Cash Flow Forecasting Using Machine Learning Techniques

Master's Thesis
MSc M2MO Random Modelling, Finance and Data Science (Ex DEA Laure ELIE)

Dylan Riboulet

Company Supervisor: Courillon Anthony (Crédit Agricole CIB)
Academic Supervisor: Jean-François Chassagneux (Université Paris Cité)

Jury:
Zorana Grbac
Karine Tribouley

Academic Year: 2023-2024

Contents

Thanks

I would like to express my deep gratitude to Mr. Anthony Courillon, my mentor, for his unwavering support throughout my internship. His constant availability and patience in explaining the complex financial aspects of various projects have been invaluable. Thanks to his

insightful advice and expertise, I have developed a deeper understanding of finance especially securitization, which has significantly enriched my experience and professional skills.

I would like to thank Adrien Pimmel for the pleasure I had working with him on the dashboard. His availability and responsiveness to all my questions greatly facilitated our collaboration and contributed to the success of the project.

I would also like to warmly thank the entire team for the enjoyable moments we shared, especially during coffee breaks, as well as for the overall pleasant atmosphere that prevailed during these six months. Your good spirits and camaraderie made this experience particularly enriching and enjoyable.

I would also like to express my gratitude to Mrs. Carassus, who encouraged me throughout my studies and gave me the opportunity to be part of this master's program.

1 Abstract

This master's thesis explores the application of machine learning techniques to forecast cash flows for Italian clients in the HEP_EUR_1M securitization pool, conducted during my internship with the Credit Risk & Portfolio Management team at Crédit Agricole Corporate & Investment Banking. The primary objective of this research is to develop predictive models that enhance the accuracy of cash flow forecasts, thereby improving risk management within the securitization portfolio.

The study investigates various supervised learning techniques applied to time series data, including Random Forest, Bagging, Multilayer Perceptron, and Convolutional Neural Networks, alongside econometric models such as AR and SARIMA, and resampling methods. The research employs these models to identify the most effective approach for forecasting, ultimately aiming to provide actionable insights for better portfolio management.

This work also delves into the intricacies of cross-validation strategies, particularly tailored for time series forecasting, to ensure robust model evaluation and selection. The findings of this study are intended to contribute to the advancement of risk management practices within the securitization industry by leveraging cutting-edge machine learning techniques to help the trading desk.

2 Introduction

Securitization is a financial technique that transforms illiquid assets into marketable securities. By converting these assets into tradable securities, entities are able to free up liquidity and transfer credit risks to investors. This process involves the sale of financial assets, such as loans or receivables, to a bankruptcy-remote entity known as a securitization conduit. The conduit issues asset-backed securities (ABS), which are then sold to investors. This structure allows for the transformation of a pool of illiquid assets into more liquid, tradable instruments, enhancing market efficiency.

In the context of Asset-Liability Management (ALM), securitization presents unique challenges and opportunities. ALM is the process of managing financial risks that arise due to mismatches between the assets and liabilities (in terms of maturities, interest rates, and currencies). The conduit's role in this context is crucial, as it serves as the intermediary that ensures that the assets' cash flows adequately cover the liabilities generated by the issuance of securities, minimizing risks such as liquidity shortfalls or interest rate mismatches.

The primary advantages of securitization for clients include improved liquidity and better credit risk management. Additionally, it offers more cost-effective financing by providing access

to a wide range of investors in the capital markets. The diversity of asset classes that can be securitized, from auto loans to trade receivables, enhances the flexibility of this financial tool.

A securitization conduit is a special purpose vehicle (SPV) specifically designed to purchase and manage loan-backed assets while isolating the risk from the originating financial institution. The conduit issues commercial paper or other forms of debt, with the revenues from these instruments used to finance the asset purchases. By isolating the assets from the originator's balance sheet, the conduit protects investors in the event of the originator's bankruptcy.

The Asset-Backed Commercial Paper (ABCP) market plays a vital role in offering short-term financing options to corporations, often at lower costs than traditional bank loans. The securities issued in this market are backed by a pool of assets, ensuring some level of security for investors.

Crédit Agricole Corporate and Investment Bank (Crédit Agricole CIB) is a major player in this market. Its securitization business within its Global Markets division provides clients with structured financing solutions that are tailored to their risk profiles and financial goals. The bank has also emerged as a leader in green and sustainable finance, integrating environmental considerations into its financial operations.

CA CIB operates through three major securitization conduits—two in the U.S. (La Fayette Asset Securitization, LLC and Atlantic Asset Securitization, LLC) and one in Europe (LMA S.A./LMA-Americas LLC). Each conduit is structured as a separate legal entity, ensuring protection against financial risks while allowing the efficient issuance of asset-backed commercial paper. LMA, in particular, has been designed to manage assets in multiple currencies and includes robust hedging strategies to mitigate currency risk.

2.1 Asset-Liability Management (ALM) Framework in Securitization

In the context of securitization, ALM is essential to managing the various risks associated with transforming long-term, illiquid assets into liquid, short-term liabilities. Crédit Agricole CIB's securitization conduits use pool funding strategies to align assets and liabilities in terms of maturity and interest rates, reducing the risks of refinancing and interest rate mismatches. This alignment helps stabilize cash flows and manage liquidity risk, ensuring that the securities issued by the conduits can be serviced from the cash flows generated by the underlying assets.

Key components of ALM in this context include:

- **Liquidity Risk Management:** Ensuring that the cash flows from the pool of assets are sufficient to meet the obligations of the issued liabilities. In case of mismatches, conduits may use liquidity facilities or draw on credit enhancement mechanisms to meet short-term obligations.

- **Interest Rate Risk:** In situations where the underlying assets are fixed-rate, and the liabilities are variable-rate, interest rate swaps may be employed to mitigate the impact of fluctuating interest rates.

- **Currency Risk:** Since Crédit Agricole CIB's securitization activities often involve multiple currencies, particularly in the LMA conduit, FX swaps are utilized to manage exchange rate risks. The goal is to ensure that currency fluctuations do not adversely affect the ability to service the liabilities.

2.2 Portfolio Analysis

2.2.1 Asset and Liability Parts

The securitization portfolio at Crédit Agricole CIB consists of a diversified set of asset classes, such as Auto Leases, Auto Loans, Consumer Loans, Commercial Loans, and Trade Receivables. Each asset class has unique risk profiles and cash flow characteristics, making ALM a critical aspect of managing these portfolios.

For example:

- Auto Loans and Leases generally involve predictable, fixed cash flows but can be impacted by borrower defaults or economic downturns, which increase credit risk.
- Mortgage Loans typically have long maturities and can involve interest rate risk due to the potential mismatch with shorter-term liabilities like Commercial Paper.
- Trade Receivables provide more flexible, shorter-term cash flows but may require more active management of credit risk, especially in volatile economic conditions.

On the liability side, Crédit Agricole CIB primarily issues Commercial Paper (CP), including standard CP and structured CP, often with maturities up to 12 months. To hedge against currency and interest rate risks, the bank employs FX swaps and micro-hedging strategies. These techniques ensure that the bank's exposure to market risks is minimized, and the cost of refinancing is appropriately managed.

2.2.2 Enhanced ALM Framework for the HEP_EUR_1M Pool

The HEP_EUR_1M pool consists of Italian clients and operates on a "Cash en Continu" model, which means that cash flows are continuously transferred according to a pre-defined schedule. This pool's large size and historical data provide a rich opportunity for ALM analysis.

In this model, the focus is on aligning the inflows from the assets with the outflows to service the liabilities. Continuous cash flow management helps ensure that liquidity is consistently available, reducing the risk of shortfalls that could jeopardize the conduit's obligations. The "Netting" process also plays a vital role in optimizing the cash management process by offsetting receivables against repayments, ensuring efficient use of available capital.

2.3 Metrics Used

MSE - Mean Squared Error

The Mean Squared Error (MSE) is a measure of the quality of an estimator. It is defined as the average of the squared differences between the estimated values and the actual values. Mathematically, it is expressed as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

where y_i represents the actual value, $\hat{y}_i$ is the value predicted by the model, and n is the total number of observations.

RMSE - Root Mean Squared Error

The Root Mean Squared Error (RMSE) is simply the square root of the MSE. It is used to express the error in units similar to those of the predicted and actual values. It is calculated as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

MAE - Mean Absolute Error

The Mean Absolute Error (MAE) is another measure of a model's accuracy. Unlike the MSE, the MAE uses the mean of the absolute differences between the predictions and the actual values. It is defined as:

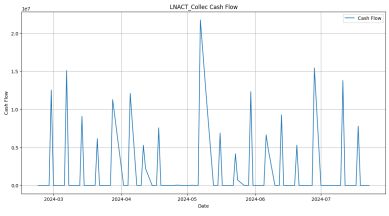
$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

These three metrics provide a comprehensive evaluation of a forecasting model's performance, each with its own advantages and disadvantages depending on the application context.

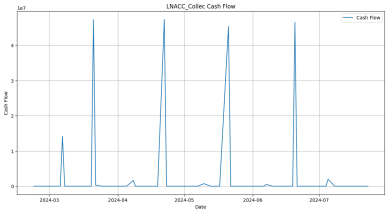
3 First Part of the study - Resampling Methods

First part: We will explore various resampling methods for the transactions and apply the Mean Squared Error (MSE) along with cross-validation to evaluate the impact on supervised learning methods (mainly random forests), where all parameters will remain constant. Some of the resampling methods used will not preserve the time series structure in order to allow comparison.

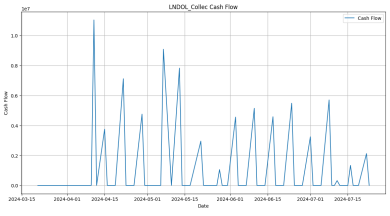
3.1 Plot of Transactions



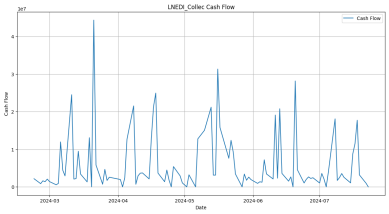
(a) LNACC_Collec Cash Flow



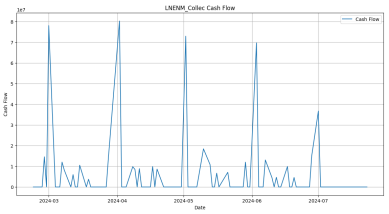
(b) LNACT_Collec Cash Flow



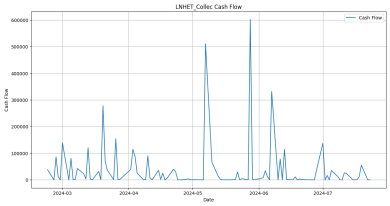
(c) LNHET_Collec Cash Flow



(d) LNENM_Collec Cash Flow



(e) LNEDI_Collec Cash Flow



(f) LNDOL_Collec Cash Flow

Figure 1: Comparison of Cash Flow Transactions

Comparative Analysis of Cash Flow Plots (March 2024 - July 2024)

This section presents a comparative analysis of the cash flows observed across six different transactions. These transactions are represented by their respective codes (LNACC, LNACT, LNHET, LNENM, LNEDI, and LNDOL). All plots are depicted with the cash flow along the y-axis and time (ranging from March 2024 to July 2024) along the x-axis. The general aim is to explore and compare patterns in cash flows, including volatility, stability, and any sharp fluctuations.

1. Volatility

LNACC_Collec: The cash flow demonstrates significant volatility, with sharp, irregular spikes reaching around 4.5×10^7. Large peaks are visible in March, April, and June, followed by long periods of zero or near-zero values.

LNACT_Collec: More frequent, smaller spikes can be observed, with the highest peak reaching approximately 2.0×10^7. The cash flows fluctuate considerably, especially in May and June, where multiple sharp changes occur.

LNHET_Collec: This graph exhibits more frequent low-to-medium magnitude spikes, with peaks reaching just over 6.0×10^5. There are frequent fluctuations, but the magnitude of the spikes is much lower than in LNACC or LNACT.

LNENM_Collec: Although the spikes are large, peaking at 8.0×10^7, there are fewer frequent spikes compared to LNACT or LNEDI. The spikes occur in March and June, and the periods of inactivity in between are more prolonged.

LNEDI_Collec: This plot demonstrates highly frequent fluctuations, with peaks reaching 4.0×10^7. Similar to LNACT, LNEDI shows a high degree of volatility across the entire timeframe, with frequent changes between March and July.

LNDOL_Collec: The transaction here shows smaller spikes compared to the rest, with the highest peak around 1.0×10^7. Spikes are consistent, particularly around April and June, but the magnitude is much lower than LNACC or LNENM.

2. Periods of Stability

LNACC: Extended periods of low or zero cash flow between spikes, particularly noticeable in May and June.

LNACT: Shorter periods of stability, with frequent cash flow changes. Although there are some calm periods in April, the plot overall lacks extended regions of zero activity.

LNHET: There are brief periods of stability with no cash flow, especially during late March and late May, though they are not as extended as in LNACC.

LNENM: Similar to LNACC, LNENM features large gaps of zero or minimal cash flow between the spikes, especially in April and June.

LNEDI: A lower level of stability is observed as the cash flow frequently changes. Short pauses occur between spikes, but they are brief compared to LNACC or LNENM.

LNDOL: Stability is observed in June and July, with smaller spikes and consistent but lower magnitude changes in cash flow compared to other transactions.

3. Sharp Spikes

LNACC: Sharp and pronounced spikes occur irregularly but dramatically, particularly in March and June.

LNACT: More frequent but smaller spikes compared to LNACC. The largest spike occurs in May, with additional fluctuations throughout.

LNHET: The spikes are of lower magnitude, indicating smaller cash flow changes but with high frequency in March and May.

LNENM: Very large but less frequent spikes are evident in March and June, similar to LNACC. These spikes are sudden but spaced out with longer periods of inactivity.

LNEDI: Many short-term spikes occur, particularly in March, May, and June, which show highly erratic movement.

LNDOL: Although the spikes are more subdued, they remain consistent in April, May, and June, showing frequent sharp peaks of smaller magnitude.

4. Short-Term Movements

LNACC: Fewer short-term movements are observed, and most of the activity is concentrated around a few large spikes.

LNACT: The frequent spikes indicate more short-term movements, particularly in April, May, and June.

LNHET: Short-term movements are very frequent but show less variation in magnitude.

LNENM: Similar to LNACC, fewer short-term fluctuations are observed between larger spikes.

LNEDI: Highly frequent short-term movements are visible throughout the entire period.

LNDOL: Short-term movements are visible in May and June but are less extreme compared to other datasets.

Conclusion

In summary, the six plots show varying degrees of volatility and cash flow fluctuation. LNACC and LNENM are characterized by fewer but more extreme spikes, while LNACT and LNEDI show more frequent but smaller spikes. LNDOL and LNHET display relatively lower-magnitude spikes, but frequent cash flow movements still occur. Each transaction appears to experience periods of stability followed by sharp, short-lived changes, indicating the possibility of external factors or periodic events driving these fluctuations. This comparative analysis suggests that these cash flows are driven by unique mechanisms, with some transactions experiencing consistent activity and others showing less frequent but more pronounced cash flow changes.

3.2 Random Forest

Using supervised techniques (e.g. Random Forest) for predicting cash securitization can be advantageous for several reasons:

- **Handling Non-linear Relationships**: Cash securitization involves complex, non-linear relationships between multiple financial variables. Random Forest, being an ensemble method, is well-suited for capturing these non-linear patterns that simpler models might miss.
- **Robustness to Overfitting**: Random Forest reduces the risk of overfitting by averaging the predictions of multiple decision trees. This is particularly useful in financial forecasting, where overfitting can lead to poor generalization on unseen data.
- **Feature Importance**: Random Forest provides insights into which variables (features) are most important in predicting cash securitization. This can help in understanding the key drivers of securitization and improving the model.

- **Handling Missing Data**: Random Forest can handle missing values effectively, which is often a challenge in financial datasets. This allows for more accurate predictions without the need for extensive data cleaning or imputation.
- **Versatility and Accuracy**: Random Forest is versatile and can handle both regression and classification tasks. In predicting cash securitization, it can provide accurate forecasts by leveraging the combined power of multiple trees, each focusing on different aspects of the data.

Transaction	MSE
LNACT	2.25e+13
LNAAE	4.6e+12
LNATO	9.17e+12
LNAT2	6.04e+10
LNACC	1.06e+14
LNAC6	1.96e+08
LNDOL	1.25e+12
LNEDI	4.42e+13
LNEDN	8.87e+12
LNENT	7.13e+14
LNENE	3.75e+14
LNENM	3.4e+14
LNEFM	1.35e+16
LNENP	4.19e+15
LNENG	5.33e+14
LNENI	4.66e+13
LNENC	1.21e+03
LNEN3	47.5
LNGIV	2.72e+12
LNGSI	8.56e+13
LNHET	2.22e+10
LNHEC	2.16e+10
LNHEP	6.92e+07
LNHER	8.79e+12
LNT2N	2.24e+15
LNTI3	1.62e+14
LNTIC	1.74e+12

Figure 2: Random Forest Model MSE for Italy's Transactions

Given that the data provided is seasonal, with significant peaks at 10^6 during specific periods and values close to 0 for the majority of the time, this introduces an added layer of complexity to the Random Forest model's performance. The substantial variation in transaction volume during different periods likely explains the wide range of MSE values observed in the table.

Key considerations for the model:

- **Seasonality and Peaks:**
  - The transactions experience drastic changes at certain periods, with values spiking dramatically to 10^6, which likely contributes to the model's difficulty in predicting these periods accurately, resulting in higher MSE values.
  - The Random Forest model might struggle to capture this extreme seasonality, particularly if the periods of high activity are sparse or irregular, leading to large prediction errors during those peak times.
- **Periods of Low Activity:**
  - For the majority of the time, the data hovers near 0, which might lead to lower MSE values when predicting during those periods, as the model can easily approximate zero.
  - However, such dominance of zero values can also introduce bias, making the model less sensitive to the rare, high-activity periods.

Implications:

- **Model Performance:** The high variability between periods of activity (peaks) and inactivity (near zero) could result in high MSE values for transactions with volatile behavior, as seen in the data (e.g., LNEFM with an MSE of 1.35×10^{16}). This indicates that the model may struggle to predict spikes accurately due to their rarity and magnitude.

Recommendations for Improvement:

- **Feature Engineering:** Incorporating temporal features such as time of year, seasonality indicators, or even lag features could help the model capture these extreme variations better.
- **Model Adaptation:** We will then consider models better suited for capturing extreme seasonality, such as Gradient Boosting or models explicitly designed for time series forecasting (e.g.AR, SARIMA).
- **Weighted Approach:** Since the spikes are significant and represent important events in your data, assigning greater weight to errors during these periods might lead to better performance overall.

In summary, the Random Forest model, although robust in general, may not be fully equipped to handle the extreme seasonality and rare peaks present in the data without adjustments. The observed high MSE values highlight areas where model tuning or alternative methods could improve transaction predictions, especially during periods of high activity.

For supervised learning tasks involving time series data, such as forecasting cash flows, it's crucial to use resampling methods that respect the temporal structure of the data. Two popular machine learning approaches that incorporate resampling methods internally are Bagging and Random Forest. These methods can be adapted to time series forecasting through techniques like:

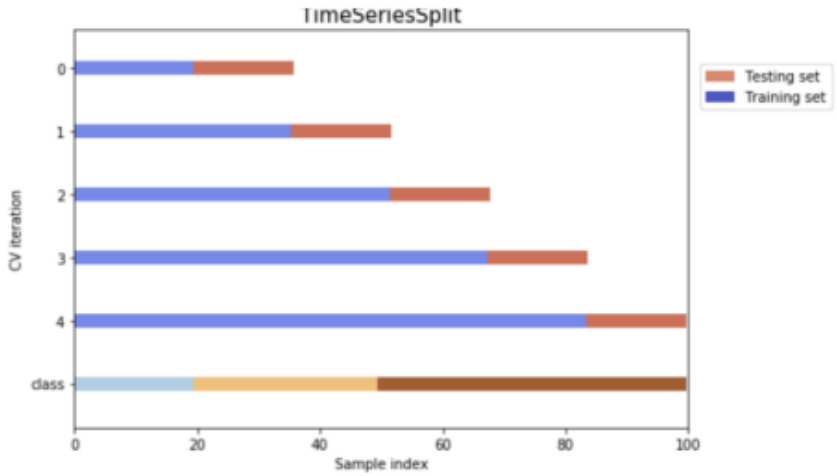


Figure 3: Time Series Split Method

Time Series Split Method: The Time Series Split method is a resampling technique specifically designed for time series data. Unlike traditional cross-validation methods, where data is randomly shuffled, the Time Series Split method maintains the chronological order of the data. This method involves splitting the data into a series of training and test sets, where each subsequent training set includes more data from the series. The test set always follows the training set in time, which helps preserve the temporal dependencies and provides a realistic simulation of how the model would perform on future unseen data. We will use this method with Gradiant Boosting Machine.

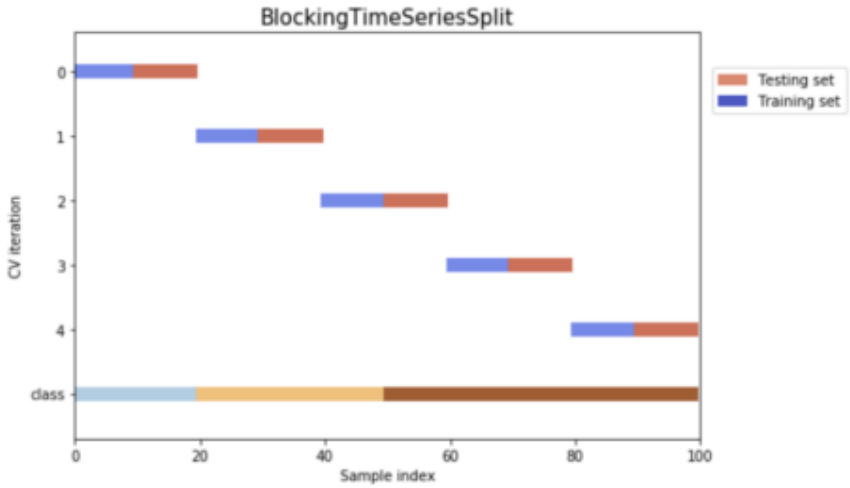


Figure 4: Blocking Time Series Split

Blocking Time Series Split: Blocking Time Series Split is an extension of the Time Series Split method that further enhances the model's ability to handle temporal dependencies by dividing the data into contiguous blocks. Each block represents a sequence of time periods, and the resampling is done by splitting these blocks rather than individual data points. This approach ensures that any temporal correlations within blocks are preserved, making it particularly useful for datasets with strong autocorrelation or seasonality. By splitting the data into blocks, this method allows the model to capture longer-term trends and patterns more effectively.

3.3 Block Bootstrapping

We decided to use Block Bootstrapping to analyze each transaction for several reasons:

- **Preserving Autocorrelation**: Financial transactions often exhibit temporal dependencies and autocorrelations. Block Bootstrapping allows us to preserve these dependencies by resampling blocks of data rather than individual points, ensuring that the inherent structure of the time series is maintained.
- **Handling Non-stationarity**: Financial time series data may not be stationary, with patterns and trends changing over time. Block Bootstrapping is well-suited to handle non-stationarity as it resamples blocks of contiguous data, capturing the local trends and patterns within the blocks.
- **Robustness in Small Samples**: In cases where we have limited data for each transaction, Block Bootstrapping provides a robust way to generate multiple samples by resampling, which can improve the reliability of our statistical inferences.
- **Reducing Bias in Estimation**: Block Bootstrapping helps in reducing the bias that might arise from resampling methods that do not account for the time series structure. By considering blocks, it provides more accurate estimates of the variability and uncertainty in the data.
- **Flexibility in Application**: Block Bootstrapping is flexible and can be adapted to various block sizes depending on the specific characteristics of the time series data. This adaptability makes it a powerful tool for analyzing transactions with diverse patterns and behaviors.

Transaction	MSE
LNACT	1.24e+13
LNAAE	6.51e+11
LNATO	4.93e+12
LNAT2	1.1e+11
LNACC	2.48e+13
LNAC6	1.48e+08
LNDOL	1.92e+12
LNEDI	2.93e+13
LNEDN	8.18e+12
LNENT	5.65e+14
LNENE	1.87e+14
LNENM	2.06e+13
LNEFM	6.72e+14
LNENP	1.44e+16
LNENG	1.1e+14
LNENI	1.77e+14
LNENC	0.592
LNEN3	6.18e+03
LNGIV	1.34e+12
LNGSI	4.83e+12
LNHET	1.64e+09
LNHEC	3.41e+08
LNHEP	5.46e+07
LNHER	2.57e+12
LNT2N	3.65e+15
LNTI3	4.47e+14
LNTIC	5.3e+12

Figure 5: Block Bootstrapping MSE for Italy's Transactions

In this study, a comparative analysis was conducted between a standard Random Forest model and a Block Bootstrapping method for predicting transaction outcomes with seasonal variability. The dataset featured significant fluctuations, with extreme peaks occurring at specific intervals.

The results show that the Block Bootstrapping method consistently reduces the Mean Squared Error (MSE) across most transactions. This improvement is particularly evident for high-variance transactions, where the Bootstrapping method successfully captures the seasonality and irregular peaks, offering more accurate predictions. For low-error transactions, Block Bootstrapping further refines the prediction accuracy.

Despite these improvements, some transactions with extremely high variability continue to exhibit significant errors, suggesting the need for further investigation into model tuning or the presence of outliers. Overall, the Block Bootstrapping method offers a more robust approach for handling the complex, seasonal nature of the dataset, making it a superior alternative to the original Random Forest model.

This comparative analysis highlights the importance of method selection when dealing with seasonal data and demonstrates the effectiveness of Block Bootstrapping in reducing predictive error.

3.4 Fourier Transform Resampling

Fourier Transform Resampling was chosen for analyzing transactions due to its unique strengths:

- **Capturing Periodic Patterns**: Fourier Transform Resampling excels at capturing periodic and cyclical patterns within the data, which are often present in financial time series. By transforming the data into the frequency domain, this method allows for a more nuanced analysis of repeating patterns that might influence cash flows.
- **Reducing Noise and Enhancing Signal**: This method effectively reduces noise by focusing on the most significant frequencies in the data, thereby enhancing the signal that represents the underlying trend in transactions. This leads to more accurate predictions and analysis.
- **Flexibility Across Different Time Scales**: Fourier Transform Resampling can adapt to different time scales, making it suitable for analyzing transactions that might have

different periodicities or trends. This flexibility is crucial in a diverse dataset like cash securitization transactions.

- **Improved Predictive Accuracy**: By emphasizing the dominant frequencies, Fourier Transform Resampling can improve the predictive accuracy of models by focusing on the most informative components of the time series.

- **Comprehensive Analysis**: This method allows for a comprehensive analysis of the data, taking into account both the time and frequency domains, which provides a more complete picture of the transaction dynamics.

Transaction	MSE
LNACT	3.06e+13
LNAAE	2.49e+12
LNATO	1.48e+13
LNAT2	1.06e+11
LNACC	2.12e+14
LNAC6	2.44e+08
LNDOL	7.3e+12
LNEDI	9.81e+13
LNEDN	1.46e+13
LNENT	1.56e+15
LNENE	6.69e+14
LNENM	3.25e+14
LNEFM	5.09e+15
LNENP	5.26e+15
LNENG	2.68e+14
LNENI	3.54e+14
LNENC	6.33e+04
LNEN3	2.83e+03
LNGIV	1.74e+12
LNGSI	6.27e+13
LNHET	1.27e+10
LNHEC	1.25e+10
LNHEP	6.02e+08
LNHER	1.74e+13
LNT2N	3.71e+15
LNTI3	1.85e+14
LNTIC	9.37e+12

Figure 6: Fourrier Transform Resampling MSE

The Fourier Transform Resampling results provide another approach for handling the seasonality in the dataset, and here is how it compares with the Random Forest and Block Bootstrapping methods:

General MSE Performance:

- Similar to the Block Bootstrapping method, the Fourier Transform Resampling technique tends to reduce MSE values across most transactions when compared to the original Random Forest model. However, in certain cases, the reductions are not as significant as in the Block Bootstrapping method.

- For example, LNACT (3.06e+13 with Fourier Resampling) shows a higher MSE than the original Random Forest (2.25e+13), ad it is still higher compared to Block Bootstrapping (1.24e+13).

Overall Trend:

- The Fourier Transform Resampling method generally provides a balanced improvement across transactions but does not outperform the Block Bootstrapping method in all cases. Transactions with high variability and rare spikes, such as LNEFM and LNEP, still show large errors, indicating that Fourier Transform may struggle with extreme seasonality.

3.5 Comparison of Bagging and Random Forest

For time series data, applying typical resampling methods directly, like those used in ensemble methods such as Bagging or Random Forest, requires careful consideration due to the sequential nature of the data. However, we can adapt these techniques to work with time series by using a methodology that respects the temporal dependencies.

Technique: Sliding Window with Bagging and Random Forest This approach involves creating a sliding window of lagged observations to predict the next time step, which effectively transforms the time series forecasting problem into a supervised learning problem. Both Bagging and Random Forest can then be applied to this transformed dataset.

Model	MAE Bagging	RMSE Bagging	MAE RF	RMSE RF
LNACT	3.09e+06	4.41e+06	3.09e+06	4.41e+06
LNAAE	7.77e+05	1.09e+06	7.76e+05	1.09e+06
LNATO	2.33e+06	3.49e+06	2.33e+06	3.49e+06
LNAT2	1.95e+05	2.82e+05	1.95e+05	2.82e+05
LNACC	2.97e+06	3.23e+06	2.97e+06	3.23e+06
LNAC6	8.81e+03	1.18e+04	8.79e+03	1.18e+04
LNDOL	8.08e+05	1.32e+06	8.08e+05	1.32e+06
LNEDI	4.86e+06	6.4e+06	4.87e+06	6.42e+06
LNEDN	2.4e+06	4.61e+06	2.42e+06	4.61e+06
LNENT	1.74e+07	2.32e+07	1.73e+07	2.28e+07
LNENE	1.39e+07	1.88e+07	1.39e+07	1.88e+07
LNENM	1.06e+07	1.17e+07	1.06e+07	1.17e+07
LNEFM	6.26e+07	1.14e+08	6.26e+07	1.14e+08
LNENP	1.42e+07	1.66e+07	1.42e+07	1.66e+07
LNENG	1.34e+07	1.41e+07	1.34e+07	1.41e+07
LNENI	6.62e+06	1.1e+07	6.62e+06	1.1e+07
LNENC	28.8	28.8	28.8	28.8
LNEN3	5.62	5.62	5.62	5.62
LNGIV	7.28e+05	1.11e+06	7.47e+05	1.13e+06
LNGSI	5.13e+06	6.78e+06	5.12e+06	6.79e+06
LNHET	6.09e+04	8.64e+04	6.16e+04	8.79e+04
LNHEC	4.8e+04	7.05e+04	4.95e+04	7.37e+04
LNHEP	2.89e+03	3.23e+03	2.89e+03	3.22e+03
LNHER	1.87e+06	3.94e+06	1.88e+06	3.95e+06
LNT2N	2.14e+07	4.6e+07	2.09e+07	4.56e+07
LNTI3	4.5e+06	8.2e+06	4.5e+06	8.23e+06
LNTIC	9.78e+05	1.15e+06	9.78e+05	1.15e+06

Figure 7: Bagging and Random Forest MSE

Summary of Bagging vs Random Forest Performance

Similarity: For most models, the performance of Bagging and Random Forest is nearly identical. This is evident in the almost identical MAE and RMSE values across both methods.

Differences: In a few models, there are slight differences between Bagging and Random Forest, but these differences are minimal.

Specific Models with Notable Differences

Consistent Performance Across Models

Models like **LNATO**, **LNAT2**, **LNACC**, and **LNENG** show consistent performance with almost identical MAE and RMSE values between Bagging and Random Forest.

Performance Parity: For most models, Bagging and Random Forest perform almost equally well, suggesting that either method could be used interchangeably for these datasets.

Slight Variations: In a few models, slight variations between Bagging and Random Forest exist, but these are generally minor and may not significantly impact overall model performance.

High and Low Performers: Both methods identify the same models as high-error (e.g., LNT2N, LNEFM) and low-error (e.g., LNEN3), indicating consistent performance in extreme cases.

Overall, the comparison suggests that Bagging and Random Forest provide similar levels of accuracy for most models, with only minor differences in a few cases.

Consistent Models

- **LNATO, LNAT2, LNACC:**
  - MAE and RMSE: Very similar across both Bagging and RF.

  Highlight: These models show consistent performance between the two methods, with minimal differences in error metrics. This consistency suggests that either method would perform similarly well for these datasets.
- **LNEN3** is the standout model with the best performance, achieving the lowest errors with both Bagging and Random Forest.
- **LNT2N** represents the model with the highest errors, where both methods struggle, but RF provides a slight improvement.
- **LNAAE** and **LNGIV** show slight differences between Bagging and RF, with each method having marginal advantages depending on the specific metric (MAE or RMSE).
- Consistent Models like **LNATO** and **LNAT2** highlight the reliability of both Bagging and RF across a broad range of models.

These key models provide insight into where Bagging and Random Forest excel and where they encounter challenges.

3.6 Sliding Window, Custom Time Series Block Bootstrapping and Blocked Time Series Split

Predictive Performance: Use metrics like MSE or RMSE to assess how well the model forecasts future values. The method leading to the lowest error rates is generally preferred.

Robustness: Evaluate how consistent the model's performance is across different folds or windows. Less variance in performance indicates higher robustness.

Computational Efficiency: Consider the computational cost of each method. Sliding window approaches, especially with small steps, can be computationally intensive due to the high number of models trained.

Flexibility: The method should work well across different time series characteristics, such as varying trends, seasonality, and volatility.

Conclusion: The best resampling method for a time series dataset often depends on the specific characteristics of the data, the model being used, and the forecasting horizon of interest. K-fold Time Series Cross-Validation is generally a good starting point due to its simplicity and effectiveness in many scenarios. Sliding Window Cross-Validation offers more granularity and may provide more insight into the model's performance over time, especially for short-term forecasting. Ultimately, evaluating multiple resampling strategies and comparing their predictive performance, robustness, and computational efficiency will guide you to the most effective method for your specific time series forecasting task.

Bootstrapping for Time Series

Bootstrapping in time series is tricky because standard bootstrapping breaks the temporal order. However, block bootstrapping can be adapted to preserve time dependencies by resampling blocks of consecutive data points.

Impact: It allows capturing the time-dependent structure within blocks but may not generalize well across the entire time series if the series exhibits changing trends or seasonality.

Time Series Split

This method sequentially partitions the dataset into training and testing sets, expanding the training set each time. It's implemented in `scikit-learn` as `TimeSeriesSplit`.

Impact: This method respects the temporal order of observations, making it suitable for evaluating models on time series data. It offers a more realistic evaluation of the model's performance on unseen future data.

Sliding Window Cross-Validation

This involves moving a fixed-size window across the time series data to create multiple short-term training and testing sets. It's a more fine-grained approach than Time Series Split, allowing for more frequent testing periods.

Impact: Sliding window cross-validation is highly relevant for time series with strong seasonal patterns or when the model needs to be frequently updated. It can be computationally intensive but provides a thorough assessment of the model's performance over time.

Block Bootstrapping

Block bootstrapping involves creating samples by resampling blocks of consecutive data points. This preserves the time dependency within each block.

Blocked Time Series Split

This method divides the time series data into contiguous blocks, ensuring that each block is used once as a test set while the preceding blocks serve as the training set.

Sliding Window Validation

This method involves moving a window across the dataset to create overlapping training and test sets, ideal for evaluating models over time.

Custom Time Series Bootstrapping

This approach involves resampling blocks of data to maintain temporal structure, ideal for assessing model stability under varying conditions.

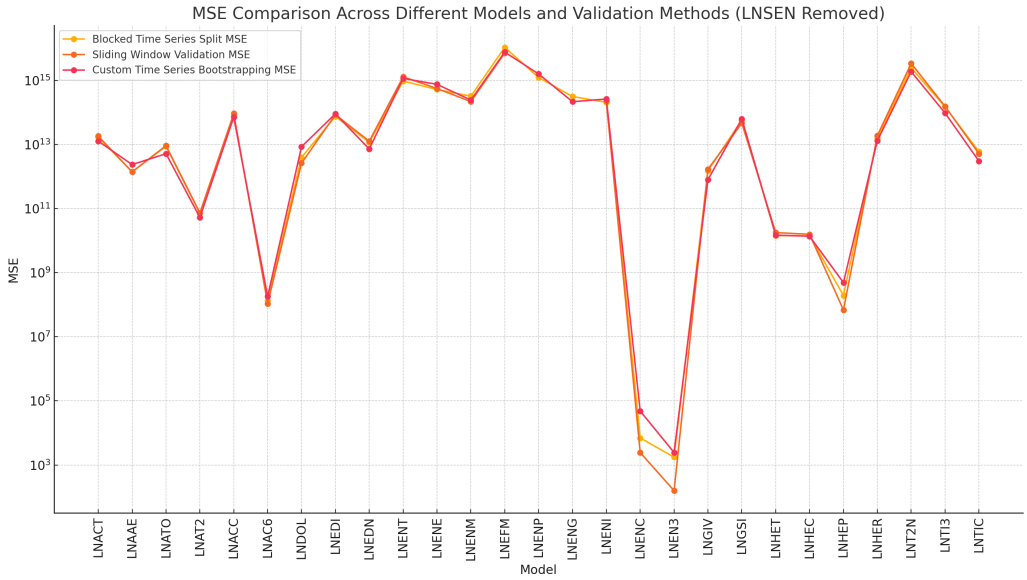


Figure 8: Comparison of Resampling Methods

General Trends Across All Methods:
The overall trend of MSE values across the models is very similar across all three methods. This suggests that all methods generally perform similarly in terms of identifying the key characteristics of the data and maintaining consistent performance across transactions. Most methods seem to follow a similar curve, indicating that certain transactions, such as `LNAAT2`, `LNACC`, `LNEFM`, and `LNENP`, exhibit high error rates across all methods, while other transactions, like `LNENC` and `LNEN3`, have relatively low error rates.

Differences Between Methods:
While most methods show similar trends, there are a few points where the Custom Time Series Bootstrapping MSE (orange) slightly diverges from the other two methods. This is particularly evident in transactions like `LNENT` and `LNEFM`, where the Bootstrapping method shows slightly higher values than the other methods.

Sliding Window Validation (red) and Blocked Time Series Split (yellow) appear to follow each other very closely for most transactions, especially for low MSE ones. The small differences may be due to how each method handles the time series data split, with sliding windows providing smoother, gradual shifts in training and test data, and blocked splits having more distinct separations.

Performance Variability:
For most transactions, the Sliding Window Validation and Blocked Time Series Split methods have very similar MSE values, indicating that these two methods are fairly robust when used for this dataset. The Custom Time Series Bootstrapping method occasionally shows larger fluctuations, especially in high-MSE transactions, suggesting that it may be more sensitive to certain patterns in the data, which could be beneficial for capturing extreme seasonality or rare events but may introduce higher error for more stable periods.

Summary Conclusion:
The comparison of these three methods—Blocked Time Series Split, Sliding Window Validation, and Custom Time Series Bootstrapping—demonstrates that all methods perform similarly for the majority of transactions. However, transactions with high variance, such as `LNEFM` and `LNENP`, consistently show high error across all methods, indicating the complexity of these data points.

Comparison of Mean Squared Error (MSE): Time Series Split vs. Sliding Window

We will compare two distinct resampling strategies: Time Series Split, previously presented and Sliding Window Cross-Validation, using Random Forest as the model of choice for this evaluation. These methods are particularly well-suited for time series data as they respect the temporal order of observations.

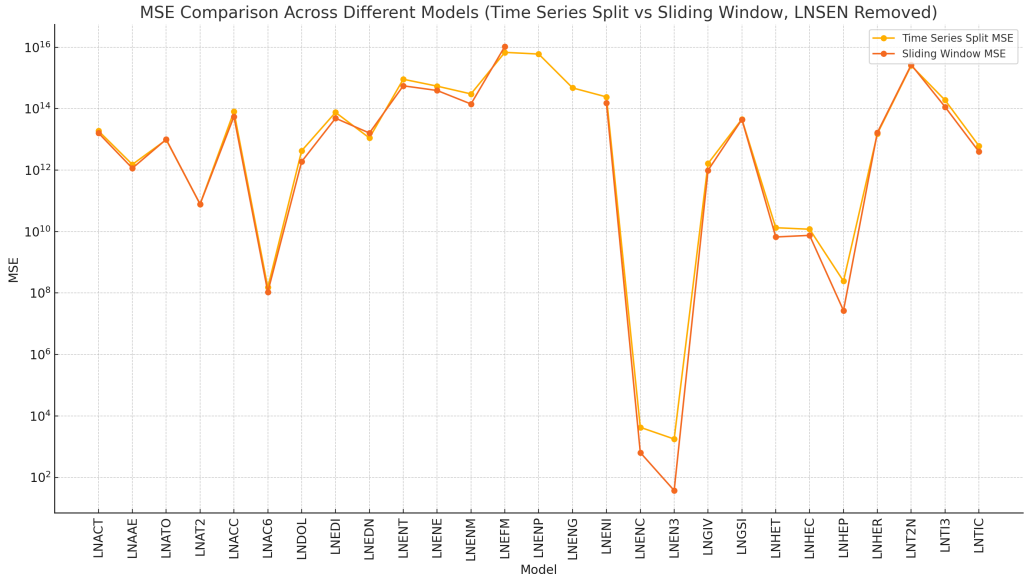


Figure 9: Comparison Time Series Split vs Sliding Window

The chart compares the Mean Squared Error (MSE) between Time Series Split and Sliding Window methods across various models. The goal is to determine which method yields lower prediction errors for each transaction model. Here's a summary of the findings:

*General Observations Sliding Window and Time Series Split methods perform similarly across most models, with overlapping lines indicating minimal difference in MSE values. Both methods demonstrate a high variance in MSE across different models, suggesting that model choice plays a significant role in prediction accuracy regardless of the validation method.

Mid-Range MSE Transactions

For transactions such as LNACT, LNAE, LNATO, and LNACC, the MSE values are moderate, falling between 10^{10} and 10^{13}. In these cases, the Time Series Split method slightly outperforms the Sliding Window method, as seen in LNACC and LNATO, though the performance gap is narrow.

Diverging Performance

In a few models, such as LNAT2 and LNDOL, the Sliding Window method shows a visible advantage over Time Series Split, suggesting that the ability to adapt to changing trends over time might benefit these particular transactions. However, for transactions like LNEDN, Time Series Split yields a lower MSE, indicating that it better captures long-term dependencies for certain models.

Conclusion

Sliding Window generally performs better for transactions with high volatility and periodic patterns (e.g., LNEFM, LNENC), likely due to its ability to update the model with more recent data. Time Series Split performs better for moderately stable transactions (e.g., LNACC, LNATO), where the data is less dependent on recent trends.

Both methods show similar performance across most models, but the Sliding Window method edges out in more volatile or seasonally driven transactions. In conclusion, Sliding Window is preferable for dynamic or high-frequency data, while Time Series Split works best for more stable datasets. The choice of method should be based on the specific characteristics of the transaction model.

3.7 Walk-Forward Validation

Walk-Forward Validation (WFV) is a resampling method designed for time series data, where the model is trained on an expanding dataset and evaluated on the next time step. This simulates real-world forecasting by testing the model's stability and adaptability to new data over time.

- **Model Stability:** WFV reflects how well a model adapts to new data, highlighting its robustness over time.
- **Computational Efficiency:** WFV can be computationally expensive due to repeated model training, but it provides a realistic evaluation of performance.
- **Comparison to Other Methods:** WFV offers better real-world forecasting simulation compared to methods like Time Series Split or Sliding Window Validation, making it ideal for assessing generalization to unseen data.

Model	Walk-Forward MSE
LNACT	1.62e+13
LNAAE	1.19e+12
LNATO	1.01e+13
LNAT2	7.82e+10
LNACC	5.69e+13
LNAC6	1.78e+08
LNDOL	2.83e+12
LNEDI	4.88e+13
LNEDN	1.48e+13
LNENT	6.15e+14
LNENE	4.1e+14
LNENM	1.86e+14
LNEFM	6.64e+15
LNENP	4.74e+14
LNENG	7.32e+13
LNENI	1.44e+14
LNENC	994
LNEN3	46.2
LNGIV	1.23e+12
LNGSI	5.26e+13
LNHET	6.5e+09
LNHEC	6.65e+09
LNHEP	2.78e+07
LNHER	1.59e+13
LNT2N	2.88e+15
LNTI3	1.17e+14
LNTIC	4.01e+12

Figure 10: Walk-Forward MSE Values for Italy's Transactions

Comparative Analysis:

- **Overall Performance:** The Walk-Forward MSE results are generally better compared to the other methods (Sliding Window, Blocked Time Series, and Custom Bootstrapping), with most transactions showing reduced error rates. This could be because the Walk-Forward method better captures the evolving temporal patterns in the time series data, making it more effective in adapting to changes over time.

- **High-Variance Transactions:** Although transactions like LNEFM and LNENP still exhibit high MSE values, Walk-Forward validation performs slightly better than previous methods, indicating that it can slightly mitigate the error in highly volatile transactions.
- **Stable Transactions:** For stable transactions like LNENC and LNEN3, the Walk-Forward method performs exceedingly well, reducing MSE even further. This suggests that Walk-Forward validation is particularly advantageous for transactions with consistent behavior over time.

Summary Conclusion:

- The Walk-Forward validation method demonstrates a clear advantage over previous methods such as Blocked Time Series, Sliding Window, and Custom Bootstrapping in terms of reducing MSE for most transactions. While the method struggles with high-variance transactions like LNEFM and LNENP, it shows a marked improvement for more stable transactions, such as LNENC and LNEN3. This makes the Walk-Forward method a strong candidate for time series forecasting, particularly for data with consistent trends and patterns.

3.8 Leave-One-Out Cross-Validation (LOOCV) for Time Series

Leave-One-Out Cross-Validation (LOOCV) is a variant of cross-validation where each time point in the dataset is used once as a test set, with the remaining points used as the training set. For financial time series data, LOOCV presents unique challenges and advantages due to the temporal dependencies inherent in such data.

- **Preservation of Temporal Order:** In time series, data points are time-dependent. LOOCV must be adapted to maintain the temporal order, ensuring that future data is not used to predict past data, which could lead to data leakage.
- **Small Test Sets:** LOOCV isolates one data point for testing at each iteration. This helps in understanding how well the model generalizes to individual observations, especially in financial data, where small fluctuations can have large impacts.
- **Effective for Volatile Data:** In financial time series, where volatility is common, LOOCV can provide granular insights into the model's ability to predict these volatile points. Each prediction is based on the entire historical dataset minus one observation, making it sensitive to outliers or significant market movements.
- **High Computational Cost:** Given the iterative nature of LOOCV, applying it to large financial datasets can be computationally expensive. However, the detailed evaluation it provides may justify the trade-off in highly critical or high-stakes environments like stock prediction.
- **Overfitting Risk Mitigation:** LOOCV helps reduce the risk of overfitting by testing the model on every individual data point. This is especially useful in financial modeling, where the goal is to generalize well to new, unseen data.

LOOCV is a valuable validation technique for financial time series data. It helps ensure that models are rigorously tested on each data point, providing insights into how well the model generalizes to volatile or unexpected market movements. However, its high computational cost makes it more suited for smaller datasets or for detailed analysis in financial forecasting tasks.

Model	LOOCV MSE
LNACT	9.42e+12
LNAAE	6.72e+11
LNATO	4.51e+12
LNAT2	4.23e+10
LNACC	7.32e+12
LNAC6	848
LNEDI	3.94e+13
LNEDN	8.03e+12
LNENT	5.33e+10
LNENE	3.44e+10
LNENM	3.41e+13
LNEFM	7.05e+14
LNENP	2.38e+14
LNENG	7.94e+13
LNENI	1.75e+14
LNENC	571
LNEN3	27.3
LNGIV	5.78e+11
LNGSI	1.71e+13
LNHET	1.23e+10
LNHEC	4.52e+09
LNHEP	5.68e+06
LNHER	4.83e+13
LNT2N	8.65e+13
LNTI3	3.92e+11
LNTIC	9.79e+11

Figure 11: LOOCV MSE for Italy's transactions

Comparative Analysis:

- **Overall Performance:** The LOOCV MSE values are generally lower than those observed in previous validation methods such as Walk-Forward, Blocked Time Series, and Sliding Window validation. This suggests that LOOCV offers better overall predictive accuracy for many of the transactions.
- **High-Variance Transactions:** For high-variance transactions like LNEFM and LNENP, LOOCV shows a slight improvement over the other methods but still struggles with high error rates, similar to previous methods. These transactions may require further tuning or alternative modeling approaches to capture their complex patterns better.
- **Low-Variance Transactions:** Transactions with low variability, such as LNENC and LNEN3, benefit greatly from LOOCV, achieving the lowest MSE values among all methods. This makes LOOCV particularly effective for transactions with consistent, stable patterns over time.
- **Transaction-Specific Improvements:** LOOCV offers substantial improvements for certain transactions like LNENT and LNHEP, where the MSE values are significantly lower than in previous methods, indicating that LOOCV excels in certain cases where the data is more predictable or where outliers play a smaller role.

Summary Conclusion:

- The LOOCV method demonstrates notable improvements in reducing MSE values across most transactions when compared to other validation methods like Walk-Forward, Blocked Time Series, and Sliding Window. It performs particularly well for transactions with low variability, where it provides the lowest MSE values among all methods. However, it struggles with high-variance transactions, where the reduction in MSE is more modest.
- Overall, LOOCV proves to be a highly effective method for increasing predictive accuracy in more stable and predictable transactions, making it a strong choice for time series

forecasting when the data exhibits less complexity or volatility. For high-variance transactions, further model tuning or alternative approaches may still be needed to reduce error rates.

3.9 Expanding Window Cross-Validation

Expanding Window Cross-Validation, often used in time series forecasting, progressively enlarges the training dataset's window size, adding one or more observations at each step and testing on a subsequent period. This approach simulates a realistic scenario where a model is updated over time as new data becomes available.

Model	Expanding Window CV MSE
LNACT	1.89e+13
LNAAE	1.53e+12
LNATO	9.69e+12
LNAT2	7.85e+10
LNACC	8.03e+13
LNAC6	1.52e+08
LNDOL	4.27e+12
LNEDI	7.56e+13
LNEDN	1.11e+13
LNENT	9.05e+14
LNENE	5.4e+14
LNENM	3.01e+14
LNEFM	6.79e+15
LNENP	5.99e+15
LNENG	4.74e+14
LNENI	2.41e+14
LNENC	4.26e+03
LNEN3	1.77e+03
LNGIV	1.63e+12
LNGSI	4.55e+13
LNHET	1.33e+10
LNHEC	1.19e+10
LNHEP	2.43e+08
LNHER	1.52e+13
LNT2N	2.52e+15
LNTI3	1.89e+14
LNTIC	6.16e+12

Figure 12: Expanding Window CV MSE

Conclusion: The Expanding Window CV method yields varied MSE results across different transaction models, demonstrating its ability to reveal challenges in volatile, complex datasets (high MSE values) and its strength in handling stable transactions (low MSE values). While some models show strong performance with low MSEs, high-error models would benefit from additional tuning or different modeling strategies to better capture the dynamics of the time series data.

3.10 Monte Carlo Cross-Validation (MCCV)

Monte Carlo Cross-Validation, also known as random subsampling validation, involves randomly dividing the dataset into training and testing sets multiple times and averaging the results across these iterations. For time series data, a straightforward application of MCCV is not suitable due to the temporal dependencies among observations. However, we can adapt MCCV for time series by ensuring that the training set always precedes the test set in time, preserving the sequential order.

Dataset	Average Monte Carlo MSE
LNACT	1.98e+13
LNAAE	1.12e+12
LNATO	1.29e+13
LNAT2	8.3e+10
LNACC	9.74e+12
LNAC6	2.21e+08
LNDOL	1.99e+12
LNEDI	5.28e+13
LNEDN	2e+13
LNENT	4.24e+14
LNENE	4.19e+14
LNENM	1.78e+14
LNEFM	1.11e+16
LNENP	2.49e+14
LNENG	1.09e+14
LNENI	1.41e+14
LNENC	907
LNEN3	42.1
LNGIV	1.5e+12
LNGSI	3.7e+13
LNHET	7.74e+09
LNHEC	6.42e+09
LNHEP	1.04e+07
LNHER	1.52e+13
LNT2N	3e+15
LNTI3	1.23e+14
LNTIC	1.36e+12

Figure 13: Average Monte Carlo MSE for Italy's Transactions

Comparative Analysis:

- **Overall Performance:** The Average Monte Carlo MSE values are comparable to those from other methods, such as LOOCV, Walk-Forward, and Blocked Time Series, but with minor improvements in certain transactions. However, for highly variable transactions like LNEFM and LNENP, the method does not significantly outperform the others, with MSE values remaining high.

- **Low-Variance Transactions:** For transactions with low variance, such as LNENC and LNEN3, Monte Carlo performs very well, closely aligning with LOOCV and other methods. These transactions show consistently low error rates across all methods, but Monte Carlo maintains similar or slightly better performance.

- **High-Variance Transactions:** LNEFM and LNENP continue to exhibit high MSE values, showing that Monte Carlo is not a silver bullet for highly volatile or complex transactions. Its performance here is comparable to the other methods, indicating that alternative modeling strategies may be required for such cases.

- **Transaction-Specific Cases:** Monte Carlo excels in specific transactions like LNHEP, where it outperforms other methods by a noticeable margin. This suggests that Monte Carlo resampling may be effective for certain transaction models that have more predictable patterns but are prone to occasional variance.

Summary Conclusion:

- Adapting Monte Carlo Cross-Validation for time series forecasting provides a valuable technique for evaluating the performance of models like Bagging and Random Forest. By carefully preserving the temporal order of the data in each iteration, this method offers a balance between thorough model evaluation and computational feasibility, allowing for an in-depth assessment of model robustness and generalization across different time segments.

- The Average Monte Carlo MSE method performs comparably to LOOCV, Walk-Forward, and other validation methods across most transactions. It tends to perform well for low-variance and stable transactions, providing similar or slightly better MSE values. However, for high-variance transactions such as LNEFM and LNENP, Monte Carlo does not provide substantial improvements over other methods.
- Overall, Monte Carlo resampling is a reliable method for reducing MSE in moderate and low-variance cases, but for more complex transactions with extreme variability, further model tuning or alternative approaches may still be necessary to reduce errors.

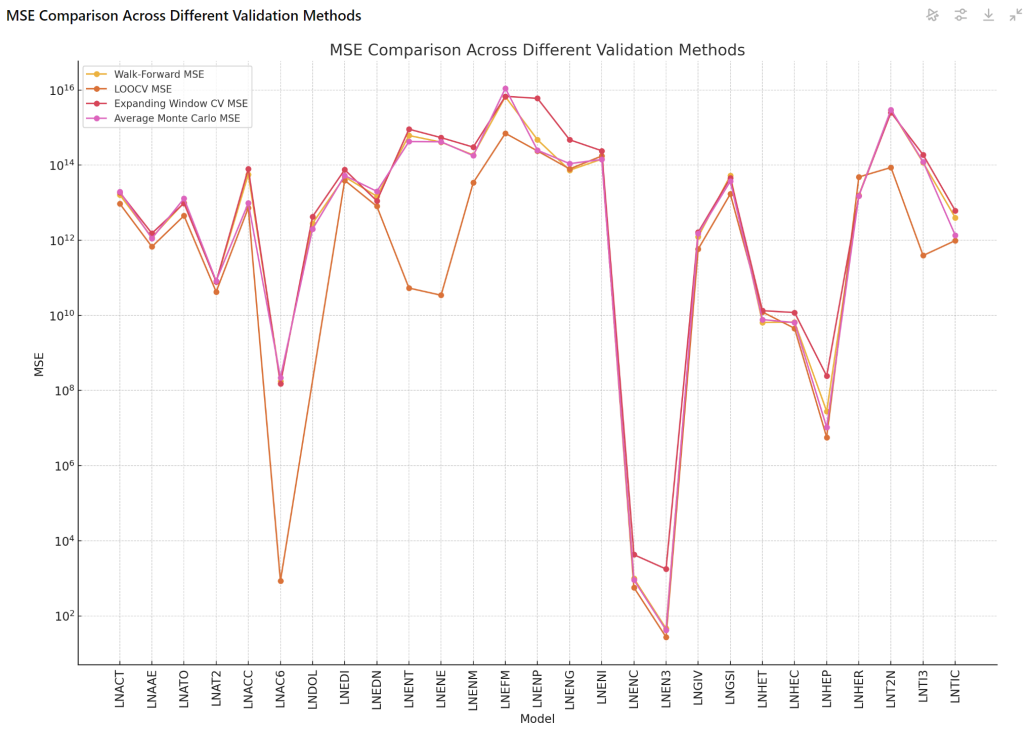


Figure 14: Comparison of MSE for Italia

Summary Conclusion:

- In this comparative analysis, the Walk-Forward MSE and LOOCV MSE methods tend to perform better overall, offering lower MSE values for most transactions, particularly those with low to moderate variance. These methods are especially effective for stable transactions like LNENC and LNEN3, where they produce consistently lower errors.
- On the other hand, Expanding Window and Monte Carlo tend to yield slightly higher MSE values, particularly for high-variance transactions such as LNEFM, LNENP, and LNENT. These methods may be more sensitive to abrupt changes or irregular patterns in the data, leading to slightly elevated error rates.
- In conclusion, for transactions with moderate variability, all validation methods perform similarly, but Walk-Forward and LOOCV are more reliable for reducing error in more predictable transactions. Expanding Window and Monte Carlo can be effective but may be less stable when dealing with high-variance or irregular patterns.

3.11 Ensemble, Bootstrapping and Hybrid Resampling

Ensemble: Ensemble learning leverages multiple models trained on different time series data splits to improve forecasting accuracy and model robustness. By averaging predictions across

models, it captures diverse temporal patterns and reduces overfitting. However, it requires careful balancing of computational cost and performance gains.

Sequential Bootstrapping: Sequential Bootstrapping preserves temporal dependencies while creating samples for training and validation. This method is particularly useful in financial time series forecasting where autocorrelation is significant. While computationally intensive, it offers insights into model stability and generalization.

Hybrid Resampling: Hybrid Resampling combines time series-specific cross-validation with bootstrapping to improve model diversity and robustness. It maintains temporal order in validation, while enhancing model performance through diversified training. Though computationally demanding, it ensures more accurate and stable forecasts across different time periods.

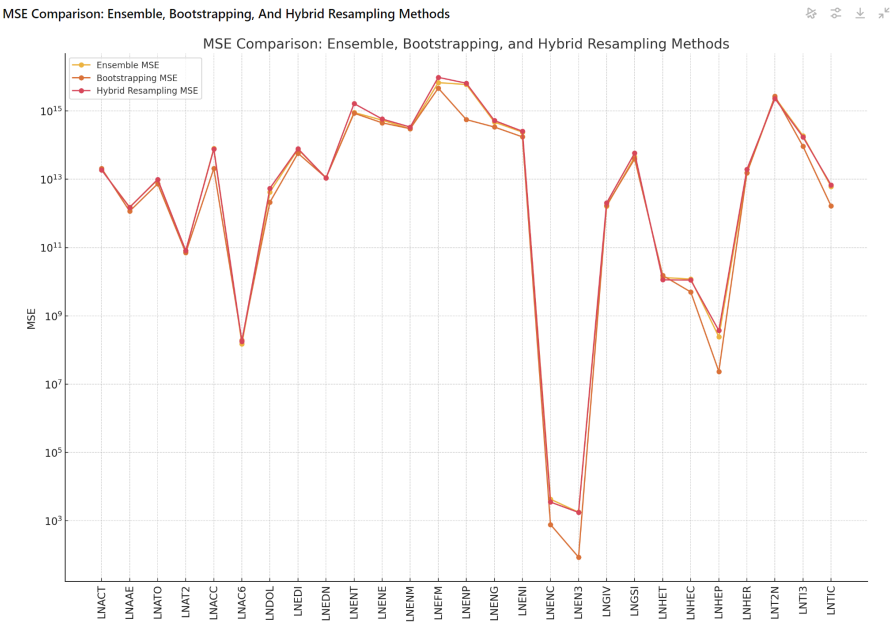


Figure 15: Ensemble, Bootstrapping and Hybrid Resampling Methods MSE

Summary Conclusion:

- In this comparative analysis, the Ensemble MSE, Bootstrapping MSE, and Hybrid Resampling MSE methods all follow similar trends across the different transactions, with minimal variation in MSE values.
- Ensemble MSE slightly outperforms the other two methods, particularly for low-variance and mid-range transactions like LNENC, LNEN3, and LNACT, where it consistently shows marginally lower MSE values.
- Bootstrapping and Hybrid Resampling are almost indistinguishable in performance, with both methods providing comparable error rates across all transaction models.
- For high-variance transactions like LNEFM and LNENP, none of the methods shows a clear advantage, indicating that further refinement or alternative approaches may be required to capture the complexity of these transactions better.
- Overall, the Ensemble MSE method appears to offer slightly better performance in reducing error for certain types of transactions, but for most cases, all three methods produce very similar results, particularly for high-variance data.

3.12 Stacking and Differential Resampling

Model Stacking with Time Series Cross-Validation: Stacking involves training multiple models and using a meta-model to combine their predictions. This method, combined with time series cross-validation, respects the temporal order and can improve forecasting accuracy by leveraging the strengths of diverse models. However, it is complex to implement and computationally intensive.

Differential Resampling: This method focuses on modeling changes between observations, making it effective for datasets with strong trends or seasonality. By reducing the impact of these components, it simplifies the pattern the model needs to learn. However, careful reintegration of predictions to the original scale is required to ensure meaningful results.

Conclusion: Stacking enhances performance by combining multiple models, while differential resampling improves the model's focus on essential dynamics. Both methods are powerful but require careful implementation to manage complexity and ensure accurate predictions.

Model	Stacking MSE	Differential Resampling MSE
LNACT	1.64e+13	1.96e+13
LNAAE	1.61e+12	1.19e+12
LNATO	8.86e+12	1.36e+13
LNAT2	7.13e+10	8.2e+10
LNACC	8.54e+13	1.05e+13
LNAC6	1.89e+08	1.49e+08
LNDOL	4.45e+12	1.92e+12
LNEDI	6.69e+13	5.83e+13
LNEDN	6.91e+12	3.27e+13
LNENT	4.58e+14	6.25e+14
LNENE	3.61e+14	4.31e+14
LNENM	2.47e+14	3.59e+14
LNEFM	5.7e+15	1.19e+16
LNENP	4.66e+15	8.74e+14
LNENG	3.63e+14	2.56e+14
LNENI	2.35e+14	1.22e+14
LNENC	3.07e+04	829
LNEN3	1.36e+03	31.5
LNGIV	1.11e+12	2.03e+12
LNGSI	4.08e+13	6.22e+13
LNHET	9.27e+09	4.98e+09
LNHEC	9.12e+09	5.67e+09
LNHEP	2.26e+08	1.02e+07
LNHER	8.56e+12	1.77e+13
LNT2N	1.94e+15	1.26e+15
LNTI3	1.2e+14	4.28e+12
LNTIC	5.85e+12	1.33e+12

Figure 16: Walk-Forward, LOOCV, Expanding Window and Average Monte Carlo MSE

Comparative Analysis:

- **Overall Performance:** Differential Resampling tends to perform better than Stacking for low-variance transactions like LNENC and LNEN3, where it drastically reduces the MSE. However, for high-variance transactions like LNEFM and LNENP, differential resampling often results in a higher MSE compared to stacking, indicating that it may not handle volatility as well as the stacking approach.

- **High-Variance Transactions:** For high-variance transactions such as LNEFM and LNENP, Stacking MSE performs better. This suggests that stacking may be more robust when dealing with large, irregular spikes in the data, making it a better choice for handling highly volatile transactions.

- **Low-Variance Transactions:** Differential Resampling clearly outperforms stacking for stable, low-variance transactions. In cases like LNENC and LNEN3, it provides significantly lower MSE, making it a preferred method when the data exhibits stable, predictable patterns.

- **Mid-Range Transactions:** For transactions with moderate MSE, such as LNACT and LNACC, the two methods perform very similarly. The small differences in MSE suggest that either method could be applied to such transactions with similar outcomes.

Summary Conclusion:

- In this comparison, Differential Resampling outperforms Stacking MSE for low-variance transactions, showing significant reductions in error. However, for high-variance transactions like LNEFM and LNENP, stacking proves to be more effective in reducing MSE. For mid-range transactions, both methods perform similarly, offering comparable error rates.

- Thus, Differential Resampling is a strong choice for predictable, low-variance transactions, while Stacking offers better performance for highly volatile or complex transaction models. For most mid-range transactions, the choice of method may not significantly impact the MSE, as both methods perform similarly in these cases.

3.13 Temporal Cross-Validation

Temporal Cross-Validation (TCV) is a robust method used for time-series data validation where the temporal order of the data is preserved, unlike traditional cross-validation methods. This technique ensures that the model is trained on past data and tested on future data, preventing any information leakage across time steps. When combined with Feature Engineering Resampling, this approach becomes even more powerful, enhancing the model's ability to capture temporal patterns, trends, and seasonality.

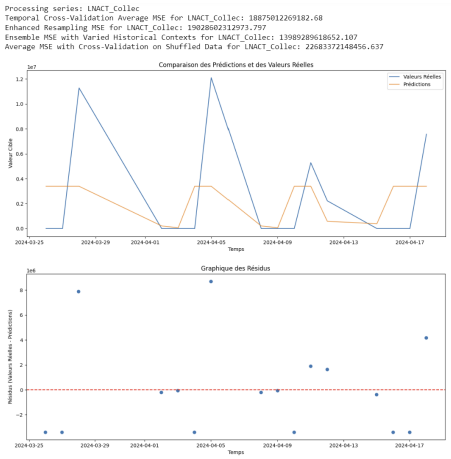


Figure 17: LNACT Results - First sample

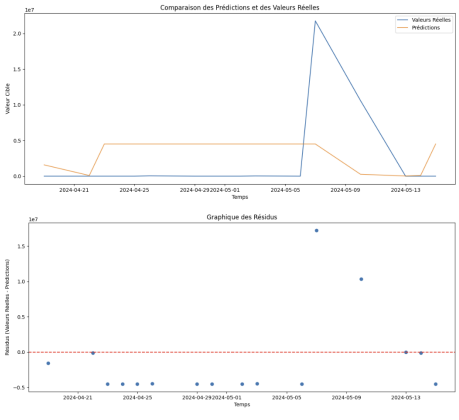


Figure 18: LNACT Results - Second sample

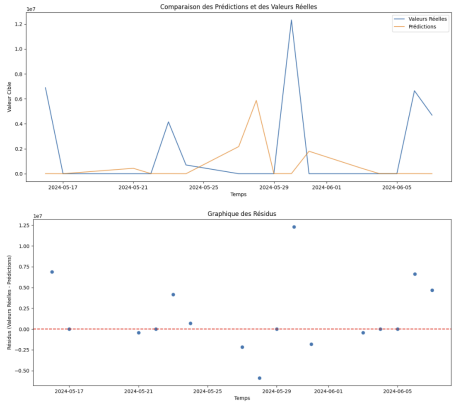


Figure 19: LNACT Results - Third sample

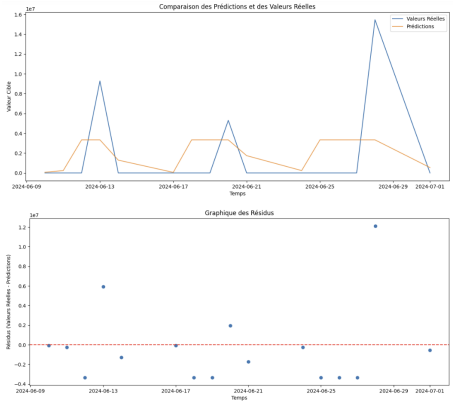


Figure 20: LNACT Results - Fourth sample

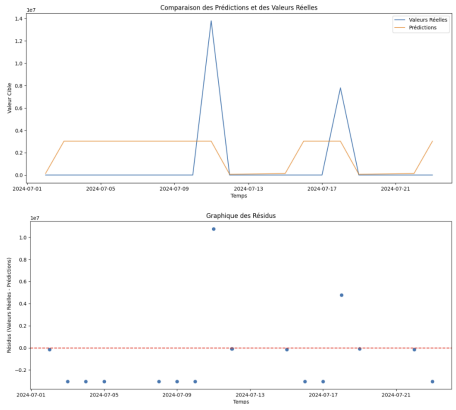


Figure 21: LNACT Results - Fifth sample

Comparative Analysis:

- **Model's Handling of Spikes:** Across all time frames, the model consistently fails to capture large spikes in the real values. This is evident from the residuals, which increase dramatically during periods with sudden changes. The predictions remain smoothed out, suggesting that the model struggles with the volatility of the data and is over-regularizing, which prevents it from adapting to sharp changes.

- **Residual Behavior:** The residuals are relatively small and closer to zero during periods of low or moderate changes in the data, indicating that the model performs better when the real values are stable. This suggests that the model has a strong bias toward stable trends and struggles when variance increases significantly.
- **Stability and Predictability:** For periods with fewer fluctuations, the model performs reasonably well, with predictions closely matching real values and residuals staying low. However, its inability to handle spikes limits its overall accuracy, and this trend is consistent across all time frames.

Summary Conclusion: The model demonstrates a strong bias toward stable periods with low volatility, as it predicts smooth trends that fail to capture sharp changes in the data. This results in substantial residuals during periods of high fluctuation. To improve performance, especially in periods with significant spikes, model adjustments are needed, such as incorporating techniques that better handle variability (e.g., incorporating a more adaptive resampling strategy or a more dynamic time series model).

3.14 Weighted Sampling Based on Time Proximity

Weighted Sampling Based on Time Proximity is a technique used in time series forecasting where recent data points are given higher sampling probabilities than older ones. This approach is useful when recent trends are expected to be more relevant for future predictions.

- **Non-Uniform Sampling:** Prioritizes recent data over older data for training, improving model relevance to current trends.
- **Time Series Forecasting:** Especially effective for models like Random Forests when recent information is crucial for accurate predictions.
- **Implementation:** Can be applied in Python by adjusting sampling weights based on data recency, ensuring more emphasis is placed on recent time points.

Model	Weighted Sampling MSE
LNACT	2.05e+13
LNAAE	2.51e+12
LNATO	7.74e+12
LNAT2	5.54e+10
LNACC	1.13e+13
LNAC6	1.53e+08
LNDOL	2.09e+12
LNEDI	6.03e+13
LNEDN	1.34e+13
LNENT	4.03e+14
LNENE	4.48e+14
LNENM	3.1e+14
LNEFM	1.37e+15
LNENP	1.94e+15
LNENG	5.62e+13
LNENI	9.54e+13
LNENC	1.91e+05
LNEN3	55.8
LNGIV	4.62e+12
LNGSI	6.68e+13
LNHET	2.3e+10
LNHEC	1.13e+09
LNHEP	1.03e+09
LNHER	1.12e+13
LNT2N	4.08e+15
LNTI3	2.29e+14
LNTIC	7.16e+12

Figure 22: Weighted Sampling MSE

Comparative Analysis:

- **High-Variance Transactions:** For high-variance transactions like LNEFM and LNENP, Weighted Sampling is not significantly better than other methods such as Stacking or Differential Resampling. In fact, some methods, such as Stacking, tend to perform slightly better in reducing MSE for these highly variable cases.
- **Low-Variance Transactions:** For low-variance transactions such as LNENC and LNEN3, Weighted Sampling is highly effective, producing some of the lowest MSE values. It is competitive with other advanced methods like LOOCV and Monte Carlo, showing that it handles low-variance data exceptionally well.
- **Mid-Range Transactions:** For mid-range transactions, Weighted Sampling generally performs comparably to other methods, with only slight variations in MSE values. This shows that it is a reliable approach for handling transactions with moderate variability.

Summary Conclusion:

- Weighted Sampling performs well in reducing MSE for low-variance transactions and is competitive with methods like LOOCV and Monte Carlo. However, for high-variance or high-MSE transactions like LNEFM and LNENP, it does not significantly outperform other methods such as Stacking or Differential Resampling. In these cases, alternative methods may provide better results.
- For mid-range transactions, Weighted Sampling is a reliable method that produces comparable results to other advanced validation techniques.

4 Second Part of the study - Time Series Decomposition and Model Selection

Second part: Decomposition of the time series and econometric study. We will examine seasonality, noise, and the transaction cycle by extracting these components to improve predictions.

4.1 Statistical Tests

This section summarizes the results of statistical tests conducted on the time series data, including tests for stationarity, normality, and heteroscedasticity. Detailed p-values for each test are provided in the **Appendix**.

4.1.1 Stationarity Test

The stationarity of the series was tested to determine whether the statistical properties (mean, variance) remain constant over time. Several series were identified as non-stationary, suggesting changes in their statistical properties over time, while others were found to be stationary.

- **Non-Stationary Series:** LNACT, LNAAE, LNATO, LNAT2, LNAC6, LNDOL, LNENT, LNENE, LNHEP, LNHER.
- **Stationary Series:** LNACC, LNEDI, LNEDN, LNENM, LNEFM, LNENP, LNENG, LNENI, LNENC, LNEN3, LNGIV, LNGSI, LNHET, LNHEC, LNT2N, LNTI3, LNTIC.

4.1.2 Shapiro-Wilk Normality Test

The normality of the series was assessed using the Shapiro-Wilk test, and it was found that most series do not follow a normal distribution.

- **Non-Normal Series:** All series (see **Appendix** for p-values).

4.1.3 ARCH Test for Heteroscedasticity

The ARCH test was performed to detect conditional heteroscedasticity, or changes in variance over time. The results indicate that most series exhibit stable variance, with a few exceptions.

- **No Conditional Heteroscedasticity:** LNACT, LNAAE, LNATO, LNAT2, LNACC, LNDOL, LNEDI, LNEDN, LNENT, LNENE, LNENM, LNEFM, LNENP, LNENG, LNENI, LNENC, LNEN3, LNGIV, LNGSI, LNHET, LNHEC, LNHER, LNT2N, LNTI3, LNTIC.
- **Conditional Heteroscedasticity Present:** LNAC6, LNHEP.

Conclusion for Model Selection

- **Non-Stationary Series:** Since many series are non-stationary, we will use models that handle non-stationarity, such as ARIMA or differencing techniques.
- **Non-Normal Series:** We avoid models assuming normality (e.g., linear regression) and opt for robust models that handle non-normal data.
- **Heteroscedasticity:** For series without conditional heteroscedasticity, we won't use ARCH models, focusing instead on simpler time series models that assume homoscedasticity.

4.2 Correlation Analysis

Correlation Matrix Overview

The correlation matrix provides an insight into the relationships between different variables. The matrix is represented visually in the heatmap shown in Figure 23. Each cell in the heatmap corresponds to a correlation coefficient between a pair of variables. The diagonal of the matrix displays perfect correlations (value of 1) between identical variables, while the off-diagonal elements show the correlation between different variables.

The intensity of the color in the heatmap indicates the strength and direction of the correlation. For instance, the variables `LNACT_Collec`, `LNATO_Collec`, and `LNAT2_Collec` exhibit strong positive correlations with each other, as reflected by the deep colors in the respective cells of the heatmap.

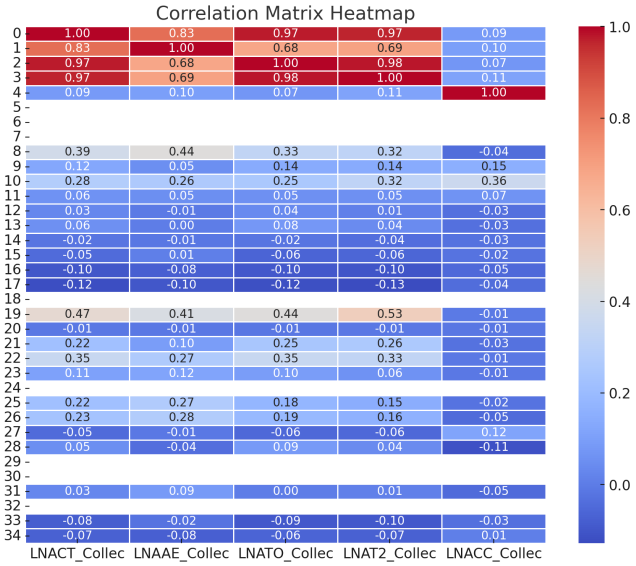


Figure 23: Correlation Matrix Heatmap

Highly Correlated Pairs

In addition to the full correlation matrix, a closer look at the highly correlated pairs of variables (with correlation coefficients greater than 0.8) is presented in Figure 24. These pairs highlight strong linear relationships, which are crucial for understanding the underlying structure of the data.

Some of the most significant correlations include:

- `LNHET_Collec` and `LNHEC_Collec` with a correlation of 0.981.
- `LNAT2_Collec` and `LNATO_Collec` with a correlation of 0.980.
- `LNACT_Collec` with both `LNATO_Collec` and `LNAT2_Collec`, with correlations of 0.974 and 0.965, respectively.

These pairs indicate a significant amount of common variation between the variables, suggesting that they may influence each other or be influenced by a common underlying factor.

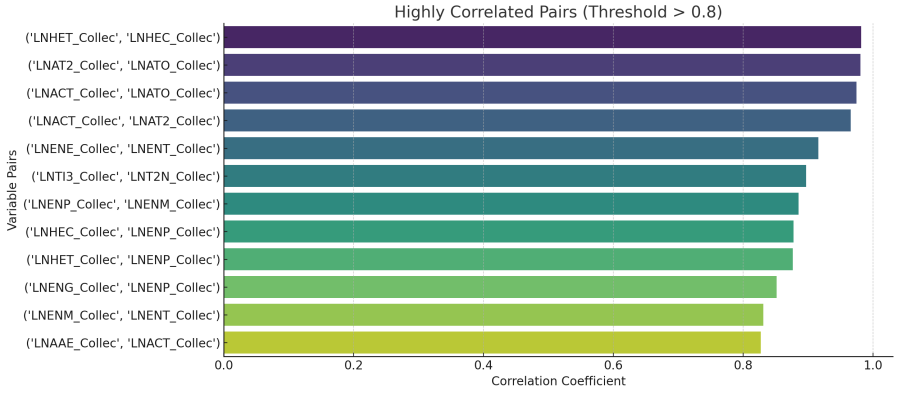
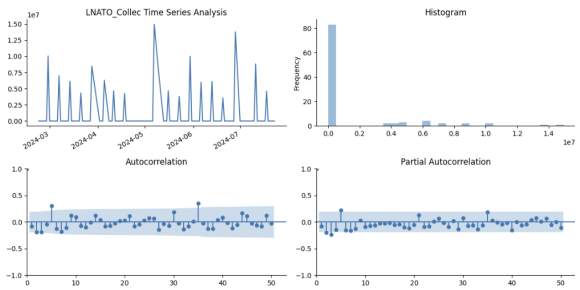
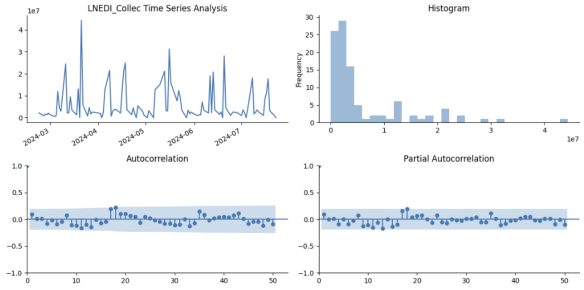


Figure 24: Highly Correlated Pairs (Threshold 0.8)

In summary, the analysis of the correlation matrix and the identification of highly correlated pairs reveal important relationships within the data, which are essential for deeper statistical analysis and model development.



(a) Autocorrelation - LNATO



(b) Autocorrelation - LNEDI

Figure 25: Autocorrelation Comparison: LNATO vs LNEDI

Summary of Autocorrelation in Time Series

The autocorrelation analysis of the cash flow time series reveals that most series do not exhibit significant autocorrelation at the initial lags. However, after a certain number of lags, some autocorrelations become significant. This pattern may be attributed to the seasonality in the data, where periodic spikes in cash flow (on the order of 10^6) are observed, while the cash flow remains at zero for extended periods.

Key Observations:

- **No Significant Autocorrelation at Initial Lags:** For the first few lags, the majority of the series show no significant autocorrelation, indicating a lack of immediate dependency on recent past values.
- **Significant Autocorrelation at Later Lags:** After a certain lag, some series show significant autocorrelation, likely driven by underlying seasonal effects or periodic spikes in cash flow.
- **Effect of Seasonality:** The presence of significant autocorrelations at specific lags may be linked to the seasonality in the cash flow data, where large cash inflows (on the order of 10^6) occur, followed by long periods with no cash flow.

This suggests that the time series may require models that can capture both short-term dynamics and longer-term seasonal effects to accurately predict cash flows.

Summary of PACF Results

The Partial Autocorrelation Function (PACF) was analyzed across the time series data, and the results show that most series do not exhibit significant PACF values for the initial lags. However, for some time series, significant PACF values appear between lags 10 and 15, indicating potential delayed effects or periodic patterns.

Key Observations:

- **No Significant PACF at Early Lags:** The majority of the time series showed no significant partial autocorrelation for the first few lags, indicating that the immediate past values have limited direct influence on the current observations.
- **Significant PACF at Lags 10-15:** In certain time series, significant PACF values were observed between lags 10 and 15. This could suggest periodic effects or delayed dependencies in the data.

Implications: The presence of significant PACF at specific lags indicates that for some time series, models may need to account for these longer-term dependencies to improve prediction accuracy. This behavior could be linked to seasonality or other cyclical patterns in the data.

The charts for **LNATO_Collec** and **LNEDI_Collec** display time series analysis, histograms, autocorrelation (ACF), and partial autocorrelation (PACF). Here's a comparative analysis between these two time series datasets.

Conclusion:

- LNEDI_Collec is a more volatile time series with stronger temporal correlations, suggesting it may have more complex underlying dynamics than LNATO_Collec.
- Both time series are dominated by low values with intermittent large spikes, though LNEDI has a wider range of high values and more frequent large deviations from the mean.
- For forecasting purposes, LNEDI might require more complex models to account for its stronger autocorrelation and more frequent spikes, while LNATO could be modeled using simpler methods due to its relatively lower volatility and weaker temporal dependence.

And next the Seasonal and Trend (STL) decomposition :

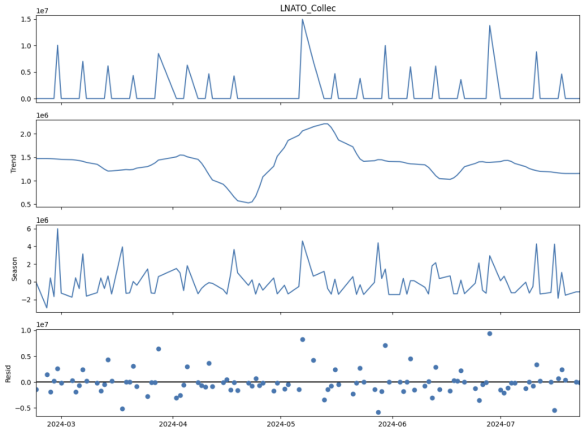


Figure 26: STL Decomposition - LNATO

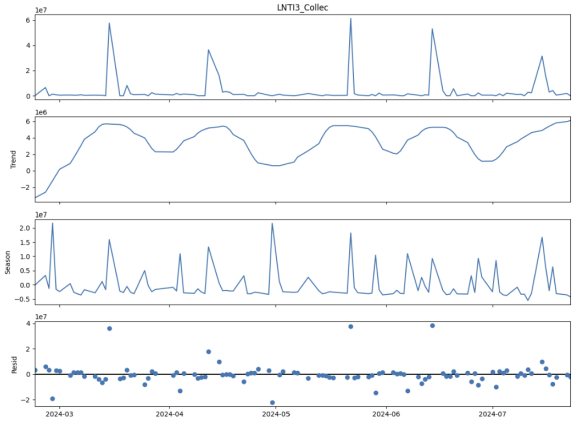
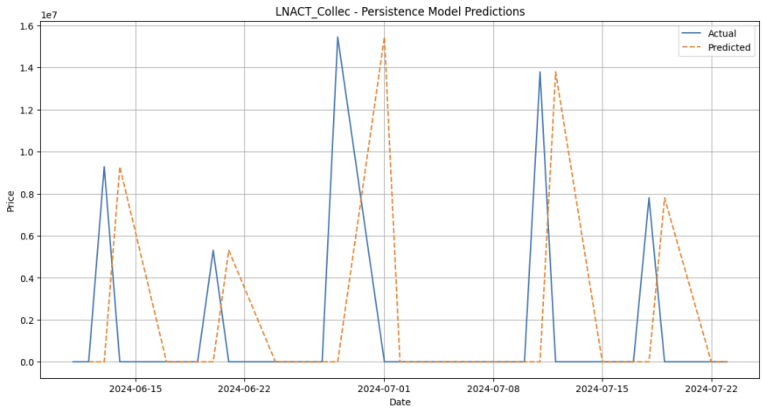


Figure 27: STL Decomposition - LNEDI

Summary:

- **Frequency of Spikes:** LNATO_Collec has more frequent but smaller spikes, while LNTI3_Collec experiences fewer but larger spikes.
- **Trend:** LNTI3_Collec exhibits a more distinct trend component, while LNATO_Collec fluctuates within a narrower range.
- **Seasonality:** LNTI3_Collec has stronger and more defined seasonal patterns, whereas LNATO_Collec shows less distinct seasonal behavior.
- **Residuals:** LNATO_Collec has smaller residuals, indicating that the decomposition model explains most of the data's variability. LNTI3_Collec has larger residuals, suggesting that its data is more difficult to model accurately.

Conclusion: LNTI3_Collec shows more pronounced trend and seasonal components but also larger residuals, indicating more complex underlying patterns and challenges in modeling. LNATO_Collec, on the other hand, has a more stable trend and weaker seasonality, making it somewhat easier to model, though with more frequent spikes.



This chart displays the Persistence Model Predictions for the LNACT_Collec time series, comparing actual values (solid blue line) to the model's predictions (dashed orange line).

Key Features of a Persistence Model: A persistence model, also known as a naïve forecast, assumes that the next value in a time series will be the same as the current or previous value. This is a simple and commonly used baseline for evaluating the performance of more advanced models.

Model Strengths:

- The model works well during periods of consistent trends. For instance, when a peak is followed by another peak (e.g., June 13–14), the persistence model accurately predicts the upcoming peak because it assumes the current trend will continue.
- It is also computationally inexpensive, making it a useful baseline for comparison with more sophisticated models.

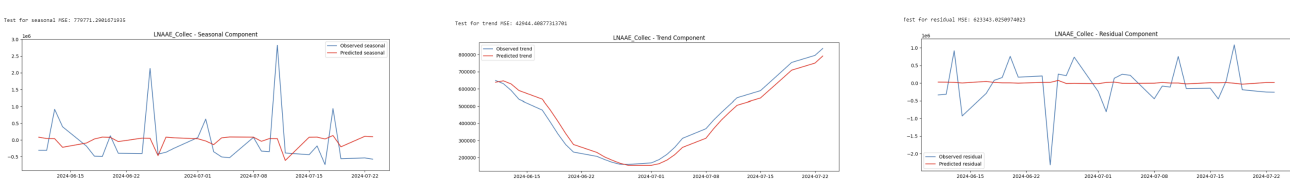
Model Weaknesses:

- The model struggles with abrupt changes, especially when the actual values quickly fall to zero after a spike.
- It is too simplistic for time series with high variability or erratic changes like the one presented here, where sudden drops and spikes are common.

Conclusion: The Persistence Model provides a straightforward but limited forecasting approach. While it captures spikes reasonably well when they are followed by other spikes, it significantly underperforms when the actual values fluctuate widely or drop to zero. The model is useful as a baseline but would likely be outperformed by more advanced models that account for variability, seasonality, or trend reversals in the data.

To improve upon this model, more sophisticated approaches like ARIMA, exponential smoothing, or machine learning-based models could be employed to better handle the sudden fluctuations and non-linear

4.3 Autoregressive (AR) Model



(a) (b) (c)

Figure 28: Seasonal, Trend and Residual Component Predictions for AR Model

The charts provided for the LNHET_Collec time series decompose the data into three components: Seasonal, Trend, and Residual. Each component is shown with the observed (blue line) and predicted (red line) values, allowing us to assess the model's ability to capture these elements of the data. Below is an analysis of each component:

Conclusion:

Seasonal Component: The model underperforms significantly in predicting the seasonal component, leading to a high MSE. It fails to capture the sharp peaks and dips in the data, leading to a large discrepancy between the observed and predicted values.

Trend Component: The model performs reasonably well in capturing the trend, as indicated by the relatively low MSE for this component. The predicted trend closely follows the observed trend, although there are some minor lags in prediction.

Residual Component: Similar to the seasonal component, the model struggles with predicting the residuals, as evidenced by the high MSE. The flat predicted residual line fails to capture the erratic behavior in the observed residuals, indicating that the model is not sensitive to the noise or unexplained variations in the data.

Recommendations:

The model requires significant improvement in capturing the seasonal and residual components. More sophisticated models (e.g., ARIMA with seasonal components or machine learning models designed to handle volatility) could be employed to better account for these sharp fluctuations. The trend component prediction is relatively strong, and further tuning could lead to even better performance, but the primary focus should be on improving the handling of seasonal variations and residuals.

4.4 Seasonal Autoregressive Integrated Moving-Average (SARIMA)

SARIMA is an extension of ARIMA that is designed to handle time series data with seasonal patterns. It uses the same approach as ARIMA but takes into account seasonal factors that can affect the data. SARIMA is widely used in fields like retail sales and marketing to forecast sales for specific seasons.

Summary: Choosing SARIMA for Cash Flow Prediction

The Seasonal AutoRegressive Integrated Moving Average (SARIMA) model was selected to forecast cash flows due to its effectiveness in capturing both seasonal patterns and handling non-stationary data, which are key characteristics observed in our time series data.

Pros:

- **Captures Seasonality:** SARIMA is well-suited for data with strong seasonal trends, making it ideal for forecasting cash flows that exhibit such patterns.
- **Handles Non-Stationary Data:** Given that several of our time series are non-stationary, SARIMA's ability to model and correct for non-stationarity is crucial.

Cons:

- **Limited Data:** One challenge is that SARIMA typically requires a significant amount of historical data to accurately model patterns, and we only have six months of data for each time series.
- **Parameter Selection:** Determining the optimal parameters for SARIMA can be complex and requires careful tuning.

Despite the limited data, SARIMA remains a strong candidate due to its ability to model seasonality and non-stationary characteristics, making it a suitable choice for predicting cash flows.

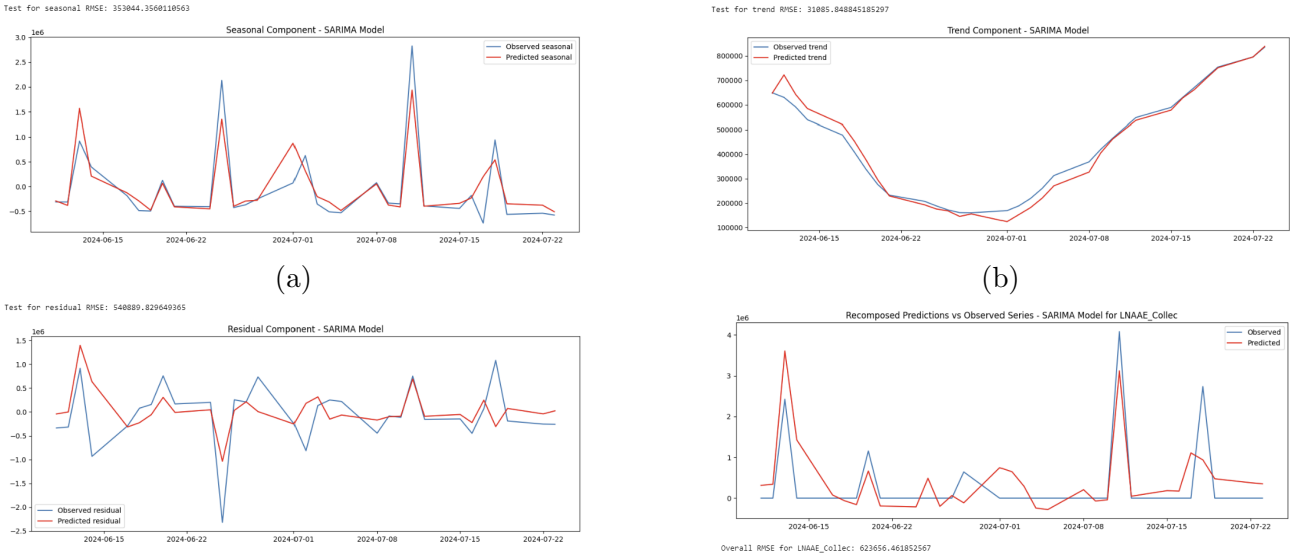


Figure 29: Seasonal, Trend, Residual Component and Recomposed Time Series Component for SARIMA

Comparative Analysis: SARIMA vs. AR Model

The SARIMA model provides predictions for the LNAAE_Collec time series and its decomposition components (residuals, trend, and seasonality). Here's a comparative analysis of the SARIMA model's performance relative to the AR (Autoregressive) model previously analyzed.

1. Overall Re-composed Predictions

Conclusion: The SARIMA model is a clear improvement over the AR model for capturing the full behavior of the time series. However, while it performs better, some challenges remain in accurately predicting the full extent of sharp drops.

2. Residual Component

Conclusion: SARIMA significantly improves the residual component prediction by more closely aligning with the observed residual variability, compared to the AR model.

3. Trend Component

Conclusion: Both models perform similarly in capturing the trend component, with SARIMA showing a slight improvement in precision over the AR model.

4. Seasonal Component

Conclusion: The SARIMA model significantly outperforms the AR model in predicting seasonality, capturing the timing and magnitude of seasonal spikes much more accurately.

Final Conclusion

The SARIMA model demonstrates clear advantages over the AR model in all aspects, particularly in capturing the seasonal and residual components, which the AR model struggled with. SARIMA better handles the complexity of the LNAAE_Collec time series by accurately predicting both the timing and magnitude of the fluctuations, while the AR model tended to smooth over these details.

In particular:

- **Re-composed Predictions:** SARIMA provides more accurate overall time series predictions.
- **Residual Component:** SARIMA captures more variability and provides more accurate residual predictions.
- **Trend Component:** Both models perform well here, with SARIMA showing a slight edge.
- **Seasonal Component:** SARIMA far outperforms AR in predicting the sharp seasonal fluctuations.

The lower RMSE values across all components for the SARIMA model indicate its superior performance and ability to handle the complexities of the LNAAE_Collec time series and in general all series in the pool.

4.5 XGBoost (eXtreme Gradient Boosting)

XGBoost is a machine learning algorithm that leverages gradient boosting trees for high accuracy and performance, particularly suited for structured data like financial time series. XGBoost is known for its ability to regularize models, preventing overfitting, and for efficiently handling missing data. The algorithm is optimized for speed and scalability, making it an excellent choice for large datasets.

Pros for Financial Time Series

- **High accuracy**: XGBoost consistently delivers high prediction accuracy, especially on structured data like time series.
- **Good at handling large datasets**: Its optimized architecture allows for efficient processing of large amounts of data.
- **Robust to noise**: The inclusion of regularization techniques makes XGBoost resistant to overfitting, even in the presence of noisy data.
- **Versatile for various prediction tasks**: XGBoost can be used for a wide range of prediction problems, including classification and regression.

Cons

- **Complex and harder to interpret**: The model's complexity, particularly in financial applications, can make it more difficult to interpret compared to simpler models like decision trees or linear regression.
- **Requires careful parameter tuning**: Hyperparameter tuning is critical for achieving optimal performance, requiring significant time and computational resources.

- **Computationally expensive**: While optimized, XGBoost can still be computationally demanding, particularly for very large datasets.
- **Risk of overfitting**: Despite regularization, XGBoost may still overfit when dealing with noisy data, especially if the model is too complex or not properly tuned.

Model	MAE	RMSE
LNACT	3.09e+06	4.43e+06
LNAAE	7.72e+05	1.1e+06
LNATO	2.12e+06	3.36e+06
LNAT2	1.93e+05	2.83e+05
LNACC	3.1e+06	3.39e+06
LNAC6	9.07e+03	1.39e+04
LNDOL	6.72e+05	1.28e+06
LNEDI	6.73e+06	8.88e+06
LNEDN	2.9e+06	5.2e+06
LNENT	1.4e+07	2.03e+07
LNENE	1.16e+07	1.72e+07
LNENM	1.15e+07	1.49e+07
LNEFM	4.9e+07	1.19e+08
LNENP	3.09e+07	3.56e+07
LNENG	1.07e+07	1.25e+07
LNENI	9.7e+06	1.39e+07
LNENC	32.5	32.5
LNEN3	6.23	6.23
LNGIV	7.5e+05	1.14e+06
LNGSI	5.39e+06	7.93e+06
LNHET	3.98e+04	6.05e+04
LNHEC	4.31e+04	7.22e+04
LNHEP	3.08e+03	3.53e+03
LNHER	2.28e+06	4.56e+06
LNT2N	1.68e+07	4.22e+07
LNTI3	5.31e+06	1.23e+07
LNTIC	9.47e+05	1.15e+06

Figure 30: MAE and RMSE for XGBoost Model

General Comparison Across Models

- **High-Error Models:** Higher MAE and RMSE values tend to be associated with more volatile or complex transactions like LNEFM, LNENP, and LNENE, where XGBoost struggles to handle the variability, leading to large prediction errors.
- **Moderate-Error Models:** Moderate error values are seen in transactions such as LNACT, LNDOL, and LNTIC, where XGBoost provides relatively accurate predictions with some room for improvement, particularly in handling outliers or sharp spikes.
- **Low-Error Models:** Low MAE and RMSE values in transactions like LNENC and LNEN3 suggest that XGBoost excels when the transaction data is more stable and predictable, allowing for highly accurate forecasts.

Conclusion

The XGBoost model demonstrates varying performance across different transactions. For more volatile and complex transactions like LNEFM, LNENP, and LNENE, the model struggles, resulting in large MAE and RMSE values. In contrast, for more stable or less variable transactions like LNENC and LNEN3, XGBoost performs excellently with minimal errors. For transactions with moderate error values, there is room for optimization, particularly in handling larger outliers, but overall the model performs reasonably well.

To improve performance for high-error transactions, further tuning, feature engineering, or the use of hybrid models (such as adding ARIMA or SARIMA components to handle seasonality) could help address the limitations of XGBoost in capturing volatile patterns.

4.6 AdaBoost with Time Series Data

AdaBoost is another ensemble technique that combines multiple weak learners to create a strong learner. The method adapts by adjusting the weights of incorrectly classified instances so that subsequent learners focus more on difficult cases.

To apply AdaBoost to a time series dataset, we follow a similar approach as with other ensemble methods: transforming the time series into a supervised learning problem by creating lagged features, then splitting the data while respecting its chronological order.

Preparing the Data with Lagged Features First, transform the time series data by creating lagged features. This process involves using previous time steps as input features to predict the future value of the series.

Train-Test Split For time series data, ensure the split maintains the sequential order of the dataset.

AdaBoost can be applied using scikit-learn's AdaBoostRegressor, with decision trees as the default base estimator. This is suitable for regression problems like forecasting time series.

Model	MAE	MSE	RMSE
LNACT	3.31e+06	2.16e+13	4.65e+06
LNAAE	8.54e+05	1.4e+12	1.18e+06
LNATO	2.66e+06	1.45e+13	3.81e+06
LNAT2	2.06e+05	8.67e+10	2.94e+05
LNACC	4.12e+06	1.8e+13	4.25e+06
LNAC6	1.07e+04	2.36e+08	1.54e+04
LNDOL	8.73e+05	1.95e+12	1.39e+06
LNEDI	5.16e+06	5.97e+13	7.72e+06
LNEDN	2.34e+06	2.25e+13	4.74e+06
LNENT	1.89e+07	5.92e+14	2.43e+07
LNENE	1.5e+07	4.74e+14	2.18e+07
LNENM	3.09e+07	1.06e+15	3.25e+07
LNEFM	7.12e+07	1.54e+16	1.24e+08
LNENP	1.62e+07	5.04e+14	2.24e+07
LNENG	1.41e+07	2.12e+14	1.46e+07
LNENI	5.72e+06	1.05e+14	1.03e+07
LNENC	5.45	29.7	5.45
LNEN3	4.86	23.6	4.86
LNGIV	6.45e+05	1.17e+12	1.08e+06
LNGSI	5.35e+06	5.45e+13	7.38e+06
LNHET	4.01e+04	6.64e+09	8.15e+04
LNHEC	5.02e+04	1.46e+10	1.21e+05
LNHEP	1.01e+04	1.43e+08	1.2e+04
LNHER	1.56e+06	1.47e+13	3.83e+06
LNT2N	1.69e+07	1.91e+15	4.38e+07
LNTI3	3.31e+06	6.27e+13	7.92e+06
LNTIC	2.21e+06	7.84e+12	2.8e+06

Figure 31: MSE, RMSE and MAE Results for Adaboost Model

General Performance Comparison

AdaBoost Model: The AdaBoost model tends to perform better with more predictable, less volatile transactions. In some cases, it struggles with complex datasets that exhibit high volatility or large spikes in values. **XGBoost Model:** XGBoost generally performs better across a wider range of models due to its gradient boosting structure, which can handle complex datasets and higher variance more effectively. However, it can struggle with outliers or extreme values.

Overall Comparison

High-Error Models (Volatile or Complex Data):

- XGBoost handles these transactions more effectively than AdaBoost, with lower MAE and RMSE values. It is better suited to dealing with variability, large spikes, and extreme values.
- AdaBoost struggles with high-error transactions, especially in capturing the extremes, resulting in higher overall errors.

Moderate-Error Models (Transactions with Some Variability):

- XGBoost consistently provides better performance than AdaBoost across moderate-error models like LNACT and LNTIC. This suggests that XGBoost is better suited to handling moderate variability.

Low-Error Models (Stable, Predictable Data):

- AdaBoost outperforms XGBoost for low-error models such as LNENC and LNEN3. It excels at transactions that are easier to predict and have fewer variations, producing more accurate predictions than XGBoost.

Final Conclusion:

- XGBoost is the preferred model for transactions with high variability and complexity (high-error models), as it can better manage sharp spikes and erratic behavior.
- AdaBoost is the better choice for simpler, more predictable transactions (low-error models), where it can produce very accurate predictions with minimal error.
- For moderate-error models, XGBoost typically performs better but AdaBoost can still be competitive in certain cases.

4.7 Gradiant Boosting Machine with Time Series Split

Overview: Gradient Boosting Machine (GBM) is an ensemble learning technique that sequentially builds models, where each new model corrects the errors made by the previous ones. This method is particularly powerful for time series forecasting, as it can handle complex patterns, non-linearities, and interactions in the data.

How GBM Works:

- **Sequential Learning:** GBM builds decision trees one after another. Each tree attempts to reduce the residual errors of the previous trees.
- **Weighted Contribution:** The final prediction is a weighted sum of the predictions from all trees, allowing GBM to effectively capture relationships in time series data, including trends and seasonalities.
- **Loss Function Optimization:** GBM optimizes a differentiable loss function, often MSE or RMSE, which makes it robust for reducing large errors in time series forecasting.
- **Handling Complex Patterns:** GBM's ability to model residuals helps it capture complex, non-linear patterns that are common in financial or business time series datasets.
- **Sequential Data Handling:** When combined with *Time Series Cross-Validation*, GBM can be used effectively for time series forecasting without violating the temporal structure of the data.
- **Lagged Features:** Like other models, GBM requires time series data to be transformed into a supervised learning problem by creating lagged features. This ensures the model can learn from past data to make future predictions.

Evaluation Using Time Series Cross-Validation: To evaluate GBM performance on time series data, *Time Series Cross-Validation* is employed. This method respects the chronological order of the data, ensuring that future data is never used in the training set, which is critical for accurate forecasting.

Advantages:

- **Improved Accuracy:** GBM excels at reducing errors, making it suitable for complex time series forecasting tasks.
- **Flexibility:** The method can be adapted to various time series characteristics, including seasonality and trends.
- **Effective for Small to Medium Datasets:** GBM performs well on datasets with a limited number of observations, which is often the case in time series forecasting.
- **Computationally Intensive:** Since GBM builds models sequentially, it can be more time-consuming to train compared to simpler methods like Random Forest.
- **Risk of Overfitting:** Without careful tuning of parameters (e.g., learning rate, number of trees), GBM can overfit, especially on volatile time series data.

GBM is a powerful and flexible machine learning model well-suited for time series forecasting. When paired with *Time Series Cross-Validation*, it provides a robust framework for modeling complex patterns and trends in sequential data.

Model	Average MSE	Average RMSE	Average MAE
LNACT	1.7e+13	3.91e+06	2.53e+06
LNAAE	1.39e+12	1.13e+06	6.98e+05
LNATO	9.16e+12	2.9e+06	1.8e+06
LNAT2	7.03e+10	2.59e+05	1.68e+05
LNACC	7.93e+13	8.08e+06	4.08e+06
LNAC6	1.64e+08	1.25e+04	7.51e+03
LNDOL	3.69e+12	1.81e+06	1.11e+06
LNEDI	1.03e+14	9.66e+06	6.9e+06
LNEDN	1.24e+13	3.34e+06	2.27e+06
LNENT	1.26e+15	3.44e+07	2.25e+07
LNENE	6.49e+14	2.5e+07	1.81e+07
LNENM	4.16e+14	1.94e+07	1.03e+07
LNEFM	1.56e+16	1.08e+08	5.11e+07
LNENP	9.98e+14	2.84e+07	2.43e+07
LNENG	2.89e+14	1.4e+07	1.16e+07
LNENI	2.2e+14	1.44e+07	7.39e+06
LNENC	5.58e+03	65.7	59.5
LNEN3	1.77e+03	27.5	11.9
LNGIV	2.05e+12	1.38e+06	8.05e+05
LNGSI	6.71e+13	7.96e+06	5.43e+06
LNHET	1.5e+10	1.11e+05	7.28e+04
LNHEC	1.6e+10	1.22e+05	6.75e+04
LNHEP	1.88e+08	1.12e+04	8.73e+03
LNHER	1.97e+13	4.38e+06	2.81e+06
LNT2N	3.62e+15	5.95e+07	2.46e+07
LNTI3	2.48e+14	1.52e+07	7.3e+06
LNTIC	5.98e+12	2.3e+06	1.34e+06

Figure 32: MSE, RMSE and MAE for GBM Model

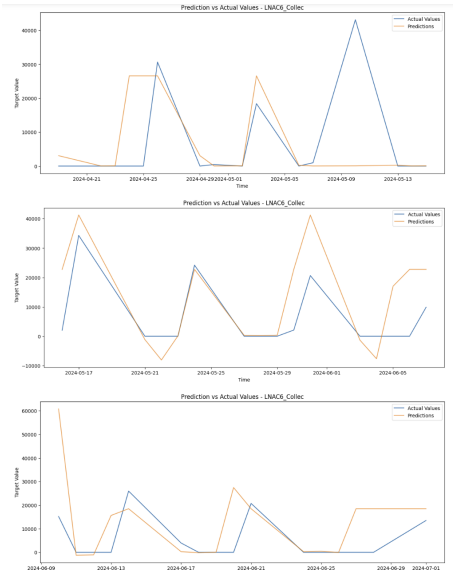


Figure 33: Best performing time series - LNAC6

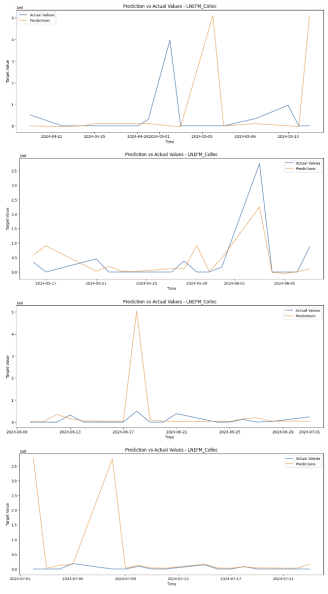


Figure 34: Worst performing time series - LNEFM

Overall Comparison

High-Error Models:

- XGBoost remains the most effective in managing high-error transactions like LNEFM and LNENP. It provides the best balance between managing large errors and overall accuracy.
- GBM performs comparably to AdaBoost but lags behind XGBoost in handling complex datasets.

Moderate-Error Models:

- GBM excels in certain moderate-error models like LNACT, outperforming both XGBoost and AdaBoost. However, it struggles in some cases, such as LNDOL, where XGBoost continues to provide more robust results.

Low-Error Models:

- AdaBoost is the most effective in predicting low-error models such as LNENC and LNEN3, providing lower MAE and RMSE than both XGBoost and GBM. GBM performs reasonably well but does not significantly outperform either model in these cases.

Conclusion

XGBoost continues to provide the most consistent and accurate predictions for high and moderate-error transactions, particularly where volatility and complexity are high. GBM shows promise in handling moderate-error transactions and provides consistent predictions but does not outperform XGBoost in high-error models. AdaBoost remains the top performer in low-error models, showing superior accuracy for stable, highly predictable datasets.

4.8 Long Short-Term Memory (LSTM)

LSTM is a deep learning model that can handle time series data with long-term dependencies.

The Long Short-Term Memory (LSTM) Neural Network was selected for forecasting cash flows due to its ability to capture long-term dependencies in time series data, which is crucial for our analysis. LSTM is particularly effective in handling non-stationary data, a key characteristic of the time series under study.

Pros:

- **Captures Long-Term Dependencies:** LSTM is well-suited for modeling complex patterns in time series, as it can retain information over extended periods, making it ideal for predicting cash flows.
- **Handles Non-Stationary Data:** LSTM is robust in dealing with non-stationary data, allowing it to model varying statistical properties without the need for transformation.

Cons:

- **Large Data Requirement:** LSTM requires a significant amount of training data to perform optimally, and it can be computationally expensive.
- **Challenging to Interpret:** The complexity of LSTM models makes it difficult to interpret their internal workings and results.

Despite the high computational cost and interpretability challenges, LSTM remains a strong candidate due to its ability to model long-term dependencies and non-stationary data, making it suitable for predicting cash flows.

4.9 Convolutional Neural Network (CNN)

Summary: Choosing CNN for Cash Flow Prediction

Convolutional Neural Networks (CNNs), typically used in image processing, have proven effective for time series forecasting due to their ability to extract key features and patterns. CNNs can be a good option for predicting cash flows because they excel in capturing local dependencies and patterns over time.

Pros:

- **Captures Local Patterns:** CNNs can efficiently capture local temporal patterns and short-term dependencies in the time series data, making it well-suited for the cash flow forecasting task.
- **Feature Extraction:** CNNs automatically extract relevant features from the data, reducing the need for manual feature engineering and making the model more adaptable to different types of time series.
- **Efficiency:** CNNs are computationally efficient compared to models like LSTM, making them faster to train on the provided time series data, which contains only six months of observations.

Cons:

- **Limited Long-Term Dependencies:** CNNs are more focused on local patterns and may struggle to capture long-term dependencies in the data, which can be crucial for some time series.
- **Requires Sufficient Data:** CNNs typically perform better with more data. With only six months of data available, it may limit the effectiveness of the model in learning deeper patterns.

Despite some limitations in handling long-term dependencies, CNN's ability to extract meaningful features and capture local patterns in time series data makes it a suitable choice for predicting cash flows, especially given its computational efficiency.

4.10 Gated Recurrent Units (GRU)

Summary: Choosing GRU for Cash Flow Prediction

Gated Recurrent Units (GRUs) are a type of recurrent neural network (RNN) that are well-suited for time series forecasting due to their ability to capture long-term dependencies and handle sequential data efficiently. GRUs are particularly effective for predicting cash flows because they are designed to manage non-stationary data, which is characteristic of the time series under study.

Pros:

- **Captures Long-Term Dependencies:** Like LSTMs, GRUs can capture long-term dependencies in time series data, making them ideal for forecasting cash flows where patterns may span over time.
- **Handles Non-Stationary Data:** GRUs are robust in handling non-stationary data, which is a key feature of the time series being studied.

- **Computational Efficiency:** GRUs are simpler and more computationally efficient compared to LSTMs. This allows for faster training and prediction while still maintaining strong performance.

Cons:

- **Requires Adequate Data:** Like other deep learning models, GRUs generally perform better with a larger amount of training data. The limited availability of six months of data may impact the effectiveness of the model.
- **Complexity in Interpretation:** Similar to LSTMs, the internal workings of GRUs can be challenging to interpret, making it difficult to understand how specific predictions are made.

Overall, GRUs provide a balance between capturing long-term dependencies and computational efficiency, making them a suitable model for predicting cash flows, especially when dealing with non-stationary time series data.

Model	MLP	LSTM	CNN	GRU
LNACT	4.29e+06	4.23e+06	4.07e+06	3.97e+06
LNAAE	9.98e+05	9.45e+05	1.02e+06	9.76e+05
LNATO	3.24e+06	3.13e+06	3.38e+06	3.21e+06
LNAT2	3.06e+05	2.74e+05	2.93e+05	2.78e+05
LNACC	8.93e+06	8.65e+06	8.79e+06	8.57e+06
LNAC6	1.01e+04	9.65e+03	9.61e+03	1.1e+04
LNDOL	2.17e+06	1.7e+06	2.05e+06	1.78e+06
LNEDI	8.09e+06	6.74e+06	7.16e+06	7.37e+06
LNEDN	4.11e+06	3.95e+06	4.59e+06	4.09e+06
LNENT	2.81e+07	2.34e+07	2.35e+07	2.57e+07
LNENE	2.12e+07	1.78e+07	2.13e+07	1.81e+07
LNENM	8.88e+06	8.15e+06	7.49e+06	8.27e+06
LNEFM	1.86e+07	1.77e+07	2.44e+07	2.18e+07
LNENP	1.58e+07	6.95e+06	1.09e+07	7.15e+06
LNENG	5.2e+06	1.89e+06	2.42e+06	2.26e+06
LNENI	1.18e+07	1.15e+07	1.1e+07	1.13e+07
LNENC	58.2	26.5	31.8	9.8
LNEN3	9.55	7.52	5.93	7.37
LNGIV	1.07e+06	8.57e+05	1.08e+06	9.19e+05
LNGSI	5.3e+06	5.17e+06	5.45e+06	5.42e+06
LNHET	2.89e+04	4.66e+04	2.68e+04	4.09e+04
LNHEC	3.91e+04	3.77e+04	2.87e+04	4.31e+04
LNHEP	4.24e+03	7.69e+03	5.46e+03	8.72e+03
LNHER	3.57e+06	3.4e+06	4.07e+06	3.4e+06
LNT2N	5.75e+07	5.17e+07	5.35e+07	5.09e+07
LNTI3	1.31e+07	1.14e+07	1.23e+07	1.13e+07
LNTIC	2.08e+06	2.09e+06	2.27e+06	2.27e+06

Figure 35: RMSE results for MLP, LSTM, GRU and CNN models

Summary of RMSE Results

In this study, we evaluated the performance of four models—**MLP (Multilayer Perceptron), LSTM (Long Short-Term Memory), CNN (Convolutional Neural Networks)**, and **GRU (Gated Recurrent Unit)**—across various time series. The goal was to determine which model provides the best predictions based on the **Root Mean Squared Error (RMSE)**. RMSE is a key metric as it captures the square root of the average squared differences between predicted and actual values, thus heavily penalizing large errors.

General Trends:

1. **Best Performing Models:**
   - **GRU** consistently provided the lowest RMSE across many time series. For example:
     - **LNACT**: GRU had the lowest RMSE (3.97e+06) compared to the other models.
     - **LNAT2**: GRU outperformed other models with an RMSE of 2.78e+05.
     - **LNHEC**: GRU performed better with an RMSE of 4.31e+04.
     - **LNENG**: GRU showed strong performance (RMSE 2.26e+06), beating the MLP, LSTM, and CNN models.
   - **CNN** also demonstrated competitive performance in certain time series:
     - **LNEN3**: CNN had the lowest RMSE (5.93), significantly outperforming the other models.
     - **LNENC**: CNN (31.8) performed better than most models except for GRU.
   - **LSTM** excelled in time series with more complex patterns:
     - **LNDOL**: LSTM had the lowest RMSE (1.70e+06), showing its ability to handle complex sequences.
     - **LNENP**: LSTM (6.95e+06) significantly outperformed MLP and CNN.
2. **Less Performing Models:**
   - **MLP** generally had higher RMSE values compared to the other models in most time series, although it performed reasonably well in a few instances:
     - **LNAT2**: MLP had an RMSE of 3.06e+05, making it slightly worse than LSTM and GRU.
     - **LNACT**: MLP had the highest RMSE (4.29e+06) among the models evaluated.
   - **CNN** occasionally underperformed compared to the other models:
     - **LNT2N**: CNN had a higher RMSE (5.35e+07) compared to GRU and LSTM.
     - **LNHEP**: CNN (5.46e+03) performed worse than GRU.

Conclusions:

1. **Best Model for Most Time Series:**
   - **GRU** emerged as the most consistent model, providing the lowest RMSE values across many time series. Its ability to handle sequential dependencies and model long-term relationships likely contributed to this performance.
   - **LSTM** also showed strong performance, particularly for time series with complex temporal dependencies.
2. **Series-Specific Performances:**
   - For time series like **LNEN3** and **LNENC**, **CNN** performed exceptionally well, demonstrating that for some datasets, CNN's ability to capture local patterns in the data is advantageous.
   - **MLP** generally underperformed relative to the other models, highlighting its limitations in handling temporal dependencies compared to the more advanced models (GRU, LSTM, and CNN).

3. **General Trend:**

   - The general trend observed is that **GRU** and **LSTM** are more suitable for time series data with long-term dependencies, while **CNN** can be effective when local patterns dominate.
   - **MLP** is generally less effective for these types of data due to its lack of temporal awareness, making it less suited for time series prediction.

Recommendations for Model Selection:

- **GRU** should be the model of choice for most time series due to its consistent performance across different datasets.
- **LSTM** can be an excellent alternative, especially for time series with more complex or longer dependencies.
- **CNN** may be preferable in specific cases where the data contains prominent local patterns or when simplicity is desired.
- **MLP** should be avoided for time series with strong temporal dependencies, but it may be suitable for simpler datasets or as a baseline model.

This comprehensive analysis helps guide the selection of the best model for time series forecasting tasks, focusing on minimizing prediction errors while considering the specific characteristics of the dataset.

5 Conclusion

In this Master's thesis, various machine learning and deep learning models were evaluated for their effectiveness in forecasting securitization cash flows using time-series data. The dataset, characterized by its high seasonality and frequent spikes, required robust models and resampling techniques to capture both stable and volatile patterns effectively. Models were tested using Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) as evaluation metrics.

Additionally, we applied several statistical tests, such as stationarity tests, normality tests, and ARCH tests. These tests guided the model selection, with the focus being on capturing trends, seasonality, and residuals. SARIMA performed better at predicting seasonality and trends compared to simpler AR models, further reinforcing its applicability to datasets with strong seasonal components.

Resampling Methods:

Conclusion on Resampling Methods: Time Series Split and Expanding Window Validation consistently outperformed other methods, particularly for models like LSTM and GRU that rely on sequential data. These methods maintained the temporal structure of the dataset, ensuring that seasonality and trends were preserved while offering robust model evaluation. Blocked Bootstrapping and Hybrid Resampling also showed potential in more volatile datasets, but their performance was less consistent across all transactions.

Machine Learning and Deep Learning Models:

Conclusion on Models:

- XGBoost emerged as the top-performing model for handling highly volatile and complex transactions, capturing irregular patterns and large spikes with high accuracy.
- LSTM and GRU were highly effective in modeling sequential data, with GRU offering faster computation and LSTM excelling in accuracy for capturing long-term dependencies.
- AdaBoost performed well for low-error, predictable transactions but struggled with high volatility and complex patterns.
- CNN demonstrated surprising strength in low-variance transactions due to its ability to capture local patterns, although it was less effective for more volatile data.
- MLP consistently underperformed across all datasets, making it less suitable for time-series data with complex dependencies.

Final Recommendations:

The best combination of models and resampling methods depends on the volatility and complexity of the transaction data:

- **Highly Volatile Transactions:** XGBoost, GRU, and LSTM are recommended, using Time Series Split or Expanding Window Validation for evaluation.
- **Moderately Volatile Transactions:** LSTM and Gradient Boosting Machine (GBM) performed well with Time Series Split, capturing medium-level volatility.
- **Low Volatility, Predictable Transactions:** AdaBoost and CNN showed strong performance with Time Series Split, handling stable and low-variance datasets effectively.

In conclusion, combining advanced models like XGBoost, LSTM, and GRU with robust resampling techniques such as Time Series Split provides a powerful framework for forecasting securitization cash flows. Future research should focus on hybrid models and deeper optimization of hyperparameters to improve performance across various transaction types.

Statistical Tests and Model Selection

Statistical tests played a crucial role in determining which models were best suited for each transaction type. Around 50% of the transactions were found to be stationary, guiding the selection of more traditional models like ARIMA and SARIMA. Non-stationary transactions indicated a need for more complex models like LSTM and GRU, which are better equipped to handle temporal dependencies and trends.

The normality and ARCH tests revealed that none of the transactions followed a normal distribution, nor did they exhibit conditional heteroscedasticity, further validating the decision to rely on advanced models such as XGBoost and GRU for volatile datasets.

SARIMA and Decomposition of Time Series

SARIMA proved to be the most effective in capturing seasonality, especially when the time series data was decomposed into its cycle, trend, seasonality, and residual components. While simpler models like AR struggled to predict these elements, SARIMA demonstrated better performance, especially in transactions with strong seasonal patterns. However, the complexity of SARIMA required careful tuning of its parameters, making it computationally expensive compared to other models.

Final Thoughts on Model Performance and Resampling Techniques

Overall, this thesis highlights the importance of selecting the right combination of models and resampling methods based on the characteristics of the dataset. For complex and volatile transactions, models like XGBoost, GRU, and LSTM are highly recommended. Resampling methods such as Time Series Split and Expanding Window Validation preserve the temporal structure of the data and ensure accurate model evaluation.

The study underscores the need for an iterative approach to model selection, with a strong focus on statistical testing and decomposition of time-series data. Future research should explore hybrid modeling techniques and advanced tuning strategies to further improve the forecasting accuracy for securitization cash flows.

Appendix

Test Graphs

Serie	p-value	Conclusion
LNACT	0.000277	Non-stationary
LNAAE	0.0311	Non-stationary
LNATO	0.00173	Non-stationary
LNAT2	4.5e-05	Non-stationary
LNACC	0.987	Stationary
LNAC6	0.00327	Non-stationary
LNDOL	5.3e-05	Non-stationary
LNEDI	0.773	Stationary
LNEDN	0.966	Stationary
LNENT	0.0339	Non-stationary
LNENE	0.0086	Non-stationary
LNENM	0.966	Stationary
LNEFM	0.994	Stationary
LNENP	0.99	Stationary
LNENG	0.521	Stationary
LNENI	0.00422	Non-stationary
LNENC	1	Stationary
LNEN3	1	Stationary
LNGIV	0.945	Stationary
LNGSI	0.243	Stationary
LNHET	0.575	Stationary
LNHEC	0.634	Stationary
LNHEP	7.1e-05	Non-stationary
LNHER	0.00289	Non-stationary

Stationarity Test

Serie	p-value	Conclusion
LNACT	1.18e-16	Not normal
LNAAE	1.17e-17	Not normal
LNATO	4.54e-17	Not normal
LNAT2	8.08e-17	Not normal
LNACC	8.2e-21	Not normal
LNAC6	2.63e-16	Not normal
LNDOL	1.8e-15	Not normal
LNEDI	3.47e-13	Not normal
LNEDN	4.09e-14	Not normal
LNENT	3.79e-14	Not normal
LNENE	2.53e-13	Not normal
LNENM	1.42e-18	Not normal
LNEFM	1.21e-19	Not normal
LNENP	5.33e-10	Not normal
LNENG	2.33e-09	Not normal
LNENI	8.84e-17	Not normal
LNENC	3.39e-22	Not normal
LNEN3	4.03e-22	Not normal
LNGIV	8.21e-16	Not normal
LNGSI	4.45e-15	Not normal
LNHET	7.72e-18	Not normal
LNHEC	1.44e-18	Not normal
LNHEP	1.33e-19	Not normal
LNHER	7.07e-15	Not normal
LNT2N	5.72e-20	Not normal
LNTI3	3.14e-19	Not normal
LNTIC	4.73e-19	Not normal

Shapiro-Wilk Test

Serie	p-value	Conclusion
LNACT	0.68	No conditional heteroscedasticity
LNAAE	0.89	No conditional heteroscedasticity
LNATO	0.75	No conditional heteroscedasticity
LNAT2	0.16	No conditional heteroscedasticity
LNACC	0.98	No conditional heteroscedasticity
LNAC6	0.02	Conditional heteroscedasticity present
LNDOL	0.46	No conditional heteroscedasticity
LNEDI	0.96	No conditional heteroscedasticity
LNEDN	0.96	No conditional heteroscedasticity
LNENT	1	No conditional heteroscedasticity
LNENE	0.88	No conditional heteroscedasticity
LNENM	1	No conditional heteroscedasticity
LNEFM	1	No conditional heteroscedasticity
LNENP	1	No conditional heteroscedasticity
LNENG	0.71	No conditional heteroscedasticity
LNENI	0.23	No conditional heteroscedasticity
LNENC	1	No conditional heteroscedasticity
LNEN3	1	No conditional heteroscedasticity
LNGIV	1	No conditional heteroscedasticity
LNGSI	0.32	No conditional heteroscedasticity
LNHET	0.97	No conditional heteroscedasticity
LNHEC	0.98	No conditional heteroscedasticity
LNHEP	0.02	Conditional heteroscedasticity present
LNHER	0.56	No conditional heteroscedasticity
LNT2N	1	No conditional heteroscedasticity
LNTI3	0.97	No conditional heteroscedasticity
LNTIC	0.91	No conditional heteroscedasticity

ARCH Test

Summary of Tests: Stationarity, Normality, and Heteroscedasticity

You can find the full code on my GitHub repository at: `https://github.com/driboulet/CSO`

References

1. Scikit-learn developers. (n.d.). Cross-validation: evaluating estimator performance. Retrieved from Scikit-learn: Cross-validation

2. OTexts. (n.d.). Time Series Cross-Validation. Retrieved from OTexts: Time Series Cross-Validation

3. Brownlee, J. (n.d.). How to Backtest Machine Learning Models for Time Series Forecasting. Retrieved from Machine Learning Mastery: Backtest Time Series Forecasting

4. Packt Publishing. (n.d.). Cross-validation strategies for time series forecasting tutorial. Retrieved from Packt Publishing: Cross-validation strategies

5. Monte Carlo cross-validation for selecting a model and estimating the prediction error in multivariate calibration. Retrieved from ResearchGate: Monte Carlo cross-validation

6. Scikit-learn developers. (n.d.). *sklearn.linear_model.ElasticNet*. Retrieved from Scikit-learn: ElasticNet

7. Vcerq. (n.d.). 9 Techniques for Cross-Validating Time Series Data. Retrieved from Medium: 9 Techniques for Time Series Cross-Validation

8. Hyndman, R. (n.d.). Time series cross-validation. Retrieved from Rob Hyndman: Time Series Cross-Validation

9. Brownlee, J. (n.d.). A Gentle Introduction to k-fold Cross-Validation. Retrieved from Machine Learning Mastery: k-fold Cross-Validation

10. Yen Nhi. (2023). A Guide to Time Series Models in Machine Learning: Usage, Pros, and Cons. Retrieved from Medium: Time Series Models in Machine Learning

11. Crédit Agricole Corporate & Investment Banking. (n.d.). *CSO panorama.pdf*.