actuals - predictions) / actuals)) * 100

# Print the results
cat("MAE:", round(mae, 2), "\n")
cat("RMSE:", round(rmse, 2), "\n")
cat("MAPE:", round(mape, 2), "%\n")

test_predictions <- predict(model, newdata = holdout_data)

# Step 4: Actuals from test set
test_actuals <- holdout_data$Autosales

# Step 5: Error metrics
# MAE
test_mae <- mean(abs(test_actuals - test_predictions))

# RMSE
test_rmse <- sqrt(mean((test_actuals - test_predictions)^2))

# MAPE
test_mape <- mean(abs((test_actuals - test_predictions) / test_actuals)) * 100

# Step 6: Print results
cat("Test Set Evaluation:\n")
cat("MAE:", round(test_mae, 2), "\n")
cat("RMSE:", round(test_rmse, 2), "\n")
cat("MAPE:", round(test_mape, 2), "%\n")
# 1. Get predictions from model2
predictions2 <- predict(model2, newdata = train_data)

# 2. Actual values
actuals <- train_data$Autosales

# 3. Calculate metrics for model2
# MAE - Mean Absolute Error
mae2 <- mean(abs(actuals - predictions2))

# RMSE - Root Mean Squared Error
rmse2 <- sqrt(mean((actuals - predictions2)^2))
```

```
# MAPE - Mean Absolute Percentage Error
mape2 <- mean(abs((actuals - predictions2) / actuals)) * 100

# Print the results
cat("MAE (model2):", round(mae2, 2), "\n")
cat("RMSE (model2):", round(rmse2, 2), "\n")
cat("MAPE (model2):", round(mape2, 2), "%\n")



# Step 1: Predictions from model2 on the holdout (test) data
test_predictions2 <- predict(model2, newdata = holdout_data)

# Step 2: Actual values from the holdout set
test_actuals <- holdout_data$Autosales

# Step 3: Error metrics for model2
# MAE
test_mae2 <- mean(abs(test_actuals - test_predictions2))

# RMSE
test_rmse2 <- sqrt(mean((test_actuals - test_predictions2)^2))

# MAPE
test_mape2 <- mean(abs((test_actuals - test_predictions2) / test_actuals)) * 100

# Step 4: Print results
cat("Test Set Evaluation (model2):\n")
cat("MAE:", round(test_mae2, 2), "\n")
cat("RMSE:", round(test_rmse2, 2), "\n")
cat("MAPE:", round(test_mape2, 2), "%\n")
```

```
#model 2/periodogram
library(TSA)

# Step 1: Detrend Autosales using linear regression
detrend_model <- lm(Autosales ~ GTrend + D1 + D2 + GTrend:D1 + GTrend:D2, data =
train_data)
detrended_series <- residuals(detrend_model)

# Step 2: Compute the periodogram
prdgrm <- periodogram(detrended_series, col = "blue")

# Step 3: Extract frequencies and convert to periods
freqs <- prdgrm$freq
spec <- prdgrm$spec
periods <- 1 / freqs

# Step 4: Identify top 8 harmonic components
top_indices <- order(spec, decreasing = TRUE)[1:8]
top_frequencies <- freqs[top_indices]
top_periods <- periods[top_indices]
top_amplitudes <- spec[top_indices]

# Step 5: Display top frequencies and their periods
top_harmonics <- data.frame(
  Rank = 1:8,
  Frequency = round(top_frequencies, 4),
  Period = round(top_periods, 2),
  Amplitude = round(top_amplitudes, 4)
)

print(top_harmonics)

# Step 6: Plot both periodogram and spectrum vs. period
par(mfrow = c(1, 2))  # side-by-side plots

# Original periodogram
periodogram(detrended_series, col = "blue", main = "Periodogram (Freq Domain)")

# Spectrum vs. Period
plot(periods, spec, type = "h", col = "blue", lwd = 2,
     xlab = "Period (months)", ylab = "Spectral Density",
     main = "Spectrum vs. Period")

# Define top periods from the periodogram
```

```
top_periods <- c(12, 3, 2.4, 4, 72, 11.52, 6, 3.95)

# Generate harmonic terms for top periods in train_data
for (i in seq_along(top_periods)) {
  p <- top_periods[i]
  train_data[[paste0("cos_", i)]] <- cos(2 * pi * train_data$Trend / p)
  train_data[[paste0("sin_", i)]] <- sin(2 * pi * train_data$Trend / p)
}
for (i in seq_along(top_periods)) {
  p <- top_periods[i]
  holdout_data[[paste0("cos_", i)]] <- cos(2 * pi * holdout_data$Trend / p)
  holdout_data[[paste0("sin_", i)]] <- sin(2 * pi * holdout_data$Trend / p)
}


# Construct formula dynamically
harmonic_terms <- paste0(
  rep(c("cos_", "sin_"), each = length(top_periods)),
  rep(1:length(top_periods), times = 2),
  collapse = " + "
)

formula_text <- paste(
  "Autosales ~ GTrend + D1 + D2 + GTrend:D1 + GTrend:D2 +",
  harmonic_terms
)

# Fit model
fit_custom_sinusoidal <- lm(as.formula(formula_text), data = train_data)
summary(fit_custom_sinusoidal)


# 1. Get predictions from the sinusoidal model
pred_updated <- predict(fit_custom_sinusoidal, newdata = train_data)

# 2. Actual values
actuals_updated <- train_data$Autosales

# 3. Calculate metrics

# MAE - Mean Absolute Error
mae_updated <- mean(abs(actuals_updated - pred_updated))

# RMSE - Root Mean Squared Error
```

```
rmse_updated <- sqrt(mean((actuals_updated - pred_updated)^2))

# MAPE - Mean Absolute Percentage Error
mape_updated <- mean(abs((actuals_updated - pred_updated) / actuals_updated)) * 100

# 4. Print the results
cat("Model Performance (Sinusoidal Model on Training Data):\n")
cat("MAE:", round(mae_updated, 2), "\n")
cat("RMSE:", round(rmse_updated, 2), "\n")
cat("MAPE:", round(mape_updated, 2), "%\n")

# Step 3: Predict on the test data
test_preds_updated <- predict(fit_custom_sinusoidal, newdata = holdout_data)

# Step 4: Actual values
test_actuals_updated <- holdout_data$Autosales

# Step 5: Compute error metrics

# MAE
test_mae_updated <- mean(abs(test_actuals_updated - test_preds_updated))

# RMSE
test_rmse_updated <- sqrt(mean((test_actuals_updated - test_preds_updated)^2))

# MAPE
test_mape_updated <- mean(abs((test_actuals_updated - test_preds_updated) /
test_actuals_updated)) * 100

# Step 6: Print results
cat("Model Performance (Sinusoidal Model on Holdout Data):\n")
cat("MAE:", round(test_mae_updated, 2), "\n")
cat("RMSE:", round(test_rmse_updated, 2), "\n")
cat("MAPE:", round(test_mape_updated, 2), "%\n")
```

```
####PART 2
library(forecast)
Autosales_ts <- ts(autodata$Autosales, start = c(1995, 1), frequency = 12)

# Split data for training (first 288 months) and forecasting (next 24 months)
train_data2 <- Autosales_ts[1:288]

# Fit a Simple Exponential Smoothing model
simple_es <- ses(train_data2, h = 24)
summary(simple_es)
acf(simple_es$residuals)
# Get the fitted values (training period)
pred_es <- fitted(simple_es)

# Length of the complete time series (Autosales data)

n <- length(Autosales_ts)
n_train=(n-24)
hold_pred=rep(0,n)
hold_pred[n_train]=pred_es[n_train]
alpha=0.595

for (t in ((n_train+1):n)){
  hold_pred[t]=alpha*Autosales_ts[t-1]+(1-alpha)*hold_pred[t-1]

}
mape_hold=mean(abs(Autosales_ts[(n_train+1):n]-hold_pred[(n_train+1):n])/Autosales_ts[(n_tra
in+1):n])
mape_hold
```

```
#next model
holt_es=holt(train_data2, h=24)
summary(holt_es)
pred_holt=fitted(holt_es)

L_t=rep(0,n)
T_t=rep(0,n)

# Parameters and Initials from the output

alpha =0.6047
beta  = 0.006


L_0=709804.7665
T_0=10202.7532


# Updating period 1
L_t[1]=alpha*train_data2[1]+(1-alpha)*(L_0+T_0)
T_t[1]=beta*(L_t[1]-L_0)+(1-beta)*T_0

for (t in 2:n_train){
  L_t[t]=alpha*train_data2[t]+(1-alpha)*(L_t[t-1]+T_t[t-1])
  T_t[t]=beta*(L_t[t]-L_t[t-1])+(1-beta)*T_t[t-1]

}

all_pred=rep(0,n)

# first n_train periods are from the training sample based model
all_pred[1:n_train]=pred_holt[1:n_train]



for (t in (n_train+1):n){

  # ONE-STEP AHEAD FORCASTS FOR THE HOLD-OUT SAMPLE
  all_pred[t]=L_t[t-1]+T_t[t-1]

  # UPDATING FOR THE HOLD-OUT SAMPLE

  L_t[t]=alpha*Autosales_ts[t]+(1-alpha)*(L_t[t-1]+T_t[t-1])
  T_t[t]=beta*(L_t[t]-L_t[t-1])+(1-beta)*T_t[t-1]
```

```
}

Month=c((n_train+1):n)
comp=cbind(Month,Autosales_ts[(n_train+1):n],all_pred[(n_train+1):n])
colnames(comp)=c("Month","actual","predicted")
comp

mape_hold=mean(abs(Autosales_ts[(n_train+1):n]-all_pred[(n_train+1):n])/Autosales_ts[(n_train
+1):n])
mape_hold




#next model
# Ensure your train_data2 is a time series object
train_data2_ts <- ts(train_data2, start = c(1995, 1), frequency = 12)  # Assuming the data starts
in January 1995

# Now apply Holt-Winters method
library(forecast)

hw_es <- hw(train_data2_ts)
summary(hw_es)

# Get the fitted values
pred_hw <- hw_es$fitted

# Obtaining the level, slope and seasonal estimates for training set
L_t=rep(0,n)
T_t=rep(0,n)
S_t=rep(0,n)

# Parameters and Initials from the output

alpha=0.431
beta=0.0011
gamma=0.0001

L_0=728490.3655
T_0=771.1513
```

```
S0=c(1:12)
S0[1]=-19873.11
S0[2]=-87730.71
S0[3]=-42397.36
S0[4]=-24982.2
S0[5]=63384.79
S0[6]=24617.08
S0[7]=64135.27
S0[8]=94665.99
S0[9]=25279.47
S0[10]=78767.33
S0[11]=-46305.33
S0[12]=-129561.2


L_t[1]=alpha*(train_data2_ts[1]-S0[12])+(1-alpha)*(L_0+T_0)
T_t[1]=beta*(L_t[1]-L_0)+(1-beta)*T_0
S_t[1]=gamma*(train_data2_ts[1]-L_t[1])+(1-gamma)*S0[12]

# 2 to 12

for (s in 2:12){
  L_t[s]=alpha*(train_data2_ts[s]-S0[12-s+1])+(1-alpha)*(L_t[s-1]+T_t[s-1])
  T_t[s]=beta*(L_t[s]-L_t[s-1])+(1-beta)*T_t[s-1]
  S_t[s]=gamma*(train_data2_ts[s]-L_t[s])+(1-gamma)*S0[12-s+1]

}
for (t in 13:n_train){

  L_t[t]=alpha*(train_data2_ts[t]-S_t[t-12])+(1-alpha)*(L_t[t-1]+T_t[t-1])
  T_t[t]=beta*(L_t[t]-L_t[t-1])+(1-beta)*T_t[t-1]
  S_t[t]=gamma*(train_data2_ts[t]-L_t[t])+(1-gamma)*S_t[t-12]

}


all_pred2=rep(0,n)

# first n_train periods are from the training sample based model
all_pred2[1:n_train]=pred_hw[1:n_train]



for (t in (n_train+1):n){
```

```
  # ONE-STEP AHEAD FORECASTS FOR THE HOLD-OUT SAMPLE
  all_pred2[t]=L_t[t-1]+T_t[t-1]+S_t[t-12]

  # UPDATING FOR THE HOLD-OUT SAMPLE

  L_t[t]=alpha*(Autosales_ts[t]-S_t[t-12])+(1-alpha)*(L_t[t-1]+T_t[t-1])
  T_t[t]=beta*(L_t[t]-L_t[t-1])+(1-beta)*T_t[t-1]
  S_t[t]=gamma*(Autosales_ts[t]-L_t[t])+(1-gamma)*S_t[t-12]

}

Month=c((n_train+1):n)
comp=cbind(Month,Autosales_ts[(n_train+1):n],all_pred2[(n_train+1):n])
colnames(comp)=c("Month","actual","predicted")
comp

mape_hold=mean(abs(Autosales_ts[(n_train+1):n]-all_pred2[(n_train+1):n])/Autosales_ts[(n_trai
n+1):n])
mape_hold
rmse_hold <- sqrt(mean((Autosales_ts[(n_train + 1):n] - all_pred2[(n_train + 1):n])^2))
cat("RMSE: ", rmse_hold, "\n")

# MAE: Mean Absolute Error
mae_hold <- mean(abs(Autosales_ts[(n_train + 1):n] - all_pred2[(n_train + 1):n]))
cat("MAE: ", mae_hold, "\n")

# Read in the data
autodata <- read.csv("USAutosales.csv", header = TRUE)

# Convert the 'Month' column from numeric (1,2,3,...) to month names
autodata$Month <- month.name[autodata$Month]

# Add the 'Year' column
autodata$Year <- 1995 + (seq_along(autodata$Month) - 1) %/% 12

autodata$Trend <- seq_along(autodata$Month)

# Create a new Trend variable that resets for each period
autodata$GTrend <- c(
  seq(1, 166),              # First period: 1 to 166
  seq(1, 233 - 166),        # Second period: 1 to (232 - 166) = 66
  seq(1, 312 - 233)         # Third period: 1 to (312 - 232) = 80
)
```

```
autodata$D1 <- ifelse(autodata$Trend >= 167 & autodata$Trend <= 233, 1, 0)  # Period 2
autodata$D2 <- ifelse(autodata$Trend >= 234, 1, 0)                          # Period 3
train_data <- autodata[1:288, ]  # First 288 rows for training
holdout_data <- autodata[289:312, ]
mlrfit <- lm(Autosales ~ GTrend + D1 + D2 + GTrend:D1 + GTrend:D2 + Inflation + Unemploy +
PPI + GasPrices, data = train_data)

# Display summary of the model
summary(mlrfit)


# Predict on the training data
train_predictions <- predict(mlrfit, newdata = train_data)

# Actual values
actuals <- train_data$Autosales

# Errors
errors <- actuals - train_predictions

# Mean Absolute Error (MAE)
mae <- mean(abs(errors))

# Root Mean Squared Error (RMSE)
rmse <- sqrt(mean(errors^2))

# Mean Absolute Percentage Error (MAPE)
mape <- mean(abs(errors / actuals)) * 100

# Print results
cat("MAE:", mae, "\n")
cat("RMSE:", rmse, "\n")
cat("MAPE:", mape, "%\n")


train_predictions <- predict(mlrfit, newdata = holdout_data)

# Actual values
actuals <- holdout_data$Autosales

# Errors
errors <- actuals - train_predictions
```

```
# Mean Absolute Error (MAE)
mae <- mean(abs(errors))

# Root Mean Squared Error (RMSE)
rmse <- sqrt(mean(errors^2))

# Mean Absolute Percentage Error (MAPE)
mape <- mean(abs(errors / actuals)) * 100

# Print results
cat("MAE:", mae, "\n")
cat("RMSE:", rmse, "\n")
cat("MAPE:", mape, "%\n")
```

```
####PART 3
# Define top 6 periods from the periodogram
top_periods <- c(12, 3, 2.4, 4, 72, 11.52)  # First 6 periods

# Generate harmonic terms for top periods in train_data and holdout_data
for (i in seq_along(top_periods)) {
  p <- top_periods[i]
  train_data[[paste0("cos_", i)]] <- cos(2 * pi * train_data$Trend / p)
  train_data[[paste0("sin_", i)]] <- sin(2 * pi * train_data$Trend / p)

  holdout_data[[paste0("cos_", i)]] <- cos(2 * pi * holdout_data$Trend / p)
  holdout_data[[paste0("sin_", i)]] <- sin(2 * pi * holdout_data$Trend / p)
```

```
}

# Construct harmonic term part of the formula dynamically (6 harmonics × 2 terms each = 12
terms)
harmonic_terms <- paste0(
  rep(c("cos_", "sin_"), each = length(top_periods)),
  rep(1:length(top_periods), times = 2),
  collapse = " + "
)

# Combine original model formula with seasonal (harmonic) terms
full_formula_text <- paste(
  "Autosales ~ GTrend + D1 + D2 + GTrend:D1 + GTrend:D2 + Inflation + Unemploy + PPI +
GasPrices +",
  harmonic_terms
)

# Fit the updated model with selected seasonal harmonics and economic variables
mlrfit_seasonal6 <- lm(as.formula(full_formula_text), data = train_data)
summary(mlrfit_seasonal6)
# --- Training Set Evaluation ---
train_preds6 <- predict(mlrfit_seasonal6, newdata = train_data)
actuals6 <- train_data$Autosales
errors6 <- actuals6 - train_preds6

# Metrics
mae_train6 <- mean(abs(errors6))
rmse_train6 <- sqrt(mean(errors6^2))
mape_train6 <- mean(abs(errors6 / actuals6)) * 100

# Print results for training
cat("TRAINING SET - FULL MODEL WITH TOP 6 HARMONICS:\n")
cat("MAE:", round(mae_train6, 2), "\n")
cat("RMSE:", round(rmse_train6, 2), "\n")
cat("MAPE:", round(mape_train6, 2), "%\n\n")

# --- Holdout Set Evaluation ---
holdout_preds6 <- predict(mlrfit_seasonal6, newdata = holdout_data)
actuals_holdout6 <- holdout_data$Autosales
errors_holdout6 <- actuals_holdout6 - holdout_preds6

# Metrics
mae_holdout6 <- mean(abs(errors_holdout6))
rmse_holdout6 <- sqrt(mean(errors_holdout6^2))
```

```
mape_holdout6 <- mean(abs(errors_holdout6 / actuals_holdout6)) * 100

# Print results for holdout
cat("HOLDOUT SET - FULL MODEL WITH TOP 6 HARMONICS:\n")
cat("MAE:", round(mae_holdout6, 2), "\n")
cat("RMSE:", round(rmse_holdout6, 2), "\n")
cat("MAPE:", round(mape_holdout6, 2), "%\n")

# Extract residuals
resid_seasonal6 <- residuals(mlrfit_seasonal6)

# Plot ACF
acf(resid_seasonal6, main = "ACF of Residuals - Full Model with Top 6 Harmonics")
Box.test(resid_seasonal6, lag = 24, type = "Box-Pierce")


library(ggplot2)

# Step 1: Add predictions and label to each dataset
train_data$Set <- "Train"
holdout_data$Set <- "Holdout"

train_data$Predicted <- train_preds6
holdout_data$Predicted <- holdout_preds6

# Step 2: Combine the data
combined_data <- rbind(train_data, holdout_data)

# Step 3: Create the plot
ggplot(combined_data, aes(x = Trend)) +
  geom_line(aes(y = Autosales, color = "Actual"), size = 1) +
  geom_line(aes(y = Predicted, color = "Predicted"), size = 1, linetype = "dashed") +
  scale_color_manual(values = c("Actual" = "blue", "Predicted" = "red")) +
  labs(title = "Actual vs Predicted Autosales (Train + Holdout)",
       y = "Autosales", x = "Trend", color = "Legend") +
  geom_vline(xintercept = max(train_data$Trend), linetype = "dotted", color = "red") +
  annotate("text", x = max(train_data$Trend), y = max(combined_data$Autosales),
           label = "Train/Test Split", color = "red", vjust = -.5, hjust = 1.05) +
  theme_minimal()



#correlations/plots
```

```
# Subset by period
period1 <- autodata[autodata$Trend >= 1 & autodata$Trend <= 166, ]
period2 <- autodata[autodata$Trend >= 167 & autodata$Trend <= 233, ]
period3 <- autodata[autodata$Trend >= 234 & autodata$Trend <= 312, ]

# Variables to include
vars <- c("Autosales", "GTrend", "Inflation", "Unemploy", "PPI", "GasPrices")

# Compute correlation matrices for each period
cor_matrix_p1 <- cor(period1[, vars])
cor_matrix_p2 <- cor(period2[, vars])
cor_matrix_p3 <- cor(period3[, vars])

vars_full <- c("Autosales", "Trend", "Inflation", "Unemploy", "PPI", "GasPrices")

# Compute correlation matrix
cor_matrix_full <- cor(autodata[, vars_full])

# Plot it nicely
library(corrplot)

corrplot(cor_matrix_full, method = "color", type = "upper",
         tl.col = "black", tl.srt = 45, addCoef.col = "black", number.cex = 0.7,
         title = "Correlation Matrix — Full Period (1995–2021)", mar = c(0,0,2,0))
```

```
####PART 4

# Create time series
autosales_ts <- ts(autodata$Autosales, start = c(1995, 1), frequency = 12)

# Plot time series
ts.plot(autosales_ts, main = "Auto Sales Time Series", ylab = "Sales", col = "darkgreen")
acf(autodata$Autosales)
```

```
dMShare = diff(autodata$Autosales)
acf(dMShare, col = 'blue', main = "Differenced Series")
pacf(dMShare, col = 'blue', main = "Differenced Series")

diff_seasonal <- diff(autodata$Autosales, lag = 12)
acf(diff_seasonal, col = 'blue', main = "Differenced Series")
pacf(diff_seasonal, col = 'blue', main = "Differenced Series")
n_MShare = autodata$Autosales[1:(length(autodata$Autosales)-24)]

library(forecast)
fit2 <- Arima(n_MShare,
              order = c(5, 1, 1),
              seasonal = list(order = c(1, 1, 0), period = 12))
fit2
accuracy(fit2)

hold<- autodata[289:312, ]
# Extract only the holdout autosales series
hold_ts <- ts(hold$Autosales, start = c(2019, 1), frequency = 12)

# Forecast using the previously fitted model
holdfit <- Arima(hold_ts, model = fit2)

# Summary of the fitted model on holdout data
summary(holdfit)



library(forecast)

# Fit ARIMA(1,1,0)(1,1,0)[12] model
fit2 <- Arima(n_MShare,
              order = c(2, 1, 0),
              seasonal = list(order = c(1, 1, 0), period = 12))
summary(fit2)



fit_ma1 <- Arima(n_MShare,
                 order = c(0, 1, 1),  # MA(1), differencing = 1
                 seasonal = list(order = c(1, 1, 0), period = 12))  # Seasonal differencing
summary(fit_ma1)

# Forecast on holdout data using previously fitted model
```

```
holdfit <- Arima(hold_ts, model = fit_ma1)

# Summary of model fitted to holdout
accuracy(holdfit)

# Check accuracy on training data
accuracy(fit2)
# Extract holdout data
hold <- autodata[289:312, ]

# Convert holdout series to time series object
hold_ts <- ts(hold$Autosales, start = c(2019, 1), frequency = 12)

# Forecast on holdout data using previously fitted model
holdfit <- Arima(hold_ts, model = fit2)

# Summary of model fitted to holdout
accuracy(holdfit)




all_predictors <- c( "Unemploy", "PPI", "GasPrices")

# Create xreg matrices
x_train <- model.matrix(as.formula(paste("~", paste(all_predictors, collapse = " + "))), data =
train_data)[, -1]
x_hold <- model.matrix(as.formula(paste("~", paste(all_predictors, collapse = " + "))), data =
holdout_data)[, -1]


# Fit ARIMA(5,1,1)(1,1,0)[12] with xreg to training data
fit_reg_seasonal <- Arima(train_data$Autosales,
                          order = c(5, 1, 1),
                          seasonal = list(order = c(1, 1, 0), period = 12),
                          xreg = x_train)
summary(fit_reg_seasonal)


xreg_holdout <- model.matrix(
  as.formula(paste("~", paste(all_predictors, collapse = " + "))),
  data = holdout_data
)[, -1]  # Remove intercept
```

```
hold <- autodata[289:312, ]

# Convert holdout series to time series object
hold_ts <- ts(hold$Autosales, start = c(2019, 1), frequency = 12)

# Forecast on holdout data using previously fitted model
holdfit <- Arima(hold_ts, model = fit_reg_seasonal, xreg= xreg_holdout)

# Summary of model fitted to holdout
accuracy(holdfit)

library(forecast)
all_predictors <- c( "GasPrices","Unemploy")

# Create xreg matrices
x_train <- model.matrix(as.formula(paste("~", paste(all_predictors, collapse = " + "))), data =
train_data)[, -1]
x_hold <- model.matrix(as.formula(paste("~", paste(all_predictors, collapse = " + "))), data =
holdout_data)[, -1]


# Fit ARIMA(5,1,1)(1,1,0)[12] with xreg to training data
fit_reg_seasonal <- Arima(train_data$Autosales,
                      order = c(2, 1, 0),
                      seasonal = list(order = c(1, 1, 0), period = 12),
                      xreg = x_train)
summary(fit_reg_seasonal)


xreg_holdout <- model.matrix(
  as.formula(paste("~", paste(all_predictors, collapse = " + "))),
  data = holdout_data
)[, -1]  # Remove intercept
hold <- autodata[289:312, ]

# Convert holdout series to time series object
hold_ts <- ts(hold$Autosales, start = c(2019, 1), frequency = 12)

# Forecast on holdout data using previously fitted model
holdfit <- Arima(hold_ts, model = fit_reg_seasonal, xreg= xreg_holdout)

# Summary of model fitted to holdout
accuracy(holdfit)
```