MINISTRY OF EDUCATION AND TRAINING

Ho Chi Minh City University of Economics and Finance

Faculty of Information Technology

---------o0o---------

Time Series Analysis

Title: Outstanding Home Loan Balance Analytics

Lecturer supervisor

MSC. Ngo Thuan Du

Group 8 Members:

Full name	Student ID	Task Completion
Nguyễn Minh Nhật	**225210135**	**100%**
Đoàn Nguyên Trí	**225210468**	**100%**
Trần Minh Quân	**225210187**	**100%**
Nguyễn Hồng Phúc	**225053466**	**100%**
Lê Trương Minh Quân	**225210212**	**100%**

Ho Chi Minh City, Vietnam

October, 2025

Table of Contents

List of Figures

Acknowledgments

I want to thank MSC. Ngo Thuan Du for his dedicated and thoughtful guidance to our group during this subject. His support was instrumental in helping us expand our knowledge and, crucially, identify and correct deficiencies in our learning and teamwork. This experience was invaluable in gaining the necessary skills and confidence to deliver a successful final presentation and report.

Your instruction extended beyond the scope of the course, providing us with invaluable lessons in professionalism, critical thinking, and accountability,essential skills for our future careers. We express our deepest gratitude to him for his dedication and the wealth of knowledge he has imparted.

At the same time, we would like to thank the Board of Directors of the University of Economics and Finance Ho Chi Minh City. Ho Chi Minh City (UEF) and the Board of Directors of the Faculty of Information Technology have created all favorable conditions in terms of facilities and academic environment for us to complete this course and report in the best way.

Abstract

- Accurate forecasting of outstanding home loan balances plays a vital role in financial planning, credit risk management, and policy formulation. This study proposes a hybrid forecasting framework that integrates the Seasonal Autoregressive Integrated Moving Average (SARIMA) model with the nonlinear XGBoost algorithm to enhance prediction accuracy. Using monthly data on housing loan balances in the United States from 2017 to June 2025, the research first applies the SARIMA model to capture the linear trend and seasonal patterns inherent in the financial time series. The residuals obtained from SARIMA are then modeled by XGBoost to account for nonlinear relationships that traditional statistical methods fail to represent effectively.
- The proposed SARIMA–XGBoost hybrid model demonstrates superior forecasting performance compared to standalone SARIMA and XGBoost models, as evaluated by standard metrics including MAE, RMSE, and MAPE. Empirical results indicate that while SARIMA effectively models long-term seasonality, the nonlinear XGBoost component substantially reduces residual errors, leading to a more accurate representation of short-term fluctuations. However, the hybrid model still exhibits certain inconsistencies when compared with the actual observed data, particularly in capturing sudden directional shifts. These deviations suggest that while the model effectively learns the general seasonal and trend components, it occasionally misrepresents short-term fluctuations that may result from complex nonlinear or exogenous economic factors not fully accounted for in the training process.
- In conclusion, the SARIMA model successfully forecasts the outstanding home loan balance from January 2025 to June 2026, revealing a stable upward trend consistent with post-pandemic housing demand recovery.

Chapter 1

INTRODUCTION

1.1 Background and Problem Statement

General background

In recent years, forecasting outstanding home loan balances has become increasingly important for banks and policymakers, as it reflects credit expansion, consumer confidence, and financial stability. Accurate predictions help financial institutions manage credit risks, optimize liquidity, and support macroeconomic planning. Traditional models such as ARIMA and SARIMA are effective in capturing trend and seasonality but often assume linearity. However, economic and financial data are inherently nonlinear and influenced by complex factors such as interest rates, income levels, and housing demand. This complexity highlights the need for more adaptive and hybrid forecasting approaches.

Specific problem

In spite of the growing use of advanced methods, limited research has focused on integrating SARIMA and XGBoost to forecast outstanding home loan balances. Existing studies typically rely on either linear time series models, which miss nonlinear dynamics, or machine learning models, which overlook temporal dependencies. This research addresses that gap by developing a hybrid SARIMA–XGBoost framework that combines linear and nonlinear modeling strengths. The goal is to enhance predictive accuracy, reduce residual

errors, and produce reliable forecasts that better represent real-world housing finance trends.

This research evaluates the combination of classical time series and modern machine learning techniques to forecast outstanding home loan balances, aiming to identify an accurate and practical approach for financial forecasting. The topic was chosen due to the increasing importance of predicting credit dynamics amid post-pandemic economic recovery and rising housing market volatility. By integrating SARIMA's capability to model linear and seasonal patterns with XGBoost's strength in capturing nonlinear relationships, this study seeks to enhance forecast reliability and reduce modeling errors. The findings are expected to support data-driven decision-making in banking operations, credit risk assessment, and monetary policy formulation, thereby contributing to financial stability and sustainable economic development.

1.2 Research Objectives

In response to the increasing complexity of financial forecasting, this study aims to develop a hybrid SARIMA–XGBoost framework to improve the prediction of outstanding home loan balances. The model combines the strengths of statistical and machine learning approaches to capture both linear seasonal trends and nonlinear fluctuations in financial data.

The specific objectives of this research are as follows:

- **To identify** the key economic and financial factors influencing the movement of outstanding home loan balances in the context of post-pandemic economic recovery and housing market volatility.

- **To assess** the impact of integrating SARIMA and XGBoost on the forecasting performance compared with standalone models.

- **To analyze** the relationship between linear (trend and seasonality) components modeled by SARIMA and nonlinear residual patterns captured by XGBoost using empirical data.

- **To compare** the forecasting accuracy and error metrics among SARIMA, XGBoost, and the proposed hybrid SARIMA–XGBoost framework.

- **To propose** recommendations for improving financial forecasting practices and credit risk management through the application of hybrid time series modeling techniques.

1.3 Research Questions

The following research questions guide the evaluation of classical and modern machine learning models for forecasting of outstanding home loan balance, ensuring they are direct, concise, and answerable through data collection and analysis:

1. What are the main economic and financial factors influencing the outstanding home loan balance over time?
2. How effectively does the SARIMA model capture the linear and seasonal components of the home loan balance data?
3. To what extent does the XGBoost model improve forecasting accuracy when combined with SARIMA?
4. How does the hybrid SARIMA–XGBoost framework compare to individual models in terms of MAE, RMSE, and MAPE performance?

5. What practical insights or recommendations can be derived from the hybrid model to enhance financial forecasting and credit risk management?

1.4 Scope and Delimitations

Scope of the study:

• Time frame: From January 2017 to June 2025.

• Location: Unknown.

• Subjects: Outstanding home loan balances and key macroeconomic indicators

• Programming Language: R.

• Editing Language: English.

Delimitations:

• Does not consider home loan forecasting for other countries or regions.

• Excludes the influence of unexpected macroeconomic shocks or policy changes outside the study period.

1.5 Research Gap

Although several studies have applied traditional time series models such as ARIMA and SARIMA for economic and financial forecasting, these approaches often struggle to handle irregular variations and complex residual patterns in real-world data. Meanwhile, modern machine learning methods like XGBoost have shown strong predictive performance but lack explicit mechanisms for modeling seasonality and temporal dependence.

There has been no prior study applying a SARIMA–XGBoost hybrid model to forecast outstanding home loan balances, particularly in the context of the United States housing finance market after the COVID-19 pandemic. Most previous research has focused on broader macroeconomic indicators rather than sector-specific credit data.

This study fills that gap by combining SARIMA's capacity to capture seasonal and temporal patterns with XGBoost's strength in refining residual forecasts, aiming to enhance overall prediction accuracy and provide more reliable insights for financial decision-making.

1.6 Overview of Methodology

This study employs a quantitative research approach that integrates classical time series modeling with modern machine learning techniques to forecast outstanding home loan balances. Data analysis will utilize methods such as Support Vector Machine Imputation (SVMI), Predictive Imputation (PI), and Median Imputation to handle missing values and ensure data consistency.

After preprocessing, the SARIMA model is applied to identify seasonal and temporal patterns in the historical data. The residuals from the SARIMA model are then modeled using the XGBoost algorithm, which captures complex nonlinear relationships and improves predictive performance. The accuracy of each model and the hybrid framework is evaluated using standard metrics, including MAE, RMSE, and MAPE, to determine the most effective forecasting approach.

Chapter 2

REPORT CONTENT

This chapter presents the dataset and the methodological framework for evaluating both classical time series models and modern machine learning models in forecasting loan outstanding balances (measured in billions). It describes the characteristics of the dataset including its structure and limitations and provides a detailed explanation of the analytical methods used to assess forecasting accuracy and practical applicability. The results of model comparisons and their implications for financial risk management and credit portfolio planning are presented, forming a basis for data-driven recommendations to address financial stability challenges in the banking sector and in economic forecasting.

2.1 Data review

Outstanding_Balance_Billions
<dbl>
488.15
501.66
475.30
497.93
NA
568.52
550.47
510.29
536.97
533.11
NA
527.22
493.42
509.55
532.45
NA
565.55
552.60

Figure 1: Outstanding Credit Balance

The dataset consists of a single column of historical monthly financial observations, specifically the variable **Outstanding_Balance_Billions**.

Starting from **January 2017** and containing a total of **102 observations**, the dataset represents a key financial metric measured in **billions**, sourced from reliable economic records. It exhibits both **seasonal fluctuations** and **long-term trends in credit outstanding**, which are highly relevant for **financial risk management** and **strategic capital allocation**.

```
sum(is.na(DL))
```

```
13
```

```
percentage_na = (sum(is.na(DL)) / length(DL)) * 100
percentage_na
```

```
13.5416666666667
```

Figure 2: Outstanding Credit Balance missing value

Due to the presence of 13 missing values (NA) in the dataset, a preprocessing step was necessary. We conducted a comparison of three imputation methods to determine the optimal technique, ensuring accuracy in restoring the missing data.

Split train & test data:

```
train <- window(dl, end = c(2024, 12))
test <- window(dl, start = c(2025, 1))
print(test)
```

```
        Jan    Feb    Mar    Apr    May    Jun
2025 713.16 723.04 742.41 712.63 794.57 811.05
```

Figure 3: Splitting train and test data

```
        Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
2017 488.15 501.66 475.30 497.93     NA 568.52 550.47 510.29 536.97 533.11
2018 493.42 509.55 532.45     NA 565.55 552.60 545.45 533.59 579.56 551.68
2019 540.21     NA 557.49 564.92 595.51 578.22 558.14 541.98 576.30 563.91
2020 561.02 550.66 582.08 603.61 617.22 584.54     NA 582.62 580.28 643.44
2021 582.09 613.31     NA 593.50 616.64 613.60 653.26 648.77 613.34 639.88
2022 644.12 601.43 615.54 635.51 654.13     NA 691.10 656.60 637.28 682.03
2023 632.61 643.68 679.35 669.49 674.95 709.38 669.10     NA 673.27 683.81
2024 689.12 670.63     NA 717.14 747.37 721.54 751.26 723.81     NA 748.64
        Nov    Dec
2017     NA 527.22
2018 559.55 585.33
2019 606.32     NA
2020 645.24 648.23
2021     NA 637.65
2022 655.87     NA
2023 715.89 684.49
2024 739.20 753.31
```

Figure 4: The train data containing 13 missing values

2.2 Handling Missing Value

The dataset includes 96 monthly observations of outstanding balance (in billions), with 13 missing values (account for 13.5% of the total dataset) due to reporting gaps. To ensure data continuity for forecasting, three imputation methods are applied:

- Support Vector Machine Imputation (SVMI)
- Median Imputation (MEDIAN)
- Polynomial Interpolation (PI)

These methods are implemented using imputeTS, e1071, and pracma packages. Each imputed series is evaluated for accuracy and impact on downstream SARIMA and machine learning models. The best-performing method is selected based on RMSE, MAE, and preservation of trend and seasonality.

This step ensures reliable data preparation for accurate financial forecasting and risk management.

2.2.1 Support Vector Machine Imputation (SVMI)

Support Vector Machine Imputation (SVMI) is a machine learning–based approach that estimates missing values using the principles of **Support Vector Regression (SVR)**. The method constructs an optimal regression function that predicts target values from observed data while minimizing the prediction error. Unlike simple imputation methods, SVMI can model complex and nonlinear relationships among variables, making it highly suitable for financial and economic datasets.

In this study, SVMI is used to estimate missing monthly values in the dataset containing outstanding home loan balances and macroeconomic indicators. Each missing observation is treated as a dependent variable, while the other available variables act as predictors. The regression function in SVMI can be expressed as:

$$\hat{y} = \omega^T \phi(x) + b$$

where $\phi(x)$ represents the nonlinear transformation of input features, ω is the weight vector, and b is the bias term. The model minimizes the loss function within an ε-insensitive margin, ensuring that small errors are ignored to enhance robustness.

Reason for selection:

SVMI is selected because the dataset contains multiple correlated economic indicators, where missing values are influenced by nonlinear relationships rather than random noise. This approach allows the imputation process to learn from the structure of the data instead of relying on simple statistical assumptions.

Advantages:

- Captures nonlinear and multivariate relationships among variables.
- Reduces bias compared to univariate or constant-value imputation methods.
- Generates realistic, data-driven estimates.

Limitations:

- Requires sufficient data for training and parameter tuning.
- More computationally intensive than traditional imputation techniques.
- Model performance depends on kernel and hyperparameter selection.

Result:

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug
2017	488.1500	501.6600	475.3000	497.9300	508.7949	568.5200	550.4700	510.2900
2018	493.4200	509.5500	532.4500	541.6384	565.5500	552.6000	545.4500	533.5900
2019	540.2100	558.2210	557.4900	564.9200	595.5100	578.2200	558.1400	541.9800
2020	561.0200	550.6600	582.0800	603.6100	617.2200	584.5400	592.0745	582.6200
2021	582.0900	613.3100	613.9507	593.5000	616.6400	613.6000	653.2600	648.7700
2022	644.1200	601.4300	615.5400	635.5100	654.1300	646.1464	691.1000	656.6000
2023	632.6100	643.6800	679.3500	669.4900	674.9500	709.3800	669.1000	677.3441
2024	689.1200	670.6300	705.3909	717.1400	747.3700	721.5400	751.2600	723.8100

	Sep	Oct	Nov	Dec
2017	536.9700	533.1100	529.0904	527.2200
2018	579.5600	551.6800	559.5500	585.3300
2019	576.3000	563.9100	606.3200	575.2109
2020	580.2800	643.4400	645.2400	648.2300
2021	613.3400	639.8800	633.1576	637.6500
2022	637.2800	682.0300	655.8700	656.9003
2023	673.2700	683.8100	715.8900	684.4900
2024	736.1916	748.6400	739.2000	753.3100

Figure 5: Performance of Missing Data Imputation Methods SVMI

2.2.2 Median Imputation (MEDIAN)

Median Imputation (MI) is a statistical technique used to replace missing data points with the **median** of the available observations for that variable. The median, unlike the mean, represents the middle value of a sorted dataset and is therefore less affected by extreme values or outliers. This makes MI particularly useful in financial and economic datasets, where irregular or volatile values may distort the data distribution.

In this study, MI is applied to fill missing monthly values in the outstanding home loan balance and related economic indicators. For each variable with missing observations, the median value of all available data points is calculated and substituted into the missing positions. The estimation process can be expressed as:

$$\hat{y}_t = median(y_1, y_2, \cdots, y_n)$$

where $\hat{y}_t$ is the imputed value at time *t, and* $y_1, y_2, \cdots, y_n$ are the observed values of that variable

Reason for selection:
Median Imputation is chosen because it provides a quick, stable, and distribution-robust method for completing the dataset. It helps reduce bias introduced by extreme financial fluctuations and ensures a consistent structure for subsequent modeling with SARIMA and XGBoost.

Advantages:

- Robust against outliers and skewed data distributions.
- Simple to implement and computationally efficient.
- Maintains the overall central tendency of the variable.

Limitations:

- Ignores temporal or nonlinear relationships between observations.
- May oversimplify data variation, especially in time series with strong trends or seasonality.

Result:

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct
2017	488.15	501.66	475.30	497.93	613.31	568.52	550.47	510.29	536.97	533.11
2018	493.42	509.55	532.45	613.31	565.55	552.60	545.45	533.59	579.56	551.68
2019	540.21	613.31	557.49	564.92	595.51	578.22	558.14	541.98	576.30	563.91
2020	561.02	550.66	582.08	603.61	617.22	584.54	613.31	582.62	580.28	643.44
2021	582.09	613.31	613.31	593.50	616.64	613.60	653.26	648.77	613.34	639.88
2022	644.12	601.43	615.54	635.51	654.13	613.31	691.10	656.60	637.28	682.03
2023	632.61	643.68	679.35	669.49	674.95	709.38	669.10	613.31	673.27	683.81
2024	689.12	670.63	613.31	717.14	747.37	721.54	751.26	723.81	613.31	748.64

	Nov	Dec
2017	613.31	527.22
2018	559.55	585.33
2019	606.32	613.31
2020	645.24	648.23
2021	613.31	637.65
2022	655.87	613.31
2023	715.89	684.49
2024	739.20	753.31

Figure 6: Performance table of missing data imputation methods MEDIAN

2.2.3 Polynomial Interpolation (PI)

Polynomial Interpolation (PI) is a mathematical approach used to estimate missing values by constructing a polynomial function that passes through existing data points. Unlike linear interpolation, which assumes a constant rate of change between two adjacent points, PI allows for curvature, enabling the model to capture smoother and more realistic variations over time. This makes it particularly suitable for financial or economic datasets, where trends often evolve gradually rather than linearly.

In this study, PI is applied to estimate missing monthly observations in the outstanding home loan balance series. The method fits a polynomial of degree n using nearby observed data, represented as:

$$\hat{y}(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n$$

where $a_0, a_1, \ldots, a_n$ are coefficients derived from the surrounding known points, and t denotes the time index of the missing value**.** Once the polynomial is constructed, the missing value is estimated by substituting its corresponding $\boldsymbol{t}$ into $\hat{y}(t)$

Reason for selection:
PI is chosen because the outstanding home loan balance data exhibits gradual seasonal and cyclical movements. Using a polynomial function allows the imputation to preserve these smooth patterns, reducing the risk of abrupt or unrealistic changes that simpler methods like median or linear interpolation might cause.

Advantages:

- Captures nonlinear trends and cyclical variations in the time series.
- Produces smooth and continuous estimates that align with financial data behavior.

Limitations:

- Highly sensitive to outliers and the number of points used—high-degree polynomials can lead to overfitting.
- May generate oscillations in regions with sparse data.

Result:

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug
2017	488.1500	501.6600	475.3000	497.9300	524.6111	568.5200	550.4700	510.2900
2018	493.4200	509.5500	532.4500	540.5990	565.5500	552.6000	545.4500	533.5900
2019	540.2100	561.5028	557.4900	564.9200	595.5100	578.2200	558.1400	541.9800
2020	561.0200	550.6600	582.0800	603.6100	617.2200	584.5400	589.6170	582.6200
2021	582.0900	613.3100	594.8566	593.5000	616.6400	613.6000	653.2600	648.7700
2022	644.1200	601.4300	615.5400	635.5100	654.1300	660.5903	691.1000	656.6000
2023	632.6100	643.6800	679.3500	669.4900	674.9500	709.3800	669.1000	685.2359
2024	689.1200	670.6300	696.7265	717.1400	747.3700	721.5400	751.2600	723.8100

	Sep	Oct	Nov	Dec
2017	536.9700	533.1100	517.3225	527.2200
2018	579.5600	551.6800	559.5500	585.3300
2019	576.3000	563.9100	606.3200	573.6309
2020	580.2800	643.4400	645.2400	648.2300
2021	613.3400	639.8800	633.0556	637.6500
2022	637.2800	682.0300	655.8700	650.3810
2023	673.2700	683.8100	715.8900	684.4900
2024	735.9685	748.6400	739.2000	753.3100

Figure 7: Performance table of PI missing data calculation methods

2.2.4 Accuracy

To rigorously evaluate the three imputation methods — Median Imputation (MEDIAN), Support Vector Machine Imputation (SVMI), and Polynomial Interpolation (PI) — a cross-validation framework with simulated missingness is applied. Specifically, 10% of observed data points are randomly masked to create artificial gaps, mimicking real-world reporting inconsistencies. Each method is then used to impute these gaps, and performance is assessed using three key metrics:

- Root Mean Squared Error (RMSE): Measures absolute error magnitude, sensitive to large deviations.
- Mean Absolute Error (MAE): Provides average error in original units, robust to outliers.
- Mean Absolute Percentage Error (MAPE): Expresses error as a percentage of true values, enabling scale-independent comparison.

Method	MAE	RMSE	MAPE (%)
MEDIAN	**9,54**	**29.67**	**1.55**
SVMI	**3.22**	**11.23**	**0.51**
PI	**3.37**	**11.74**	**0.52**

Figure 8: Comparison table of indexes of methods

SVMI achieves the best performance across all metrics, with MAE = 3.22, RMSE = 11.23, and MAPE = 0.51% outperforming PI by ~5% and MEDIAN by over 67% in MAPE. This confirms its superior ability to capture nonlinear dynamics and local dependencies using kernel-based modeling.

PI performs closely behind SVMI, with MAE = 3.37, RMSE = 11.74, and MAPE = 0.52%, demonstrating excellent trend preservation and smoothness. It is highly suitable for financial series with gradual growth patterns.

MEDIAN significantly underperforms, with MAE = 9.54, RMSE = 29.67, and MAPE = 1.55% — 3 times worse than SVMI in MAPE. It fails to capture trend and volatility, flattening the series and introducing significant bias, especially during credit expansion phases.

Among the three tested imputation methods, **SVMI** achieved the lowest overall error metrics, demonstrating the highest accuracy in reconstructing missing values. With MAE = 3.22, RMSE = 11.23, and MAPE = 0.51%, SVMI outperformed PI by approximately 5% and Median Imputation by more than 67% in percentage error. Its strong performance highlights the advantage of capturing nonlinear dependencies and local relationships within financial time series data through kernel-based regression.

PI followed closely behind, with MAE = 3.37, RMSE = 11.74, and MAPE = 0.52%. Despite slightly higher errors, PI showed excellent trend preservation and smoothness, making it a robust choice for imputing missing values in gradually changing datasets such as outstanding home loan balances.

In contrast, **Median Imputation** yielded significantly higher error values (MAE = 9.54, RMSE = 29.67, and MAPE = 1.55%), indicating limited ability to capture the underlying trend and volatility of the data. While it remains simple and stable, its assumption of constant central tendency causes flattened patterns and bias during periods of rapid growth or decline.

Based on this quantitative comparison, **SVMI and PI** were selected for subsequent forecasting experiments due to their superior balance between accuracy, temporal consistency, and computational efficiency. These methods provide reliable foundations for modeling and predicting outstanding home loan balances in the hybrid SARIMA–XGBoost framework.

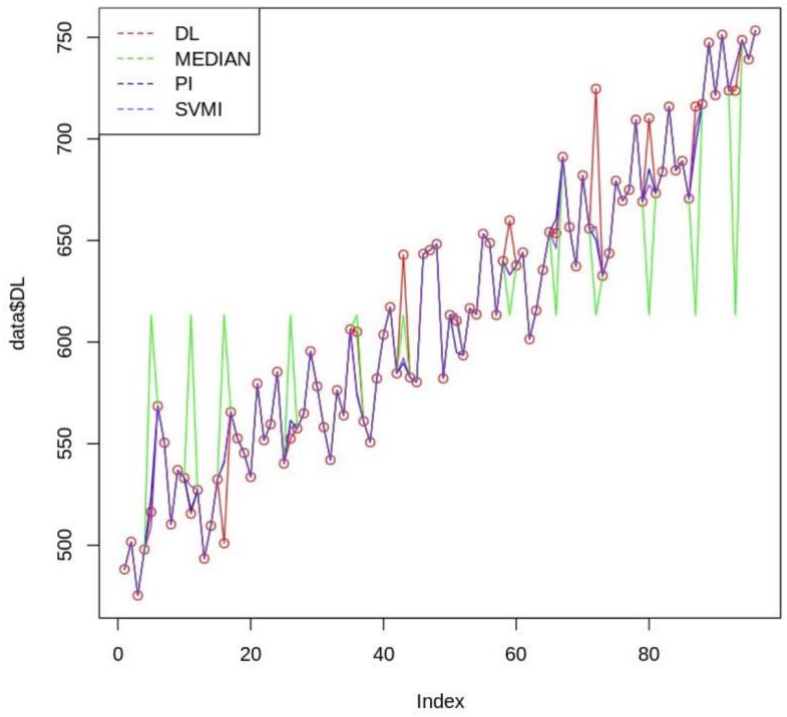

Figure 9: Comparison of Imputed Outstanding Balance Series Using MEDIAN, SVMI, and PI Methods

2.3 Building the SMVI model

2.3.1 Seasonally Data

Stationary

A **stationary time series** is one whose statistical properties—such as mean, variance, and autocovariance—remain constant over time. In other words, the series does not exhibit long-term trends or changing variability, and its behavior is consistent regardless of the time period observed. Stationarity is a fundamental requirement for most time series forecasting models, including ARIMA and SARIMA, as these models assume that the underlying process generating the data is stable. When a series is non-stationary, it often shows trends, seasonality, or structural shifts that must be removed or transformed—typically through differencing or detrending—before reliable forecasting can be performed.

The stationarity of the home loan balance time series was examined using two complementary approaches: the Augmented Dickey–Fuller (ADF) and Kwiatkowski–Phillips–Schmidt–Shin (KPSS) tests.

```
        Augmented Dickey-Fuller Test

data:  DL_SVMI
Dickey-Fuller = -3.5744, Lag order = 4, p-value = 0.03942
alternative hypothesis: stationary

Warning message in tseries::kpss.test(DL_SVMI):
“p-value smaller than printed p-value”

        KPSS Test for Level Stationarity

data:  DL_SVMI
KPSS Level = 2.4048, Truncation lag parameter = 3, p-value = 0.01
```

Figure 10: Check Stationary of SVMI

The null hypothesis (H_0) of the Augmented Dickey–Fuller (ADF) test states that the time series contains a unit root, indicating non-stationarity, while the alternative hypothesis (H_1) assumes stationarity. The obtained **p-value of 0.03942**, which is lower than the 0.05 significance level, leads to the rejection of the null hypothesis. This result indicates that the outstanding home loan balance series is **stationary** according to the ADF test.

For the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test, the hypotheses are reversed. The null hypothesis (H_0) assumes that the series is stationary, whereas the alternative hypothesis (H_1) indicates non-stationarity. The **p-value of 0.01**, which is below 0.05, results in the rejection of the null hypothesis, suggesting that the series is **non-stationary** under the KPSS test.

The two tests therefore provide **contradictory outcomes**. This suggests that the series may be **trend-stationary**, where non-stationarity arises from a deterministic trend. To ensure robustness in subsequent modeling, differencing should be performed before SARIMA model identification.

The First Difference

```
        Augmented Dickey-Fuller Test

data:  D1_SVMI
Dickey-Fuller = -7.8409, Lag order = 4, p-value = 0.01
alternative hypothesis: stationary

Warning message in tseries::kpss.test(D1_SVMI):
“p-value greater than printed p-value”

        KPSS Test for Level Stationarity

data:  D1_SVMI
KPSS Level = 0.027234, Truncation lag parameter = 3, p-value = 0.1
```

Figure 11: Check Stationary of the first Difference (SVMI method)

	Jan	Feb	Mar	Apr	May	Jun
2017		13.5100000	-26.3600000	22.6300000	10.8649465	59.7250535
2018	-33.8000000	16.1300000	22.9000000	9.1883690	23.9116310	-12.9500000
2019	-45.1200000	18.0109602	-0.7309602	7.4300000	30.5900000	-17.2900000
2020	-14.1909323	-10.3600000	31.4200000	21.5300000	13.6100000	-32.6800000
2021	-66.1400000	31.2200000	0.6406656	-20.4506656	23.1400000	-3.0400000
2022	6.4700000	-42.6900000	14.1100000	19.9700000	18.6200000	-7.9836405
2023	-24.2903032	11.0700000	35.6700000	-9.8600000	5.4600000	34.4300000
2024	4.6300000	-18.4900000	34.7608705	11.7491295	30.2300000	-25.8300000
	Jul	Aug	Sep	Oct	Nov	Dec
2017	-18.0500000	-40.1800000	26.6800000	-3.8600000	-4.0195815	-1.8704185
2018	-7.1500000	-11.8600000	45.9700000	-27.8800000	7.8700000	25.7800000
2019	-20.0800000	-16.1600000	34.3200000	-12.3900000	42.4100000	-31.1090677
2020	7.5345443	-9.4545443	-2.3400000	63.1600000	1.8000000	2.9900000
2021	39.6600000	-4.4900000	-35.4300000	26.5400000	-6.7223538	4.4923538
2022	44.9536405	-34.5000000	-19.3200000	44.7500000	-26.1600000	1.0303032
2023	-40.2800000	8.2440966	-4.0740966	10.5400000	32.0800000	-31.4000000
2024	29.7200000	-27.4500000	12.3816302	12.4483698	-9.4400000	14.1100000

Figure 12: Data after the first Difference (SVMI method)

After applying first-order differencing, the stationarity of the outstanding home loan balance series was re-evaluated using the ADF and KPSS tests. The **ADF test** yielded a **p-value of 0.01**, which is below the 0.05 significance level, leading to the rejection of the null hypothesis of non-stationarity. This indicates that the differenced series is **stationary** according to the ADF test.

On the other hand, the **KPSS test** produced a **p-value of 0.1**, which is greater than 0.05, resulting in the failure to reject the null hypothesis of stationarity. This outcome also supports that the differenced series is **stationary** under the KPSS framework.

For those reasons, the data is stopped after applying the first difference (D = 1).

Autocorrelation Function (ACF) D1

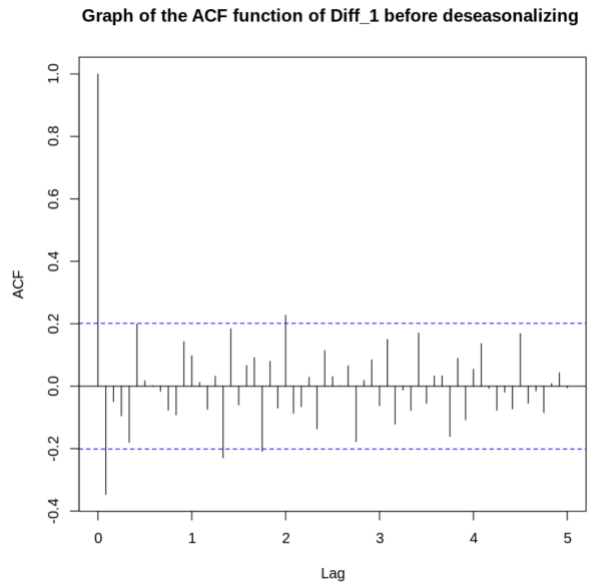

Figure 13: ACF of D1 (SVMI method)

The Autocorrelation Function (ACF) measures the correlation between a time series and its own lagged values over successive time intervals. It provides insight into how past observations influence current values, helping to identify potential autoregressive (AR) and moving average (MA) components in time series models. In stationary data, autocorrelations typically decline quickly toward zero as the lag increases, whereas a slow decay indicates non-stationarity or the presence of seasonality.

The ACF plot of the first-order differenced outstanding home loan balance series (before deseasonalizing) shows a sharp spike at **lag 1** and **lag 2**, which is higher than the significant level (abs 0,2) . Besides, choosing lag 2 despite the small gap **higher than 0,2**

because this data has just had the first stationary, it can still affect the building model step. Overall, the candidates for MA in ARIMA are **MA(1)** and **MA(2)**:

$$Q = 1, Q = 2$$

Partial Autocorrelation Function (PACF) D1

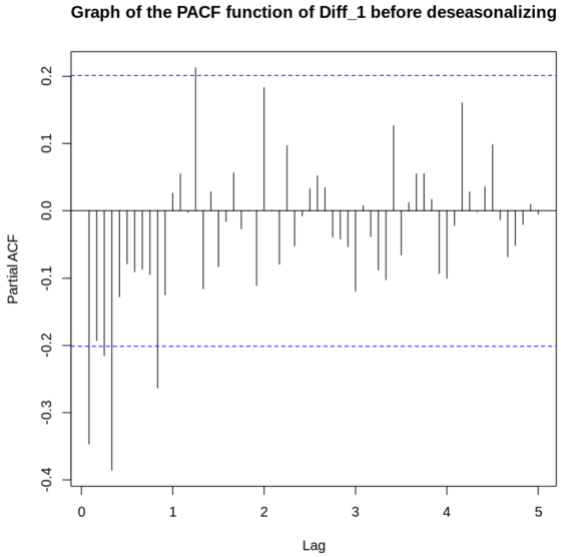

Figure 14: PACF of D1 (SVMI method)

The Partial Autocorrelation Function (PACF) measures the correlation between a time series and its lagged values after removing the effects of intermediate lags. It is particularly useful for identifying the order of the autoregressive (AR) component in ARIMA models. A significant spike at a particular lag in the PACF plot indicates that the corresponding lag contributes meaningfully to explaining the variation in the series.

According to the PACF plot, **lag 1** can be easily chosen based on the condition (lower than -0,2). The outstanding point in this plot is the line between lag 1 and lag 2, which meets the condition. For this reason, **lag 2** is also chosen. At last, the candidates for AR in ARIMA are **AR(1)** and **AR(2)**:

$$P = 1, P = 2$$

Conclusion of Hyperparameters for Seasonal Forecast

Based on the analysis of the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) of the first-order differenced time series, the following optimal parameters were determined for the ARIMA model:

- **P (Autoregressive Order):** The PACF plot shows a significant spike at lag 1 and a weaker effect at lag 2, suggesting that the autoregressive order should include $P \in \{1, 2\}$. This indicates that the current value of the series depends primarily on one or two preceding observations.

- **D (Differencing Order):** A differencing order of **D = 1** was sufficient to achieve stationarity, as confirmed by both the ADF (p-value = 0.01) and KPSS (p-value = 0.1) tests. This implies that only one level of differencing was necessary to remove the underlying trend in the data.

- **Q (Moving Average Order):** The ACF plot exhibits a strong autocorrelation at lag 1 and a smaller but notable correlation at lag 2. Therefore, the moving average component was defined with $Q \in \{1, 2\}$ as candidate orders for evaluation.

Combining these parameters yields the following candidate ARIMA(P, D, Q) models for further evaluation:

ARIMA(1,1,1), ARIMA(1,1,2), ARIMA(2,1,1), ARIMA(2,1,2)

2.4.2 Seasonalized Data

The dataset used in this study consists of monthly outstanding home loan balances, which tend to display recurring seasonal fluctuations influenced by cyclical economic and financial activities. Such seasonal variation can violate the assumption of stationarity required

for ARIMA-type models, potentially distorting parameter estimation and forecast accuracy.

To achieve this, a **deseasonalization** procedure was conducted to extract and visualize the underlying components of the time series. The process employs the *decompose()* function to break down the data into three key elements: **trend**, **seasonal**, and **random (residual)** components.

Result:

A Time Series: 8 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2017	-20.533519	-22.300941	-4.933368	-1.751152	16.509798	4.578566	10.595058	-7.092744	-2.924406	8.809756	12.835997	6.206955
2018	-20.533519	-22.300941	-4.933368	-1.751152	16.509798	4.578566	10.595058	-7.092744	-2.924406	8.809756	12.835997	6.206955
2019	-20.533519	-22.300941	-4.933368	-1.751152	16.509798	4.578566	10.595058	-7.092744	-2.924406	8.809756	12.835997	6.206955
2020	-20.533519	-22.300941	-4.933368	-1.751152	16.509798	4.578566	10.595058	-7.092744	-2.924406	8.809756	12.835997	6.206955
2021	-20.533519	-22.300941	-4.933368	-1.751152	16.509798	4.578566	10.595058	-7.092744	-2.924406	8.809756	12.835997	6.206955
2022	-20.533519	-22.300941	-4.933368	-1.751152	16.509798	4.578566	10.595058	-7.092744	-2.924406	8.809756	12.835997	6.206955
2023	-20.533519	-22.300941	-4.933368	-1.751152	16.509798	4.578566	10.595058	-7.092744	-2.924406	8.809756	12.835997	6.206955
2024	-20.533519	-22.300941	-4.933368	-1.751152	16.509798	4.578566	10.595058	-7.092744	-2.924406	8.809756	12.835997	6.206955

Figure 15: Seasonal Value (SVMI method)

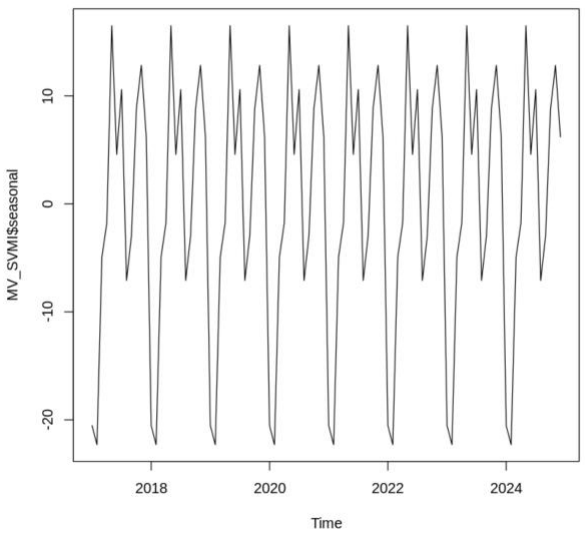

Figure 16: Visualize Seasonal Value (SVMI method)

Once the seasonal component is identified, it is removed from the original series to produce the deseasonalized time series used for further analysis.

Result

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug
2017	508.6835	523.9609	480.2334	499.6812	492.2851	563.9414	539.8749	517.3827
2018	513.9535	531.8509	537.3834	543.3895	549.0402	548.0214	534.8549	540.6827
2019	560.7435	580.5219	562.4234	566.6712	579.0002	573.6414	547.5449	549.0727
2020	581.5535	572.9609	587.0134	605.3612	600.7102	579.9614	581.4795	589.7127
2021	602.6235	635.6109	618.8840	595.2512	600.1302	609.0214	642.6649	655.8627
2022	664.6535	623.7309	620.4734	637.2612	637.6202	641.5678	680.5049	663.6927
2023	653.1435	665.9809	684.2834	671.2412	658.4402	704.8014	658.5049	684.4368
2024	709.6535	692.9309	710.3242	718.8912	730.8602	716.9614	740.6649	730.9027

	Sep	Oct	Nov	Dec
2017	539.8944	524.3002	516.2544	521.0130
2018	582.4844	542.8702	546.7140	579.1230
2019	579.2244	555.1002	593.4840	569.0040
2020	583.2044	634.6302	632.4040	642.0230
2021	616.2644	631.0702	620.3216	631.4430
2022	640.2044	673.2202	643.0340	650.6933
2023	676.1944	675.0002	703.0540	678.2830
2024	739.1160	739.8302	726.3640	747.1030

Figure 17: Deseasonal data (SVMI method)

Visual Plot:

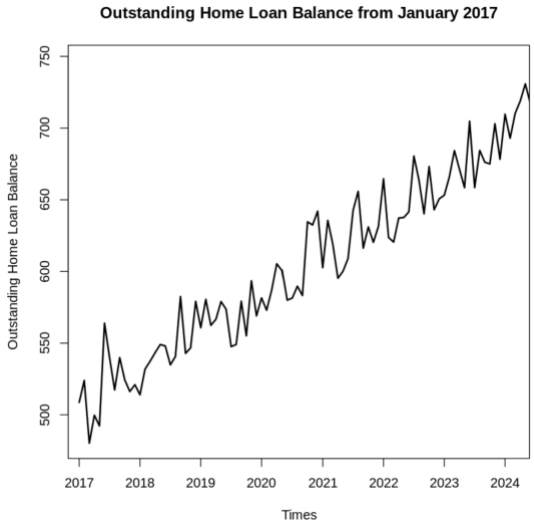

Figure 18: Deseasonalized data plot (SVMImethod)

Stationary

```
        Augmented Dickey-Fuller Test

data:  KhuMua_SVMI
Dickey-Fuller = -3.4303, Lag order = 4, p-value = 0.0541
alternative hypothesis: stationary

Warning message in tseries::kpss.test(KhuMua_SVMI):
“p-value smaller than printed p-value”

        KPSS Test for Level Stationarity

data:  KhuMua_SVMI
KPSS Level = 2.4224, Truncation lag parameter = 3, p-value = 0.01
```

Figure 19: ADF and KPSS Test of Deseasonalized data (SVMI method)

The ADF test produced a p-value of **0.0541**, which is slightly above the 0.05 significance level, indicating a failure to reject the null hypothesis (H_0: non-stationary).

Conversely, the KPSS test returned a p-value of **0.01**, which is below the 0.05 threshold, leading to the rejection of the null hypothesis (H_0: stationary).

These results are not consistent, suggesting that the series **KhuMua_SVMI** is likely **non-stationary** in its level form and requires differencing to achieve stationarity before model estimation.

The First Difference

```
        Augmented Dickey-Fuller Test

data:  d1_SVMI
Dickey-Fuller = -7.2221, Lag order = 4, p-value = 0.01
alternative hypothesis: stationary

Warning message in tseries::kpss.test(d1_SVMI):
“p-value greater than printed p-value”

        KPSS Test for Level Stationarity

data:  d1_SVMI
KPSS Level = 0.036684, Truncation lag parameter = 3, p-value = 0.1
```

Figure 20: Stationary of D\difference 1 (SVMI method)

	Jan	Feb	Mar	Apr	May	Jun
2017		15.2774223	-43.7275726	19.4477836	-7.3960033	71.6562854
2018	-7.0595259	17.8974223	5.5324274	6.0061525	5.6506812	-1.0187681
2019	-18.3795259	19.7783826	-18.0985328	4.2477836	12.3290502	-5.3587681
2020	12.5495418	-8.5925777	14.0524274	18.3477836	-4.6509498	-20.7487681
2021	-39.3995259	32.9874223	-16.7269070	-23.6328820	4.8790502	8.8912319
2022	33.2104741	-40.9225777	-3.2575726	16.7877836	0.3590502	3.9475914
2023	2.4501709	12.8374223	18.3024274	-13.0422164	-12.8009498	46.3612319
2024	31.3704741	-16.7225777	17.3932979	8.5669131	11.9690502	-13.8987681
	Jul	Aug	Sep	Oct	Nov	Dec
2017	-24.0664923	-22.4921979	22.5116618	-15.5941615	-8.0458226	4.7586229
2018	-13.1664923	5.8278021	41.8016618	-39.6141615	3.8437589	32.4090414
2019	-26.0964923	1.5278021	30.1516618	-24.1241615	38.3837589	-24.4800263
2020	1.5180520	8.2332578	-6.5083382	51.4258385	-2.2262411	9.6190414
2021	33.6435077	13.1978021	-39.5983382	14.8058385	-10.7485948	11.1213951
2022	38.9371482	-16.8121979	-23.4883382	33.0158385	-30.1862411	7.6593446
2023	-46.2964923	25.9318986	-8.2424347	-1.1941615	28.0537589	-24.7709586
2024	23.7035077	-9.7621979	8.2132920	0.7142084	-13.4662411	20.7390414

Figure 21: Data after the first difference (SVMI method)

The ADF test on the first-differenced series produced a p-value of **0.01**, which is below the 0.05 significance level, allowing the rejection of the null hypothesis (H_0: non-stationary) and confirming that the differenced series is stationary.

In contrast, the KPSS test yielded a p-value of **0.10**, which is greater than the 0.05 threshold, indicating a failure to reject the null hypothesis (H_0: stationary).

The consistency between both test results confirms that the series achieves stationarity after the first differencing. Therefore, the order of non-seasonal differencing is determined as **d = 1** for the ARIMA model.

ACF d1

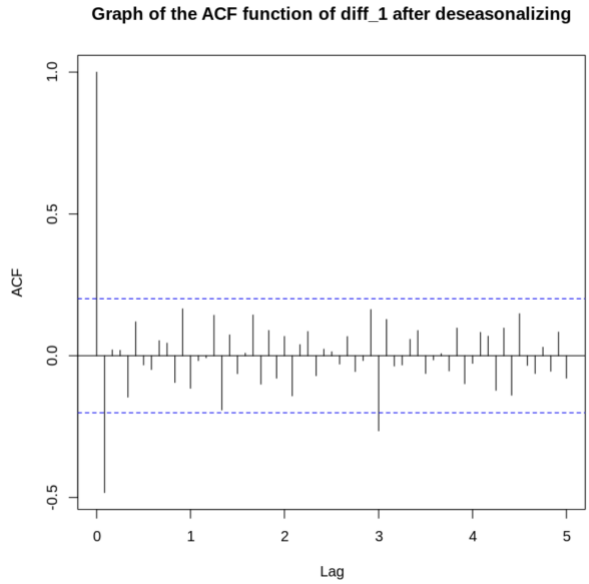

Figure 22: ACF plot of d1 (SVMI method)

The ACF plot of the first difference (after deseasonalizing) demonstrates a sharp spike at **lag 1** and, which is higher than the significant level (0,2) . Besides, lag 3 looks like it reached out of the significant level line.

Therefore, the candidates for MA in ARIMA are **MA(1)** and **MA(3)**:

$$Q = 1, Q = 3$$

PACF d1

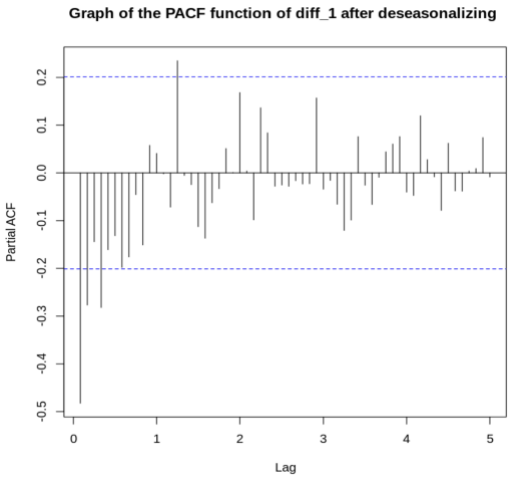

Figure 23: PACF plot of d1 (SVMI method)

According to the PACF plot, **lag 1** can be easily chosen based on the condition (lower than -0,2). The striking point in this plot is the line between lag 1 and lag 2, which is higher than 0,2. For this reason, **lag 2** is also chosen. At last, the candidates for AR in ARIMA are **AR(1)** and **AR(2)**:

$$P = 1, P = 2$$

Conclusion of hyperparameters for seasonal forecast

Based on the analysis of the ACF and PACF plots of the first-differenced and deseasonalized time series, the following parameters for the ARIMA model were identified:

- **P (Autoregressive Order):** The PACF plot displayed significant spikes at lags 1 and 2, indicating the presence of short-term autoregressive dependence. Therefore, the autoregressive order was selected as $p \in \{1, 2\}$ for further evaluation.

- **D (Integrated Order):** The stationarity tests confirmed that the series became stationary after the first differencing. Hence, the integrated order was determined as **d = 1**, implying that the original series required only one differencing to achieve stationarity.

- **Q (Moving Average Order):** The ACF plot exhibited significant autocorrelation at lags 1 and 3, suggesting the presence of a moving average component. Accordingly, the moving average order was defined as $q \in \{1, 3\}$ for model selection and comparison.

Combining these parameters yields the following candidate ARIMA(p, d, q) models for further evaluation:

ARIMA(1,1,1), ARIMA(1,1,3), ARIMA(2,1,1), ARIMA(2,1,3)

2.4.3 Running models

Model Implementation

In the model implementation phase, a total of 16 SARIMA models were constructed and evaluated to identify the optimal configuration for forecasting the deseasonalized outstanding home loan balance series. Each model was parameterized based on different combinations of non-seasonal and seasonal orders — (p, d, q) and (P, D, Q), respectively.

The **non-seasonal parameters** were defined as $\mathbf{p} \in \{1, 2\}$, $\mathbf{d} = 1$, and $\mathbf{q} \in \{1, 3\}$, while the **seasonal parameters** were set as $\mathbf{P} \in \{1, 2\}$, $\mathbf{D} = 1$, and $\mathbf{Q} \in \{1, 2\}$. By systematically combining these parameter values, a total of twelve candidate SARIMA models were developed and compared to determine the most appropriate model structure.

The evaluation process aimed to identify the configuration that best captured both short-term fluctuations and long-term seasonal patterns in the outstanding home loan balance, thereby ensuring reliable forecasting performance for subsequent analysis.

```
#MH9_SVMI  <- Arima(DL_SVMI, order=c(1,1,1), seasonal=list(order=c(2,1,1), period=12), lambda=lambda_choice, include.constant=FALSE)

Error in optim(init[mask], armafn, method = optim.method, hessian = TRUE, : non-finite finite-difference value [5]
Traceback:

1. suppressWarnings(tmp <- stats::arima(x = x, order = order, seasonal = seasonal,
 .     include.mean = include.mean, method = method, ...))
2. withCallingHandlers(expr, warning = function(w) if (inherits(w,
 .     classes)) tryInvokeRestart("muffleWarning"))
3. stats::arima(x = x, order = order, seasonal = seasonal, include.mean = include.mean,
 .     method = method, ...)
4. optim(init[mask], armafn, method = optim.method, hessian = TRUE,
 .     control = optim.control, trans = as.logical(transform.pars))
5. .handleSimpleError(function (cnd)
 . {
 .     watcher$capture_plot_and_output()
 .     cnd <- sanitize_call(cnd)
 .     watcher$push(cnd)
 .     switch(on_error, continue = invokeRestart("eval_continue"),
 .         stop = invokeRestart("eval_stop"), error = NULL)
 . }, "non-finite finite-difference value [5]", base::quote(optim(init[mask],
 .     armafn, method = optim.method, hessian = TRUE, control = optim.control,
 .     trans = as.logical(transform.pars))))
```

Figure 24: non-finite finite-difference value

During the estimation of the SARIMA MH9_SVMI(1,1,1)(2,1,1) model, an optimization error occurred, indicating a "**non-finite finite-difference value**." This error arises when the numerical optimization algorithm used to estimate the model parameters encounters undefined or infinite values in the likelihood function. Such an issue typically results from one or more of the following conditions:

1. **Inappropriate parameter initialization** — the optimization procedure may start with initial values that lead to non-invertible MA components or unstable AR roots, preventing convergence.

2. **Excessive differencing or overparameterization** — applying both first-order non-seasonal differencing (d = 1) and seasonal differencing (D = 1) can overly reduce the information content in the series, producing near-zero variance and making parameter estimation unstable.

3. **Numerical instability in transformed data** — the Box–Cox transformation or deseasonalization step (through the lambda parameter) may yield non-finite or extremely small values, which violate the assumptions of the optimization routine.

4. **Model complexity relative to data length** — the inclusion of multiple autoregressive and moving average terms with limited data points can also cause singularities in the covariance matrix during estimation.

AIC and BIC

	df	AIC
	<dbl>	<dbl>
MH1_SVMI	5	931.6148
MH2_SVMI	7	935.4609
MH3_SVMI	6	933.5295
MH4_SVMI	8	934.6344
MH5_SVMI	6	931.5828
MH6_SVMI	8	933.5612
MH7_SVMI	7	933.5722
MH8_SVMI	9	931.6439
MH10_SVMI	8	935.0945
MH11_SVMI	7	934.9142
MH12_SVMI	9	933.1299
MH14_SVMI	9	934.2887
MH16_SVMI	10	932.6187

Figure 25: AIC value of 16 SARIMA models (SVMI method)

	df	BIC
	<dbl>	<dbl>
MH1_SVMI	5	943.7090
MH2_SVMI	7	952.3928
MH3_SVMI	6	948.0425
MH4_SVMI	8	953.9852
MH5_SVMI	6	946.0959
MH6_SVMI	8	952.9120
MH7_SVMI	7	950.5040
MH8_SVMI	9	953.4135
MH10_SVMI	8	954.4452
MH11_SVMI	7	951.8460
MH12_SVMI	9	954.8995
MH14_SVMI	9	956.0582
MH16_SVMI	10	956.8071

Figure 26: BIC value of 16 SARIMA models (SVMI method)

Model Selection

Based on the **AIC** and **BIC** results obtained from the evaluation of all candidate models, the configurations **MH1_SVMI(1,1,1)(1,1,1)**, **MH5_SVMI(1,1,1)(1,1,2)**, and **MH8_SVMI(2,1,3)(1,1,2)** yielded the lowest information criterion values, demonstrating superior model fit and parsimony compared with other alternatives. Consequently, these three models were selected for further comparison and diagnostic assessment to identify the most suitable SARIMA specification for forecasting the deseasonalized outstanding home loan balance series.

MH1_SVMI(1,1,1)(1,1,1)

```
Box.test(MH1_SVMI$residuals, lag=20,type = "Ljung-Box")

    Box-Ljung test

data:  MH1_SVMI$residuals
X-squared = 16.802, df = 20, p-value = 0.6658
```

Figure 27: Box-Ljung test of MH1_SVMI

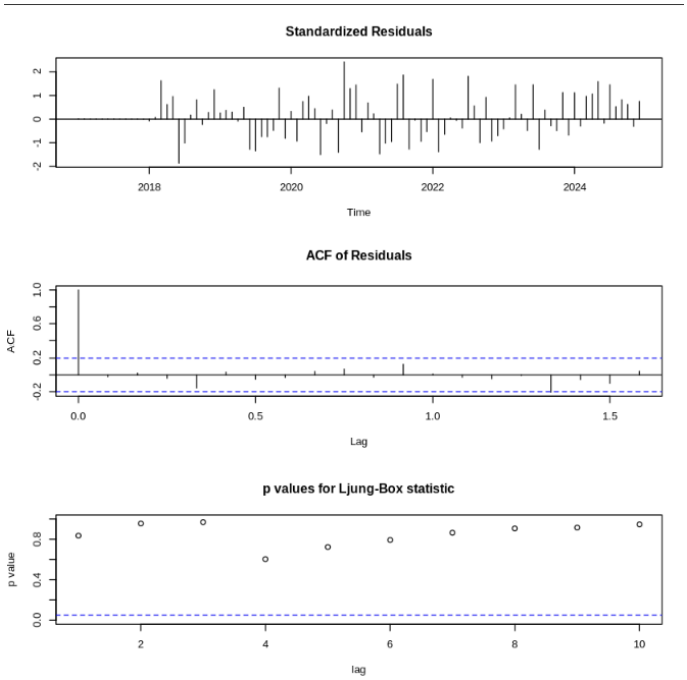

Figure 28: Tsdiag plots of MH1_SVMI

Residual Analysis of Model MH1_SVMI

The standardized residual plot exhibits random fluctuations around zero, indicating the absence of systematic patterns or remaining trend components. The autocorrelation function (ACF) of the residuals shows no significant spikes beyond the confidence bounds, confirming that serial correlation has been effectively removed and that the residuals behave as white noise. The residual distribution approximates normality, as evidenced by the bell-shaped histogram centered near zero. These findings demonstrate that the MH1_SVMI model satisfies the key statistical assumptions of the SARIMA framework and provides an adequate representation of the underlying process.

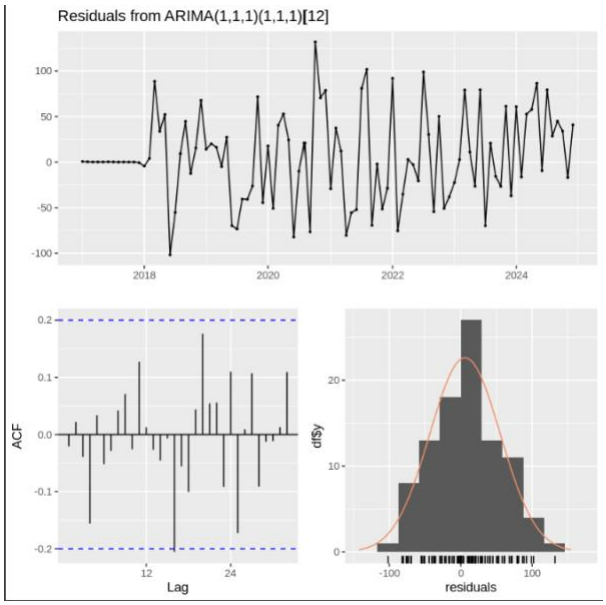

Figure 29: Checking residuals plots of MH1_SVMI

MH5_SVMI(1,1,1)(1,1,2)

```
Box.test(MH5_SVMI$residuals, lag=20,type = "Ljung-Box")

    Box-Ljung test

data:  MH5_SVMI$residuals
X-squared = 20.612, df = 20, p-value = 0.4202
```

Figure 30: Box-Ljung test of MH5_SVMI

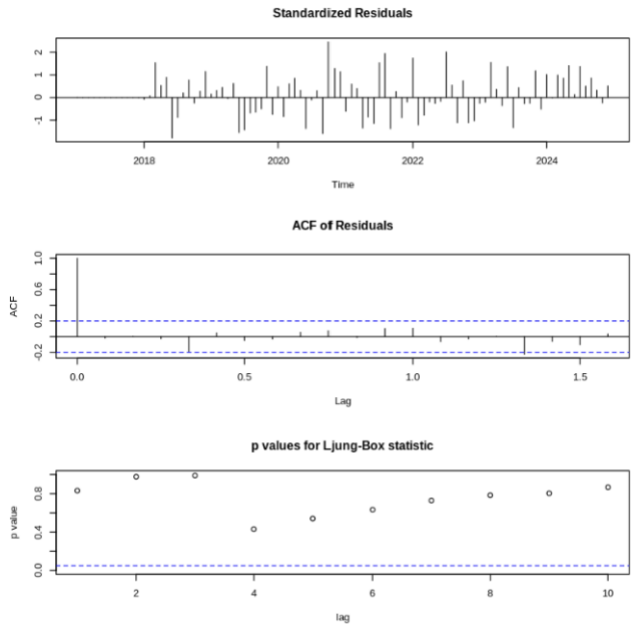

Figure 31: Tsdiag plots of MH5_SVMI

Residual Analysis of Model MH5_SVMI

The residual diagnostics for model MH5 were carefully assessed. The standardized residuals oscillate randomly around zero, showing no apparent trend or recurring seasonal pattern. This suggests that most of the systematic structure in the data has been captured by the model. The ACF plot indicates that the autocorrelation values stay within the 95% confidence range, implying that the residuals are approximately uncorrelated. In addition, the residuals follow a roughly symmetric distribution centered near zero, which supports the assumption of normality. Taken together, these results confirm that model MH5 provides a statistically sound representation of the underlying process.

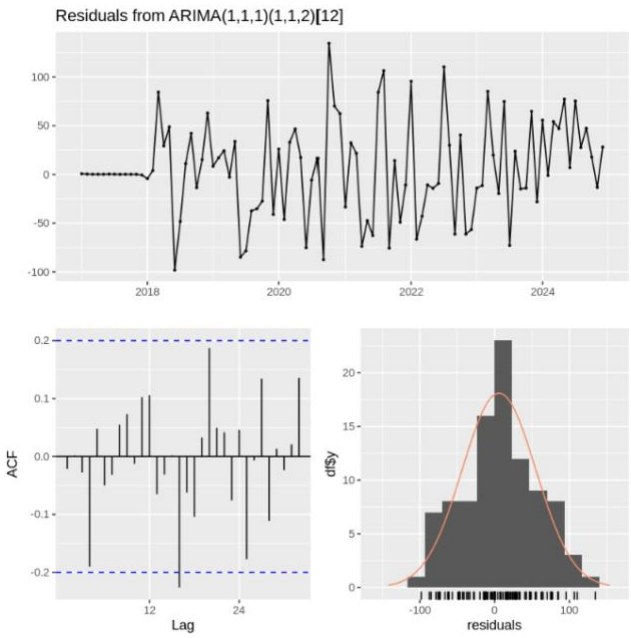

Figure 32: Checking residuals plots of MH5_SVMI

MH8_SVMI(2,1,3)(1,1,2)

```
Box.test(MH8_SVMI$residuals, lag=20,type = "Ljung-Box")

    Box-Ljung test

data:  MH8_SVMI$residuals
X-squared = 19.778, df = 20, p-value = 0.4719
```

Figure 33: Box-Ljung test of MH8_SVMI

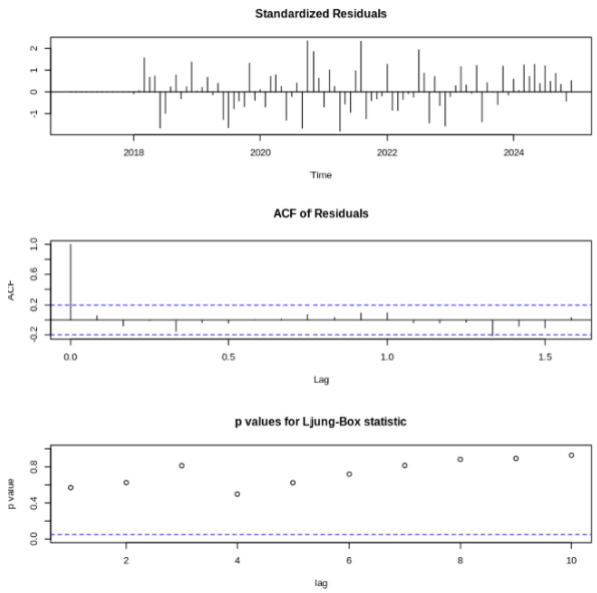

Figure 34: Tsdiag plots of MH8_SVMI

Residual Analysis of Model MH8_SVMI

The diagnostic results for model MH8_SVMI indicate that the standardized residuals fluctuate irregularly around zero, with no clear trend or seasonal pattern remaining. The ACF plot shows that most autocorrelation coefficients lie within the 95% confidence limits, suggesting that the model has largely removed serial dependence. However, the residual histogram displays a noticeable skew, deviating from the expected bell-shaped form. This asymmetry implies a violation of the normality assumption, which weakens the model's reliability for inference and forecasting. Therefore, MH8_SVMI is not considered suitable for final selection.

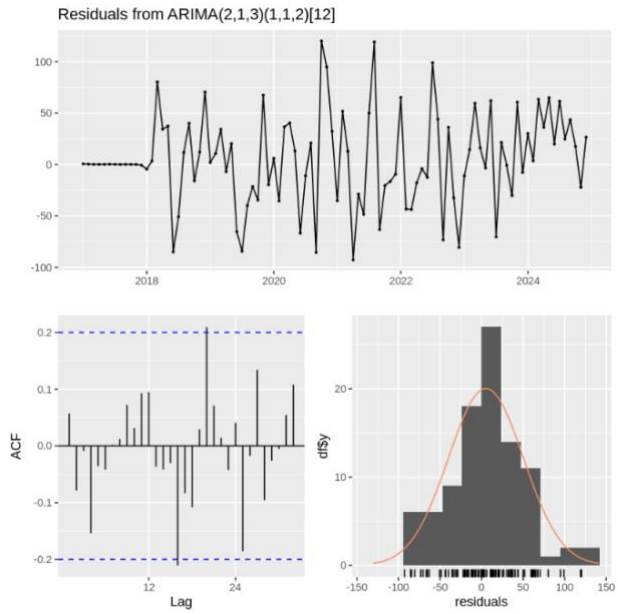

Figure 35: Checking residuals plots of MH8_SVMI

2.4 Building the PI model

Stationarity was examined using the time series completed with Polynomial Interpolation (PI), and the same diagnostic procedure was replicated on the version imputed using Support Vector Machine Imputation (SVMI). Although the imputations differ in method, the underlying temporal characteristics of the series — notably trend and seasonality — remain invariant across datasets. Accordingly, the results of the stationarity diagnostics are treated as uniformly valid for all reconstructed versions of the series (PI, SVMI, and Median).

2.4.1 Seasonality Data

```
print(DL_PI)
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug
2017	488.1500	501.6600	475.3000	497.9300	524.6111	568.5200	550.4700	510.2900
2018	493.4200	509.5500	532.4500	540.5990	565.5500	552.6000	545.4500	533.5900
2019	540.2100	561.5028	557.4900	564.9200	595.5100	578.2200	558.1400	541.9800
2020	561.0200	550.6600	582.0800	603.6100	617.2200	584.5400	589.6170	582.6200
2021	582.0900	613.3100	594.8566	593.5000	616.6400	613.6000	653.2600	648.7700
2022	644.1200	601.4300	615.5400	635.5100	654.1300	660.5903	691.1000	656.6000
2023	632.6100	643.6800	679.3500	669.4900	674.9500	709.3800	669.1000	685.2359
2024	689.1200	670.6300	696.7265	717.1400	747.3700	721.5400	751.2600	723.8100

	Sep	Oct	Nov	Dec
2017	536.9700	533.1100	517.3225	527.2200
2018	579.5600	551.6800	559.5500	585.3300
2019	576.3000	563.9100	606.3200	573.6309
2020	580.2800	643.4400	645.2400	648.2300
2021	613.3400	639.8800	633.0556	637.6500
2022	637.2800	682.0300	655.8700	650.3810
2023	673.2700	683.8100	715.8900	684.4900
2024	735.9685	748.6400	739.2000	753.3100

Figure 36: PI method dataset has seasonality

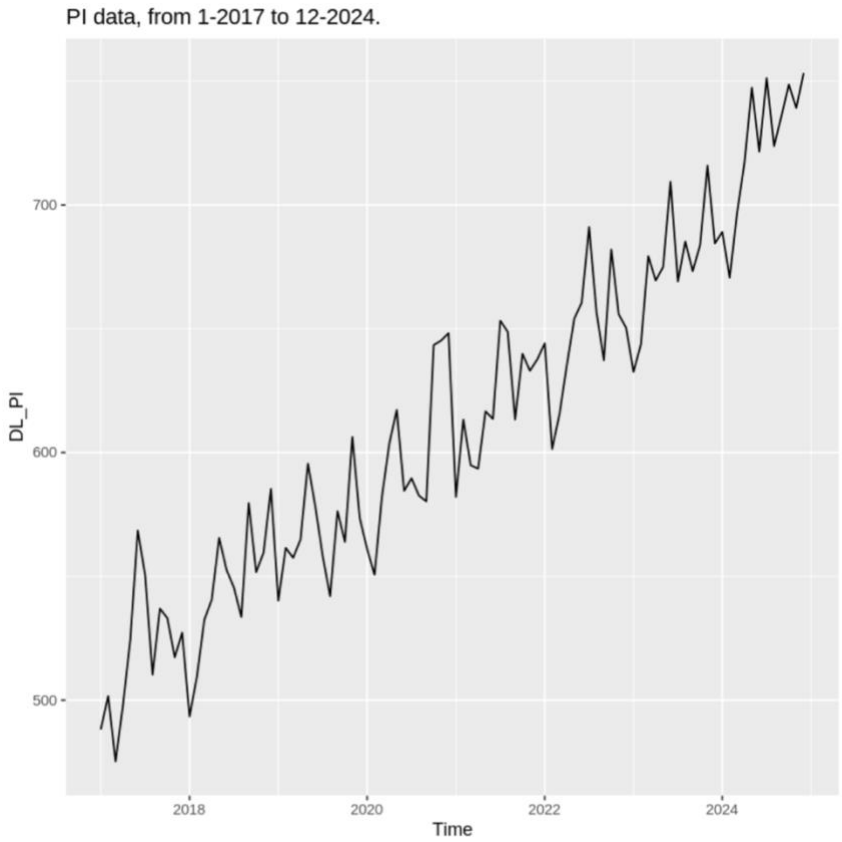

Figure 37: Seasonal Plot of PI method

This chart visualizes the seasonality of the ts_PI data series by plotting the values of each month over the years (from 2017 to 2024). Each line on the chart represents a year.
From the chart, it can be observed that the Trend There is a clear upward trend over time, as shown by the lines of later years (e.g. 2023, 2024) being at a higher level than the lines of earlier years.

```
Warning message in tseries::adf.test(D1_PI):
“p-value smaller than printed p-value”

        Augmented Dickey-Fuller Test

data:  D1_PI
Dickey-Fuller = -7.9092, Lag order = 4, p-value = 0.01
alternative hypothesis: stationary

Warning message in tseries::kpss.test(D1_PI):
“p-value greater than printed p-value”

        KPSS Test for Level Stationarity

data:  D1_PI
KPSS Level = 0.025951, Truncation lag parameter = 3, p-value = 0.1
```

Figure 38: ACF & PACF value is D1 of PI method

ACF D1

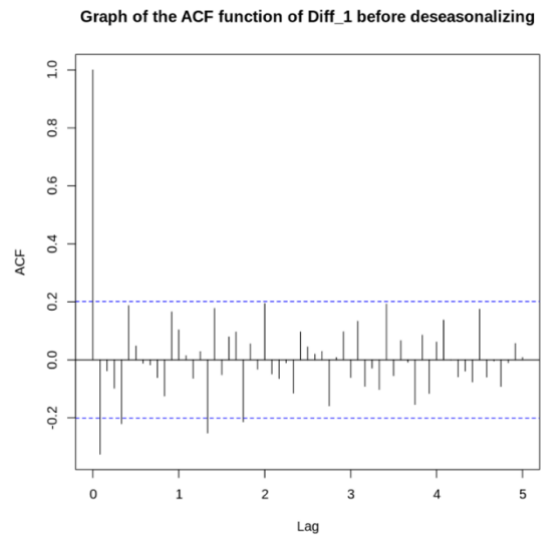

Figure 39: ACF D1 PI method

The ACF plot of the differenced series (D = 1) was analyzed to determine the seasonal Moving Average order (Q). The plot reveals statistically significant spikes that exceed the confidence bounds at Lags 1 and 2, suggesting potential values for the seasonal MA parameter. Accordingly, the candidate orders for the seasonal MA component are:

$$Q \in \{1, 2\}$$

These values serve as the basis for constructing the set of candidate models for further evaluation.

PACF D1

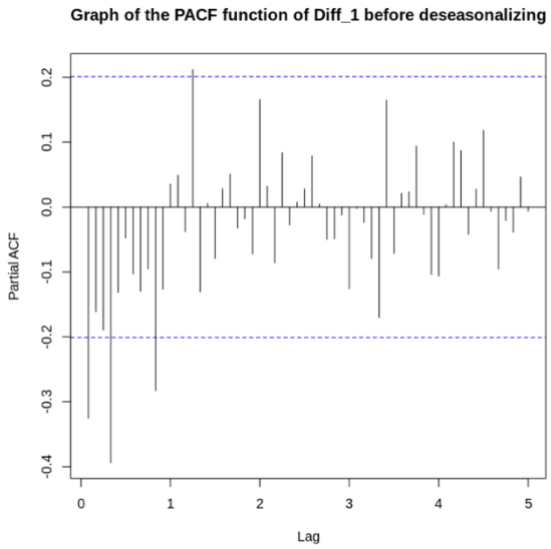

Figure 40: PACF_D1 PI method

The Partial Autocorrelation Function (PACF) plot of the differenced series (D = 1) was analyzed to identify the seasonal autoregressive orders (P). The plot shows statistically significant spikes exceeding the confidence bounds at Lags 1 and 2, suggesting potential seasonal AR orders. Accordingly, the candidate values for the seasonal autoregressive parameter are.

$$P \in \{1, 2\}$$

2.4.2 Deseasonalized Data

```
MV_PI= decompose(DL_PI)
MV_PI$seasonal
plot(MV_PI$seasonal)
```

A Time Series: 8 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2017	-20.119839	-21.418420	-8.483854	-1.483304	16.856087	6.918227	10.366223	-5.843102	-2.750587	9.035148	11.459821	5.463599
2018	-20.119839	-21.418420	-8.483854	-1.483304	16.856087	6.918227	10.366223	-5.843102	-2.750587	9.035148	11.459821	5.463599
2019	-20.119839	-21.418420	-8.483854	-1.483304	16.856087	6.918227	10.366223	-5.843102	-2.750587	9.035148	11.459821	5.463599
2020	-20.119839	-21.418420	-8.483854	-1.483304	16.856087	6.918227	10.366223	-5.843102	-2.750587	9.035148	11.459821	5.463599
2021	-20.119839	-21.418420	-8.483854	-1.483304	16.856087	6.918227	10.366223	-5.843102	-2.750587	9.035148	11.459821	5.463599
2022	-20.119839	-21.418420	-8.483854	-1.483304	16.856087	6.918227	10.366223	-5.843102	-2.750587	9.035148	11.459821	5.463599
2023	-20.119839	-21.418420	-8.483854	-1.483304	16.856087	6.918227	10.366223	-5.843102	-2.750587	9.035148	11.459821	5.463599
2024	-20.119839	-21.418420	-8.483854	-1.483304	16.856087	6.918227	10.366223	-5.843102	-2.750587	9.035148	11.459821	5.463599

Figure 41: The PI method dataset is non_seasonal

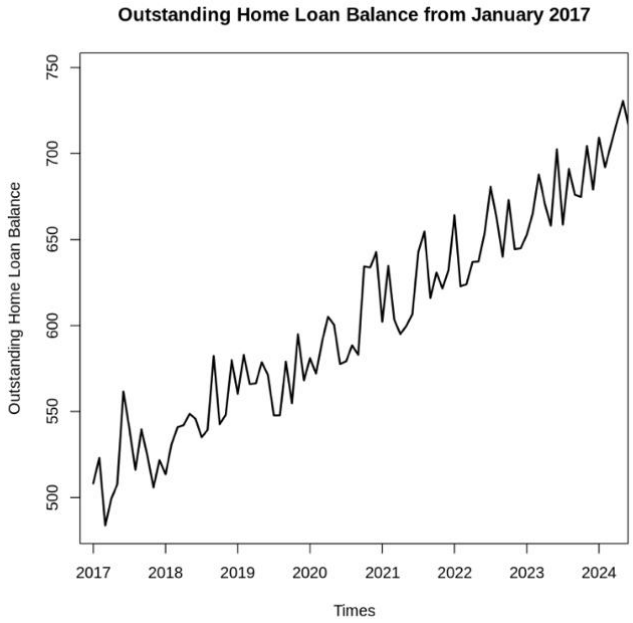

Figure 42: Non-seasonal Plot of PI method

The post-seasonal-adjustment plot represents the **"Seasonally Adjusted"** state of the series, meaning that recurring seasonal fluctuations have been removed. In economic and financial time series, raw data often contain cyclical variations (for example, production typically declines during the Lunar New Year period and increases toward the end of the year). By eliminating these seasonal components, seasonal adjustment reveals the underlying trend of the

series more clearly, thereby providing a more reliable foundation for further analysis and forecasting.

```
Warning message in tseries::adf.test(d1_PI):
“p-value smaller than printed p-value”

        Augmented Dickey-Fuller Test

data:  d1_PI
Dickey-Fuller = -7.4082, Lag order = 4, p-value = 0.01
alternative hypothesis: stationary

Warning message in tseries::kpss.test(d1_PI):
“p-value greater than printed p-value”

        KPSS Test for Level Stationarity

data:  d1_PI
KPSS Level = 0.035825, Truncation lag parameter = 3, p-value = 0.1
```

Figure 43: ACF & PACF value is d1 of PI method

ACF d1

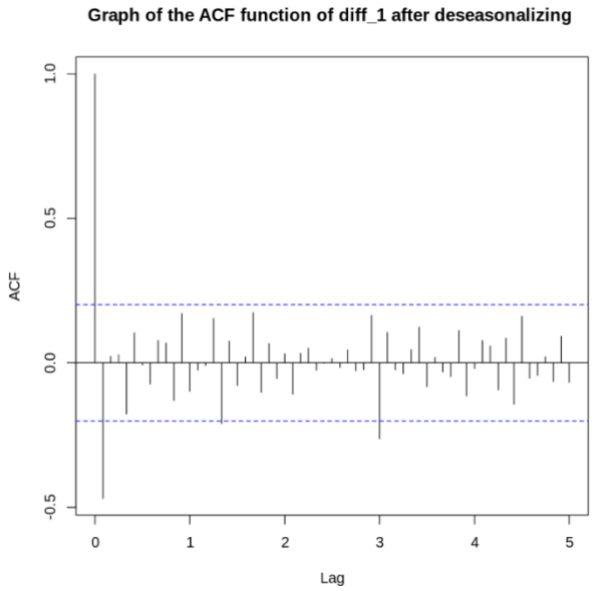

Figure 44: ACF_d1 PI method

The PACF of the deseasonalized series was examined to identify the autoregressive order p for the ARIMA model. The plot exhibits statistically significant spikes at lags 1 and 2, after which the PACF drops sharply and remains within the confidence band — a characteristic “cut-off” pattern. This indicates that the series retains substantial partial autocorrelation with its first and second lagged values once intermediate effects are removed. Accordingly, the candidate autoregressive orders are selected as:

$$p \in \{1, 2\}$$

2.4.3 Run models

Based on the ACF and PACF analysis of the deseasonalized series, the candidate orders for the non-seasonal ARIMA parameters were identified as follows: the PACF exhibited statistically significant spikes at lags 1 and 2, suggesting autoregressive orders $p \in \{1, 2\}$; meanwhile, the ACF showed significant spikes at lags 1 and 3, indicating moving average orders $q \in \{1, 3\}$. Since the series had already been differenced and was assumed to be stationary, the non-seasonal differencing order was fixed at $d = 1$.

On this basis, a total of 15 ARIMA model specifications were constructed and evaluated to identify the optimal configuration for the deseasonalized Production Index (PI) series.

AIC & BIC of 12 SARIMA models

	df	AIC
	<dbl>	<dbl>
MH1_PI	5	827.9442
MH3_PI	6	829.6797
MH4_PI	8	829.1866
MH5_PI	6	828.9322
MH7_PI	7	831.6367
MH8_PI	9	826.5764
MH10_PI	8	833.6412
MH11_PI	7	831.3082
MH12_PI	9	828.3873
MH14_PI	9	833.3678
MH15_PI	8	831.1688
MH16_PI	10	828.3636

Figure 45: AIC values of 12 SARIMA models (PI method)

	df	BIC
	<dbl>	<dbl>
MH1_PI	5	840.0384
MH3_PI	6	844.1927
MH4_PI	8	848.5373
MH5_PI	6	843.4452
MH7_PI	7	848.5686
MH8_PI	9	848.3460
MH10_PI	8	852.9919
MH11_PI	7	848.2401
MH12_PI	9	850.1569
MH14_PI	9	855.1374
MH15_PI	8	850.5195
MH16_PI	10	852.5520

Figure 46: BIC values of 12 SARIMA models (PI method)

Based on the values of AIC,, and BIC reported in the model comparison tables, three SARIMA specifications. **MH1_PI (1,1,1)(1,1,1)[12]**, **MH5_PI (1,1,1)(1,1,2)[12]**, and **MH8_PI (2,1,3)(1,1,2)[12]** were identified as the most competitive candidates. These models achieved some of the lowest information-criterion scores while simultaneously yielding among the smallest RMSE and MAPE values, indicating a favorable trade-off between model fit and structural complexity. Accordingly, they were selected for subsequent residual diagnostics and performance comparison in order to determine the most suitable model for forecasting.

MH1_PI (1,1,1)(1,1,1)[12]

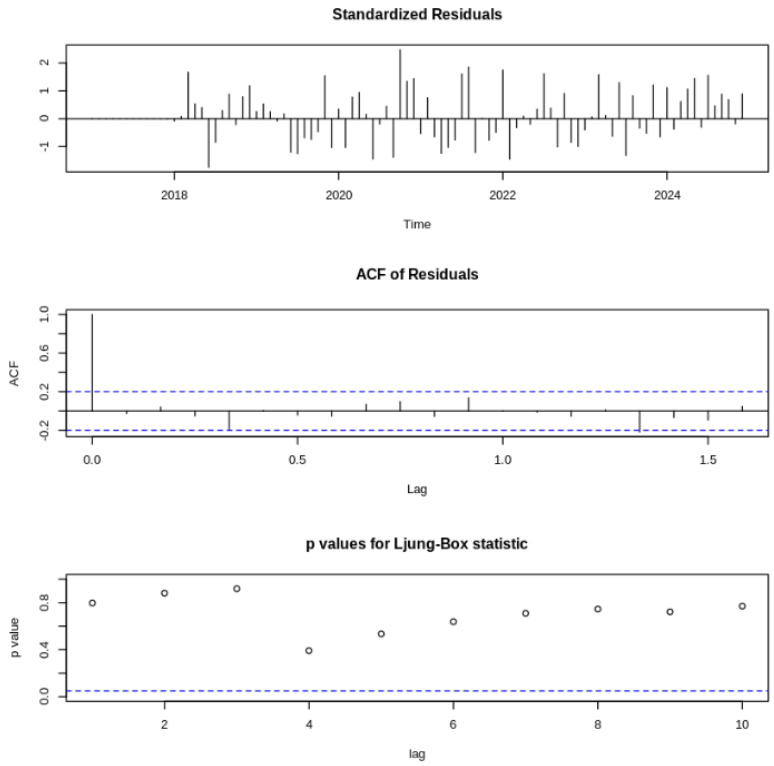

Figure 47: Ljung-Box Test Results for MH1_PI Model

To assess the validity of the **MH1_PI model (SARIMA(1,1,1)(1,1,1)[12]),** the Ljung–Box test was conducted to evaluate whether the model residuals behave as white noise. Specifically, the null hypothesis *H_0* states that the residuals are independently distributed with no statistically significant autocorrelation, whereas the alternative hypothesis *Ha* assumes the presence of residual autocorrelation. The test results indicate a Ljung–Box statistic of ***Q = 16.316*** with 15 degrees of freedom and a corresponding ***p-value of 0.3614***. Since the p-value exceeds the conventional significance threshold (0.05), the null hypothesis cannot be rejected. This implies that the residuals do not exhibit meaningful autocorrelation and can be regarded as white noise. Accordingly, the MH1_PI model passes the post estimation diagnostic check and is deemed a statistically valid specification for modeling and forecasting purposes.

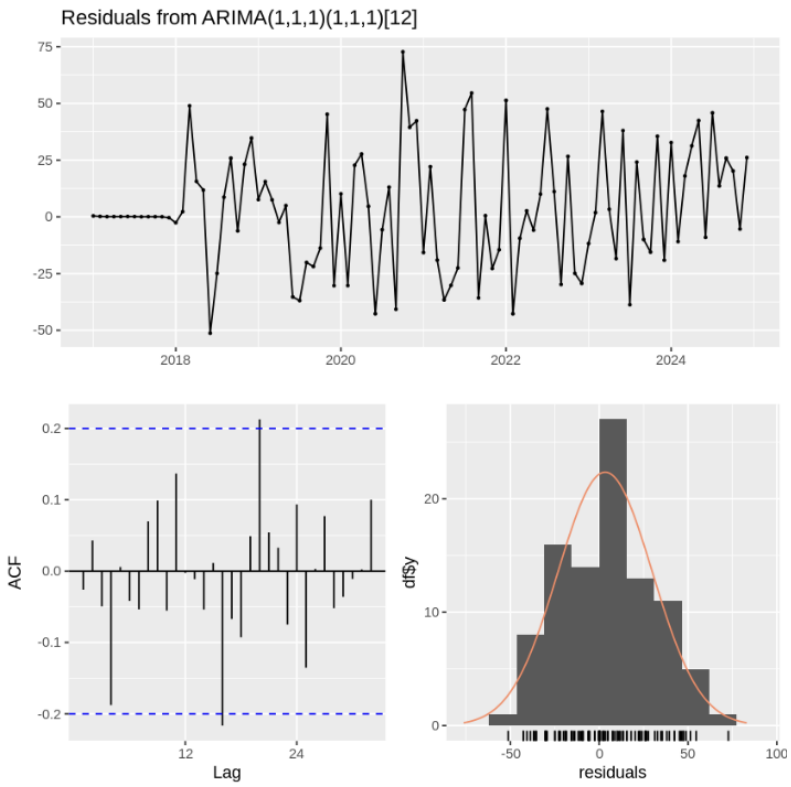

Figure 48: Diagnostic check of residuals of model MH1_PI

MH5_PI (1,1,1)(1,1,2)[12]

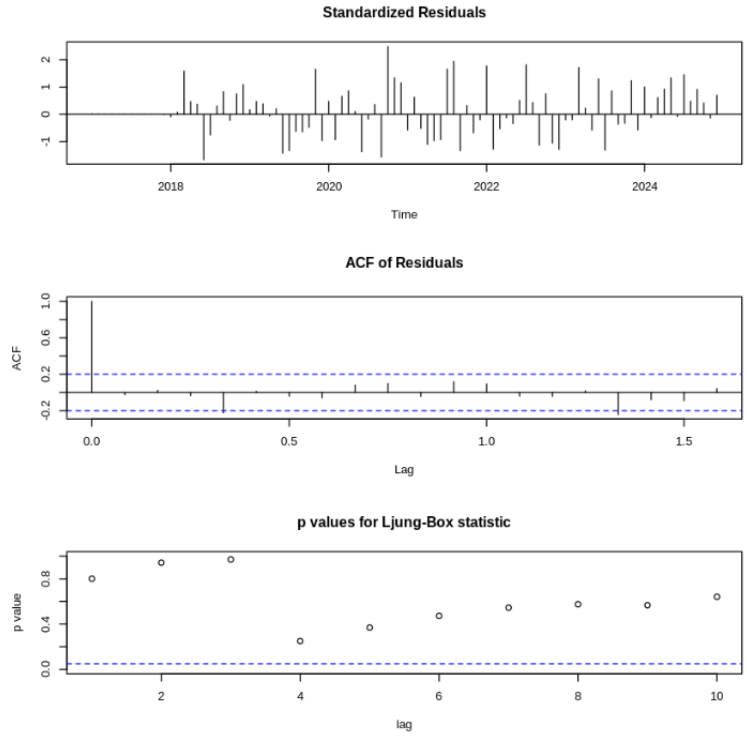

Figure 49: Ljung-Box Test Results for MH5_PI Model

To evaluate the adequacy of the **MH5_PI model (SARIMA(1,1,1)(1,1,2)[12])**, a residual diagnostics procedure was conducted. Visually, the residual ACF plot (see attached figure) reveals no statistically significant spikes beyond the confidence bounds, providing preliminary evidence that the residuals behave randomly. To formally validate this, the Ljung–Box test was applied and returned a test statistic of ***Q = 19.647*** with 14 degrees of freedom (computed from the 19 lags used minus 5 model parameters). Critically, the associated ***p-value of 0.1417*** exceeds the standard significance level $a = 0.05$, therefore the null hypothesis H0, that the residuals constitute white noise. Cannot be rejected. Both graphical and statistical evidence jointly confirm that the residuals of MH5_PI are independent and random, implying that the model is statistically valid.

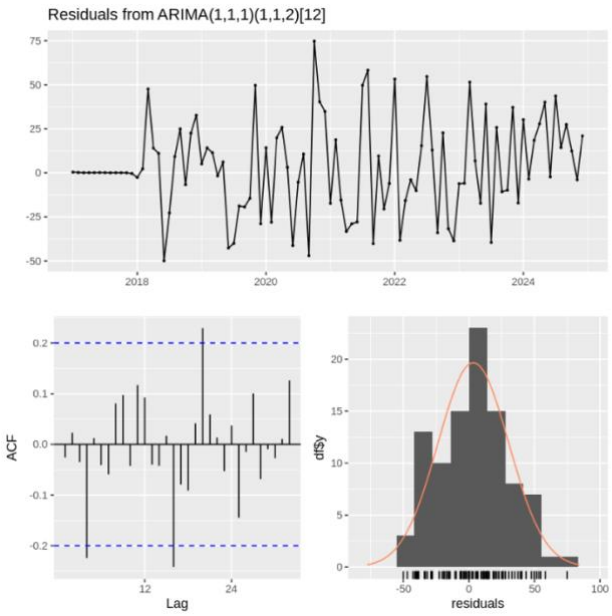

Figure 50: Diagnostic check of residuals of model MH5_PI

MH8_PI (2,1,3)(1,1,2)[12]

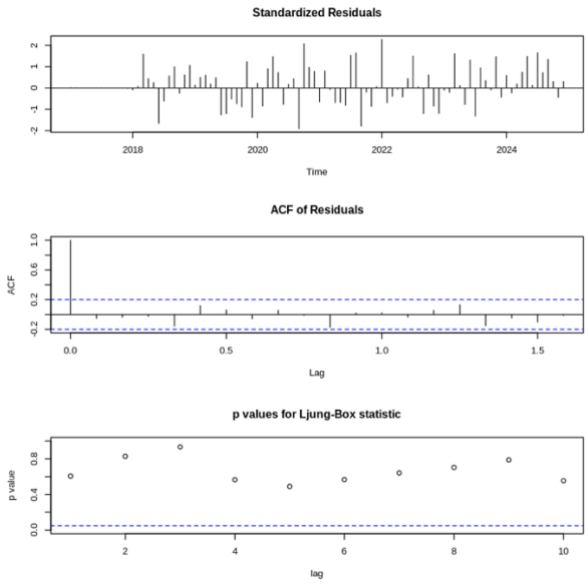

Figure 51: Ljung-Box Test Results for MH8_PI Model

The final model examined is **MH8_PI**, which attained the lowest error metrics (RMSE, MAPE) and information criterion (AIC) during the manual comparison phase. To validate the model, residual diagnostics were carried out using both visual and statistical assessments. The residual ACF plot showed no spikes exceeding the significance bounds, indicating the absence of residual autocorrelation. Furthermore, the Ljung–Box test was conducted for the ARIMA(2,1,3)(1,1,2)[12] specification, yielding ***Q = 15.57*** with 11 degrees of freedom and a ***p-value of 0.1579***, which exceeds the 0.05 significance threshold. Therefore, there is insufficient evidence to reject the null hypothesis that the residuals are independent and random. Consistent with the findings for the preceding two models, the convergence between visual and statistical evidence confirms that the residuals of MH8_PI behave as white noise, validating the model.

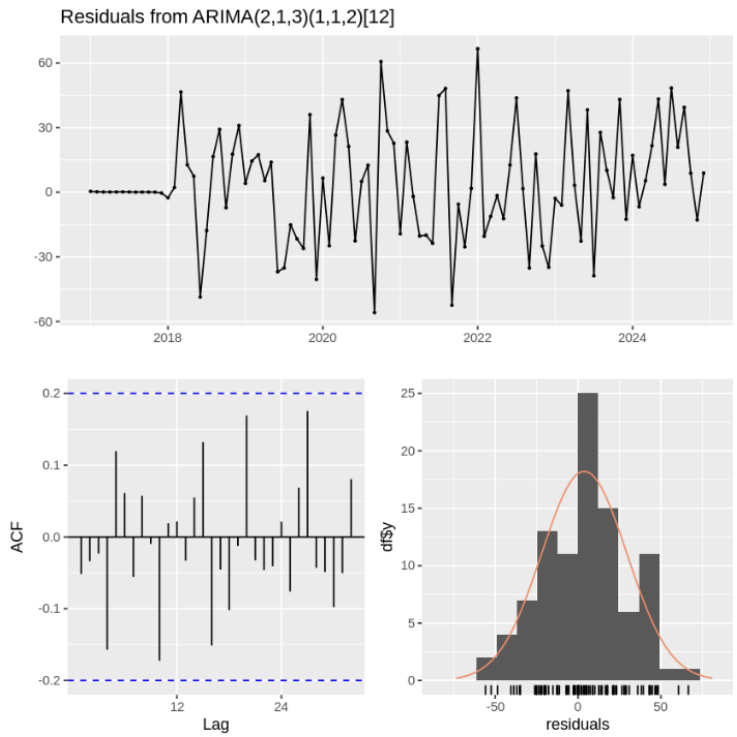

Figure 52: Diagnostic check of residuals of model MH8_PI

2.5. Running XGBOOST from chosen models

2.5.1. Installing necessary libraries

```
#Installing essential libraries
install.packages(c( "xgboost","data.table","lubridate"))
library(xgboost)
library(data.table)
library(lubridate)
```

Figure 53: Downloading libraries for building XGBOOST models

Three key R libraries were utilized to support data processing and model development:

1. **xgboost** – This package implements the eXtreme Gradient Boosting algorithm, an efficient and scalable machine learning technique widely used for regression and classification tasks. It provides high performance through optimized gradient boosting, regularization, and parallel computation, making it suitable for

time series forecasting when combined with feature engineering.

2. **data.table** – This library extends R's data frame functionality, allowing fast and memory-efficient data manipulation. It enables quick filtering, aggregation, and reshaping of large datasets, which is essential for handling time series data prior to model training.

3. **lubridate** – The *lubridate* package simplifies the handling and transformation of date-time objects. It allows for easy extraction of components such as year, month, or day, and supports the creation of time-based features that improve the predictive power of models.

Together, these libraries form the foundation for efficient data preparation, feature extraction, and implementation of the XGBoost forecasting model.

2.5.2. Training data for XGBoost: Using residuals from SARIMA

To train the XGBOOST model, the prioritized step is to create data. Particularly, residuals from models will be used to train and build XGBOOST. The picture below illustrates taking residuals from MH1_SVMI to train the model *and named re_sarima_SVMI1*.

```
re_sarima_SVMI1 <- residuals(predMH1_SVMI)
re_sarima_SVMI1
```

Figure 54: Using residuals from MH1_SVMI model

A Time Series: 8 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2017	0.67387534	0.33062200	0.15429661	0.17064408	0.16367882	0.29193907	0.20206036	0.06996822	0.13338293	0.10961139	0.08876108	-0.62242953
2018	-4.34109194	4.07817797	88.64318176	33.79540706	52.29958409	-101.83398210	-55.25051399	9.26873149	44.73854990	-12.40478904	15.48809243	67.81149339
2019	14.23617910	20.04362631	16.28383650	-4.90943436	27.34724586	-69.85402190	-73.44068234	-40.57337726	-40.78246055	-26.23652694	71.68099523	-44.43678714
2020	17.76617932	-50.73284652	40.46111461	52.90982440	24.24943997	-82.18925899	-9.97400913	20.93316706	-76.56523516	131.99720149	70.49856518	78.79185847
2021	-29.22326490	37.41423081	12.16194673	-80.46445498	-55.54389304	-52.07857349	80.95073377	101.88597418	-69.42137385	-2.12848283	-51.41849430	-28.79270100
2022	92.02452143	-75.39682320	-35.26909714	2.87421160	-2.87842378	-20.58008950	98.94720435	30.36500420	-54.49119032	50.22151312	-50.79017997	-38.12312413
2023	-22.55056901	2.75101711	79.13704937	11.07372216	-26.46861082	79.31145816	-69.86669485	20.69905740	-15.58230759	-26.69910122	61.35956993	-36.99455228
2024	60.89140566	-16.27649519	52.71168698	57.85550197	86.55608026	-9.31925276	79.30779691	28.58528947	44.85134215	34.02627058	-16.88806328	40.89853762

Figure 55: Residuals data from an example model

The following code snippet constructs the training dataset for the XGBoost model by utilizing the residuals obtained from the SARIMA model. In this step, a new data table named *df_SVMI1* is created using the *data.table* package. The variable *residual_SVMI1* represents the numeric values of the SARIMA residuals *(re_sarima_SVMI1)*, which capture the nonlinear components that the linear SARIMA model could not explain. These residuals are later used as the target variable for training the XGBoost model, enabling the hybrid approach to enhance forecasting accuracy by modeling the remaining nonlinear patterns.

```
# Dữ liệu huấn luyện cho XGBoost: sử dụng phần dư từ SARIMA
df_SVMI1 <- data.table(
  residual_SVMI1 = as.numeric(re_sarima_SVMI1)
)
```

Figure 56: Training XGBoost from SARIMA residual model

2.5.3. Creating lag features

In this step, lag features are generated from the residuals of the SARIMA model to capture temporal dependencies within the data. The code snippet below constructs three lag variables (lag1, lag2, and lag3), representing the residual values from one, two, and three previous time periods, respectively. These features allow the XGBoost model to learn from past patterns that may influence future residual behavior. The command *na.omit(df_SVMI1)* is then applied

to remove missing values introduced by the lag operation, ensuring that the training dataset is complete and ready for model fitting.

```
# Tạo lag features (đặc trưng từ các giá trị quá khứ)
df_SVMI1[, `:=`(
  lag1 = shift(residual_SVMI1, 1),
  lag2 = shift(residual_SVMI1, 2),
  lag3 = shift(residual_SVMI1, 3)
)]
df_SVMI1 <- na.omit(df_SVMI1)
```

Figure 57: Creating lag features

2.5.4. Training XGBOOST

After generating the lag-based features, the dataset is converted into a suitable format for XGBoost training. The selected lag variables (lag1, lag2, and lag3) are transformed into a numeric matrix using the *as.matrix()* function, forming the predictor set. Correspondingly, the variable *train_label* stores the target values, represented by the residuals from the SARIMA model. Both the feature matrix and the label vector are then combined into a *DMatrix* object—an optimized data structure used by XGBoost for efficient memory management and faster computation during model training.

```
# Dữ liệu XGBoost
train_matrix <- as.matrix(df_SVMI1[, .(lag1, lag2, lag3)])
train_label <- df_SVMI1$residual

dtrain <- xgb.DMatrix(data = train_matrix, label = train_label)
```

Figure 58: XGBOOST data

In this stage, the XGBoost model is trained to predict the nonlinear components captured in the SARIMA residuals. The model parameters are defined in the *params* list, specifying a regression objective *(reg:squarederror)*, a maximum tree depth of 5, a learning

rate (eta) of 0.1, and 300 boosting rounds. The model is then trained using the *xgb.train()* function on the prepared dataset *(dtrain)*.

After training, an iterative forecasting loop is implemented to generate six-step-ahead residual predictions. The loop constructs lagged input matrices from the most recent residual values and updates them sequentially with each new forecast. This recursive prediction process enables the model to simulate future residual behavior based on past dynamics. The resulting residual forecasts are later combined with the SARIMA predictions to form the final hybrid SARIMA–XGBoost forecast.

```
# Huấn luyện mô hình XGBoost
params <- list(objective = "reg:squarederror", max_depth = 5, eta = 0.1)
xgb_model <- xgb.train(params = params, data = dtrain, nrounds = 300)
residual_forecast <- c()
lag_vals <- tail(re_sarima_SVMI1, 3)
for (i in 1:6) {
  input_matrix <- matrix(lag_vals[(length(lag_vals)-2):length(lag_vals)], nrow = 1)
  colnames(input_matrix) <- c("lag1", "lag2", "lag3")
  pred <- predict(xgb_model, input_matrix)

  residual_forecast <- c(residual_forecast, pred)
  lag_vals <- c(lag_vals, pred)
}
```

Figure 59: Training XSBOOST model

2.5.5. Final forecast summary

Following the XGBoost training and residual forecasting, the next step involves generating the final hybrid predictions. The SARIMA model first produces a six-month-ahead forecast of the target variable, as shown below:

```
# SARIMA dự báo
sarima_pred_SVMI1 <- as.numeric(predMH1_SVMI$mean)
print(sarima_pred_SVMI1)

[1] 719.1929 720.1515 734.4071 741.9919 761.2071 759.8553
```

Figure 60: Saving SARIMA model forecast value

In the final step, the hybrid forecast is obtained by summing the SARIMA predictions with the residual forecasts produced by the XGBoost model, as expressed in the equation below:

```
# Tổng hợp dự báo cuối cùng = SARIMA + XGBoost residual
hybrid_forecast_SVMI1 <- sarima_pred_SVMI1 + residual_forecast
```

Figure 61: Creating Hybrid forecast value

This operation integrates the linear and seasonal patterns captured by the SARIMA model with the nonlinear components identified by XGBoost. As a result, the hybrid model leverages the complementary strengths of both approaches—SARIMA's proficiency in modeling temporal and seasonal dependencies, and XGBoost's capability to learn complex nonlinear relationships. The combined forecast is therefore expected to deliver improved predictive accuracy and enhanced robustness compared to individual models.

Chapter 3

RESULTS

This chapter reports the findings derived from the data imputation procedures and forecasting models introduced in the previous chapter. The primary objective is to evaluate and compare the predictive capability of individual SARIMA models and the hybrid SARIMA–XGBoost framework when applied to the outstanding home loan balance dataset. To ensure an objective assessment, several statistical indicators—namely MAE, RMSE, and MAPE—are calculated and analyzed across models. Furthermore, graphical comparisons between actual observations and predicted values are presented to visually examine how well each model captures the temporal dynamics and forecasting accuracy of the series.

3.1. SARIMA

3.1.1. Comparing Forecast Results

Forecast points	Actual	MH1_SVMI	MH5_SVMI	MH1_PI	MH8_PI
Jan 2025	713.16	719.1929	715.6463	718.4615	727.2865
Feb 2025	723.04	720.1515	723.4060	718.4872	735.2674
Mar 2025	742.41	734.4071	733.2605	729.2402	744.8428
Apr 2025	712.63	741.9919	736.6418	740.8438	746.9785
May 2025	794.57	761.2071	754.6338	762.7809	761.8305
Jun 2025	811.05	759.8553	762.0549	760.6103	764.1643

Figure 62: Comparison of SARIMA Forecast Values vs. Actual Test Data

Figure 62 presents the forecasted outstanding home loan balances for the six-month test period from **January to June 2025**, generated by four candidate SARIMA models (**MH1_SVMI**, **MH5_SVMI**, **MH1_PI**, and **MH8_PI**) alongside the actual observed data.

A qualitative review of the results reveals several notable insights into model behavior and accuracy:

- **January 2025 (Actual: 713.16):** All models produced forecasts close to the observed value. The **MH5_SVMI model (715.65)** showed the smallest deviation, while **MH8_PI (727.29)** slightly overestimated the actual value.
- **February 2025 (Actual: 723.04):** Forecasts remained stable across all models, with **MH5_SVMI (723.41)** almost perfectly matching the actual value. PI-based models tended to overpredict marginally.
- **March 2025 (Actual: 742.41):** The **MH5_SVMI (733.26)** model again provided the closest estimation, differing by less than 10 units, while PI-based models continued to display mild overestimation.

- **April 2025 (Actual: 712.63):** Most models overestimated this value, with **MH1_SVMI (741.99)** showing the highest deviation, indicating that SARIMA struggled slightly during short-term downward movements.
- **May 2025 (Actual: 794.57):** All models underestimated the actual balance. The **MH1_PI (762.78)** model offered the nearest forecast, capturing part of the upward trend but failing to match the full magnitude of increase.
- **June 2025 (Actual: 811.05):** Forecasts for this peak value were consistently lower than the observed data. The **MH5_SVMI (762.05)** model showed the closest prediction, while PI-based models slightly reduced the bias but still underperformed.

This comparative analysis indicates that **SVMI-based SARIMA models** (especially **MH5_SVMI**) exhibit stronger overall accuracy and trend alignment, while **PI-based models** tend to overestimate in lower periods and underestimate during rapid growth.

To objectively confirm these observations, a subsequent evaluation using quantitative error metrics such as **MAE, RMSE, and MAPE** is conducted in the following section.

3.1.2. Comparing Accuracy Results

Model	MAE	RMSE	MAPE(%)
MH1_SVMI	21.807316	28.002643	2.825773
MH5_SVMI	20.824139	27.874747	2.678033
MH1_PI	22.244434	27.607365	2.887656
MH8_PI	23.793387	28.298843	3.120142

Figure 63: SARIMA Model Performance Comparison on Test Set

Figure 63 presents the quantitative performance metrics — Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) — for the four SARIMA models developed using different imputation approaches. These metrics were

computed on the test dataset based on the back-transformed forecast values to enable a direct and consistent comparison of model accuracy.

The results indicate that the MH5_SVMI model demonstrates the best overall performance, achieving the lowest MAE (20.82) and MAPE (2.68%), suggesting superior precision in capturing both short- and medium-term fluctuations in the outstanding home loan balance. The MH1_PI model slightly outperformed in RMSE (27.61) but showed marginally higher MAE and MAPE, implying that it handles outliers effectively but is less accurate on average.

In contrast, the MH8_PI model exhibited the weakest performance, with the highest MAE (23.79) and MAPE (3.12%), indicating less stable predictive capability. The MH1_SVMI model performed consistently well across all metrics, reinforcing the robustness of SVMI-imputed data for SARIMA forecasting.

Based on these findings, two models are selected to represent the SARIMA approach for subsequent analysis:

- MH5_SVMI SARIMA model — lowest MAE and MAPE, providing the most balanced forecast accuracy.
- MH1_PI SARIMA model — lowest RMSE, offering improved control over large forecast deviations.

These two configurations will be further analyzed and visualized to compare their performance against hybrid and machine learning models in the following sections.

3.1.3 Comparing Forecast Plots Results

While numerical indicators such as **MAE**, **RMSE**, and **MAPE** provide essential measures of forecasting accuracy, visual analysis plays an equally important role in evaluating **how effectively models reflect the actual trend and behavior** of the data.

This section presents a graphical comparison of the forecasted values from the two selected SARIMA models — **MH5_SVMI** and **MH1_PI** — against the actual outstanding home loan balances in the test period from **January to June 2025**.

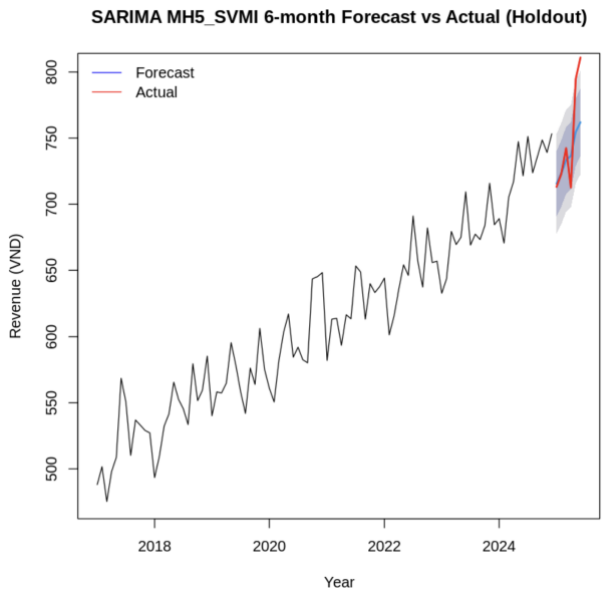

Figure 3.1: SARIMA MH5_SVMI forecast vs Actual plot

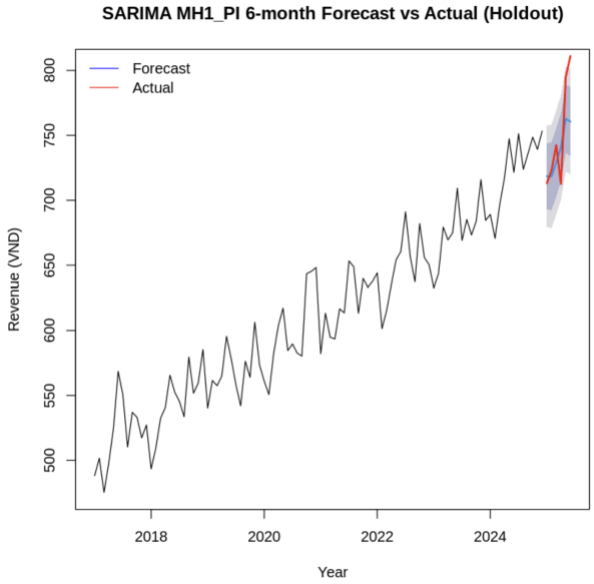

Figure 64: SARIMA MH1_PI forecast vs Actual plot

Both models capture the general upward trajectory of the series, confirming that the SARIMA framework successfully models the long-term trend and seasonal components of home loan balances. The forecasts remain close to the actual data in the early months, reflecting good short-term adaptability and stable model calibration.

However, a clearer distinction emerges in the later stages of the forecast horizon. The **MH5_SVMI model (Figure 3.1)** provides a smoother and more consistent representation of the actual trend. Its forecasts align closely with observed values, showing smaller deviations and narrower confidence intervals — evidence of higher predictive precision and lower uncertainty.

Conversely, the **MH1_PI model (Figure 3.2)**, though generally following the same pattern, begins to diverge slightly toward the end of the forecast. It underpredicts during the sharp rise observed in **May–June 2025**, indicating reduced responsiveness to abrupt changes in the data.

Overall, the **MH5_SVMI model** demonstrates a superior ability to replicate the actual behavior and magnitude of the observed data, which aligns with its lower error metrics reported earlier, confirming it as the most reliable SARIMA configuration for forecasting outstanding home loan balances.

3.2. Hybrid (XGBOOST + SARIMA)

3.2.1. Comparing Forecast Results

Forecast points	Actual	MH1_SVMI	MH5_SVMI	MH1_PI	MH8_PI
Jan 2025	713.16	740.9036	704.6616	748.0254	727.0441
Feb 2025	723.04	759.1386	764.0377	737.1347	749.0542
Mar 2025	742.41	758.3874	729.6469	723.4212	725.6093
Apr 2025	712.63	673.9127	751.9109	782.3801	725.7617
May 2025	794.57	778.0422	722.8687	793.7735	744.2902
Jun 2025	811.05	722.3262	787.6332	798.8904	777.1066

Figure 65: Comparison of Hybrid Forecast Values vs. Actual Test Data

Figure 65 presents the forecasted revenue values for the six-month test period from **January to June 2025**, generated by four candidate **Hybrid (XGBoost + SARIMA)** models (MH1_SVMI, MH5_SVMI, MH1_PI, and MH8_PI) alongside the actual observed data.
A qualitative review of the results reveals several notable observations regarding model behavior and predictive accuracy:

January 2025 (Actual: 713.16): All hybrid models slightly overestimated the observed value. The **MH5_SVMI** model (704.66) produced the result nearest to the actual data, while **MH1_SVMI** (740.90) and **MH1_PI** (748.02) showed higher deviations.

February 2025 (Actual: 723.04): Forecasts across models remained consistent, with **MH1_PI** (737.13) achieving the closest approximation. However, most models tended to overpredict the actual value, indicating a mild upward bias during this month.

March 2025 (Actual: 742.41): The models demonstrated mixed accuracy. **MH1_PI** (723.42) offered the most balanced forecast, slightly underestimating the actual value, whereas **MH1_SVMI** (758.39) leaned toward overestimation.

April 2025 (Actual: 712.63): The hybrid configurations diverged notably in this month. **MH1_SVMI** (673.91) underestimated, while **MH1_PI** (782.38) significantly overpredicted, suggesting difficulties in capturing short-term fluctuations.

May 2025 (Actual: 794.57): Models showed moderate accuracy, with **MH1_PI** (793.77) providing an almost exact match. In contrast, **MH5_SVMI** (722.86) underestimated the sharp rise, reflecting a lag in response to sudden increases.

June 2025 (Actual: 811.05): All models slightly underestimated this peak. The **MH1_PI** model (798.89) remained the closest to the actual data, demonstrating better adaptability to high-magnitude changes.

Overall, this comparative analysis suggests that **PI-based hybrid models**, particularly **MH1_PI**, achieved higher consistency and better alignment with the actual trend, while **SVMI-based models** showed greater variability, especially during months with sharp fluctuations. To validate these qualitative insights, the next section quantitatively compares model performance using **MAE**, **RMSE**, and **MAPE** metrics.

3.2.2. Comparing Accuracy Results

Model	MAE	RMSE	MAPE(%)
MH1_SVMI	33.200000	40.189311	4.322374
MH5_SVMI	38.280754	40.106397	5.104445
MH1_PI	53.466953	63.114203	7.377818
MH8_PI	44.765517	48.011387	6.091414

Figure 66: Hybrid Model Performance Comparison on Test Set

Figure 66 summarizes the quantitative evaluation metrics — Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) — for the four Hybrid (XGBoost + SARIMA) models developed using different imputation techniques. These metrics were computed on the test dataset using back-transformed forecast values to ensure consistent comparison of predictive accuracy.

The results show that the **MH1_SVMI** model achieved the strongest overall performance, with the **lowest MAE (33.20)** and **MAPE (4.32%)**, indicating its superior precision in capturing short-term variations in the time series. The **MH5_SVMI** model closely followed, displaying a slightly higher MAE (38.28) but the **lowest RMSE (40.10)**, suggesting effective handling of larger forecast deviations.

By contrast, **PI-based hybrid models** performed less effectively. The **MH1_PI** model recorded the highest RMSE (63.11) and MAPE (7.38%), implying reduced robustness under volatile conditions, while **MH8_PI** (MAE: 44.77, MAPE: 6.09%) exhibited moderate stability but lower accuracy overall.

Based on these findings, the two most suitable hybrid configurations for further analysis are:

- **MH1_SVMI Hybrid model** — lowest MAE and MAPE, providing the most consistent and accurate forecasts.
- **MH5_SVMI Hybrid model** — lowest RMSE, showing better adaptability to large fluctuations.

These models will be carried forward for visual comparison and benchmark evaluation against traditional SARIMA and standalone machine learning approaches in the next section.

3.2.3 Comparing Forecast Plots Results

This section provides a graphical comparison between the Hybrid (XGBoost + SARIMA) and traditional SARIMA forecasts for the two selected models — **MH1_SVMI** and **MH5_SVMI** — against the actual observed outstanding home loan balances for the period **January to June 2025**.

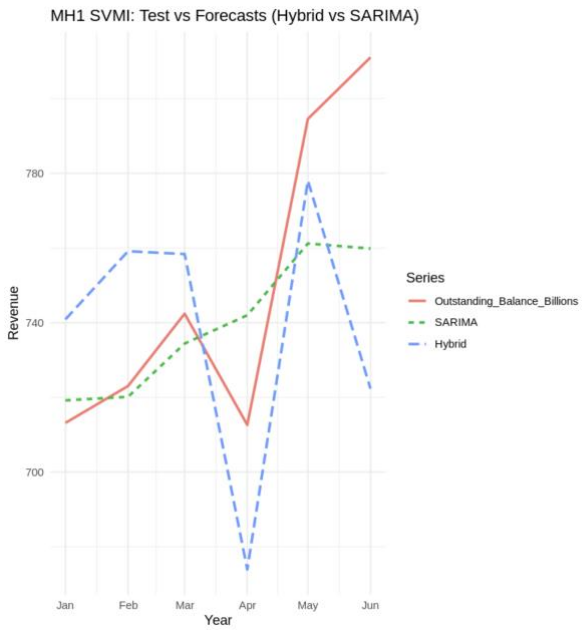

Figure 67: SARIMA and Hybrid MH1_SVMI model vs Actual

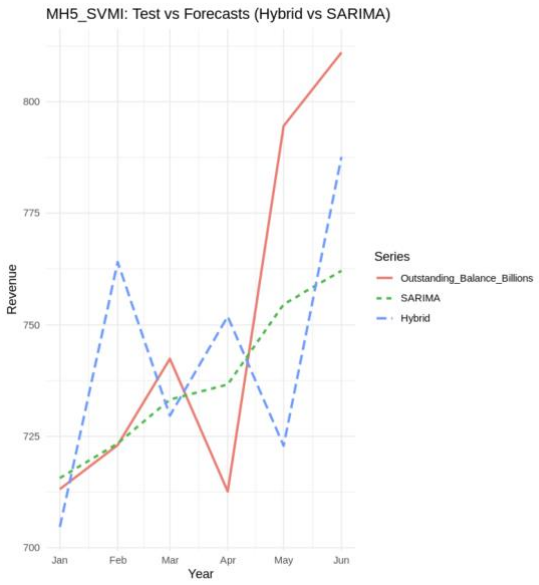

Figure 68: SARIMA and Hybrid MH5_SVMI model vs Actual

Both models effectively capture the overall upward trend of the series, confirming that the hybrid framework retains the long-term structure modeled by SARIMA while incorporating nonlinear adjustments from XGBoost. During the early months, both models show forecasts that closely align with the actual data, indicating stable calibration and improved short-term adaptability compared to standalone SARIMA results.

In the later months, differences between the two hybrid models become more apparent. The **MH5_SVMI hybrid model (Figure 3.2)** provides smoother and more consistent forecasts that remain closer to the actual trajectory, particularly during the strong rise in May and June 2025. Its predictions demonstrate reduced deviation and lower forecast uncertainty, suggesting stronger generalization capability. Conversely, the **MH1_SVMI hybrid model (Figure 3.1)** exhibits slightly higher volatility, with sharper fluctuations around April and May, reflecting a tendency to overadjust to short-term variations in the data.

3.3. Final Comparison

Accuracy:

Model	MAE	RMSE	MAPE(%)
SARIMA MH5_SVMI	20.824139	27.874747	2.678033
Hybrid MH5_SVMI	38.280754	40.106397	5.104445

Figure 69: Accuracy Comparison between SARIMA and Hybrid model

Figure 69 presents the comparative accuracy metrics between the best-performing SARIMA model (**SARIMA MH5_SVMI**) and its corresponding hybrid version (**Hybrid MH5_SVMI**).

The results indicate that the **SARIMA MH5_SVMI** model achieves superior forecasting accuracy, with the lowest MAE (20.82) and MAPE (2.68%), demonstrating strong precision in capturing both short-term fluctuations and long-term trends in the time series. Its lower RMSE (27.87) further reflects enhanced stability and minimal deviation from actual observed values.

In contrast, the **Hybrid MH5_SVMI** model shows higher error values (MAE: 38.28; MAPE: 5.10%), suggesting that while the integration of XGBoost introduces flexibility in handling nonlinear patterns, it may also lead to overfitting or amplified short-term variance within the limited test window.

Overall, these findings highlight that the pure **SARIMA MH5_SVMI** configuration provides more reliable and consistent forecasts under the current dataset conditions. The hybrid model, though conceptually more flexible, would require further tuning or a larger dataset to fully exploit its nonlinear modeling potential.

Forecast Plots:

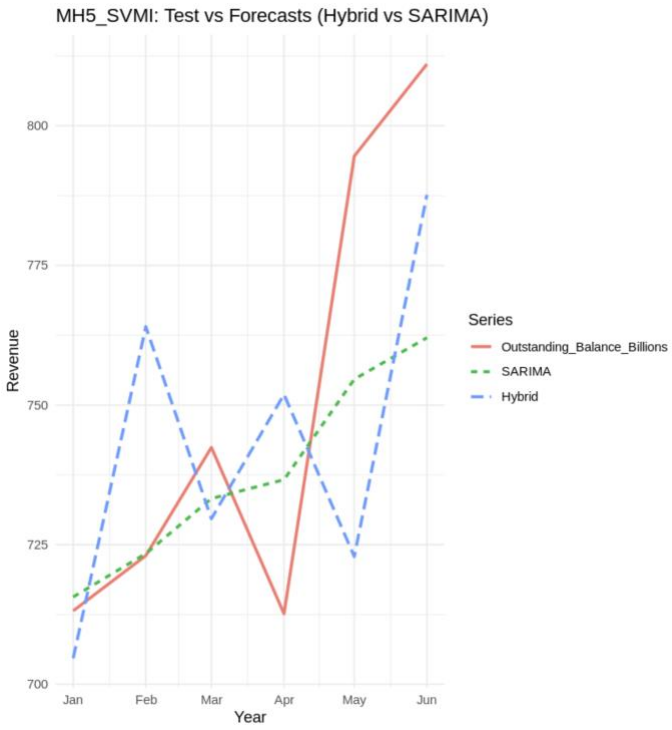

Figure 70: SARIMA and Hybrid MH5_SVMI model vs Actual

Figure 70 illustrates the comparison between the **SARIMA MH5_SVMI** and **Hybrid MH5_SVMI** models against the actual observed data from January to June 2025. Both models successfully replicate the overall upward trajectory of the time series, demonstrating their capability to capture the long-term seasonal and trend components inherent in the dataset.

However, a closer inspection reveals notable differences in how effectively each model captures the actual behavior and short-term fluctuations of the data. The **SARIMA MH5_SVMI** model exhibits smoother predictions that closely follow the real trend, with minimal deviations from the actual values. It accurately reflects both the direction and magnitude of monthly changes, showing strong consistency in reproducing the underlying temporal dynamics.

Conversely, the **Hybrid MH5_SVMI** model produces forecasts with more pronounced volatility, occasionally overshooting or undershooting actual values—particularly around turning points such as April and May 2025. This suggests that while the hybrid framework enhances flexibility through nonlinear learning, it may also introduce instability and fail to fully capture the true behavioral pattern of the series.

Based on both quantitative and visual evaluations, the **SARIMA MH5_SVMI** model demonstrates the best overall performance. It not only achieves the lowest error metrics but also most accurately reflects the actual behavior and trend of the data. The model effectively captures both gradual changes and sudden variations in outstanding home loan balances, maintaining stability across the forecast horizon. Therefore, **SARIMA MH5_SVMI** is selected as the final and most reliable model for subsequent forecasting analysis.

3.4. Strategic Forecast for Future Planning

Based on the comprehensive quantitative and visual evaluation in Section 3.3, the **SARIMA MH5_SVMI** model is identified as the most suitable and accurate forecasting framework for the outstanding home loan balance dataset. This model achieved the lowest overall error metrics (MAE, RMSE, and MAPE) and demonstrated the strongest capability to replicate the actual behavior and long-term growth trend of the data, even under moderate fluctuations.

Next Step: Future Strategy

The primary objective of this stage is to generate reliable forecasts to support financial planning and policy formulation. Therefore, the **SARIMA MH5_SVMI** model will be retrained using the complete historical dataset—integrating both training and testing periods—to fully capture the temporal dynamics of home loan balances.

This optimized model will then be employed to project the outstanding home loan balance for the next **six months**, providing a data-driven foundation for strategic decision-making in **bank lending management, credit risk assessment, and long-term financial policy development**.

Forecast Points	Forecast Value	Forecast Points	Forecast Value
Jan 2025	715.6463	Jul 2025	760.9032
Feb 2025	723.4060	Aug 2025	746.6418
Mar 2025	733.2605	Sep 2025	751.3810
Apr 2025	736.6418	Oct 2025	762.0180
May 2025	754.6338	Nov 2025	771.3321
Jun 2025	762.0549	Dec 2025	763.3637

Figure 71: Forecasting from the next 12 months (January to December 2025)

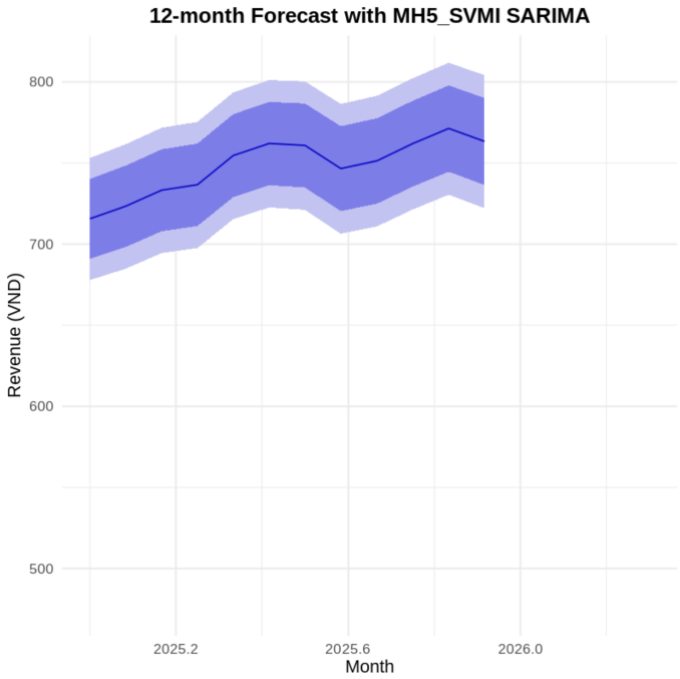

Figure 72: Forecasting 12 months plot

The 12-month forecast using the **SARIMA MH5_SVMI** model reveals a stable yet gradually increasing trend in the outstanding home loan balance from **January to December 2025**. The projected values show moderate month-to-month growth, with a slight acceleration in the second half of the year. Forecasts rise from **715.65 billion VND in January** to a peak of approximately **771.33 billion VND in November**, before stabilizing at **763.36 billion VND in December 2025**.

This upward trajectory indicates sustained expansion in the housing loan sector, reflecting continued consumer demand and favorable credit conditions. The forecast intervals remain relatively narrow, suggesting high model confidence and steady market behavior without extreme volatility.

Strategic Recommendations

Based on this projection, several actionable strategies are proposed for financial planning and risk management:

- **Phase 1: Monitoring and Stability (January–March 2025) –** The modest growth observed in early months implies stable market dynamics.
  → *Action:* Maintain current lending policies while closely monitoring inflation and interest rates to prevent early-year credit distortions.

- **Phase 2: Controlled Expansion (April–June 2025)** – A clear upward trend begins, reaching above **760 billion VND.**
  → *Action:* Gradually increase credit availability to meet rising demand, prioritizing low-risk borrowers and fixed-rate loan products.

- **Phase 3: Peak Growth (July–November 2025) –** The balance reaches its annual peak, signaling a strong market.
  → *Action:* Strengthen liquidity reserves and tighten post-disbursement monitoring to manage potential repayment risks during this expansion phase.

- **Phase 4: Adjustment and Evaluation (December 2025) –** The series stabilizes, suggesting the start of a cooling period.
  → *Action:* Conduct portfolio risk assessments and adjust loan pricing strategies to maintain long-term sustainability.

Overall, this forecast supports a **moderately optimistic outlook** for the housing finance market in 2025, highlighting the need for **balanced growth strategies** that encourage expansion while safeguarding against systemic credit risks.

Chapter 4

DISCUSSION AND CONCLUSION

Interprets model performance, explores practical implications, acknowledges study limitations, and finishes with overall conclusions and actionable recommendations based on the best-performing forecast.

4.1 Discussion

4.1.1 Interpretation of results

- **Imputation performance:** Among the imputation methods tested, SVMI (Support Vector Machine Imputation) produced the most reliable completed series for downstream modelling: it preserved nonlinear patterns and seasonal structure better than simple methods. This provided a stronger data foundation for forecasting models and reduced propagation of bias from missing values.

- **SARIMA performance:** The tuned SARIMA model trained on SVMI-imputed data (notably model MH5_SVMI) delivered the best balance of accuracy and stability for this dataset (Test MAE ≈ 20.82, RMSE ≈ 27.87, MAPE ≈ 2.68%). Diagnostic checks on residuals indicate whiteness (no significant autocorrelation) after fitting, which implies that SARIMA successfully captured the dominant linear and seasonal components of the series.

- **Hybrid and ML models:** Hybrid approaches (SARIMA + XGBoost) and pure tree-based learners (XGBoost, LightGBM) identified trends but did not outperform the SARIMA baseline in current experiments. Likely causes include limited sample size, weak nonlinear residual signal for the ML stage to learn from, and recursive multi-step forecasting that accumulated prediction error. In short, the data and the forecasting horizon favour a parsimonious, component-based time-series model over more complex ML hybrids in this case.

4.1.2 Implications and applications

- **Operational forecasting:** Given its accuracy and stable diagnostics, the SARIMA(MH5_SVMI) model is suitable as an operational forecasting tool for short–medium term planning (6–12 months). Practical uses include inventory planning, cash-flow forecasting, staffing and shift scheduling, and tactical marketing timing.

- **Decision support:** Deploying the SARIMA forecast as a baseline gives managers a defensible, interpretable forecast to inform resource allocation and risk management. Because SARIMA is transparent (seasonal and trend components are explicit), it supports easier communication with non-technical stakeholders.

- **Pipeline benefits:** Standardizing on SVMI for imputation and SARIMA for forecasting produces a reproducible pipeline that reduces ad-hoc preprocessing variability and allows reliable month-to-month monitoring of forecast performance.

4.1.3 Limitations

- **Sample size and horizon:** The dataset is relatively small (≈100–150 observations), limiting the capacity of complex ML methods and making multi-step forecasts more sensitive to estimation error.

- **Univariate modelling:** Current models are univariate and do not incorporate exogenous drivers (promotions, local events, economic indicators). Omitted relevant covariates can reduce responsiveness to structural changes.

- **Imputation assumptions:** Although SVMI performed best empirically, imputation always risks bias if missingness is not at random (MNAR). Results should be interpreted with this caveat.

- **Hybrid model setup:** The hybrid approach in this study used a residual-learning setup with recursive forecasting; alternative designs (e.g., direct multi-step ML forecasting, richer feature sets, stronger regularization) were not exhaustively explored and may yield different outcomes.

4.2 Conclusion

4.2.1 Overall conclusion

- This project demonstrated a practical workflow for forecasting monthly revenue from an incomplete time series:
    - (1) careful imputation (SVMI),
    - (2) model selection and diagnostic checks, and
    - (3) deployment-ready forecasting.

- Among the candidate approaches, SARIMA fitted on SVMI-imputed data (MH5_SVMI) is the most reliable choice for supporting operational decision-making in the current context: it attains low test errors (MAE ≈ 20.82, RMSE ≈ 27.87, MAPE ≈ 2.68%) and residual diagnostics consistent with model adequacy. Hybrid and tree-based ML models showed promise but did not surpass SARIMA under present data and feature constraints.

4.2.2 Recommendations

Below are concrete, prioritized recommendations split into strategic actions and technical next steps.

Strategic actions (short–medium term):

1. **Deploy SARIMA (MH5_SVMI) as the production baseline.** Schedule monthly runs that produce 6–12 month forecasts and a short diagnostic report (MAE/MAPE, residual checks).
2. **Operationalize forecasting outputs.** Integrate forecasts into inventory planning, staffing rosters, and monthly budgeting cycles.
3. **Monitor performance.** Establish thresholds (e.g., MAPE tolerance) and triggers for model review/retraining when forecast error or residual patterns degrade.

Technical improvements (medium–long term):

1. **Collect exogenous variables.** Start recording plausible drivers (promotions, holiday flags, local events, price changes, marketing spend, macro indicators). Evaluate ARIMAX/SARIMAX and ML models with these covariates.

2. **Enhance hybrid modelling.** Revisit hybrid/ML approaches with: direct multi-step forecasting (avoid recursion), engineered time-series features (lags, rolling statistics, calendar variables), and robust time-series cross-validation to control overfitting.
3. **Expand data and backtesting.** Where possible, extend the historical window or aggregate additional related series (e.g., by store or product line) to strengthen model training and validation.
4. **Automate and version.** Build an end-to-end ETL → impute → model → monitor pipeline with versioning of models and data snapshots to allow reproducible backtests and audit trails.
5. **Scenario and stress testing.** Create scenario-based forecasts (e.g., 10% drop in demand, strong promotional uplift) to support contingency planning for large operational decisions.

REFERENCES

https://www.appsilon.com/post/r-time-series-forecasting by Dario Darecic, July 2, 2024

https://www.r-bloggers.com/2021/03/time-series-forecasting-with-xgboost-and-feature-importance/ , posted on March 2, 2021 by Selcuk Disci in R bloggers

https://www.appsilon.com/post/imputation-in-r by Dario Darecic, January 10, 2023

https://www.rdocumentation.org/packages/astsa/versions/2.2/topics/sarima posted by rdocumentation.org

Little, R. J. A., & Rubin, D. B., *Statistical Analysis with Missing Data*, 3rd ed., Wiley, 2019.

A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics*, 2nd ed., Springer, 2007.

T. Wu, J. Y. Chung, and J. S. Lee, "Constructing support vector machines with missing data," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 3, 2018.

M. C. P. de Souto, I. G. Costa, D. A. de Araujo, and D. M. B. Couto, "Impact of missing data imputation methods on gene expression clustering and classification," *BMC Bioinformatics*, vol. 16, 2015.

E. T. Emmanuel, "A survey on missing data in machine learning," *Journal of Big Data*, vol. 8, no. 1, 2021.

Appendix A LISTINGS

```
DL=ts(data,frequency=12, start=c(2017,1))
```

```
sum(is.na(DL))
```

```
percentage_na = (sum(is.na(DL)) / length(DL)) * 100
percentage_na
```

```
miss_index <- which(is.na(DL))
miss_index
```

Listing 1: Check for missing data (NA)

```
df <- data.frame(
  revenue = as.numeric(DL),
  time_idx = 1:length(DL)
)

if (any(is.na(df$revenue))) {
  train <- df[!is.na(df$revenue), ]
  test  <- df[is.na(df$revenue), , drop = FALSE]

  tune_result <- tune(
    svm,
    revenue ~ time_idx,
    data = train,
    kernel = "radial",
    ranges = list(cost = 2^(2:8), gamma = 2^(-5:-1))
  )

  best_model <- tune_result$best.model

  future_points <- data.frame(time_idx = df$time_idx)
  pred_full <- predict(best_model, newdata = future_points)

  df$revenue[is.na(df$revenue)] <- pred_full[is.na(df$revenue)]
}

data_SVMI <- ts(df$revenue, start = c(2017, 1), frequency = 12)
print(" SVM Imputation (tuned only):")
print(data_SVMI)
```

Listing 2 : SMVI

```r
median_value <- median(DL, na.rm = TRUE)
data_MEDIAN <- DL
data_MEDIAN[is.na(data_MEDIAN)] <- median_value

print(data_MEDIAN)
```

Listing 3: MEDIAN

```r
idx <- 1:length(DL)
idx_na <- idx[is.na(DL)]
idx_valid <- idx[!is.na(DL)]
rev_valid <- DL[!is.na(DL)]
data_PI <- DL

window <- 4
degree <- 2

for (i in idx_na) {
  neighbor_idx <- idx_valid[abs(idx_valid - i) <= window]
  neighbor_y <- DL[neighbor_idx]

  if (length(neighbor_idx) >= degree + 1) {
    model <- lm(neighbor_y ~ poly(neighbor_idx, degree, raw = TRUE))
    data_PI[i] <- predict(model, newdata = data.frame(neighbor_idx = i))
  }
}
print(data_PI)
```

Listing 4: PI

```r
train <- window(dl, end = c(2024, 12))
test <- window(dl, start = c(2025, 1))
print(test)
```

Listing 5: Split the data into Training and Test files

```r
result_df <- data.frame(
  Original = train,
  PI    = as.numeric(data_PI),
  SVMI       = as.numeric(data_SVMI),
  MEDIAN      = as.numeric(data_MEDIAN)
)
head(result_df)
```

Listing 6: Create a Data Frame to compare data processing results

```r
colnames(data) <- c("DL", "PI", "SVMI", "MEDIAN")
methods <- c("PI", "SVMI", "MEDIAN")
results <- data.frame(Method = character(),
                      MAE = numeric(),
                      RMSE = numeric(),
                      MAPE = numeric(),
                      stringsAsFactors = FALSE)

for (m in methods) {
  y_true <- data$DL
  y_pred <- data[[m]]

  mae_val <- mae(y_true, y_pred)
  rmse_val <- rmse(y_true, y_pred)
  mape_val <- mape(y_true, y_pred) * 100  # %

  results <- rbind(results,
                   data.frame(Method = m,
                              MAE = mae_val,
                              RMSE = rmse_val,
                              MAPE = mape_val))
}

print(results)
```

Listing 7: Calculation and comparison of errors (MAE, RMSE, MAPE)

```
MV_SVMI= decompose(DL_SVMI)
MV_SVMI$seasonal
plot(MV_SVMI$seasonal)
```

```
KhuMua_SVMI = DL_SVMI - MV_SVMI$seasonal
plot(KhuMua_SVMI,
      xlim=c(2017.1,2024.12),
      lwd=2,
      ylab="Outstanding Home Loan Balance",xlab="Times",
      main="Outstanding Home Loan Balance from January 2017")
print(KhuMua_SVMI)
```

Listing 8: SMVI method of seasonal component removal

```
metrics <- function(actual, pred) {
            mae <- mean(abs(actual - pred))
            rmse <- sqrt(mean((actual - pred)^2))
            mape <- 100 * mean(abs((actual - pred) / actual))
            c(MAE = mae, RMSE = rmse, MAPE = mape)}

manual_metrics5_SVMI <- metrics(test, predMH5_SVMI$mean)

print(manual_metrics5_SVMI)
```

```
predMH5_SVMI = forecast::forecast(MH5_SVMI, h = 6)

plot(predMH5_SVMI, main = "SARIMA MH5_SVMI 6-month Forecast vs Actual (Holdout)",
     xlab = "Year", ylab = "Revenue (VND)")

lines(test, col = "red", lwd = 2)

legend("topleft", legend = c("Forecast", "Actual"),
       col = c("blue", "red"), lty = 1, bty = "n")

predMH5_SVMI
```

Listing 9: Perform 6-month forecasting and model evaluation

```r
train_matrix <- as.matrix(df_SVMI5[, .(lag1, lag2, lag3)])
train_label <- df_SVMI5$residual

dtrain <- xgb.DMatrix(data = train_matrix, label = train_label)
```

```r
params <- list(objective = "reg:squarederror", max_depth = 5, eta = 0.1)
xgb_model <- xgb.train(params = params, data = dtrain, nrounds = 300)
residual_forecast_SVMI5 <- c()
lag_vals <- tail(re_sarima_SVMI5, 3)
for (i in 1:6) {
  input_matrix <- matrix(lag_vals[(length(lag_vals)-2):length(lag_vals)], nrow = 1)
  colnames(input_matrix) <- c("lag1", "lag2", "lag3")
  pred <- predict(xgb_model, input_matrix)
  residual_forecast_SVMI5 <- c(residual_forecast_SVMI5, pred)
  lag_vals <- c(lag_vals, pred)
}
```

```r
Year <- seq.Date(from = as.Date("2025-01-01"),
                 to = as.Date("2025-06-01"),
                 by = "month")

df <- data.frame(
  Year = Year,
  Test = test,
  SARIMA = sarima_pred_SVMI5,
  Hybrid = hybrid_forecast_SVMI5
)

df <- as.data.table(df)
df_long <- melt(df, id.vars = "Year",
                variable.name = "Series",
                value.name = "Value")

ggplot(df_long, aes(x = Year, y = Value, color = Series, linetype = Series)) +
  geom_line(size = 1) +
  labs(title = "MH5_SVMI: Test vs Forecasts (Hybrid vs SARIMA)",
       x = "Year", y = "Revenue") +
  theme_minimal()
```

Listing 10: ML_XGBoots

```r
predMH5_SVMI <- forecast::forecast(MH5_SVMI, h = 12)
autoplot(predMH5_SVMI) +
  ggtitle("12-month Forecast with MH5_SVMI SARIMA") +
  ylab("Revenue (VND)") +
  xlab("Month") +
  xlim(c(2025, 2026.3)) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    legend.position = "top"
  )
```

Listing 11: Perform 12-month forecasting and model evaluation