Regularized Vector Autoregressive Model for Time Series Data with Multiple Covariates

Waweru, Esther Nyakio

Submitted in partial fulfilment of the requirements for the degree of Master of Science in Statistical Science of Strathmore University

Strathmore Institute of Mathematical Sciences
Strathmore University
Nairobi, Kenya

June 2025

Declaration

I declare that this work has not been previously submitted and approved for award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Name: **Waweru Esther Nyakio**

Signature: 

Date: March 27, 2025

Approval

The thesis of Waweru Esther Nyakio was reviewed and approved by :

Dr Jacob Ong'ala

Institute of Mathematical Sciences

Strathmore University.

Dr. Godfrey Madigu

Dean, Institute of Mathematical Sciences,

Strathmore University.

Prof. Bernard Shibwabo

Director of Graduate Studies,

Strathmore University

Abstract

This study develops a Regularized Vector Autoregressive (VAR) model to address challenges like high dimensionality, multicollinearity, and overfitting in time series forecasting with external covariates. Traditional VAR models often struggle with scalability and stability in high-dimensional contexts. By incorporating Ridge, Lasso, and Elastic Net regularization techniques, the model enhances forecasting accuracy and model interpretability. Using a real-world dataset of Walmart sales, including weekly sales alongside economic and environmental covariates, the methodology applies preprocessing, regularized model formulation, and cross-validation for parameter tuning. Performance is evaluated using metrics like Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), comparing traditional and regularized VAR models. The findings demonstrate the utility of regularized VAR models in handling complex time series data influenced by external covariates, with implications for broader applications in fields such as finance, healthcare, and environmental science.

KEY WORDS: Regularized VAR model, Time Series Forecasting, Multiple Covariates, High Dimensionality, Overfitting.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ENR	Elastic Net Regression	LASSO	Least Absolute Shrinkage and Selection Operator
MAE	Mean Absolute Error	MAPE	Mean absolute percentage error
RR	Ridge Regression	RMSE	Root Mean Square Error
VAR	Vector Autoregressive		

Acknowledgement

First and foremost, I am deeply grateful to the Almighty God for His provision, grace, and the gift of good health throughout my study period.

I would also like to extend my heartfelt gratitude to my supervisor, Dr. Jacob Ong'ala, for his unparalleled support, dedication, and guidance throughout the journey of completing this thesis.

I am also deeply thankful to my parents, brother, and friends for their unwavering encouragement and assistance during the study.

Lastly, my sincere appreciation goes to the faculty, staff, and students in our department for making my time at Strathmore University such a memorable and enriching experience.

Dedication

This thesis is dedicated to God Almighty, with gratitude for the wisdom and good health bestowed upon me.

Chapter 1

Introduction

1.1 Introduction

1.1.1 Background Information

Time series data is a sequence of temporal data observations recorded usually at regular time intervals, where each observation is connected to its preceding and succeeding observations. This kind of data becomes highly useful for the identification of trends and making future forecasts, as it captures patterns over time that are more often than not influenced by previous occurrences. In economics, medicine, environmental science, and many other fields, understanding complex systems with temporal dependencies relies on the ability to analyze and model time series data (Shumway and Stoffer, 2011).

Time series analysis can be of either univariate or multivariate type. A univariate time series focuses on the analysis of a single variable observed over time (Box et al., 2015). On the contrary, a multivariate time series involves multiple interrelated variables, where each can influence and get influenced by others. In this way, there forms a much more complex system which faces challenges such as high dimensionality, overfitting and multicollinearity, which can hinder analysis and interpretation of these variables ((Hamilton, 1994b); (Belsley et al., 1980)).

The impact of time series analysis can be felt in many fields. In economics, it helps model stock market dynamics and monitor unemployment rates. Authors such as Fama (1970) and Stock and Watson (2011) have used time series techniques to study economic variability and forecast future trends. Epidemiologists use time series data to track the evaluation of diseases, such influenza outbreak patterns over time. Studies

by Shaman and Kohn (2009) and Simonsen et al. (2011) demonstrate that such time series could predict seasonal variations in the spread of diseases. In medical research, time series is critical for monitoring patient vitals like blood pressure which might be helpful when assesing the effectiveness of hypertension drugs being taken (Shumway and Stoffer, 2011). Neuroscience applies time series techniques to the analysis of brain-wave activity in order to understand what the brain reacts to under certain stimuli (Betzel et al., 2016); (Damoiseaux et al., 2006); (Faskowitz et al., 2020). Environmental scientists apply time series techniques while analyzing changes in global temperature and examining health impacts that are related to pollution exposure, as shown by Hansen et al. (2010) and Jhun et al. (2014). Finally, in engineering, time series is instrumental in improving speech recognition technologies and analyzing seismic activity, with foundational work by Rabiner and Juang (1993) advancing these areas.

Statistical approaches to multivariate time series often rely on the Vector Autoregressive (VAR) model in which each variable in a system is modelled as a function of its past values and values of other variables. As seen in Cavalcante et al. (2016), a k-dimensional vector time series is represented by $\{Y_t\} = \{(y_{1,t}, y_{2,t}, \ldots, y_{k,t})'\}$. When this system is modeled as a vector autoregressive process of order p ($\text{VAR}_k[p]$), the future values of each of the k variables are determined based on the lagged values of all the variables in the system. This is expressed mathematically as follows:

$$Y_t = \Phi + \sum_{l=1}^{p} B^{(l)} \cdot Y_{t-l} + \varepsilon_t, \tag{1.1}$$

where Φ is a vector of constant terms, $B^{(l)} \in \mathbb{R}^{k \times k}$ denotes matrix of coefficients corresponding to each lag l, and $\varepsilon_t \sim N(0, \Sigma_\varepsilon)$ is the white noise disturbance term. The variance of the error process is given by $\text{Var}(\varepsilon_t) = \Sigma_\varepsilon \, \forall t$, where $\Sigma_\varepsilon \in \mathbb{R}^{k \times k}$ is the contemporaneous covariance matrix of the innovations across the k time series. It is symmetric and positive definite.

To reformulate the VAR model in matrix notation, let $Y = (Y_1, Y_2, \ldots, Y_T)$ represent the $k \times T$ response matrix and $B = (B^{(1)}, B^{(2)}, \ldots, B^{(p)})$ denote the $k \times kp$ coefficient

matrix. The explanatory variables are denoted as $Z = (Z_1, Z_2, \ldots, Z_T)$, a $kp \times T$ matrix, where $Z_t = (Y'_{t-1}, Y'_{t-2}, \ldots, Y'_{t-p})$. The error terms are collected into the $k \times T$ matrix $E = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_T)$ (Cavalcante et al., 2016).

For notation simplicity, let $m = kp$. The system can then be represented compactly as

$$Y = \Phi 1' + BZ + E, \tag{1.2}$$

in which 1 is a $T \times 1$ vector of ones.

The VAR model assumes that the time series data Y_t is stationary, meaning its mean and variance do not change over time. It also assumes that the error terms ε_t are uncorrelated across time (i.e., no serial correlation), have zero mean $\mathbb{E}[\varepsilon_t] = 0 \quad \text{for all} \quad t$ and constant variance (homoscedasticity), such that $\text{Var}(\varepsilon_t) = \Sigma_\varepsilon \quad \text{for all} \quad t$. The explanatory variables Z_t, which are lagged values of Y_t, are assumed to be exogenous, meaning they are uncorrelated with the error terms $\text{Cov}(Z_t, \varepsilon_t) = 0 \quad \text{for all} \quad t$. Additionally, perfect multicollinearity does not exist among the explanatory variables, implying that the matrix Z has full rank. These assumptions ensure the model's validity and allow for efficient and unbiased estimation of parameters.

In multivariate time series analysis, the challenge of high dimensionality arises when the model has numerous interdependent variables, each of with their lagged versions to account for temporal dependencies. For instance, a dataset with 15 variables and 4 lags for each will produce around 60 predictors, rapidly inflating the feature space. This may make the model to become overly complex, as the addition of lagged variables increases the risk of overfitting,a situation in which a model captures noise rather than meaningful patterns in the training data. In other words, as more component series are added, overparameterization occurs in a VAR model, resulting in a bad tradeoff between flexibility and good forecasting performance (Nicholson et al., 2020). However, even though the model might fit the training datasets very well, it often poorly predicts the exact values for previously unseen data, resulting in low robustness (Hastie et al., 2009).

With the increase in dimensionality, multicollinearity which is defined as a scenario in which predictor variables exhibit highly correlation becomes a significant issue. In time series data, historical values of any variable are inherently correlated, and variables may share common trends or seasonal patterns, further worsening the situation. The model has problems in allocating explanatory power among correlated predictors, resulting in complications in trying to isolate the effects of individual predictors hence making parameter estimates unstable. It suffers from reduced interpretability due to the presence of highly correlated predictors and may also lead to inflated variances of coefficients, therefore deteriorating predictive reliability (Chatfield, 2016); (Hyndman and Athanasopoulos, 2018). In such cases, even small changes in the data can cause drastic variations in the coefficients, thus making the model unreliable for inference and prediction (Banbura et al., 2010).

These challenges are further compounded with the inclusion of multiple covariates; for every new variable added, not only does the number of predictors increase, but so do complex interactions and potential redundancies in the model. The feature space quickly expands, increasing the computational cost of the model and making it prone to overfitting. Relationship among variables also makes it hard to determine which features really help the model in predictive ability and which are just adding noise (Nicholson et al., 2020).

1.1.2 Regularization Techniques in Linear Models

Regularization Techniques are widely used when it comes to handling high dimensionality, overfitting and multicollinearity of linear models. They introduce penalties during the estimation process to reduce the overall complexity of the model and discourage it from fitting training data too closely hence improve generalizability. These techniques are found to be very effective in high dimensional cases where the number of predictors may be equal to or even larger than the number of observations, resulting in unstable parameter estimates and reduced predictive performance. By balancing the tradeoff between model complexity and fit to data, regularization methods enhance the sta-

bility and interpretability of linear models, hence making them extremely useful for forecasting and analysis of complex temporal data.

Ridge regression, also known as L_2 regularization, is one of the most common methods of regularization, first proposed by Hoerl and Kennard (1970). The method modifies the Ordinary Least Squares (OLS) estimation process by adding a penalty term proportional to the square of the regression coefficients. The penalty shrinks the magnitude of the coefficients thus making the model less prone to multicollinearity. Unlike OLS, Ridge ensures that no predictor becomes dominant as a result of high correlation with others, and hence is particularly effective at handling multicollinear data. While Ridge does retains all predictors in the model, it spreads out their influence more evenly, thereby enhancing predictive stability. It is particularly valuable in high dimensional settings, like multivariate time series with many lagged variables, where the complexity in the model can lead to overfitting (Hoerl and Kennard, 1970); (Hastie et al., 2009). The ridge regression is given by:

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2, \tag{1.3}$$

where y_i is the observed response for the i^{th} observation, x_{ij} is the value of the j^{th} predictor for observation i, β_0 is the intercept, β_j are the predictor coefficients, and $\lambda \geq 0$ is the regularization parameter that controls the degree of shrinkage on the coefficients.

Lasso Regression (Least Absolute Shrinkage and Selection Operator), first proposed by Tibshirani (1996), is an extension to the concept of regularization by applying an L1 penalty. This penalty not only shrinks coefficients but also forces some to become exactly zero, thus doing variable selection. In high dimensional data, this method finds the most relevant predictors to simplify the model and increase interpretability. In multivariate time series, where redundant lagged variables or covariates are common, Lasso is a very effective tool to exclude the less relevant features and retain the most predictive ones. However, Lasso may have problems with grouped predictors, where

highly correlated variables may compete for inclusion and result in arbitrary exclusions (Tibshirani, 1996); (Zou and Hastie, 2005).The Lasso regression is given by

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|, \tag{1.4}$$

where y_i is the observed response for the i^{th} observation, x_{ij} is the predictor variable for observation i and predictor j, β_0 is the intercept, β_j are the regression coefficients, and $\lambda \geq 0$ is the regularization parameter that applies an L_1 penalty, encouraging sparsity in the coefficient estimates.

For elastic net, it combines the benefits of Ridge and Lasso by adding both L1 and L2 penalties in its regularization term. Introduced by Zou and Hastie (2005), it is especially useful in situations where predictor variables are highly correlated, because it retains clusters of correlated variables rather than selecting one from the group at random. This makes Elastic Net especially valuable for multivariate time series models where it frequently arises that there is a set of correlated predictors, such as lagged instances of the same variable. Combining effectively the bias reduction characteristic of Ridge regression with the ability of variable selection inherent in Lasso, this approach provides a flexible framework appropriate for high-dimensional modeling (Zou and Hastie, 2005); (Hastie et al., 2009). The Elastic Net regression is given by

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\left(\alpha\sum_{j=1}^{p}|\beta_j|+(1-\alpha)\sum_{j=1}^{p}\beta_j^2\right), \tag{1.5}$$

where y_i is the observed response for the i^{th} observation, x_{ij} is the predictor variable, β_0 is the intercept, β_j are the regression coefficients, $\lambda \geq 0$ is the overall penalty term, and $\alpha \in [0,1]$ is the mixing parameter that controls the trade-off between the Lasso (L_1) and Ridge (L_2) penalties.

While regularization techniques like Lasso, Ridge, and Elastic Net have shown promising results in dealing with problems of multicollinearity, overfitting, and high dimension-

ality, there are still significant gaps in understanding their relative performances in multivariate time series models. In particular, it is not known which regularization method best balances predictive accuracy, model stability, and interpretability in the presence of temporal dependencies and complex data structures.

1.1.3 Statement of the Problem

Despite the important role of time series forecasting in finance, medicine, and environmental science, there have always been some big challenges whenever an application involved multivariate data. Noise, multicollinearity, and overfitting which are nagging triumviral problems block the route to good model performance and reduce generalization, especially in regard to dynamic pattern variations unfolding over time. A very important challenge a model faces is high dimensionality, since its parameter space increases exponentially once several covariates and their lagged versions are integrated. This increases the computational burden apart from the risk of overfitting, when the model learns from the noises rather than meaningful patterns and diminishes predictive powers. Besides, due to the inherent correlations among lagged variables and covariates, multicollinearity causes instabilities in the parameter estimates that, in turn, reduce interpretability and reliability of the model. While traditional VAR models are usually very useful, they are computationally demanding in high-dimensional contexts. Therefore, it does present the need for the development of methodologies that would handle such data complexities without losing accuracy and strength.

Regularization techniques offer solutions through shrinkage or selection of coefficients which influence mitigation against overfitting, solve multicollinearity issues, and result in more stable models. Although most of the literature related to regularization has focused on cross-section data, their applications to the VAR models in time series forecasting remains underexplored. Specifically, the understanding of such techniques working for high dimensional time series data with many covariates and dependencies over time remains scanty. It is with this backdrop that the present study has been designed to fill the lacuna by formulating and testing the efficiency of regularized VAR

models in modeling complicated patterns, enhancing forecasting accuracy, and practical utility in real-world multivariate time series applications.

1.1.4 Significance of the Study

Given the limitations of traditional forecasting models in handling multivariate time series data,this study aims to fill an essential gap by examining the role of regularization techniques, specifically Ridge, Lasso and Elastic Net within the context of VAR models. Regularization methods, which have proven to reduce overfitting and enhancing stability in cross-sectional data, hold promise for time series applications (Tibshirani, 1996); (Zou and Hastie, 2005). By adapting these methods,this study not only seeks to enhance forecasting accuracy but also to address computational issues in typical high dimensional, real world datasets where standard VAR models faces problems (Nicholson et al., 2017).

The results of this research are expected to enhance the development of more resilient and comprehensible forecasting models, offering considerable advantages to sectors such as finance, healthcare and environmental science, where proficient management of intricate datasets is essential. For example, financial institutions can utilize these models to forecast market trends and reduce risks, whereas healthcare applications may gain from enhanced predictive monitoring of patient health indicators (Basu and Michailidis, 2015);(Choi et al., 2017). This research also aims at further developing the general understanding of the important role of regularization in overcoming multicollinearity and high dimensionality in time series data by studying effective methods that stabilize or optimize time series models in general (Hamilton, 1994a); (Hastie et al., 2009).

That said, the time series data keeps growing in complexity with the introduction of multiple covariates and lagged variables. The present research, therefore, tries to set the general grounds for embedding effective regularization techniques into the VAR models. Major concerns would range from the instability of parameters or overfitting models toward models that are more interpretable and flexible in a wide range of time series

applications (Nicholson et al., 2017); (Zou and Hastie, 2005). The current study will set a benchmark to apply Ridge, Lasso, and Elastic Net in a VAR framework with a view to provide actionable insights that enhance forecasting accuracy and computational efficiency and drive critical decisions in those very industries where reliability and precision are nonnegotiable.

1.1.5 Research Aim and Objectives

The primary aim of this study is to develop a Regularized VAR model for time series data with multiple covariates.

Objectives of the study:

1. To formulate Ridge, Lasso and Elastic Net regularization techniques within VAR framework.
2. To evaluate the performance of regularized VAR model using multivariate time series dataset with multiple covariates.
3. To compare forecasting accuracy of regularized VAR models against traditional VAR models.

1.1.6 Research Questions

This study will address the following research questions :

1. How can Ridge, Lasso, and Elastic Net regularization techniques be adapted and formulated for integration into the VAR framework for multivariate time series modeling?
2. How effectively do regularized VAR models perform in handling multivariate time series datasets with multiple covariates, particularly in mitigating challenges such as high dimensionality and multicollinearity?

3. Do regularized VAR models demonstrate superior forecasting accuracy and computational efficiency compared to traditional VAR models when applied to multivariate time series data?

Chapter 2

Literature Review

2.1 Time Series Forecasting

Time series forecasting plays an essential role in disciplines such as finance, economics, and environmental science by enabling predictions of future trends based on historical data. This analytical technique is primarily applied to gain insight into the dynamic structure of a process, study interdependencies between the variables at play, carry out seasonal adjustments-which would be applicable in, say, economic data related to GDP and unemployment rates - perform regression and causality analysis, and make forecasts covering both point estimates and interval estimates, with due consideration for initial condition sensitivity (Kanjilal and Ghosh, 2021). The basic models used in this prediction can be broadly put into two groups: statistical and machine-learning methods, each with its own merits depending on the complication level in the data and the period of the forecast (Box et al., 2015); (Hyndman and Athanasopoulos, 2018).

A very well-known statistical method is the Autoregressive Integrated Moving Average (ARIMA) which is flexible and powerful in capturing various kinds of trends within the data. In the ARIMA model, there are three key components: autoregression - the usage of past values to forecast future values; differencing(I) which helps make the mean stationary over time and moving average(MA) maps the relationship between observations and residual errors from previous forecasts. Originally developed by Box and Jenkins, ARIMA is particularly useful for forecasting time series data that exhibit trends or seasonality (Box et al., 2015); (Hyndman and Athanasopoulos, 2018).

Exponential smoothing methods, particularly ETS (Exponential Smoothing State Space Model) offers a robust framework by explicitly accounting for level,trend and seasonal

components. ETS models do a weighted average over time, where the weights decaying exponentially into the past, giving much more weight to newer observations.The framework allows for both additive and multiplicative specifications,hence being flexible for different types of time series. The state space formulation provides a more accurate way of parameter estimation; hence it can handle the missing observations and outliers in the series efficiently. These models have broad applications in business and economic studies due to their good performance in dealing with the trend and seasonality (Hyndman and Athanasopoulos, 2018).

For time series data containing inherent seasonal patterns, Seasonal ARIMA (SARIMA) is a powerful augmentation of ARIMA. SARIMA adds seasonal autoregressive (SAR), seasonal differencing (SD), and seasonal moving average (SMA) components into the non-seasonal parts of the ARIMA model, allowing it to capture periodic patterns at repeated intervals (e.g., monthly or quarterly).This model is particularly useful for forecasting in applications like retail sales, weather patterns, and economic data,which usually possess some form of seasonality (Box et al., 2015); (Hyndman and Athanasopoulos, 2018).

Seasonal-Trend Decomposition using Loess (STL) comes into play as a complementary approach as SARIMA can be less flexible when seasonality changes over time.STL is a non-parametric decomposition method that separates the time series into three components: seasonal, trend and residual. These enhance the ability to identify the time-varying seasonal dynamics with high accuracy. This approach utilizes local polynomial regression (LOESS) to progressively smooth each component and is hence useful not only for exploratory data analysis but also in cases when seasonality is changing or structural changes may have affected the seasonal patterns (Cleveland et al., 1990).

The Vector Autoregressive (VAR) model is is broadly recognized for its ability to examine dynamic connections between various time series while refraining from enforcing theoretical causal frameworks. Its strength enables scholars and practitioners to

identify temporal interrelationships among variables, rendering it a crucial instrument in macroeconomic and financial investigations.

Throughout the years, progress has enhanced the VAR framework to tackle new challenges encountered in multivariate time series analysis. Time varying parameter VAR models with stochastic volatility (TVP-VAR-SV) are more adaptable to structural changes and heteroscedasticity, allowing subtle analyzes of economic shocks and financial volatility (Primiceri, 2005). Similarly, Bayesian VAR (BVAR) models cope with overparameterization by introducing prior distributions, leading to increased precision in forecasting and computational efficiency (Koop and Korobilis, 2010). These advancements have facilitated the application of VAR in various fields, including inflation dynamics, financial stability, and spillovers in energy markets, thus providing valuable insights into sectoral connections and guiding regulatory frameworks (Nakajima, 2011); (Diebold and Yilmaz, 2012).

Notwithstanding these improvement, traditional VAR models face encounter difficulties in high-dimensional contexts, where the quantity of variables and lags may surpass the number of available observations. This situation has prompted the incorporation of regularized techniques such as lasso and hierarchical lag structures (HLag). These techniques alleviate issues related to overparameterization and improve interpretability, particularly in datasets characterized by complex covariate interactions. Nevertheless, numerous regularized methods reduce modeling complexity by presuming uniform lag orders, potentially limiting their ability to adequately represent intricate interdependencies (Nicholson et al., 2020); (Basu and Michailidis, 2015).

Recent studies underscore the insufficient examination of regularized VAR models within datasets characterized by multiple covariates, particularly in the realms of macroeconomics and energy markets. Methodologies such as HLag integrate lag selection directly into the regularization framework, enhancing forecast accuracy and providing outputs that are easy to interpret. However, their wider applicability to real-world datasets is still constrained, indicating potential avenues for additional research (Nicholson et al., 2020); (Basu and Michailidis, 2015).

2.2 Regularization in Regression Models

Ridge, Lasso, and Elastic Net regularization techniques have become paramount in regression analysis as they help address the challenges of overfitting, multicollinearity, and variable selection while dealing with high dimensional data.The justification for these methods leans on penalizing model complexity to enhance predictive accuracy and stability. By adding constraints to the regression coefficients, these methods produce models that generalize better to unseen data.

Ridge Regression, or L_2 regularization was first proposed by Hoerl and Kennard (1970) and has since become seminal in dealing with multicollinearity in regression models by adding an L_2 penalty to the sum of squared residuals. It shrinks regression coefficients thus reducing their variance and improving the robustness of the model. Recent works have revisited ridge regression in order to address problems related to high-dimensional contexts by proposing a number of techniques, such as implementation of de-biased and thresholded ridge regression targeting model selection, estimation of prediction intervals, among others. Zhang and Politis (2022) applied a hybrid bootstrap algorithm - theory developed for prediction intervals together with evidence of its practical robustness. Other studies have explored rank-based ridge regression, which relaxes the normality assumption of data and provides robustness against outliers, hence finding its strong applications in fields such as bioinformatics and finance (Emura et al., 2024).

Lasso, or L_1 regularization,was proposed by Tibshirani (1996), which ervolutionized regression modeling by introducing an L_1 penalty that allows variable selection by shrinking some coefficients exactly to zero. Its utility in sparse modeling found wide applications in fields like genomics, where the data is intrinsically high dimensional. Adaptive Lasso, an extension presented by Zou (2006), further enhanced the variable selection by weighting penalties differently across predictors, hence increasing consistency. In the area of machine learning , scalable solutions such as coordinate descent algorithms have made Lasso efficient for huge datasets. The technique has been applied in fields ranging from economics to natural language processing, proving to be vital

for interpretable and parsimonious models (Tibshirani, 1996);(Zou, 2006) and (Hastie et al., 2009).

Elastic net regression, introduced by Zou and Hastie (2005), integrates the advantages of both ridge and Lasso by combining L_1 and L_2 penalties. Hence,it is really efficient in case of multicollinear data, or when the selection of group predictors is required. Recent studies have highlighted its applicability in high dimensional genomic data analysis where the variables are highly intercorrelated. Elastic net was also proven by Zou and Hastie (2005) to address Lasso's limitation in selecting groups of correlated variables, hence becoming very popular for datasets with nested group structures. Cross-validation techniques have been further integrated to optimize the trade-off between penalties, thus improving its predictive accuracy in applications like healthcare and finance (Zou and Hastie, 2005); (Hastie et al., 2009).

These regularization techniques include slight increases in bias for a more significant decrease in variance with the aim of predictive accuracy on new data. Ridge, Lasso, and Elastic Net enable the models to adapt flexibly to complex data structures and hence offer a method which is both predictive and interpretable. Put all together, these methods form a powerful framework for regression analysis and allow feature selection and model stabilization — crucial for robust and reliable forecasts and insights.

2.3 Applications of Regularization in Time Series Forecasting

Regularization methods like Lasso, Ridge, and Elastic Net, have gained prominence in time series analysis, as they offer solutions to the problem of multicollinearity, overfitting, and high dimensionality.

Lasso(L1-regularization) has proven to be beneficial in the development of sparse models by purposefully reducing the influence of non-essential variables to zero. Medeiros and Mendes (2016) broadened its applicability to high dimensional time series forecasting,

introducing adaptive Lasso (adaLasso) for effective model selection and prediction. This methodology demonstrated superior efficacy in predicting U.S. inflation, surpassing conventional ARIMA models by accommodating non-Gaussian and heteroskedastic error distributions (Medeiros and Mendes, 2016).

In the study of renewable energy forecasting, Cavalcante et al. (2016) utilized a Lasso-based vector autoregression (VAR) approach to predict wind power over very short time horizons. This methodology leveraged spatio temporal information coming from different wind power plants in order to achieve scalability and computational efficiency. Their Lasso VAR framework is capable of handling such high-dimensional data with better error reduction in forecasts compared to traditional VAR models (Cavalcante et al., 2016).

Ridge regression(L_2-regularization) has been integral to stabilizing parameter estimates in multivariate time series. Ballarin (2024) explored its role in VAR modeling for economic inference, emphasizing its benefits for estimating impulse response functions. The Ridge method was particularly effective in reducing bias and variance, especially when parameters exhibited anisotropic penalization. Cross-validation techniques validated the optimal penalty selection for practical applications (Ballarin, 2024).

Elastic Net combines the strengths of both L_1 and L_2 penalties, addressing the weaknesses of the Lasso, which often realizes poor performance under correlated predictors. Zou and Hastie (2005) noted its adaptability, especially in economic time series which is often plagued by several challenges such as groupings of variables and multicollinearity. Elastic Net has also been applied to generalized linear models (GLMs) for for handling both the dense and sparse predictor matrices for an enhanced economic and financial forecasting accuracy (Zou and Hastie, 2005).

The advancements in regularization methods have really transformed both univariate and multivariate high dimensional settings for time series forecasting. However, their application to time series data with multiple covariates, such as regularized VAR models, remain largely an area of research that is rather unexploited. Farther studies in

this respect can indicate how complex relationships among the covariates are handled while keeping model interpretability and computational efficiency.

Chapter 3

Methodology

3.1 Data Collection

The obtained dataset is the Walmart sales data sourced from Kaggle. It contains weekly sales data of various stores combined with associated covariates such as temperature, fuel price, consumer price index (CPI), unemployment rate, and holiday flags. This dataset is well-suited for time series analysis due to its temporal granularity and multivariate nature. The dataset comprises of 6,435 observations that cover various weeks and different store locations, rendering it particularly suitable for examining temporal trends and the interactions among variables over time.

The dataset contains several key variables. The target variable being analyzed is the 'Weekly Sales' variable, which is the actual sales generated by a store on a weekly basis. The covariates are Holiday Flag - a binary variable which takes on a value of 1 for weeks that contain holidays and 0 otherwise; Temperature - the average weekly temperature of area store is located; Fuel Price - the average cost of fuel ; CPI - the consumer price index of the week, reflecting the level of economic conditions; Unemployment, denoting the unemployment rate. Moreover, Store serves as an identifier for individual store locations and there are 45 in total, while Date represents the year, month, day and date the data was recorded, and the interval between each date is one week. Figure A.1 in appendix A shows a snippet of the dataset.

CPI is treated as a covariate in this study as it primarily reflects broader economic conditions rather than being directly influenced by the sales. It serves as an external factor that may impact consumer purchasing power but is not inherently driven by the sales performance of the individual stores. However, CPI could be considered an

endogenous variable if there is strong feedback between retail sales and inflationary trends. For instance, if rising sales contribute to increased demand and higher prices, CPI might exhibit a dynamic relationship with sales justifying its inclusion as an endogenous variable. For this study, CPI is assumed to be exogenous ensuring that the model focuses on how macroeconomic conditions influence sales without assuming a direct bidirectional dependence.

This dataset was selected due to its multivariate structure and the inclusion of covariates that are relevant for time series modeling. It lends itself well to investigating complex relationships among variables through Regularized Vector Autoregression (VAR) methodology due to the realistic setting of the economic and retail environment it models. Its completeness allows the study of how the exogenous factors - economic conditions and weather - relate to retail sales over time.

Before analysis, the dataset will undergo several preprocessing steps to ensure suitability for the VAR framework. The Date column will be converted into a proper datetime format to facilitate time series analysis. Variables will be tested for stationarity, and any necessary transformations, such as differencing or detrending, will be applied. To enhance comparability and improve model performance, the data will also be standardized. Furthermore, the data may be aggregated or segmented by store if required by the VAR modeling assumptions. Missing or irregular values will be cleaned for maintaining the sanctity of the data. This cleaned dataset forms the very foundation for using the Regularized VAR method and meeting the objectives of the study.

3.1.1 Model Formulation

Multivariate Time Series with covariates

Consider equation (1.2) from the Introduction, in which Y is a $k \times T$ response matrix, B is a $k \times kp$ matrix of coefficients, Z is a $kp \times T$ matrix of explanatory variables. To include multiple covariates, the model is extended by adding a separate term for

the influence of covariates X_t. The generalized model with the inclusion of multiple covariates would look like this

$$Y_t = \Phi 1' + BZ_t + CX_t + E_t, \tag{3.1}$$

where Y_t is the vector of endogenous variables at time t, $\Phi 1'$ is the intercept terms for each endogenous variable, Z_t is a $T \times kp$ matrix of lagged endogeneous variables, BZ_t is the relationship between the lagged endogenous variables Z_t and the current values of Y_t,CX_t models the influence of exogenous variables (covariates) X_t on the endogenous variables Y_t, where X_t is a $T \times q$ matrix of covariates and E_t is the error term.

Lasso Regression

Introduced by Tibishirani,Lasso introduces a L_1-norm penalty to the regression model,encouraging sparsity by shrinking some coefficients to exactly zero. Lasso-VAR directly applies the L_1-norm regularization to the autoregressive coefficient matrix B, selecting only the most relevant lags and variables. This makes it particularly suited for high-dimensional time series models.

Lasso-VAR is given by

$$\frac{1}{2}\|Y - BZ\|_2^2 + \lambda \sum_{l=1}^{p} \|B^{(l)}\|_1, \tag{3.2}$$

where λ is the tuning parameter which controls the amount of shrinkage.

When we now incorporate covariates, Lasso-VAR with covariates is given by

$$\frac{1}{2}\|Y - \Phi 1' - BZ_t - CX_t\|_2^2 + \lambda_B \sum_{l=1}^{p} \|B^{(l)}\|_1 + \lambda_C \|C\|_1, \tag{3.3}$$

where λ_B controls shrinkage of the lagged endogenous coefficients and λ_C controls shrinkage of the covariate coefficents.

Ridge Regression

Developed by Arthur E. Hoerl and Robert W. Kennard in 1970, in Ridge regularization, we replace the L_1-norm ($\sum_{l=1}^{p} \|B^{(l)}\|$) with the L_2-norm squared ($\sum_{l=1}^{p} \|B^{(l)}\|_2^2$). Ridge-VAR incorporates the L_2-norm regularization to stabilize the estimation of the coefficient matrix B, especially when predictors are highly correlated. This ensures that even small coefficients are retained, which can be useful in dense systems.

Ridge-VAR is given by

$$\frac{1}{2}\|Y - BZ\|_2^2 + \lambda \sum_{l=1}^{p} \|B^{(l)}\|_2^2, \tag{3.4}$$

where $\lambda \geq 0$ is the tuning parameter controlling the regularization.

Ridge-VAR with covariates is given by

$$\frac{1}{2}\|Y - \Phi 1' - BZ_t - CX_t\|_2^2 + \lambda_B \sum_{l=1}^{p} \|B^{(l)}\|_2^2 + \lambda_C \|C\|_2^2, \tag{3.5}$$

where $\sum_{l=1}^{p} \|B^{(l)}\|_2^2$ and $\|C\|_2^2$ are the Ridge penalties for the lagged endogenous variables and the covariates, respectively.

Elastic-Net Regression

Introduced by Hui Zou and Trevor Hastie,Elastic-Net combines L_1-norm (Lasso) and L_2-norm (Ridge) penalties, balancing sparsity and stability by retaining correlated predictors while still encouraging some sparsity.ElasticNet-VAR applies this combined penalty to the VAR framework, making it ideal for situations where the variables are numerous and correlated, but some sparsity is still desirable.

ElasticNet-VAR is given by

$$\frac{1}{2}\|Y - BZ\|_2^2 + (1-\alpha)\lambda \sum_{l=1}^{p} \|B^{(l)}\|_1 + \alpha\lambda \sum_{l=1}^{p} \|B^{(l)}\|_2^2. \tag{3.6}$$

This approach uses two tuning parameters: λwhich controls the overall regularization strength and α which balances the effects of Lasso and Ridge. α serves as the mixing parameter that smoothly transitions between the two types of regularization.When $\alpha = 1$, the model corresponds to Ridge regularization, while when $\alpha = 0$, it corresponds to Lasso regularization. Therefore, it is crucial to set α within the range of 0 to 1.

ElasticNet-VAR with covariates is given by

$$\frac{1}{2}\|Y - \Phi 1' - BZ_t - CX_t\|_2^2 + (1-\alpha)\lambda_B \sum_{l=1}^{p} \|B^{(l)}\|_1 + \alpha\lambda_B \sum_{l=1}^{p} \|B^{(l)}\|_2^2 + (1-\alpha)\lambda_C \|C\|_1 + \alpha\lambda_C \|C\|_2^2, \tag{3.7}$$

where λ_B and λ_C separate regularization strengths for the lagged endogenous coefficients and covariate coefficients respectively; and α is a mixing parameter between lasso and ridge penalties and should be between 0 and 1.

While the formulations of the Lasso-VAR, Ridge-VAR and ElasticNet-VAR models are presented above, the detailed optimization procedures used for parameter estimation are provided in A.1 in Appendix A. These include the algorithmic steps based on coordinate descent tailored for models with covariates.

3.2 Model Training and Optimization

For each of the datasets, we fit the following models and evaluate their performance Unregularized VAR, Lasso-VAR with covariates, Ridge-VAR with covariates and ElasticNet-VAR with covariates.

The models are developed with R software using the glmnet package, with the exception of the elastic net model. The elastic net requires an additional package, caret, since it involves two tuning parameters.

Selecting Penalty Parameter

Optimization of the regularized parameters for VAR with covariates is done with K-fold cross-validation to ensure that the model generalizes well by minimizing the out-of-sample prediction error.

The data can be partitioned into K equal-sized folds. On each run, the model is trained on K-1 folds and tested on the remaining fold. The procedure is repeated K times such that every run selects a different fold as the test. Finally, the average of all K test errors is taken as an estimate of the out-of-sample performance of the model.

The cross-validation procedure has been conducted in order to tune the regularization parameter(s). The aim is to find the value of the penalty parameter (or parameters depending on the type of regularization technique used) that minimizes the average out-of-sample prediction error across all folds. The penalty parameter controls the degree of shrinkage imposed on the model coefficients, and, therefore, there is a balance between model complexity and predictive accuracy.

For each fold, the model is trained with a candidate value of the penalty parameter. The parameter that results in the lowest average prediction error across the folds is chosen as the best regularization parameter. In the case of ElasticNet, the cross-validation process involves optimizing two parameters instead of just one, which adds an extra layer of flexibility in capturing relationships in the data by combining L_1 and L_2 regularization.

Optimal lag order

The choice of lag order is, of course, crucial to questions of how much history is embedded in the dynamics of the model.Two very popular approaches for selecting the lag order include the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC); AIC penalizes model complexity based on the likelihood function,

where the lower the AIC, the better the model and BIC which is similar but penalizes heavily for model complexity, in particular for higher lag orders.

The lag order p is chosen by computing AIC and BIC for a range of possible lag lengths - from 1 to some maximum value. For each lag order, the corresponding values of AIC and BIC will be computed, and the lag order that minimizes these criteria is selected as the optimal lag length for the model. That would mean the model is not over-complex yet it will retain enough time-dependent relations within the data.

Both AIC and BIC balance model fit against complexity but AIC favors more parameter-heavy models, whereas BIC is generally more conservative and sticks with simpler models. Both are going to be computed here and used in tandem so that the chosen lag order strikes the best balance for the forecast's accuracy.

3.3 Performance Evaluation Metrics

The performance results of the unregularized Vector Autoregression model are compared to those from the regularized VAR models by applying the performance metrics : Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE).

The unregularized VAR is used as the base model in capturing the linear dependencies of multiple time series variables over time without applying any regularization. Its performances on the metrics will offer a baseline to evaluate the efficiency of regularized VAR models in the reduction of forecasting error.

The Mean Absolute Error(MAE) measures the average magnitude of absolute errors between the predicted and actual values. MAE is given by

$$\frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|, \tag{3.8}$$

where n is the number of observations,y_t is the actual value at time t and $\hat{y}_t$ is forecasted value at time t .

Root Mean Squared Error (RMSE) quantifies the average squared difference between forecasted and actual values, giving emphasis on larger errors due to squared terms. RMSE is given by

$$\sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}. \tag{3.9}$$

Mean Absolute Percentage Error(MAPE) is the average absolute error as a percentage of the actual values which gives a scale-independent measure of forecast accuracy. It is given by

$$\frac{100}{n}\sum_{t=1}^{n}|\frac{y_t - \hat{y}_t}{y_t}|. \tag{3.10}$$

The model that shows the lowest values for these metrics will be considered the most accurate and effective.

3.4 Variable Selection and Interpretation

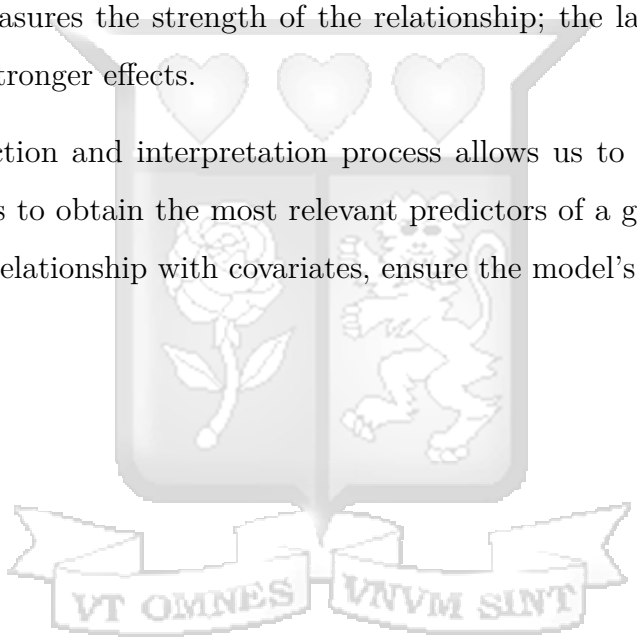
In the unregularized VAR model, the estimation for all the variables is done, and hence the relationships of predictors with the target variable can be directly seen. However, this might cause some problems; for example, overfitting may occur when there are lots of covariates. Because of this, regularized VAR models will be used that add penalties for shrinking the coefficients of less relevant predictors, hence more stable and interpretable results.

Instead, Lasso and Elastic Net models do the variable filtering by shrinking some coefficients to zero using L_1 and combined L_1/L_2, respectively. In that case, the covariates with non-zero coefficients in the forecast are considered the significant ones by the two models. On the contrary, Ridge does not filter variables but penalizes the size of the coefficients, therefore allowing for an overview of relative predictor importance.

To evaluate the statistical significance of the coefficients, hypothesis tests such as t-tests have been utilized to identify any coefficient that is statistically different from zero. Coefficients whose p-values are less than the cutoff value, often 0.05,are considered statistically significant, and might imply that the respective variables importantly affect the target variable. In the case of regularized models, nonzero coefficients mean the important predictors, and zero coefficients reflect variables that contribute little.

The magnitude and the signs of the coefficients will be carefully examined. A positive coefficient would imply a direct relationship, whereas a negative coefficient would show that the covariate and dependent variable are inversely related. The magnitude of the coefficient measures the strength of the relationship; the larger coefficients are indicative of the stronger effects.

The variable selection and interpretation process allows us to select and interpret the variables so as to obtain the most relevant predictors of a given target variable, understand their relationship with covariates, ensure the model's generalizability and interpretability.



Chapter 4

Results and Interpretation

4.1 Introduction

This chapter presents the results of the study from data preprocessing, exploratory analysis, model development and model evaluation. Several transformations are done to structure data for time series models. Descriptive statistics provide an overview of the distribution of sales across the 45 stores and the external covariates. Stationary checks are conducted using Augmented Dickey-Fuller (ADF) test. The modeling process explores both unregularized and regularized Vector Autoregression (VAR) models. The unregularized VAR encountered challenges because of multicollinearity hence variable selection was done based on the Variance Inflation Factor (VIF). Regularized VAR models (Ridge,Lasso and Elastic Net) are applied and forecasting accuracy was evaluated. To show that regularized VAR models deals with multicollinearity, they were tested on the full dataset. Performance metrics,RMSE,MAPE and MAE are reported.

4.2 Descriptive Statistics

The data consists of weekly sales for 45 Walmart stores together with several external factors that influence sales. In this study, the dependent variable Y is weekly sales. To structure data for time series analysis, the data was pivoted into a wide format where each store's sales were represented as a separate column instead of a being stored in a single variable. This allowed the Regularized VAR model to analyse store level sales trends individually while also capturing dependencies between stores over time.

The Exogenous covariates (X_t) included macroeconomic indicators fuel price, consumer price index (CPI) and unemployment rate which were aggregated as weekly averages across all stores. Additionally, temperature was included as a weather based covariate, reflecting seasonal variations that might impact consumer behavior. A holiday flag was also incorporated, taking a value of 1 if any store experienced a holiday week, capturing the effects of holidays on sales fluctuations.

The aggregation of these variables as weekly averages was a modeling choice that ensured the covariates represented broader economic and environmental conditions rather than store specific fluctuations. This assumption aligns with the fact that macroeconomic indicators and general weather conditions tend to influence consumer behavior at a regional or national level rather than affecting individual stores in isolation.The date column was formatted and used as the primary time index.

After data transformation, the final data consisted of 51 variables which include 45 weekly store sales as the endogenous variables Y_t , the date column and the aggregated exogenous variables X_t.

Temperature ranges from 30.48°F to 82.18°F, capturing seasonal variations. Fuel prices fluctuate between $2.67 and $3.99 per gallon. CPI ranges from 167.5 to 176.7, while unemployment varies between 6.95% and 8.62%.

Table 4.1 shows descriptive statistics for exogenous covariates.

Table 4.1: Descriptive Statistics for Exogenous Variables

Statistic	Temperature (°F)	Fuel Price ($)	CPI	Unemployment (%)
Min	30.48	2.67	167.5	6.95
1st Qu.	47.72	2.885	168.4	7.51
Median	61.05	3.49	171.4	8.15
Mean	60.66	3.36	171.6	8.00
3rd Qu.	74.70	3.73	174.7	8.43
Max	82.18	3.99	176.7	8.62

For store sales, the descriptive statistics for the top five and bottom five performing stores are presented in Table 4.2 , while the full table with all stores is provided in the Table A.1 in Appendix A. Store sales vary significantly, with Store 20 exhibiting the highest average weekly sales ($\sim$ 2.1M) and Store 33 among the lowest ($\sim$ 260K).

Table 4.2: Descriptive Statistics for Top 5 and Bottom 5 Performing Stores

Store	Min	1st Qu.	Median	Mean	3rd Qu.	Max
Top 5 Stores						
Store 20	161017	1950866	2053165	2107677	2155186	3766687
Store 4	1762539	1929611	2073951	2094713	2175039	3676389
Store 14	1479515	1873298	2004330	2020978	2125780	3818686
Store 13	1633663	1877476	1958824	2003620	2041469	3595903
Store 2	1650394	1803501	1879107	1925751	1956927	3436008
Bottom 5 Stores						
Store 33	209986	242492	258427	259862	275236	331174
Store 44	241937	283882	298080	302749	319938	376234
Store 5	260637	294696	310338	318012	329861	507900
Store 36	270678	320540	373268	373512	426490	489372
Store 38	303909	350366	380870	385732	414198	499268

4.3 Time Series Plot

Sales Trends for the top five and bottom five stores as per total sales was plotted. Figure 4.1 shows time series plot for the top 5 Walmart Stores. From the plot, we can see consistently high sales levels with significant spikes during holiday seasons indicating heavy reliance on holiday driven revenue. Outside the peak periods, the sales hover between 1.8 and 2.5 million suggesting a stable weekly performance with minor fluctuations. Store 4 and Store 20 have relatively higer baseline sales suggesting stronger operational consistency. Store 14 exhibits greater volatility especially in the

later months which can be due to sensitivity to external shocks. Store 2 has lower baseline sales bust still experiences robust holiday peaks.

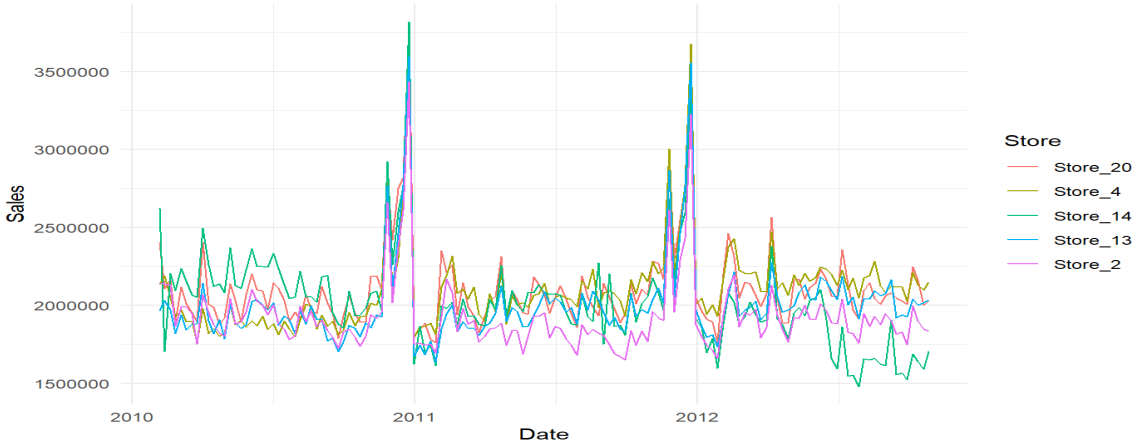


Figure 4.1: Time series plot for Top 5 Stores

However, the bottom five stores in Figure 4.2 show different sales behaviour. While holiday spikes are still present, they are less dramatic and do not show the same sharp peaks as in the top stores. Store 38 has a more pronounced seasonal pattern and a relatively higher baseline sales while Store 36 demonstrates a gradual decline in sales hinting at demand related issues. Store 33 and store 44 show steady but a flat performance which may suggested limited market expansion. Unlike the top stores, the bottom 5 stores exhibit a more visible cyclic pattern without the overwhelming holiday influence.

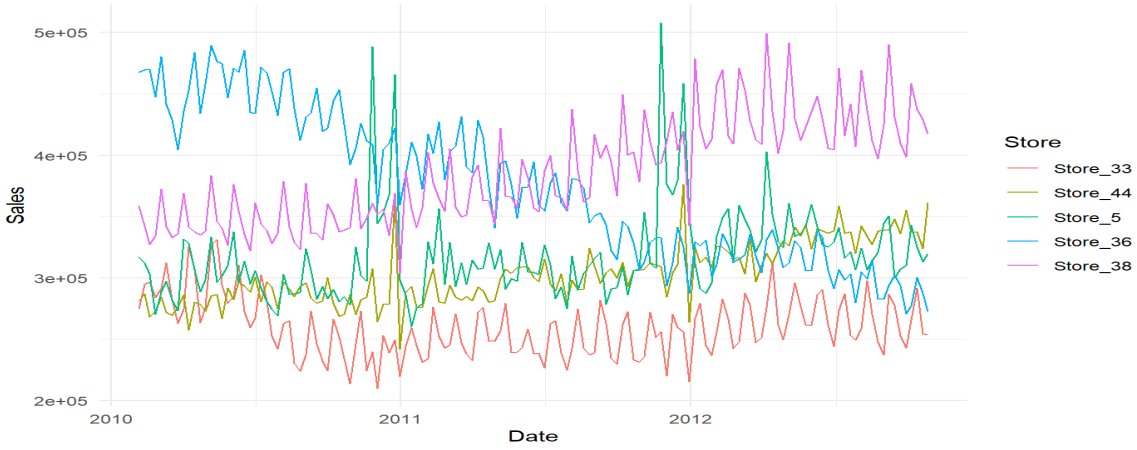


Figure 4.2: Time series plot for Bottom 5 Stores.

Figure 4.3 shows plots for the exogenous variables. There is a clear seasonal pattern for temperature as it peaks around mid year and drops during winter months. This aligns with the cyclic sales patterns suggesting weather sensitive products and seasonal shopping behaviour resulted to the periodic increase and decrease in weekly sales. There is a general upward trend for fuel prices with some fluctuations suggesting suggests external market conditions influence prices and there may be periods of volatility. The higher fuel prices could decrease consumer spending partially explaining the increased variability in sales during those periods. The CPI shows a steady increasing trend over time which indicates inflationary pressures in the economy. The rising proces may make customers adjust their spending habits flattening trends in sales for some stores. The unemployment rate follows a stepwise downward trend suggesting a gradual improvement in employment conditions over time. This would suggest improved economic conditions which could boost consumer confidence in spending leading to increased sales observed in some stores.

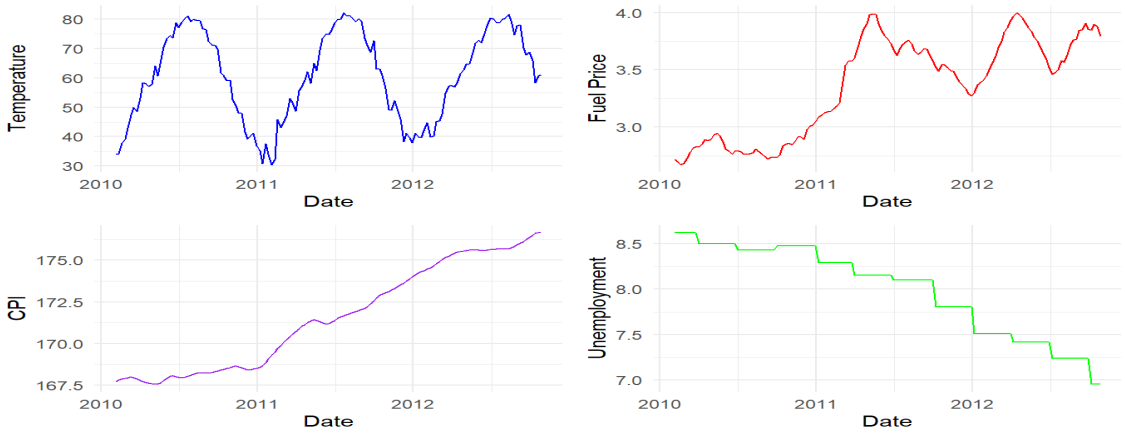


Figure 4.3: Time Series Plots for Covariates

4.4 Stationarity

The Augmented Dickey-Fuller (ADF) test was conducted to test the stationarity of the endogenous variables (store sales) and the exogenous variables. The results indicated that some of the store sales series (Stores 30, 33, 36, 38, 42, 43, and 44) were non-

stationary so were the exogenous variables Fuel Price, CPI, and Unemployment. Since stationarity is a prerequisite for VAR modeling, the necessary transformations were applied to meet this requirement.

Uniform differencing was applied to both endogenous variables and exogenous covariates to maintain consistency across the multivariate VAR system. This approach was intended to ensure that all series were integrated to the same order, preserving the system's compatibility within a VAR framework. All variables were subjected to first-order differencing which involved subtracting each observation from the previous one to remove trends and stabilize the mean of the series. After application of first differencing, some exogenous covariates such as Temperature, Fuel Price and CPI remained non-stationary. To address this, second-order differencing was applied to enforce stationarity across the system. The results of the stationary test are in the Table A.2 in appendix A.

Following these transformations, checks for possible overdifferencing was done using Ljung-Box test for serial correlation and variance analysis. Potential overdifferencing was seen in some exogenous covariates namely Temperature (Ljung–Box p-value = 0.0000; variance ratio = 0.1030), Fuel Price (p-value = 0.0004; variance ratio = 0.0088), CPI (p-value = 0.0053; variance ratio = 0.0001), and Unemployment (p-value = 0.0001; variance ratio = 0.0228) showing very low variance and diminished serial dependence suggesting potential loss of informative signal. This suggests that while uniform differencing was maintained to ensure methodological rigor and system-wide consistency in the VAR framework, it may have compromised the integrity of some variables by overdifferencing series that were already stationary or close to stationary. This could potentially attenuate important dynamic relationships and reduce the explanatory power of those variables. A more tailored transformation strategy, where each series is transformed based on its individual stationarity characteristics, might better preserve valuable signal while still ensuring overall model stability. Though not implemented in this study for consistency, such an approach warrants further

investigation in future research particularly for models incorporating mixed-integration series. Results of the check for overdifference are in Table A.3 in appendix A.

4.5 Model Building

4.5.1 Optimal Lag Order Selection

The VARselect() function in R was employed to determine the optimal lag length using several key criteria viz; the Akaike Information Criterion (AIC), Schwarz Criterion (SC or BIC), Hannan-Quinn Criterion (HQIC) and Final Prediction Error (FPE). The results indicated that AIC, HQIC and FPE suggested a lag length of 10 favoring a model that captures long-term dependencies and complex dynamics. In contrast, BIC recommended a lag length of 3 prioritizing parsimony to prevent overfitting. Given that multiple criteria converged on a lag length of 10, this was selected for building the models. Although BIC suggested shorter lag lengths for more parsimonious models, priority was placed on capturing rich temporal structures and maximising predictive accuracy making lag length of 10 the most optimal choice.

4.5.2 Model Formulation

An initial Vector Autoregression (VAR) model was estimated using all the variables but encountered a singularity matrix error. A singularity matrix error arises when the predictor matrix cannot be inverted, and this typically occurs with perfect collinearity among the variables, typically caused by multicollinearity. To address this issue, a Variance Inflation Factor (VIF) analysis was conducted, and variables with a VIF greater than 10 were removed to help reduce multicollinearity. The VIF results are in Table A.4 in appendix A.

The VAR model was then re-estimated using the reduced set of variables which include the endogenous variables Stores 1,17,30,36,37,38,43 and 44 and the covariates Temperature, CPI, Fuel price, Unemployment rate and Holiday flag.

The model was estimated with a constant term, and the inclusion of exogenous variables provided additional explanatory power.This formulation serves as the foundation for the subsequent application of Ridge, Lasso, and Elastic Net regression to further improve predictive performance.

To effectively fit the regularized VAR models,the data was preprocessed by constructing lagged endogenous variables up to lag order 10 and aligning them with exogenous covariates to form the predictor matrix. Missing values introduced due to lagging were removed to ensure consistency across all models. Figure A.2 in appendix A shows a snippet of the final structured dataset.

To assess the effectiveness of regularization techniques in handling multicollinearity, Ridge VAR, Lasso VAR and Elastic Net VAR were also applied to the full dataset without removing highly collinear variables.

Selection of Penalty Parameter

To select the optimal penalty parameters for ridge, Lasso and Elastic Net models, cross validation was used. This allows for balancing model complexity with predictive accuracy ensuring that the model generalizes well to unseen data. The goal was to minimize the cross-validated mean squared error (CVMSE) and identify the corresponding optimal values for the penalty parameters(λ for Ridge and Lasso and both λ and α for Elastic Net).

For Ridge and Lasso models, `cv.glmnet` function in R was used to determine the optimal lambda that minimizes the CVMSE. Table 4.3 shows the cross-validation results for each endogenous variable in Ridge VAR model. In this study, the Ridge model was built using lambda.1se to avoid overfitting and promote more stable coefficient estimates across all endogenous series while still maintaining strong predictive performance.

Table 4.3: Optimal λ and Cross-Validation MSE for Endogenous Variables (Ridge)

Endogenous Variable	**Optimal λ**	**CV MSE**
Store_1	12.641573	38136267782
Store_17	11.354012	18607538332
Store_30	9.067145	486029711
Store_36	8.344653	327283734
Store_37	9.779605	635827723
Store_38	10.017530	854628729
Store_43	9.574481	1343376520
Store_44	9.879046	514238784

Table 4.4 shows the cross-validation results for each endogenous variable in Lasso VAR model.

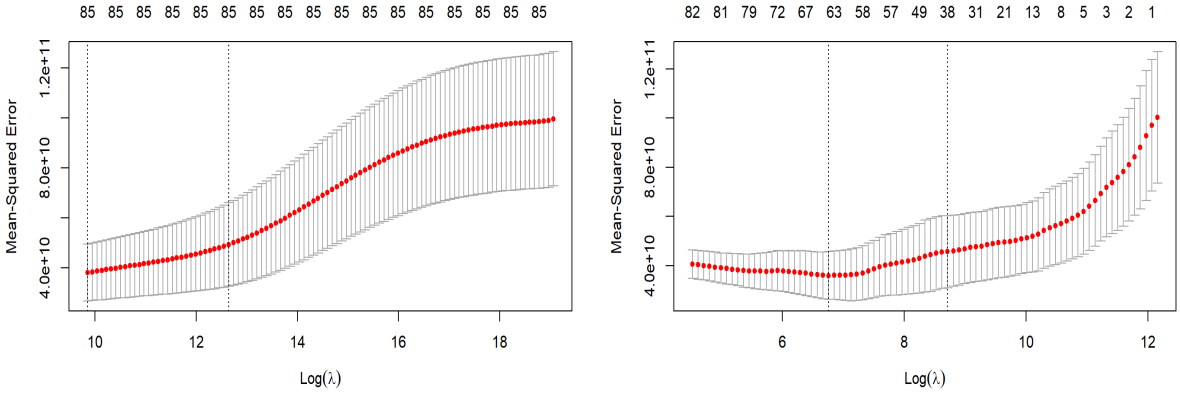
Table 4.4: Optimal λ and Cross-Validation MSE for Endogenous Variables (Lasso)

Endogenous Variable	**Optimal λ**	**CV MSE**
Store_1	8.7109	35922401061
Store_17	8.3537	15137665128
Store_30	6.9041	386933936
Store_36	6.3677	254084794
Store_37	6.0350	430742993
Store_38	5.9938	606585574
Store_43	5.7368	855239011
Store_44	7.3439	383658600

Figure 4.4(a) illustrates the cross-validation results for Store 1. The red dots represent the CVMSE for each value of λ on a logarithmic scale. The first vertical dashed line indicates λ_{min}, the value that achieves the lowest cross-validation error. The second vertical dashed line represents λ_{1se}, which corresponds to the largest λ within one standard error of the minimum CVMSE (chosen to be $\lambda = 12$). In this study, λ_{1se}

was selected to achieve a more regularized and stable solution, balancing predictive performance with model simplicity and preventing overfitting.

Figure 4.4(b) illustrates how the cross-validation mean squared error evolves with changes in λ. A smooth upward trend is visible as it increases beyond the optimal range, indicating growing model bias. The relatively flat section near the lower λ values shows where the model balances bias and variance effectively. For Lasso, $\lambda_{1\text{se}}$ which was approximately 8.7 was selected to ensure a more stable and parsimonious model, reducing the risk of overfitting while preserving good predictive accuracy.



(a) (b)

Figure 4.4: Panel (a) shows the Cross Validation Plot for Store 1 (Ridge). Panel (b) shows the Cross Validation Plot for Store 1 (Lasso).

For Elastic Net, a two-step cross-validation approach was employed to determine the optimal combination of α (the mixing parameter between Ridge and Lasso) and λ (the regularization penalty). First, a grid search over a sequence of α values ranging from 0 (pure Ridge) to 1 (pure Lasso) in increments of 0.1 was performed. For each α, a cross-validated Elastic Net model was fit using `cv.glmnet`, and the minimum cross-validation mean squared error (CVMSE) was recorded. The α value that resulted in the lowest CVMSE was selected as the optimal α. Subsequently, for this chosen α, the corresponding optimal λ was identified using the one-standard-error rule (λ_{1se}), providing a more stable and regularized solution. This approach ensured that both

hyperparameters were selected in a data-driven manner, balancing predictive accuracy and model simplicity.

Table 4.5 shows the optimal α and λ for each endogenous variable.

Table 4.5: Optimal Alpha, Lambda, and CVMSE for Elastic Net

Endogenous Variable	**Optimal Alpha**	**Optimal Lambda**	**CV_MSE**
Store 1	0.3	9.263634	28607413161
Store 17	0.9	7.807798	15348394759
Store 30	0.9	5.893066	331297910
Store 36	0.7	5.235821	239461452
Store 37	0.4	6.765220	433318103
Store 38	0.8	5.565728	541590837
Store 43	1.0	5.829873	796754428
Store 44	0.8	7.101851	366205172

Cross-validation was also performed on the full dataset to determine that the selected optimal penalty parameters were reliable and consistent in the overall framework. The overall validation process proved that the lambda and alpha values chosen for every endogenous variable showed stability when evaluated in the overall multivariate context. The in-depth results of this cross-validation on the full dataset are shown in the Appendix A.5 to enhance confidence in the chosen parameters and the forecasting process.

4.5.3 Feature Selection

Feature selection was performed using Lasso VAR and Elastic Net VAR which automatically identified the most relevant predictors by shrinking irrelevant coefficients to zero. Due to the large number of lagged variables in the dataset, listing all selected variables is impractical. Instead, we present the top 10 most frequently selected features across all endogenous variables in Tables 4.6 and 4.7 .

Table 4.6: Top 10 Most Frequently Selected Predictors for Lasso

Predictor	**Frequency**
Store 1 lag5	8
Store 1 lag6	8
Store 17 lag10	8
Store 30 lag3	8
Store 36 lag10	8
Store 37 lag10	8
Store 38 lag10	8
Store 43 lag3	8
Store 44 lag10	8
Unemployment	8

Table 4.7: Top 10 Most Frequently Selected Predictors for Elastic Net

Predictor	**Frequency**
Fuel Price	8
Holiday Flag	8
Store 1 lag2	8
Store 1 lag5	8
Store 1 lag6	8
Store 17 lag1	8
Store 17 lag10	8
Store 30 lag3	8
Store 36 lag3	8
Store 36 lag6	8

The comparison of the top 10 most frequently selected predictors from the Lasso and Elastic Net models provides valuable insights into the primary drivers of store sales. Both models consistently identified lagged sales variables, notably for specific stores

namely Store 1 lag5, Store 1 lag6, Store 17 lag10, Store 30 lag3 , underscoring the strong temporal dependencies and the importance of past performance in forecasting future sales.

Elastic Net demonstrates a greater tendency to retain external covariates such as Fuel Price and Holiday Flag compared to Lasso which enforces a stricter sparsity constraint. This distinction suggests that Elastic Net is better equipped to capture the broader macroeconomic effects while Lasso emphasizes more parsimonious and direct relationships.

The recurrent selection of multiple lags for the same store across both models emphasizes the significance of not only individual predictors but also the lag structure demonstrating that short to medium term historical patterns are critical in explaining sales dynamics. The selection patterns reveal that both internal store level lagged variables and certain external covariates play pivotal roles in predictive accuracy.

Table 4.8 and 4.9 show the top 10 frequently selected predictors for the full dataset where no variables have been removed. Both models highlight the influence of past sales and external covariates like Holiday Flag .

Table 4.8: Top 10 Most Frequently Selected Predictors for Lasso on Full Dataset

Predictor	**Frequency**
Store 17 lag10	42
Holiday Flag	41
Store 28 lag4	40
Store 5 lag8	40
Store 33 lag3	39
Store 38 lag10	39
Store 35 lag4	38
Store 16 lag4	36
Store 5 lag2	36
Store 7 lag2	36

Table 4.9: Top 10 Most Frequently Selected Predictors for Elastic Net on Full Dataset

Predictor	**Frequency**
Store 17 lag10	43
Holiday Flag	41
Store 5 lag8	41
Store 28 lag4	40
Store 38 lag10	40
Store 33 lag3	39
Store 35 lag4	38
Store 16 lag4	37
Store 5 lag2	37
Store 7 lag2	37

4.6 Model Evaluation

Performance was evaluated using RMSE, MAPE and MAE and compared among the models. The Table 4.10 below shows summarized results for dataset where variables with VIF greater than 10 have beem removed. Detailed results for each store are in A.6 in appendix A.

Table 4.10: Comparison of RMSE, MAE, and MAPE across different VAR models.

Model	RMSE	MAE	MAPE
Unregularized VAR	156492.248	115410.540	754.713
Ridge VAR	49794.4183	33942.3897	212.6658
Lasso VAR	39407.4359	27958.6160	212.8482
Elastic Net VAR	34730.0150	24897.0260	182.9038

The Unregularized VAR model performed poorly exhibiting high RMSE, MAE and MAPE . This suggests that the model struggles with prediction inaccuracy which

may be due to overfitting, unstable parameter estimates or multicollinearity among predictors. Upon applying regularized VAR models, there was a significant reduction in errors across all measures indicating improved generalization and stability in parameter estimates.

Ridge VAR significantly improved performance, reducing RMSE to 49,794.42 suggesting better generalization. Lasso VAR did even better (RMSE = 39,407.44) by automatically eliminating irrelevant predictors which also resulted in an improved MAPE. This demonstrates that its focus on the most relevant features enhanced proportional accuracy. Elastic Net VAR delivered the best overall performance with the lowest RMSE , lowest MAE and lowest MAPE of 182.90. This indicates that the combination of L_1 and L_2 penalties provided the best trade-off between bias reduction and feature selection. Although Elastic Net retained slightly less important features which may have had some marginal impact on its percentage accuracy (MAPE), it provided superior overall predictive power compared to Ridge and Lasso.

The averaged performance metrics for the models for the full dataset with multicollinearity is shown in Table 4.11. Detailed results for each store are in A.7 in appendix A.

Table 4.11: Comparison of performance metrics across different VAR models for full dataset.

Model	RMSE	MAE	MAPE
Ridge VAR	184763.0490	110491.4225	455.9238
Lasso VAR	61229.1079	44308.5253	405.7737
Elastic Net VAR	56567.8745	41183.1329	411.6562

When the full dataset was used without removing multicollinear variables, performance deteriorated across all models. Ridge VAR showed a substantial increase in RMSE and MAE suggesting that multicollinearity caused instability in parameter estimates. Lasso VAR improved upon Ridge by enforcing sparsity, but still exhibited higher errors than

in Table 4.10. Elastic Net VAR remained the best performer with RMSE of 56,567.87 showing that the combined regularization helped control the impact of multicollinearity.

4.7 Forecasting

The forecasting performance of the models was evaluated by generating out-of-sample forecasts and comparing them to actual sales data. To achieve this, the dataset was partitioned into a training set, consisting of all observations except the last ten weeks and a test set comprising the most recent ten weeks of data. The models were fitted separately for each store using the training data.

Once the models were trained, forecasts were generated for each store over the test period by applying the fitted models to the test set predictors. The forecasted sales values were then compared to the actual sales in the test set. To evaluate the forecasting accuracy of the models, we compared the predicted store sales against actual sales and assessed performance using RMSE, MAE and MAPE.

Average forecasting metrics of the dataset with variables removed is shown in Table 4.12. Details for each endogenous variable is shown in Appendix A.8 in appendix A.

Table 4.12: Comparison of forecasting performance metrics across different models.

Model	**RMSE**	**MAE**	**MAPE**
Unregularized VAR	286808.756	248350.238	6643.208
Ridge VAR	64516.796	46769.226	917.229
Lasso VAR	73147.765	51229.773	1032.168
Elastic Net VAR	80797.49	63445.53	1313.69

Ridge VAR stands out as the best performing model as it achieves the lowest RMSE, MAE and MAPE displaying the most accurate forecasts with the least prediction error. Lasso VAR also performs well but slightly underperforms Ridge VAR, particularly in terms of MAE and MAPE. Elastic Net VAR shows worse performance than both Ridge

and Lasso VAR on all metrics despite a better performance in in-sample predictions. This suggests that the combination of L_1 and L_2 regularization in Elastic Net may capture relationships effectively within the training data but may not generalize well to unseen data. Unregularized VAR exhibits the worst performance with the highest errors across all metrics indicating the significant improvements in forecasting accuracy offered by regularization methods like Ridge, Lasso and Elastic Net.

Figure 4.5 demonstrates the same as above

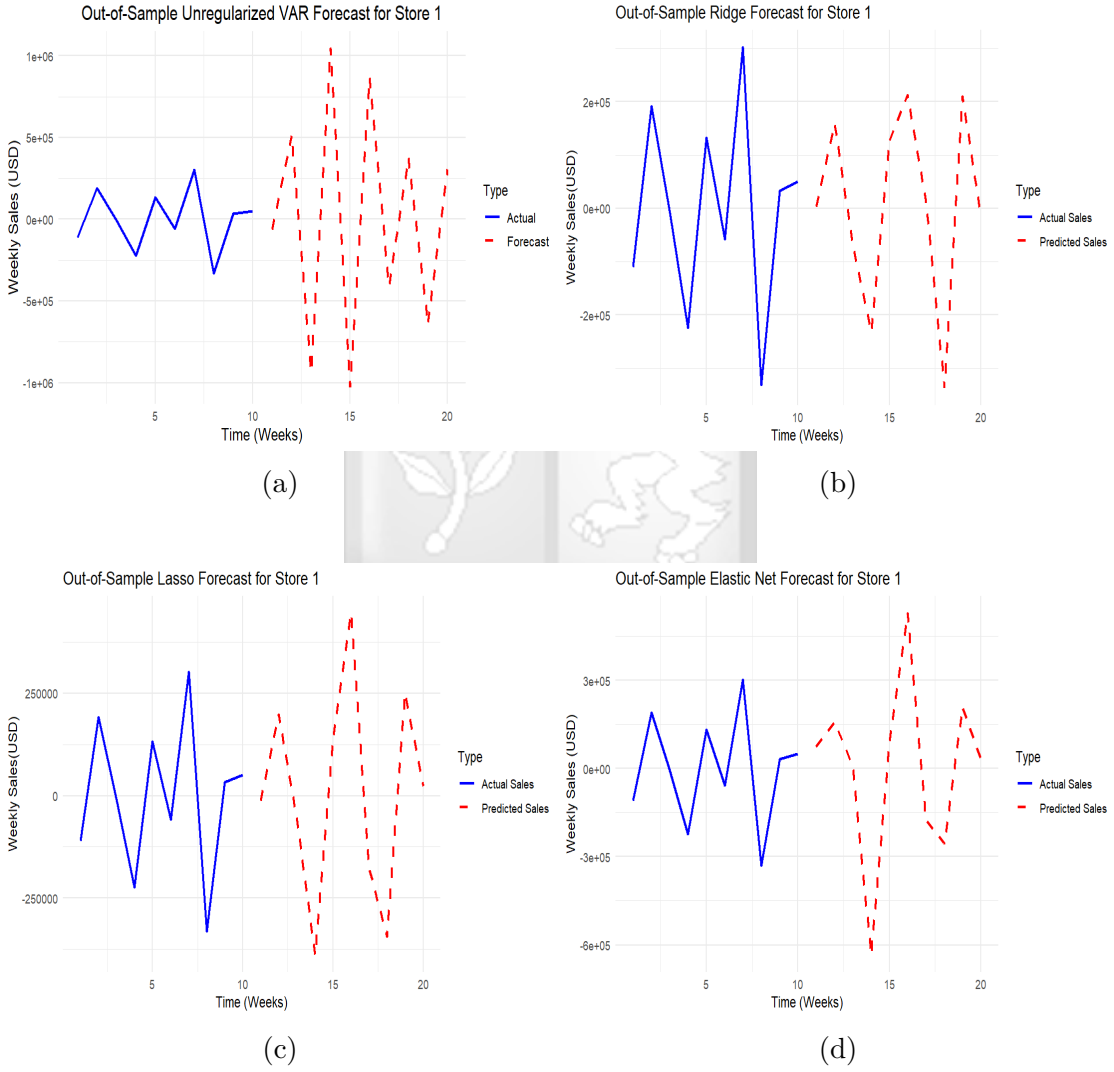


Figure 4.5: Panel (a) shows the Forecasting for Store 1 (Unregularized VAR).
Panel (b) shows the Forecasting for Store 1 (Ridge VAR).
Panel (c) shows the Forecasting for Store 1 (Lasso VAR).
Panel (d) shows the Forecasting for Store 1 (Elastic Net VAR).

The results in Table 4.13 indicate clear differences in how Ridge VAR, Lasso VAR and Elastic Net VAR handle forecast accuracy in full dataset without variables removed. Details for each endogenous variable is shown in Appendix A.9 in appendix A.

Table 4.13: Comparison of forecasting performance metrics across different VAR models on full dataset.

Model	**RMSE**	**MAE**	**MAPE**
Ridge VAR	84730.421	66570.326	1156.467
Lasso VAR	90155.410	74821.134	3898.391
Elastic Net VAR	90237.698	74795.874	3813.091

Based on the forecasting performance metrics, Ridge VAR is the best performing model having the lowest RMSE, MAE and MAPE indicating more accurate and stable predictions. Lasso VAR and Elastic Net VAR both perform worse with higher RMSE and MAE values. Lasso has an RMSE of 90,155 and MAPE of 3,898.391 while Elastic Net has a slightly higher RMSE but a marginally better MAPE. This suggests that Elastic Net does not significantly improve over Lasso and may even introduce additional noise, making it less effective than Ridge.

Figure 4.6 clearly demonstrates the behavior of the models and reinforce the findings. Ridge VAR closely follows the actual sales trend demonstrating its ability to generalize well across different periods. Lasso and Elastic Net exhibit larger deviations from actual sales with frequent mismatches in peaks and troughs. While both models struggle, Elastic Net appears to have even larger discrepancies in certain regions aligning with its higher RMSE. Overall, Ridge VAR is the most reliable model while Lasso and Elastic Net underperform with no significant advantage from Elastic Net over Lasso.

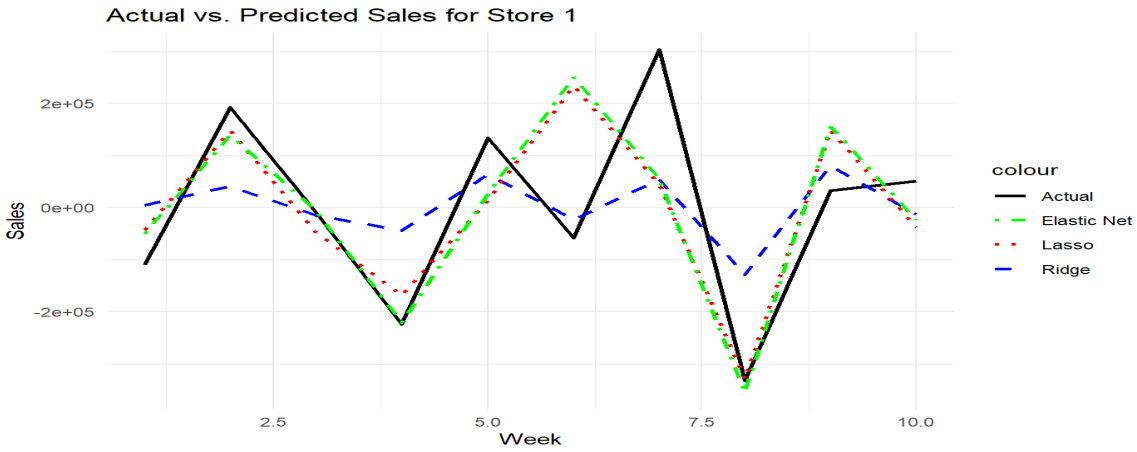


Figure 4.6: Actual vs Predicted for Store 1

Chapter 5

Discussions, Conclusions and Recommendations

5.1 Discussions

The inclusion of external covariates such as unemployment rate, fuel price, CPI, temperature and holiday indicators was motivated by their theoretical and practical relevance to consumer spending behavior and retail operations. Feature selection results from regularized models provide empirical support for their inclusion. Specifically, the Holiday Flag variable consistently emerged as one of the top predictors across both Lasso and Elastic Net models, reflecting the strong influence of holiday periods on sales volumes which often experience spikes due to promotions and seasonal shopping patterns. In the reduced dataset, Unemployment ranked among the top 10 variables in the Lasso model, capturing labor market conditions that affect consumers' disposable income and confidence, thereby influencing retail demand. Similarly, Fuel Price and Holiday Flag were selected by Elastic Net as key predictors, indicating that fluctuations in transportation and logistics costs, as well as seasonal patterns, play significant roles in shaping sales behavior. The Consumer Price Index (CPI), while not always among the top predictors, serves as a broad measure of inflation and cost of living, influencing consumers' purchasing power over time. Temperature acts as a proxy for seasonal weather conditions, impacting shopping patterns for seasonal goods and potentially affecting store foot traffic. Although individual covariate significance varied depending on model specification and dataset scope, the overall set of covariates enriched the model's capacity to capture macroeconomic and temporal dynamics beyond store-level

sales history. This comprehensive approach contributes to improved robustness and forecasting accuracy by incorporating a range of external factors influencing retail sales.

The results of this research show the challenges faced and the performance of Regularized Vector Autoregressive (VAR) models in forecasting multivariate time series that consist of multiple covariates. Originally, when trying to fit the unregularized VAR there was a singularity matrix error likely caused by multicollinearity.To overcome this problem, variables that had a Variance Inflation Factor value higher than 10 were removed, making the estimation of the unregularized VAR possible. However, the high values of RMSE, MAE, and MAPE of the model indicated ongoing overfitting and poor forecasting performance.

To build on this, a comparative analysis of the models showed distinct performance trends. The findings indicate a significant gap between in-sample and out-of-sample forecast accuracy. In particular, the Elastic Net VAR model showed a better performance in in-sample predictions while Ridge VAR performed better than all the other models in out-of-sample forecasting. On the other hand, the unregularized VAR model showed the worst performing results in both scenarios. This reinforces the necessity of regularization in attaining consistent and reliable high-dimensional time series forecasting.

The effectiveness of the Elastic Net VAR in in-sample performance can be attributed to its ability to balance the penalties of Ridge (L_2) and Lasso (L_1), thus allowing it the choice of selecting only the most relevant predictors while at the same time retaining some coefficient shrinkage. This results in a model that effectively fits the training data by removing unnecessary variables while leveraging a wide set of features. However, this does not necessarily translate to out-of-sample forecasting. The extreme shrinkage imposed by the Lasso side of the Elastic Net limits the model's ability to generalize leading to large forecasting errors when applied to unseen data. This is especially evident in time series data sets where complex interdependencies between the time points are crucial and overly sparse models can fail to capture long-term relationships.

On the contrary, the Ridge VAR displayed superior out-of-sample performance reflecting its ability to forecast new data. The Ridge method includes all predictors while at the same time shrinking their coefficients thus minimizing variance without completely eliminating important features. The method ensures the retention of relevant lagged relationships and external covariates in the model leading to predictions with increased generalizability and stability. The lower RMSE, MAE and MAPE for Ridge VAR during out-of-sample tests validate its ability to prevent overfitting making it the most reliable model for real-world predictions.

The unregularized VAR model had the worst in-sample and out-of-sample forecasting performance mostly due to multicollinearity and overfitting issues. Without any penalization, the model overestimated the importance of some features leading to highly volatile parameter estimates. This, in turn, resulted in poor generalization, high variance and significant forecasting errors. The poor performance of the unregularized VAR further supports the necessity of regularization techniques in time series forecasting especially in high-dimensional settings.

These findings reinforce the importance of regularized VAR models in retail forecasting particularly for businesses managing high-dimensional sales data with multiple external factors. While this study focuses on Walmart sales, similar techniques could be adapted for other large-scale retail operations facing challenges in demand forecasting, seasonal trend analysis, and inventory management. The ability of Ridge VAR to provide stable, generalizable forecasts makes it particularly useful for retailers aiming to optimize stock levels, predict consumer demand, and adjust pricing strategies based on external economic conditions.

5.2 Conclusions

This study examined the effectiveness of Regularized Vector Autoregressive (VAR) models in forecasting while addressing key challenges such as multicollinearity, overfitting and high dimensionality. Traditional VAR models proved to be unstable and performed

poorly in both in-sample and out-of-sample predictions. By applying Ridge, Lasso and Elastic Net regularization, the study demonstrated that regularization significantly enhances forecasting accuracy.

Among the regularized models, Ridge VAR emerged as the most reliable, particularly for out-of-sample forecasting as it retained all predictors while shrinking their coefficients ensuring better generalization. Lasso and Elastic Net, although effective in reducing model complexity, showed limitations in long-term forecasting accuracy. Lasso's aggressive feature selection may have eliminated important predictors while Elastic Net, despite balancing L_1 and L_2 penalties, did not significantly outperform Ridge. The study's findings suggest that regularized VAR models, especially Ridge, can be valuable tools for forecasting sales in retail environments, where businesses must account for multiple external factors that influence demand.

5.3 Recommendations

Based on the findings, several recommendations can be made. Retail businesses could adopt Ridge VAR as the primary predictive model, given its superior ability to balance complexity reduction with forecasting accuracy. Ridge has shown capacity to handle high-dimensional time series data hence making it a practical choice for improving sales forecasts and optimizing inventory management. Although Lasso and Elastic Net are useful for feature selection, they should be applied selectively to avoid removing critical lagged variables which may compromise forecast accuracy.

Regularized VAR models can also be extended beyond sales forecasting to other applications in retail such as price prediction, promotional impact analysis and customer demand forecasting. This would enhance decision-making in competitive markets. Lastly, future studies should explore hybrid approaches that integrate regularized VAR models with machine learning techniques such as deep learning or ensemble methods to further improve forecasting accuracy. These hybrid models could capture nonlinear

relationships and dynamic dependencies in sales data providing even greater predictive power for retail businesses.

To further contextualize the performance and practical utility of regularized VAR models, future studies could compare them with more advanced machine learning approaches for time series forecasting, such as Facebook Prophet, Long Short-Term Memory (LSTM) networks or other deep learning models. Such comparisons could help provide deeper insights into the relative strengths of statistical regularization versus data-driven methods in capturing temporal dynamics.

5.4 Limitation of the Study

While this study provides valuable insights into the use of regularized VAR models, it has several limitations. The findings are based on Walmart sales data, which limits the generalizability of results to other industries. While the conclusions may be applicable to retail forecasting, further studies are required to validate these models in sectors such as finance, healthcare or energy markets.

The study considers only a limited set of covariates which are fuel price, CPI, unemployment, temperature and holiday flags. However, retail sales are also influenced by other factors such as advertising spend, competitor pricing and supply chain disruptions which were not included in this analysis. Future research should incorporate a broader set of covariates to improve forecasting accuracy.

Another limitation is the computational complexity of regularized VAR models. Unlike traditional time series models, regularized VAR requires substantial computing resources, particularly during hyperparameter tuning. This could pose challenges for businesses with limited data infrastructure. The use of differencing techniques to ensure stationarity may have led to over-differencing in some variables potentially losing meaningful long-term relationships. Future research could explore alternative approaches to address this issue.

By addressing these limitations and extending the scope of future research, the potential of regularized VAR models in time series forecasting can be further refined, making them even more valuable for businesses and researchers alike.

References

Ballarin, G. (2024). Ridge regularized estimation of var models for inference. *Journal of Time Series Analysis*.

Banbura, M., Giannone, D., and Reichlin, L. (2010). Large bayesian vector auto regressions. *Journal of Applied Econometrics*, 25(1):71–92.

Basu, S. and Michailidis, G. (2015). Regularized estimation in sparse high-dimensional time series models. *The Annals of Statistics*, 43(4):1535 – 1567.

Belsley, D., Kuh, E., and Welsch, R. (1980). *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. Wiley Series in Probability and Statistics. Wiley.

Betzel, R. F., Fukushima, M., He, Y., Zuo, X.-N., and Sporns, O. (2016). Dynamic fluctuations between segregation and integration in human brain functional networks. *Nature Communications*, 7:10340.

Box, G., Jenkins, G., Reinsel, G., and Ljung, G. (2015). *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley.

Cavalcante, R., Brasileiro, R., Souza, V., Nobrega, J., and Oliveira, A. (2016). Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications*, 55.

Chatfield, C. (2016). *The Analysis of Time Series: An Introduction, Sixth Edition*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press.

Choi, E., Bahadori, M. T., Kulas, J. A., Schuetz, A., Stewart, W. F., and Sun, J. (2017). Retain: An interpretable predictive model for healthcare using reverse time attention mechanism.

Cleveland, R. B., Cleveland, W. S., McRae, J. E., and Terpenning, I. J. (1990). Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–33.

Damoiseaux, J. S., Rombouts, S. A. R. B., and Barkhof, F. e. a. (2006). Consistent resting-state networks across healthy subjects. *Proceedings of the National Academy of Sciences*, 103(37):13848–13853.

Diebold, F. and Yilmaz, K. (2012). Better to give than to receive: Predictive directional measurement of volatility spillovers. *International Journal of Forecasting*, 28(1):57–66.

Emura, T., Matsumoto, K., Uozumi, R., and Michimae, H. (2024). g.ridge: An r package for generalized ridge regression for sparse and high-dimensional linear models. *Symmetry*, 16(2).

Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *Journal of Finance*, 25(2):383–417.

Faskowitz, J., Esfahlani, F. Z., Jo, Y., Sporns, O., and Betzel, R. F. (2020). Edge-centric functional connectivity: An approach for studying brain network dynamics. *NeuroImage*, 218:116896.

Hamilton, J. (1994a). *Time Series Analysis*. Book collections on Project MUSE. Princeton University Press.

Hamilton, J. D. (1994b). *Time Series Analysis*. Princeton University Press.

Hansen, J., Ruedy, R., Sato, M., and Lo, K. (2010). Global surface temperature change. *Reviews of Geophysics*, 48:RG4004.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. Springer.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Hyndman, R. and Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*. OTexts, Australia, 2nd edition.

Jhun, I., Fann, N., Zanobetti, A., and Hubbell, B. (2014). Effect modification of ozone-related mortality risks by temperature in 97 us cities. *Environment International*, 73:128–134.

Kanjilal, K. and Ghosh, S. (2021). Asymmetric and regime switching behaviour of GDP and energy nexus in India: new evidences. *Macroeconomics and Finance in Emerging Market Economies*, 14(1):45–65.

Koop, G. and Korobilis, D. (2010). Bayesian multivariate time series methods for empirical macroeconomics. *Foundations and Trends® in Econometrics*, 3(4):267–358.

Medeiros, M. C. and Mendes, E. F. (2016). ℓ_1-regularization of high-dimensional time-series models with non-gaussian and heteroskedastic errors. *Journal of Econometrics*, 191(1):255–271.

Nakajima, J. (2011). Time-varying parameter var model with stochastic volatility: An overview of methodology and empirical applications. *Monetary and Economic Studies*, 29:107–142.

Nicholson, W. B., Matteson, D. S., and Bien, J. (2017). Varx-l: Structured regularization for large vector autoregressions with exogenous variables. *International Journal of Forecasting*, 33(3):627–651.

Nicholson, W. B., Wilms, I., Bien, J., and Matteson, D. S. (2020). High dimensional forecasting via interpretable vector autoregression. *Journal of Machine Learning Research*, 21(166):1–52.

Primiceri, G. E. (2005). Time varying structural vector autoregressions and monetary policy. *The Review of Economic Studies*, 72(3):821–852.

Rabiner, L. and Juang, B. (1993). *Fundamentals of Speech Recognition*. Prentice-Hall Signal Processing Series: Advanced monographs. PTR Prentice Hall.

Shaman, J. and Kohn, M. (2009). Absolute humidity modulates influenza survival, transmission, and seasonality. *Proceedings of the National Academy of Sciences of the United States of America*, 106:3243–8.

Shumway, R. and Stoffer, D. (2011). *Time Series Analysis and Its Applications: With R Examples*. Springer, 9 edition.

Simonsen, L., Viboud, C., Taylor, R., Miller, M., Del Giudice, G., and Rappuoli, R. (2011). The epidemiology of influenza and its control. *Birkhauser Advances in Infectious Diseases*.

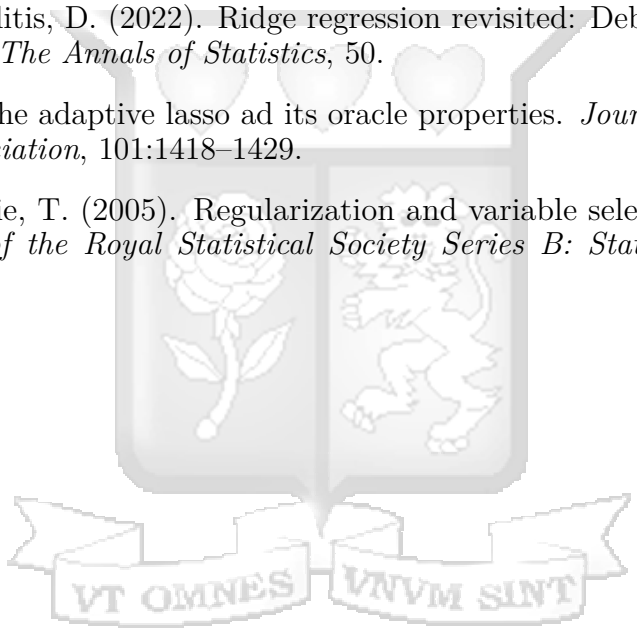
Stock, J. and Watson, M. (2011). *Introduction to Econometrics (3rd edition)*. Addison Wesley Longman. Professor Stock receives royalties for this text.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Zhang, Y. and Politis, D. (2022). Ridge regression revisited: Debiasing, thresholding and bootstrap. *The Annals of Statistics*, 50.

Zou, H. (2006). The adaptive lasso ad its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320.



Appendix A

Additional results

A.1 Optimization Procedure for Regularized VAR Models

A.1 Algorithmic Steps for Estimation

All regularized VAR models with covariates (Lasso-VAR, Ridge-VAR, and ElasticNet-VAR) are estimated using a block coordinate descent algorithm. The following steps describe the general procedure.

Algorithm A.1: Coordinate Descent for Regularized VAR with Covariates

Require: Response matrix $Y \in \mathbb{R}^{k \times T}$, lag matrix $Z \in \mathbb{R}^{kp \times T}$, covariate matrix $X \in \mathbb{R}^{q \times T}$, regularization parameters λ_B, λ_C, and mixing parameter α (for Elastic Net).

1: **Center the data:**

$$\tilde{Y} = Y - \bar{Y}, \quad \tilde{Z} = Z - \bar{Z}, \quad \tilde{X} = X - \bar{X}$$

2: **Initialize:** $B^{(0)} = 0$, $C^{(0)} = 0$

3: **repeat**

4: Compute residual: $R_B = \tilde{Y} - C^{(t)}\tilde{X}$

5: Update $B^{(t+1)}$ using penalized regression of R_B on $\tilde{Z}$

6: Compute residual: $R_C = \tilde{Y} - B^{(t+1)}\tilde{Z}$

7: Update $C^{(t+1)}$ using penalized regression of R_C on $\tilde{X}$

8: **until** Convergence

9: **Recover intercept:** $\hat{\Phi} = \bar{Y} - \hat{B}\bar{Z} - \hat{C}\bar{X}$

Notes on Implementation. Lasso, Ridge, and Elastic Net differ only in the type of penalty applied: Lasso uses an L_1 penalty, Ridge uses an L_2 penalty, and Elastic Net combines both through a mixing parameter α. The parameter updates are computed using coordinate descent, typically implemented via the `glmnet` package. For Elastic Net, cross validation, often through the `caret` package in R, is used to tune the hyperparameters λ_B, λ_C and α.

A.2 Initial Dataset

	Store <int>	Date <chr>	Weekly_Sales <dbl>	Holiday_Flag <int>	Temperature <dbl>	Fuel_Price <dbl>	CPI <dbl>	Unemployment <dbl>
1	1	05-02-2010	1643691	0	42.31	2.572	211.0964	8.106
2	1	12-02-2010	1641957	1	38.51	2.548	211.2422	8.106
3	1	19-02-2010	1611968	0	39.93	2.514	211.2891	8.106
4	1	26-02-2010	1409728	0	46.63	2.561	211.3196	8.106
5	1	05-03-2010	1554807	0	46.50	2.625	211.3501	8.106
6	1	12-03-2010	1439542	0	57.79	2.667	211.3806	8.106

Figure A.1: Snipppet for Initial Dataset

A.3 Descriptive Statistics for all stores

Table A.1: Descriptive Statistics of Store Sales

Variable	Min	1st Qu.	Median	Mean	3rd Qu.	Max
Store 1	1316899	1458105	1534850	1555264	1614892	2387950
Store 2	1650394	1803501	1879107	1925751	1956927	3436008
Store 3	339597	367861	395107	402704	420759	605990
Store 4	1762539	1929611	2073951	2094713	2175039	3676389
Store 5	260637	294696	310338	318012	329861	507900
Store 6	1261253	1456147	1524390	1564728	1624713	2727575
Store 7	372674	497653	557166	570617	621263	1059715
Store 8	772539	855905	893400	908750	929021	1511641
Store 9	452905	506767	536538	543981	560149	905325
Store 10	1627707	1740771	1827522	1899425	1944111	3749058
Store 11	1100419	1267236	1323243	1356383	1399332	2306265
Store 12	802106	940078	981616	1009002	1048359	1768250
Store 13	1633663	1877476	1958824	2003620	2041469	3595903
Store 14	1479515	1873298	2004330	2020978	2125780	3818686
Store 15	454183	565183	603319	623313	641587	1368318
Store 16	368600	466043	508520	519248	567151	1004731
Store 17	635863	817519	872818	893581	939734	1309227
Store 18	540923	1000579	1060433	1084718	1134006	2027507
Store 19	1181205	1351960	1408969	1444999	1487577	2678206
Store 20	1761017	1950866	2053165	2107677	2155186	3766687
Store 21	596218	688849	737014	756069	772894	1587258
Store 22	774262	961476	996629	1028501	1052934	1962445
Store 23	1016756	1261919	1358444	1389865	1447165	2734277
Store 24	1057290	1254317	1339630	1356755	1408801	2386016
Store 25	558795	655485	685677	706722	719413	1295391
Store 26	809833	938053	996724	1002912	1055369	1573983
Store 27	1263535	1629423	1731935	1775216	1872249	3078162
Store 28	1079669	1189279	1266460	1323522	1451185	2026026
Store 29	395987	494461	518628	539451	552041	1130927
Store 30	369722	426071	438069	438580	451765	519355
Store 31	1198072	1336420	1378340	1395901	1425429	2068943
Store 32	955464	1098315	1144902	1166568	1186363	1959527
Store 33	209986	242492	258427	259862	275236	331174
Store 34	836718	920239	950154	966782	979580	1620748
Store 35	576332	801523	849779	919725	1003066	1781867
Store 36	270678	320540	373268	373512	426490	489372
Store 37	451328	507030	518124	518900	530123	605792
Store 38	303909	350366	380870	385732	414198	499268
Store 39	1158698	1316617	1416006	1450668	1520606	2554483
Store 40	764015	896638	954234	964128	1001263	1648829
Store 41	991942	1166352	1243815	1268125	1332054	2263723
Store 42	428954	515976	556046	556404	593055	674919
Store 43	505406	605518	634815	633325	659815	725043
Store 44	241937	283882	298080	302749	319938	376234
Store 45	617208	722792	764014	785981	801676	1682862

A.4 Stationarity Results

Table A.2: Augmented Dickey-Fuller (ADF) Test Results for Endogenous and Exogenous Variables

Variable	Before Differencing	After First Differencing	After Second Differencing
Endogenous Variables (Store Sales)			
Store 1	0.01	0.01	0.01
Store 2	0.01	0.01	0.01
Store 3	0.01	0.01	0.01
Store 4	0.01	0.01	0.01
Store 5	0.01	0.01	0.01
Store 6	0.01	0.01	0.01
Store 7	0.01	0.01	0.01
Store 8	0.01	0.01	0.01
Store 9	0.01	0.01	0.01
Store 10	0.01	0.01	0.01
Store 11	0.01	0.01	0.01
Store 12	0.01	0.01	0.01
Store 13	0.01	0.01	0.01
Store 14	0.01	0.01	0.01
Store 15	0.01	0.01	0.01
Store 16	0.01	0.01	0.01
Store 17	0.01	0.01	0.01
Store 18	0.01	0.01	0.01
Store 19	0.01	0.01	0.01
Store 20	0.01	0.01	0.01
Store 21	0.01	0.01	0.01
Store 22	0.01	0.01	0.01
Store 23	0.01	0.01	0.01
Store 24	0.01	0.01	0.01
Store 25	0.01	0.01	0.01
Store 26	0.01	0.01	0.01
Store 27	0.01	0.01	0.01
Store 28	0.01	0.01	0.01
Store 29	0.01	0.01	0.01
Store 30	0.13	0.01	0.01
Store 31	0.01	0.01	0.01
Store 32	0.01	0.01	0.01
Store 33	0.40	0.01	0.01
Store 34	0.01	0.01	0.01
Store 35	0.01	0.01	0.01
Store 36	0.14	0.01	0.01
Store 37	0.04	0.01	0.01
Store 38	0.57	0.01	0.01
Store 39	0.01	0.01	0.01
Store 40	0.01	0.01	0.01
Store 41	0.01	0.01	0.01
Store 42	0.82	0.01	0.01
Store 43	0.43	0.01	0.01
Store 44	0.13	0.01	0.01
Store 45	0.01	0.01	0.01
Exogenous Variables			
Temperature	0.02	0.40	0.01
Fuel Price	0.34	0.07	0.01
CPI	0.08	0.11	0.01
Unemployment	0.78	0.01	0.01
Holiday Flag	0.01	0.01	0.01

Table A.3: Check for Overdifferencing for Endogenous and Exogenous Variables

Variable	Ljung-Box p-value	Variance ratio (differenced / original)	Interpretation
Endogenous Variables (Store Sales)			
Store 1	0.0000	3.8882	No signs of overdifferencing
Store 2	0.0000	3.4449	No signs of overdifferencing
Store 3	0.0000	2.5685	No signs of overdifferencing
Store 4	0.0000	3.2561	No signs of overdifferencing
Store 5	0.0000	3.4831	No signs of overdifferencing
Store 6	0.0000	2.9782	No signs of overdifferencing
Store 7	0.0000	1.7305	No signs of overdifferencing
Store 8	0.0000	3.5554	No signs of overdifferencing
Store 9	0.0000	3.1223	No signs of overdifferencing
Store 10	0.0000	2.8492	No signs of overdifferencing
Store 11	0.0000	3.2192	No signs of overdifferencing
Store 12	0.0000	3.3380	No signs of overdifferencing
Store 13	0.0000	3.3839	No signs of overdifferencing
Store 14	0.0000	3.2249	No signs of overdifferencing
Store 15	0.0000	3.0804	No signs of overdifferencing
Store 16	0.0000	2.0185	No signs of overdifferencing
Store 17	0.0000	4.9870	No signs of overdifferencing
Store 18	0.0000	3.2269	No signs of overdifferencing
Store 19	0.0000	3.4863	No signs of overdifferencing
Store 20	0.0000	3.4130	No signs of overdifferencing
Store 21	0.0000	3.0154	No signs of overdifferencing
Store 22	0.0000	3.2410	No signs of overdifferencing
Store 23	0.0000	2.5147	No signs of overdifferencing
Store 24	0.0000	3.8844	No signs of overdifferencing
Store 25	0.0000	2.3551	No signs of overdifferencing
Store 26	0.0000	3.8425	No signs of overdifferencing
Store 27	0.0000	3.6899	No signs of overdifferencing
Store 28	0.0000	5.1088	No signs of overdifferencing
Store 29	0.0000	3.4860	No signs of overdifferencing
Store 30	0.0000	3.8399	No signs of overdifferencing
Store 31	0.0000	4.2589	No signs of overdifferencing
Store 32	0.0000	3.6547	No signs of overdifferencing
Store 33	0.0000	2.4581	No signs of overdifferencing
Store 34	0.0000	4.0729	No signs of overdifferencing
Store 35	0.0000	2.7582	No signs of overdifferencing
Store 36	0.0000	0.3696	No signs of overdifferencing
Store 37	0.0000	6.1071	No signs of overdifferencing
Store 38	0.0000	2.0791	No signs of overdifferencing
Store 39	0.0000	2.8561	No signs of overdifferencing
Store 40	0.0000	4.2549	No signs of overdifferencing
Store 41	0.0000	3.3729	No signs of overdifferencing
Store 42	0.0000	3.5253	No signs of overdifferencing
Store 43	0.0000	3.5655	No signs of overdifferencing
Store 44	0.0000	2.6900	No signs of overdifferencing
Store 45	0.0000	3.3908	No signs of overdifferencing
Exogenous Variables			
Temperature	0.0000	0.1030	Possible overdifferencing (low variance)
Fuel Price	0.0004	0.0088	Possible overdifferencing (low variance)
CPI	0.0053	0.0001	Possible overdifferencing (low variance)
Unemployment	0.0001	0.0228	Possible overdifferencing (low variance)
Holiday Flag	0.0000	6.4335	No signs of overdifferencing

A.5 VIF analysis

Table A.4: Variance Inflation Factor (VIF) Analysis

Variable Name	VIF
Store 2	60.75
Store 3	11.67
Store 4	67.17
Store 5	28.67
Store 6	56.66
Store 7	22.77
Store 8	53.86
Store 9	39.87
Store 10	61.47
Store 11	40.29
Store 12	40.66
Store 13	71.47
Store 14	52.85
Store 15	84.70
Store 16	10.54
Store 17	6.59
Store 18	41.21
Store 19	53.94
Store 20	42.27
Store 21	56.08
Store 22	54.75
Store 23	37.77
Store 24	53.48
Store 25	42.43
Store 26	21.88
Store 27	22.43
Store 28	28.98
Store 29	31.55
Store 30	9.83
Store 31	71.04
Store 32	79.11
Store 33	11.78
Store 34	53.30
Store 35	37.51
Store 36	6.64
Store 37	7.74
Store 38	9.23
Store 39	81.14
Store 40	67.33
Store 41	99.24
Store 42	16.52
Store 43	8.31
Store 44	7.34
Store 45	90.50
Temperature	2.30
Fuel Price	1.52
CPI	1.31
Unemployment	3.06
Holiday Flag	6.37

A.6 Model Building

	Store_44_lag9	Store_1_lag10	Store_17_lag10	Store_30_lag10	Store_33_lag10	Store_36_lag10	Store_37_lag10	Store_38_lag10
[1,]	5122.77	1611968	800714.0	-33861.31	1968.00	717.33	510382.5	-14976.98
[2,]	11538.20	1409728	749549.6	8817.45	-12798.06	-22761.59	513615.8	6984.81
[3,]	-12426.44	1554807	783300.1	260.36	7432.12	32683.99	519255.7	38017.16
[4,]	-2566.63	1439542	763961.8	-4401.14	20676.11	-38769.23	513015.3	-30215.97
[5,]	6655.29	1472516	752034.5	-22453.57	-29925.27	-12582.21	460020.7	-8998.45
[6,]	9918.01	1404430	793097.6	-3279.01	-19341.97	-24413.48	515778.0	2832.64

	Store_43_lag10	Store_44_lag10	Fuel_Price	CPI	Unemployment	Holiday_Flag
[1,]	-23921.44	-18900.83	1.706667e-02	4.444445e-09	0	0
[2,]	-40294.76	5122.77	3.053333e-02	6.666653e-09	0	0
[3,]	39897.26	11538.20	-2.520000e-02	2.253177e-02	0	0
[4,]	-13213.11	-12426.44	-3.802222e-02	1.351907e-01	0	0
[5,]	22711.55	-2566.63	-3.480000e-02	-2.222208e-09	0	0
[6,]	-45000.56	6655.29	6.666667e-05	1.555554e-08	0	0

Figure A.2: Snipppet for final data after creating lags

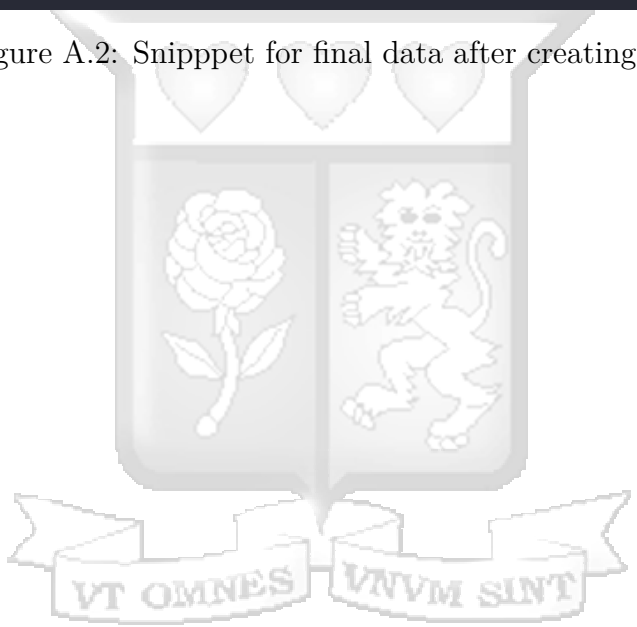


Table A.5: Optimal lambda and CV MSE for each store using Ridge on the full dataset

Endogenous Variable	Optimal Lambda	CV MSE
Store 1	15.54169	46529146160
Store 2	16.97543	96186141918
Store 3	15.52704	3565776876
Store 4	16.19885	99925519462
Store 5	14.66305	2863344546
Store 6	16.18729	70950410847
Store 7	16.13330	14421842188
Store 8	15.22785	17944890167
Store 9	14.47476	7499776781
Store 10	16.31765	123750068614
Store 11	15.84460	41586222426
Store 12	16.10632	39304116495
Store 13	16.23245	98010498832
Store 14	16.22094	131816453075
Store 15	15.27531	22594242745
Store 16	15.40926	8585544287
Store 17	15.63356	31143671300
Store 18	15.83362	47815006891
Store 19	16.01524	63419769784
Store 20	17.67937	142039951133
Store 21	15.85459	26483370250
Store 22	15.76381	38291560465
Store 23	15.81620	75620900444
Store 24	15.92835	54649392662
Store 25	14.93369	14163723751
Store 26	15.71901	22038910504
Store 27	15.78163	96627021132
Store 28	15.67538	79695896376
Store 29	16.07332	19221209870
Store 30	13.48697	707347620
Store 31	15.71322	30500733789
Store 32	15.35482	30866043239
Store 33	12.78216	487480476
Store 34	15.13883	21318458961
Store 35	15.89877	65485447871
Store 36	12.60193	695815260
Store 37	13.83752	1180745320
Store 38	13.27371	1430254205
Store 39	16.63441	69933553726
Store 40	15.39220	26051530817
Store 41	16.55308	56799230887
Store 42	13.61775	2396198543
Store 43	13.44332	2644061920
Store 44	13.64691	779060589
Store 45	16.99689	33108813876

Table A.6: Optimal lambda and CV MSE for each store using Lasso on the full dataset

Endogenous Variable	Optimal Lambda	CV MSE
Store 1	8.122244	17422262277
Store 2	8.439581	29750051510
Store 3	6.991193	1624248797
Store 4	8.546821	44267471177
Store 5	6.638890	801239305
Store 6	8.767849	27699944540
Store 7	7.504417	6044059385
Store 8	7.854930	10192484741
Store 9	7.101832	2560795294
Store 10	8.712143	41638024668
Store 11	7.913476	16100457244
Store 12	7.803059	12766260168
Store 13	8.999073	42391445290
Store 14	9.266663	74217141380
Store 15	8.041933	11824918658
Store 16	7.152516	3090288027
Store 17	7.841988	6695504050
Store 18	8.786317	27464498441
Store 19	8.409728	36793563744
Store 20	9.422622	76701432673
Store 21	7.969978	9873151728
Store 22	8.065265	16434238445
Store 23	8.396755	37036501028
Store 24	8.136777	19047913610
Store 25	7.560763	7485032875
Store 26	7.648329	7144735543
Store 27	8.548252	39879616187
Store 28	8.488519	25972903503
Store 29	8.142194	10560177705
Store 30	6.393144	304617475
Store 31	8.293784	17512091215
Store 32	8.214480	12991235189
Store 33	6.572159	184041703
Store 34	7.951977	8650368728
Store 35	8.107191	23843642858
Store 36	6.950129	266442310
Store 37	7.627516	553830787
Store 38	6.226405	371512793
Store 39	8.331148	24206159536
Store 40	7.972759	11780902815
Store 41	8.296338	18189711691
Store 42	6.942576	568778249
Store 43	6.721635	839885935
Store 44	5.994887	237808242
Store 45	7.902845	11177695031

Table A.7: Elastic Net Cross-Validation Results (Full Dataset, 45 Stores)

Store	Optimal Alpha	Optimal Lambda (log)	CV MSE
Store 1	1.0	8.0757	15996369264
Store 2	0.8	8.6162	29321814117
Store 3	0.8	6.8887	1228600631
Store 4	0.5	9.1935	36089865827
Store 5	0.4	7.4621	811411502
Store 6	1.0	8.2096	14660481657
Store 7	0.8	7.3554	3852554170
Store 8	0.5	8.2690	7619724740
Store 9	0.3	8.2128	2365288847
Store 10	0.9	8.6780	35705012090
Store 11	0.7	8.3632	15981564921
Store 12	1.0	7.9891	13696756100
Store 13	1.0	8.5339	35134031937
Store 14	0.5	9.5412	52033280614
Store 15	0.9	7.8682	9062277219
Store 16	0.9	7.1648	2592898879
Store 17	0.9	8.0404	8161262610
Store 18	0.7	8.5383	18543698675
Store 19	1.0	8.2237	29390307058
Store 20	1.0	8.5388	35261801427
Store 21	0.7	8.0476	8485862620
Store 22	0.9	8.5893	16003104265
Store 23	0.5	9.1829	37233572289
Store 24	0.6	8.5546	20103559108
Store 25	0.8	7.6444	5218335471
Store 26	0.4	8.7042	6161508343
Store 27	0.8	8.7249	35419479696
Store 28	1.0	8.2094	21391839055
Store 29	0.5	8.2771	7645851652
Store 30	0.9	6.3590	273984109
Store 31	0.9	7.9805	11423601396
Store 32	0.8	8.2050	9370887833
Store 33	0.8	6.1441	145579086
Store 34	1.0	7.6264	7431646984
Store 35	1.0	8.3398	23061784730
Store 36	0.4	6.5174	263292783
Store 37	0.4	8.1252	497340569
Store 38	0.7	6.7226	425204802
Store 39	0.5	8.9778	20736329329
Store 40	1.0	7.7867	8078806198
Store 41	0.8	8.3799	17941816493
Store 42	1.0	6.8961	568378861
Store 43	0.8	6.7122	742053593
Store 44	0.7	6.3981	259731417
Store 45	0.6	8.7393	10824118631

A.7 Peformance Metrics for Data without Multicollinearity

Table A.8: Performance Metrics Unregularized VAR

Store	RMSE	MAE	MAPE
Store 1	429127.81	325159.40	792.31
Store 17	348388.19	237771.36	575.69
Store 30	65647.87	46270.21	581.84
Store 36	53232.06	43232.63	977.20
Store 37	83440.85	60331.78	1710.53
Store 38	100587.88	78493.77	517.61
Store 43	117766.00	95825.26	421.70
Store 44	53747.34	36199.92	460.83

Table A.9: Performance Metrics Ridge VAR

Store	RMSE	MAE	MAPE
Store 1	175329.98	107859.56	149.23
Store 17	106119.47	74494.25	304.71
Store 30	15802.71	12129.29	255.99
Store 36	12204.80	9760.78	269.25
Store 37	19612.99	14788.25	287.64
Store 38	24159.29	17913.72	82.88
Store 43	27006.96	21536.40	113.99
Store 44	18119.15	13056.86	237.63

Table A.10: Performance Metrics Lasso VAR

Store	RMSE	MAE	MAPE
Store 1	129942.67	83241.73	128.41
Store 17	93738.31	69888.12	297.53
Store 30	15788.12	12119.21	215.48
Store 36	11640.77	9333.10	276.31
Store 37	13810.13	10849.13	322.41
Store 38	15735.07	11989.16	109.74
Store 43	17365.46	13684.06	83.51
Store 44	17238.96	12564.42	269.39

Table A.11: Performance Metrics Elastic Net VAR

Store	RMSE	MAE	MAPE
Store 1	117196.32	75290.41	129.29
Store 17	79459.33	60982.81	305.92
Store 30	11861.11	9247.47	143.58
Store 36	8639.71	6835.80	83.04
Store 37	13902.82	10880.33	348.04
Store 38	13427.93	10375.02	110.82
Store 43	17716.05	13949.07	84.05
Store 44	15636.85	11615.29	258.50

A.8 Peformance Metrics for Full Dataset

Table A.12: Performance Metrics Ridge VAR (Full Dataset)

Store	RMSE	MAE	MAPE
Store 1	206021.88	141995.67	153.91
Store 2	339346.49	190625.48	236.17
Store 3	65496.94	45090.45	142.64
Store 4	322001.44	184324.45	128.74
Store 5	57683.14	35311.26	136.02
Store 6	278094.51	171321.96	206.35
Store 7	131600.85	72379.02	220.91
Store 8	135555.45	82623.82	3647.10
Store 9	84770.47	50237.91	4333.02
Store 10	367135.50	190629.05	263.78
Store 11	210958.98	135636.34	172.76
Store 12	213407.93	110613.64	1130.91
Store 13	334442.91	195828.35	218.28
Store 14	367259.96	220824.61	203.17
Store 15	152035.87	80965.38	221.68
Store 16	98432.90	57264.86	152.27
Store 17	181899.88	115886.37	209.88
Store 18	228481.78	132906.75	176.97
Store 19	261760.00	158906.93	326.02
Store 20	420311.87	254250.56	317.10
Store 21	177755.18	88129.95	176.82
Store 22	204089.06	120519.58	323.38
Store 23	269711.49	174579.96	193.49
Store 24	244299.98	164155.93	506.07
Store 25	121652.88	68086.46	324.72
Store 26	155065.59	105075.69	1115.50
Store 27	307452.68	193163.15	1627.20
Store 28	275431.54	198335.22	121.53
Store 29	153446.68	83446.71	138.37
Store 30	25230.56	18389.79	220.41
Store 31	182331.33	106210.11	154.14
Store 32	174835.15	101591.26	289.04
Store 33	21219.97	17348.16	62.41
Store 34	145632.01	88360.39	497.33
Store 35	269422.40	140088.56	389.29
Store 36	23464.71	19212.56	340.06
Store 37	33267.59	24660.58	280.94
Store 38	35536.93	27283.18	109.63
Store 39	286767.62	156082.82	199.33
Store 40	161036.14	109226.48	126.37
Store 41	264941.81	143339.75	205.36
Store 42	45632.41	38961.89	83.96
Store 43	46032.72	36984.22	113.13
Store 44	28157.38	20536.96	176.96
Store 45	205224.60	100721.81	143.44

Table A.13: Performance Metrics Lasso VAR (Full Dataset)

Store	RMSE	MAE	MAPE
Store 1	64917.01	47587.51	94.15
Store 2	88995.26	63657.51	108.59
Store 3	22002.94	16168.21	633.53
Store 4	101580.81	75070.79	134.60
Store 5	15898.22	11629.55	196.63
Store 6	96975.75	69784.15	136.72
Store 7	38299.17	28635.04	245.66
Store 8	49286.63	37033.38	5045.05
Store 9	24949.29	18365.99	3905.69
Store 10	113965.52	81060.07	201.76
Store 11	62133.59	45812.09	88.44
Store 12	56657.63	40356.64	613.44
Store 13	127053.49	89331.94	206.52
Store 14	155869.59	103690.99	162.42
Store 15	56511.55	40834.42	197.61
Store 16	27206.08	20414.17	104.08
Store 17	49092.99	36775.83	163.12
Store 18	101353.62	68943.89	246.97
Store 19	91590.09	67538.39	407.11
Store 20	162991.55	112635.20	276.67
Store 21	56698.94	41423.57	172.31
Store 22	67293.63	47994.22	154.95
Store 23	99632.32	75425.52	129.40
Store 24	74956.50	57980.10	452.26
Store 25	40610.83	30302.66	558.82
Store 26	41424.25	30514.88	355.85
Store 27	104822.68	74930.82	763.03
Store 28	88259.22	64595.28	88.56
Store 29	60178.91	43489.91	228.34
Store 30	10279.02	8081.09	122.29
Store 31	68659.61	49716.17	126.34
Store 32	63923.06	47713.23	171.93
Store 33	8743.72	6975.84	38.87
Store 34	48922.25	37499.00	183.48
Store 35	72465.93	51097.76	262.40
Store 36	12682.89	9907.40	216.13
Store 37	18810.05	14492.02	278.60
Store 38	11101.85	8439.67	58.81
Store 39	81280.33	59615.12	181.59
Store 40	49006.00	35066.17	57.07
Store 41	75135.51	56196.53	139.75
Store 42	15095.80	11846.71	49.98
Store 43	16135.67	12636.77	66.75
Store 44	8560.51	6995.09	116.80
Store 45	53299.62	35622.34	116.74

Table A.14: Performance Metrics Elastic Net VAR (Full Dataset)

Store	RMSE	MAE	MAPE
Store 1	62928.26	46093.47	90.76
Store 2	87849.46	62528.19	108.66
Store 3	19049.24	14051.91	529.88
Store 4	102088.00	74446.40	123.77
Store 5	16201.94	11534.06	205.02
Store 6	73207.59	54141.09	107.03
Store 7	31624.15	24036.53	197.94
Store 8	44436.91	33893.89	5748.08
Store 9	26715.81	19188.19	3636.68
Store 10	106322.33	76526.59	192.09
Store 11	67159.85	48968.67	93.76
Store 12	62639.96	44087.20	673.68
Store 13	102628.18	75756.73	172.92
Store 14	132353.27	86943.40	130.25
Store 15	49908.27	36716.73	170.42
Store 16	26032.06	19530.62	100.73
Store 17	52038.29	38865.35	167.90
Store 18	76682.15	54514.09	207.54
Store 19	84327.17	63211.24	347.65
Store 20	103455.48	75072.11	178.93
Store 21	50275.58	37453.27	167.82
Store 22	83232.71	58200.83	180.74
Store 23	107404.57	80213.22	152.18
Store 24	73257.42	56660.94	416.04
Store 25	38797.55	29257.53	539.17
Store 26	47536.41	34141.63	967.89
Store 27	103584.36	74215.68	713.95
Store 28	74801.14	55158.23	75.16
Store 29	48051.70	35014.29	198.99
Store 30	9743.80	7706.45	122.95
Store 31	55374.26	40559.52	91.20
Store 32	57445.23	43522.43	160.54
Store 33	6702.12	5354.66	30.37
Store 34	40142.23	30797.64	148.24
Store 35	82306.27	57350.36	314.86
Store 36	7848.90	6145.43	60.24
Store 37	17293.39	13344.26	208.84
Store 38	12139.64	9230.80	61.95
Store 39	83993.50	61299.95	191.22
Store 40	44153.65	31748.79	52.32
Store 41	70960.30	53462.65	130.48
Store 42	14780.27	11574.99	48.88
Store 43	14265.75	11167.02	59.48
Store 44	8832.90	7175.31	113.02
Store 45	64982.34	42378.63	134.31

A.9 Forecasting Metrics for Data without Multicollinearity

Table A.15: Forecast Performance Metrics Unregularized VAR

Store	RMSE	MAE	MAPE
Store 1	793125.11	698817.87	1741.90
Store 17	725679.65	597933.22	1084.05
Store 30	118314.44	102023.66	1892.97
Store 36	71892.57	62747.55	836.71
Store 37	187497.24	161948.97	44116.14
Store 38	144542.86	125755.23	665.90
Store 43	155985.94	144384.92	1934.32
Store 44	97432.23	93190.48	873.67

Table A.16: Forecasting Performance Metrics Ridge VAR

Store	RMSE	MAE	MAPE
Store 1	157563.95	110290.05	226.55
Store 17	190105.06	129296.36	132.15
Store 30	25461.11	21040.60	458.44
Store 36	24233.28	19990.79	318.66
Store 37	14090.03	11366.50	5726.23
Store 38	45553.30	32187.88	122.13
Store 43	34018.81	29836.27	184.20
Store 44	25108.82	20145.36	169.48

Table A.17: Forecasting Metrics Lasso VAR

Store	RMSE	MAE	MAPE
Store 1	255625.21	166918.97	268.56
Store 17	175725.90	124092.20	118.03
Store 30	20582.86	17061.81	461.14
Store 36	23743.13	18669.69	271.98
Store 37	15440.63	12663.88	6687.52
Store 38	41039.91	30428.12	130.09
Store 43	35370.57	29834.69	263.99
Store 44	17653.92	10168.82	56.03

Table A.18: Forecasting Performance Metrics Elastic Net VAR

Store	RMSE	MAE	MAPE
Store 1	283767.32	197786.98	231.57
Store 17	181624.86	161162.42	190.81
Store 30	22846.54	18288.42	413.43
Store 36	29326.27	23774.12	329.66
Store 37	16698.09	13477.80	8647.90
Store 38	35908.27	26158.97	152.77
Store 43	42483.13	39710.16	347.90
Store 44	33725.43	27205.37	195.49

A.10 Forecasting Metrics for Full Dataset

Table A.19: Ridge VAR Forecasting Performance Metrics for All Stores

Store	RMSE	MAE	MAPE
Store 1	135401.72	111796.15	87.20
Store 2	134499.16	103513.79	84.15
Store 3	35240.95	27565.55	1754.06
Store 4	129186.28	114784.18	221.79
Store 5	32621.07	24984.94	100.84
Store 6	117470.59	95373.33	115.69
Store 7	28713.26	25072.00	338.90
Store 8	55140.00	47049.46	39732.70
Store 9	55020.17	41994.01	79.38
Store 10	63831.37	47305.28	188.94
Store 11	120814.08	89863.03	100.91
Store 12	79222.86	66767.69	87.73
Store 13	121268.89	94129.40	311.67
Store 14	216115.35	164882.53	1140.25
Store 15	25659.68	18707.02	154.75
Store 16	65579.42	45065.09	157.49
Store 17	238323.26	159801.42	110.30
Store 18	64306.65	54659.31	250.17
Store 19	83730.24	65591.73	104.94
Store 20	128586.50	104379.73	743.93
Store 21	58389.22	45530.89	208.07
Store 22	72681.85	58821.17	129.66
Store 23	125552.00	97064.57	780.21
Store 24	155355.75	131869.02	146.72
Store 25	31145.76	24135.98	183.56
Store 26	63115.20	56822.14	270.80
Store 27	154635.84	119458.12	134.84
Store 28	234953.68	187998.42	119.96
Store 29	122726.10	75919.28	106.73
Store 30	16062.60	14299.44	203.05
Store 31	66351.07	48939.40	100.50
Store 32	53253.31	43991.21	244.39
Store 33	22959.71	18704.25	70.38
Store 34	51172.64	39888.02	312.92
Store 35	86428.06	63228.17	137.40
Store 36	18813.46	15248.43	114.02
Store 37	18866.28	15257.58	2036.12
Store 38	42873.38	31621.32	90.51
Store 39	84309.67	73102.98	62.24
Store 40	122836.37	98955.19	114.81
Store 41	106173.62	87024.92	107.98
Store 42	53507.64	46891.64	76.93
Store 43	44549.67	35393.58	171.34
Store 44	29141.44	24932.78	122.07
Store 45	46283.11	37280.52	130.00

Table A.20: Lasso VAR Forecasting Performance Metrics for All Stores

Store	RMSE	MAE	MAPE
Store 1	140561.73	107945.57	178.41
Store 2	121284.33	96132.54	132.02
Store 3	25929.72	22486.18	14311.01
Store 4	129522.80	116199.93	208.89
Store 5	35237.27	28159.44	875.75
Store 6	130976.59	109517.63	386.68
Store 7	82822.85	66179.55	1310.81
Store 8	56747.51	47047.14	140435.43
Store 9	41732.58	34853.85	94.25
Store 10	150263.76	125663.90	466.91
Store 11	135919.37	115843.66	278.89
Store 12	111769.25	93472.64	214.67
Store 13	155262.01	105179.59	198.06
Store 14	101834.43	83461.83	1178.96
Store 15	63255.12	56653.05	971.19
Store 16	63185.50	54071.50	337.78
Store 17	150436.35	105607.53	119.69
Store 18	99505.52	81010.60	593.59
Store 19	97972.45	89818.42	167.54
Store 20	123246.38	111162.36	2244.14
Store 21	128606.57	113254.41	1436.18
Store 22	67510.72	60809.14	171.47
Store 23	167899.12	156765.11	1566.65
Store 24	143517.03	131302.92	462.09
Store 25	75009.32	64446.52	1148.25
Store 26	78433.17	66437.22	660.52
Store 27	107692.44	86160.08	180.24
Store 28	177453.14	172305.31	341.49
Store 29	128719.99	88421.72	178.70
Store 30	17795.15	15990.33	403.51
Store 31	138045.19	109484.50	361.89
Store 32	72499.37	56412.69	361.34
Store 33	20444.96	14989.50	118.15
Store 34	67922.40	61203.00	452.01
Store 35	138485.31	111123.08	384.43
Store 36	12888.66	11171.26	176.42
Store 37	12972.37	11888.20	1226.62
Store 38	37654.77	26483.53	76.71
Store 39	122546.74	103904.66	116.99
Store 40	48744.87	38777.85	87.95
Store 41	137745.63	104837.94	330.40
Store 42	25920.69	23079.02	56.74
Store 43	27696.73	23109.90	51.57
Store 44	16563.73	12812.33	92.67
Store 45	66759.86	51313.92	279.91

Table A.21: Elastic Net VAR Forecasting Performance Metrics for All Stores

Store	RMSE	MAE	MAPE
Store 1	139764.17	100250.45	134.04
Store 2	126103.27	100734.05	138.44
Store 3	26511.20	22908.89	14530.54
Store 4	130468.28	114291.04	216.80
Store 5	34926.89	28100.17	860.45
Store 6	130806.50	108574.65	391.74
Store 7	82146.74	65759.99	698.82
Store 8	56457.44	45966.19	136481.20
Store 9	42181.54	34922.44	89.34
Store 10	153611.71	128993.18	500.64
Store 11	137541.70	117039.54	280.50
Store 12	119009.29	99454.45	224.81
Store 13	137275.74	100374.09	387.86
Store 14	110024.51	85586.85	1123.93
Store 15	62991.63	56323.95	797.69
Store 16	62533.69	53883.03	347.52
Store 17	147846.85	104216.35	119.27
Store 18	90183.52	73693.76	559.58
Store 19	111658.82	101949.79	188.54
Store 20	136936.72	105551.02	2837.60
Store 21	121335.15	105914.33	1320.83
Store 22	72005.79	65336.20	176.40
Store 23	176630.87	160713.08	1755.50
Store 24	138634.17	125751.57	443.47
Store 25	71847.05	61442.14	915.67
Store 26	80004.31	68883.85	624.36
Store 27	129961.56	104710.45	215.73
Store 28	178258.17	173255.28	340.99
Store 29	132683.84	95149.08	215.23
Store 30	18207.37	16377.55	429.31
Store 31	113865.08	96274.33	301.57
Store 32	71942.60	55506.49	353.69
Store 33	19849.09	14424.02	116.06
Store 34	66481.79	59577.10	442.12
Store 35	130699.04	106867.65	355.98
Store 36	20791.93	17611.40	292.86
Store 37	14455.55	12942.35	1308.19
Store 38	38525.18	26776.48	84.92
Store 39	118005.93	101559.19	115.49
Store 40	48115.51	35845.43	77.17
Store 41	123118.90	101196.61	294.86
Store 42	27008.12	24155.10	57.91
Store 43	29435.93	24802.01	91.02
Store 44	16505.23	12751.85	88.12
Store 45	63348.05	49416.94	262.38

VT OMNES
VNVM SINT

Appendix B

Ethical Clearance Confirmation

Strathmore UNIVERSITY

7th February 2025

Ms Waweru Esther,
esther.nyakio@strathmore.edu

Dear Ms Waweru,

RE: Regularized Vector Autoregressive Model for Time Series Data with Multiple Covariates

This is to inform you that SU-ISERC has reviewed and **approved** your above **SU-masters** proposal. Your application reference number is **SU-ISERC2611/25.** The approval period is from **7th February 2025 to 6th February 2026.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used.
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-ISERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-ISERC within 72 hours of notification.
iv. Any changes anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-ISERC within 72 hours.
v. Clearance for the export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to the expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days of completion of the study to SU-ISERC.

Before commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,

Mr Ambrose Rachier,
Chairperson; SU-ISERC

Ole Sangale Rd, Madaraka Estate. PO Box 59857-00200, Nairobi, Kenya. Tel +254 (0)703 034000
Email admissions@strathmore.edu www.strathmore.edu

VT OMNES
VNVM SINT

Appendix C

Similarity Report

Regularized Vector Autoregressive Model for time series data with multiple covariates.pdf

ORIGINALITY REPORT

6%	4%	4%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Georgia Institute of Technology Main Campus Student Paper	1%
2	Submitted to Strathmore University Student Paper	<1%
3	Submitted to Erasmus University of Rotterdam Student Paper	<1%
4	github.com Internet Source	<1%
5	Submitted to University of San Diego Student Paper	<1%
6	Submitted to RMIT University Student Paper	<1%
7	Submitted to Queen Mary and Westfield College Student Paper	<1%
8	Submitted to University of Sunderland Student Paper	<1%
9	Submitted to London Business School Student Paper	<1%

strathprints.strath.ac.uk

Appendix D

R code

```
## ----setup, include=FALSE
    ------------------------------------------------------------------------


knitr::opts_chunk$set(echo = TRUE)



##
    ------------------------------------------------------------------------


#install.packages('tseries')
#install.packages('vars')
#install.packages('corrr')
#install.packages("FactoMineR")
#install.packages("lmtest")
#install.packages("BigVAR")
#install.packages("Metrics")
#install.packages("vars")
#install.packages("gridExtra")
#install.packages("forecast")



##
    ------------------------------------------------------------------------


# Load libraries
```

```r
library(dplyr)
library(car)
library(reshape2)
library(lubridate)
library(caret)
library(tidyverse)
library(ggplot2)
library(tseries)
library(corrr)
library(FactoMineR)
library(vars)
library(lmtest)
library(BigVAR)
library(glmnet)
library(Metrics)
library(vars)
library(gridExtra)
library(extrafont)
library(patchwork)  # For arranging multiple plots
library(forecast)


##
   ------------------------------------------------------------------------


#Load dataset
setwd("C://Users//User NA//OneDrive//Documents//Master//Thesis//
   /Rcode//dataset")
data <- read.csv('Walmart.csv')
head(data)
attach(data)

```

```
49
50 ##
    ----------------------------------------------------------------------
51 nrow(data)
52
53
54 ##
    ----------------------------------------------------------------------
55 #Check for missing values
56 sum(is.na(data))
57
58
59 ##
    ----------------------------------------------------------------------
60 #Check data type
61 str(data)
62
63
64 ##
    ----------------------------------------------------------------------
65 #Format date column to date format
66 data$Date <- dmy(data$Date)
67 str(data)
68
69
70 ##
    ----------------------------------------------------------------------
```

```r
duplicates <- duplicated(data)
sum(duplicates)


##
    ------------------------------------------------------------------------


#summary statistics
summary(data)


##
    ------------------------------------------------------------------------


# Check for multiple values of exogenous variables on the same
    date
exog_check <- data %>%
  group_by(Date) %>%
  summarise(
    unique_temps = n_distinct(Temperature),
    unique_fuel = n_distinct(Fuel_Price),
    unique_cpi = n_distinct(CPI),
    unique_unemployment = n_distinct(Unemployment),
    unique_holiday = n_distinct(Holiday_Flag)
  )

# Print rows where there are multiple values (should be 1 for
    all if identical)
print(filter(exog_check, unique_temps > 1 | unique_fuel > 1 |
    unique_cpi > 1 | unique_unemployment > 1 | unique_holiday >
    1))

```

```
95
96
97  ##
        ----------------------------------------------------------------------
98  # Pivot data:
99  data_wide <- data %>%
100   dplyr::select(Store, Date, Weekly_Sales) %>%
101   pivot_wider(names_from = Store,
102               values_from = c(Weekly_Sales),
103               names_prefix = "Store_")
104 # Compute average exogenous variables per week
105 exog_avg <- data %>%
106   group_by(Date) %>%
107   summarise(
108     Temperature = mean(Temperature, na.rm = TRUE),
109     Fuel_Price = mean(Fuel_Price, na.rm = TRUE),
110     CPI = mean(CPI, na.rm = TRUE),
111     Unemployment = mean(Unemployment, na.rm = TRUE),
112     Holiday_Flag = ifelse(any(Holiday_Flag == 1), 1, 0)
113   )
114
115 # Merge averaged exogenous variables back into walmart_wide
116 final_data <- left_join(data_wide, exog_avg, by = "Date")
117
118
119 # View final structure
120 head(final_data)
121
122
```

```
123 ##
    ------------------------------------------------------------------------

124 dim(final_data)
125
126
127
128 ##
    ------------------------------------------------------------------------

129 summary(final_data)
130
131 #Store sales vary significantly across locations with Store 20
    having one of the highest average weekly sales (~2.1M) and
    Store 37 has one of the lowest (~518K).
132 #Sales distribution is skewed (mean > median in most stores),
    indicating some weeks had very high sales (e.g., holiday
    spikes).
133 #Temperature ranges from 30.48 F to 82.18 F (winter vs. summer
    variability).
134 #Median is 61.05 F, meaning most recorded weeks had mild
    temperatures.
135 #Fuel Price ($ per gallon)Ranges from $2.67 to $3.99,
    reflecting fluctuations in fuel prices over time.
136 #CPI (Consumer Price Index) varies between 167.5 and 176.7,
    capturing inflation trends.
137 #Unemployment Rate (%) varies from 6.95% to 8.62%, suggesting
    economic fluctuations over time.
138 #Holiday Flag (Binary: 1 = Holiday Week, 0 = Non-Holiday Week):
    Mean: 0.0699 hence Only ~7% of weeks in the dataset are
    holiday weeks.
139
```

```
140
141 ##
    ------------------------------------------------------------------
142 sales_data <- final_data %>% dplyr::select(Date,starts_with("
    Store_"))
143 head(sales_data)
144
145
146 ##
    ------------------------------------------------------------------
147 # Compute total sales for each store
148 store_sums <- colSums(sales_data[, -1])  # Exclude Date column
149 top_stores <- names(sort(store_sums, decreasing = TRUE)[1:5])
150 bottom_stores <- names(sort(store_sums, decreasing = FALSE)
    [1:5])
151
152
153 ##
    ------------------------------------------------------------------
154 #Filter for top stores
155 top_sales <- sales_data[, c("Date", top_stores)]
156 top_sales_long <- melt(top_sales, id.vars = "Date", variable.
    name = "Store", value.name = "Sales")
157
158 # Plot top sales
159 ggplot(top_sales_long, aes(x = Date, y = Sales, group = Store,
    color = Store)) +
160   geom_line() +
161   theme_minimal() +
```

```
  theme(text = element_text(family = "Times New Roman")) +
  labs( x = "Date", y = "Sales")


##
    ------------------------------------------------------------------


#Filter for bottom stores
bottom_sales <- sales_data[, c("Date", bottom_stores)]
bottom_sales_long <- melt(bottom_sales, id.vars = "Date",
    variable.name = "Store", value.name = "Sales")

# Plot top sales
ggplot(bottom_sales_long, aes(x = Date, y = Sales, group =
    Store, color = Store)) +
  geom_line() +
  theme_minimal() +
  theme(text = element_text(family = "Times New Roman")) +
  labs( x = "Date", y = "Sales")


##
    ------------------------------------------------------------------


# Plot Temperature over Time
p1 = ggplot(final_data, aes(x = Date, y = Temperature)) +
  geom_line(color = "blue") +
  labs(x = "Date", y = "Temperature") +
  theme_minimal() +
  theme(text = element_text(family = "Times New Roman"))

p1
```

```
188
189
190 ##
    ----------------------------------------------------------------------
191 # Plot Fuel Price over Time
192 p2 = ggplot(final_data, aes(x = Date, y = Fuel_Price)) +
193   geom_line(color = "red") +
194   labs(x = "Date", y = "Fuel Price") +
195   theme_minimal() +
196   theme(text = element_text(family = "Times New Roman"))
197
198 p2
199
200
201 ##
    ----------------------------------------------------------------------
202 # Plot CPI over Time
203 p3 = ggplot(final_data, aes(x = Date, y = CPI)) +
204   geom_line(color = "purple") +
205   labs(x = "Date", y = "CPI") +
206   theme_minimal() +
207   theme(text = element_text(family = "Times New Roman"))
208
209 p3
210
211
212 ##
    ----------------------------------------------------------------------
213 # Plot Unemployment over Time
```

```
p4 = ggplot(final_data, aes(x = Date, y = Unemployment)) +
  geom_line(color = "green") +
  labs(x = "Date", y = "Unemployment") +
  theme_minimal() +
  theme(text = element_text(family = "Times New Roman"))

p4


##
    ------------------------------------------------------------------------


# Plot Holiday Flag over Time
p5 = ggplot(final_data, aes(x = Date, y = Holiday_Flag)) +
  geom_point(data = subset(final_data, Holiday_Flag == 1),
             aes(x = Date, y = Holiday_Flag),
             color = "red", size = 3, shape = 19) +
  scale_y_continuous(breaks = c(0, 1), labels = c("No Holiday",
      "Holiday")) +
  labs(x = "Date", y = "Holiday Indicator") +
  theme_minimal() +
  theme(text = element_text(family = "Times New Roman")) +
  ggtitle("Holiday Occurrences (Red dots indicate holidays)")

p5



##
    ------------------------------------------------------------------------


# Combine them into a grid
```

```
241 (p1 | p2) / (p3 | p4) / p5
242
243 #There is a clear seasonal pattern for temperature as it peaks
    around mid year and drops during winter months.
244 #There is a general upward trend for fuel prices with some
    fluctuations suggesting suggests external market conditions
    influence prices and there may be periods of volatility.
245 #The CPI shows a steady increasing trend over time which
    indicates inflationary pressures in the economy. #The
    unemployment rate follows a stepwise downward trend
    suggesting a gradual improvement in employment conditions
    over time.
246 #The holiday variable is binary (0 or 1) showing spikes at
    specific points in time. These spikes indicate the
    occurrence of major holidays which could affect sales
    patterns.
247
248
249
250 ##
    ------------------------------------------------------------------------
251 summary(sales_data %>% dplyr::select(-Date))
252
253
254 ##
    ------------------------------------------------------------------------
255 # Define Endogenous Variables (Store Sales)
256 endog_vars <- final_data %>% dplyr::select(starts_with("Store_"
    ))
257
```

```r
# Define Exogenous Variables
exog_vars <- final_data %>% dplyr::select(Temperature,Fuel_Price, CPI, Unemployment, Holiday_Flag)

# Function to check stationarity using ADF test
check_stationarity <- function(series) {
  adf_test <- adf.test(series, alternative = "stationary")
  return(adf_test$p.value)  # If p-value < 0.05, the series is stationary
}

# Apply ADF test to all endogenous variables
stationarity_results_endog <- sapply(endog_vars, check_stationarity)
print("Stationarity of Endogenous Variables:")
print(stationarity_results_endog)

# Apply ADF test to all exogenous variables
stationarity_results_exog <- sapply(exog_vars, check_stationarity)
print("Stationarity of Exogenous Variables:")
print(stationarity_results_exog)


##----------------------------------------------------------------------


# Store column names before differencing
endog_names <- names(final_data %>% dplyr::select(starts_with("Store_")))
exog_names <- names(final_data %>% dplyr::select(Temperature, Fuel_Price, CPI, Unemployment, Holiday_Flag))
```

```
282
283 # Apply first differencing to endogenous variables
284 diff_endog_vars_all <- diff(as.matrix(final_data %>% dplyr::
    select(starts_with("Store_"))) , differences = 1)
285
286 # Apply first differencing to exogenous variables
287 diff_exog_vars_all <- diff(as.matrix(final_data %>% dplyr::
    select(Temperature, Fuel_Price, CPI, Unemployment, Holiday_
    Flag)), differences = 1)
288
289 # Restore column names after differencing
290 colnames(diff_endog_vars_all) <- colnames(final_data %>% dplyr
    ::select(starts_with("Store_")))
291 colnames(diff_exog_vars_all) <- colnames(final_data %>% dplyr::
    select(Temperature, Fuel_Price, CPI, Unemployment, Holiday_
    Flag))
292
293 # Create a new dataset with shifted rows
294 final_data_diff2 <- final_data[-1, ]
295
296 # Assign differenced values back while ensuring the dimensions
    match
297 final_data_diff2[, colnames(diff_endog_vars_all)] <- diff_endog
    _vars_all
298 final_data_diff2[, colnames(diff_exog_vars_all)] <- diff_exog_
    vars_all
299
300
301
302 ##
    ------------------------------------------------------------------
```

```
# Apply ADF test to first differenced endogenous variables

diff_endog_vars_all <- as.data.frame(diff_endog_vars_all)
diff_exog_vars_all <- as.data.frame(diff_exog_vars_all)


stationarity_results_endog_diff2 <- sapply(diff_endog_vars_all,
    check_stationarity)
print("Stationarity of Endogenous Variables:")
print(stationarity_results_endog_diff2)

# Apply ADF test to first differenced exogenous variables
stationarity_results_exog_diff2 <- sapply(diff_exog_vars_all,
    check_stationarity)
print("Stationarity of Exogenous Variables:")
print(stationarity_results_exog_diff2)



##
    ----------------------------------------------------------------------
#Apply second differencing since some exogenous covariates
    Temperature,Fuel Price and CPI are still non-stationary

# Store column names before differencing
endog_names <- names(final_data_diff2 %>% dplyr::select(starts_
    with("Store_")))
exog_names <- names(final_data_diff2 %>% dplyr::select(
    Temperature, Fuel_Price, CPI, Unemployment, Holiday_Flag))

# Apply second differencing to endogenous variables
```

```
diff2_endog_vars_all <- diff(as.matrix(final_data_diff2%>%
    dplyr::select(starts_with("Store_"))) , differences = 1)

# Apply second differencing to exogenous variables
diff2_exog_vars_all <- diff(as.matrix(final_data_diff2 %>%
    dplyr::select(Temperature, Fuel_Price, CPI, Unemployment,
    Holiday_Flag)), differences = 1)

# Restore column names after differencing
colnames(diff2_endog_vars_all) <- colnames(final_data_diff2 %>%
     dplyr::select(starts_with("Store_")))
colnames(diff2_exog_vars_all) <- colnames(final_data_diff2 %>%
    dplyr::select(Temperature, Fuel_Price, CPI, Unemployment,
    Holiday_Flag))

# Create a new dataset with shifted rows
diff2_final_data <- final_data_diff2[-1, ]

# Assign differenced values back while ensuring the dimensions
    match
diff2_final_data[, colnames(diff2_endog_vars_all)] <- diff2_
    endog_vars_all
diff2_final_data[, colnames(diff2_exog_vars_all)] <- diff2_exog
    _vars_all


##
    ------------------------------------------------------------------------


# Apply ADF test to second differenced variables

diff2_endog_vars_all <- as.data.frame(diff2_endog_vars_all)
```

```r
diff2_exog_vars_all <- as.data.frame(diff2_exog_vars_all)


diff2_stationarity_results_endog <- sapply(diff2_endog_vars_all,check_stationarity)
print("Stationarity of Endogenous Variables:")
print(diff2_stationarity_results_endog)

# Apply ADF test to second differenced exogenous variables
diff2_stationarity_results_exog <- sapply(diff2_exog_vars_all, check_stationarity)
print("Stationarity of Exogenous Variables:")
print(diff2_stationarity_results_exog)


##------------------------------------------------------------------------


#Check if over-differenced

check_overdifferencing <- function(original_series, differenced_series, var_name) {
  # Variance check
  var_original <- var(original_series, na.rm = TRUE)
  var_diff <- var(differenced_series, na.rm = TRUE)
  variance_ratio <- var_diff / var_original

  # Ljung-Box test
  lb_test <- Box.test(differenced_series, lag = 10, type = "Ljung-Box")

  # Interpretation logic
```

```r
  interpretation <- if (variance_ratio < 0.3 & lb_test$p.value > 0.05) {
    "Possible overdifferencing (low variance & no autocorrelation)"
  } else if (variance_ratio < 0.3) {
    "Possible overdifferencing (low variance)"
  } else if (lb_test$p.value > 0.05) {
    "No autocorrelation detected -- review differencing"
  } else {
    "No signs of overdifferencing"
  }

  # Print summary
  cat(paste0("\nResults for ", var_name, ":\n"))
  cat(paste0("Ljung-Box p-value: ", round(lb_test$p.value, 4), "\n"))
  cat(paste0("Variance ratio (differenced / original): ", round(variance_ratio, 4), "\n"))
  cat(paste0("Interpretation: ", interpretation, "\n"))

  return(data.frame(
    Variable = var_name,
    LB_P_Value = round(lb_test$p.value, 4),
    Variance_Ratio = round(variance_ratio, 4),
    Interpretation = interpretation
  ))
}
```

```
401 ##
    ------------------------------------------------------------------------

402 # DataFrame to store results
403 overdiff_results <- data.frame()
404
405 # Filter out non-numeric columns (like Date)
406 numeric_vars <- colnames(diff2_final_data)[sapply(diff2_final_
    data, is.numeric)]
407
408 for (var_name in numeric_vars) {
409    res <- check_overdifferencing(
410     original_series = final_data[[var_name]],
411     differenced_series = diff2_final_data[[var_name]],
412     var_name = var_name
413   )
414   overdiff_results <- rbind(overdiff_results, res)
415 }
416
417 print(overdiff_results)
418
419
420 ##
    ------------------------------------------------------------------------

421 # Fit a regression model to calculate VIF
422 vif_data <- diff2_final_data %>% dplyr::select(-starts_with("
    Store_"))  # Exogenous variables
423 vif_model <- lm(Store_1 ~ ., data = diff2_final_data[,-1]) #
    Store 1 is a placeholder
424
425 # Calculate VIF scores
```

```r
vif_values <- vif(vif_model)

# Print variables with high VIF
print("Variance Inflation Factors (VIF):")
print(vif_values)

# Identify highly collinear variables (VIF > 10)
high_vif_vars <- names(vif_values[vif_values > 10])

print("Highly Collinear Variables (VIF > 10):")
print(high_vif_vars)


##
    ------------------------------------------------------------------------


remove_vif10 <- diff2_final_data %>% dplyr::select(-one_of(high
    _vif_vars))
var_data <- remove_vif10[,-1]
head(var_data)


##
    ------------------------------------------------------------------------


# Select optimal lag using VARselect()

lag_selection <- VARselect(var_data %>% dplyr::select(starts_
    with("Store_")),  # Select endogenous variables
                           lag.max = 10,  # Set maximum lag to
                              test
```

```
                              type = "const")  # Include constant
                                 term

# Print recommended lag lengths
print("Optimal Lags Selected:")
print(lag_selection$selection)

#Optimal lag for AIC is 10 and BIC is 10


##
    ------------------------------------------------------------------------


# Fit Unregularized VAR Model
var_model <- VAR(y = var_data %>% dplyr::select(starts_with("
    Store_")),  # Endogenous variables
                 p = 10,  # Optimal lag length from VARselect()
                 type = "const",  # Include constant term
                 exogen = var_data %>% dplyr::select(-starts_
                     with("Store_")))  # Exogenous variables

# Print model summary
summary(var_model)


##
    ------------------------------------------------------------------------


# Check stability condition (All eigenvalues should be <1)
stability_test <- roots(var_model)
print(stability_test)

```

```
#Some roota are >1 suggesting instability


##
    ------------------------------------------------------------------


#Get in sample predictions
var_insample_preds <- as.data.frame(fitted(var_model))

#Get actual values
actual_values_in <- var_data[, colnames(var_insample_preds)]

# Ensure data is numeric
actual_values_in <- as.data.frame(sapply(actual_values_in, as.
    numeric))
var_insample_preds <- as.data.frame(sapply(var_insample_preds,
    as.numeric))


##
    ------------------------------------------------------------------


# Function to calculate RMSE, MAE, and MAPE
calculate_metrics <- function(actual, predicted) {
  rmse_value <- rmse(actual, predicted)
  mae_value <- mae(actual, predicted)
  mape_value <- mean(abs((actual - predicted) / actual), na.rm
     = TRUE) * 100


```

```
499   return(c(RMSE = rmse_value, MAE = mae_value, MAPE = mape_
          value))
500 }
501
502 # Apply function correctly
503 error_metrics <- sapply(1:ncol(actual_values_in), function(var)
        {
504   calculate_metrics(actual_values_in[, var], var_insample_preds
          [, var])
505 })
506
507 # Transpose for better readability
508 error_metrics <- as.data.frame(t(error_metrics))
509 colnames(error_metrics) <- c("RMSE", "MAE", "MAPE")
510 rownames(error_metrics) <- colnames(actual_values_in)
511
512 # Print results
513 print("In-Sample Error Metrics for VAR Model:")
514 print(error_metrics)
515
516
517
518
519 ##
       ------------------------------------------------------------------

520 # Compute the average of RMSE, MAE, and MAPE
521 average_in_sample_metrics <- colMeans(error_metrics, na.rm =
       TRUE)
522
523 # Print the average metrics
524 print("Average In Sample Performance Metrics:")
```

```
525 print(average_in_sample_metrics)
526
527
528 ##
    ------------------------------------------------------------------------

529 #Out of Sample
530
531 # Extract all variable names from datamat
532 all_var_names <- colnames(var_model$datamat)
533
534
535 # Get number of endogenous variables
536 num_endog <- var_model$K
537
538 # Define current endogenous variable names (the original ones,
    not lagged)
539 endog_names <- all_var_names[1:num_endog]
540
541 # Identify lagged endogenous variables (they start with the
    same names as endogenous variables)
542 lagged_endog_vars <- all_var_names[1:(num_endog * var_model$p)
    + num_endog]  # Removes all lags of endogenous variables
543
544
545 # Extract the remaining variables (these are true exogenous
    variables)
546 exog_names <- setdiff(all_var_names, c(endog_names, lagged_
    endog_vars))
547
548 # Check if "const" is present
549 if ("const" %in% exog_names) {
```

```r
  # Remove constant from exogenous variable list
  exog_names <- setdiff(exog_names, "const")
}

# Select last known exogenous values and repeat for 10 future steps
future_exog <- var_data %>% dplyr::select(all_of(exog_names)) %>% tail(1)
future_exog <- future_exog[rep(1, 10), ]  # Repeat last known values

# Ensure column names match exactly
colnames(future_exog) <- exog_names

# Convert to matrix
future_exog <- as.matrix(future_exog)

# Forecast next 10 steps using the correct exogenous variables
var_forecast <- predict(var_model, n.ahead = 10, dumvar = future_exog)

# Print forecast summary
print("Forecasted Values:")
print(var_forecast)


## ----------------------------------------------------------------------------


# Extract predicted and actual values

# Extract original endogenous variable names
```

```r
endog_names <- colnames(var_model$y)

# Extract forecasted values
predicted_values <- sapply(endog_names, function(var) var_forecast$fcst[[var]][, "fcst"])
predicted_values <- as.data.frame(predicted_values)

# Extract actual values from the last 10 observations
actual_values <- tail(var_data %>% dplyr::select(all_of(endog_names)), 10)

# Ensure data is numeric
actual_values <- as.data.frame(sapply(actual_values, as.numeric))
predicted_values <- as.data.frame(sapply(predicted_values, as.numeric))


##------------------------------------------------------------------------------


# Function to compute RMSE, MAE, and MAPE
calculate_metrics <- function(actual, predicted) {
  rmse_value <- rmse(actual, predicted)
  mae_value <- mae(actual, predicted)
  mape_value <- mean(abs((actual - predicted) / actual), na.rm = TRUE) * 100
  return(c(RMSE = rmse_value, MAE = mae_value, MAPE = mape_value))
}

```

```r
# Apply metrics to each endogenous variable
performance_metrics <- sapply(1:ncol(actual_values), function(i) {
  calculate_metrics(actual_values[, i], predicted_values[, i])
})

# Convert to readable format
performance_results <- as.data.frame(t(performance_metrics))
colnames(performance_results) <- c("RMSE", "MAE", "MAPE")
rownames(performance_results) <- colnames(actual_values)

# Print final performance metrics
print("Performance Metrics for Unregularized VAR Model:")
print(performance_results)


##------------------------------------------------------------------------


# Compute the average of RMSE, MAE, and MAPE
average_metrics <- colMeans(performance_results, na.rm = TRUE)

# Print the average metrics
print("Average Performance Metrics:")
print(average_metrics)


##------------------------------------------------------------------------


# Select the variable you want to plot (Store 1)
store_var <- "Store_1"
```

```
627
628 # Extract forecast data for Store 1
629 store_forecast <- data.frame(
630   step = 1:10,
631   forecast = var_forecast$fcst[[store_var]][, "fcst"],
632   lower = var_forecast$fcst[[store_var]][, "lower"],
633   upper = var_forecast$fcst[[store_var]][, "upper"]
634 )
635
636 # Get last 10 actual observations for Store 1
637 store_actual <- var_data %>%
638   dplyr::select(all_of(store_var)) %>%
639   tail(10) %>%
640   mutate(time = 1:10, actual = .[[1]]) %>%
641   dplyr::select(time, actual)
642
643 # Shift forecast steps to start from time = 10 (end of actual
    data)
644 store_forecast <- store_forecast %>%
645   mutate(time = 10 + step)
646
647 # Plot actual and forecast for Store 1
648 ggplot() +
649   geom_line(data = store_actual, aes(x = time, y = actual),
      color = "blue", size = 1) +
650   geom_line(data = store_forecast, aes(x = time, y = forecast),
       color = "red", linetype = "dashed", size = 1) +
651   geom_ribbon(data = store_forecast, aes(x = time, ymin = lower
      , ymax = upper), fill = "grey80", alpha = 0.5) +
652   labs(
653     title = paste("Out-of-Sample Unregularized VAR Forecast for
        ", store_var),
```

```
    x = "Time Step",
    y = "Weekly Sales"
  ) +
  theme_minimal(base_size = 14) +
  theme(text = element_text(family = "Times New Roman"))


##
    ------------------------------------------------------------------


# Function to create lags
create_lags <- function(data, p) {
  lags <- lapply(1:p, function(i) {
    lagged <- rbind(matrix(NA, i, ncol(data)), data[1:(nrow(
       data) - i), ])
    colnames(lagged) <- paste0(colnames(data), "_lag", i)
    lagged
  })
  do.call(cbind, lags)
}


##
    ------------------------------------------------------------------


y <- as.matrix(var_data %>% dplyr::select(starts_with("Store_")
    ))
x <- as.matrix(var_data %>% dplyr::select(-starts_with("Store_"
    )))

p = 10
```

```
679
680 # Create lagged endogenous variables (Z_t)
681 y_lags <- create_lags(y, p)
682
683 # Combine all variables
684 X <- cbind(y_lags, x)  # Include current exogenous covariates (
    X_t)
685 #Y <- y[(p + 1):nrow(y), ]
686 Y <- y
687
688 # Remove rows with NA (due to lagging)
689 complete_cases <- complete.cases(X)
690 X <- X[complete_cases, ]
691 Y <- Y[complete_cases, ]
692
693 # Split X into Z_t (lagged endogenous) and X_t (current
    exogenous)
694 #Z_t <- X[, 1:(ncol(y) * p)]  # ncol(y) * p because each
    endogenous variable has p lags
695 #X_t <- X[, (ncol(y) * p + 1):ncol(X)]  # Current exogenous
    variables
696
697
698
699 ##
    ------------------------------------------------------------------
700
701 # Initialize an empty list to store Ridge models for each
    response variable
702 ridge_models <- list()
703
```

```r
# Fit Ridge Regression separately for each column in Y
for (i in 1:ncol(Y)) {
  response_var <- colnames(Y)[i]  # Store name
  ridge_models[[response_var]] <- cv.glmnet(X, Y[, i], alpha =
     0)  # Ridge regression (alpha = 0)
}

# Print Ridge model summaries
print("Fitted Ridge VAR Models:")
print(ridge_models)


##
   ------------------------------------------------------------------------


# Set plotting parameters to use Times New Roman
windowsFonts(Times = windowsFont("Times New Roman"))
par(family = "Times")


##
   ------------------------------------------------------------------------


# Extract the cross-validation results for one response
   variable (e.g., first store)
store_name <- colnames(Y)[1]  # Select first store for
   visualization
cv_ridge <- ridge_models[[store_name]]  # Retrieve the Ridge
   model

# Plot Cross-Validation Error vs. Lambda
plot(cv_ridge)
```

```
728 par(family = "Times")
729
730
731
732 #title(paste("Cross-Validation Error vs. Lambda for", store_
       name))
733
734
735 ##
       ------------------------------------------------------------------
736 # Initialize an empty dataframe to store results
737 results_df <- data.frame(
738   Endogenous_Variable = character(),
739   Optimal_Lambda = numeric(),
740   CV_MSE = numeric(),
741   stringsAsFactors = FALSE
742 )
743
744 # Loop through each cross-validated Ridge model
745 for (i in 1:length(ridge_models)) {
746   response_var <- names(ridge_models)[i]  # Get the name of the
         endogenous variable
747   model <- ridge_models[[i]]  # Get the model for the current
        variable
748
749   # Extract optimal lambda and cross-validated MSE
750   optimal_lambda <- model$lambda.1se
751   cv_mse <- min(model$cvm)
752
753   # Add results to the dataframe
754   results_df <- rbind(results_df, data.frame(
```

```
    Endogenous_Variable = response_var,
    Optimal_Lambda = log(optimal_lambda),
    CV_MSE = cv_mse,
    stringsAsFactors = FALSE
  ))
}

# Print the results dataframe
print("Summary of Cross-Validated Ridge Models:")
print(results_df)


##
    ------------------------------------------------------------------------


# Initialize a list to store predictions
ridge_predictions <- list()

# Predict separately for each response variable (store sales)
for (i in 1:ncol(Y)) {
  store_name <- colnames(Y)[i]  # Store name
  ridge_predictions[[store_name]] <- predict(ridge_models[[
     store_name]], newx = X)  # Get predicted values
}

# Print structure after storing predictions
print("Updated ridge_predictions Structure:")
print(str(ridge_predictions))
```

```
783 ##
    ------------------------------------------------------------------------

784 # Initialize an empty dataframe for performance results
785 ridge_performance_results <- data.frame(Store = character(),
    RMSE = numeric(), MAE = numeric(), MAPE = numeric())
786
787 # Compute metrics for each store
788 for (i in 1:ncol(Y)) {
789   store_name <- colnames(Y)[i]  # Store name
790
791   if (!is.null(ridge_predictions[[store_name]])) {  # Ensure
        predictions exist
792     actual_values <- Y[, i]  # Actual sales values
793     predicted_values <- ridge_predictions[[store_name]][, 1]  #
          Extract the first lambda prediction
794
795     # Compute RMSE, MAE, MAPE
796     rmse_value <- rmse(actual_values, predicted_values)
797     mae_value <- mae(actual_values, predicted_values)
798     mape_value <- mean(abs((actual_values - predicted_values) /
          actual_values), na.rm = TRUE) * 100
799
800     # Store results
801     ridge_performance_results <- rbind(ridge_performance_
        results,
802                                        data.frame(Store = store_name,
                                                   RMSE = rmse_value, MAE =
                                                   mae_value, MAPE = mape_
                                                   value))
803   } else {
```

```
804       print(paste("Warning: No predictions found for", store_name
              ))
805     }
806   }
807
808   # Print final performance metrics
809   print("Performance Metrics for Ridge VAR Model:")
810   print(ridge_performance_results)
811
812
813   ##
        ------------------------------------------------------------------------


814   # Compute the average of RMSE, MAE, and MAPE
815   average_ridge_metrics <- colMeans(ridge_performance_results
        [,-1], na.rm = TRUE)
816
817   # Print the average metrics
818   print("Average ridge Performance Metrics:")
819   print(average_ridge_metrics)
820
821
822   ##
        ------------------------------------------------------------------------


823   store_name <- "Store_1"
824
825   # Extract actual and predicted values for that store
826   actual_values <- Y[, store_name]
827   predicted_values <- ridge_predictions[[store_name]][, 1]  # The
        column with predictions
828
```

```
# Prepare a combined data frame
plot_data <- data.frame(
  time = 1:length(actual_values),
  actual = actual_values,
  predicted = predicted_values
)

# Plot actual vs predicted
ggplot(plot_data, aes(x = time)) +
  geom_line(aes(y = actual), color = "blue", size = 1) +
  geom_line(aes(y = predicted), color = "red", linetype = "
     dashed", size = 1) +
  labs(
    title = paste("Ridge VAR Forecast vs. Actual for", store_
       name),
    x = "Time",
    y = "Weekly Sales"
  ) +
  theme_minimal(base_size = 14) +
  scale_y_continuous(labels = scales::comma)


##
    ------------------------------------------------------------------


#Out of sample
# Define the number of test observations (e.g., last 10 weeks)
test_size <- 10

# Training set (all rows except the last 'test_size' rows)
X_train1 <- X[1:(nrow(X) - test_size), ]
Y_train1 <- Y[1:(nrow(Y) - test_size), ]
```

```
857
858
859 # Test set (last 'test_size' rows)
860 X_test1 <- X[(nrow(X) - test_size + 1):nrow(X), ]
861 Y_test1 <- Y[(nrow(Y) - test_size + 1):nrow(Y), ]
862
863
864 ##
    ------------------------------------------------------------------------

865 # Initialize lists to store trained models
866 ridge_train1_models <- list()
867
868 # Train separate models for each store's sales
869 for (i in 1:ncol(Y_train1)) {
870   store_name <- colnames(Y_train1)[i]  # Store name
871
872   # Ridge Regression (alpha = 0)
873   ridge_train1_models[[store_name]] <- cv.glmnet(X_train1, Y_
        train1[, i], alpha = 0)
874
875 }
876
877
878 ##
    ------------------------------------------------------------------------

879 # Initialize lists to store predictions
880 ridge_train1_predictions <- list()
881
882 # Predict separately for each response variable (store sales)
883
```

```r
for (i in 1:ncol(Y_test1)) {
  store_name <- colnames(Y_test1)[i]  # Store name

  # Generate forecasts
  ridge_train1_predictions[[store_name]] <- (predict(ridge_
     train1_models[[store_name]],newx = X_test1, s = "lambda.1
     se"))

}


##
   ------------------------------------------------------------------------


# Function to compute RMSE, MAE, and MAPE
compute_metrics <- function(actual, predicted) {
  rmse <- sqrt(mean((actual - predicted)^2))
  mae <- mean(abs(actual - predicted))
  mape <- mean(abs((actual - predicted) / actual)) * 100
  return(c(RMSE = rmse, MAE = mae, MAPE = mape))
}

# Store metrics for each model
ridge_train1_metrics <- sapply(1:ncol(Y_test1), function(i) {
  compute_metrics(Y_test1[, i], ridge_train1_predictions[[
     colnames(Y_test1)[i]]])
})

# Convert to data frame and add store names
ridge_train1_metrics <- as.data.frame(t(ridge_train1_metrics))
ridge_train1_metrics$Store <- colnames(Y_test1)  # Add store
   names as a column
```

```r
ridge_train1_metrics <- ridge_train1_metrics[, c("Store", "RMSE", "MAE", "MAPE")]  # Reorder columns

# View results
print(ridge_train1_metrics)




##------------------------------------------------------------------------------

# Compute the average of RMSE, MAE, and MAPE

average_ridge_train1_metrics <- colMeans(ridge_train1_metrics[,-1], na.rm = TRUE)

# Print the average metrics
print("Average ridge_train_metrics:")
print(average_ridge_train1_metrics)


##------------------------------------------------------------------------------

#plot for store 1
store_name <- "Store_1"
actual_values <- as.numeric(Y_test1[, store_name])
predicted_values <- ridge_train1_predictions[[store_name]]

# Create a dataframe for plotting
time_steps <- seq_along(actual_values)
```

```
forecast_steps <- (length(actual_values) + 1):(length(actual_values) + length(predicted_values))

plot_data <- data.frame(
  Time = c(time_steps, forecast_steps),
  Weekly_Sales = c(actual_values, predicted_values),
  Type = c(rep("Actual", length(actual_values)), rep("Forecast", length(predicted_values)))
)

# Plot using ggplot
ggplot(plot_data, aes(x = Time, y = Weekly_Sales, color = Type, linetype = Type)) +
  geom_line(size = 1) +
  scale_color_manual(values = c("Actual" = "blue", "Forecast" = "red")) +
  scale_linetype_manual(values = c("Actual" = "solid", "Forecast" = "dashed")) +
  labs(title = paste("Out-of-Sample Ridge Forecast for", store_name),
       x = "Time Step",
       y = "Weekly Sales") +
  theme_minimal() +
  theme(text = element_text(family = "Times New Roman"))


##------------------------------------------------------------------


# Initialize an empty list to store Lasso models for each response variable
lasso_models <- list()
```

```
959
960 # Fit Lasso Regression separately for each column in Y
961 for (i in 1:ncol(Y)) {
962   response_variable <- colnames(Y)[i]  # Store name
963   lasso_models[[response_variable]] <- cv.glmnet(X, Y[, i],
        alpha = 1)  # Lasso regression (alpha = 1)
964 }
965
966 # Print Lasso model summaries
967 print("Fitted Lasso VAR Models:")
968 print(lasso_models)
969
970
971 ##
    ------------------------------------------------------------------------

972 # Extract the cross-validation results for one response
    variable (e.g., first store)
973 store_name <- colnames(Y)[1]  # Select first store for
    visualization
974 cv_lasso <- lasso_models[[store_name]]  # Retrieve the Lasso
    model
975
976 # Plot Cross-Validation Error vs. Lambda
977 plot(cv_lasso)
978
979
980
981 ##
    ------------------------------------------------------------------------

982 # Initialize an empty dataframe to store results
```

```r
lasso_results_df <- data.frame(
  Endogenous_Variable = character(),
  Optimal_Lambda = numeric(),
  CV_MSE = numeric(),
  stringsAsFactors = FALSE
)

# Loop through each cross-validated Lasso model
for (i in 1:length(lasso_models)) {
  response_variable <- names(lasso_models)[i]  # Get the name of the endogenous variable
  model <- lasso_models[[i]]  # Get the model for the current variable

  # Extract optimal lambda and cross-validated MSE
  optimal_lambda <- model$lambda.1se
  cv_mse <- min(model$cvm)

  # Add results to the dataframe
  lasso_results_df <- rbind(lasso_results_df, data.frame(
    Endogenous_Variable = response_variable,
    Optimal_Lambda = log(optimal_lambda),
    CV_MSE = cv_mse,
    stringsAsFactors = FALSE
  ))
}

# Print the results dataframe
print("Summary of Cross-Validated Lasso Models:")
print(lasso_results_df)
```

```
1013 ##
     ------------------------------------------------------------------------

1014 # Initialize a list to store predictions
1015 lasso_predictions <- list()
1016
1017 # Predict separately for each response variable (store sales)
1018 for (i in 1:ncol(Y)) {
1019   store_name <- colnames(Y)[i]  # Store name
1020   lasso_predictions[[store_name]] <- predict(lasso_models[[
         store_name]], newx = X)  # Get predicted values
1021 }
1022
1023 # Print structure after storing predictions
1024 print("Updated lasso_predictions Structure:")
1025 print(str(lasso_predictions))
1026
1027
1028
1029 ##
     ------------------------------------------------------------------------

1030 # Initialize an empty dataframe for performance results
1031 lasso_performance_results <- data.frame(Store = character(),
     RMSE = numeric(), MAE = numeric(), MAPE = numeric())
1032
1033 # Compute metrics for each store
1034 for (i in 1:ncol(Y)) {
1035   store_name <- colnames(Y)[i]  # Store name
1036
1037   if (!is.null(lasso_predictions[[store_name]])) {  # Ensure
         predictions exist
```

```
1038      actual_values <- Y[, i]  # Actual sales values
1039      predicted_values <- lasso_predictions[[store_name]][, 1]  #
              Extract the first lambda prediction
1040
1041      # Compute RMSE, MAE, MAPE
1042      rmse_value <- rmse(actual_values, predicted_values)
1043      mae_value <- mae(actual_values, predicted_values)
1044      mape_value <- mean(abs((actual_values - predicted_values) /
              actual_values), na.rm = TRUE) * 100
1045
1046      # Store results
1047      lasso_performance_results <- rbind(lasso_performance_
             results,
1048                                         data.frame(Store = store_name,
                                                RMSE = rmse_value, MAE =
                                                mae_value, MAPE = mape_
                                                value))
1049    } else {
1050      print(paste("Warning: No predictions found for", store_name
             ))
1051    }
1052  }
1053
1054  # Print final performance metrics
1055  print("Performance Metrics for Lasso VAR Model:")
1056  print(lasso_performance_results)
1057
1058
1059  ##
       ------------------------------------------------------------------
1060  # Compute the average of RMSE, MAE, and MAPE
```

```r
average_lasso_metrics <- colMeans(lasso_performance_results[,-1], na.rm = TRUE)

# Print the average metrics
print("Average Lasso Performance Metrics:")
print(average_lasso_metrics)


##------------------------------------------------------------------

# Extract all predictor names from X (including lagged endogenous variables and covariates)
all_predictors <- colnames(X)  # Assuming X contains all predictors used in the model

# Create an empty data frame for the feature selection table
feature_table <- data.frame(Predictor = all_predictors)  # Now includes both lagged Y and covariates


##------------------------------------------------------------------

# Initialize an empty list to store selected features
lasso_selected_features <- list()

# Loop through each Lasso model and extract non-zero coefficients
for (response_var in names(lasso_models)) {
  # Get the fitted Lasso model
```

```r
  model <- lasso_models[[response_var]]

  # Extract coefficients at the best lambda (lambda.min)
  coeffs <- coef(model, s = "lambda.min")

  # Convert to a data frame
  coeffs_df <- as.data.frame(as.matrix(coeffs))

  # Get names of non-zero coefficients (excluding intercept)
  nonzero_features <- rownames(coeffs_df)[coeffs_df$`s1` != 0]
  nonzero_features <- nonzero_features[nonzero_features != "(Intercept)"]  # Remove intercept

  # Store results
  lasso_selected_features[[response_var]] <- nonzero_features
}

# Add selection results for each endogenous variable
for (response_var in names(lasso_selected_features)) {
  feature_table[[response_var]] <- ifelse(feature_table$Predictor %in% lasso_selected_features[[response_var]], "Yes", "No")
}

# Print the final feature selection table
print(feature_table)



##
```

```r
# Combine all selected features across response variables
selected_all <- unlist(lasso_selected_features)  # Flatten list of selected features

# Count occurrences of each predictor
feature_counts <- table(selected_all)

# Convert to a data frame
feature_summary <- as.data.frame(feature_counts)
colnames(feature_summary) <- c("Predictor", "Frequency")

# Sort predictors by frequency of selection (descending order)
feature_summary <- feature_summary[order(-feature_summary$Frequency), ]

# Display only the top 10 predictors
top_features <- head(feature_summary, 10)  # Adjust the number as needed
print(top_features)



##------------------------------------------------------------------------


# Display only the top 10 or 20 predictors
bottom_features <- tail(feature_summary, 10)  # Adjust the number as needed
print(bottom_features)


```

```
##
    ------------------------------------------------------------------------


# Initialize lists to store trained models
lasso_train1_models <- list()

# Train separate models for each store's sales
for (i in 1:ncol(Y_train1)) {
  store_name <- colnames(Y_train1)[i]  # Store name

  # Lasso Regression (alpha = 1)
  lasso_train1_models[[store_name]] <- cv.glmnet(X_train1, Y_
     train1[, i], alpha = 1)

}


##
    ------------------------------------------------------------------------


# Initialize lists to store predictions
lasso_train1_predictions <- list()

# Predict separately for each response variable (store sales)

for (i in 1:ncol(Y_test1)) {
  store_name <- colnames(Y_test1)[i]  # Store name

  # Generate forecasts
  lasso_train1_predictions[[store_name]] <- (predict(lasso_
     train1_models[[store_name]],newx = X_test1, s = "lambda.1
     se"))
```

```
1161
1162 }
1163
1164
1165 ##
     ------------------------------------------------------------------------
1166 # Function to compute RMSE, MAE, and MAPE
1167 compute_metrics <- function(actual, predicted) {
1168   rmse <- sqrt(mean((actual - predicted)^2))
1169   mae <- mean(abs(actual - predicted))
1170   mape <- mean(abs((actual - predicted) / actual)) * 100
1171   return(c(RMSE = rmse, MAE = mae, MAPE = mape))
1172 }
1173
1174 # Store metrics for each model
1175 lasso_train1_metrics <- sapply(1:ncol(Y_test1), function(i) {
1176   compute_metrics(Y_test1[, i], lasso_train1_predictions[[
        colnames(Y_test1)[i]]])
1177 })
1178
1179 # Convert to data frame and add store names
1180 lasso_train1_metrics <- as.data.frame(t(lasso_train1_metrics))
1181 lasso_train1_metrics$Store <- colnames(Y_test1)  # Add store
     names as a column
1182 lasso_train1_metrics <- lasso_train1_metrics[, c("Store", "RMSE
     ", "MAE", "MAPE")]  # Reorder columns
1183
1184 # View results
1185 print(lasso_train1_metrics)
1186
1187
```

```
1188
1189
1190 ##
     ----------------------------------------------------------------------

1191 # Compute the average of RMSE, MAE, and MAPE
1192
1193 average_lasso_train1_metrics <- colMeans(lasso_train1_metrics
     [,-1], na.rm = TRUE)
1194
1195 # Print the average metrics
1196 print("Average lasso_train_metrics:")
1197 print(average_lasso_train1_metrics)
1198
1199
1200 ##
     ----------------------------------------------------------------------

1201 #plot for store 1
1202 store_name <- "Store_1"
1203 actual_values <- as.numeric(Y_test1[, store_name])
1204 predicted_values <- lasso_train1_predictions[[store_name]]
1205
1206 # Create a dataframe for plotting
1207 time_steps <- seq_along(actual_values)
1208 forecast_steps <- (length(actual_values) + 1):(length(actual_
     values) + length(predicted_values))
1209
1210 plot_data <- data.frame(
1211   Time = c(time_steps, forecast_steps),
1212   Weekly_Sales = c(actual_values, predicted_values),
```

```
  Type = c(rep("Actual", length(actual_values)), rep("Forecast"
    , length(predicted_values)))
)

# Plot using ggplot
ggplot(plot_data, aes(x = Time, y = Weekly_Sales, color = Type,
    linetype = Type)) +
  geom_line(size = 1) +
  scale_color_manual(values = c("Actual" = "blue", "Forecast" =
     "red")) +
  scale_linetype_manual(values = c("Actual" = "solid", "
    Forecast" = "dashed")) +
  labs(title = paste("Out-of-Sample Lasso Forecast for", store_
    name),
       x = "Time Step",
       y = "Weekly Sales") +
  theme_minimal() +
  theme(text = element_text(family = "Times New Roman"))


##
    ------------------------------------------------------------


# Define a sequence of alpha values to try (from pure Ridge to
   pure Lasso)
alpha_values <- seq(0, 1, by = 0.1)

# Initialize a list to store the best models for each response
   variable
eNet_models <- list()
best_alpha <- list()
```

```r
best_lambda <- list()

# Fit Elastic Net separately for each column in Y
for (i in 1:ncol(Y)) {
  response_var <- colnames(Y)[i]  # Store variable name
  cv_results <- list()  # Store cross-validation results for different alphas

  # Loop over different alpha values
  for (alpha in alpha_values) {
    cv_model <- cv.glmnet(X, Y[, i], alpha = alpha)  # Cross-validated Elastic Net
    cv_results[[as.character(alpha)]] <- cv_model
  }

  # Find the best alpha (minimizing cross-validation error)
  min_mse <- sapply(cv_results, function(model) min(model$cvm))  # Get min CV error for each alpha
  best_alpha_index <- which.min(min_mse)
  best_alpha[[response_var]] <- alpha_values[best_alpha_index]

  # Store the best model (corresponding to the best alpha)
  eNet_models[[response_var]] <- cv_results[[as.character(best_alpha[[response_var]])]]
  best_lambda[[response_var]] <- eNet_models[[response_var]]$lambda.1se  # Store optimal lambda
}

# Print best alpha and lambda values for each response variable
print("Best Alpha Values for Each Variable:")
print(best_alpha)

```

```
print("Best Lambda Values for Each Variable:")
print(best_lambda)



##
    ------------------------------------------------------------------------


# Print Elastic model summaries
print("Fitted Elastic Net VAR Models:")
print(eNet_models)


##
    ------------------------------------------------------------------------


# Extract the cross-validation results for one response
    variable (e.g., first store)
store_name <- colnames(Y)[1]  # Select first store for
    visualization
cv_eNet <- eNet_models[[store_name]]  # Retrieve the eNet model

# Plot Cross-Validation Error vs. Lambda
plot(cv_eNet)


##
    ------------------------------------------------------------------------


# Initialize an empty dataframe to store results
eNet_results_df <- data.frame(
  Endogenous_Variable = character(),
```

```
1287    Alpha = numeric(),
1288    Optimal_Lambda = numeric(),
1289    CV_MSE = numeric(),
1290    stringsAsFactors = FALSE
1291  )
1292
1293  # Loop through each cross-validated elastic net model
1294  for (i in 1:length(eNet_models)) {
1295    response_var <- names(eNet_models)[i]  # Get the name of the
          endogenous variable
1296    model <- eNet_models[[i]]  # Get the model for the current
          variable
1297
1298    # Extract optimal lambda and cross-validated MSE
1299    optimal_lambda <- model$lambda.1se
1300    cv_mse <- min(model$cvm)
1301    alpha_value <- best_alpha[[response_var]]
1302
1303    # Add results to the dataframe
1304    eNet_results_df <- rbind(eNet_results_df, data.frame(
1305      Endogenous_Variable = response_var,
1306      Optimal_Alpha = alpha_value,
1307      Optimal_Lambda = log(optimal_lambda),
1308      CV_MSE = cv_mse,
1309      stringsAsFactors = FALSE
1310    ))
1311  }
1312
1313  # Print the results dataframe
1314  print("Summary of Cross-Validated Elastic Net Models:")
1315  print(eNet_results_df)
1316
```

```
1317
1318 ##
     ------------------------------------------------------------------
1319 # Initialize a list to store predictions
1320 eNet_predictions <- list()
1321
1322 # Predict separately for each response variable (store sales)
1323 for (i in 1:ncol(Y)) {
1324   store_name <- colnames(Y)[i]  # Store name
1325   eNet_predictions[[store_name]] <- predict(eNet_models[[store_
        name]], newx = X)  # Get predicted values
1326 }
1327
1328 # Print structure after storing predictions
1329 print("Updated eNet_predictions Structure:")
1330 print(str(eNet_predictions))
1331
1332
1333
1334 ##
     ------------------------------------------------------------------
1335 # Initialize an empty dataframe for performance results
1336 eNet_performance_results <- data.frame(Store = character(),
     RMSE = numeric(), MAE = numeric(), MAPE = numeric())
1337
1338 # Compute metrics for each store
1339 for (i in 1:ncol(Y)) {
1340   store_name <- colnames(Y)[i]  # Store name
1341
```

```
1342   if (!is.null(eNet_predictions[[store_name]])) {  # Ensure
       predictions exist
1343     actual_values <- Y[, i]  # Actual sales values
1344     predicted_values <- eNet_predictions[[store_name]][, 1]  #
         Extract the first lambda prediction
1345
1346     # Compute RMSE, MAE, MAPE
1347     rmse_value <- rmse(actual_values, predicted_values)
1348     mae_value <- mae(actual_values, predicted_values)
1349     mape_value <- mean(abs((actual_values - predicted_values) /
         actual_values), na.rm = TRUE) * 100
1350
1351     # Store results
1352     eNet_performance_results <- rbind(eNet_performance_results,
1353                                       data.frame(Store = store_name,
                                                     RMSE = rmse_value, MAE =
                                                     mae_value, MAPE = mape_
                                                     value))
1354   } else {
1355     print(paste("Warning: No predictions found for", store_name
         ))
1356   }
1357 }
1358
1359 # Print final performance metrics
1360 print("Performance Metrics for Elastic Net VAR Model:")
1361 print(eNet_performance_results)
1362
1363
1364 ##
```

```r
# Compute the average of RMSE, MAE, and MAPE
average_eNet_metrics <- colMeans(eNet_performance_results[,-1], na.rm = TRUE)

# Print the average metrics
print("Average Elastic Net Performance Metrics:")
print(average_eNet_metrics)


##------------------------------------------------------------------------


# Extract all predictor names from X (including lagged endogenous variables and covariates)
all_predictors <- colnames(X)  # Assuming X contains all predictors used in the model

# Create an empty data frame for the feature selection table
eNet_feature_table <- data.frame(Predictor = all_predictors)  # Now includes both lagged Y and covariates


##------------------------------------------------------------------------


# Initialize an empty list to store selected features
eNet_selected_features <- list()

# Loop through each Lasso model and extract non-zero coefficients
for (response_var in names(eNet_models)) {
```

```r
  # Get the fitted Lasso model
  model <- eNet_models[[response_var]]

  # Extract coefficients at the best lambda (lambda.min)
  coeffs <- coef(model, s = "lambda.min")

  # Convert to a data frame
  coeffs_df <- as.data.frame(as.matrix(coeffs))

  # Get names of non-zero coefficients (excluding intercept)
  nonzero_features <- rownames(coeffs_df)[coeffs_df$`s1` != 0]
  nonzero_features <- nonzero_features[nonzero_features != "(Intercept)"]  # Remove intercept

  # Store results
  eNet_selected_features[[response_var]] <- nonzero_features
}

# Add selection results for each endogenous variable
for (response_var in names(eNet_selected_features)) {
  eNet_feature_table[[response_var]] <- ifelse(eNet_feature_table$Predictor %in% eNet_selected_features[[response_var]], "Yes", "No")
}

# Print the final feature selection table
print(eNet_feature_table)
```

```
1415 ##
     ------------------------------------------------------------------------

1416 # Combine all selected features across response variables
1417 eNet_selected_all <- unlist(eNet_selected_features)  # Flatten
     list of selected features
1418
1419 # Count occurrences of each predictor
1420 eNet_feature_counts <- table(eNet_selected_all)
1421
1422 # Convert to a data frame
1423 eNet_feature_summary <- as.data.frame(eNet_feature_counts)
1424 colnames(eNet_feature_summary) <- c("Predictor", "Frequency")
1425
1426 # Sort predictors by frequency of selection (descending order)
1427 eNet_feature_summary <- eNet_feature_summary[order(-eNet_
     feature_summary$Frequency), ]
1428
1429 # Display only the top 10 or 20 predictors
1430 eNet_top_features <- head(eNet_feature_summary, 10)  # Adjust
     the number as needed
1431 print(eNet_top_features)
1432
1433
1434
1435 ##
     ------------------------------------------------------------------------

1436 #Out of sample forecasts
1437
1438 # Define a sequence of alpha values to try (from pure Ridge to
     pure Lasso)
```

```r
alpha_values <- seq(0, 1, by = 0.1)

# Initialize a list to store the best models for each response variable
eNet_train1_models <- list()
best_alpha_train1 <- list()
best_lambda_train1 <- list()

# Fit Elastic Net separately for each column in Y
for (i in 1:ncol(Y_train1)) {
  response_var_train1 <- colnames(Y_train1)[i]  # Store variable name
  cv_results_train1 <- list()  # Store cross-validation results for different alphas

  # Loop over different alpha values
  for (alpha in alpha_values) {
    cv_model_train1 <- cv.glmnet(X_train1, Y_train1[, i], alpha = alpha)  # Cross-validated Elastic Net
    cv_results_train1[[as.character(alpha)]] <- cv_model_train1
  }

  # Find the best alpha (minimizing cross-validation error)
  min_mse_train1 <- sapply(cv_results_train1, function(model) min(model$cvm))  # Get min CV error for each alpha
  best_alpha_index <- which.min(min_mse_train1)
  best_alpha_train1[[response_var_train1]] <- alpha_values[best_alpha_index]

  # Store the best model (corresponding to the best alpha)
```

```
  eNet_train1_models[[response_var_train1]] <- cv_results_
    train1[[as.character(best_alpha_train1[[response_var_
    train1]])]]
  best_lambda_train1[[response_var_train1]] <- eNet_train1_
    models[[response_var_train1]]$lambda.min  # Store optimal
    lambda
}


##
    ------------------------------------------------------------------------


# Initialize lists to store predictions
eNet_train1_predictions <- list()

# Predict separately for each response variable (store sales)

for (i in 1:ncol(Y_test1)) {
  store_name <- colnames(Y_test1)[i]  # Store name

  # Generate forecasts
  eNet_train1_predictions[[store_name]] <- (predict(eNet_train1
    _models[[store_name]],newx = X_test1, s = "lambda.min"))

}


##
    ------------------------------------------------------------------------


# Function to compute RMSE, MAE, and MAPE
compute_metrics <- function(actual, predicted) {
```

```
1486   rmse <- sqrt(mean((actual - predicted)^2))
1487   mae <- mean(abs(actual - predicted))
1488   mape <- mean(abs((actual - predicted) / actual)) * 100
1489   return(c(RMSE = rmse, MAE = mae, MAPE = mape))
1490 }
1491
1492 # Store metrics for each model
1493 eNet_train1_metrics <- sapply(1:ncol(Y_test1), function(i) {
1494   compute_metrics(Y_test1[, i], eNet_train1_predictions[[
         colnames(Y_test1)[i]]])
1495 })
1496
1497 # Convert to data frame and add store names
1498 eNet_train1_metrics <- as.data.frame(t(eNet_train1_metrics))
1499 eNet_train1_metrics$Store <- colnames(Y_test1)  # Add store
       names as a column
1500 eNet_train1_metrics <- eNet_train1_metrics[, c("Store", "RMSE",
       "MAE", "MAPE")]  # Reorder columns
1501
1502 # View results
1503 print(eNet_train1_metrics)
1504
1505
1506
1507
1508 ##
       ------------------------------------------------------------------
1509 # Compute the average of RMSE, MAE, and MAPE
1510 average_eNet_train1_metrics <- colMeans(eNet_train1_metrics
       [,-1], na.rm = TRUE)
1511
```

```
# Print the average metrics
print("Average eNet_train_metrics:")
print(average_eNet_train1_metrics)


##
    ------------------------------------------------------------------------


#plot for store 1
store_name <- "Store_1"
actual_values <- as.numeric(Y_test1[, store_name])
predicted_values <- eNet_train1_predictions[[store_name]]

# Create a dataframe for plotting
time_steps <- seq_along(actual_values)
forecast_steps <- (length(actual_values) + 1):(length(actual_
    values) + length(predicted_values))

plot_data <- data.frame(
  Time = c(time_steps, forecast_steps),
  Weekly_Sales = c(actual_values, predicted_values),
  Type = c(rep("Actual", length(actual_values)), rep("Forecast"
      , length(predicted_values)))
)

# Plot using ggplot
ggplot(plot_data, aes(x = Time, y = Weekly_Sales, color = Type,
     linetype = Type)) +
  geom_line(size = 1) +
  scale_color_manual(values = c("Actual" = "blue", "Forecast" =
      "red")) +
```

```
  scale_linetype_manual(values = c("Actual" = "solid", "
    Forecast" = "dashed")) +
  labs(title = paste("Out-of-Sample Elastic Net Forecast for",
    store_name),
       x = "Time Step",
       y = "Weekly Sales") +
  theme_minimal() +
  theme(text = element_text(family = "Times New Roman"))


##
  ------------------------------------------------------------------------


full_data <- diff2_final_data[,-1]
head(full_data)


##
  ------------------------------------------------------------------------


y_full <- as.matrix(full_data %>% dplyr::select(starts_with("
  Store_")))
x_full <- as.matrix(full_data %>% dplyr::select(-starts_with("
  Store_")))

p = 10

# Create lagged endogenous variables (Z_t)
y_lags_full <- create_lags(y_full, p)

# Combine all variables
```

```
X_full <- cbind(y_lags_full, x_full)  # Include current exogenous covariates (X_t)
Y_full <- y_full[(p + 1):nrow(y_full), ]
Y_full <- y_full

# Remove rows with NA (due to lagging)
complete_cases <- complete.cases(X_full)
X_full <- X_full[complete_cases, ]
Y_full <- Y_full[complete_cases, ]

# Split X into Z_t (lagged endogenous) and X_t (current exogenous)
#Z_t <- X[, 1:(ncol(y) * p)]  # ncol(y) * p because each endogenous variable has p lags
#X_t <- X[, (ncol(y) * p + 1):ncol(X)]  # Current exogenous variables



##------------------------------------------------------------------------------


# Initialize an empty list to store Ridge models for each response variable
ridge_full_models <- list()

# Fit Ridge Regression separately for each column in Y
for (i in 1:ncol(Y_full)) {
  response_var <- colnames(Y_full)[i]  # Store name
  ridge_full_models[[response_var]] <- cv.glmnet(X_full, Y_full[, i], alpha = 0)  # Ridge regression (alpha = 0)
```

```
}

# Print Ridge model summaries
print("Fitted Ridge VAR Models:")
print(ridge_full_models)


##
    ------------------------------------------------------------------------


# Extract the cross-validation results for one response
    variable (e.g., first store)
store_name <- colnames(Y_full)[10]  # Select first store for
    visualization
cv_ridge_full <- ridge_full_models[[store_name]]  # Retrieve
    the Ridge model

# Plot Cross-Validation Error vs. Lambda
plot(cv_ridge_full)


##
    ------------------------------------------------------------------------


# Initialize an empty dataframe to store results
ridge_full_results_df <- data.frame(
  Endogenous_Variable = character(),
  Optimal_Lambda = numeric(),
  CV_MSE = numeric(),
  stringsAsFactors = FALSE
)

```

```r
# Loop through each cross-validated Ridge model
for (i in 1:length(ridge_full_models)) {
  response_var <- names(ridge_full_models)[i]  # Get the name of the endogenous variable
  model <- ridge_full_models[[i]]  # Get the model for the current variable

  # Extract optimal lambda and cross-validated MSE
  optimal_lambda <- model$lambda.1se
  cv_mse <- min(model$cvm)

  # Add results to the dataframe
  ridge_full_results_df <- rbind(ridge_full_results_df, data.frame(
    Endogenous_Variable = response_var,
    Optimal_Lambda = log(optimal_lambda),
    CV_MSE = cv_mse,
    stringsAsFactors = FALSE
  ))
}

# Print the results dataframe
print("Summary of Cross-Validated Ridge Models:")
print(ridge_full_results_df)


## ------------------------------------------------------------------------


# Initialize a list to store predictions
ridge_full_predictions <- list()

```

```r
# Predict separately for each response variable (store sales)
for (i in 1:ncol(Y_full)) {
  store_name <- colnames(Y_full)[i]  # Store name
  ridge_full_predictions[[store_name]] <- predict(ridge_full_models[[store_name]], newx = X_full)  # Get predicted values
}

# Print structure after storing predictions
print("Updated ridge_predictions Structure:")
print(str(ridge_full_predictions))



##------------------------------------------------------------------------------


# Initialize an empty dataframe for performance results
ridge_full_performance_results <- data.frame(Store = character(), RMSE = numeric(), MAE = numeric(), MAPE = numeric())

# Compute metrics for each store
for (i in 1:ncol(Y_full)) {
  store_name <- colnames(Y_full)[i]  # Store name

  if (!is.null(ridge_full_predictions[[store_name]])) {  # Ensure predictions exist
    actual_values <- Y_full[, i]  # Actual sales values
    predicted_values <- ridge_full_predictions[[store_name]][, 1]  # Extract the first lambda prediction

    # Compute RMSE, MAE, MAPE
```

```
1661      rmse_value <- rmse(actual_values, predicted_values)
1662      mae_value <- mae(actual_values, predicted_values)
1663      mape_value <- mean(abs((actual_values - predicted_values) /
              actual_values), na.rm = TRUE) * 100
1664
1665      # Store results
1666      ridge_full_performance_results <- rbind(ridge_full_
             performance_results,
1667                                              data.frame(Store = store_name,
                                                     RMSE = rmse_value, MAE =
                                                    mae_value, MAPE = mape_
                                                    value))
1668    } else {
1669      print(paste("Warning: No predictions found for", store_name
             ))
1670    }
1671  }
1672
1673  # Print final performance metrics
1674  print("Performance Metrics for full Ridge VAR Model:")
1675  print(ridge_full_performance_results)
1676
1677
1678  ##
        ------------------------------------------------------------------------


1679  # Compute the average of RMSE, MAE, and MAPE
1680  average_ridge_full_metrics <- colMeans(ridge_full_performance_
        results[,-1], na.rm = TRUE)
1681
1682  # Print the average metrics
1683  print("Average Ridge Performance Metrics:")
```

```
print(average_ridge_full_metrics)


##
    ------------------------------------------------------------------------


# Initialize an empty list to store Lasso models for each
    response variable
lasso_full_models <- list()

# Fit Lasso Regression separately for each column in Y
for (i in 1:ncol(Y_full)) {
  response_var <- colnames(Y_full)[i]  # Store name
  lasso_full_models[[response_var]] <- cv.glmnet(X_full, Y_full
     [, i], alpha = 1)  # Lasso regression (alpha = 1)
}

# Print Lasso model summaries
print("Fitted Lasso VAR Models:")
print(lasso_full_models)


##
    ------------------------------------------------------------------------


# Extract the cross-validation results for one response
    variable (e.g., first store)
store_name <- colnames(Y_full)[10]  # Select first store for
    visualization
cv_lasso_full <- lasso_full_models[[store_name]]  # Retrieve
    the lasso model

```

```r
# Plot Cross-Validation Error vs. Lambda
plot(cv_lasso_full)


##
    ------------------------------------------------------------------------


# Initialize an empty dataframe to store results
lasso_full_results_df <- data.frame(
  Endogenous_Variable = character(),
  Optimal_Lambda = numeric(),
  CV_MSE = numeric(),
  stringsAsFactors = FALSE
)

# Loop through each cross-validated Lasso model
for (i in 1:length(lasso_full_models)) {
  response_var <- names(lasso_full_models)[i]  # Get the name
     of the endogenous variable
  model <- lasso_full_models[[i]]  # Get the model for the
     current variable

  # Extract optimal lambda and cross-validated MSE
  optimal_lambda <- model$lambda.1se
  cv_mse <- min(model$cvm)

  # Add results to the dataframe
  lasso_full_results_df <- rbind(lasso_full_results_df, data.
     frame(
    Endogenous_Variable = response_var,
    Optimal_Lambda = log(optimal_lambda),
    CV_MSE = cv_mse,
```

```
    stringsAsFactors = FALSE
  ))
}

# Print the results dataframe
print("Summary of Cross-Validated Lasso Models:")
print(lasso_full_results_df)


##
    ------------------------------------------------------------------------


# Initialize a list to store predictions
lasso_full_predictions <- list()

# Predict separately for each response variable (store sales)
for (i in 1:ncol(Y_full)) {
  store_name <- colnames(Y_full)[i]  # Store name
  lasso_full_predictions[[store_name]] <- predict(lasso_full_
     models[[store_name]], newx = X_full)  # Get predicted
     values
}

# Print structure after storing predictions
print("Updated lasso_full_predictions Structure:")
print(str(lasso_full_predictions))


##
    ------------------------------------------------------------------------
```

```r
# Initialize an empty dataframe for performance results
lasso_full_performance_results <- data.frame(Store = character(), RMSE = numeric(), MAE = numeric(), MAPE = numeric())

# Compute metrics for each store
for (i in 1:ncol(Y_full)) {
  store_name <- colnames(Y_full)[i]  # Store name

  if (!is.null(lasso_full_predictions[[store_name]])) {  # Ensure predictions exist
    actual_values <- Y_full[, i]  # Actual sales values
    predicted_values <- lasso_full_predictions[[store_name]][, 1]  # Extract the first lambda prediction

    # Compute RMSE, MAE, MAPE
    rmse_value <- rmse(actual_values, predicted_values)
    mae_value <- mae(actual_values, predicted_values)
    mape_value <- mean(abs((actual_values - predicted_values) / actual_values), na.rm = TRUE) * 100

    # Store results
    lasso_full_performance_results <- rbind(lasso_full_performance_results,
                                            data.frame(Store = store_name, RMSE = rmse_value, MAE = mae_value, MAPE = mape_value))
  } else {
    print(paste("Warning: No predictions found for", store_name))
  }
}
```

```
1783
1784 # Print final performance metrics
1785 print("Performance Metrics for full Lasso VAR Model:")
1786 print(lasso_full_performance_results)
1787
1788
1789 ##
    ------------------------------------------------------------------------

1790 # Compute the average of RMSE, MAE, and MAPE
1791 average_lasso_full_metrics <- colMeans(lasso_full_performance_
    results[,-1], na.rm = TRUE)
1792
1793 # Print the average metrics
1794 print("Average Lasso Performance Metrics:")
1795 print(average_lasso_full_metrics)
1796
1797
1798 ##
    ------------------------------------------------------------------------

1799 # Extract all predictor names from X (including lagged
    endogenous variables and covariates)
1800 all_predictors <- colnames(X_full)  # Assuming X contains all
    predictors used in the model
1801
1802 # Create an empty data frame for the feature selection table
1803 lasso_full_feature_table <- data.frame(Predictor = all_
    predictors)  # Now includes both lagged Y and covariates
1804
1805
1806
```

```
1807 ##
     ------------------------------------------------------------------------

1808 # Initialize an empty list to store selected features
1809 lasso_full_selected_features <- list()
1810
1811 # Loop through each Lasso model and extract non-zero
     coefficients
1812 for (response_var in names(lasso_full_models)) {
1813   # Get the fitted Lasso model
1814   model <- lasso_full_models[[response_var]]
1815
1816   # Extract coefficients at the best lambda (lambda.min)
1817   coeffs <- coef(model, s = "lambda.min")
1818
1819   # Convert to a data frame
1820   coeffs_df <- as.data.frame(as.matrix(coeffs))
1821
1822   # Get names of non-zero coefficients (excluding intercept)
1823   nonzero_features <- rownames(coeffs_df)[coeffs_df$`s1` != 0]
1824   nonzero_features <- nonzero_features[nonzero_features != "(
        Intercept)"]  # Remove intercept
1825
1826   # Store results
1827   lasso_full_selected_features[[response_var]] <- nonzero_
        features
1828 }
1829
1830 # Add selection results for each endogenous variable
1831 for (response_var in names(lasso_full_selected_features)) {
```

```
1832    lasso_full_feature_table[[response_var]] <- ifelse(lasso_full
          _feature_table$Predictor %in% lasso_full_selected_features
          [[response_var]], "Yes", "No")
1833  }
1834
1835  # Print the final feature selection table
1836  print(lasso_full_feature_table)
1837
1838
1839
1840  ##
        ------------------------------------------------------------------
1841  # Combine all selected features across response variables
1842  lasso_full_selected_all <- unlist(lasso_full_selected_features)
          # Flatten list of selected features
1843
1844  # Count occurrences of each predictor
1845  lasso_full_feature_counts <- table(lasso_full_selected_all)
1846
1847  # Convert to a data frame
1848  lasso_full_feature_summary <- as.data.frame(lasso_full_feature_
        counts)
1849  colnames(lasso_full_feature_summary) <- c("Predictor", "
        Frequency")
1850
1851  # Sort predictors by frequency of selection (descending order)
1852  lasso_full_feature_summary <- lasso_full_feature_summary[order
        (-lasso_full_feature_summary$Frequency), ]
1853
1854  # Display only the top 10 or 20 predictors
```

```
lasso_full_top_features <- head(lasso_full_feature_summary, 10)
    # Adjust the number as needed
print(lasso_full_top_features)



##
   ------------------------------------------------------------------------


# Define a sequence of alpha values to try (from pure Ridge to
   pure Lasso)
alpha_values <- seq(0, 1, by = 0.1)

# Initialize a list to store the best models for each response
   variable
eNet_full_models <- list()
best_alpha_full <- list()
best_lambda_full <- list()

# Fit Elastic Net separately for each column in Y
for (i in 1:ncol(Y_full)) {
  response_var_full <- colnames(Y_full)[i]  # Store variable
     name
  cv_results_full <- list()  # Store cross-validation results
     for different alphas

  # Loop over different alpha values
  for (alpha in alpha_values) {
    cv_model_full <- cv.glmnet(X_full, Y_full[, i], alpha =
       alpha)  # Cross-validated Elastic Net
    cv_results_full[[as.character(alpha)]] <- cv_model_full
  }
```

```
1879
1880    # Find the best alpha (minimizing cross-validation error)
1881    min_mse_full <- sapply(cv_results_full, function(model) min(
            model$cvm))  # Get min CV error for each alpha
1882    best_alpha_index <- which.min(min_mse_full)
1883    best_alpha_full[[response_var_full]] <- alpha_values[best_
            alpha_index]
1884
1885    # Store the best model (corresponding to the best alpha)
1886    eNet_full_models[[response_var_full]] <- cv_results_full[[as.
            character(best_alpha_full[[response_var_full]])]]
1887    best_lambda_full[[response_var_full]] <- eNet_full_models[[
            response_var_full]]$lambda.1se  # Store optimal lambda
1888 }
1889
1890 # Print best alpha and lambda values for each response variable
1891 print("Best Alpha Values for Each Variable:")
1892 print(best_alpha_full)
1893
1894 print("Best Lambda Values for Each Variable:")
1895 print(best_lambda_full)
1896
1897
1898
1899 ##
       ------------------------------------------------------------------------
1900 # Print Elastic Net model summaries
1901 print("Fitted Elastic Net VAR Models:")
1902 print(eNet_full_models)
1903
1904
```

```
##
    ------------------------------------------------------------------------

# Extract the cross-validation results for one response
    variable (e.g., first store)
store_name <- colnames(Y_full)[10]
cv_eNet_full <- eNet_full_models[[store_name]]  # Retrieve the
    elastic net model

# Plot Cross-Validation Error vs. Lambda
plot(cv_eNet_full)


##
    ------------------------------------------------------------------------

# Initialize an empty dataframe to store results
eNet_full_results_df <- data.frame(
  Endogenous_Variable = character(),
  Alpha = numeric(),
  Optimal_Lambda = numeric(),
  CV_MSE = numeric(),
  stringsAsFactors = FALSE
)

# Loop through each cross-validated Ridge model
for (i in 1:length(eNet_full_models)) {
  response_var <- names(eNet_full_models)[i]  # Get the name of
      the endogenous variable
  model <- eNet_full_models[[i]]  # Get the model for the
     current variable

```

```
1929    # Extract optimal lambda and cross-validated MSE
1930    optimal_lambda <- log(model$lambda.1se)
1931    cv_mse <- min(model$cvm)
1932    alpha_value <- best_alpha_full[[response_var]]
1933
1934    # Add results to the dataframe
1935    eNet_full_results_df <- rbind(eNet_full_results_df, data.
           frame(
1936      Endogenous_Variable = response_var,
1937      Optimal_alpha = alpha_value,
1938      Optimal_Lambda = optimal_lambda,
1939      CV_MSE = cv_mse,
1940      stringsAsFactors = FALSE
1941    ))
1942  }
1943
1944  # Print the results dataframe
1945  print("Summary of Cross-Validated Elastic net Models:")
1946  print(eNet_full_results_df)
1947
1948
1949  ##
       ------------------------------------------------------------------------


1950  # Initialize a list to store predictions
1951  eNet_full_predictions <- list()
1952
1953  # Predict separately for each response variable (store sales)
1954  for (i in 1:ncol(Y_full)) {
1955    store_name <- colnames(Y_full)[i]  # Store name
```

```
1956    eNet_full_predictions[[store_name]] <- predict(eNet_full_
            models[[store_name]], newx = X_full)  # Get predicted
            values
1957  }
1958
1959  # Print structure after storing predictions
1960  print("Updated eNet_predictions Structure:")
1961  print(str(eNet_full_predictions))
1962
1963
1964
1965  ##
        ------------------------------------------------------------------------
1966  # Initialize an empty dataframe for performance results
1967  eNet_full_performance_results <- data.frame(Store = character()
        , RMSE = numeric(), MAE = numeric(), MAPE = numeric())
1968
1969  # Compute metrics for each store
1970  for (i in 1:ncol(Y_full)) {
1971    store_name <- colnames(Y_full)[i]  # Store name
1972
1973    if (!is.null(eNet_full_predictions[[store_name]])) {  #
          Ensure predictions exist
1974      actual_values <- Y_full[, i]  # Actual sales values
1975      predicted_values <- eNet_full_predictions[[store_name]][,
            1]  # Extract the first lambda prediction
1976
1977      # Compute RMSE, MAE, MAPE
1978      rmse_value <- rmse(actual_values, predicted_values)
1979      mae_value <- mae(actual_values, predicted_values)
```

```
1980      mape_value <- mean(abs((actual_values - predicted_values) /
              actual_values), na.rm = TRUE) * 100
1981
1982      # Store results
1983      eNet_full_performance_results <- rbind(eNet_full_
              performance_results,
1984                                             data.frame(Store = store_name,
                                                     RMSE = rmse_value, MAE =
                                                    mae_value, MAPE = mape_
                                                    value))
1985    } else {
1986      print(paste("Warning: No predictions found for", store_name
              ))
1987    }
1988  }
1989
1990  # Print final performance metrics
1991  print("Performance Metrics for full Elastic Net VAR Model:")
1992  print(eNet_full_performance_results)
1993
1994
1995  ##
        ----------------------------------------------------------------------


1996  # Compute the average of RMSE, MAE, and MAPE
1997  average_eNet_full_metrics <- colMeans(eNet_full_performance_
        results[,-1], na.rm = TRUE)
1998
1999  # Print the average metrics
2000  print("Average Elastic Net Performance Metrics:")
2001  print(average_eNet_full_metrics)
2002
```

```
2003
2004 ##
     ------------------------------------------------------------------------
2005 # Extract all predictor names from X (including lagged
     endogenous variables and covariates)
2006 all_predictors <- colnames(X_full)  # Assuming X contains all
     predictors used in the model
2007
2008 # Create an empty data frame for the feature selection table
2009 eNet_full_feature_table <- data.frame(Predictor = all_
     predictors)  # Now includes both lagged Y and covariates
2010
2011
2012
2013 ##
     ------------------------------------------------------------------------
2014 # Initialize an empty list to store selected features
2015 eNet_full_selected_features <- list()
2016
2017 # Loop through each Lasso model and extract non-zero
     coefficients
2018 for (response_var in names(eNet_full_models)) {
2019   # Get the fitted Lasso model
2020   model <- eNet_full_models[[response_var]]
2021
2022   # Extract coefficients at the best lambda (lambda.min)
2023   coeffs <- coef(model, s = "lambda.min")
2024
2025   # Convert to a data frame
2026   coeffs_df <- as.data.frame(as.matrix(coeffs))
```

```
2027
2028    # Get names of non-zero coefficients (excluding intercept)
2029    nonzero_features <- rownames(coeffs_df)[coeffs_df$`s1` != 0]
2030    nonzero_features <- nonzero_features[nonzero_features != "(
            Intercept)"]  # Remove intercept
2031
2032    # Store results
2033    eNet_full_selected_features[[response_var]] <- nonzero_
            features
2034  }
2035
2036  # Add selection results for each endogenous variable
2037  for (response_var in names(eNet_full_selected_features)) {
2038    eNet_full_feature_table[[response_var]] <- ifelse(eNet_full_
            feature_table$Predictor %in% eNet_full_selected_features[[
            response_var]], "Yes", "No")
2039  }
2040
2041  # Print the final feature selection table
2042  print(eNet_full_feature_table)
2043
2044
2045
2046  ##
        ------------------------------------------------------------------
2047  # Combine all selected features across response variables
2048  eNet_full_selected_all <- unlist(eNet_full_selected_features)
        # Flatten list of selected features
2049
2050  # Count occurrences of each predictor
2051  eNet_full_feature_counts <- table(eNet_full_selected_all)
```

```
2052
2053 # Convert to a data frame
2054 eNet_full_feature_summary <- as.data.frame(eNet_full_feature_
       counts)
2055 colnames(eNet_full_feature_summary) <- c("Predictor", "
       Frequency")
2056
2057 # Sort predictors by frequency of selection (descending order)
2058 eNet_full_feature_summary <- eNet_full_feature_summary[order(-
       eNet_full_feature_summary$Frequency), ]
2059
2060 # Display only the top 10 or 20 predictors
2061 eNet_full_top_features <- head(eNet_full_feature_summary, 10)
       # Adjust the number as needed
2062 print(eNet_full_top_features)
2063
2064
2065
2066 ##
       ------------------------------------------------------------------------
2067 #Forecasting
2068
2069
2070 # Define the number of test observations (e.g., last 10 weeks)
2071 test_size <- 10
2072
2073 # Training set (all rows except the last 'test_size' rows)
2074 X_train <- X_full[1:(nrow(X_full) - test_size), ]
2075 Y_train <- Y_full[1:(nrow(Y_full) - test_size), ]
2076
2077
```

```r
# Test set (last 'test_size' rows)
X_test <- X_full[(nrow(X_full) - test_size + 1):nrow(X_full), ]
Y_test <- Y_full[(nrow(Y_full) - test_size + 1):nrow(Y_full), ]



##
    ------------------------------------------------------------------------


# Initialize lists to store trained models
ridge_train_models <- list()
lasso_train_models <- list()


# Train separate models for each store's sales
for (i in 1:ncol(Y_train)) {
  store_name <- colnames(Y_train)[i]  # Store name

  # Ridge Regression (alpha = 0)
  ridge_train_models[[store_name]] <- cv.glmnet(X_train, Y_
      train[, i], alpha = 0)

  # Lasso Regression (alpha = 1)
  lasso_train_models[[store_name]] <- cv.glmnet(X_train, Y_
      train[, i], alpha = 1)

}
```

```
2104 ##
     ------------------------------------------------------------------------

2105 # Define a sequence of alpha values to try (from pure Ridge to
     pure Lasso)
2106 alpha_values <- seq(0, 1, by = 0.1)
2107
2108 # Initialize a list to store the best models for each response
     variable
2109 eNet_train_models <- list()
2110 best_alpha_train <- list()
2111 best_lambda_train <- list()
2112
2113 # Fit Elastic Net separately for each column in Y
2114 for (i in 1:ncol(Y_train)) {
2115   response_var_train <- colnames(Y_train)[i]  # Store variable
       name
2116   cv_results_train <- list()  # Store cross-validation results
       for different alphas
2117
2118   # Loop over different alpha values
2119   for (alpha in alpha_values) {
2120     cv_model_train <- cv.glmnet(X_train, Y_train[, i], alpha =
         alpha)  # Cross-validated Elastic Net
2121     cv_results_train[[as.character(alpha)]] <- cv_model_train
2122   }
2123
2124   # Find the best alpha (minimizing cross-validation error)
2125   min_mse_train <- sapply(cv_results_train, function(model) min
       (model$cvm))  # Get min CV error for each alpha
2126   best_alpha_index <- which.min(min_mse_train)
```

```
2127    best_alpha_train[[response_var_train]] <- alpha_values[best_
            alpha_index]
2128
2129    # Store the best model (corresponding to the best alpha)
2130    eNet_train_models[[response_var_train]] <- cv_results_train[[
            as.character(best_alpha_train[[response_var_train]])]]
2131    best_lambda_train[[response_var_train]] <- eNet_train_models
            [[response_var_train]]$lambda.min  # Store optimal lambda
2132  }
2133
2134
2135
2136  ##
       ------------------------------------------------------------------------

2137  # Initialize lists to store predictions
2138  ridge_train_predictions <- list()
2139  lasso_train_predictions <- list()
2140  eNet_train_predictions <- list()
2141
2142  # Predict separately for each response variable (store sales)
2143  for (i in 1:ncol(Y_test)) {
2144    store_name <- colnames(Y_test)[i]  # Store name
2145
2146    # Generate forecasts
2147    ridge_train_predictions[[store_name]] <- as.numeric(predict(
            ridge_train_models[[store_name]],newx = X_test, s = "
            lambda.1se"))
2148    lasso_train_predictions[[store_name]] <- as.numeric(predict(
            lasso_train_models[[store_name]],newx = X_test, s = "
            lambda.1se"))
```

```
2149    eNet_train_predictions[[store_name]] <-as.numeric(predict(
            eNet_train_models[[store_name]],newx = X_test, s = "lambda
            .1se"))
2150  }
2151
2152
2153
2154  ##
        ------------------------------------------------------------------

2155  # Function to compute RMSE, MAE, and MAPE
2156  compute_metrics <- function(actual, predicted) {
2157    rmse <- sqrt(mean((actual - predicted)^2))
2158    mae <- mean(abs(actual - predicted))
2159    mape <- mean(abs((actual - predicted) / actual)) * 100
2160    return(c(RMSE = rmse, MAE = mae, MAPE = mape))
2161  }
2162
2163  # Function to store metrics for each model
2164  get_metrics_for_model <- function(predictions, Y_test) {
2165    metrics_list <- list()  # Initialize an empty list to store
           the results
2166    for (i in 1:ncol(Y_test)) {
2167      store_name <- colnames(Y_test)[i]
2168      metrics <- compute_metrics(Y_test[, i], predictions[[store_
             name]])
2169      metrics_list[[i]] <- c(Store = store_name, metrics)
2170    }
2171
2172    # Combine the list of metrics into a data frame
2173    metrics_df <- do.call(rbind, metrics_list)
```

```
2174    metrics_df <- as.data.frame(metrics_df, stringsAsFactors =
           FALSE)
2175
2176    # Convert numeric columns to proper types
2177    metrics_df$RMSE <- as.numeric(metrics_df$RMSE)
2178    metrics_df$MAE <- as.numeric(metrics_df$MAE)
2179    metrics_df$MAPE <- as.numeric(metrics_df$MAPE)
2180
2181    return(metrics_df)
2182  }
2183
2184  # Store metrics for each model (with store names)
2185  ridge_train_metrics <- get_metrics_for_model(ridge_train_
       predictions, Y_test)
2186  lasso_train_metrics <- get_metrics_for_model(lasso_train_
       predictions, Y_test)
2187  eNet_train_metrics <- get_metrics_for_model(eNet_train_
       predictions, Y_test)
2188
2189
2190  ##
       ----------------------------------------------------------------------
2191  # Print results
2192  print("Ridge VAR Forecasting Metrics:")
2193  print(ridge_train_metrics)
2194
2195
2196  ##
       ----------------------------------------------------------------------
2197  # Compute the average of RMSE, MAE, and MAPE
```

```r
average_ridge_train_metrics <- colMeans(ridge_train_metrics[,-1], na.rm = TRUE)

# Print the average metrics
print("Average ridge_train_metrics:")
print(average_ridge_train_metrics)


##------------------------------------------------------------------------------

print("Lasso VAR Forecasting Metrics:")
print(lasso_train_metrics)


##------------------------------------------------------------------------------

# Compute the average of RMSE, MAE, and MAPE
average_lasso_train_metrics <- colMeans(lasso_train_metrics[,-1], na.rm = TRUE)

# Print the average metrics
print("Average lasso_train_metrics:")
print(average_lasso_train_metrics)


##------------------------------------------------------------------------------

print("Elastic Net VAR Forecasting Metrics:")
print(eNet_train_metrics)
```

```
2222
2223
2224 ##
     ----------------------------------------------------------------------

2225 # Compute the average of RMSE, MAE, and MAPE
2226 average_eNet_train_metrics <- colMeans(eNet_train_metrics[,-1],
     na.rm = TRUE)
2227
2228 # Print the average metrics
2229 print("Average eNet_train_metrics:")
2230 print(average_eNet_train_metrics)
2231
2232
2233 ##
     ----------------------------------------------------------------------

2234 # Choose a store to visualize
2235 store_to_plot <- colnames(Y_test)[1]  # First store in dataset
2236
2237 # Create a data frame for plotting
2238 forecast_df <- data.frame(
2239   Week = (1:test_size),
2240   Actual = Y_test[, store_to_plot],
2241   Ridge = ridge_train_predictions[[store_to_plot]],
2242   Lasso = lasso_train_predictions[[store_to_plot]],
2243   ElasticNet = eNet_train_predictions[[store_to_plot]]
2244 )
2245
2246 # Plot actual vs. predicted values
2247 ggplot(forecast_df, aes(x = Week)) +
2248   geom_line(aes(y = Actual, color = "Actual"), size = 1) +
```

```
2249   geom_line(aes(y = Ridge, color = "Ridge"), size = 1, linetype
           = "dashed") +
2250   geom_line(aes(y = Lasso, color = "Lasso"), size = 1, linetype
           = "dotted") +
2251   geom_line(aes(y = ElasticNet, color = "Elastic Net"), size =
          1, linetype = "dotdash") +
2252   labs(title = paste("Actual vs. Predicted Sales -", store_to_
          plot),
2253        y = "Sales", x = "Week") +
2254   scale_color_manual(values = c("Actual" = "black", "Ridge" = "
          blue",
2255                                 "Lasso" = "red", "Elastic Net"
                                           = "green")) +
2256   theme_minimal() +
2257   theme(text = element_text(family = "Times New Roman"))
2258
2259
2260
2261 ##
       ------------------------------------------------------------------------

2262 # Choose the store to plot
2263 store_to_plot <- colnames(Y_test)[1]  # First store in dataset
2264
2265 # Create a data frame for plotting
2266 forecast_df <- data.frame(
2267   Week = 1:test_size,
2268   Actual = Y_test[, store_to_plot],
2269   Ridge = ridge_train_predictions[[store_to_plot]],
2270   Lasso = lasso_train_predictions[[store_to_plot]],
2271   ElasticNet = eNet_train_predictions[[store_to_plot]]
2272 )
```

```
2273
2274 # Convert to long format for faceting
2275 forecast_long <- forecast_df %>%
2276   pivot_longer(cols = -c(Week, Actual), names_to = "Model",
         values_to = "Forecast") %>%
2277   pivot_longer(cols = c(Actual, Forecast), names_to = "Type",
         values_to = "Sales")
2278
2279 # Generate the plot
2280 ggplot(forecast_long, aes(x = Week, y = Sales, color = Type,
      linetype = Type)) +
2281   geom_line(size = 1) +
2282   facet_wrap(~Model, scales = "free_y") +  # Separate panel for
        each model
2283   scale_color_manual(values = c("Actual" = "blue", "Forecast" =
        "red")) +
2284   scale_linetype_manual(values = c("Actual" = "solid", "
       Forecast" = "dashed")) +
2285   labs(title = paste("Out-of-Sample Forecast Comparison for",
       store_to_plot),
2286        x = "Time Step", y = "Weekly Sales") +
2287   theme_minimal() +
2288   theme(text = element_text(family = "Times New Roman"))
2289
2290
2291 ##
      ------------------------------------------------------------------------
2292 knitr::purl("Rcode(2).Rmd")
```

Listing D.1: Full R script for regularized VAR forecasting