Forecasting Foreign Currency Inflow and Outflow Amount: Case Study Based On Customer Base Of A Leading Commercial Bank In Sri Lanka

University of Colombo

Industry Research Project

Department of Statistics

Applied Statistics Four-Year Degree Program

J.M.D. Madushan Jaya Sri – s14379

05/2023

Certificate of Originality

This study is entirely original work on my part and has never been submitted to a university or other academic institution for a degree. To the best of my knowledge, it doesn't contain any content that has been published or written by someone else unless it's explicitly recognized.

Candidate: J.M.D. Madushan Jaya Sri

Signature: 

Date: 07/05/2023

This is to certify that this dissertation is based on the work carried out by J.M.D. Madushan Jaya Sri under my supervision.

Academic Supervisor:

Avanthi Saumyamala

Lecturer (Probationary)

Department of Statistics.

Faculty of Science

University of Colombo

Signature: ……

Date: 07/05/2023

Industrial Supervisor:

Mahesh Weerakoon

Machine learning Engineer

Department of IT (AI and analytics)

Signature: 

Date: 07/05/2023

Acknowledgement

First, I wish to express my gratitude to Dr. A.A Sunethra for providing a chance of having to do an internship at this bank. Then I have to thank Mr. Chaminda Abeysinghe, the head of the AI and Analytics division for selecting me as an internship trainee. Working as a trainee data analyst was an interesting thing. During these six months of internship, I learnt about big economic data handling and many more areas of dealing with such data including machine learning techniques. Especially, how to extract data from data warehouse and analyze, modeling, forecasting them using different platforms such as Python, Power BI, etc.

I want to thank Mr. Mahesh Weerakoon, my industrial mentor, and Mrs. Avanthi Saumyamala, my academic mentor, for their greatest help in this project.

On the bank side, I am grateful to Mr. Vimukthi Perera (Manager-data engineering/ETL), Mr. Gopi Doraisamy (AI - engineer) and all other staff members who worked with me and also, I must be thankful to all the university lecturers who taught me. Specially, Dr. H. A. S. G. Dharmarathne, Dr. K.A.D. Deshani and Mrs. Hasani S. Karunarathna for their kind support to conduct this project.

Finally, I would extend my gratitude towards my family members and friends for their support in carrying out this work successfully.

Abstract

A country's economic success is greatly influenced by the inflows and outflows of foreign currency, which are important economic activity indicators. An outflow can reduce a nation's foreign reserves and have a negative impact on the balance of payments, whereas an inflow can increase foreign reserves.

The goal of this project is to analyze a bank's foreign currency inflow and outflow trends and to choose the best forecasting model to predict these patterns. An extensive dataset with historical information (2018 - 2023) on foreign exchange rates and transactions was used in the study. The data were cleaned, preprocessed, and analyzed using a variety of statistical approaches, including data visualization, statistical inference, and time series analysis.

A thorough review of the data yielded a number of important conclusions. First, there were no clear seasonal or cyclical patterns in both the inflow and outflow of foreign currency. Second, the analysis showed a significant link between changes in the exchange rate, average fixed deposit interest rate, inflation rate with inflows and outflows of foreign cash. Thirdly, the deep neural network models have proven successful in forecasting future outflow amounts in USD. The Long Short-Term Memory (LSTM) model was found to be the best model for forecasting future inflows and outflows of foreign currency after a thorough analysis of several forecast models, including ARIMA, LSTM, and VAR with exogenous variables. Here the measurement for choosing best model was Root Mean Squared Error (RMSE).

In addition, the study revealed that the USD rate variable was not an important variable in forecasting the foreign currency inflow and outflow amount while the all other exogenous variables - Exchange rate, Average fixed deposit interest rate, Inflation rate play significant roles in forecasting.

There were different currency types. Among them major inflow and outflow currency types were, USD, AUD, EURO, JPY and GBP etc. and there were some client categories such as INDIVIDUALS, SME AND LARGE COPERATES etc. the bivariate

and multivariate analysis proved that a category with fewer transactions can be in charge of a sizable transfer of funds.

Overall, the study's findings provide information on the bank's foreign exchange inflow and outflow trends and emphasize the value of precise forecasting for risk management and strategic decision-making. The LSTM model offers a better accuracy for forecasting future foreign exchange transactions and can assist the bank in managing its exposure to currency risk and streamlining its foreign exchange operations.

Key words:

ARIMA, ARIMAX, SARIMAX, GRU, LSTM, VARX, RMSE

Table of contents

List of figures

List of tables

1. Introduction

1.1 Background

The inflow and outflow of a country's foreign money can significantly affect its economic performance as the world economy becomes more interconnected. The exchange of currencies is essential to the global economy because it enables trade and commerce between organizations and individuals in other nations. Exchanging one currency for another is known as currency exchange, and supply and demand in the market determine the exchange rate. Economic, political, and social factors can affect the exchange rate, which can have a big impact on the entry and outflow of foreign currency.

The movement of foreign currency into and out of a nation is referred to as currency inflow and outflow. Inflow and outflow are terms used to describe the amount of foreign currency entering and leaving a nation, respectively. Examples of inflows and outflows include exports and foreign investments [1].

Foreign exchange inflows and outflows are crucial economic activity indicators that have a big impact on a nation's economic success. Outflow can cause a country's foreign reserves to decline and have a detrimental effect on the balance of payments, whereas inflow can boost foreign reserves. Thus, it is essential for nations to effectively regulate the entrance and outflow of foreign cash. For banks to effectively manage their own financial performance and risk, they must comprehend and analyze the inflow and outflow of foreign money from their customer base. Foreign exchange risk exposure, profitability, and liquidity of a bank can all be impacted by the inflow and outflow of foreign currency.

Therefore, accurate forecasting of foreign currency inflow and outflow can help banks make informed decisions about financial planning and risk management. Additionally, accurate predictions can improve customer satisfaction and retention by offering appropriate services and products.

When considering the built model, the model needs to be monitored and modified frequently because it is based on previous data and might not account for unforeseen events or changes in the economic environment.

1.2 Objectives

The first objective of the project is to identify the special aspects of foreign currency inflow and outflow patterns. Then it allows to update policies and adjust constraints by considering individual customer behavior. The next aim is to develop a forecasting model for inflow and outflow amounts. Then it will be helpful to suggest decisions based on the future periods of inflow and outflow (monthly, quarterly, and annually).

1.3 Description of the data

The dataset used, was about the foreign currency income and outcome transactions for the last 5 years starting from 2018. The initial incoming foreign exchange dataset was consisted of 524113 transactions and 43 attributes. And there was another dataset which was contained the currency conversion rates. It had 06 attributes and 8902 records. Also, there were two more datasets as inflation rate dataset and interest rates dataset. Except the initial dataset, all 03 other datasets were monthly data. And inflow dataset was a daily recorded one.

The selected variables from all 04 datasets were as follows,

- Initial foreign transactions (inflow and outflow) dataset. *(daily)*

Table 1: Inflow and Outflow transactions

Variable Name	*Description*	*Variable Type*
CIF_ID	Customer Id	None type (unique for identification of a customer)
TRAN_DATE	Transaction Date	Date-time
TRAN_AMT	Transaction Amount (inflow / outflow)	Continuous
BASLE_CODE_DESC	Related business category	Categorical
CURRDESC	Currency type	Categorical

Here it was not mentioned all 43 variables, since almost 35 variables are not related to the scenario. Therefore, the most important attributes were given by the bank to conduct the project.

Note:

Basel Types, as used in the context of banking, describe several clientele groups that a bank might have based on their size, line of business, and creditworthiness. By limiting its risk exposure, establishing lending guidelines and interest rates, and adhering to legal obligations, the bank can benefit from these categories.

To classify customers according to their size, financial demands, and credit risk, the banking industry frequently uses categories like Individuals, SMEs, Microfinance, Large Corporates, Middle Market Corporates, Staff, and Corporates.

The credit standards, interest rates, and borrowing caps that apply to each Basel Type may vary. As a result, knowing how much money comes in and goes out of each of these categories can help the bank determine prospective growth areas, evaluate credit risk, and create focused marketing campaigns.

- Interest Rates dataset. *(monthly)*

Table 2: Interest Rates (monthly)

Variable Name	Description	Variable Type
AVG (T. FDINT)	Average Monthly fixed interest rate	Continuous
FDMONYEAR	Month and year	Date-time

- Inflation Rates dataset. *(monthly)*

Table 3: Inflation Rates (monthly)

Variable Name	Description	Variable Type
INF_Y_M	Month and year	Date-time
M_inf_rate	Monthly Inflation Rate	Continuous

- Currency Conversion Rates dataset. (monthly)

Table 4: Exchange Rates (monthly)

Variable Name	Description	Variable Type
CURRDESC	Currency type	Categorical
REV_RATE	Converted rate	Continuous
CRRYEARMON	Month and year	Date-time
CONVERT_DATE	Date	Date-time

1.4 Data collection procedure

Initially, the inflow and outflow datasets were taken by the data warehouse. In order to extract the data, the SQL queries in Oracle data base was used.

The inflation rates for the considered period (2018 – 2022) were taken from the Central Bank of Sri Lanka. The site https://www.cbsl.gov.lk/ was used.

Additionally, the customer details were extracted from the WHCIF table used in the Finacle database. That was given by the bank.

2. Theory and Methodology

2.1 Distribution plots

An illustration of a dataset's distribution in a graph is called a distribution plot. It displays how frequently or how many observations fall into certain intervals or bins of a continuous variable. The terms density plot and kernel density plot are also used to describe a distribution plot.

Using a histogram, which is a bar graph that displays the frequency of data points that fall into each bin, is the most typical method of visualizing a distribution plot. A kernel density plot, which displays a smooth estimate of the data's probability density function (PDF), is another technique to describe the distribution of a variable.

To provide extra details about the data distribution, such as the median, quartiles, and outliers, distribution plots can also incorporate additional elements including box plots, violin plots, and rug plots [2] in addition to histograms and kernel density plots.

In this instance, distribution plots were drawn for the transaction amount (inflow / outflow) according to the kind of currency.

2.2 ACF plot

The term "Autocorrelation Function" (or "ACF") refers to a graphical tool that is used to display the autocorrelation of a time series. A time series' autocorrelation is a measurement of its correlation with a lagged version of itself.

The lag (k) at which the correlation is computed is shown on the x-axis of an ACF plot, while the y-axis displays the correlation coefficient (r) between the time series and its lagged variants. A lag of 0 denotes the correlation between the series and the current time step, a lag of 1 the correlation between the series and its previous time step, a lag of 2 the correlation between the series and its second previous time step, and so on.

The ACF plot is a helpful tool for identifying autocorrelation patterns in a time series. For instance, if there is a high positive correlation at lag 1, it is likely that the time series can be adequately described using an autoregressive (AR) model since the value at the previous time step is a good predictor of the current value. The time series may be better characterized using a moving average (MA) model if there is a substantial negative

correlation at lag 1. This indicates that the value at the previous time step is a poor predictor of the current value.

2.3 PACF plot

A PACF plot, which stands for "Partial Autocorrelation Function," is a visual tool for displaying the partial autocorrelation of a time series. The correlation between a time series and a lagged version of itself is measured by partial autocorrelation, which accounts for any intermediate lags.

The partial correlation coefficient (r) between the time series and its lag versions is displayed on the y-axis of a PACF plot, while the lag (k) at which the partial correlation is computed is displayed on the x-axis. A lag of 0 represents the partial correlation between the series and the current time step, a lag of 1 represents the partial correlation between the series and its previous time step, a lag of 2 represents the partial correlation between the series and its second previous time step, and so on.

A helpful tool for determining the order of an autoregressive (AR) model for a time series is the PACF plot. The time series may be well-modeled using an AR model with that order if there is a large decline in partial autocorrelation at a certain lag.

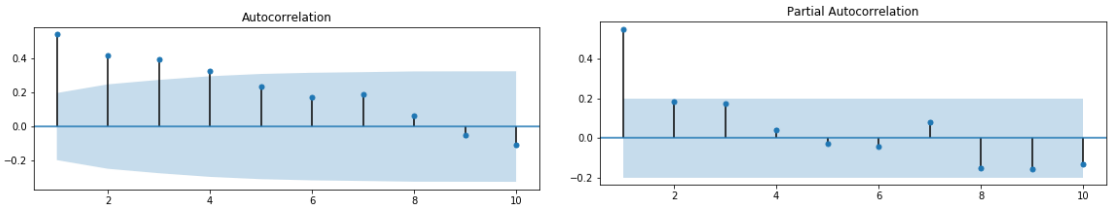

Figure 2-1 :ACF and PACF plots

2.4 Stationary vs non-stationary

A stationary time series is one whose statistical characteristics, such as mean, variance, and autocorrelation, do not change over time in the context of time series analysis. In other words, the time series' statistical characteristics remain consistent over time.

On the other hand, a non-stationary time series is one whose statistical characteristics fluctuate with time. For instance, a non-stationary time series' variance or mean may change over time, or both, depending on the circumstances. Trends, seasonal patterns, or other patterns in the data might cause non-stationarity.

It is crucial to remember that the majority of time series found in real-world applications are non-stationary. As a result, before using time series modeling techniques like ARIMA or SARIMA, a non-stationary time series must frequently be converted into a stationary one. Various methods, including differencing, log-transformations, and seasonal differencing, can be used to accomplish this.

Since the statistical characteristics of a stationary time series do not change over time, it is simpler to spot patterns and linkages in the data. This makes stationary time series easier to model and forecast. If there is a need of forecast for a non – stationary series, then it is recommended to make the series stationary and forecast for it and at the end of the series, the seasonal and trend components are added if there any.

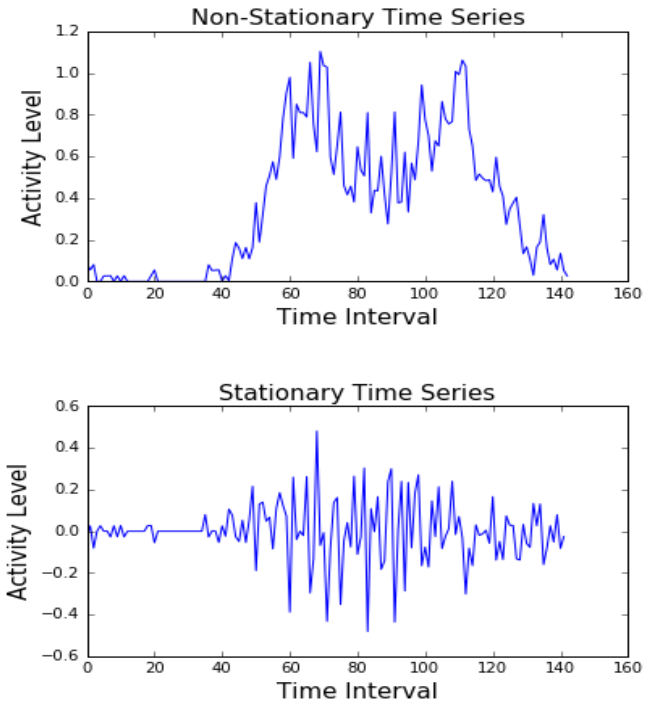

Figure 2-2: Stationary vs non- stationary

2.5 White noise

A white noise series is a stationary time series used in time series analysis that contains uncorrelated, normally distributed random variables with zero means and constant variation for each. White noise, which is a signal with equal power at all frequencies, is a term used in signal processing.

In time series analysis, the baseline or null model is frequently used to represent a white noise series. Any patterns in a time series can be attributed to external elements, such as trends or seasonality, since white noise has no correlation or pattern.

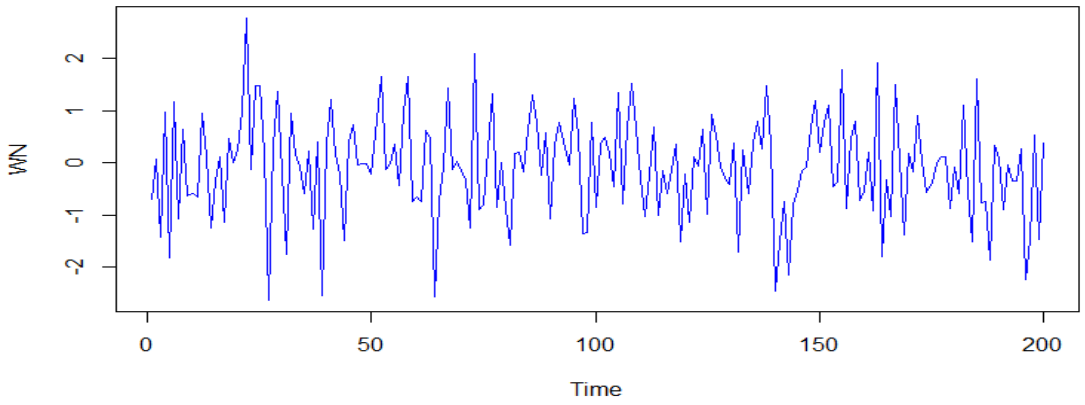

Figure 2-3: White noise series

2.6 Checking stationarity

2.6.1 Augmented Dickey-Fuller (ADF) test

In order to decide whether or not a time series is stationary, statisticians use the Augmented Dickey-Fuller (ADF) test. The fact that the series' statistical characteristics remain constant throughout time is known as stationarity, which is a crucial characteristic of time series data.[3]

The null hypothesis of this test is that the series is non-stationary which implies that it has a unit root. The alternative hypothesis is that the series is stationary. Null hypothesis is rejected if the ADF test statistic is less than the critical value. Then the series is considered as a stationary. If the test statistic is greater than the critical value, then the series is considered as a non- stationary series.

The ADF test is widely used in econometrics and finance to test for the presence of unit roots in time series data. It is also used in other fields where time series data is important, such as meteorology and engineering.

2.6.2 Kwiatkowski-Phillips-Schmidt-Shin (KPSS)

The statistical test KPSS is used to assess whether the time series is stationary or not. This test is complementary to the Augmented Dickey-Fuller test. That means, in ADF test it checks whether the series has a unit root and if exists then conclude that the series is non – stationary. But in the KPSS test, if the series has a unit root, then it concludes that the series is a stationary series. Because the null hypothesis in KPSS test is, the series is a stationary when it exists a unit root. If the test statistic is less than the critical value, then the null hypothesis is not rejected, and the series is considered as a stationary one.

For a more thorough evaluation of stationarity, the KPSS test is frequently used with the ADF test. The time series is more likely to be truly stationary if it passes both tests for stationary behavior. Further research may be required to ascertain the true nature of the series if one test implies stationarity while the other indicates non-stationarity.

2.7 Root Mean Squared Error (RMSE) and Overfitting

Root mean squared error or Root mean squared deviation is widely used in supervise machine learning to evaluate the quality of predictions. For each data point, the residual (difference between forecast and truth) is calculated, and the norm of the residuals are taken and finally mean of those are taken to find the root mean square error (RMSE). Since RMSE requires and uses correct measurements at each projected data point, supervised learning algorithms frequently use RMSE.

Root mean square error can be expressed as,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}}, \qquad \text{---- (01)}$$

where N is the number of data points, y(i) is the i^{th} measurement, and $\hat{y}(i)$ is its corresponding prediction. [4]

In machine learning, it is extremely helpful to have a single number to judge a model's performance, whether it be during training, cross-validation, or monitoring after deployment. Root mean square error is one of the most widely used measures for this.

The overfitting of a model can be judged using this RMSE value. The RMSE values for training and testing set separately. If the two values have some significance difference, then the model is considered to be as an over fitted one.

2.8 Modelling theories

2.8.1 Autoregressive Integrated Moving Average model (ARIMA)

ARIMA model is combination of Autoregressive (AR) and Moving Average (MA), and differencing process to time series data. General form of non-seasonal ARIMA model as follows [5].

$$\emptyset_p(B)(1-B)^d Y_t = \theta_0 + \theta_q(B)a_t \quad \text{---- (02)}$$

While general form of seasonal ARIMA model is [5,6]

$$\Phi_p(B^S)\emptyset_p(B)(1-B)^d(1-B^S)^d Y_t = \theta_q(B)\Theta_q(B^S)a_t \quad \text{---- (03)}$$

Where

$\emptyset_p(B)$: $coefficient\ of\ non-\ seasonal\ AR\ with\ order\ p, \emptyset_p(B) = 1 - \emptyset_1(B) - \emptyset_2(B^2) - \cdots - \emptyset_p(B^P)$

$\theta_q(B)$: $coefficient\ of\ non-\ seasonal\ MA\ with\ order\ q, \theta_q(B) = 1 - \theta_1(B) - \theta_2(B^2) - \cdots - \theta_q(B^q)$

$(1-B)^d$: $operator\ for\ differencing\ of\ order\ d$

$\Phi_p(B^S)$: $coefficient\ of\ seasonal\ AR\ with\ order\ p, \Phi_p(B^S) = 1 - \Phi_1(B^S) - \emptyset_2(B^{2S}) - \cdots - \Phi_p(B^{PS})$

$\Theta_Q(B^S)$: $coefficient\ of\ seasonal\ AR\ with\ order\ p, \Theta_Q(B^S) = 1 - \Theta_1(B^S) - \Theta_2(B^{2S}) - \cdots - \Theta_Q(B^{QS})$

$(1-B^S)^D$: $operator\ for\ differencing\ of\ seasonal\ S\ with\ order\ D$

a_t: $residual\ value\ at\ t.$

2.8.1.1 Exogenous Variables

The exogenous variable [7], X, can be any variable in which we are interested. It could be a measurement that changes over time, such as the inflation rate or the cost of another index. is a variable that categorizes the various days of the week. For the specific holiday times, a Boolean accounting is another option. Finally, it might refer to an interaction of various outside causes.

2.8.2 Autoregressive Integrated Moving Average model with exogenous variables (ARIMAX)

The ARIMA model (Autoregressive Integrated Moving Average) and explanatory variables, sometimes referred to as regressors, are used in the time series forecasting technique known as ARIMAX. Autoregressive Integrated Moving Average with Exogenous Variables is referred to as ARIMAX.

In ARIMAX forecasting, the exogenous variables are used to account for external influences that might influence the time series, while the ARIMA model is used to capture the autocorrelation and seasonality in the time series data. These external influences may be non-time series variables, such as economic indicators or weather data, or they may be other time series that are related to the target time series.[8]

The ARIMAX model can be written as:

$$Y(t) = c + \varphi 1Y(t-1) + \ldots + \varphi pY(t-p) + \theta 1e(t-1) + \ldots + \theta qe(t-q) + \beta 1X1(t) + \ldots + \beta kXk(t) + e(t) \quad \text{---- (04)}$$

Where:

- Y(t) is the target time series at time t
- c is a constant
- φ1, ..., φp are the autoregressive coefficients
- θ1, ..., θq are the moving average coefficients
- e(t) is the error term at time t
- X1(t), ..., Xk(t) are the k exogenous variables at time t
- β1, ..., βk are the regression coefficients

The target time series is forecasted using the ARIMAX model, which is computed from past data. Usually, it is assumed that the exogenous factors are known for the forecasting period.

2.8.3 Gated Recurrent Unit model (RNN)

GRU is an improved version of standard recurrent neural network. The so-called update gate[9] and reset gate are used by GRU to address the vanishing gradient issue that plagues a normal RNN. In short, these two vectors determine what data should be sent to the output. They have the unique ability to be trained to retain information from the past without having it fade away over time or to discard information that is unrelated to the forecast.[9]

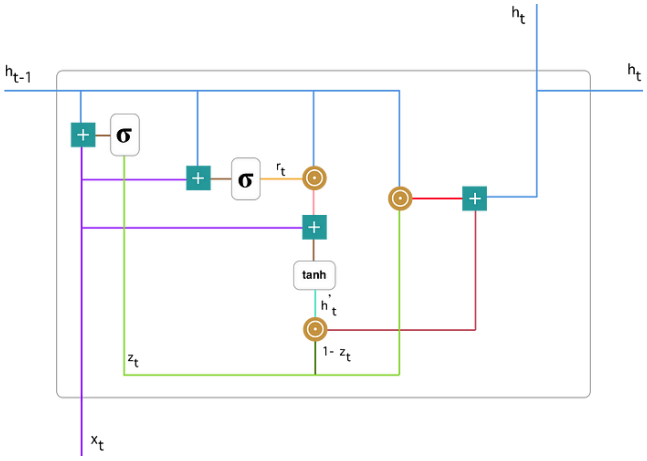

Figure 2-4: Gated Recurrent Unit

Figure 2-5: Notations of GRU

2.8.4 Long Short-Term Memory (LSTM) model

Long Short-Term Memory (LSTM) is a recurrent neural network (RNN) architectural type. Sequential data, including audio, time-series data, and natural language are processed using LSTMs.

Compared to conventional RNNs, LSTMs have a more complex framework with more gates that regulate the information flow across the network. The gates' capacity to selectively remember or forget data from earlier time steps enables the network to hold onto important data for longer periods of time.

An LSTM network's gates are constructed using learnable parameters that are changed using backpropagation through time (BPTT) during training. The LSTM architecture has a number of benefits over conventional RNNs, such as the capacity to handle long-term dependencies in the input data, the capacity to selectively remember or forget information from previous time steps, and the capacity to avoid the issue of vanishing gradients that frequently arises in deep RNNs.[10]

LSTMs deal with both Long-Term Memory (LTM) and Short-Term Memory (STM) and for making the calculations simple and effective it uses the concept of gates.

1. Forget Gate: LTM goes to forget gate and it forgets information that is not useful.
2. Learn Gate: Event (current input) and STM are combined together so that necessary information from STM can be applied to the current input.
3. Remember Gate: LTM information and STM and Event are combined together in Remember gate which works as updated LTM.
4. Use Gate: This gate also uses LTM, STM, and Event to predict the output of the current event which works as an updated STM.

The ability to add "peephole" connections, which enable the gates to directly witness the past memory state in addition to the input data and the present hidden state, is a key feature of LSTMs. This may help the model better understand long-term dependencies.

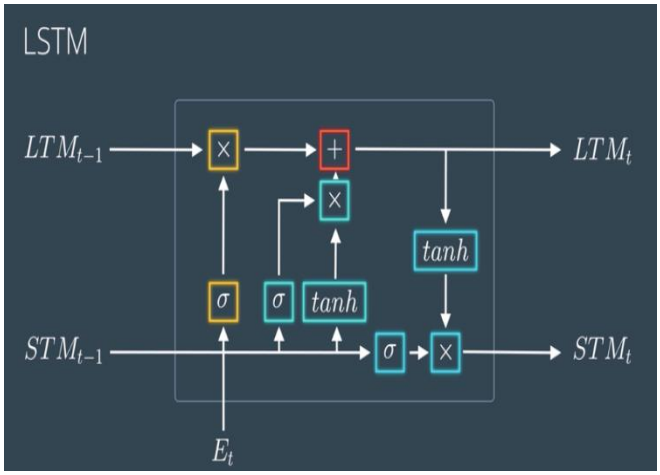

Figure 2-6: LSTM Architecture

Data Normalization

When a neural network uses the gradient descent method to discover a good response, it will face issues such as an unnecessarily long training period or difficulty in finding the optimum solution when each field in the data set has a different scale. Due to this issue, the neural network iterates often before converging, sometimes even unable to converge, which reduces accuracy. So, in order to keep the values of the various dimensions between 0 and 1, Min-Max Normalization is utilized.

2.8.5 Vector Auto Regression (VAR)

A statistical model called a VAR (Vector Autoregression) model is used to examine how different time series variables relate to one another. It states that each system variable's current value is a function of both its own prior values and the prior values of all other system variables.

For example, the system of equations for a VAR (1) model with two time series (variables `Y1` and `Y2`) is as follows.

$$Y_{1,t} = \alpha_1 + \beta_{11,1} Y_{1,t-1} + \beta_{12,1} Y_{2,t-1} + \epsilon_{1,t}$$
$$Y_{2,t} = \alpha_2 + \beta_{21,1} Y_{1,t-1} + \beta_{22,1} Y_{2,t-1} + \epsilon_{2,t} \quad \text{---- (05)}$$

The second order VAR (2) model for two variables would include up to two lags for each variable (Y1 and Y2).

$$Y_{1,t} = \alpha_1 + \beta_{11,1} Y_{1,t-1} + \beta_{12,1} Y_{2,t-1} + \beta_{11,2} Y_{1,t-2} + \beta_{12,2} Y_{2,t-2} + \epsilon_{1,t}$$
$$Y_{2,t} = \alpha_2 + \beta_{21,1} Y_{1,t-1} + \beta_{22,1} Y_{2,t-1} + \beta_{21,2} Y_{1,t-2} + \beta_{22,2} Y_{2,t-2} + \epsilon_{2,t} \quad \text{---- (06)}$$

[11]

2.9 Methodology

The methodology is a crucial aspect of any research project, as it determines the approach taken to gather and analyze data.

This approach involved several key steps, collecting data, and analyzing the results. In the following sections, a detailed description of each step, as well as the tools and techniques used to execute them are explained.

2.9.1 Data preprocessing and wrangling steps

- The dataset of inflow was loaded into Spyder IDE: This step involves importing the dataset of inflow into the Spyder IDE, which is an Integrated Development Environment (IDE) used for data analysis and programming. Then null columns were filtered, and necessary columns were transferred into required data types. For example, if there is a date column, it may be converted to a "datetime" data type to allow for easier manipulation and analysis. There were some columns which had no values thus those were removed. After that, duplicate rows were removed from the dataset to avoid any biases that may be introduced by having multiple identical entries. There were 91 such duplicates.
- Then the transaction date ("TRAN_DATE") column was used to extract the year, month, and day separately. These values are then stored in new columns in the dataset itself. Extracted year, month and days were assigned into new columns as "T_year","T_Month", and "T_Day" respectively.
- As the next step, the exchange rates dataset was loaded and removed duplicates and null values. There were 4426 duplicates. And a new column named "CRRYEARMON" was created in the inflow dataset by combining the columns "T_year" and "T_Month". Then the column named "TRAN_CRNCY_CODE" in the inflow dataset was renamed into "CURRDESC".
- Then it was needed to be merged the exchange dataset and inflow dataset. To do that first, the "CRRYEARMON" column in the exchange dataset is converted into string type and merged it with the inflow dataset using the "CRRYEARMON" and "TRAN_CRNCY_CODE" as composite primary key. New dataset was named as "dataset_new".
- After that, all currencies were converted into Sri Lankan rupees by multiplying the transaction amount of inflow ("TRAN_AMT") and exchange rate ("REV_RATE"). The new values were stored in a new column named as "Tran_in_LKR" in the inflow dataset.
- It was needed to analyze currencies in USD. Because some of the foreign currencies been converted to USD when it comes to Sri Lanka. For some examples, Saudi Riyal, Thai baht and Emirati Dirham are some currencies which are been converted to USD when comes to Sri Lanka. So, the data of USD rates

and dates were extracted from the exchange rate dataset and merged it with the "dataset_new". Here, the USD rates were taken from the column named "REV_RATE" and dates were taken from the column named "CURRDESC" in the exchange rates dataset. The selected rates were saved as "usd_rate" and year – month were saved as "CURRDESC" itself. Then all Sri Lankan rupees were converted into US Dollars by dividing the "Tran_in_LKR" by "usd_rate". The new merged dataset was taken as the "dataset_new_with_USD".

- As the next step, the dataset containing interest rates was loaded and removed any duplicate rows. There were 19 such duplicates. In order to merge this data with the dataset_new_with_USD, the column named "FDMONYEAR" was renamed into "CRRYEARMON" and converted to string type. After that, merging process were done with the dataset_new_with_USD dataset by using the "CRRYEARMON" as primary key. The name of the new merged dataset was "dataset_new_with_USD_and_intr".
- Then the inflation rates dataset was loaded and removed duplicates. Also, the column named "INF_Y_M" was renamed into "CRRYEARMON" and converted to string type. After that, it was merged with the "dataset_new_with_USD_and_intr" by using the column "CRRYEARMON:" as the reference key and that was the final inflow dataset named as "final_inflow_dataset".
- Then EDA was conducted to the above preprocessed dataset.
- The next thing was to get the time series dataset to use for forecasting purposes. In order to do that, "final_inflow_dataset" was grouped by the "TRAN_DATE". When grouping a dataset, it can be given some aggregation functions with that as well. So, in this case, the aggregations were as follows,
    - Amount_in_USD → sum,
    - usd_rate → mean,
    - M_inf_rate → mean,
    - AVG(T.FDINT) → mean,
    - BASLE_CODE_DESC → mode

    And new grouped table was named as DF_grouped.

- Time series dataset should have an equal time interval and corresponding responses. But the dataset that was created previously had not been consisted with the equal time intervals. The reason for that was some dates had been dropped within the considered time range. (01/01/2018 to 31/12/2022). In order to overcome that problem, manually created a data frame including all dates between that end date points. That data frame was named as “dataf”.
- Then the “dataf” and “DF_grouped” datasets were merged together by using the “TRAN_DATE” as reference key and named as “TS_inflow”. Then the null records in the "TS_inflow" were filled by forward and backward filling. This means that missing values were filled in using the previous or next non-null value. This was done to ensure that there were no missing values in the dataset, which could affect the accuracy of our time series forecasting.
- As the second part of the analysis, outflow dataset was loaded in to the spyder IDE and all the above steps were followed in order to get the “final_outflow_dataset” and “TS_outflow”. This involved filtering out null columns, converting data types, removing duplicates, and merging with exchange rate, interest rate, and inflation rate datasets. Finally, four datasets were saved as,
    - final_inflow_dataset.
    - final_outflow_dataset.
    - TS_inflow
    - TS_outflow

2.9.2 Model Forecasting procedure

ARIMA

1. First considered the amount in USD as a single time series and checked its stationarity. (Two tests were used, ADF and KPSS)
2. Since the series was not stationary, then followed the methods in order to make the series stationary. (The trend and seasonal components of the series were broken out and separated.)
3. Plotted the ACF and PACF in order to check the correlation of lag values.

4. Determined the AR and MA terms from the plots and conducted the ARIMA model to the train series by dividing the series in to two parts as train and test set. (ratio – 4:1)
5. At the end of the model, trend and seasonal components were added accordingly and predicted for the test set
6. Finally, model validation was conducted and checked the RMSE value for the train set and test set respectively and determine the goodness of the model.

ARIMAX with Auto- ARIMA

1. In order to perform the model, the exogenous variables were scaled, and then divided the data into train and test set as previously done. (The stationary series was used for the forecasting purpose)
2. Used the Auto-Arima python library to fit the optimum model and finally trend and seasonal components were added.
3. At the end of the model, RMSE values were calculated.

SARIMAX (ARIMA with seasonal and exogenous)

1. To fit the SARIMAX model, the seasonal period was given and fitted the forecasting model to the training set. Finally seasonal and trend components were added.
2. Then RMSE values were calculated and determined whether the model is adequate.

Gated Recurrent Unit model (RNN)

- Initial amount in USD series was made stationary and then all variables were scaled prior to inject to the model.
- After that, the whole dataset was divided into two parts as training and test set. (ratio – 4:1)
- Reshaped the train set in order to input to the neural network.
- Trained the initial model with arbitrary number of GRU layers, batch sizes and inputs for 200 epochs.
- Applied early stopping criteria in order to stop the overfitting the neural network.

- Tried for various number of GRU layers with different units, epochs and batch sizes.
- Predicted for the test set and found the RMSE values for training and test set separately in order investigate whether the model is overfitting or not.

Long – Short Term Model (LSTM)

- Same as the GRU model, but here it was used the LSTM layers instead of GRU layers.
- Following the same procedures followed to GRU forecasting, RMSE values were obtained to the training and test set separately and evaluated the model accuracy.

Vector auto Regression with exogenous (VARX)

- After removing the non -Stationary aspect, the stationary series was divided into two parts as training and test (ratio – 4:1)
- Then the VAR model in statsmodels package in python was used for several lag value. In order to do that, range between 1 to 10 were used.
- By getting the minimum value for AIC and BIC, the best model was chosen and checked the RMSE value for training and test set separately.

Model validation and Residual diagnosis

- At the end of each model building, the corresponding model validation was conducted using the K-Fold cross validation
- In order to check the residuals,
    - Plotted the residuals. – needed to be randomly scattered.
    - Plotted ACF and PACF for residuals. – no correlation lags are preferred.
    - Normal PP plot – should be lied on the line
    - Plotted the histogram of residuals – should be normally distributed.

3. Preliminary Analysis

3.1 Univariate analysis

3.1.1 Distribution plot of inflow and outflow

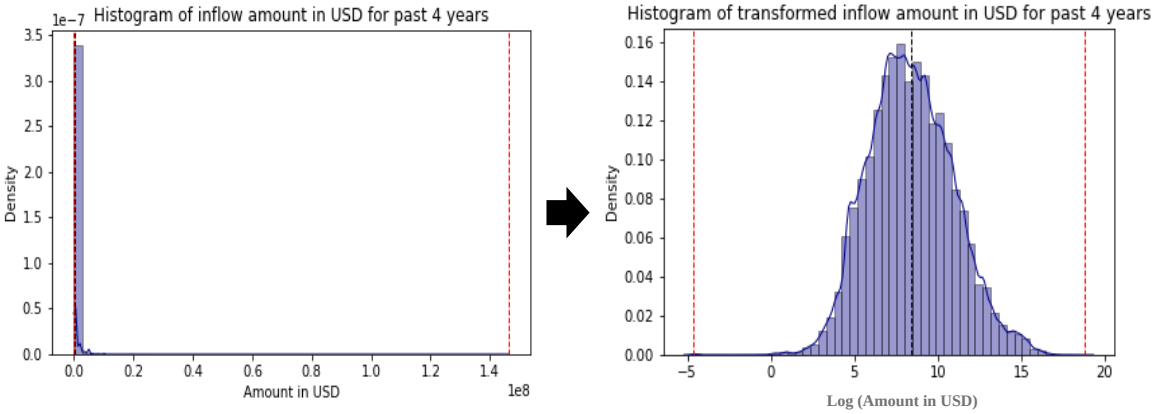

Figure 3-1: Inflow distribution

Figure 3-1 shows the distribution of the amount of inflow in USD. From the left chart, it is clear that the distribution is positively skewed.

The first and second red dotted lines from the left shows the minimum and maximum values respectively. And the black dotted line shows the mean value. Here it is clear that, even there was a very large maximum value, the mean value was closer to the minimum value. Therefore, it can be said that most of transactions were done up to 5×10^6 USD. In order to eliminate the skewedness of the distribution, transformation techniques can be used. The right-side chart shows the transformed distribution of inflow. Here it was used the logarithm of the initial amount of inflow, as the transformation technique. The transformed distribution shows almost normally distributed amounts.

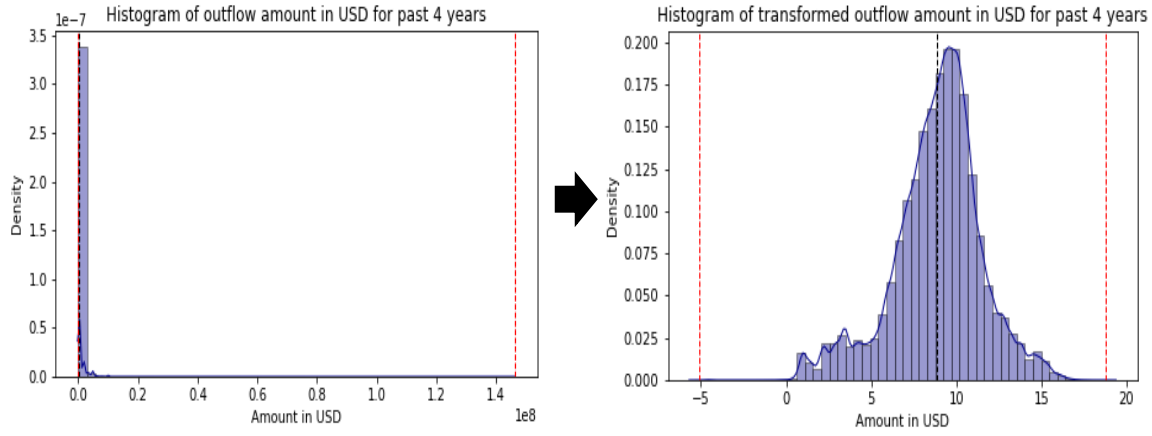

Figure 3-2:Outflow distribution

The outflow amount is displayed in USD in Figure 3-2. The outflow transactions had obviously taken place in the same way as the inflow. Since the distribution was positively skewed, the mean value of the outflow transactions was similarly closer to the lowest value. The distribution has shifted slightly in relation to the inflow after using the transformation approach, but its typical shape has not changed.

Then these transformed values can be used for further analysis and model forecasting.

3.1.2 Line plot of inflow and outflow amount

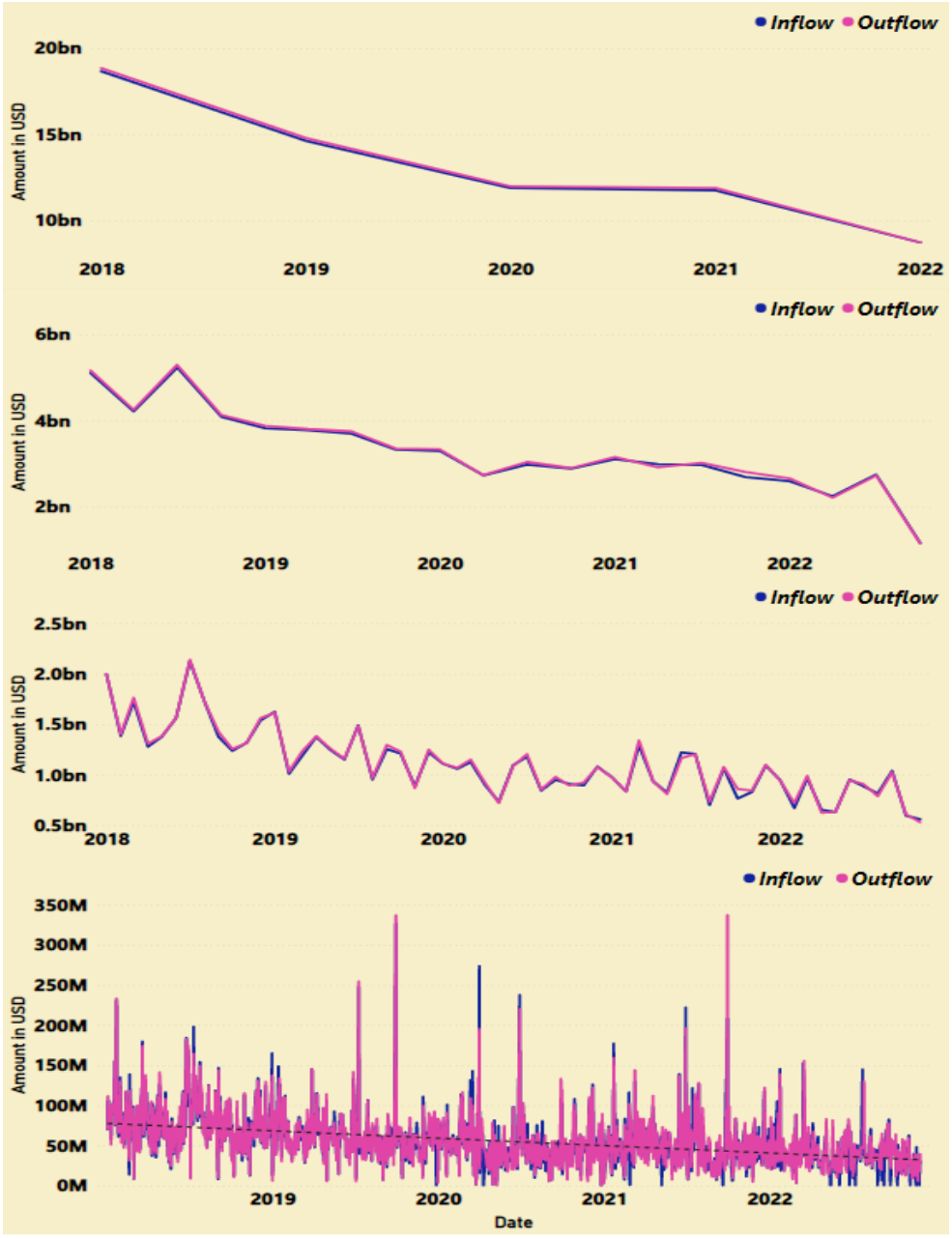

Figure 3-3: Difference between Inflow and Outflow

The trend of the inflow and outflow amounts in USD for the chosen time period of 2018 to 2022 is depicted by the line plot (figure 3-3). The only small changes over time between the two lineages are essentially identical. These variations are slightly more apparent whether the data is examined on a monthly, quarterly, or annual basis, but they are still rather little and have no bearing on the general trend.

Based on the data, it can be said that the inflow and outflow amounts are highly symmetric, with both exhibiting consistent patterns across time. Therefore, additional exploratory data analysis (EDA) may not be required for the combined inflow and outflow numbers. Instead, the focus can be shifted to analyzing either the inflow or outflow amounts in more detail. The similarity of the two lines suggests that the overall inflow and outflow amounts were about equal throughout the selected time period.

3.1.3 Line chart of Inflow time series

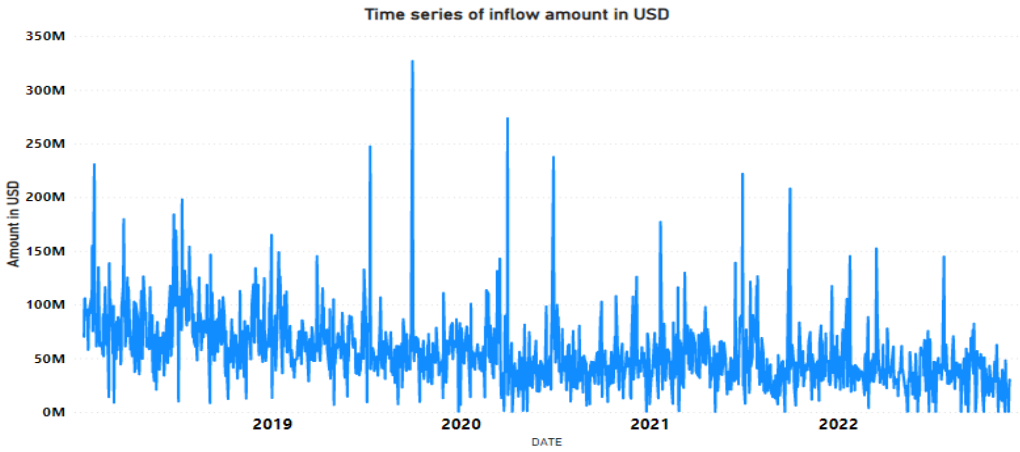

Figure 3-4: Inflow Amount in USD

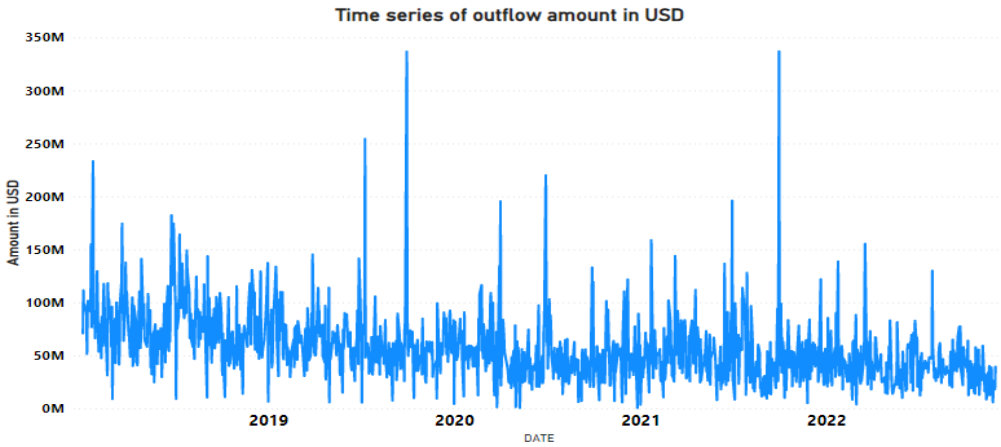

Figure 3-5: Outflow Amount in USD

Even there cannot be clearly seen a trend in the graph at a glance, but when observing carefully, it can be seen that, there is some slightly negative trend when comes to 2023. Also there is no specific seasonal or cyclic pattern. There are random fluctuations which were distributed all over the considered time period.

3.1.4 Number of transactions by Basle types (client categories)

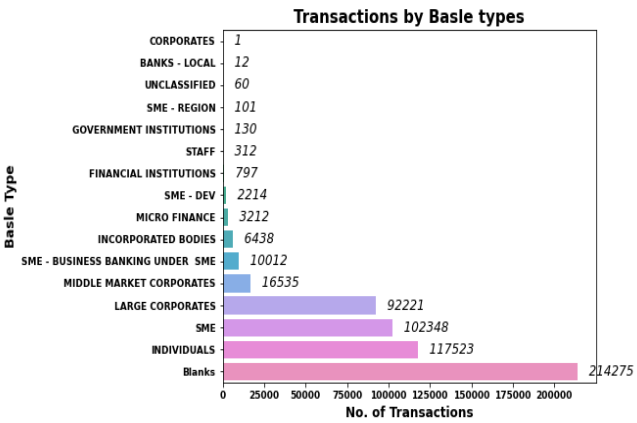

Figure 3-7: No. of Transactions by Basles for inflow

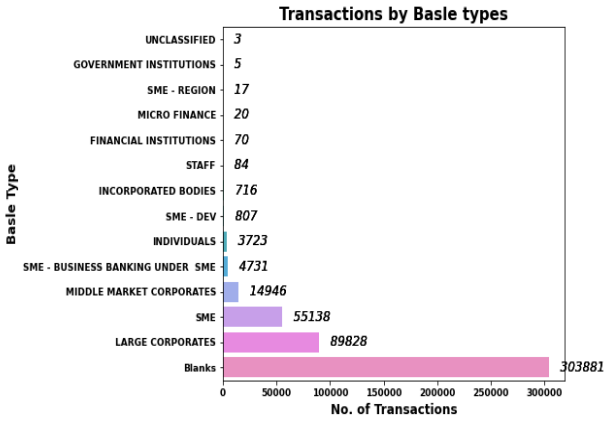

Figure 3-6: No. of Transactions by Basles for outflow

For the customer categories for inflow and outflow transactions in the chosen bank from 2018 to 2022, two bar charts have been made. persons account for the majority of transactions in the inflow chart (117,523), followed by blanks (214,275), which the bank is probably going to classify as persons. Large corporations come in second place with 92,221 transactions, and SMEs come in third with 102,348 transactions. The remaining categories have far fewer transactions.

The outflow chart, on the other hand, demonstrates a varied breakdown of transactions among the customer categories. Large corporations, with 89,828 transactions, are the largest group, followed by SMEs with 55,138 transactions. With 14,946 transactions, middle market corporates are the third largest category. There are fewer transactions in the remaining categories.

The largest groups are arranged differently in the inflow and outflow charts, as can be seen. The largest number of transactions come in from the individuals group, whereas the largest number of transactions leave from the major corporates category. The second-largest category in both charts, SMEs have a significantly higher volume of transactions in the inflow chart. The outflow chart also reveals a greater number of transactions for middle market corporates.

For the bank, these differences in how transactions are distributed among the various client categories on the inflow and outflow charts can be a valuable source of information. For instance, the high volume of transactions coming from individuals in the inflow chart may show that the bank's marketing initiatives are successful in attracting individuals, whereas the high volume of transactions coming from large corporations in the outflow chart may show that the bank needs to improve its rapport with these clients in order to keep them.

3.1.5 Percentages of Retail and Cooperative

It is recommended to divide those categories into two sorts for ease of understanding: retail, which might be considered to be at the individual level, and other cooperative, which is seen to be at the corporate level. Buying and selling foreign currency by individuals or small enterprises for a variety of purposes, such as travel, online shopping, or investing in foreign markets, is referred to as retail customer dealing with foreign currency. Retail customers typically use banks, currency exchange companies, or online marketplaces to swap their local currency for foreign cash. A corporate customer is a company or organization that buys products or services from another company, typically for business or industrial usage as opposed to personal use. Small and large firms alike can be corporate clients, and they can buy a wide variety of goods and services, such as raw materials, tools, software, and consulting services.

The percentage of retail and cooperative transactions during the time period under consideration is shown in the following pie charts.

- Retail – Individual, Staff, Unclassified, Blanks
- Cooperative – Except Retail

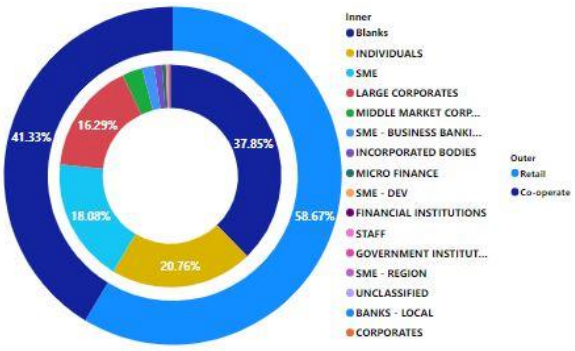

Figure 3-8: Inflow Transactions by Retail and co-operate

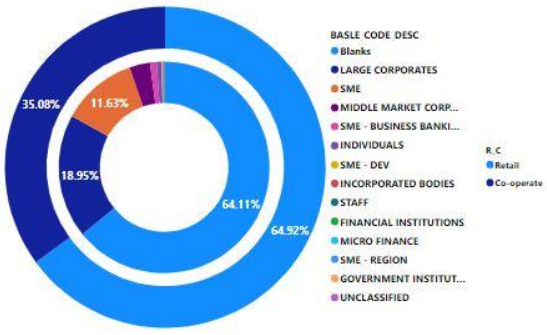

Figure 3-9: Outflow Transactions by Retail and co-operate

Figures 3-8 and 3-9 show that the retail category accounts for the majority of both inflow and outflow. And those percentages are 58.67% inflow and 64.92% outflow. The virtually empty category in the outflow chart is associated with the retail industry specifically. However, the INDIVIDUAL and BLANKS categories in the inflow chart are connected to the retail category. Large corporates and SME transactions, which fall under the category of cooperation, are the most frequent ones among the inflow transactions. And the outflow transactions follow a nearly same pattern.

3.2 Bivariate analysis

3.2.1 Transaction amount in USD by Basle types (client categories)

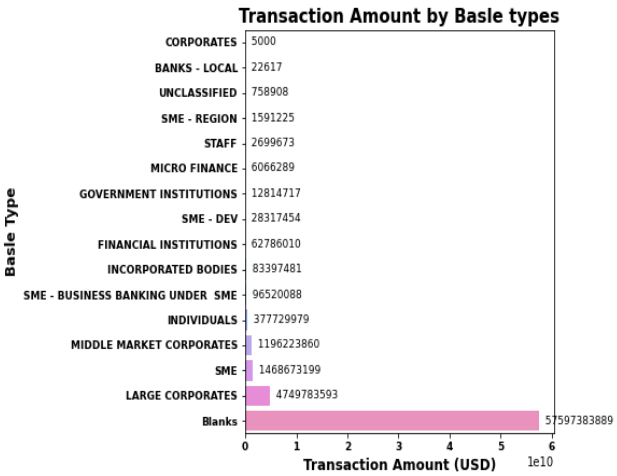

Figure 3-11: Transaction amount of inflow

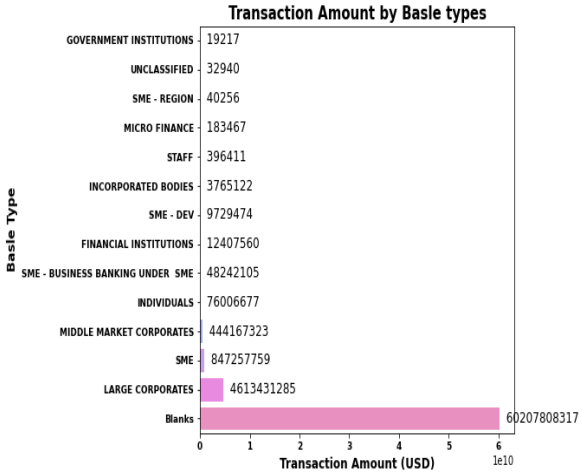

Figure 3-10: Transaction amount of outflow

Figure 3-11's bar chart of inflow transactions demonstrates that the 'Blanks' category had the greatest transaction value, followed by 'Individuals', 'Middle Market Corporates', 'SME', and 'Large Corporates'. The 'Banks - Local', 'Unclassified', and 'SME - Region' categories, on the other hand, have the lowest transaction quantities. The largest transaction amount is about four times bigger than the second-largest transaction amount, demonstrating a huge disparity in transaction amounts between categories.

Figure 3-10's bar graph of outflow transactions demonstrates that the 'Blanks' category had the greatest transaction value, followed by 'Large Corporates,' 'SME,' and 'Middle Market Corporates. The 'SME-Region, ''Unclassified', and 'Government Institutions' categories have the lowest transaction quantities. There is a huge discrepancy in transaction quantities between categories, as seen by the fact that the highest transaction amount is around 10 times more than the second-highest transaction amount.

The order of the categories based on transaction amounts differs between the inflow and outflow bar charts, as can be seen. As an illustration, the 'Large Corporates' category has the second-highest transaction amount in the inflow chart yet the largest transaction amount in the outflow chart. The fourth-highest ranking is held by the category "Middle Market Corporates." When assessing bank transactions, it's crucial to take both transaction counts and quantities into account. Even if a category may have a lot of transactions, that doesn't mean a lot of money is necessarily being transferred through those transactions. On the other hand, a category with fewer transactions can be in charge of a sizable transfer of funds.

For instance, "Large Corporates" had the most transactions overall, followed by "SME" and "Individuals." This shows that although though the "SME" group had the most transactions, the "Large Corporates" category was in charge of a much higher sum of money being transferred.

The variations in the "Micro Finance" category ranks are another intriguing phenomena. Even though it was the sixth biggest category in terms of transaction counts, it was only the eighth highest category in terms of transaction amounts. This would suggest that even though there were a lot of transactions in this category, they tended to be for lesser sums.

3.2.2 Average transactional amount in USD by currency types

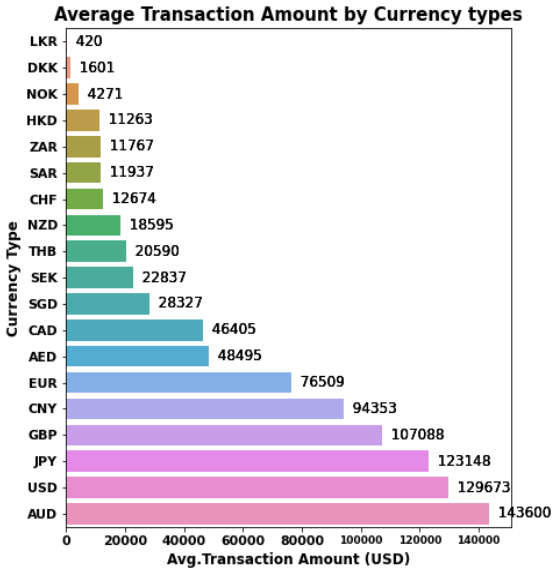

Figure 3-13: Inflow Avg.transaction in USD

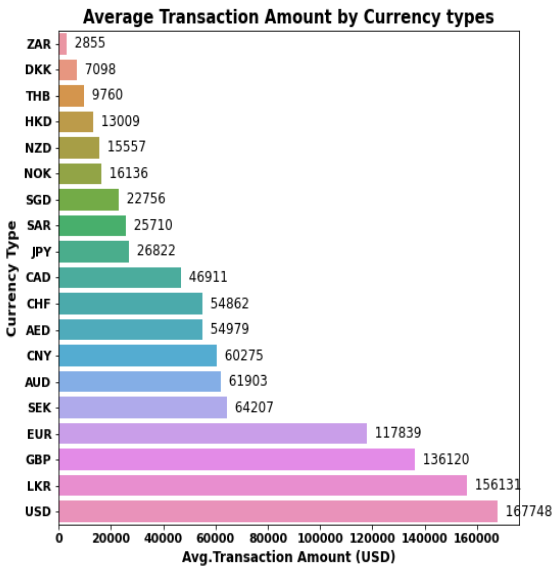

Figure 3-12: Outflow Avg.transaction in USD

The average transaction value in USD for various currency kinds is shown in the inflow bar chart (Figure 3-13). The Australian Dollar (AUD), with an average transaction amount of 143,600.79 USD, has the largest average transaction amount of any currency kind. The Sri Lankan Rupee (LKR), on the other hand, has the lowest average transaction amount with an average of 420.00 USD.

Figure 3-12's outflow bar chart shows the average transaction amount in USD for various currency kinds. With an average sum of 167,748.66 USD, the US Dollar (USD) is the currency type with the greatest average transaction amount. With an average transaction amount of 2,855.82 USD, the South African Rand (ZAR) is the currency type with the lowest average transaction amount.

3.3 Multivariate analysis

3.3.1 stacked bar graph of the average amount in USD by currency type by year

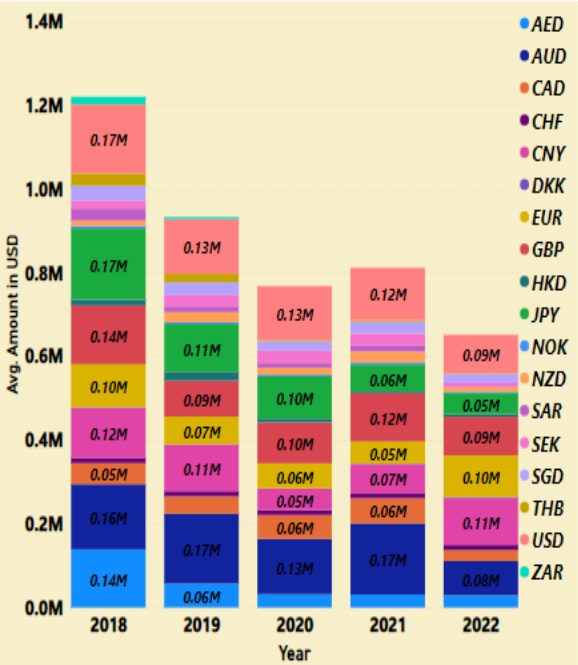

Figure 3-14: Avg. Inflow amount in USD by currency types by year

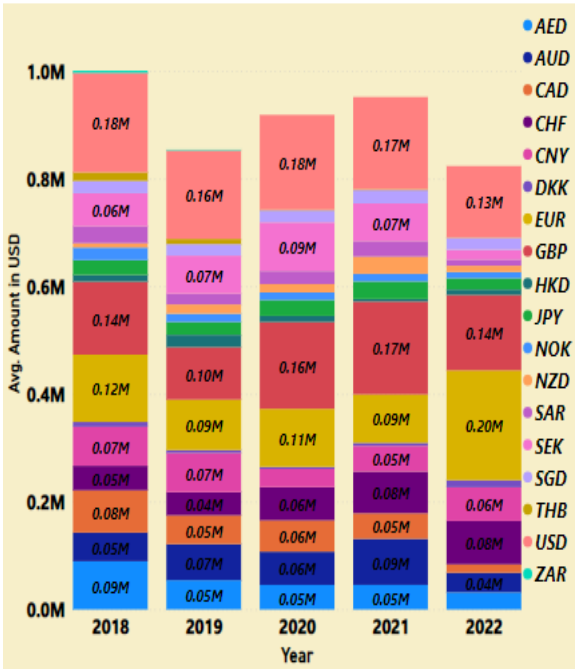

Figure 3-15: Avg. Outflow amount in USD by currency types by year

It is clear from the stacked bar chart of inflow (figure 3-14) that the four major currency types—AUD, GBP, JPY, and USD—have significant average amount in USD. Across all years, the average amount of AUD inflow is consistently high, peaking in 2019.

This suggests that there has been a sizable amount of AUD influx over the years, with a potential increase in 2019 due to certain outside variables or occurrences.

Similar to that, there is a peak in 2018 and a high average amount of GBP influx over all years. This shows that GBP has historically been a preferred currency for inflows and has had an increase in inflows in 2018.

The average JPY inflow is extremely large in 2018 and 2019, declines greatly in 2020 and 2021, and then begins to increase somewhat in 2022. This could be brought on by changes in the market environment, changes in economic conditions, and fluctuations in currency rates.

Last but not least, there is a peak in 2021 and the average amount of inflow in USD is constantly high over all years. This shows that the US dollar has historically been a favoured currency for inflows, with a potential increase in 2021 due to certain outside circumstances or occurrences.

The stacked bar chart, in general, offers helpful insights into the inflow amount data by currency types, emphasizing the substantial inflows in AUD, GBP, JPY, and USD.

The outflow stacking plot (figure 3-15) displays the average outflow for each type of currency from 2018 to 2022. The Y-axis, like the inflow plot, displays the average outflow in millions, while the X-axis displays the years from 2018 to 2022.

The graphic makes it obvious that USD has the biggest average outflow, followed by EUR and GBP. The outflows of these three currencies have been consistently strong over the years, peaking in 2021 for the USD and GBP and in 2018 for the EUR.

Compared to other currency types, the average outflow of Australian dollars is rather low, peaking in 2019. On the other hand, JPY's outflow is higher in 2018 and 2019 before drastically declining in 2020 and 2021 before somewhat increasing in 2022. Overall, the outflow stacked map reveals that USD, EUR, and GBP are the three currencies that are used the most often for outflow transactions, with AUD and JPY having smaller outflow quantities.

3.3.2 Stacked bar graph of the average amount in USD by Basles type by year

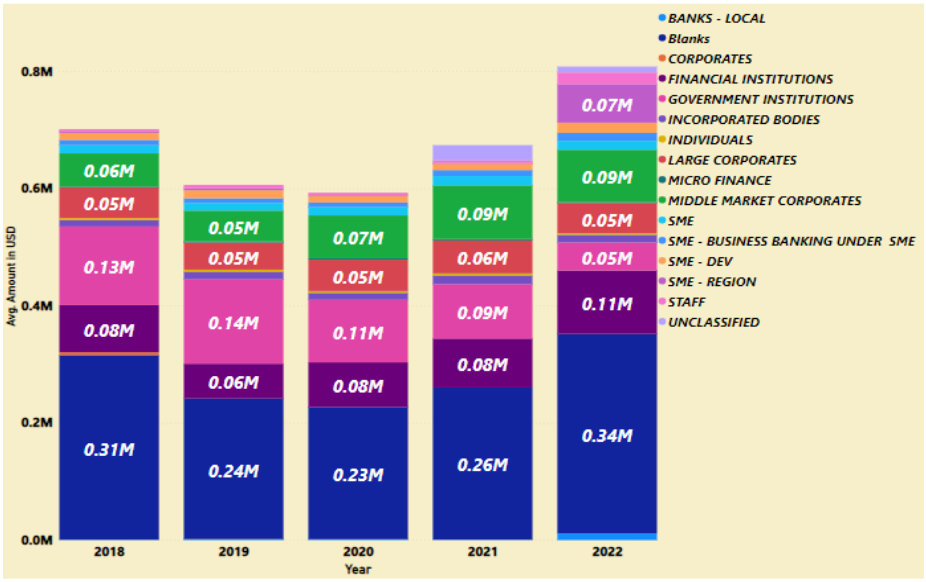

Figure 3-16:Inflow Avg.amount by basles

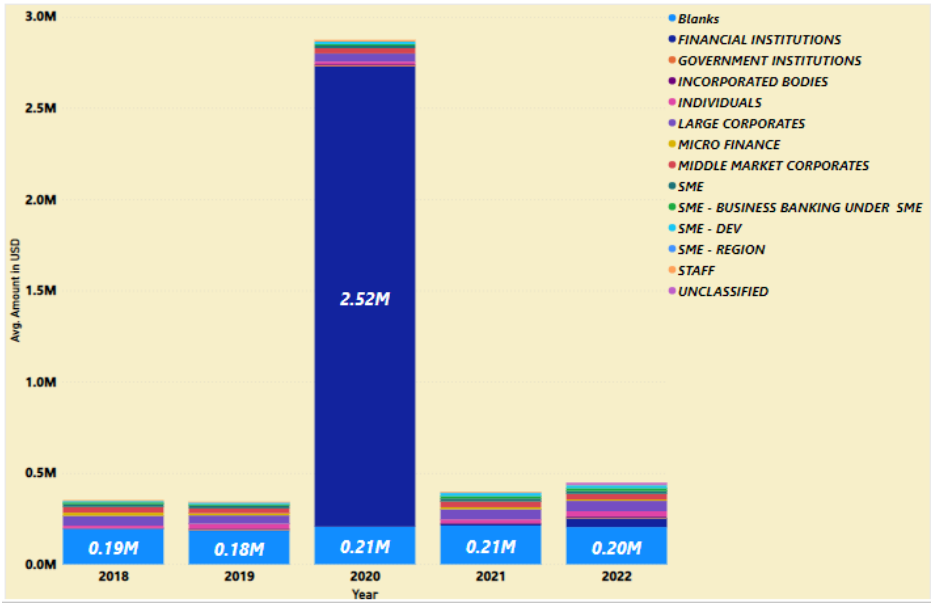

Figure 3-17: Outflow Avg.amount by basles

By comparing above two graphs it can be clearly seen that in 2020 there is a significant difference in inflow and outflow of FINANCIAL INSTITUTIONS. The reason for that may be as follows.

The COVID-19 pandemic, which had a significant negative effect on the international economy, caused the world to endure an unparalleled global health

catastrophe in 2020. Lockdowns and other preventative measures were implemented by governments all around the world in response to the virus, which caused a significant drop-in economic activity. Reduced cash flows resulted from businesses, including corporate entities, being obliged to curtail or cease activities.

3.3.3 Pair plots for the variables of time series dataset.

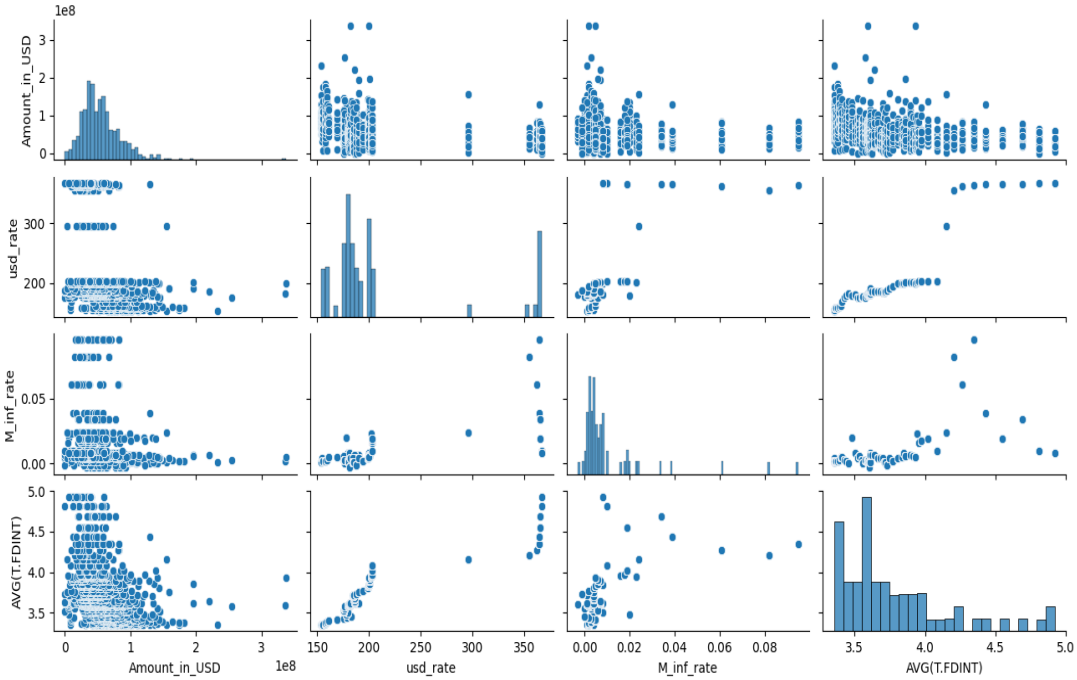

Figure 3-18: Scatter plots and histograms for time series variables

According to the above pair plots, the distribution of all 4 variables are positively skewed distributions. From the scatter plot of amount in USD vs usd rate, there can be seen 3 clusters, so that the first cluster is between the usd rates of 150 and 200 and other two clusters are in around the 300 and 360 respectively. Also, there is some negative trend which means when usd rate goes up, then amount of usd inflow goes down.

By observing the scatter plot of amount in USD inflow vs monthly inflation rate (M_inf_rate), there are two major clusters and 4 more small clusters. Two major clusters are around the values of 0.005 and 0.02.

The scatter plot of amount in USD inflow and average fixed deposit interest rate shows the negative trend which means when the interest rate increases then the amount of inflow decreases gradually.

3.3.4 Correlation checking for time series variables.

Considering the all 4 continuous variables, Amount_in_USD , usd_rate, M_inf_rate, AVG (T.FDINT). The heatmap was plotted in order to get an idea about the correlations between pair wise.

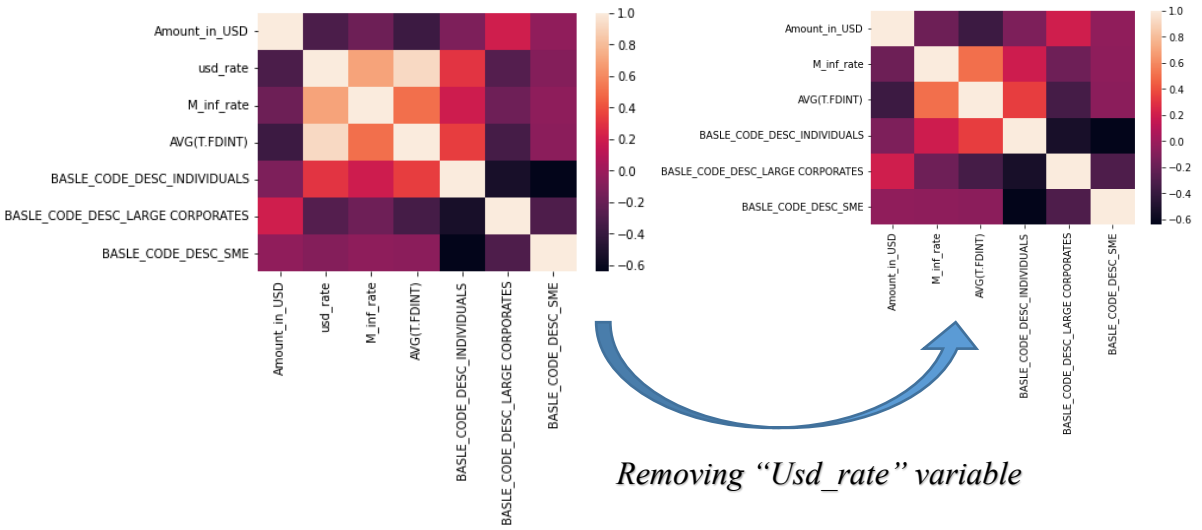

Figure 3-19: Correlation heatmaps for time series variables

Here, the left side heatmap shows the initial variables' correlation. Since there was some correlation between the usd_rate and inflation rate and average fixed deposit interest rate, the usd_rate variable was needed to be removed. Then the right side heatmap shows the rest variables' correlation, there was no significant correlation between the rest variables.

3.3.5 Checking multicollinearity among the time series variables.

```
...: print(vif)
                 variables         VIF
0                    const    0.000000
1           Amount_in_USD     1.199189
2                 usd_rate    13.331127
3              M_inf_rate      2.586986
4              AVG(T.FDINT)    10.080127
```

```
...: print(vif)
                 variables         VIF
0                    const    0.000000
1           Amount_in_USD     1.185153
2              M_inf_rate      1.339233
3              AVG(T.FDINT)    1.668901
```

Figure 3-20: VIF values for time series variables

In order to check the overall multicollinearity of all 4 continuous variables. The Variance Inflation Factor (VIF) was considered. There it had some VIF values which

were greater than 5. By removing the maximum VIF value of “usd_rate”, all other multicollinearity was eliminated.

Therefore, when conducting forecasting from this dataset, it is recommended to use the other variables excluding the “usd_rate”.

4. Advanced Analysis

4.1 Considering the Inflow Amount

4.1.1 Checking stationarity

4.1.1.1 Applying Augmented Dickey-Fuller (ADF) test

Null hypothesis = The considered series is non- stationary.

Alternative hypothesis = The considered series is stationary.

Table 5: ADF test results

ADF Test Statistic	p-value	1% Critical Value	5% Critical Value	10% Critical Value
-5.976034	1.88657e-07	-3.43396	-2.86314	-2.56762

ADF test statistic was discovered to be -5.976034, with a p-value of 1.88657e-07. The critical values were, respectively, -3.43396, -2.86314, and -2.56762 for the 1%, 5%, and 10% levels. These findings indicate that the data was stationary, which supports the alternative hypothesis.

4.1.1.2 Applying Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test

Null hypothesis = The considered series is stationary.

Alternative hypothesis = The considered series is non-stationary.

Table 6: KPSS test results

KPSS Test Statistic	p-value	10% Critical Value	5% Critical Value	2.5% Critical Value	1% Critical Value
4.909238544891569	0.01	0.347	0.463	0.574	0.739

Then the dataset was subjected to the KPSS test, and the results revealed a test statistic of 4.91, indicating that the stationarity null hypothesis could not be accepted. The dataset is likely non-stationary because the p-value was 0.01. The critical values at the

10%, 5%, 2.5%, and 1% levels, respectively, were 0.347, 0.463, 0.574, and 0.739. According to this, the series shows a pattern and is not stationary.

Taking into account the aforementioned two test findings, the results of two tests reveal inconsistent results. Consequently, additional observation is required. However, a very tiny downward tendency may be seen in figure 7. The KPSS test findings can therefore be accepted. The following decomposition graphs provide as further evidence for such.

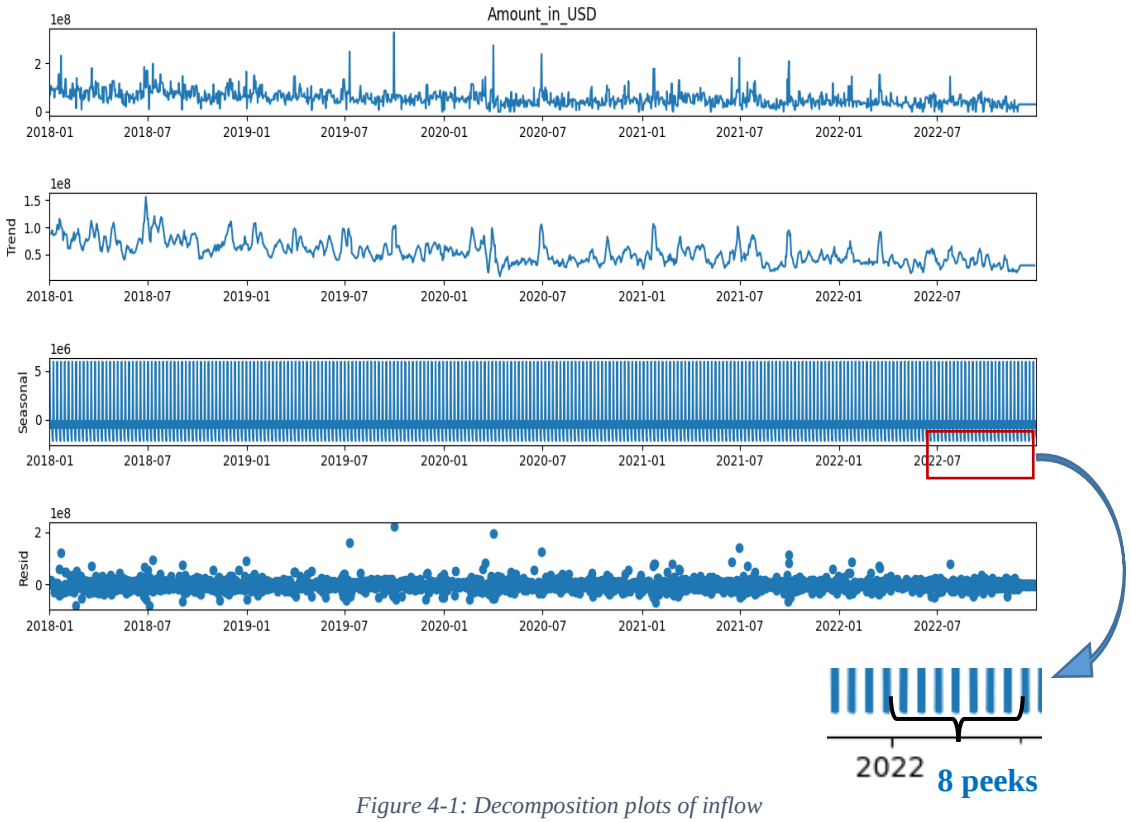

Figure 4-1: Decomposition plots of inflow

Here it can be clearly identified that there are both trend and seasonal components. By observing the figure 27, the seasonal period can be considered as **3/8** in months. Approximately it can be considered as **11 days.**

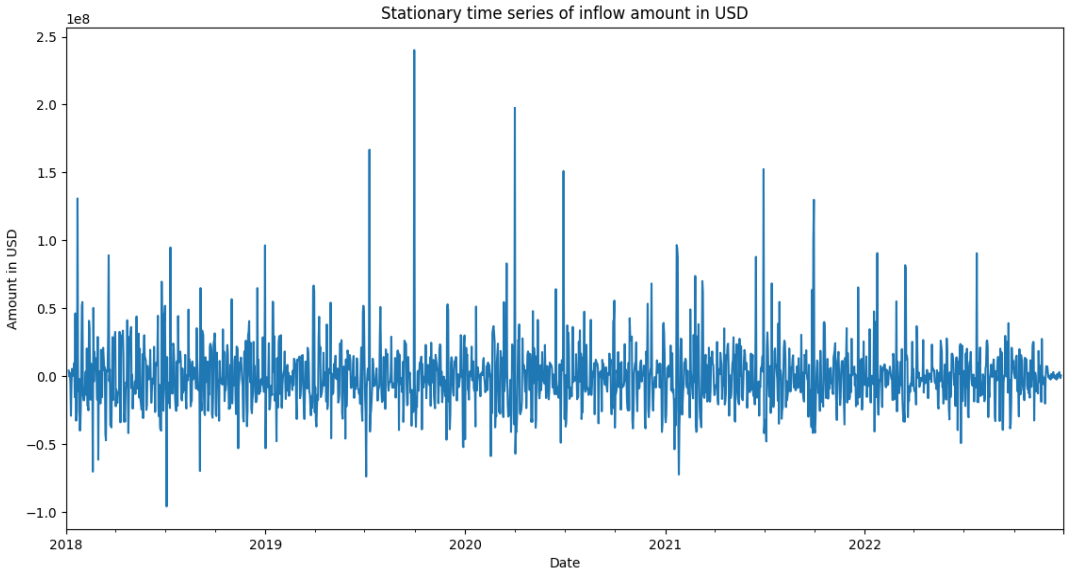

Figure 4-2: Stationary series of inflow

4.1.1.3 Making the time series stationary

The inflow amount series needs to be detrended and deseasonalized in order to become stationary. Following that action, the ADF and KPSS test results were as follows, and the series plot was as follows.

Table 7: ADF test results (after making stationary)

ADF Test Statistic	p-value	1% Critical Value	5% Critical Value	10% Critical Value
-2.4034	2.1452e-07	-2.4557	-1.2545	-3.2556

Table 8: KPSS test results (after making stationary)

KPSS Test Statistic	p-value	10% Critical Value	5% Critical Value	1% Critical Value
0.029	0.12	0.347	0.463	0.739

According to the above two tests, table 07 shows the p value which was smaller than the 0.05. thus, null hypothesis was rejected and suggests that the series is stationary. Also, table 8 shows the KPSS test results so that the series was stationary as well. Therefore, it can be said that the series has been converted to a stationary series.

4.1.2 Autoregressive Integrated Moving Average forecasting. (ARIMA)

Here, only the "Amount in USD" column should be taken into consideration in this instance, and it was assumed to be depended solely on the of time. (and not on any other exogenous variables). Since the original series was not stationary, it was used converted series which was stationary after once getting the difference of the initial.

Prior to the modelling process the dataset was split into two sets as 80% of the dataset as training set and 20% as the test set.

4.1.2.1 Autocorrelation (ACF) and Partial Autocorrelation (PACF) plots

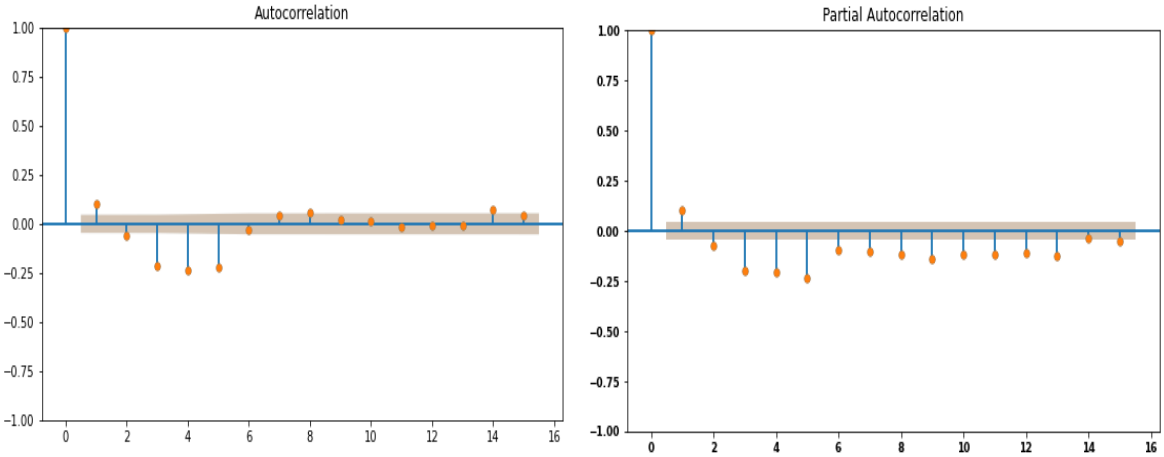

Figure 4-3: ACF and PACF plots

According to the ACF and PACF plots, it can be observed that the one significant lag value in both plots. Therefore, both AR and MA terms are 1. (AR (1) & MA (1))

Prior to those, initial series was differenced in order to make the series stationary. Therefore, the terms of p, d and q can be taken as (1,1,1). Therefore, it can be used that simple model as a startup for the model forecasting.

Following figure 4-6 shows the visualization of the training series, test series and predicted series.

```
In [53]: print(model_fit.summary())
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1452
Model:                 ARIMA(1, 1, 1)   Log Likelihood              -26820.988
Date:                Thu, 20 Apr 2023   AIC                          53647.976
Time:                        12:05:27   BIC                          53663.816
Sample:                    01-06-2018   HQIC                         53653.887
                         - 12-27-2021
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.1104      0.025      4.394      0.000       0.061       0.160
ma.L1         -0.9991      0.021    -46.799      0.000      -1.041      -0.957
sigma2      8.817e+14   2.23e-17   3.95e+31      0.000    8.82e+14    8.82e+14
===================================================================================
Ljung-Box (L1) (Q):                   0.09   Jarque-Bera (JB):             11384.60
Prob(Q):                              0.77   Prob(JB):                         0.00
Heteroskedasticity (H):               0.95   Skew:                             2.00
Prob(H) (two-sided):                  0.56   Kurtosis:                        16.12
===================================================================================
```

Figure 4-5: ARIMA (1,1,1) summary results

According to the summary results of figure 4-5, The coefficients of the AR (1) and MA (1) terms are 0.1104 and -0.9991, respectively, and they are used to fit the time series data in an ARIMA model with an autoregressive term of order 1 (AR (1)) and a moving average term of order 1 (MA (1)). In contrast, the error term from the previous time step has a large negative impact on the current value, but the past value of the time series positively impacts the current value and this influence decreases as the time lag rises. As the p-values are almost 0, the coefficients are statistically significant, indicating that the ARIMA model fits the data well and can make precise predictions about the time series' future values.

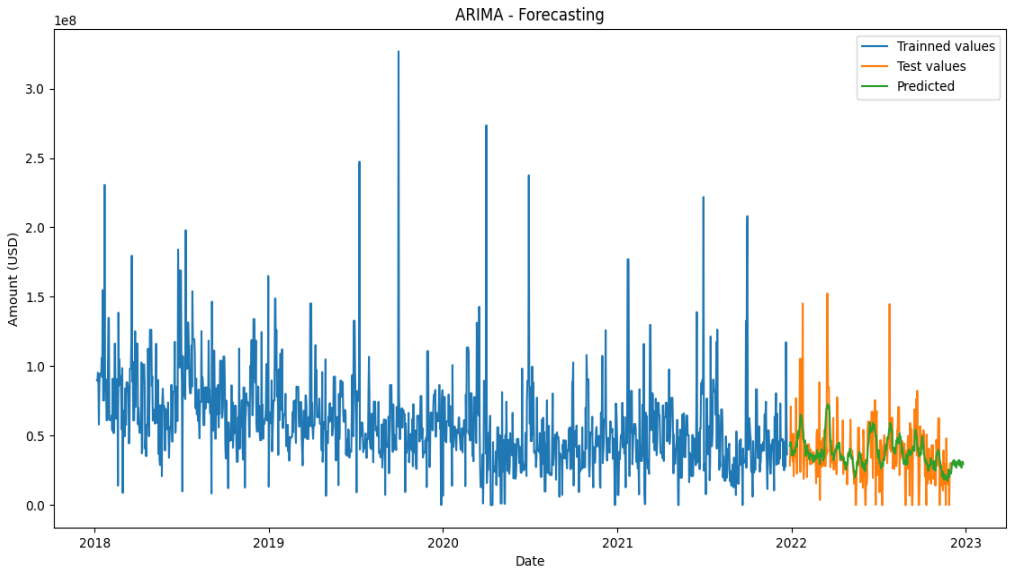

Figure 4-6: ARIMA (1,1,1) forecasting

4.1.2.2 Model validation and residual diagnosis

```
   Fold          RMSE
0   1.0  2.349736e+07
1   2.0  2.881596e+07
2   3.0  2.299111e+07
3   4.0  2.360338e+07
4   5.0  1.711438e+07
Mean RMSE: 23204437.373602398
Standard deviation of RMSE: 3711295.0350186666
```

Figure 4-7: 5 - Fold Cross validation RMSE values

The fold number and related RMSE value for each fold are displayed in the above figure. The average of the RMSE values across all folds is represented by the "Mean RMSE" value of 23204437.37, and the variation in the RMSE values across folds is shown by the "Standard Deviation of RMSE" value of 3711295.04.

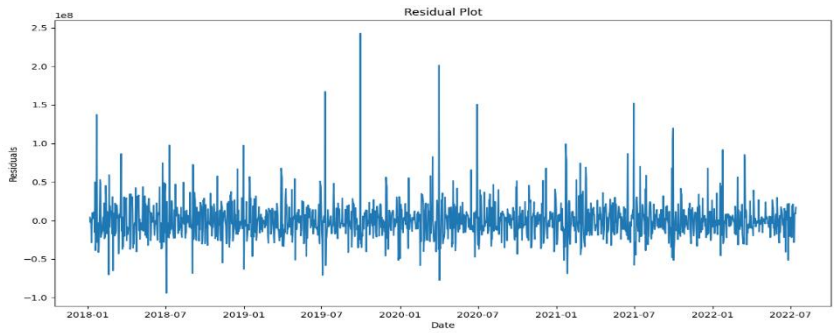

Figure 4-8: Residual plot for ARIMA (1,1,1)

The residuals in the associated representation spread at random around the zero axis. Even though there were some variations in the plot, it is still reasonable to accept that the residuals' variance should remain constant throughout time.

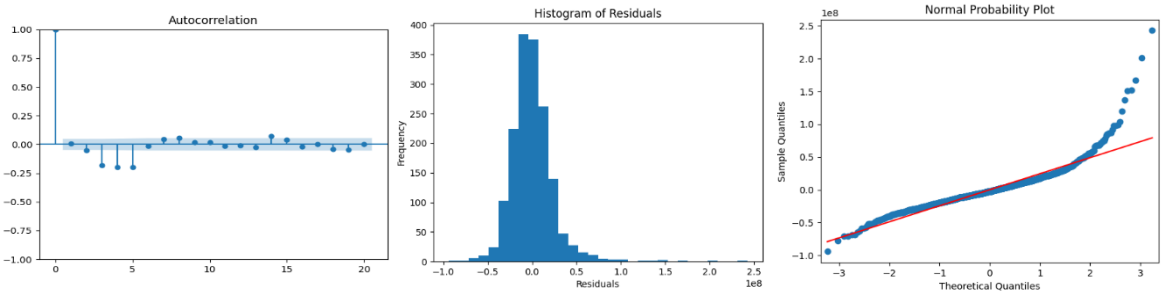

Figure 4-9: Residual Diagnosis plots for ARIMA (1,1,1)

According to the above plots, even the normality assumption of residuals were satisfied, it seems to have some correlations in 3rd ,4th and 5th lags from the ACF plot. However overall performance of residuals assumptions can be accepted.

4.1.3 Auto- Arima with exogenous variables

In order to get more tunned Arima model, it was used Auto – ARIMA modeling technique which comes under python programming. Here the other exogenous variables also used since auto – ARIMA has that facility. Therefore, all the exogenous variables (03) were scaled since those were in different scales. After performing auto – arima, following results were obtained. The best model from that was ARIMA (5,1,5). Even the output results suggested ARIMA (5,0,5), that was for the differenced series which was used to performed.

```
In [51]: print(arima_model.summary())
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1453
Model:               SARIMAX(5, 0, 5)   Log Likelihood              -26571.791
Date:                Thu, 20 Apr 2023   AIC                         53167.583
Time:                        11:59:22   BIC                         53230.959
Sample:                  01-06-2018   HQIC                        53191.230
                       - 12-28-2021
Covariance Type:                opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept   4.129e+04   4.53e+04      0.912      0.362   -4.74e+04     1.3e+05
ar.L1         -0.8391      0.184     -4.569      0.000      -1.199      -0.479
ar.L2          0.3784      0.068      5.591      0.000       0.246       0.511
ar.L3          0.7186      0.083      8.639      0.000       0.556       0.882
ar.L4         -0.2512      0.086     -2.914      0.004      -0.420      -0.082
ar.L5         -0.6169      0.115     -5.382      0.000      -0.842      -0.392
ma.L1          0.6240      0.185      3.367      0.001       0.261       0.987
ma.L2         -0.9018      0.088    -10.251      0.000      -1.074      -0.729
ma.L3         -1.2880      0.133     -9.706      0.000      -1.548      -1.028
ma.L4         -0.0012      0.128     -0.009      0.993      -0.252       0.250
ma.L5          0.5947      0.133      4.461      0.000       0.333       0.856
sigma2      4.754e+14   4.95e-05   9.61e+18      0.000    4.75e+14    4.75e+14
===================================================================================
Ljung-Box (L1) (Q):                   0.16   Jarque-Bera (JB):              6623.89
Prob(Q):                              0.69   Prob(JB):                         0.00
Heteroskedasticity (H):               0.94   Skew:                             1.26
Prob(H) (two-sided):                  0.49   Kurtosis:                        13.15
===================================================================================
```

Figure 4-10: ARIMA (5,0,5) summary results

Results from the ARIMAX model in figure 4-10, which have a log-likelihood of -26571.791 and an AIC of 53167.583, indicate that the model fits the data well. The intercept term's p-value of 0.362 indicates that it is not statistically significant. The time series' current value is influenced by its previous values up to five lags in the past, as demonstrated by the model's inclusion of an autoregressive term of order five (AR(5)). With the exception of the second lag, which has a positive influence, the coefficients of the AR components demonstrate that the time series is negatively influenced by its prior values. The "Ljung-Box (L1)" statistic tests whether there is any autocorrelation in the residuals at lag 1. In this case, the p-value of 0.77 indicates that there is no evidence of autocorrelation at lag 1.

Even the output findings pointed to an ARIMA of (5, 0, 5), that was executed to the differenced series. The optimal model should be ARIMA (5,1,5) when the beginning series is taken into consideration.

Root Mean Squared Error (RMSE) of ARIMA (1,1,1) → 17373459.02033349

Root Mean Squared Error (RMSE) of Auto- ARIMA with exogenous variables (5,1,5) → 17800035.43778879

4.1.3.1 Model validation and Residuals Diagnosis

	Fold	RMSE
0	1	2.358283e+07
1	2	2.877741e+07
2	3	2.299942e+07
3	4	2.357408e+07
4	5	1.711221e+07

Mean RMSE: 23209189.682165675
Standard deviation of RMSE: 3701217.368541227

Figure 4-11: 5 - Fold Cross validation RMSE values

The auto_arima model with exogenous variables' K-fold cross-validation results are displayed in the output. The mean RMSE value for all folds is 23,209,189.68, which provides insight into the model's overall accuracy. The RMSE's standard deviation, which measures how well the model performs across various folds, is 3,701,217.37. In conclusion, the model appears to function pretty well, however its accuracy exhibits considerable unpredictability.

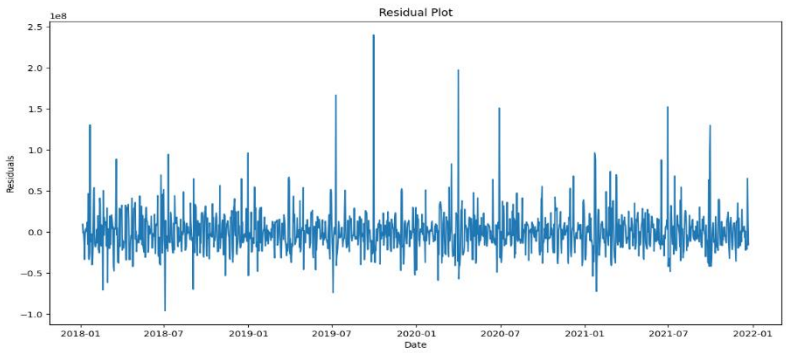

Figure 4-12: Residual plot for ARIMAX (5,1,5)

Same as for ARIMA (1,1,1), the residual plot shows randomly scattered around the zero line. Even there some random fluctuations constant variance can be seen. Therefore, the assumption is satisfied.

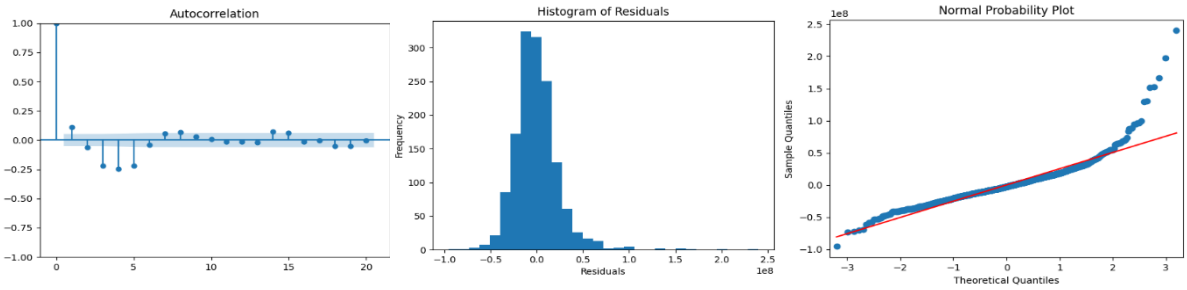

Figure 4-13: Residual Diagnosis plots for ARIMAX (5,1,5)

According to the above plots, even the normality assumption of residuals were satisfied, it seems to have some correlations in 1st ,3rd ,4th and 5th lags from the ACF plot. However overall performance of residuals assumptions can be accepted.

4.1.4 Gated Recurrent Unit model

To perform GRU model, the stationary series was used and at the end trend and seasonal components were added accordingly.

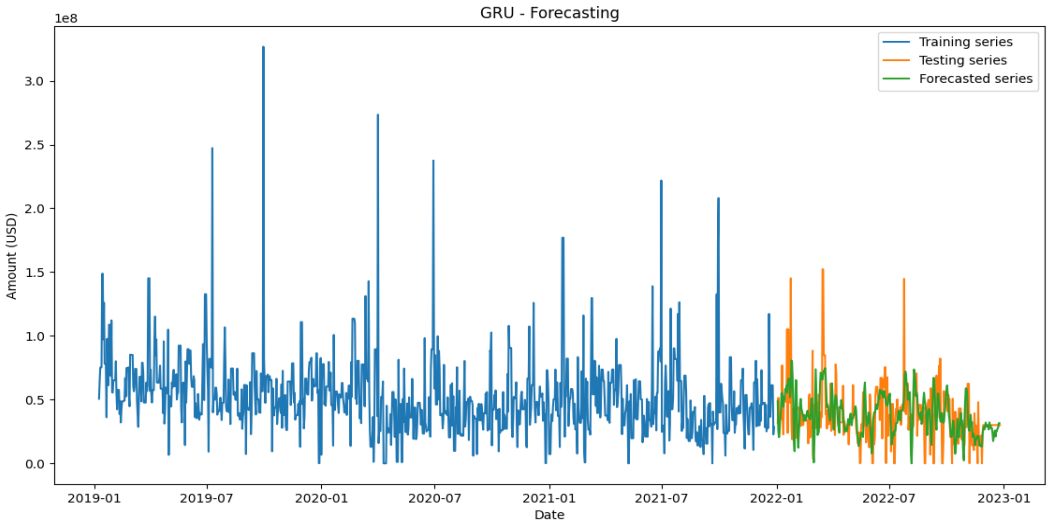

Figure 4-14: GRU forecasting

```
trainX shape == (1413, 11, 7)
trainY shape == (1413, 1)
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru_4 (GRU)                 (None, 11, 50)            8850

 gru_5 (GRU)                 (None, 11, 50)            15300

 gru_6 (GRU)                 (None, 11, 40)            11040

 gru_7 (GRU)                 (None, 50)                13800

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 49,041
Trainable params: 49,041
Non-trainable params: 0
```

Figure 4-15: GRU model summary

Root Mean Squared Error (RMSE) of GRU with exogenous → 24523312.80586105

According to the figure 4-16, training loss curve is gradually decreasing while, the validation curve slightly decreases up to 25 epochs and then it has increased. This implies that the model has been started to overfitting from 25 epochs.

Figure 4-16: GRU loss curve

The GRU model has 3 hidden layers excluding the input and output layers. The hidden 03 layers have 50, 50 and 40 units while the input layer is consisted of 50 units.

4.1.5 Long – Short Term Memory model

In order to perform the LSTM model, stationary series was used and at the end trend and seasonal components were added accordingly.

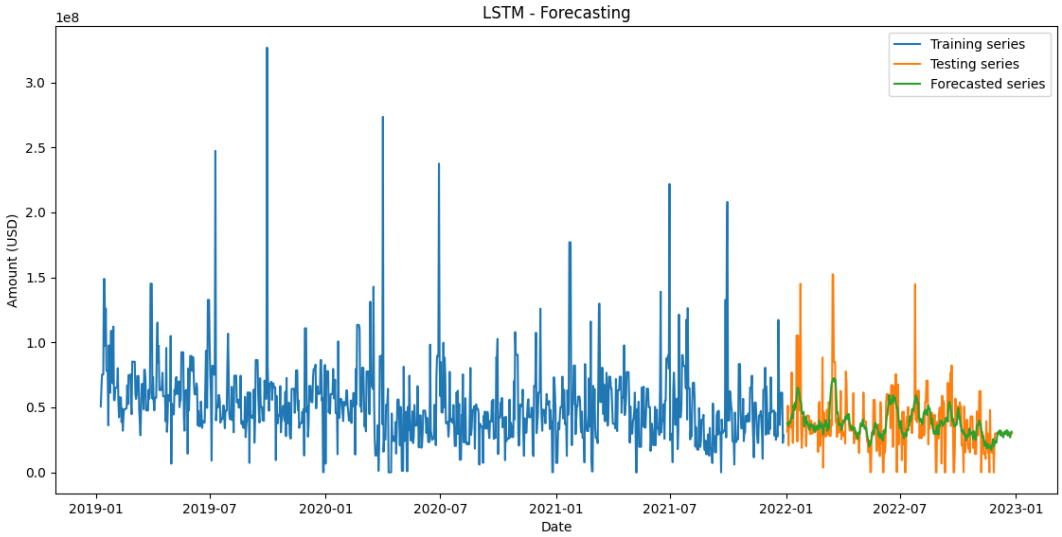

Figure 4-17: LSTM forecasting plot with exogenous variables

```
trainX shape == (1413, 11, 7)
trainY shape == (1413, 1)
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 11, 64)            18432

 dropout (Dropout)           (None, 11, 64)            0

 lstm_1 (LSTM)               (None, 32)                12416

 dropout_1 (Dropout)         (None, 32)                0

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 30,881
Trainable params: 30,881
Non-trainable params: 0
```

Figure 4-18: LSTM model summary

Root Mean Squared Error (RMSE) of LSTM with exogenous → 22075302.81752637

According to the figure 4-19, It seems to be the model performance was moderate as its both training and validation loss are gradually decreasing with the number of epochs up to 100.

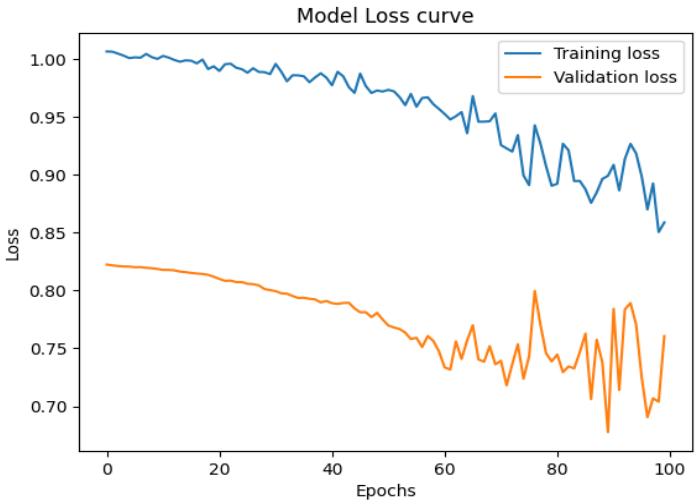

Figure 4-19: Train and validation loss curve

4.1.6 Vector Autoregression Model

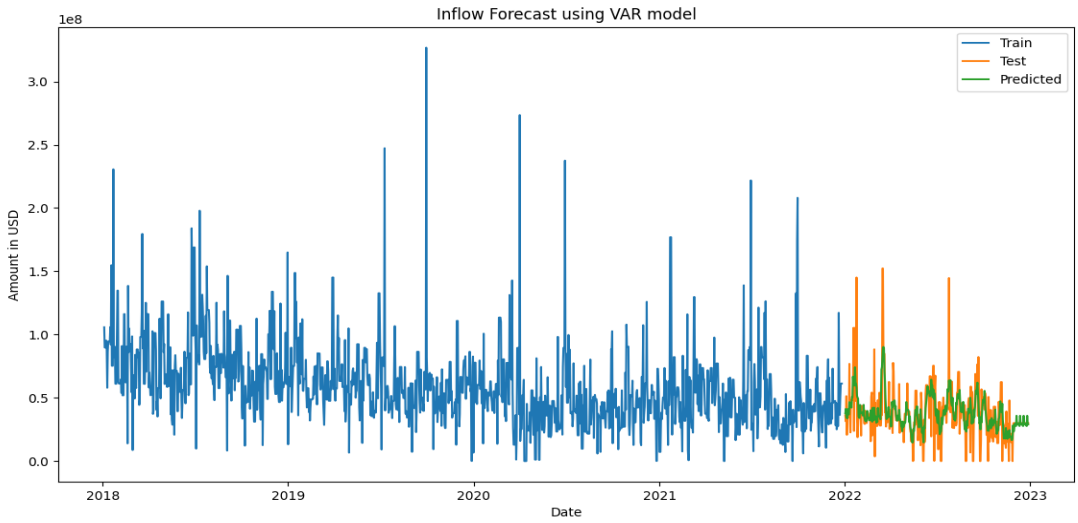

Figure 4-20: VAR model forecasting

Note: Vector Autoregressive modelling technique assumes that the data is continuous and normally distributed. Therefore, continuous variables were used to perform VAR model except all categorical variables

```
  Summary of Regression Results
================================
Model:                         VAR
Method:                        OLS
Date:           Fri, 28, Apr, 2023
Time:                     11:40:53
--------------------------------------------------------------------
No. of Equations:         3.00000    BIC:                    8.10477
Nobs:                     1448.00    HQIC:                   7.93342
Log likelihood:          -11758.8    FPE:                    2518.47
AIC:                      7.83140    Det(Omega_mle):         2392.40
--------------------------------------------------------------------
Results for equation Amount_in_USD
================================================================================================
                         coefficient       std. error           t-stat            prob
------------------------------------------------------------------------------------------------
const               -1187443.172360   11257428.826769           -0.105           0.916
L1.Amount_in_USD          -0.319061          0.026258          -12.151           0.000
L1.M_inf_rate       -1212070.481362  561596307.928092           -0.002           0.998
L1.AVG(T.FDINT)     336537461.769617  193484743.747252            1.739           0.082
L2.Amount_in_USD          -0.469313          0.027747          -16.914           0.000
L2.M_inf_rate       526089323.085088  790905885.398244            0.665           0.506
L2.AVG(T.FDINT)    -255843978.447472  269566742.338748           -0.949           0.343
L3.Amount_in_USD          -0.621018          0.029390          -21.130           0.000
L3.M_inf_rate      -223230719.389347  790899957.966306           -0.282           0.778
L3.AVG(T.FDINT)    -249066381.897157  269646496.930813           -0.924           0.356
L4.Amount_in_USD          -0.400954          0.032183          -12.459           0.000
L4.M_inf_rate      -106989962.233218  790757702.486294           -0.135           0.892
L4.AVG(T.FDINT)     -32096037.730017  269777216.833047           -0.119           0.905
L5.Amount_in_USD          -0.373155          0.032283          -11.559           0.000
L5.M_inf_rate      -284267974.014490  790395784.553458           -0.360           0.719
L5.AVG(T.FDINT)     148205896.370306  269763960.949687            0.549           0.583
L6.Amount_in_USD          -0.324584          0.029362          -11.054           0.000
L6.M_inf_rate       461471397.497668  787411773.037055            0.586           0.558
L6.AVG(T.FDINT)    -220676226.757045  269708767.170479           -0.818           0.413
L7.Amount_in_USD          -0.196157          0.027980           -7.011           0.000
L7.M_inf_rate     -1002838668.416110  785527257.944793           -1.277           0.202
L7.AVG(T.FDINT)    -194562877.795531  269757303.521417           -0.721           0.471
L8.Amount_in_USD          -0.125186          0.026698           -4.689           0.000
L8.M_inf_rate       621067694.637203  559768761.999247            1.110           0.267
L8.AVG(T.FDINT)     467929234.292899  193130345.746335            2.423           0.015
================================================================================================
```

Figure 4-21: Summary of the best model - VAR (8)

Root mean squared error of VAR (8)– 16330833.107

Figure 4-21 shows the results equation summary for the variable "Amount in USD". The lagged values of the Amount_in_USD variable are negatively linked with the present value of the variable, according to the regression results for the Amount_in_USD variable. The lagged values' coefficients are all statistically significant, proving that the past values of the variable have a big influence on the present value.

A minor negative coefficient exists for the initial lag for the M_inf_rate variable, but it is not statistically significant. The same is true for the AVG(T.FDINT) variable, which has a positive first lag coefficient but is not statistically significant.

```
Correlation matrix of residuals
                Amount_in_USD  M_inf_rate  AVG(T.FDINT)
Amount_in_USD        1.000000   -0.006858     -0.061559
M_inf_rate          -0.006858    1.000000      0.061533
AVG(T.FDINT)        -0.061559    0.061533      1.000000
```

Figure 4-22: Correlation matrix of residuals

A weak negative connection between the Amount_in_USD variable and the M_inf_rate variable and a weak negative correlation between the Amount_in_USD variable and the AVG (T. FDINT) variable can be seen in the residuals correlation matrix. The M_inf_rate variable and the AVG (T. FDINT) variable have a weakly positive association.

4.1.6.1 Model validation and Residual Diagnosis

```
RMSE values for 5-fold CV:
   Fold          RMSE
0   1.0  2.144086e+07
1   2.0  2.711942e+07
2   3.0  2.013968e+07
3   4.0  2.210281e+07
4   5.0  1.552813e+07

Mean RMSE: 21266179.234812807
Standard deviation of RMSE: 3719814.7325027916
```

Figure 4-23:5 - Fold Cross validation RMSE values

The 5-fold cross-validation's mean RMSE value is 21,266,179.23. This indicates the model's overall average performance across all folds. The model's effectiveness varies somewhat among the many folds, as shown by the RMSE values' standard deviation of 3,719,814.73.

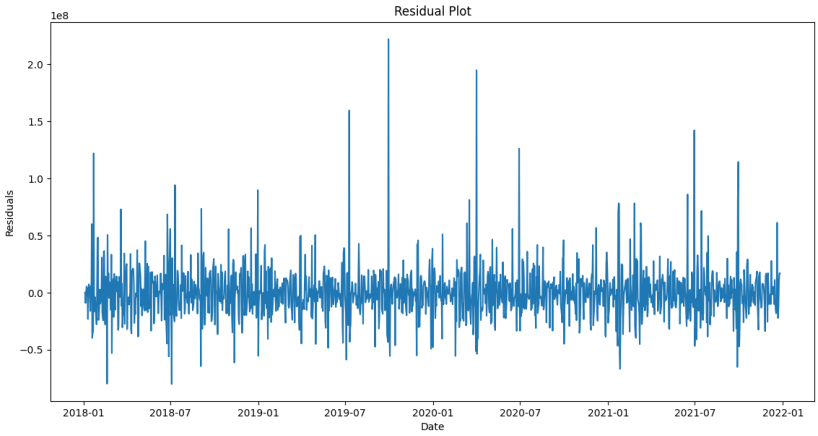

Figure 4-24: Residual plot for VAR (8)

Same as for previous plots, the residual plot shows randomly scattered around the zero line. Even there some random fluctuations constant variance can be seen. Therefore, the assumption is satisfied.

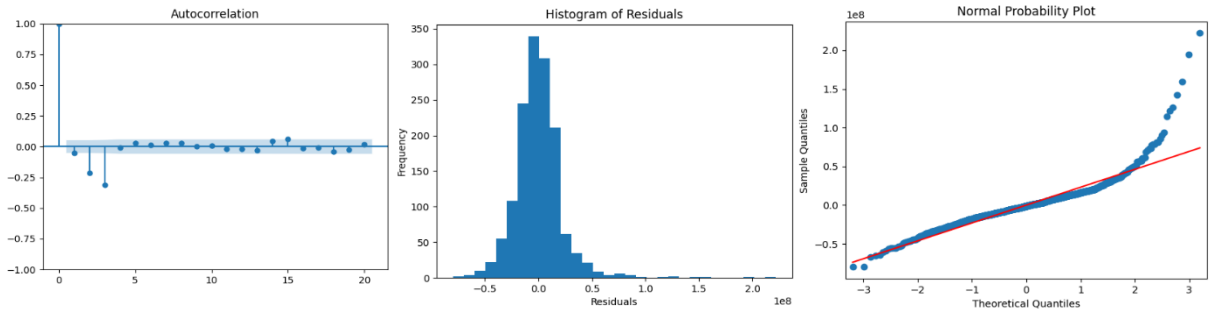

Figure 4-25 :Residual Diagnosis plots for VAR (8)

According to the above figure, ACF plot shows that there are some correlation in the lags of 2nd and 3rd . However, when observing the other two plots, there are enough evidence to accept the assumption that residuals were normally distributed.

Note: Residual diagnosis or any other assumptions checking are not conducted to the Deep learning methods. Here GRU and LSTM are deep learning forecasting method. Therefore, it was not necessary to conduct any assumption checking under that.

4.2 Summary of results.

Table 9: Inflow amount in USD Summary of model results

Modelling technique	Train RMSE	Test RMSE	Not a good fit/ Good fit
ARIMA (1,1,1)	23204437.37360240	17373459.02033349	Not a good fit
ARIMAX (5,1,5) – auto-Arima	23209189.68216568	17800035.43778879	Not a good fit
SARIMAX (1, 1, 1, 11)	45575854.34344987	366042664.92250276	Not a good fit
GRU	23478930.847437832	24523312.80586105	Good fit
LSTM	**21029384.89458393**	**22075302.81752637**	**Good fit**
VARX	21266179.23481281	16330833.107	Not a good fit

Table 10: Outflow amount in USD Summary of model results

Modelling technique	Train RMSE	Test RMSE	Not a good fit/ Good fit
ARIMA (1,1,1)	22243423.48893388	17244833.07484628	Not a good fit
ARIMAX (5,1,5) – auto-Arima	22283456.43435334	17936272.43545422	Not a good fit
SARIMAX (1, 1, 1, 11)	43234322.35322342	45587383.63885845	Not a good fit
GRU	23453253.43545432	24523312.80586105	Good fit
LSTM	**21132342.73859393**	**21733834.81752637**	**Good fit**
VARX	23848232.45454352	17043532.423432	Not a good fit

Note:

In order to perform the outflow amount forecasting models, just the same steps were followed as for the inflow. Nothing was change, only the dataset was changed, and corresponding parameters were tunned accordingly. The summary of the models taken are represented in the table 10.

The plot in figure 3-3 shows that the amounts flowing in and out fluctuate in the same manner. As a result, the model that was utilized for predicting the amount of inflow can also be used to forecast the amount of outflow. However, if that is the case, it is crucial to take into account the times when inflow and outflow behave similarly. The ability to utilize the same model for outflow is made possible by the fact that the other datasets used in this case—monthly inflation, interest rates, and exchange rates—are the same datasets as were used to develop the inflow forecasting model. Furthermore, since inflow and outflow amounts behave similarly, it is reasonable to apply the same forecasting model to both.

But here in this process, it was conducted separate new models for the outflow dataset in order to achieve more confident accuracy. Reasonably, the RMSE values for inflow and outflow were seemed to be same.

5 Discussion and Conclusion

5.1 Discussion

According to the histogram in figure 3-1, which shows a skewed distribution to the right, a significant portion of transactions were in the range of 5*106 USD. There may have been more transactions with smaller values because the mean value was closer to the minimum value. Similar to the inflow, the distribution of the outflow amount in USD is favorably skewed, which shows that the transactions underlying both the inflow and the outflow were of a similar nature.

According to the trend in Figure 3-3, the bank has consistently engaged in foreign currency trading over time. The similarities in historical fluctuations across the two lineages imply consistency and predictability in the bank's foreign exchange management practices. For instance, across the chosen time period, the bank may have maintained consistent amounts of foreign currency deposits and withdrawals from customers, which suggests a steady demand for foreign currency. The bank's foreign currency business does not appear to follow any distinct seasonal or cyclical patterns, as shown in Figures 3-4 and 3-5. This shows that the influx and outflow of foreign currency at the bank is not significantly impacted by external factors like holidays or economic cycles. For instance, the bank's foreign currency deposits and withdrawals might be quite consistent throughout the year, with no notable peaks or valleys during the holidays. According to Figure 3-6's outflow chart, large corporations represent the second-largest segment of the population. In order to draw and keep these customers for foreign currency transactions, the bank may choose to focus its efforts in this area. For instance, the bank might provide more affordable exchange rates, adaptable transaction periods, and customized solutions to fulfill the unique requirements of major organizations. From the figures 3-8 and 3-9, Retail client transactions may be included in the individual and blank categories of the inflow chart, highlighting the value of taking into account unique customer behavior. Large corporations and SMEs have similar inflow and outflow patterns, which shows that these companies have steady foreign currency demands. This information can be used to estimate future influx and outflow periods and to guide forecasting models and decision-making.

The figures 3-16 and 3-17 reveal a large disparity in the influx and outflow of financial institutions in 2020, which may be driven by the challenges the banking industry faced during the COVID-19 pandemic-induced economic crisis. This might have caused banks to tighten their lending criteria, diminish the demand for credit, and increase credit risk, which would have decreased the amount of corporate inflows. To fund their operations during the crisis, companies may have borrowed money at cheap interest rates; as a result, corporate outflow amounts may have increased significantly.

Compared to other models, the LSTM model did better in predicting the volume of the inflow and outflow. The bank is experiencing the chance to use the model's accuracy to decide on foreign currency inflows and outflows more effectively. According to the tables 09 and 10, The deep neural network models have proven successful in forecasting future outflow amounts in USD. This provides the banking industry with valuable information for risk management, cash flow management, and liquidity planning.

5.2 Conclusions

- The most frequent transaction amount is approximately 5*106 USD, the distribution is skewed to the right and big transactions have greater values, and there are more transactions with lower values. These are the main takeaways from the inflow amount distribution. These revelations could be useful while making financial decisions.
- Foreign currency deposits and withdrawals from the bank were impacted by the same factors. To better comprehend the organization’s financial performance, a thorough review of these factors is required.
- According to an examination of foreign exchange activity throughout time, the bank appears to have been successful in upholding predictability and consistency in its foreign exchange management practices. This is crucial for maintaining financial stability and making knowledgeable judgments about the risk of currency exchange. To reduce swings in foreign currency activity, the bank, for example, may have established clear standards and policies for managing foreign exchange transactions. Customers’ trust and confidence in the bank’s foreign exchange services may increase as a result of its consistency and predictability.

- Since there is no identifiable pattern in the data, it is likely that the fluctuations in foreign currency activity are random and may be caused by changes in consumer behavior or market conditions. The bank may research these findings, create plans and policies to control the irregular fluctuations, and reduce the negative tendency in order to enhance its procedures for managing foreign exchange. For instance, the bank might implement risk management plans to lessen the effect of market volatility on transactions involving foreign currencies or roll out new goods and services to draw in clients who wish to deal in foreign currencies.
- The fact that people make up the majority of transactions in the inflow chart is crucial because it implies that the foreign currency management team at the bank may need to change its rules and restrictions in order to better serve this customer base. It might also mean that the bank needs to concentrate its efforts on getting more SMEs and big businesses to use the bank for foreign exchange transactions. (Figure 3-7).
- The bank may need to learn more about the interests and preferences of large firms in order to develop regulations and limits for foreign exchange transactions with them that are more effective. Distinct kinds of major organizations could have distinct needs and goals, as evidenced by the rise in middle market corporate transactions in the outflow figure. For instance, the bank may interview and study large corporate clients to learn their opinions in order to build customized foreign exchange services and solutions that satisfy their demands.
- The COVID-19 pandemic, which significantly hurt the world economy in 2020, dominated the year. Lockdowns, broken supply chains, and decreased consumer spending were all brought on by the epidemic, which also caused unemployment rates to rise and the economy to grow more slowly. The financial industry has been harmed by this, and many banks and financial institutions are now having trouble running their businesses and providing client service. The pandemic also affected the inflow and outflow of financial institutions, as seen in figures 3-16 and 3-17, underscoring the necessity for efficient crisis management strategies and policies to lessen the effects of future crises.

- The bank can improve its financial performance and retain clients by using precise forecasting models like the LSTM. The model should be monitored and adjusted on a regular basis because it is based on historical data and may not take into account unforeseen circumstances or changes in the economic environment.
- And also, accurate predictions of future outflow amounts can help banks make strategic investments and ensure they have enough cash on hand to meet financial responsibilities. Additionally, accurate predictions can improve customer satisfaction and retention by offering appropriate services and products.

5.3 Limitations and Further improvements of the study

5.3.1 Limitations

- One drawback of the study is that it only uses data from one bank, therefore it is possible that the findings cannot be generalized to other banks or financial institutions.
- Another limitation is that the study only uses quantitative data and ignores qualitative elements like client preferences, geopolitical developments, and market mood that could affect the bank's foreign exchange activity.
- The fact that the study only considers the bank's foreign exchange activities and ignores other crucial parts of its operations, such as its lending procedures, risk management, and investment activities, is another limitation.

5.3.2 Improvements

- More data from the past and present could be incorporated into the analysis to get more up-to-date insights and patterns regarding the bank's foreign currency activities.
- And it may better to have other 3 datasets daily rather than using them as monthly wise. Then it may help the model in order to understand and learn the random behavior of the inflow and outflow amount sharply.
- A more thorough investigation of the variables, such as economic indicators, political variables, and alterations in governmental rules, that influence the bank's foreign exchange transactions would be beneficial to the analysis.
- The study could also benefit from the inclusion of other outside data sources, like information on other banks' foreign exchange activities.

6 References

[1] Chen, J. (2021). Capital Outflow: Definition and Examples. Investopedia. https://www.investopedia.com/terms/c/capital-outflow.asp

[2] Wikipedia contributors. (2020). Rug plot. Wikipedia. https://en.wikipedia.org/wiki/Rug_plot

[3] Prabhakaran, S. (2022). Augmented Dickey Fuller Test (ADF Test) – Must Read Guide. Machine Learning Plus. https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/

[4] C3.ai. (2021, September 28). Root Mean Square Error (RMSE). C3 AI. https://c3.ai/glossary/data-science/root-mean-square-error-rmse/#:~:text=Root%20mean%20square%20error%20or,true%20values%20using%20Euclidean%20distance.

[5] Wei, W. W. S. (2018b). Time Series Analysis Univariate and Multivariate Methods. Pearson.

[6] Cryer, J. D., & Chan, K. (2008). Time Series Analysis: With Applications in R. Springer Science & Business Media.

[7] McDermott, R. (2005). Experiments, Political Science. In Elsevier eBooks (pp. 901–909). Elsevier BV. https://doi.org/10.1016/b0-12-369398-5/00326-1

[8] M, S. (2021). Basic understanding of Time Series Modelling with Auto ARIMAX. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/11/basic-understanding-of-time-series-modelling-with-auto-arimax/

[9] Kostadinov, S. (2019, November 10). Understanding GRU Networks - Towards Data Science. Medium. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

[10] Singh, G. (2021). Understanding Architecture of LSTM. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/01/understanding-architecture-of-lstm/

[11] Prabhakaran, S. (2022b). Vector Autoregression (VAR) – Comprehensive Guide with Examples in Python. Machine Learning Plus. https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/

7 Appendices

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# ================================================================================
# ================================================================================
# # inflow dataset and inflow time series dataset
# ================================================================================
# ================================================================================

dataset = pd.read_csv("D:\\Projects\\14197- Madushan\\CAM_PRO\\FC IN OUT\\FC_IN.csv")

dataset.info()

dataset['TRAN_DATE'] =pd.to_datetime(dataset['TRAN_DATE'])
dataset['CLS_DATE'] = pd.to_datetime(dataset['CLS_DATE'])
dataset['DUE_DATE'] = pd.to_datetime(dataset['DUE_DATE'])


dataset['TRAN_DATE'].min()
dataset['TRAN_DATE'].max()

######### removing unnecessary columns ################
dataset.dropna(how = 'all',axis =1, inplace = True)


null = dataset.isna().sum()
print('Null records: ',null)
duplicates = dataset.duplicated().sum()
print('Duplicate records: ',duplicates)
dataset.drop_duplicates(inplace= True)

date_var = []

for i in dataset.columns:
        for k in range(0,len(dataset[i])):
            if ('/' in str(dataset[i][int(k)])) & ('/' in str(dataset[i][int(k+1)])):
                    date_var.append(i)
                    k=k+1
                    break
            else:
                break

print(date_var)

dataset['T_year'] = dataset['TRAN_DATE'].dt.strftime('%Y')
dataset['T_Month'] = dataset['TRAN_DATE'].dt.strftime('%m')
dataset['T_Day'] = dataset['TRAN_DATE'].dt.strftime('%d')
```

```
#$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

dataset = dataset[dataset['TRAN_DATE']>='2018-01-01']

# =============================================================================
# merging with exchange rate - dataset
# =============================================================================

dataset['CRRYEARMON'] = dataset['T_year'] + dataset['T_Month']

data2 = pd.read_csv('D:\\Projects\\14197- Madushan\\CAM_PRO\\FC IN OUT\\FCY_CURR_CONV.csv')

null = data2.isna().sum()
print('Null records: ',null)
duplicates = data2.duplicated().sum()
print('Duplicate records: ',duplicates)
data2.drop_duplicates(inplace= True)

data2['CRRYEARMON'] = data2['CRRYEARMON'].astype(str)

dataset.rename(columns={'TRAN_CRNCY_CODE':'CURRDESC'},inplace =True)

dataset_new = pd.merge(dataset, data2, how = "left", on = ['CURRDESC','CRRYEARMON'])
dataset_new.dtypes

dataset_new['REV_RATE'].isna().sum()
dataset_new[dataset_new['REV_RATE'].isna()]

# =============================================================================
# converting all currencies to usd
# =============================================================================

USD_RATE_DATA= data2[data2['CURRDESC']=='USD'][['REV_RATE','CRRYEARMON']]
USD_RATE_DATA['usd_rate']= USD_RATE_DATA['REV_RATE']
del USD_RATE_DATA['REV_RATE']
USD_RATE_DATA['CRRYEARMON'] = USD_RATE_DATA['CRRYEARMON'].astype(str)


dataset_new_with_USD = pd.merge(dataset_new, USD_RATE_DATA, how = "left", on = ['CRRYEARMON'])

dataset_new_with_USD['Tran_in_LKR'] = dataset_new_with_USD['REV_RATE']*dataset_new_with_USD['TRAN_AMT']

dataset_new_with_USD['Amount_in_USD'] = dataset_new_with_USD['Tran_in_LKR']/dataset_new_with_USD['usd_rate']

dataset_new_with_USD.isna().sum()


# =============================================================================
# merging with interest rate
# =============================================================================

Int_rate = pd.read_csv('D:\\Projects\\14197- Madushan\\CAM_PRO\\FC IN OUT\\FC FD Int Rate.csv')

Int_rate.duplicated().sum()
Int_rate.drop_duplicates(inplace= True)

Int_rate['CRRYEARMON']=Int_rate['FDMONYEAR']
del Int_rate['FDMONYEAR']

Int_rate['CRRYEARMON'] = Int_rate['CRRYEARMON'].astype(str)


dataset_new_with_USD_and_intr = pd.merge(dataset_new_with_USD, Int_rate, how = "left", on = ['CRRYEARMON'])

dataset_new_with_USD_and_intr.isna().sum()
```

```
# ============================================================================
# merging with inflation rate
# ============================================================================

inf_data = pd.read_excel('D:\\Projects\\14197- Madushan\\CAM_PRO\\Complete_Pro\\Inflation_rate.xlsx')


inf_data['CRRYEARMON']=inf_data['INF_Y_M']
del inf_data['INF_Y_M']

inf_data['CRRYEARMON'] = inf_data['CRRYEARMON'].astype(str)

dataset_new_with_USD_and_intr_infl = pd.merge(dataset_new_with_USD_and_intr,
                                              inf_data, how = "left", on = ['CRRYEARMON'])
dataset_new_with_USD_and_intr_infl.dtypes

dataset_new_with_USD_and_intr_infl.isna().sum()

# ============================================================================
# final full dataset
# ============================================================================

full_inflow_dataset = dataset_new_with_USD_and_intr_infl

full_inflow_dataset['TRAN_DATE']= pd.to_datetime(full_inflow_dataset['TRAN_DATE'])
DF_grouped =full_inflow_dataset.groupby(by= 'TRAN_DATE').agg(
    {'Amount_in_USD':'sum',
     'usd_rate':'mean',
     'M_inf_rate':'mean',
     'AVG(T.FDINT)':'mean',
     'BASLE_CODE_DESC':(lambda x: x.mode())}).reset_index()
# ============================================================================
# multiple mode values handling
# ============================================================================
for i, val in enumerate(DF_grouped['BASLE_CODE_DESC']):
    if isinstance(val, np.ndarray):
        if len(val) > 0:
            DF_grouped.loc[i, 'BASLE_CODE_DESC'] = val[0]
        else:
            DF_grouped.loc[i, 'BASLE_CODE_DESC'] = 'INDIVIDUALS'


# ============================================================================
# creating dates manually
# ============================================================================


dates= pd.date_range('01-01-2018','12-31-2022').to_list()
#dates= pd.date_range('01-01-2010','12-31-2022').to_list()
dates = pd.DataFrame(index=dates)
dates.reset_index(inplace=True)
dates['TRAN_DATE']=dates['index']
del dates['index']
time_series_inflow_dataset = pd.merge(dates,DF_grouped, how = 'left',on = 'TRAN_DATE')

time_series_inflow_dataset.isna().sum()

# ============================================================================
# filling null records
# ============================================================================

time_series_inflow_dataset.isna().sum()
time_series_inflow_dataset.fillna(method='ffill',inplace=True)
time_series_inflow_dataset.fillna(method='bfill',inplace=True)

time_series_inflow_dataset.dtypes
```

```
time_series_inflow_dataset.index = time_series_inflow_dataset['TRAN_DATE']
del time_series_inflow_dataset['TRAN_DATE']

# ================================================================================
# creating dummies for categorical variables
# ================================================================================

df_dummies = pd.get_dummies(time_series_inflow_dataset.select_dtypes('object'))

time_series_inflow_dataset  = pd.concat([time_series_inflow_dataset,df_dummies], axis = 1)

del time_series_inflow_dataset['BASLE_CODE_DESC']
time_series_inflow_dataset.to_csv('E:\\HNB\\camp\\Datasets-4\\final two datasets\\TS_inflow.csv')
full_inflow_dataset.to_csv('final_inflow_dataset.csv')
```

TS_outflow.csv and final_outflow_dataset.csv was obtained by following same procedure as shown above.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


# ================================================================================
# # # # # # # # # # # # # # EDA
# ================================================================================

full_inflow_dataset = pd.read_csv("E:\\HNB\\camp\\Datasets-4\\final_inflow_dataset.csv")

del full_inflow_dataset['Unnamed: 0']


time_series_inflow_dataset = pd.read_csv('E:\\HNB\\camp\\Datasets-4\\final two datasets\\TS_inflow.csv')


# ================================================================================
# checking the data
# ================================================================================

full_inflow_dataset[['CIF_ID','TRAN_DATE','TRAN_AMT','CURRDESC','BASLE_CODE_DESC']]

print(full_inflow_dataset.head())
print(full_inflow_dataset.describe())
print(full_inflow_dataset.info())

full_inflow_dataset['TRAN_DATE']= pd.to_datetime(full_inflow_dataset['TRAN_DATE'])

full_inflow_dataset['TRAN_DATE'].max()
full_inflow_dataset['TRAN_DATE'].min()
full_inflow_dataset.isna().sum()
full_inflow_dataset['BASLE_CODE_DESC'].fillna('Blanks', inplace =True)
full_inflow_dataset['BASLE_CODE_DESC'].unique()

# ================================================================================
# ================================================================================
# # # EDA - Inflow
# ================================================================================
# ================================================================================
# ================================================================================
# overall histogram- initial
# ================================================================================

sns.distplot(full_inflow_dataset['Amount_in_USD'], hist=True, kde=True,
                 color = 'darkblue',
                 hist_kws={'edgecolor':'black'},
                 kde_kws={'linewidth': 1})
plt.title("Histogram of inflow amount in USD for past 4 years")
plt.xlabel("Amount in USD")
plt.axvline(full_inflow_dataset['Amount_in_USD'].mean(), color='k', linestyle='dashed', linewidth=1)
plt.axvline(full_inflow_dataset['Amount_in_USD'].max(), color='r', linestyle='dashed', linewidth=1)
plt.axvline(full_inflow_dataset['Amount_in_USD'].min(), color='r', linestyle='dashed', linewidth=1)

plt.show()
```

```
# ================================================================================
# overall histogram- scaled
# ================================================================================

scaled_amounts = np.log(full_inflow_dataset['Amount_in_USD'])

plt.title("Histogram of transformed inflow amount in USD for past 4 years")

sns.distplot(scaled_amounts, hist=True, kde=True,
             color = 'darkblue',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 1})
plt.xlabel("Amount in USD")
plt.axvline(scaled_amounts.mean(), color='k', linestyle='dashed', linewidth=1)
plt.axvline(scaled_amounts.max(), color='r', linestyle='dashed', linewidth=1)
plt.axvline(scaled_amounts.min(), color='r', linestyle='dashed', linewidth=1)

plt.show()
# ================================================================================
# bar plot of currency types
# ================================================================================
counts = full_inflow_dataset['CURRDESC'].value_counts().sort_values()

sns.countplot(y='CURRDESC', data = full_inflow_dataset ,
             order = counts.index).set(title = 'Transactions by Currency types',
                                       ylabel = 'Currency Type', xlabel ='No. of Transactions')
plt.show()
# ================================================================================
# bar plot of basle types - counts
# ================================================================================
counts = full_inflow_dataset['BASLE_CODE_DESC'].value_counts().sort_values()
sns.countplot(y='BASLE_CODE_DESC', data = full_inflow_dataset , order = counts.index)
ax = plt.gca()
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'   {int(width)}', (x + width - 0.1, y + height / 2),
                ha='left', va='center', fontsize=12, fontstyle='italic')



plt.xticks(fontsize=9,fontweight='bold')
plt.yticks(fontsize=9,fontweight='bold')
plt.title('Transactions by Basle types', fontweight = 'bold', fontsize = 15)
plt.ylabel( 'Basle Type',fontweight='bold', fontsize=12)
plt.xlabel( 'No. of Transactions', fontweight='bold', fontsize=12)
plt.show()

# ================================================================================
# pie charts for retail and coperate - counts
# ================================================================================

full_inflow_dataset['r_c']=np.where((full_inflow_dataset['BASLE_CODE_DESC']== 'INDIVIDUALS') |
                                    (full_inflow_dataset['BASLE_CODE_DESC']== 'STAFF') |
                                    (full_inflow_dataset['BASLE_CODE_DESC']== 'UNCLASSIFIED') |
                                    (full_inflow_dataset['BASLE_CODE_DESC']== 'Blanks'),
                                    'Retail' , 'Co-operate')
df_pie = full_inflow_dataset['r_c'].value_counts().reset_index()
fig, ax = plt.subplots()
ax.pie(df_pie['r_c'], labels=df_pie['index'], autopct='%1.1f%%', startangle=90,
       pctdistance=0.8, wedgeprops=dict(width=0.4))

# Add circle to create a hole
circle = plt.Circle((0, 0), 0.4, color='white')
fig.gca().add_artist(circle)
```

```
# Set plot title
ax.set_title("Distribution of Sales")
# Show the plot
plt.show()

# ================================================================================
# bar plot of basle types - amounts
# ================================================================================
amounts = full_inflow_dataset[['Amount_in_USD','BASLE_CODE_DESC']].groupby(
    by='BASLE_CODE_DESC').sum()['Amount_in_USD'].reset_index().sort_values(by= 'Amount_in_USD')

sns.barplot(x='Amount_in_USD', data = amounts ,y='BASLE_CODE_DESC' )

ax = plt.gca()
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'  {int(width)}', (x + width + 0.1, y + height / 2),
                ha='left', va='center', fontsize=9)
plt.xticks(fontsize=9,fontweight='bold')
plt.yticks(fontsize=9,fontweight='bold')
plt.title('Transaction Amount by Basle types', fontweight = 'bold', fontsize = 15)
plt.ylabel( 'Basle Type',fontweight='bold', fontsize=12)
plt.xlabel( 'Transaction Amount (USD)', fontweight='bold', fontsize=12)

plt.show()

# ================================================================================
# bar plot for currency types - average amount in USD
# ================================================================================

Avg_amounts = full_inflow_dataset[['Amount_in_USD','CURRDESC']].groupby(
    by='CURRDESC').mean()['Amount_in_USD'].reset_index().sort_values(by= 'Amount_in_USD')
sns.barplot(x='Amount_in_USD', data = Avg_amounts ,y='CURRDESC')

ax = plt.gca()
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'  {int(width)}', (x + width + 0.1, y + height / 2),
                ha='left', va='center', fontsize=12)
plt.xticks(fontsize=11,fontweight='bold')
plt.yticks(fontsize=11,fontweight='bold')
plt.title('Average Transaction Amount by Currency types', fontweight = 'bold', fontsize = 15)
plt.ylabel( 'Currency Type',fontweight='bold', fontsize=12)
plt.xlabel( 'Avg.Transaction Amount (USD)', fontweight='bold', fontsize=12)

plt.show()

# ================================================================================
# stacked bar chart of currency types by year by avg.amount in USD
# ================================================================================

Avg_amounts_y = full_inflow_dataset[['T_year_y','Amount_in_USD','CURRDESC']].groupby(
    by=['T_year_y','CURRDESC']).mean()['Amount_in_USD'].reset_index().sort_values(by= 'T_year_y')
Avg_amounts_y = Avg_amounts_y[Avg_amounts_y['CURRDESC']!='LKR']

# pivot the dataframe to get the total amount for each currency in each year
df_pivot = Avg_amounts_y.pivot_table(index='T_year_y', columns='CURRDESC',
                                     values='Amount_in_USD', aggfunc='mean')

# create stacked bar chart
df_pivot.plot(kind='bar', stacked=True)
# set labels and title
plt.xlabel('Year')
plt.ylabel('Amount')
plt.title('Total Amount by Currency and Year')
```

```
# ================================================================================
# EDA for timeseries
# ================================================================================

time_series_inflow_dataset.columns
time_series_inflow_dataset['TRAN_DATE'] = pd.to_datetime(time_series_inflow_dataset['TRAN_DATE'])

sns.lineplot(y=time_series_inflow_dataset['Amount_in_USD'], x =time_series_inflow_dataset['TRAN_DATE'])


sns.lineplot(y=time_series_inflow_dataset['usd_rate'], x =time_series_inflow_dataset['TRAN_DATE'])
sns.lineplot(y=time_series_inflow_dataset['M_inf_rate'], x =time_series_inflow_dataset['TRAN_DATE'])
sns.lineplot(y=time_series_inflow_dataset['AVG(T.FDINT)'], x =time_series_inflow_dataset['TRAN_DATE'])

# ================================================================================
# histograms
# ================================================================================


for i in time_series_inflow_dataset.drop(['TRAN_DATE'],axis = 1).columns:
    plt.title("Histogram of "+i+" for past 4 years")

    sns.distplot(time_series_inflow_dataset[i], hist=True, kde=True,
                 color = 'darkblue',
                 hist_kws={'edgecolor':'black'},
                 kde_kws={'linewidth': 1})
    plt.xlabel(i)
    plt.axvline(time_series_inflow_dataset[i].mean(), color='k', linestyle='dashed', linewidth=1)
    plt.axvline(time_series_inflow_dataset[i].max(), color='r', linestyle='dashed', linewidth=1)
    plt.axvline(time_series_inflow_dataset[i].min(), color='r', linestyle='dashed', linewidth=1)

    plt.show()

# ================================================================================
# correlation matrix
# ================================================================================
time_series_inflow_dataset.columns

corr_matrix = time_series_inflow_dataset.corr()
plt.figure(figsize = (10,10))
sns.heatmap(corr_matrix, annot=True )
plt.title('Inflow Correlation Matrix')
plt.show()

# removing usd_rate
corr_matx = time_series_inflow_dataset.drop("usd_rate", axis =1).corr()
plt.figure(figsize = (10,10))
sns.heatmap(corr_matx, annot=True )
plt.title('Inflow Correlation Matrix')
plt.show()
# ================================================================================
# pairplots
# ================================================================================

sns.pairplot(time_series_inflow_dataset.iloc[:,:5],diag_kind = 'hist')
plt.show()
```

The code lines for outflow EDA were same as for the inflow EDA. Only the dataset was changed at the beginning of the code.

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import kpss
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error
from statsmodels.api import add_constant
from statsmodels.stats.outliers_influence import variance_inflation_factor

time_series_inflow_dataset = pd.read_csv('E:\\HNB\\camp\\Datasets-4\\final two datasets\\TS_inflow.csv')

# ================================================================================
# # # # # INFLOW
# ================================================================================

dataset = time_series_inflow_dataset.copy()

sns.heatmap(dataset.drop('usd_rate',axis =1).corr())


# read in your dataset as a pandas DataFrame
df = dataset.drop(["usd_rate"],axis =1)

# create a design matrix with the independent variables
X = df[df.columns]

# add a constant to the design matrix (required for statsmodels)
X = add_constant(X)

# calculate the VIF values for each variable
vif = pd.DataFrame()
vif["variables"] = X.columns
vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

# print the VIF values
print(vif)
# ================================================================================
# checking stationarity
# ================================================================================


# Apply ADF test
result = adfuller(dataset['Amount_in_USD'])

# Create a table to display the results
adf_table = pd.DataFrame({'ADF Test Statistic': [result[0]],
                          'p-value': [result[1]],
                          '1% Critical Value': [result[4]['1%']],
                          '5% Critical Value': [result[4]['5%']],
                          '10% Critical Value': [result[4]['10%']]})

# Display the table
print(adf_table)

# Apply KPSS test
result_kpss = kpss(dataset['Amount_in_USD'])

# Create a table to display the results
kpss_table = pd.DataFrame({'KPSS Test Statistic': [result_kpss[0]],
                          'p-value': [result_kpss[1]],
                          '10% Critical Value': [result_kpss[3]['10%']],
                          '5% Critical Value': [result_kpss[3]['5%']],
                          '2.5% Critical Value': [result_kpss[3]['2.5%']],
                          '1% Critical Value': [result_kpss[3]['1%']]})

# Display the table
print(kpss_table)

# ================================================================================
# time series components
# ================================================================================
dataset.index = pd.to_datetime(dataset.index)

seasonal_decompose(dataset['Amount_in_USD']).plot()
plt.show()
```

```
# ================================================================================
# time series components
# ================================================================================
dataset.index = pd.to_datetime(dataset.index)

seasonal_decompose(dataset['Amount_in_USD']).plot()
plt.show()


# Perform seasonal decomposition
result = seasonal_decompose(dataset['Amount_in_USD'], model='additive', period=12)

# Plot the seasonal component
result.seasonal.plot()
plt.show()


seasonal_period = 1
# Perform seasonal decomposition
result = seasonal_decompose(dataset['Amount_in_USD'], model='additive', period=seasonal_period)

# Plot the trend component
result.trend.plot()
plt.show()

# Print the first few values of the trend series
print(result.trend.head())

# ================================================================================
# making series stationary
# ================================================================================

# Load your time series data
df = dataset

# Perform seasonal decomposition
result = seasonal_decompose(df['Amount_in_USD'], model='additive', period=11)
```

```
# Extract the trend, seasonal, and residual components
trend = result.trend
seasonal = result.seasonal
residual = result.resid

# Remove the trend and seasonal pattern from the original data
detrended = df['Amount_in_USD'] - trend
deseasonalized = detrended - seasonal

# Plot the original data, trend, seasonal, and residual components
result.plot()
plt.show()

# Plot the detrended and deseasonalized data
deseasonalized.plot()
plt.title("Stationary time series of inflow amount in USD")
plt.xlabel("Date")
plt.ylabel("Amount in USD")
plt.show()

re = seasonal_decompose(deseasonalized.dropna(), model='additive', period=11)
re.plot()

diff_data = df['Amount_in_USD'].diff().dropna()
plt.plot(diff_data)

# ================================================================================
# rechecking the stationarity
# ================================================================================
result_kpss = kpss(deseasonalized.dropna())

# Create a table to display the results
kpss_table = pd.DataFrame({'KPSS Test Statistic': [result_kpss[0]],
                           'p-value': [result_kpss[1]],
                           '10% Critical Value': [result_kpss[3]['10%']],
                           '5% Critical Value': [result_kpss[3]['5%']],
                           '2.5% Critical Value': [result_kpss[3]['2.5%']],
                           '1% Critical Value': [result_kpss[3]['1%']]})

# Display the table
print(kpss_table)
```

```
# Apply ADF test
result_ADF = kpss(diff_data)

# Create a table to display the results
adf_table = pd.DataFrame({'ADF Test Statistic': [result_ADF[0]],
                          'p-value': [result_ADF[1]],
                          '1% Critical Value': [result_ADF[3]['1%']],
                          '5% Critical Value': [result_ADF[3]['5%']],
                          '10% Critical Value': [result_ADF[3]['10%']]})

# Display the table
print(adf_table)

# ================================================================================
# Scalling data
# ================================================================================

dataset = time_series_inflow_dataset.copy()
scle = StandardScaler()

dataset['usd_rate'] = scle.fit_transform(dataset['usd_rate'].values.reshape(-1,1))
dataset['M_inf_rate'] = scle.fit_transform(dataset['M_inf_rate'].values.reshape(-1,1))
dataset['AVG(T.FDINT)'] = scle.fit_transform(dataset['AVG(T.FDINT)'].values.reshape(-1,1))
#dataset['Amount_in_USD'] = scle.fit_transform(dataset['Amount_in_USD'].values.reshape(-1,1))

# ================================================================================
# ACF & PACF plots
# ================================================================================
fig , ax = plt.subplots(1,2, figsize = (20,5))

plot_acf(deseasonalized.dropna(), lags = 15,ax=ax[0])
plt.xticks(fontsize=9,fontweight='bold')
plt.yticks(fontsize=9,fontweight='bold')
plt.show()
plot_pacf(deseasonalized.dropna(), lags = 15, method = "ols",ax=ax[1])
plt.xticks(fontsize=9,fontweight='bold')
plt.yticks(fontsize=9,fontweight='bold')
plt.show()
'''
p = 1
d = 1
q = 1
'''
# ================================================================================
# ARIMA - forecasting
# ================================================================================


#df_usd = dataset['Amount_in_USD'].resample('M').sum()


#df_usd = np.log(dataset['Amount_in_USD'])
#df_usd = dataset['Amount_in_USD']
df_usd = deseasonalized.dropna()


train_size  = int(len(df_usd)*0.8)
train, test = df_usd [: train_size], df_usd [train_size:]



model = ARIMA(train , order =( 1,1,1))
model_fit =  model.fit()

predictions = model_fit.predict(start = len (train), end = len( train)+len(test)-1 , typ ='levels')
predictions_train  = model_fit.predict(start = 1, end = len( train)-1 , typ ='levels')


predictions_train.index= pd.to_datetime(predictions_train.index)


predictions.index= pd.to_datetime(predictions.index)
test.index= pd.to_datetime(test.index)
train.index = pd.to_datetime(train.index)


# ================================================================================
# predicting for 30 days ahead
# ================================================================================
future_period = 90
forecast_df = pd.DataFrame()
future_pred = model_fit.predict(start = len (df_usd), end = len(df_usd) + future_period, typ ='levels')
future_values = future_pred.values+trend.iloc[-future_period-1:]+seasonal.iloc[-future_period-1:]

predic_train = predictions_train.values+trend.iloc[:len(train)-1]+seasonal.iloc[:len(train)-1]
```

```
# Create a new time index that spans the next 12 months from T
time_index = pd.date_range(start=df_usd.index[-1], periods=future_period+1, freq='d')

future_values.index = time_index

plt.title('ARIMA - Forecasting')
plt.xlabel('Date')
plt.ylabel('Amount (USD)')

train_values = train+trend.iloc[: train_size]+seasonal.iloc[: train_size]
test_values = test+trend.iloc[train_size:]+seasonal.iloc[train_size:]
predicted_values = predictions+trend.iloc[train_size:]+seasonal.iloc[train_size:]

plt.plot(train_values, label = 'Trainned values')
plt.plot(test_values, label ='Test values')
plt.plot(predicted_values, label ='Predicted')
plt.legend()
plt.show()

plt.plot(train_values, label = 'Trainned values')
plt.plot(test_values, label ='Test values')
plt.plot(predic_train, label ='Predicted')
plt.legend()
plt.show()

plt.plot(future_values, label ='future values')
plt.legend()
plt.show()


MSE = mean_squared_error(test_values.dropna(), predicted_values.dropna())
print("RMSE :",np.sqrt(MSE))
print(mean_absolute_percentage_error(test_values.dropna(), predicted_values.dropna()))

print(model_fit.summary())

MSE = mean_squared_error(train_values.dropna()[:-1], predic_train.dropna())
print("RMSE :",np.sqrt(MSE))
print(mean_absolute_percentage_error(train_values.dropna()[:-1], predic_train.dropna()))
# ========================================================================
# Auto - ARIMA
# ========================================================================

dataset = time_series_inflow_dataset.copy()
scle = StandardScaler()

dataset['usd_rate'] = scle.fit_transform(dataset['usd_rate'].values.reshape(-1,1))
dataset['M_inf_rate'] = scle.fit_transform(dataset['M_inf_rate'].values.reshape(-1,1))
dataset['AVG(T.FDINT)'] = scle.fit_transform(dataset['AVG(T.FDINT)'].values.reshape(-1,1))
#dataset['Amount_in_USD'] = scle.fit_transform(dataset['Amount_in_USD'].values.reshape(-1,1))

dataset['Amount_in_USD']=deseasonalized
dataset.dropna(inplace = True)
dataset.drop("usd_rate",axis =1, inplace =True)
dataset.iloc[round(dataset.shape[0]*0.8)]

train_timeseries = dataset.iloc[:round(dataset.shape[0]*0.8),:]
test_timeseries = dataset.iloc[round(dataset.shape[0]*0.8):,:]

from pmdarima.arima import auto_arima

arima_model = auto_arima(train_timeseries['Amount_in_USD'],seasonal_period= 11,
                         exogenous = train_timeseries.iloc[:,1:]
                         ,trace = True)
arima_model.order

forecast = arima_model.predict(n_periods = len(test_timeseries),exogenous = test_timeseries.iloc[:,1:])

plt.plot(test_timeseries['Amount_in_USD'],label = 'Testing series')
plt.plot(forecast,label = 'Forecasted series')
plt.plot(train_timeseries['Amount_in_USD'],label = 'Training series')


test_values = test_timeseries['Amount_in_USD']+trend.iloc[train_size:]+seasonal.iloc[train_size:]
predicted_values = forecast+trend.iloc[train_size:]+seasonal.iloc[train_size:]

plt.plot(train_timeseries['Amount_in_USD']+trend.iloc[: train_size]+seasonal.iloc[: train_size], label = 'Trainned values')
plt.plot(test_timeseries['Amount_in_USD']+trend.iloc[train_size:]+seasonal.iloc[train_size:], label ='Test values')
plt.plot(forecast+trend.iloc[train_size:]+seasonal.iloc[train_size:] , label = 'Predicted')

plt.legend()
```

```
plt.title('Auto - ARIMA - Forecasting')
plt.xlabel('Date')
plt.ylabel('Amount (USD)')
plt.show()

print(arima_model.summary())
#%matplotlib qt
MSE = mean_squared_error(test_values.dropna(), predicted_values.dropna())
print("RMSE :",np.sqrt(MSE))
print(mean_absolute_percentage_error(test_values.dropna(), predicted_values.dropna()))



# ============================================================================
# ARIMAX - forecasting
# ============================================================================

dataset = time_series_inflow_dataset.copy()


dataset['Amount_in_USD']=deseasonalized
dataset.dropna(inplace = True)
dataset.drop("usd_rate", axis = 1, inplace =True)
dataset.iloc[round(dataset.shape[0]*0.8)]

train_timeseries = dataset.iloc[:round(dataset.shape[0]*0.8),:]
test_timeseries = dataset.iloc[round(dataset.shape[0]*0.8):,:]


sarimax_model = SARIMAX(train_timeseries['Amount_in_USD'], seasonal_order=(1,1,1,11), trend='c',
                        order=(1, 1, 1), exog=train_timeseries.iloc[:, 1:])

res = sarimax_model.fit(disp=True)

res.summary()

forecast = res.forecast(steps=len(test_timeseries), exog=test_timeseries.iloc[:, 1:])

forecasted = res.predict(start = test_timeseries.index[0],
                         end = test_timeseries.index[-1],
                         exog = test_timeseries.iloc[:,1:])
train_values = train_timeseries['Amount_in_USD']+trend.iloc[: train_size]+seasonal.iloc[: train_size]
test_values = test_timeseries['Amount_in_USD']+trend.iloc[train_size:]+seasonal.iloc[train_size:]
predicted_values = forecast+trend.iloc[train_size:]+seasonal.iloc[train_size:]

plt.title('ARIMAX - Forecasting')

plt.plot(train_values, label = 'Trainned values')
plt.plot(test_values, label ='Test values')
plt.plot(predicted_values, label ='Predicted')
plt.xlabel('Date')
plt.ylabel('Amount (USD)')
plt.legend()
plt.show()


MSE = mean_squared_error(test_values.dropna(), predicted_values.dropna())
print("RMSE :",np.sqrt(MSE))
print(mean_absolute_percentage_error(test_values.dropna(), predicted_values.dropna()))

print(res.summary())
```

By changing the TS_inflow into TS_outflow, above 03 models were fitted using the same codes.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from keras.models import Sequential
from keras.layers import GRU, Dense, Dropout
from keras.callbacks import EarlyStopping
from sklearn.preprocessing import StandardScaler
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error


time_series_inflow_dataset = pd.read_csv('E:\\HNB\\camp\\Datasets-4\\final two datasets\\TS_inflow.csv')

dataset = time_series_inflow_dataset.copy()
# Convert date to datetime format
dataset['TRAN_DATE'] = pd.to_datetime(dataset['TRAN_DATE'])

# Sort the dataset by date
dataset.sort_values(by=['TRAN_DATE'], inplace=True)

# Set the date as the index
dataset.set_index('TRAN_DATE', inplace=True)
#del dataset['usd_rate']

# Load the data
df = dataset

# Decompose the data into trend, seasonal, and residual components
decomposition = seasonal_decompose(df['Amount_in_USD'], model='additive', period=11)
trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

# Deseasonalize the data
deseasonalized = df['Amount_in_USD'] - seasonal

# Detrend the data
detrended = deseasonalized - trend
#detrended = deseasonalized.diff().dropna()

df['Amount_in_USD'] = detrended
df.dropna(inplace=True)
df.columns

timeseries_data = df
train_size=round(timeseries_data.shape[0]*0.8)

train_timeseries = timeseries_data.iloc[:train_size,:]
test_timeseries = timeseries_data.iloc[train_size:,:]

timeseries_data = train_timeseries.copy()

scaler = StandardScaler()
scaler = scaler.fit(timeseries_data)
timeseries_data_scaled = scaler.transform(timeseries_data)


trainX=[]
trainY=[]


n_future = 30
n_past = 11

for i in range(n_past, len(timeseries_data_scaled)-n_future+1):
    trainX.append(timeseries_data_scaled[i-n_past:i, 0:timeseries_data.shape[1]])
    trainY.append(timeseries_data_scaled[i+n_future - 1:i + n_future,0])

trainX,trainY = np.array(trainX),np.array(trainY)


trainX,trainY = np.array(trainX),np.array(trainY)

print('trainX shape == {}'.format(trainX.shape))
print('trainY shape == {}'.format(trainY.shape))


model = Sequential()
model.add(GRU(units=50, return_sequences=True, input_shape=(trainX.shape[1], trainX.shape[2])))
model.add(GRU(units=50, return_sequences=True))
model.add(GRU(units=40, return_sequences=True))
model.add(GRU(units=50))
model.add(Dense(units=1))
```

```
model.compile(optimizer = 'adam', loss= 'mse')
model.summary()
es = EarlyStopping(monitor= 'val_loss', patience= 10 )

history = model.fit(trainX,trainY,epochs=100,batch_size = 200, validation_split=0.1, verbose = 1, callbacks = [es])
plt.title("Model Loss curve")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.plot(history.history['loss'], label = 'Training loss')
plt.plot(history.history['val_loss'],label = 'Validation loss')
plt.legend()
plt.show()

n_future = len(test_timeseries)

forecast = model.predict(trainX[-n_future:])

forecast_copies = np.repeat(forecast, timeseries_data.shape[1],axis=1)
y_pred = scaler.inverse_transform(forecast_copies)[:,0]

#forecast_period = pd.date_range('12-01-2022','02-28-2023').to_list()
forecast_period = pd.date_range(list(timeseries_data.index)[-1],periods = n_future, freq ='1d').tolist()
forecasted_data = pd.DataFrame(index = forecast_period)
forecasted_data.dtypes

forecasted_data['Predicted amount']= y_pred + trend[-n_future:]+seasonal[-n_future:]

plt.plot(timeseries_data['Amount_in_USD']+trend[-len(timeseries_data):]+
         seasonal[-len(timeseries_data):],label = 'Training series')
plt.plot(test_timeseries['Amount_in_USD']+trend[-len(test_timeseries):]+
         seasonal[-len(test_timeseries):],label = 'Testing series')
plt.plot(forecasted_data,label = 'Forecasted series')
plt.title('GRU - Forecasting')
plt.xlabel('Date')
plt.ylabel('Amount (USD)')

plt.legend()
plt.show()

test_values = test_timeseries['Amount_in_USD']+trend[-len(test_timeseries):]+seasonal[-len(test_timeseries):]
print("RMSE : ",  np.sqrt(mean_squared_error(test_values[-len(trend[-len(test_timeseries):]):].fillna(0),
                                             forecasted_data[:len(test_values)].fillna(0))))
```

Same as previously done, the GRU model was fitted for outflow using same code.

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from keras.callbacks import EarlyStopping
from sklearn.preprocessing import StandardScaler
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error


time_series_inflow_dataset = pd.read_csv('E:\\HNB\\camp\\Datasets-4\\final two datasets\\TS_inflow.csv')

dataset = time_series_inflow_dataset.copy()
# Convert date to datetime format
dataset['TRAN_DATE'] = pd.to_datetime(dataset['TRAN_DATE'])

# Sort the dataset by date
dataset.sort_values(by=['TRAN_DATE'], inplace=True)

# Set the date as the index
dataset.set_index('TRAN_DATE', inplace=True)
#del dataset['usd_rate']

# Load the data
df = dataset

# Decompose the data into trend, seasonal, and residual components
decomposition = seasonal_decompose(df['Amount_in_USD'], model='additive', period=11)
trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

# Deseasonalize the data
deseasonalized = df['Amount_in_USD'] - seasonal

# Detrend the data

detrended = deseasonalized - trend
#detrended = deseasonalized.diff().dropna()

df['Amount_in_USD'] = detrended

df.dropna(inplace=True)
df.columns

timeseries_data = df
train_size=round(timeseries_data.shape[0]*0.8)

train_timeseries = timeseries_data.iloc[:train_size,:]
test_timeseries = timeseries_data.iloc[train_size:,:]

timeseries_data = train_timeseries.copy()

scaler = StandardScaler()
scaler = scaler.fit(timeseries_data)
timeseries_data_scaled = scaler.transform(timeseries_data)

trainX=[]
trainY=[]

n_future = 30
n_past = 11

for i in range(n_past, len(timeseries_data_scaled)-n_future+1):
    trainX.append(timeseries_data_scaled[i-n_past:i, 0:timeseries_data.shape[1]])
    trainY.append(timeseries_data_scaled[i+n_future - 1:i + n_future,0])

trainX,trainY = np.array(trainX),np.array(trainY)

print('trainX shape == {}'.format(trainX.shape))
print('trainY shape == {}'.format(trainY.shape))

model = Sequential()
model.add(LSTM(64, activation = 'relu', input_shape=(trainX.shape[1], trainX.shape[2]),return_sequences = True))
model.add(Dropout(0.2))

#model.add(LSTM(64,activation = 'relu', return_sequences=True))
#model.add(LSTM(16,activation = 'relu', return_sequences=True))
model.add(LSTM(32,activation = 'relu', return_sequences=False))
model.add(Dropout(0.3))
model.add(Dense(trainY.shape[1]))

model.compile(optimizer = 'adam', loss= 'mse')
model.summary()
```

```
#es = EarlyStopping(monitor= 'val_loss', patience= 10 )

history = model.fit(trainX,trainY,epochs=100,batch_size = 200, validation_split=0.1, verbose = 1)
plt.title("Model Loss curve")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.plot(history.history['loss'], label = 'Training loss')
plt.plot(history.history['val_loss'],label = 'Validation loss')
plt.legend()
plt.show()

n_future = len(test_timeseries)

forecast = model.predict(trainX[-n_future:])

forecast_copies = np.repeat(forecast, timeseries_data.shape[1],axis=1)
y_pred = scaler.inverse_transform(forecast_copies)[:,0]

#forecast_period = pd.date_range('12-01-2022','02-28-2023').to_list()
forecast_period = pd.date_range(list(timeseries_data.index)[-1],periods = n_future, freq ='1d').tolist()
forecasted_data = pd.DataFrame(index = forecast_period)
forecasted_data.dtypes

forecasted_data['Predicted amount']= y_pred + trend[-n_future:]+seasonal[-n_future:]

plt.plot(timeseries_data['Amount_in_USD']+trend[-len(timeseries_data):]+
        seasonal[-len(timeseries_data):],label = 'Training series')
plt.plot(test_timeseries['Amount_in_USD']+trend[-len(test_timeseries):]+
        seasonal[-len(test_timeseries):],label = 'Testing series')
plt.plot(forecasted_data,label = 'Forecasted series')
plt.title('LSTM - Forecasting')
plt.xlabel('Date')
plt.ylabel('Amount (USD)')

plt.legend()
plt.show()
test_values = test_timeseries['Amount_in_USD']+trend[-len(test_timeseries):]+seasonal[-len(test_timeseries):]
print("RMSE : ",  np.sqrt(mean_squared_error(test_values[-len(trend[-len(test_timeseries):]):].fillna(0),
                                        forecasted_data[:len(test_values)].fillna(0))))
```

```python
import pandas as pd
import numpy as np
from statsmodels.tsa.api import VAR
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose

df = pd.read_csv('E:\\HNB\\camp\\Datasets-4\\final two datasets\\TS_inflow.csv')


dataset = df.iloc[:,0:5]

dataset.drop("usd_rate", axis =1, inplace = True)
dataset['TRAN_DATE']= pd.to_datetime(dataset['TRAN_DATE'])
dataset.index = dataset['TRAN_DATE']
del dataset['TRAN_DATE']

result = seasonal_decompose(dataset['Amount_in_USD'])

trend = result.trend
seasonal = result.seasonal
residuals = result.resid

dataset['Amount_in_USD'] = dataset['Amount_in_USD']-trend -seasonal
dataset.dropna(inplace = True)

dataset.iloc[round(dataset.shape[0]*0.8)]

train_timeseries = dataset.iloc[:round(dataset.shape[0]*0.8),:]
test_timeseries = dataset.iloc[round(dataset.shape[0]*0.8):,:]

aic = []
for i in [1,2,3,4,5,6,7,8,9,10]:
    model = VAR(train_timeseries)
    results = model.fit(i)
    print('Order =', i)
    print('AIC: ', results.aic)
    print('BIC: ', results.bic)
    aic.append(results.aic)

result = model.fit(round(min(aic)))
print(result.summary() )

pred = result.forecast(result.endog, steps =len(test_timeseries))
# pred_train = result.forecast(result.endog, steps = len(train_timeseries))

pred_df = pd.DataFrame(pred ,index = test_timeseries.index, columns = test_timeseries.columns )
# pred_df_train = pd.DataFrame(pred_train, index = train_timeseries.index, columns  = train_timeseries.columns)

forecast = pred_df['Amount_in_USD']

test_values = test_timeseries['Amount_in_USD']+trend.iloc[-len(test_timeseries):]+seasonal.iloc[-len(test_timeseries):]
predicted_values = forecast+trend.iloc[-len(test_timeseries):]+seasonal.iloc[-len(test_timeseries):]
train_values = train_timeseries['Amount_in_USD'] +trend.iloc[:len(train_timeseries)]+seasonal.iloc[:len(train_timeseries)]

plt.figure(figsize=(12,6))
plt.plot(train_timeseries.index, train_values[3:], label = 'Train')
plt.plot(test_timeseries.index, test_values[:-3], label = 'Test')
plt.plot(pred_df.index, predicted_values[:-3], label = 'Predicted')
plt.legend()
plt.xlabel("Date")
plt.ylabel("Amount in USD")
plt.title("Inflow Forecast using VAR model")
plt.show()

mse = mean_squared_error(test_values.dropna(), predicted_values.dropna())
print(f"mean squared error : {mse :.4f}")
print("RMSE :", np.sqrt(mse))
print("MPE :", mean_absolute_percentage_error(test_values.dropna(), predicted_values.dropna()))
```

Code for model validation and Residual diagnosis

```
# ================================================================================
# model validataion and residual diagnosis
# ================================================================================
# Model Validation
from sklearn.model_selection import TimeSeriesSplit

# define the number of folds for cross validation
n_splits = 5

# define the size of each fold as a percentage of the total dataset size
fold_size = int(len(df_usd) / n_splits)

# create the time series split object
tscv = TimeSeriesSplit(n_splits=n_splits)

# initialize an empty list to store the RMSE values for each fold
rmse_values = []

# initialize an empty dataframe to store the results
results_df = pd.DataFrame(columns=["Fold", "RMSE"])

# loop over the folds in the time series split object
for i, (train_index, test_index) in enumerate(tscv.split(df_usd)):

    # split the data into train and test sets for this fold
    train_data = df_usd[train_index]
    test_data = df_usd[test_index]

    # train the model on the training set for this fold
    model = ARIMA(train_data, order=(1,1,1))
    model_fit = model.fit()

    # make predictions on the test set for this fold
    predictions = model_fit.predict(start=len(train_data), end=len(train_data)+len(test_data)-1, typ='levels')

    # calculate the RMSE for this fold and append it to the list of RMSE values
    rmse = np.sqrt(mean_squared_error(test_data, predictions))
    rmse_values.append(rmse)

    # add the results for this fold to the dataframe
    results_df = results_df.append({"Fold": i+1, "RMSE": rmse}, ignore_index=True)
# print the results dataframe
print(results_df)

# print the mean and standard deviation of the RMSE values across all folds
print("Mean RMSE:", np.mean(rmse_values))
print("Standard deviation of RMSE:", np.std(rmse_values))



# ================================================================================
# Residual Diagnosis
# ================================================================================
residuals = model_fit.resid
plt.plot(residuals)
plt.title('Residual Plot')
plt.xlabel('Date')
plt.ylabel('Residuals')
plt.show()

acf = plot_acf(residuals, lags=20)
pacf = plot_pacf(residuals, lags=20)


residuals = model_fit.resid
plt.hist(residuals, bins=30)
plt.title('Histogram of Residuals')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.show()
from statsmodels.graphics.gofplots import qqplot

qqplot(residuals, line='s')
plt.title('Normal Probability Plot')
plt.show()
```