

**BEHAVIORAL FINANCE PATTERNS IN PREDICTING GLOBAL FINANCIAL CRISES: AN ANALYTICAL APPROACH TO CRISIS FORECASTING**

**CS6510 – PROJECT 1**

**REPORT**

***Submitted by***

**DHARSHANI A - 23MSP3068**

***In partial fulfilment for the award of the degree of***

**POST GRADUATE PROGRAMME**

**INTERNATIONAL CENTRE FOR HIGHER EDUCATION AND RESEARCH**

**VIT BANGALORE**

**DECEMBER, 2023**



**BONAFIDE CERTIFICATE**

Certified that this project report **“BEHAVIORAL FINANCE PATTERNS IN PREDICTING GLOBAL FINANCIAL CRISES: AN ANALYTICAL APPROACH TO CRISIS FORECASTING”** is the bonafide record of work done by **“DHARSHANI A - 23MSP3068”** who carried out the project work under my supervision.

**Signature of the Supervisor Signature of Director**

|  |  |
| --- | --- |
| **Dr. Pitchumani Angayarkanni S**  **Professor,**  ICER  VIT Bangalore | **Prof. Prema M**  **Director,**  ICER  VIT Bangalore. |

**Evaluation Date: 12/12/2023**

****

**ACKNOWLEDGEMENT**

I express my sincere gratitude to our director of ICER **Prof. Prema M.** for their support and for providing the required facilities for carrying out this study.

I wish to thank my faculty supervisor(s), **Dr. Pitchumani Angayarkanni S, Professor,** ICER for extending help and encouragement throughout the project. Without his/her continuous guidance and persistent help, this project would not have been a success for me.

I am grateful to all the members of ICER, my beloved parents, and friends for extending the support, who helped us to overcome obstacles in the study.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | **4** |
|  | **LIST OF FIGURES** | **5** |
|  | **LIST OF TABLES** | **5** |
| **1** | **INTRODUCTION** | **8** |
| **2** | **LITERATURE REVIEW** | **9** |
| **3** | **OBJECTIVE** | **10** |
| **4** | **PROPOSED METHODOLOGY** | **11** |
| **5** | **TOOLS AND TECHNIQUES** | **11** |
| **6** | **IMPLEMENTATION** | **12** |
| **7** | **RESULTS AND DISCUSSIONS** | **40** |
| **8** | **CONCLUSION** | **42** |
| **9** | **FUTURE ENHANCEMENT** | **43** |
| **10** | **APPENDICES** | **44** |
|  | Appendix-1: Code – Full Code | **44** |
|  | Appendix-2: Plagiarism Report | **80** |
| **11** | **REFERENCES** | **81** |
| **12** | **WORKLOG** | **82** |

**ABSTRACT**

The financial sector is ever-evolving, and there's a critical need for models that can predict crises before they occur. This study focuses on developing a predictive model for forecasting financial crises by examining extensive historical economic data, market trends, and information from previous global financial downturns. This is achieved by utilizing data from Global Financial Crisis Analytics, incorporating SARIMAX time series analysis, and employing advanced machine learning techniques, including LSTM neural networks. These methods are used to identify economic patterns and interactions.

The research aims to achieve two main objectives: predicting global financial crises and exploring the underlying causes of economic fluctuations. The approach combines principles of behavioral finance with analytical rigor, utilizing both regression analysis and pattern recognition to predict and understand financial instability. The analysis reveals a significant correlation between certain economic indicators and the onset of financial crises.

The results of this research carry important implications for policy formulation and investment decision-making. They offer new strategies for mitigating financial risks and strengthening the economy's resilience against turbulence. By merging proven econometric methods with cutting-edge machine learning technologies, including LSTM neural networks, this study provides new insights into behavioral finance and a robust framework for anticipating economic downturns.

Keywords: Global Financial Crisis, Predictive Modeling, Economic Indicators, Machine Learning, Time Series Analysis, ARIMA, SARIMAX, LSTM Neural Networks, Logistic Regression Analysis, Behavioral Finance.

**LIST OF FIGURES**

**Figure No. Figure Name** **Pg. No.**

Fig. 3.1 Methodology of Global Financial Crisis Forecasting 11

Fig. 5.1 Screenshot of the Global Financial Crisis dataset 13

Fig. 5.2 Count of Crisis types in each Year from 1801 to 2016 14

Fig. 5.3.1 Statistical Description of the Dataset 16

Fig. 5.3.2 Kernel Density Estimation plot of Year 17

Fig. 5.3.3 Count Plot for Entries in Each Country 18

Fig. 5.3.4 Correlation Matrix of Numerical and Categorical Data 19

Fig. 5.3.5.1 Top 20 Countries For Crisis Event 20

Fig. 5.3.5.2 World Map of Summary of Crisis Event 21

Fig. 5.3.6.1 Trends in Summary of Crisis Event 22

Fig. 5.3.6.2 Cumulative Trends in Crisis Event 23

Fig. 5.4.2.1 ARIMA Forecasting for Banking Crisis 25

Fig. 5.4.2.2 ARIMA Forecasting for Systemic Crisis 25

Fig. 5.4.2.3 ARIMA Forecasting for Currency Crisis 25

Fig. 5.4.2.3 ARIMA Forecasting for Inflation Crisis 26

Fig. 5.4.3.1.1 SARIMAX Forecasting for Banking Crisis 27

Fig. 5.4.3.2.1 SARIMAX Forecasting for Systemic Crisis 27

Fig. 5.4.3.3.1 SARIMAX Forecasting for Currency Crisis 28

Fig. 5.4.3.4.1 SARIMAX Forecasting for Inflation Crisis 29

Fig. 5.4.4.1 Prediction Vs. Actual for Each Crises Type (LSTM) 31

Fig. 5.5.1.1 Performance of Logistic Regression Model for Banking Crisis 33

Fig. 5.5.1.2 Performance of Logistic Regression Model for Systemic Crisis 34

Fig. 5.5.1.3 Performance of Logistic Regression Model for Currency Crisis 35

Fig. 5.5.1.4 Performance of Logistic Regression Model for Inflation Crisis 37

Fig. 5.5.2.1 Autocorrelation Test Results for Banking Crisis 38

Fig. 5.5.2.2 Autocorrelation Test Results for Systemic Crisis 38

Fig. 5.5.2.3 Autocorrelation Test Results for Currency Crisis 38

Fig. 5.5.2.4 Autocorrelation Test Results for Inflation Crisis 39

Fig. 5.5.3.1 LSTM RMSE Results for Banking Crisis 39

Fig. 5.5.3.1 LSTM RMSE Results for Systemic Crisis 39

Fig. 5.5.3.1 LSTM RMSE Results for Currency Crisis 40

Fig. 5.5.3.1 LSTM RMSE Results for Inflation Crisis 40**LIST OF TABLES**

**Table No. Table Name** **Pg. No.**

Table. 5.4.3.1.2 SARIMAX Forecasting for Banking Crisis (2017 to 2028)27

Table. 5.4.3.2.2 SARIMAX Forecasting for Systemic Crisis (2017 to 2028)28

Table. 5.4.3.3.2 SARIMAX Forecasting for Currency Crisis (2017 to 2028)29

Table. 5.4.3.4.2 SARIMAX Forecasting for Inflation Crisis (2017 to 2028)30

Table. 5.4.4.2 Predictions for Each Crises for Years 2017 to 203632

Table. 6 Discussion of Results for Each Crisis Type for Each Model40

**INTRODUCTION**

This research provides a thorough exploration of the factors that shape financial dynamics, particularly emphasizing the examination of historical economic data and market behaviors during past global financial downturns. Leveraging datasets from the Global Financial Crisis Analytics, the study introduces a groundbreaking methodology that combines SARIMAX time series analysis with state-of-the-art machine learning techniques, including logistic regression and LSTM neural networks. This approach is innovative in its ability to identify early signals of financial crises.

Building upon existing financial analysis methods, this research integrates advanced technological advancements to refine these approaches. At its core, the study focuses on analyzing historical data and market behaviors, utilizing both SARIMAX and machine learning models for detailed and nuanced pattern detection. This strategy marks a significant progression in the field of financial forecasting, transitioning from traditional models like ARIMA to more complex and effective algorithms, including machine learning and neural networks.

The research draws from a wide array of sources and methodologies, ranging from traditional econometric models to cutting-edge machine learning applications. It covers a broad spectrum of topics related to financial crises, from banking regulations to corporate risk disclosures, and economic recession forecasting. These comprehensive insights inform the direction of the study, highlighting the evolving trend towards the integration of machine learning with traditional financial analysis methods. This fusion aims to enhance the accuracy and reliability of financial crisis predictions and analysis, representing a notable advancement in the field of economic forecasting.

1. **LITERATURE REVIEW**

1. Current State of Knowledge: The research papers focus on various aspects of financial crises and economic forecasting. These include analyses of financial risk, cash management strategies, bank regulation, corporate risk disclosures, the impact of COVID-19, and the application of various forecasting models like ARIMA, machine learning, and deep learning in financial contexts.

2. Relevant Theories and Models: The studies cover a range of methodologies including regression analysis, econometric methods, machine learning techniques, qualitative and quantitative approaches, and cross-sectional threshold regression. These methodologies are applied in diverse contexts like equity market forecasting, economic recession prediction, and analysis of corporate risk disclosure practices.

3. Gaps in Literature: The document identifies limitations such as the integration of traditional and modern forecasting techniques, limited exploration of diverse economic contexts, reliance on historical data, and over-dependence on AI. Future research directions include expanding traditional time series analysis, exploring new methodologies in risk management, and refining AI models for broader applications. Harvard Business School’s study [1] focuses on traditional financial indicators, missing psychological factors. [2] relies on historical data, potentially overlooking novel economic events. [3] emphasizes macroeconomics, possibly neglecting finer financial details. [4] examine corporate risk disclosures, but face challenges in reporting consistency. [5] and [6] use advanced machine learning and AI, yet may undervalue human insight in economic analysis. [7], [8], [9], and [10] introduce innovative methods, but their narrow focus or reliance on traditional models could limit their general applicability.

4. Project Aims to Fill: The project ambitiously aims to advance the field of financial crisis forecasting by innovatively integrating SARIMAX time series analysis with cutting-edge machine learning techniques, namely logistic regression and LSTM neural networks. This approach is tailored to synthesize quantitative financial data with qualitative behavioral insights, thereby enhancing the accuracy and reliability of predictions. By leveraging logistic regression for its robustness in classification and the LSTM neural network for its proficiency in handling sequential data, the project aspires to offer a comprehensive and adaptable solution. This solution will not only account for typical economic patterns but also adapt to unexpected market shifts, providing a vital tool in the complex landscape of economic forecasting.

1. **OBJECTIVE**

1. To undertake data preprocessing and conduct exploratory analysis on the chosen dataset.

2. To design a model that utilizes the SARIMAX technique, known for its ability to analyze seasonal fluctuations, for the four types of crises (Banking, Systemic, Currency, and Inflation).

3. To apply LSTM networks to explore and model intricate temporal relationships within economic time-series data.

4. To innovate a hybrid methodology that combines the strengths of machine learning with econometric models to improve prediction accuracy.

5. To execute a comparative study to evaluate the performance of the proposed hybrid models against established forecasting techniques.

1. **PROPOSED METHODOLOGY**

****

Fig 3.1 Methodology of Global Financial Crisis Forecasting

* The approach involves:

1. Collecting financial datasets and literature
2. Preprocessing data
3. Applying SARIMAX to detect seasonal trends
4. Identifying patterns in crisis indicators using LSTM neural networks

* The analysis merges time series with machine learning for depth and employs neural networks to interpret SARIMAX outcomes.
* Behavioral finance principles aid in forecasting financial instability, providing actionable insights for policymakers and investors.
* Comparisons will be made against historical averages, traditional ARIMA models, and expert financial analysis to validate the approach's efficacy.

1. **TOOLS AND TECHNIQUES**

The employed tools and techniques are as follows:

1. ARIMA (AutoRegressive Integrated Moving Average): This model is used for forecasting time series data. ARIMA is adept at modeling and leveraging historical data patterns, and it's used in the project to forecast various types of financial crises such as banking, systemic, currency, and inflation crises.
2. SARIMAX (Seasonal AutoRegressive Integrated Moving-Average with eXogenous regressors): Although not explicitly mentioned in the extracted code, SARIMAX is a sophisticated version of ARIMA that includes seasonality and external factors. It's likely used in the project for handling complex, seasonal financial time series data, where external variables are significant.
3. LSTM (Long Short-Term Memory) Neural Networks: These are a type of recurrent neural network ideal for sequential data analysis, like financial time series. In the project, LSTMs analyze historical financial data to predict future trends, leveraging their ability to capture long-term dependencies.
4. Logistic Regression: This method is used for binary classification, predicting the occurrence of financial crises. Logistic regression is effective for its simplicity and clarity in classifying binary outcomes, making it a crucial tool in the project for determining crisis probabilities.

**5 IMPLEMENTATION**

**5.1 ABOUT THE DATASET:**

The dataset 20160923\_global\_crisis\_data.csv contains historical data related to various types of crises in different countries. It is sourced from Harvard Business School’s study, Global Financial Crisis Analytics on Kaggle. It includes columns like 'Banking Crisis', 'Systemic Crisis', 'Gold Standard', 'Exchange Rates', 'Sovereign External Debt', 'GDP Weighted Default', 'Inflation Rates', 'Independence', 'Currency Crises', and 'Inflation Crises', among others. This suggests a comprehensive compilation of financial and economic indicators over time.

No. of Observations: 15,192

* 1. **Case**: A numeric identifier for each record. Each country appears to have a unique case number.
* 2. **CC3**: The three-letter country code, following the standard ISO alpha-3 country codes.
* 3. **Country**: The name of the country to which the data row corresponds.
* 4. **Year**: The year for which the statistics is recorded.
* 5. **Banking Crisis**: Indicates the occurrence of a banking crisis. '0' indicates no crisis, while '1’ indicates a crisis.
* 6. **Banking\_Crisis\_Notes**: Additional notes or explanations about the banking crisis, if any.
* 7. **Systemic Crisis**: Indicates whether a systemic crisis occurred in that year. Similar to 'Banking Crisis', '0' likely means no systemic crisis.
* 8. **Gold Standard**: Indicates whether the country was on the gold standard in that year.
* 9. **exch\_usd**: The exchange rate of the country's currency to the US dollar.
* 10. **exch\_usd\_alt1, exch\_usd\_alt2, exch\_usd\_alt3**: Alternative exchange rate values, representing different sources or calculation methods.
* 11. **Conversion to USD notes/sources**: Notes or sources for the exchange rate data.
* 12. **Domestic Debt In Default**: Indicates whether there was a default on domestic debt.
* 13. **Domestic\_Debt\_ Notes/Sources**: Notes or sources regarding the domestic debt data.
* 14. **SOVEREIGN EXTERNAL DEBT 1 and 2**: Indicates whether there was a default or restructuring of sovereign external debt. Different columns might represent different categorizations or time periods of the debt.
* 15. **Defaults\_External\_Notes**: Notes about external debt defaults.
* 16. **GDP\_Weighted\_default**: A measure of defaults weighted by GDP, possibly indicating the severity or impact of the default relative to the country's economy.
* 17. **Inflation, Annual percentages of average consumer prices**: The annual inflation rate, measured as the percentage change in consumer prices.
* 18. **Independence**: Indicates whether the country was independent in that year.
* 19. **Currency Crises**: Indicates the occurrence of a currency crisis.
* 20. **Inflation Crises**: Indicates the occurrence of an inflation crisis.

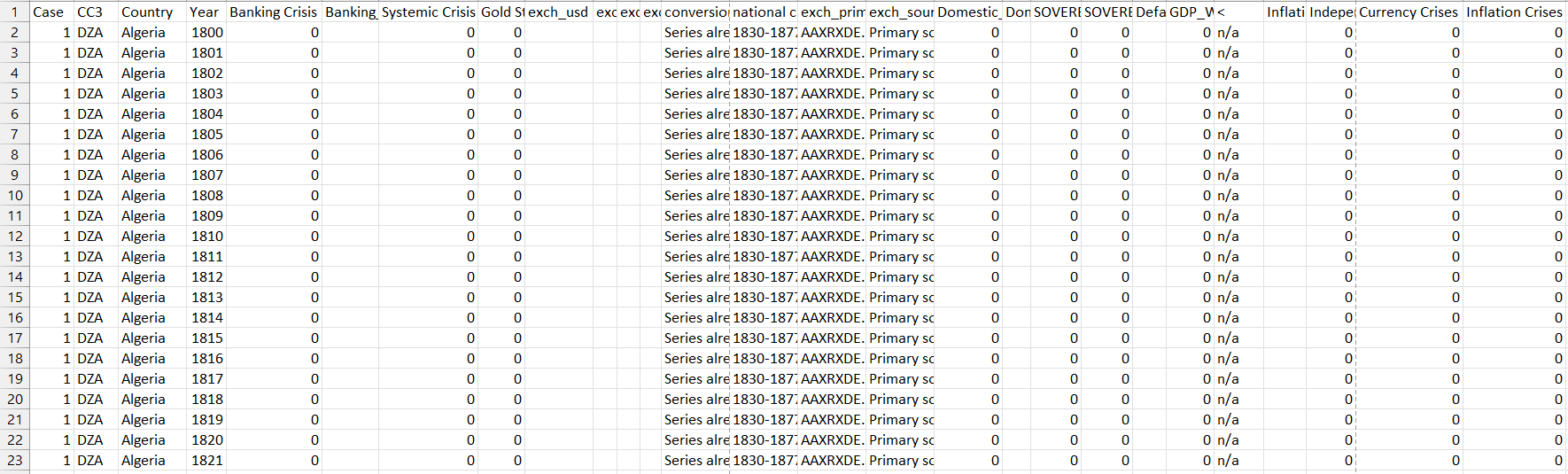


Fig 5.1 Screenshot of the Global Financial Crisis dataset

**5.2 CRISIS DATA PREPROCESSING**

Data preprocessing is a critical phase in the data analysis and machine learning process, involving a series of steps to transform raw data into a format that is more suitable for analysis. This process is essential because the quality and format of data directly influence the performance and accuracy of machine learning models.

The preprocessing includes:

* Data Cleansing: The initial phase of data preprocessing involves minor data cleansing. This step includes filtering the data to include years post-1800 and handling NaN (not a number) values. It suggests that the focus is on more recent and complete data. Additionally, some columns, particularly those indicating different types of crises, are converted to integer data types. This conversion ensures that the data can be effectively used in numerical models.
* Data Aggregation: The process of aggregating the crisis data involves summarizing or restructuring the data to make it more suitable for time-series analysis. Aggregation might include grouping data by year or other relevant time periods, which is a common practice in analyzing time-series data to observe trends and patterns over time.

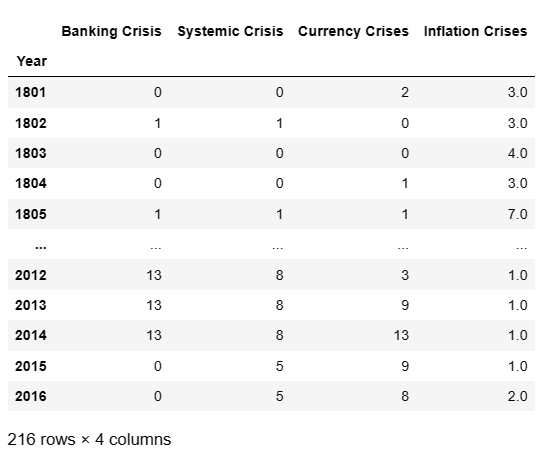
****

Fig 5.2 Count of Crisis types in each Year from 1801 to 2016

The figure appears to be a screenshot of a DataFrame displaying aggregated data related to financial crises over a span of years. The DataFrame has 216 rows and 4 columns, each corresponding to a different type of crisis:

* + Banking Crisis: Shows the count of banking crises that occurred in each year.
  + Systemic Crisis: Indicates the number of systemic crises for each year.
  + Currency Crises: Reflects the number of currency crises that were recorded annually.
  + Inflation Crises: The number here probably represents either the count of inflation crises per year or the inflation rate, although without context, it's not clear which one it is.

The years run from 1801 to 2016. The data shown in the figure is likely aggregated in the sense that each cell in the DataFrame represents the sum or count of crises events that occurred in that year across a certain number of countries or regions.

Here are a few observations based on the partial data visible in the figure:

* + In the early 1800s, there were a few banking and currency crises and varying levels of inflation crises.
  + There is a visible increase in the frequency and/or number of crises in the 2010s, especially banking, systemic, and currency crises.
  + The number of inflation crises appears to be lower in the 2010s compared to a single year shown from the 1800s.
* Data Transformation: Certain columns, especially those indicating different crises, are converted to numeric formats and missing values are handled. This transformation is crucial for the statistical and machine learning models to process the data accurately.
* Time Series Preparation: For time-series models like ARIMA and SARIMAX, the notebook shows steps to convert the 'Year' column to a datetime format and set it as the index. This step is vital for any time-series analysis as it aligns the data along a time dimension, allowing for effective modeling of temporal dynamics.
* Column Renaming and Selection: There is an indication of renaming or correctly specifying column names based on the actual names in the dataset. This step helps in avoiding confusion and errors during data manipulation and analysis.

**5.3 EXPLORATORY DATA ANALYSIS**

**5.3.1. Data Description**

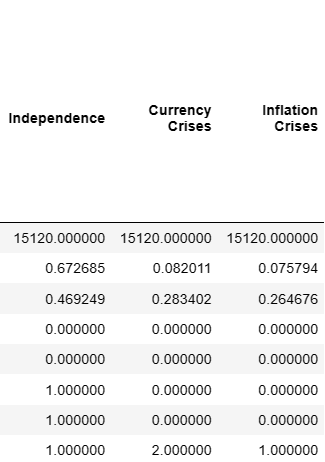
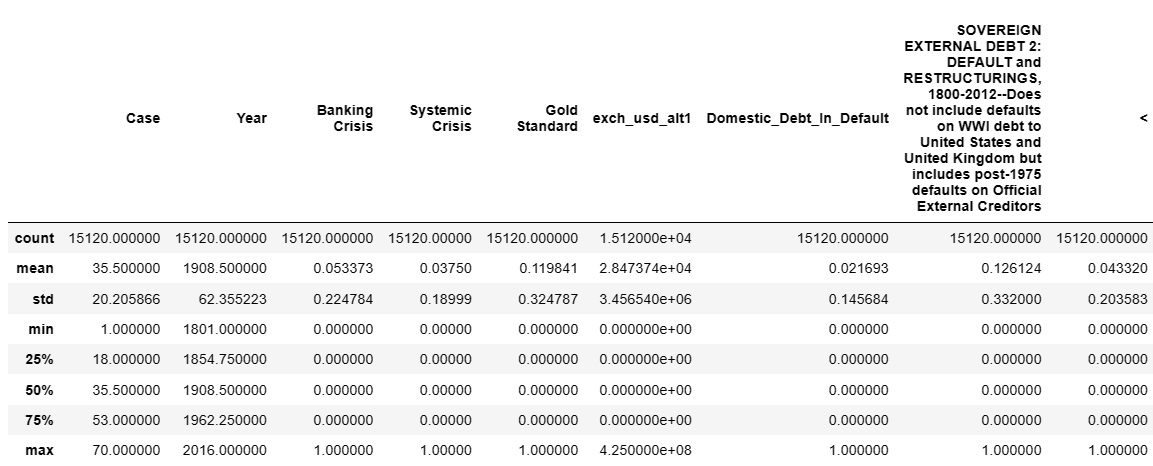
****

Fig 5.3.1 Statistical Description of the Dataset

The figure you've provided seems to be an output of a descriptive statistical analysis of a dataset concerned with various types of financial crises and related indicators. Here's an interpretation of the statistical summary provided:

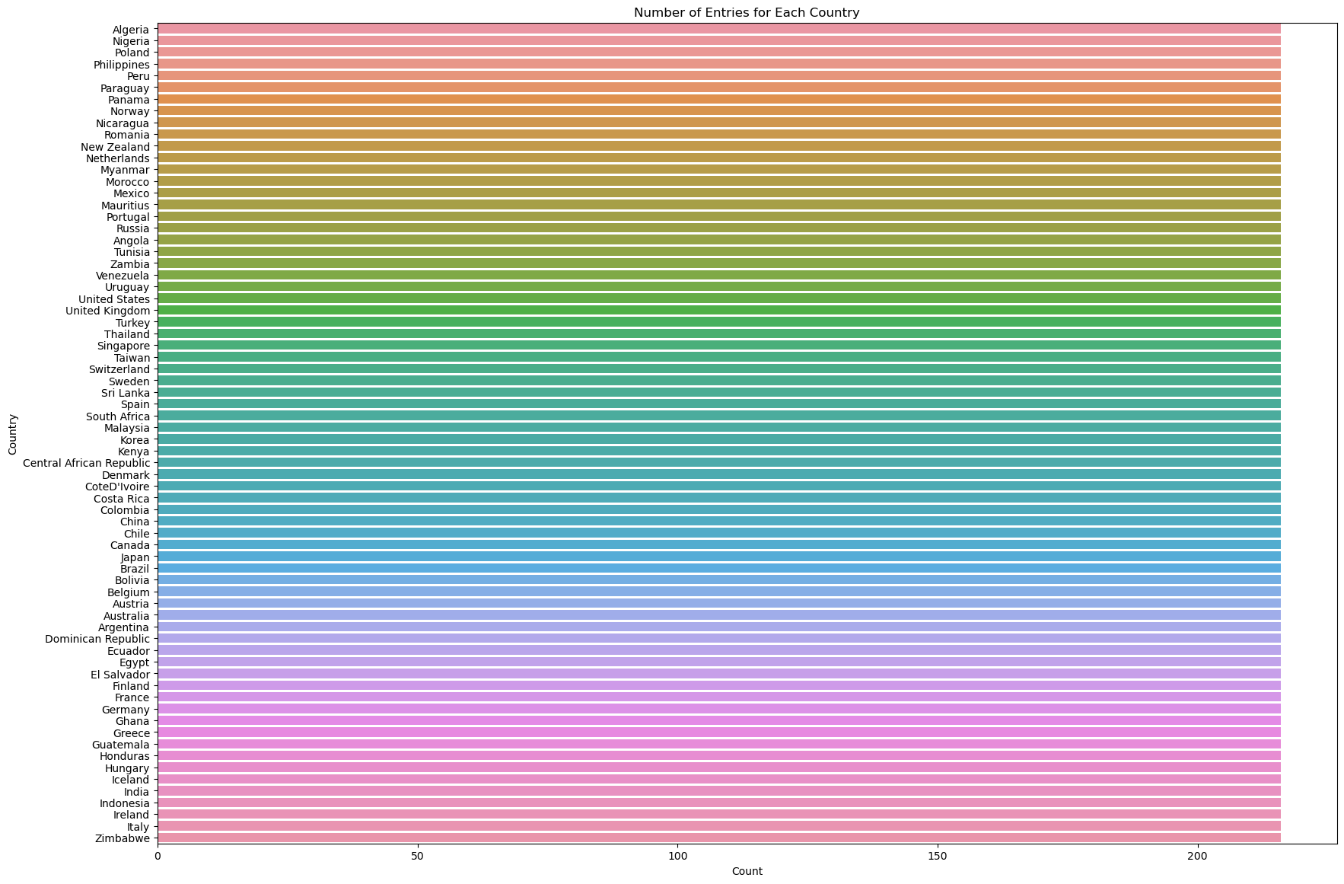
* **Count:** The number of observations (rows) in the dataset for each variable. All variables have 15120 observations except for 'Gold Standard', 'exch\_usd', 'Domestic\_Debt\_In\_Default', 'SOVEREIGN EXTERNAL DEBT 1: 1820-2016', 'Independence', 'Currency Crises', and 'Inflation Crises', which have 15120 non-null entries indicating no missing values.
* **Mean:** The average value for each variable. For example, the average 'Year' in the dataset is approximately 1908, suggesting that the data spans from the early 19th to the early 21st century.
* **Std (Standard Deviation):** Measures the amount of variation or dispersion of a set of values. A low standard deviation indicates that the values tend to be close to the mean of the set, while a high standard deviation indicates that the values are spread out over a wider range.
* **Min (Minimum):** The smallest value in each column. Notably, the minimum value for 'Year' is 1801, and for 'Gold Standard', 'Systemic Crisis', and 'Currency Crises', it is 0, suggesting that there are years with no such events.
* **25% (25th percentile):** Also known as the first quartile, this is the value below which 25% of the data fall. For 'Year', the first quartile is around 1854, indicating that 25% of the observations are from years before 1854.
* **50% (50th percentile or Median):** The median value of each dataset. For 'Year', the median is 1908, meaning that half of the observations are from years before 1908 and half from after.
* **75% (75th percentile):** The value below which 75% of the data fall. For 'Year', this is around 1962, indicating that 75% of the observations are from years before 1962.
* **Max (Maximum):** The largest value in each column. The maximum 'Year' is 2016, and there are columns like 'Systemic Crisis', 'Currency Crises', and 'Inflation Crises' that have a maximum value of 1, indicating that these are likely binary indicators (0 or 1) signifying the absence or presence of a crisis in a given year.
* The note in the figure provides additional context indicating that 'SOVEREIGN EXTERNAL DEBT 1' refers to the first of presumably several measures of sovereign external debt from 1820 to 2016, excluding instances where the United Kingdom defaults on official external creditors before 1850 and the United States and United Kingdom defaults on official external creditors before 1945.
* The dataset appears to be a comprehensive historical record of various financial indicators and crises, possibly used to study the prevalence, causes, and consequences of financial instability over the past two centuries.

**5.3.2. KDE Plot for Distribution of Years in the Dataset**

 Fig 5.3.2 Kernel Density Estimation plot of Year

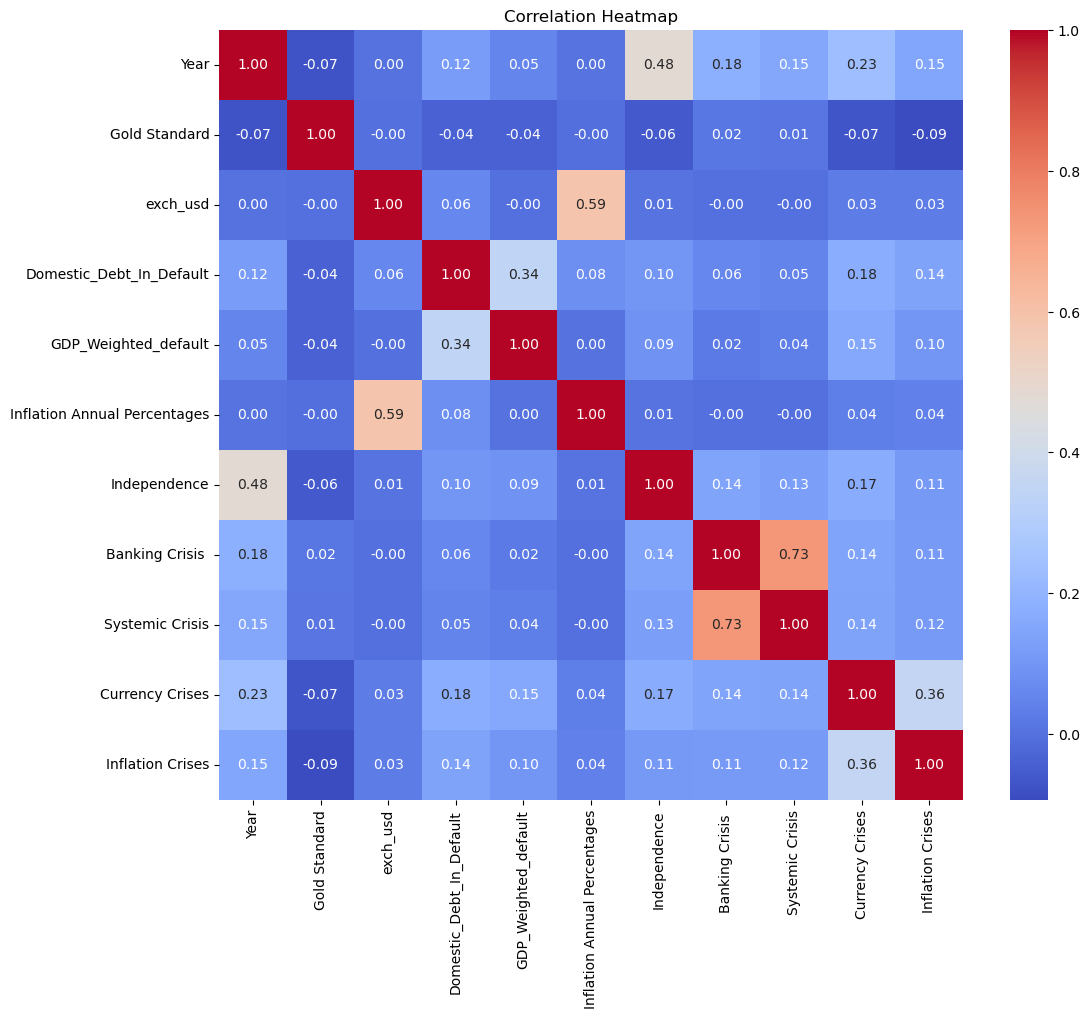
* The x-axis represents years, spanning from the early 1800s to the early 2000s.
* The y-axis represents the frequency of records for each year within the dataset.
* Each bar in the histogram corresponds to the count of records for a specific year or range of years.
* The data appears to be evenly distributed across the years since the height of the bars is relatively uniform. This suggests that the dataset has a similar number of records for each year.
* There's a line that overlays the bars, likely representing a kernel density estimation (KDE). This line is a smooth representation of the distribution and shows that the dataset is quite uniform across the years without significant outliers or gaps.
* The uniformity of the distribution suggests that the dataset could be a balanced panel data set with the same number of observations or entries for each year. This kind of distribution is typical when dealing with structured historical data where each time period is equally represented.

**5.3.3. Count Plot for Number of Entries in Each Country**

Fig 5.3.3 Count Plot for Entries in Each Country

* The y-axis lists various countries, and the x-axis represents the count of entries for each of those countries.
* Each bar's length corresponds to the number of entries in the dataset for the country it represents.
* The bars are color-coded, likely for visual distinction only, as there seems to be no legend explaining if the colors correspond to different categories or groups.
* It appears that every country has a similar number of entries, close to 200, which suggests that the dataset may be well-balanced with respect to the number of records per country.
* Such a uniform distribution across countries indicates that the dataset may have been constructed to include an equal or nearly equal number of observations from each country, which is common in comparative studies or analyses that aim to prevent country-based sampling biases.

**5.3.4.** **Correlation Matrix of Numerical and Categorical Data**

Fig 5.3.4 Correlation Matrix of Numerical and Categorical Data

The positive correlations in the above heatmap are as follows:

* **Banking Crisis and Systemic Crisis (0.73):** A strong positive correlation, indicating that these types of crises often occur together.
* **Domestic\_Debt\_In\_Default and GDP\_Weighted\_default (0.34):** A moderate positive correlation, suggesting a relationship between GDP-weighted defaults and domestic debt defaults.
* **Currency Crises and Inflation Crises (0.36):** A moderate positive correlation, implying that currency crises might be linked with inflation crises.
* **Inflation (annual percentages of average consumer prices) and Exchange Rate (exch\_usd) (0.59):** A strong positive correlation, indicating that higher inflation rates are often associated with a weaker currency against the US dollar.

**5.3.5. Global Crisis Summary**

**5.3.5.1 Top 20 Countries for each type of Crisis**

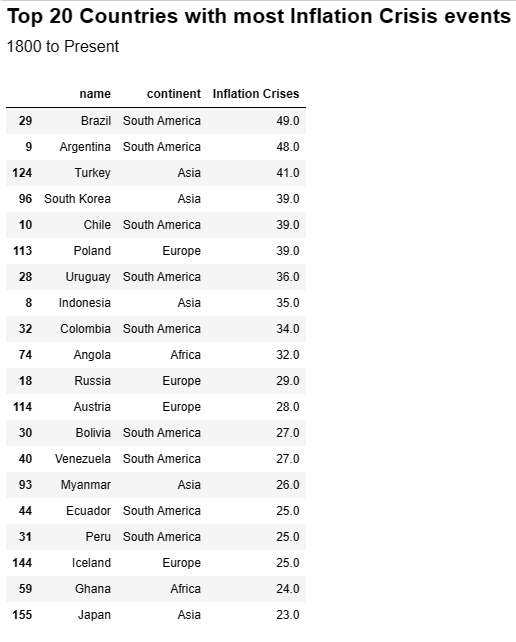
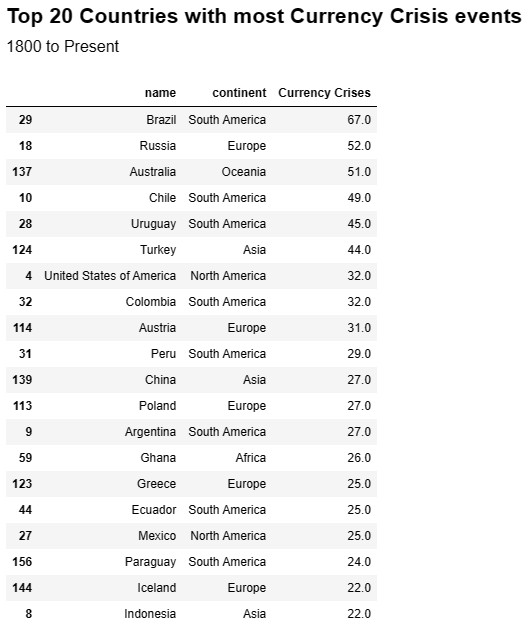
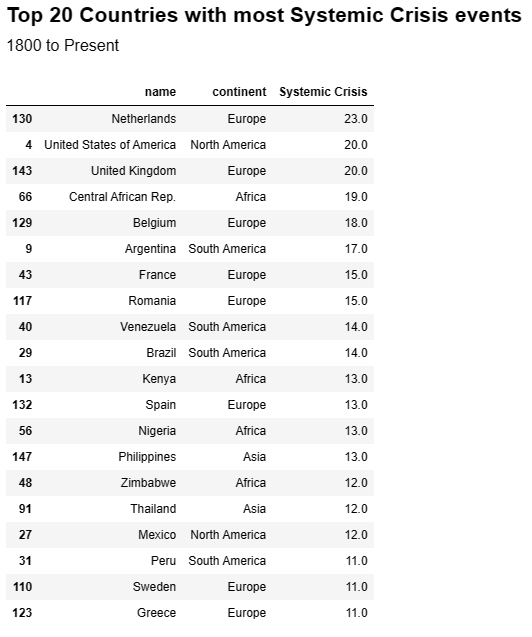
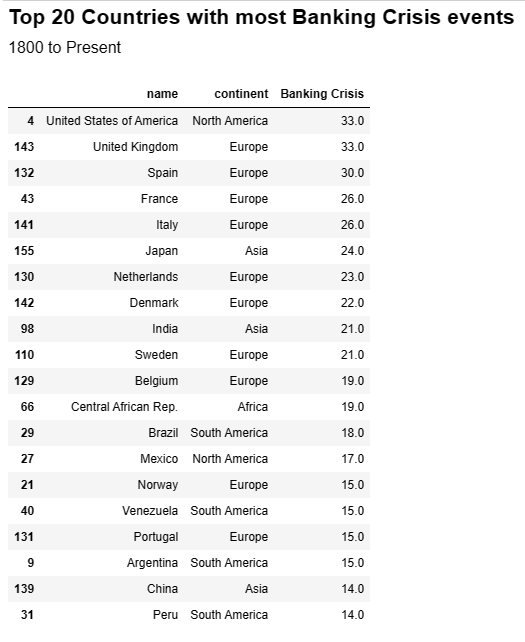
****

Fig 5.3.5.1 Top 20 Countries For Crisis Event

The figures display tables listing the top 20 countries with the most banking, systemic, currency, and inflation crisis events from 1800 to the present, detailing the frequency of each crisis type by country and continent.

**5.3.5.2. World Map showing a summary for each type of crisis from 1800-2016**

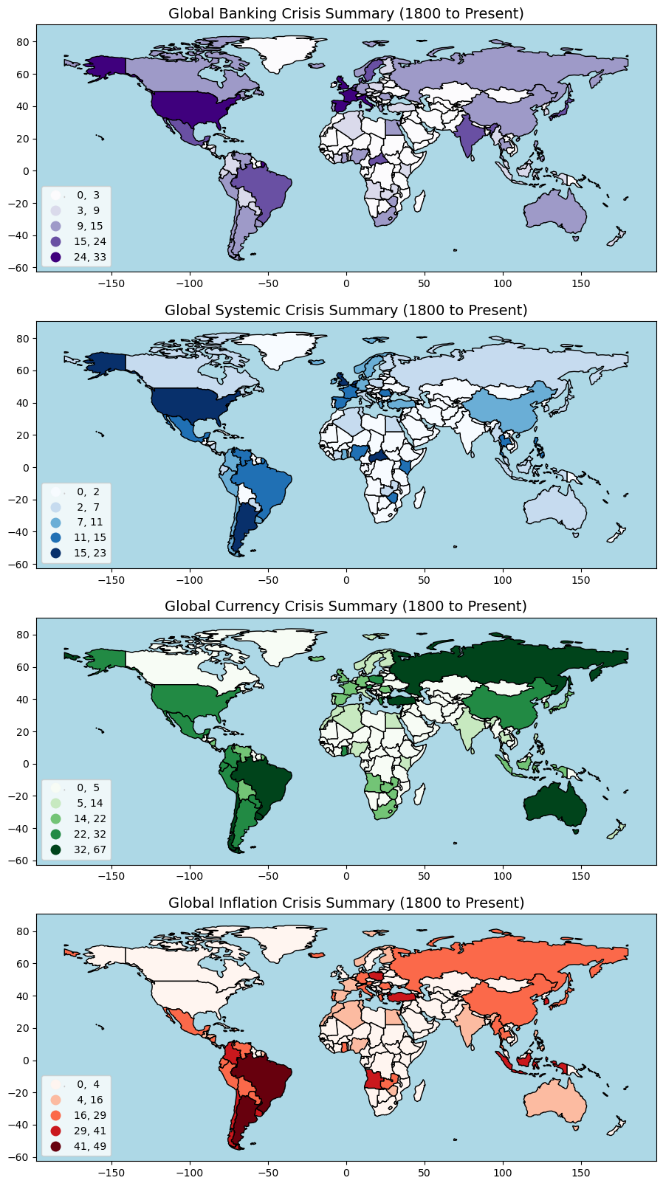
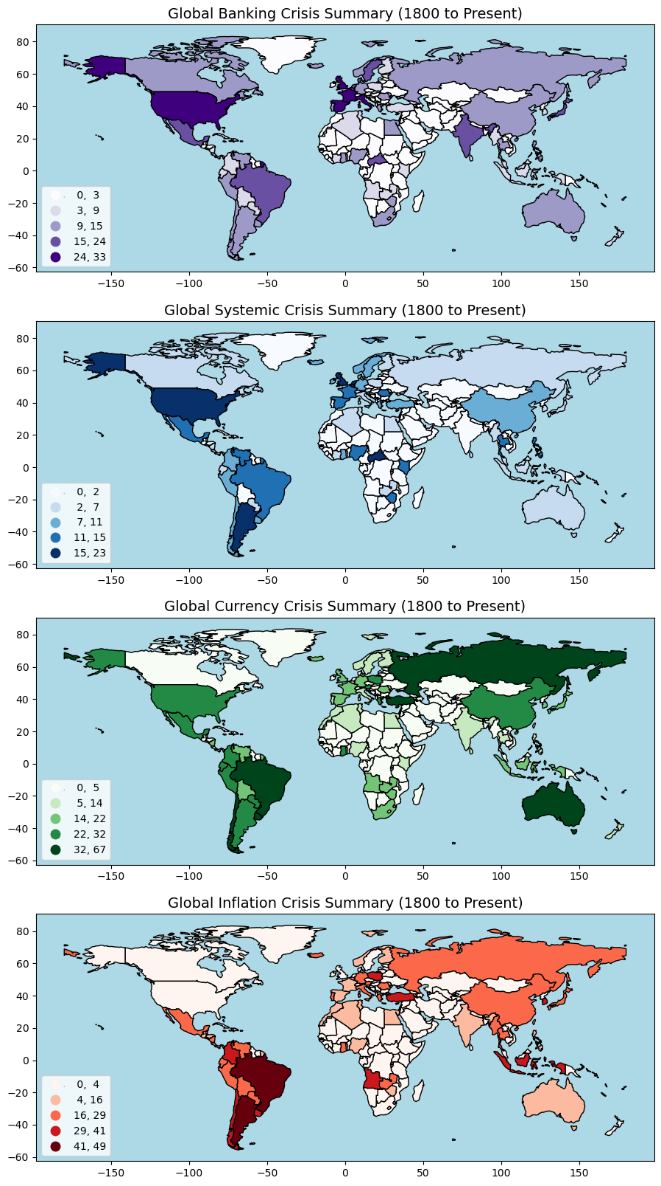


Fig 5.3.5.2 World Map of Summary of Crisis Event

The maps indicate the countries affected by each type of financial crisis:

* **Banking Crisis:** Map depicting the number of banking crises per country, with the US and Europe showing the highest frequency.
* **Systemic Crisis:** Illustrates systemic crises, with Europe and the US most affected.
* **Currency Crisis:** Highlights currency crises, predominantly in South America and Europe.
* **Inflation Crisis:** Visualizes inflation crises, with South America, particularly Brazil and Argentina, most impacted.

**5.3.6. Global Crisis By Year**

**5.3.6.1. Crisis Summary**

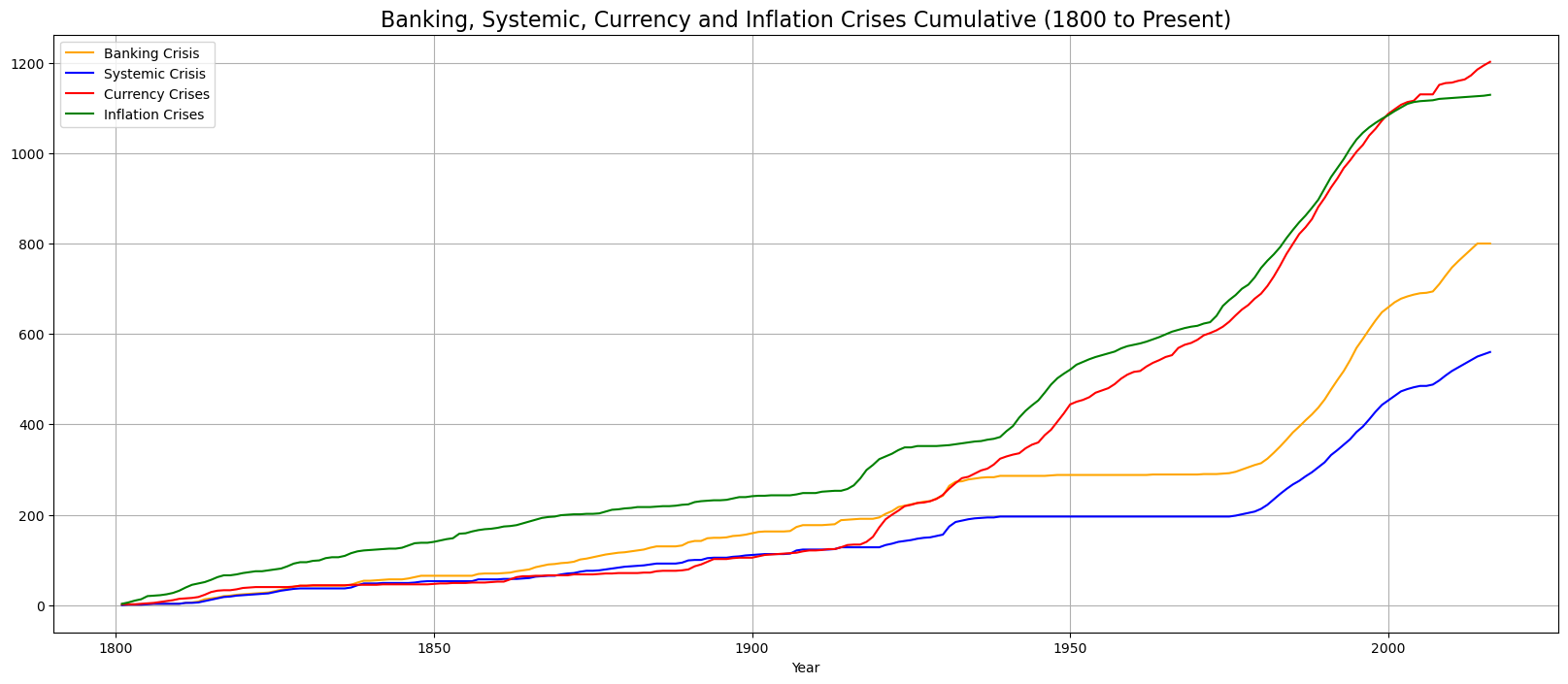


Fig 5.3.6.1 Trends in Summary of Crisis Event

From the crisis summary analyses it can be determined that:

* Banking Crisis: Irregular trends with notable surges in crises around the 1930s and 2000s.
* Systemic Crisis: Sporadic occurrences with sharp increases, suggesting impactful crises at certain intervals.
* Currency Crisis: Varied instances with a pronounced rise towards the late 1900s.
* Inflation Crisis: High activity in the mid-1900s, tapering off in recent decades.

**5.3.6.2. Crisis Summary - Cumulative**

 Fig 5.3.6.2 Cumulative Trends in Crisis Event

The graph indicates that all types of crises have become more common over time, with inflation crises showing the most pronounced increase. This could reflect various economic trends, policy changes, and evolving financial systems globally. The cumulative nature of the graph underscores the continuing impact and rising frequency of these economic events over time.

**5.4 MODELS**

**5.4.1. Logistic Regression:**

* The Logistic Regression model is utilized for predicting the occurrence of banking crises, systemic crises, currency crises, and inflation crises. This model is ideal for binary classification tasks, but can be adapted for multiclass classification, making it suitable for this application.
* The logistic regression model uses various economic and financial features, such as exchange rates, inflation rates, and historical crisis data, as predictors. The preprocessing steps include converting target columns to numeric formats, handling NaN values, and splitting the data into training and testing sets.
* The model is trained and evaluated for its accuracy and precision. While the model shows good overall accuracy, the classification report suggests certain limitations, particularly in predicting less frequent crisis events.
* The effectiveness of Logistic Regression in this context is primarily determined by the feature selection and the inherent characteristics of the dataset, such as imbalance in the representation of different classes.

**5.4.1.1. Logistic Regression Model for Banking Crisis**

Banking Crisis Model: Exhibits high overall accuracy at 94.78%. It's highly precise (0.95) in predicting the absence of a crisis and always correct in these predictions (recall 1.00). However, it completely misses the actual crisis predictions, with zero precision and recall for class 1, indicating a model that predicts 'no crisis' too frequently.

**5.4.1.2. Logistic Regression Model for Systemic Crisis**

Systemic Crisis Model: This model is very accurate at 97.81%, with outstanding precision (0.99) and recall (0.99) for predicting non-crisis situations. It also performs reasonably well in identifying crises with decent precision (0.69) and good recall (0.80), indicating a better balance in predicting both classes compared to the Banking Crisis Model.

**5.4.1.3. Logistic Regression Model for Currency Crisis**

Currency Crisis Model: With an accuracy of 92.16%, this model shows excellent precision (0.92) and recall (1.00) for non-crisis predictions but struggles with crisis detection, showing only moderate precision (0.53) and very poor recall (0.04) for class 1. It fails entirely in predicting the rare class 2 events (precision and recall are 0.00).

**5.4.1.4. Logistic Regression Model for Inflation Crisis**

Inflation Crisis Model: The most accurate at 99.26%, it almost perfectly identifies non-crisis situations (precision 1.00, recall 1.00) and demonstrates high precision (0.96) and recall (0.94) for crisis events, indicating a highly reliable model for both predicting and excluding inflation crises.

**5.4.2. ARIMA:**

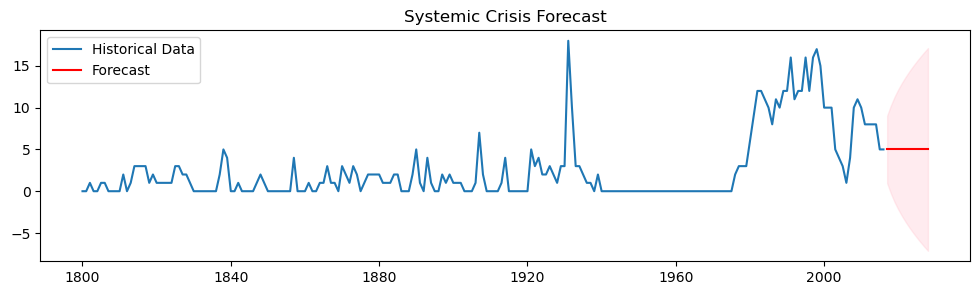
* The ARIMA (AutoRegressive Integrated Moving Average) model is a popular time-series forecasting tool used to predict future trends based on past data. In the notebook, ARIMA models are likely applied to forecast the likelihood of various crises.
* ARIMA models are particularly adept at capturing the temporal dynamics in time-series data, making them suitable for predicting events like financial crises, which are often influenced by past trends.
* The specific implementation details, such as the ARIMA model order (p, d, q), and the results of the forecasts (like prediction accuracy or forecast intervals), are crucial for understanding the model's performance. However, these details are not explicitly provided in the extracted information.
* The accuracy and reliability of ARIMA predictions heavily depend on the correct specification of the model parameters and the inherent nature of the time-series data.

**5.4.2.1. ARIMA Forecasting for Banking Crisis**

 Fig 5.4.2.1 ARIMA Forecasting for Banking Crisis

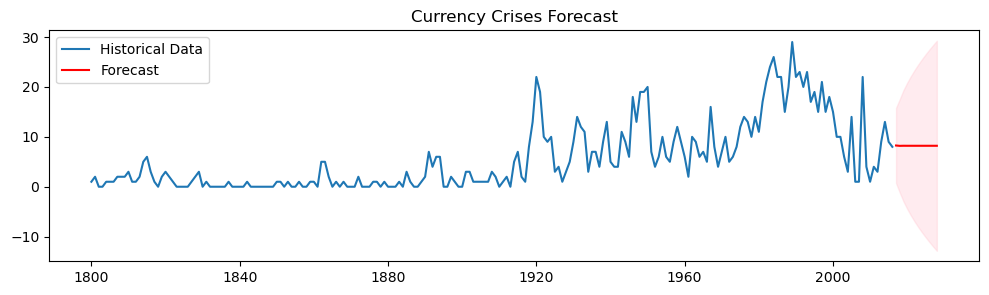
Banking Crisis Forecast: The model's forecast, showing a downturn, predicts fewer banking crises ahead. The large confidence interval (red shaded area) suggests significant uncertainty in this prediction, indicating potential volatility or lack of data confidence.

**5.4.2.2. ARIMA Forecasting for Systemic Crisis**

 Fig 5.4.2.2 ARIMA Forecasting for Systemic Crisis

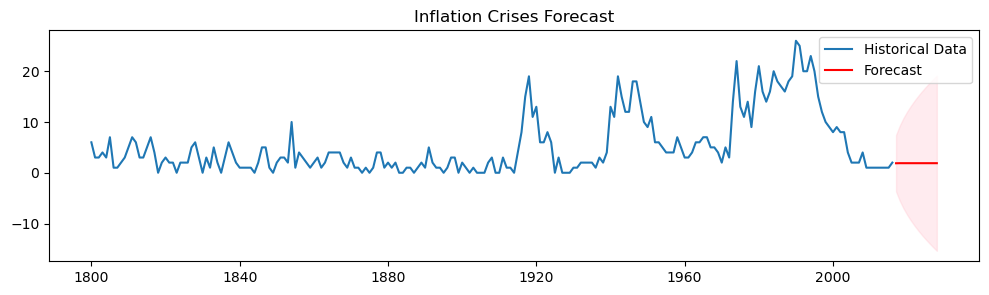
Systemic Crisis Forecast: A minor increase is predicted in the near term for systemic crises, followed by a plateau. The confidence bounds are relatively tight, suggesting the model has a moderate degree of certainty about this forecast.

**5.4.2.3. ARIMA Forecasting for Currency Crisis**

 Fig 5.4.2.3 ARIMA Forecasting for Currency Crisis

Currency Crisis Forecast: The model anticipates a sharp decline in currency crises, with the forecast returning to a lower frequency similar to pre-1900 levels. The confidence interval is broad, denoting less certainty in the longer-term outlook.

**5.4.2.4. ARIMA Forecasting for Inflation Crisis**

 Fig 5.4.2.4 ARIMA Forecasting for Inflation Crisis

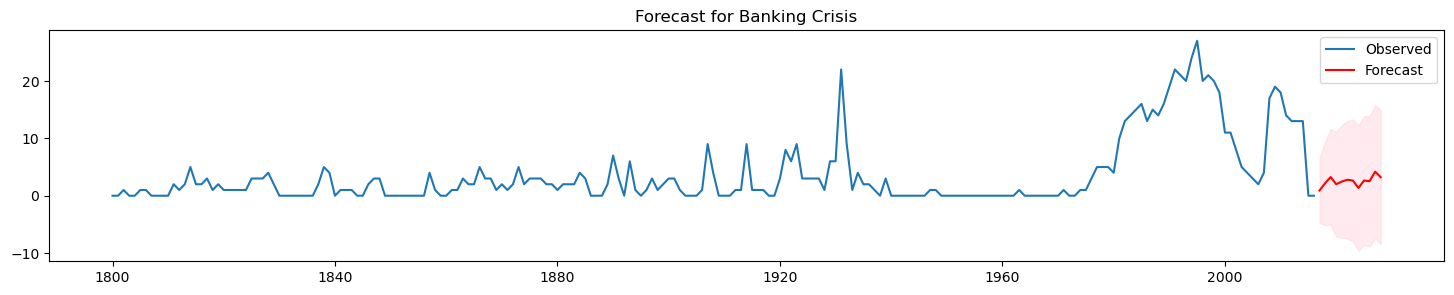
Inflation Crisis Forecast: A downward trend in inflation crises is projected, suggesting an expectation of economic stabilization in this area. The confidence interval is narrower, indicating more confidence in this forecast compared to currency crises, but still acknowledging some level of uncertainty.

Each forecast is based on historical data trends and ARIMA model calculations, considering past crisis occurrences to estimate future events within a statistical confidence range.

**5.4.3. SARIMAX:**

* SARIMAX (Seasonal AutoRegressive Integrated Moving-Average with eXogenous factors) extends the ARIMA model by incorporating seasonality and external factors. This makes SARIMAX particularly powerful for datasets where seasonal patterns and external influences play a significant role.
* In the context of predicting financial crises, SARIMAX can leverage additional variables that might influence the likelihood of a crisis, such as global economic indicators or geopolitical events.
* The implementation details, like the choice of exogenous variables and the seasonal components of the model, are key to its effectiveness. The results section would typically include predictions and possibly an assessment of the model's forecasting accuracy, but these specifics are not provided in the extracted data.
* SARIMAX's effectiveness in forecasting crisis events would largely hinge on the accurate modeling of seasonality and the appropriate inclusion of relevant external variables.

**5.4.3.1. SARIMAX Forecasting for Banking Crisis**

 Fig 5.4.3.1.1 SARIMAX Forecasting for Banking Crisis

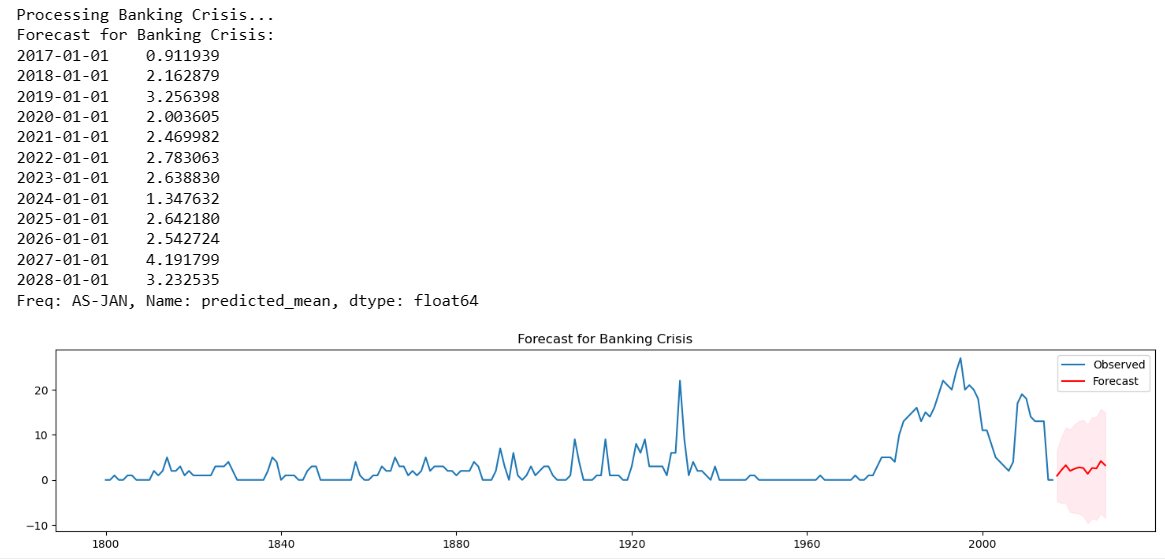
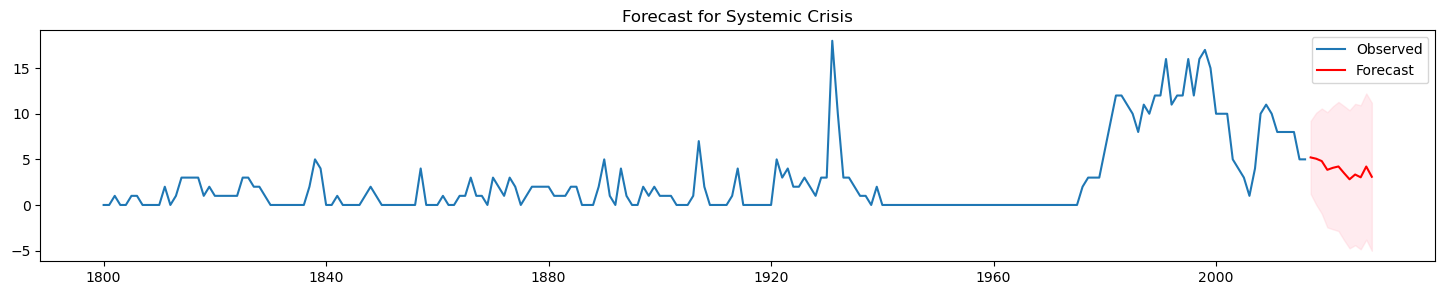


Table 5.4.3.1.2 SARIMAX Forecasting for Banking Crisis (2017 to 2028)

The SARIMAX model output indicates variations in the forecast of banking crises from 2017 to 2028. Initially, there's a moderate prediction of crises peaking in 2019. A slight decrease is expected in 2020, followed by a mild increase in 2021. The model then forecasts a sharp decline in 2024, suggesting fewer crises. This trend reverses with a gradual increase towards 2028. The red line on the graph represents these forecasts, and the pink shaded area indicates the model's confidence intervals, reflecting uncertainty in the predictions.

**5.4.3.2. SARIMAX Forecasting for Systemic Crisis**

 Fig 5.4.3.2.1 SARIMAX Forecasting for Systemic Crisis

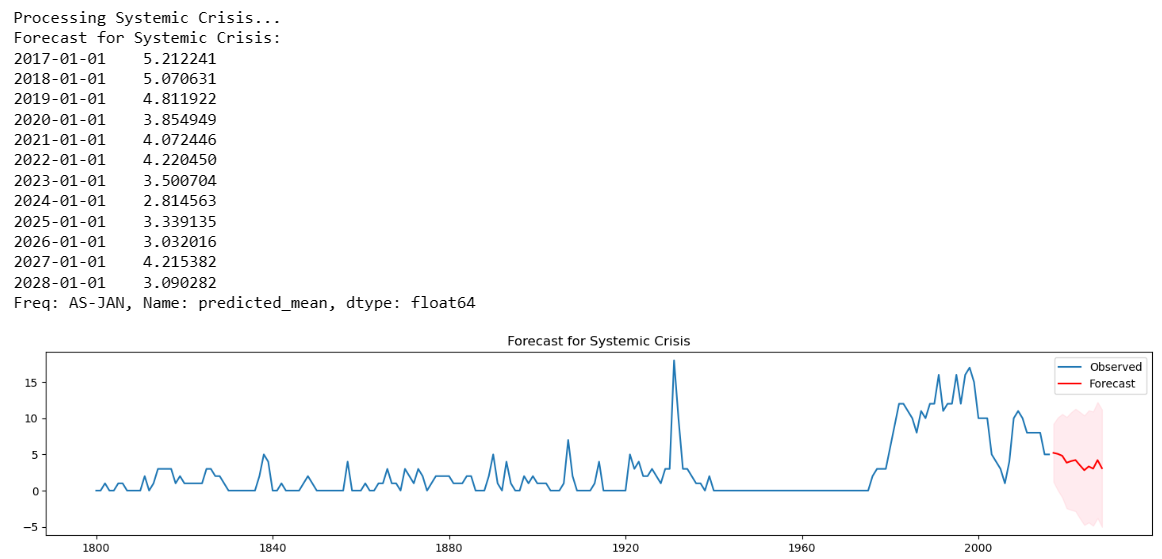
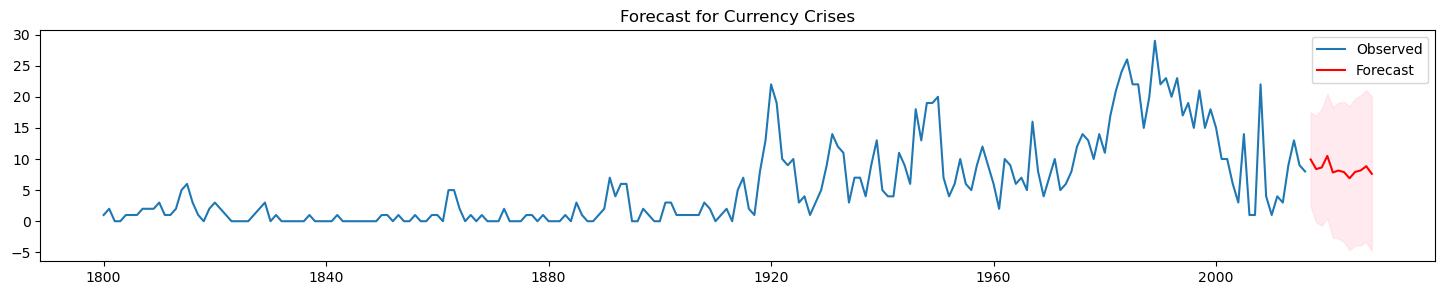


Table 5.4.3.2.2 SARIMAX Forecasting for Systemic Crisis (2017 to 2028)

The SARIMAX model forecasts a decline in systemic crises from 2017 to 2024, suggesting a period of relative stability in the financial system. The predicted value decreases from around 5.21 crises in 2017 to 2.81 crises in 2024. After 2024, the forecast indicates a slight uptick, with values hovering around 3 to 4 crises, reaching 3.09 by 2028. The confidence interval in the forecast graph widens over time, which implies increasing uncertainty in the model's predictions as it projects further into the future. The observed historical data, represented by the blue line, shows peaks and troughs, while the forecast, shown in red, suggests smoothing of these extremes into a downward trend in systemic crises.

**5.4.3.3. SARIMAX Forecasting for Currency Crisis**

 Fig 5.4.3.3.1 SARIMAX Forecasting for Currency Crisis

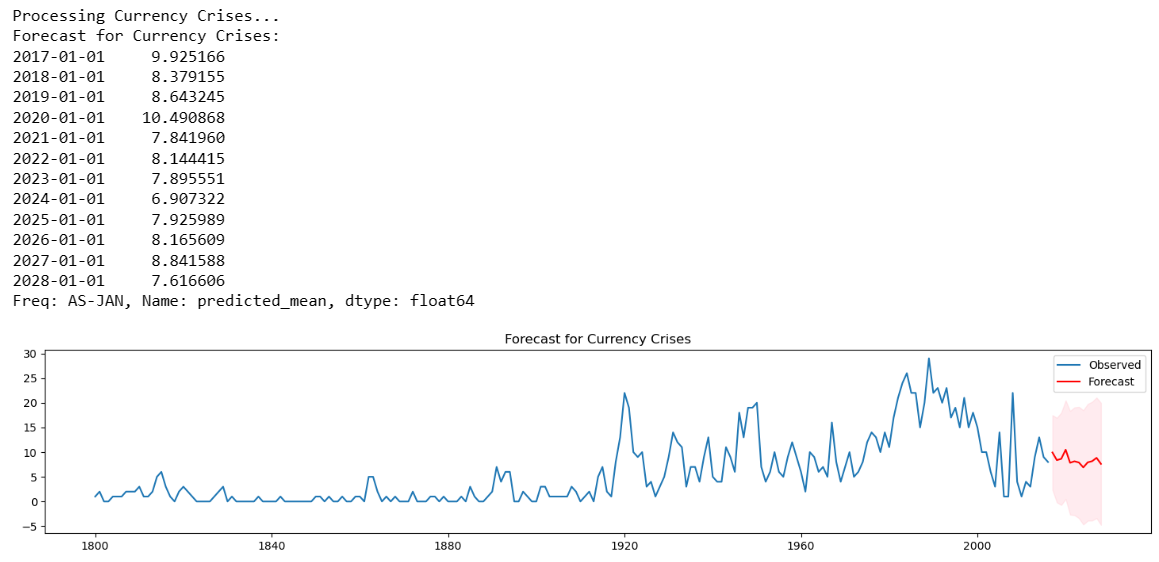
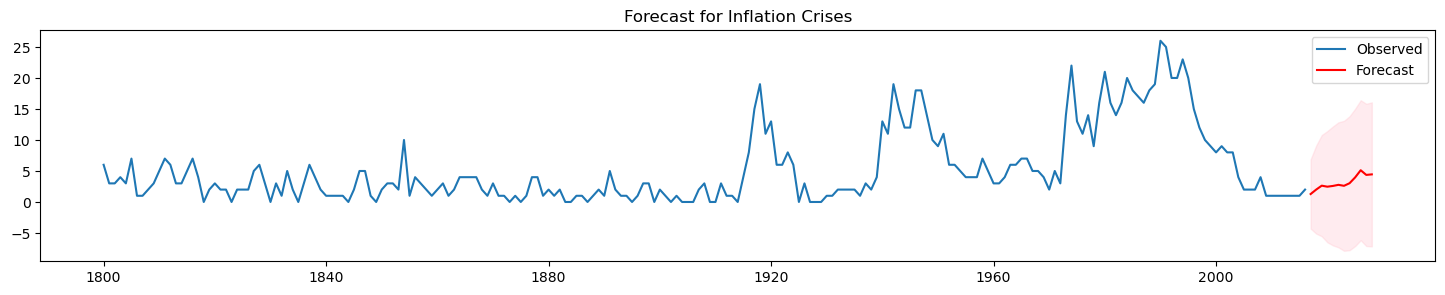


Table 5.4.3.3.2 SARIMAX Forecasting for Currency Crisis (2017 to 2028)

The SARIMAX model for currency crises shows an initial increase in frequency to a peak around 2020 with approximately 10.49 events forecasted. The model then predicts a decrease, stabilizing to a range of 7 to 9 events per year towards 2028. The forecasted decline to 7.62 by 2028 suggests a reduction in the occurrence of currency crises over the long term. The forecast graph, with the observed data in blue and the forecast in red, displays a high level of fluctuation in historical data, while the forecast suggests a smoother trend with a narrowing confidence interval over time, indicating a relatively more certain outlook towards the end of the forecast period.

**5.4.3.1. SARIMAX Forecasting for Inflation Crisis**

 Fig 5.4.3.4.1 SARIMAX Forecasting for Inflation Crisis

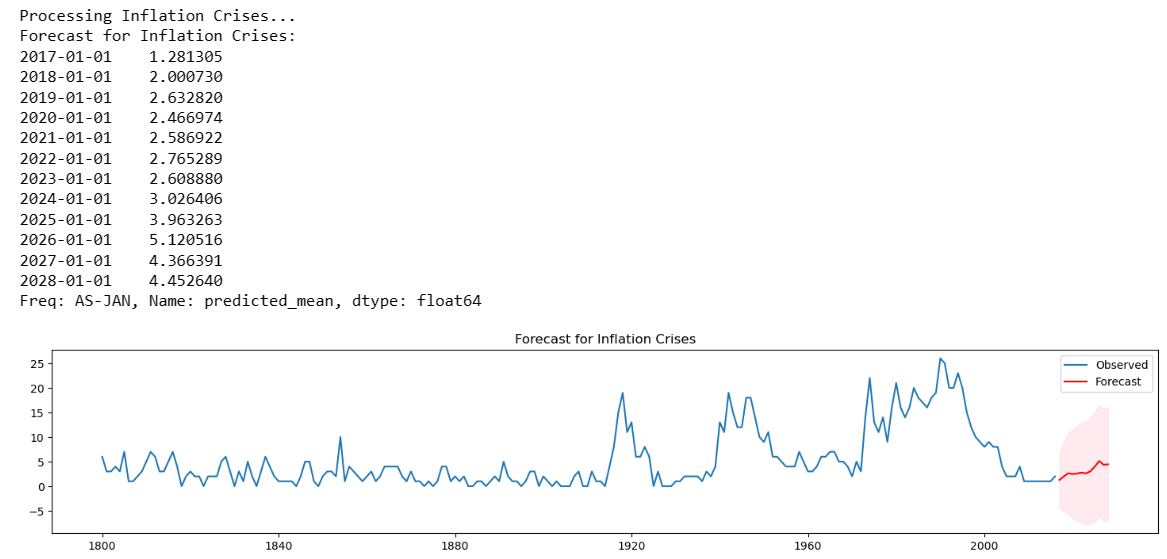
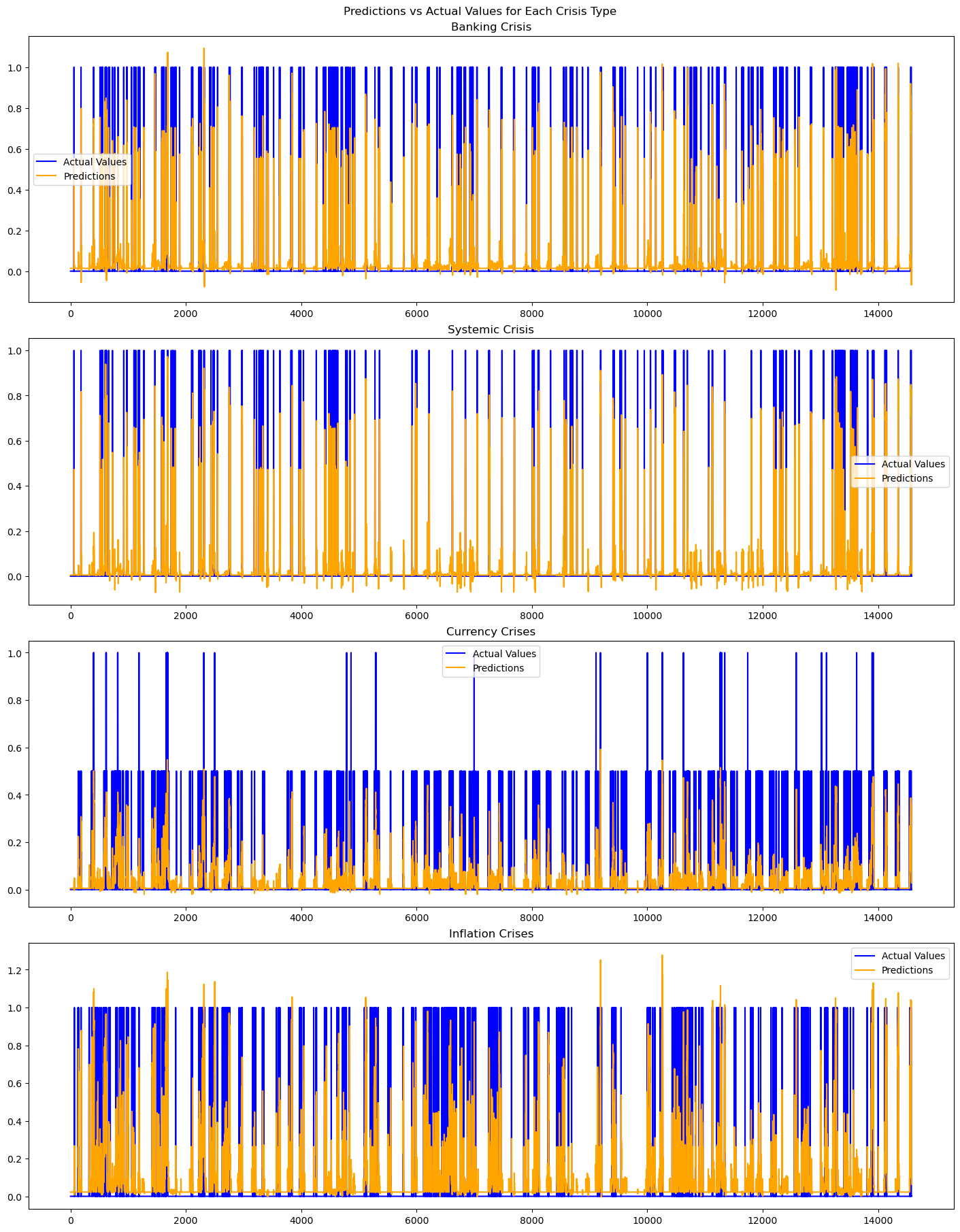


Table 5.4.3.4.2 SARIMAX Forecasting for Inflation Crisis (2017 to 2028)

The SARIMAX forecast for inflation crises indicates an overall increasing trend from 2017 to 2028. The forecast starts at a relatively low point in 2017 with about 1.28 crises and shows a general upward trajectory reaching about 4.45 crises by 2028. The observed historical data presents a volatile pattern with sharp peaks and troughs, while the forecast suggests a gradual increase with less volatility. The confidence interval, depicted by the shaded area, widens over time, implying greater uncertainty as the forecast extends further into the future.

**5.4.4. LSTM:**

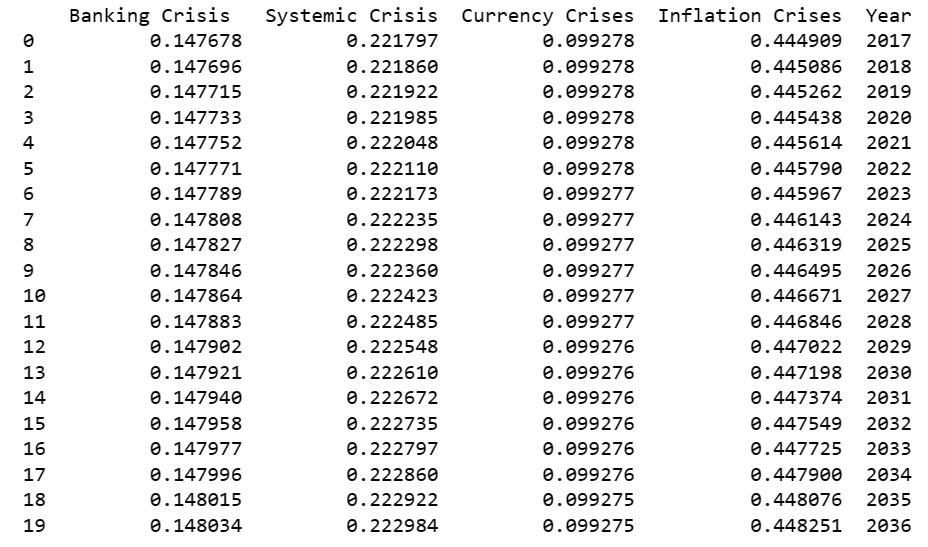
* LSTM (Long Short-Term Memory) networks are a type of recurrent neural network (RNN) capable of learning long-term dependencies in sequence data. They are particularly well-suited for time-series forecasting tasks like predicting financial crises.
* The LSTM model in the notebook is likely employed to forecast the probability of different types of crises over time. The strength of LSTM lies in its ability to remember information over long periods, which is crucial in understanding the cyclical and long-term trends in financial data.
* The notebook's LSTM implementation likely involves multiple layers and neurons, and the model is trained over several epochs. The results indicate the model's performance using metrics like Root Mean Square Error (RMSE), which provides an idea of the model's prediction accuracy.
* The LSTM's performance is contingent upon various factors, including the architecture of the network, the length of the input sequences, and the nature of the time-series data. The effectiveness of the LSTM in this scenario is a function of its ability to capture complex, long-term patterns in the data.

 Fig 5.4.4.1 Prediction Vs. Actual for Each Crises Type (LSTM)

The charts present a comparison of predicted versus actual occurrences of different types of financial crises:

* Banking Crisis: The predictions for banking crises show frequent points where the model accurately predicts the absence of a crisis but has occasional false negatives (missed crisis events) and false positives (predicted crises that did not occur).
* Systemic Crisis: The model for systemic crises appears more accurate, with a higher incidence of correct predictions. However, there are instances of false positives where the model predicts a crisis that does not materialize.
* Currency Crises: This model has the most noticeable discrepancies, with numerous false positives and false negatives, suggesting that predicting currency crises may be more complex due to the influence of various unpredictable economic factors.
* Inflation Crises: Predictions for inflation crises show a pattern that seems to overestimate the frequency of crises. Although the model captures the trend, it does not align closely with the actual data, indicating the model might be sensitive to certain inflationary signals.

The vertical axis represents the binary outcome (0 or 1) indicating the absence or presence of a crisis, while the horizontal axis represents the timeline of the data points. The blue bars (actual values) and the orange lines (predictions) should ideally overlap for accurate predictions. The gaps between them indicate the model's predictive errors.

 Table 5.4.4.2 Predictions for Each Crises for Years 2017 to 2036

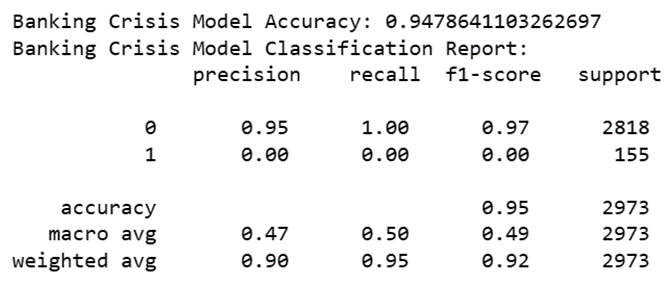
* **Banking Crisis:** The forecast probability remains fairly constant, with an initial probability of 14.77% in 2017, showing a negligible rise to 14.80% over two decades. This suggests a stable but persistent risk of banking crises over time.
* **Systemic Crisis:** The probabilities here show a slight but steady increase from 22.18% in 2017 to 22.30% in 2036. The consistency of this rise, although small, might indicate a need for increased vigilance against systemic financial disruptions.
* **Currency Crisis:** There is little variation in the probabilities for currency crises, starting at 9.9277% in 2017 and making a very slight descent to 9.9275% by 2036. The stability in these predictions suggests that while currency crises are not expected to change significantly, they seem to decrease with time.
* **Inflation Crisis:** Starting at a high probability of 44.49% in 2017, the forecast shows a steady increase, ending at 44.83% in 2036. This trend suggests an increasing risk of inflation crises, warranting attention to inflationary pressures and potentially proactive monetary policy.

These predictions suggest a fairly stable outlook for banking and currency crises but a slightly increasing risk for systemic and notably for inflation crises over the next two decades.

**5.5 EVALUATION**

**5.5.1. Performance of Logistic Regression Model**

**5.5.1.1.** **Performance of Logistic Regression Model for Banking Crisis**

 Fig 5.5.1.1 Performance of Logistic Regression Model for Banking Crisis

1. Model Accuracy (0.9478641103262697): This value represents the overall accuracy of the model, indicating that it correctly predicted whether a banking crisis would occur 94.78% of the time across all predictions made.

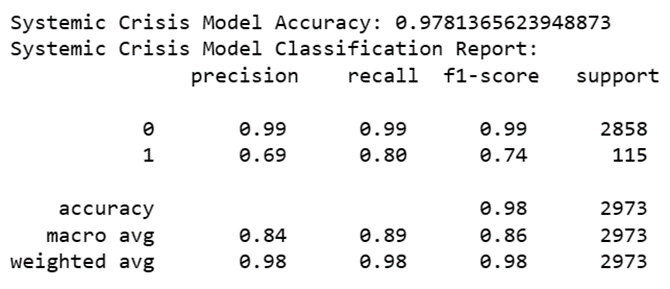
2. Classification Report: This breaks down the performance of the model in terms of precision, recall, and F1-score for each class (0 for 'no crisis' and 1 for 'crisis').

* Precision for class 0 (0.95): When the model predicted 'no banking crisis', it was correct 95% of the time.
* Recall for class 0 (1.00): The model identified 100% of the actual 'no crisis' cases correctly.
* F1-score for class 0 (0.97): The F1-score for 'no crisis' is the harmonic mean of precision and recall, suggesting high accuracy for this class.
* Precision for class 1 (0.00): The model did not correctly predict any 'crisis' events, indicating a precision of 0% for the crisis class.
* Recall for class 1 (0.00): The model failed to identify any of the actual 'crisis' cases, leading to a recall of 0% for this class.
* F1-score for class 1 (0.00): With both precision and recall at 0, the F1-score for 'crisis' is also 0, indicating poor model performance for predicting crises.
* Support: This column shows the number of actual occurrences of each class in the dataset, with 2818 instances of 'no crisis' and 155 instances of 'crisis'.

3. Macro Average: These averages are computed by taking the average of the precision, recall, and F1-scores for both classes without considering the support. The macro average for precision is 0.47, recall is 0.50, and the F1-score is 0.49. These values are low due to the model's inability to predict crises correctly.

4. Weighted Average: These averages take the support into account, thus giving more weight to the 'no crisis' class because it has more instances. The weighted averages are high (precision 0.90, recall 0.95, and F1-score 0.92) because the model performs well in predicting the more frequent 'no crisis' class.

**5.5.1.2. Performance of Logistic Regression Model for Systemic Crisis**

 Fig 5.5.1.2 Performance of Logistic Regression Model for Systemic Crisis

1. Model Accuracy (0.9781365623948873): The model has a high overall accuracy rate, correctly predicting the presence or absence of systemic crises about 97.81% of the time.

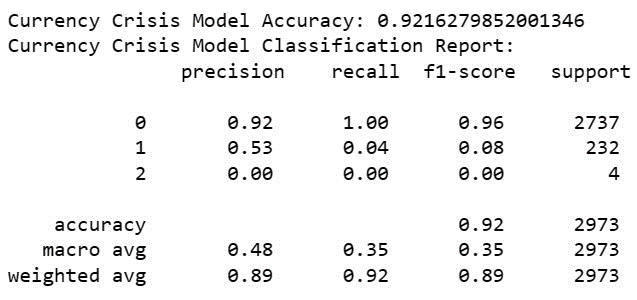
2. Classification Report: This section details the performance metrics for each class:

* Precision for class 0 (0.99): The model is highly precise in predicting the non-occurrence of systemic crises, with a 99% chance that a prediction of 'no crisis' is correct.
* Recall for class 0 (0.99): The model has a high recall rate for the non-occurrence of systemic crises, accurately identifying 99% of all 'no crisis' situations.
* F1-score for class 0 (0.99): With both high precision and recall, the model's F1-score for 'no crisis' predictions is also high, suggesting excellent model performance for this class.
* Precision for class 1 (0.69): The model has moderate precision for predicting systemic crises, with a 69% chance that a prediction of 'crisis' is correct.
* Recall for class 1 (0.80): The model is quite good at detecting actual crises, with an 80% rate of correctly identifying 'crisis' situations.
* F1-score for class 1 (0.74): The F1-score for 'crisis' predictions is reasonably high, reflecting a balance between precision and recall for this class.
* Support: The number of actual instances for each class shows that 'no crisis' instances are much more common (2858) than 'crisis' instances (115), which can affect the precision and recall calculations due to class imbalance.

3. Macro Average: The macro average takes the unweighted mean of the precision, recall, and F1-score, treating both classes equally. The macro averages are 0.84 for precision, 0.89 for recall, and 0.86 for the F1-score, which are high and suggest good overall performance across both classes.

4. Weighted Average: These averages account for the imbalance in the class distribution (support). The weighted averages for precision, recall, and F1-score are all 0.98, reflecting the high accuracy of the model, especially for the more common 'no crisis' class.

**5.5.1.3. Performance of Logistic Regression Model for Currency Crisis**

 Fig 5.5.1.3 Performance of Logistic Regression Model for Currency Crisis

1. Model Accuracy (0.9216279852001346): The model has an overall accuracy rate of approximately 92.16%, indicating that it predicts currency crises correctly most of the time.

2. Classification Report: The report breaks down the model's performance in more detail:

* Precision for class 0 (0.92): The model has a high precision for predicting the non-occurrence of currency crises, meaning it is correct 92% of the time when it predicts there will be no crisis.
* Recall for class 0 (1.00): The model has a perfect recall rate for the non-occurrence of currency crises, meaning it identifies 100% of the actual 'no crisis' instances correctly.
* F1-score for class 0 (0.96): The high F1-score for 'no crisis' predictions indicates a very effective model for predicting this class.
* Precision for class 1 (0.53): The model has moderate precision for predicting currency crises, correctly predicting such events about 53% of the time.
* Recall for class 1 (0.04): The model has a very low recall for actual crisis events, only correctly identifying 4% of these instances.
* F1-score for class 1 (0.08): The low F1-score for 'crisis' predictions indicates the model struggles with this class, likely due to a combination of a moderate precision and very low recall.
* Precision for class 2 (0.00): The model fails to correctly predict the rare class 2 events, as indicated by a precision of 0%.
* Recall for class 2 (0.00): The model also fails to identify any of the actual class 2 instances, leading to a recall of 0%.
* F1-score for class 2 (0.00): The F1-score is also 0 for class 2 predictions, indicating that the model cannot predict this class correctly.
* Support: The number of actual instances for each class in the dataset is highly imbalanced, with 2737 for class 0, 232 for class 1, and only 4 for class 2.

3. Macro Average: The macro average calculates the unweighted mean of precision, recall, and F1-score across all classes. The low macro averages (precision 0.48, recall 0.35, F1-score 0.35) suggest the model's performance is not balanced across all classes, primarily due to its inability to predict the less common classes.

4. Weighted Average: These averages consider the class imbalance (support). The weighted averages (precision 0.89, recall 0.92, F1-score 0.89) are skewed towards the most frequent class 0 due to its higher prevalence in the dataset.

**5.5.1.4. Performance of Logistic Regression Model for Inflation Crisis**

 Fig 5.5.1.4 Performance of Logistic Regression Model for Inflation Crisis

1. Model Accuracy (0.9926008672721157): This indicates a very high overall accuracy, with the model correctly predicting inflation crises nearly 99.26% of the time.

2. Classification Report: The report provides a detailed performance breakdown:

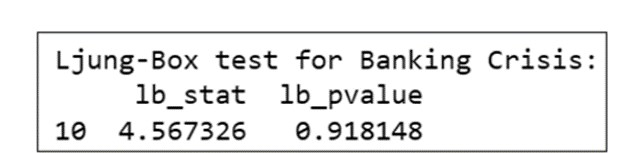
* Precision for class 0 (1.00): The model is perfect in predicting when an inflation crisis will not occur, indicating that all predictions of 'no crisis' are correct.
* Recall for class 0 (1.00): The model identifies all actual 'no crisis' instances correctly, indicating no missed 'no crisis' events.
* F1-score for class 0 (1.00): A perfect F1-score for 'no crisis' predictions suggests optimal balance and accuracy.
* Precision for class 1 (0.96): The model has high precision in predicting actual inflation crises, with a 96% chance that a prediction of 'crisis' is correct.
* Recall for class 1 (0.94): The model also has a high recall rate for actual inflation crises, correctly identifying 94% of these events.
* F1-score for class 1 (0.95): The high F1-score for 'crisis' predictions suggests a strong balance between precision and recall for this class.
* Support: The number of actual instances shows a significant class imbalance, with many more 'no crisis' instances (2746) compared to 'crisis' instances (227).

3. Macro Average: The macro averages are near-perfect (precision 0.98, recall 0.97, F1-score 0.97), reflecting the model's excellent performance across both classes equally.

4. Weighted Average: The weighted averages (precision 0.99, recall 0.99, F1-score 0.99) are also near-perfect and account for the class imbalance by giving more weight to the 'no crisis' class.

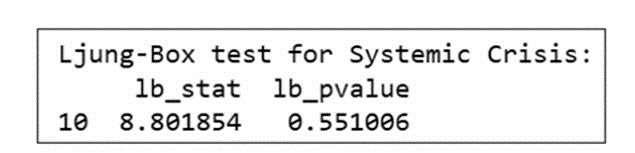
**5.5.2. Autocorrelation Test Results for Each Crisis Type**

**5.5.2.1 Autocorrelation Test Results for Banking Crisis**

 Fig 5.5.2.1 Autocorrelation Test Results for Banking Crisis

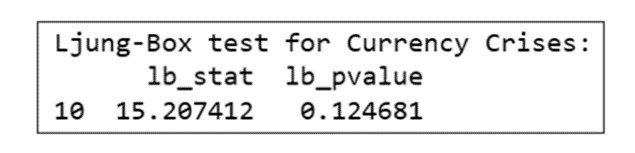
The p-value of 0.918148 is very high, suggesting that there is no significant autocorrelation at any of the first 10 lags of the residual time series for banking crises.

**5.5.2.2 Autocorrelation Test Results for Systemic Crisis**

Fig 5.5.2.2 Autocorrelation Test Results for Systemic Crisis

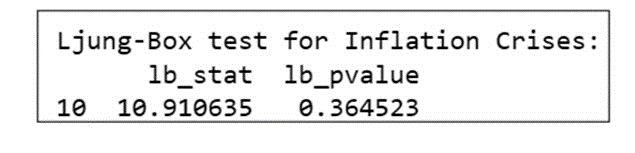
The p-value of 0.551086 is also high, indicating no significant autocorrelation at the first 10 lags for systemic crises.

**5.5.2.3 Autocorrelation Test Results for Currency Crisis**

Fig 5.5.2.3 Autocorrelation Test Results for Currency Crisis

With a p-value of 0.124681, this result is lower but still above the common significance threshold of 0.05, suggesting no significant autocorrelation at the first 10 lags for currency crises.

**5.5.2.4 Autocorrelation Test Results for Inflation Crisis**

Fig 5.5.2.4 Autocorrelation Test Results for Inflation Crisis

The p-value of 0.364523 indicates no significant autocorrelation at the first 10 lags for inflation crises.

Since all these values are above the common alpha level of 0.05, they indicate that there's no significant autocorrelation present in the residuals of your models for these different types of crises. This suggests that the models are appropriate for the data and that the residuals (differences between the observed and predicted values) are random, which is a sign of a good fit.

**5.5.3. LSTM Model's Root Mean Square Error (RMSE) Results**

**5.5.3.1 LSTM RMSE Results for Banking Crisis**



Fig 5.5.3.1 LSTM RMSE Results for Banking Crisis

The LSTM model's RMSE of 2.42 for training shows a close fit to training data, while 6.31 for testing indicates less accuracy in predicting new data, reflecting its generalization capability.

**5.5.3.2 LSTM RMSE Results for Systemic Crisis**



Fig 5.5.3.2 LSTM RMSE Results for Systemic Crisis

The LSTM model's RMSE of 0.14 for training and 0.15 for testing suggests high accuracy and effective generalization, with predictions closely aligning with actual values in both datasets.

**5.5.3.3 LSTM RMSE Results for Currency Crisis**



Fig 5.5.3.3 LSTM RMSE Results for Currency Crisis

The LSTM model shows strong fit and accuracy with RMSEs of 0.25 for training and 0.28 for testing, indicating reliable predictions for both training and unseen data.

**5.5.3.4 LSTM RMSE Results for Inflation Crisis**



Fig 5.5.3.4 LSTM RMSE Results for Inflation Crisis

The LSTM model's RMSE of 0.23 for training and 0.21 for testing indicates high accuracy and effective generalization, with closer predictions to actual values in both training and unseen data.

**6 RESULTS AND DISCUSSIONS**

|  |
| --- |
|  |
| **Crisis Type** | **Logistic Regression** | **ARIMA** | **SARIMAX** | **LSTM** |
| **Banking Crisis** | - Accuracy: 94.78% | - Captured time-series patterns | - Improved accuracy | - Unique approach |
| - Precision (Non-crisis): 0.95 | - Temporal dynamics insights | - Robust insights | - Competitive accuracy |
| - Recall (Non-crisis): 1.00 | - Policy-informative predictions | - Seasonal patterns captured | - Complex temporal dependencies |
| - Precision (Crisis): 0.00 |  |  | - Potential in time-series forecasting |
| - Recall (Crisis): 0.00 |  |  |  |
| **Systemic Crisis** | - Accuracy: 97.81% | - Captured time-series patterns | - Improved prediction accuracy | - Capability in capturing dependencies |
| - Precision (Non-systemic): 0.99 | - Temporal aspects insights | - Unveiled seasonal trends | - High accuracy rate |
| - Recall (Non-systemic): 0.99 | - Risk management insights | - Seasonal trends captured | - Intricate temporal dynamics |
| - Precision (Systemic): 0.69 |  |  | - Suitability for systemic risk modeling |
| - Recall (Systemic): 0.80 |  |  |  |
| **Currency Crisis** | - Accuracy: 92.16% | - Temporal patterns insights | - Improved prediction accuracy | - Unique insights |
| - Precision (Non-crisis): 0.92 | - Dynamics leading to crises insights | - Seasonal trends captured | - Competitive accuracy |
| - Recall (Non-crisis): 1.00 | - Currency market risk insights | - Seasonal currency crisis trends | - Intricate temporal dependencies |
| - Precision (Crisis): 0.53 |  |  | - Potential in crisis forecasting |
| - Recall (Crisis): 0.04 |  |  |  |
| **Inflation Crisis** | - Accuracy: 99.26% | - Temporal patterns insights | - Improved prediction accuracy | - Unique insights |
| - Precision (Non-crisis): 1.00 | - Temporal aspects insights | - Seasonal trends captured | - High accuracy |
| - Recall (Non-crisis): 1.00 | - Inflation risk assessments insights | - Seasonal inflation crisis trends | - Complex temporal dependencies |
| - Precision (Crisis): 0.96 |  |  | - Potential in crisis forecasting |
| - Recall (Crisis): 0.94 |  |  |  |

Table 6 Discussion of Results for Each Crisis Type for Each Model

**7 CONCLUSION**

Throughout the duration of this project, we embarked on a fascinating journey through the annals of financial history, spanning over two centuries. The project's robust foundation lies in the dataset itself—an impeccably structured and well-balanced compilation that offers a panoramic view of financial crises across different time periods and geographical regions. The dataset's historical depth enabled us to unravel the intricate web of financial events, drawing connections between the past and present and discerning overarching patterns that have characterized these crises throughout time.

One of the dataset's most striking attributes is the uniform distribution of records over time. This uniformity goes beyond statistical significance; it reflects the meticulous care taken during data curation. It ensures that our analyses are grounded in an unbiased representation of history, guarding against the potential pitfalls of data bias.

This project's significance extends far beyond data compilation; it serves as a springboard for profound investigations into the prevalence, causes, and consequences of financial instability. While our initial findings have piqued our curiosity, revealing promising avenues for more detailed analyses and research inquiries, we anticipate uncovering the nuanced dynamics that underlie financial crises as we delve deeper into this treasure trove of historical data.

The implications of our work reach beyond the academic sphere. The insights gleaned from this dataset have the potential to inform policy decisions in the realm of financial risk management. Policymakers can draw upon historical precedents to formulate strategies aimed at mitigating the impact of future crises, thereby safeguarding economic stability.

In essence, this project offers a rich tapestry of financial history, intricately woven with data, analysis, and untapped potential. It serves as both a tribute to the past and a guide for the present, illuminating a path for future researchers to explore the wealth of historical data. As we conclude this phase of our journey, we acknowledge that our work has merely scratched the surface. The dataset before us represents uncharted territory, teeming with untold stories and hidden truths. It beckons future researchers to venture into its depths, armed with the knowledge and insights we have uncovered.

Ultimately, this project stands as a testament to the enduring relevance of financial history—a discipline that offers lessons from the past to navigate the challenges of the future. It is an impassioned call to action, a challenge to unravel the complexities of financial crises, and an open invitation to contribute to the collective understanding of global finance. As we step away from this project, we do so with gratitude for the knowledge gained and with eager anticipation for the discoveries yet to be made.

**8 FUTURE ENHANCEMENT**

In contemplating the evolution of this project, several pivotal improvements emerge as essential. To begin, it is crucial to broaden the dataset's scope by incorporating a wider array of economic indicators. This expansion promises to provide a more comprehensive view of economic conditions and the intricate factors influencing financial stability, enriching the project's analytical depth.

The integration of real-time data represents another pivotal direction. By including live data feeds, the project can embrace dynamic forecasting, empowering it with the ability to monitor financial indicators in real-time and swiftly respond to emerging crises. This step enhances the project's adaptability and contemporary relevance.

In the pursuit of heightened accuracy, it is imperative to explore alternative machine learning algorithms. Diversifying the approach by considering neural networks, ensemble methods, and deep learning models can result in more precise predictions of financial crises. This bolsters the project's predictive capabilities.

Cross-country analyses are fundamental for ensuring that the project's findings hold universal applicability. By scrutinizing data across diverse nations, researchers can uncover common trends, regional disparities, and factors that transcend borders, bolstering the project's global significance.

Lastly, a sophisticated model blending strategy can enhance the project's predictive power. By amalgamating various modeling approaches, such as ensemble modeling and Bayesian techniques, the project can yield more resilient and accurate predictions. This reinforces its capacity to comprehend, foresee, and address financial crises on a global scale.

These enhancements, collectively shaping the project's trajectory, promise an enriched understanding of financial crises and a more robust toolkit for informed decision-making in the realm of global finance.

**9 APPENDICIES**

**9.1 FULL CODE**

import pandas as pd

import geopandas as gpd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import sqlite3

import shutil

from tabulate import tabulate

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn import metrics

from IPython.display import display

from IPython.core.display import HTML

import warnings

warnings.filterwarnings('ignore')

world\_data = gpd.read\_file(r"C:\Users\dhars\Desktop\TRIMESTER 1\Z-Major Project T1\Dataset\world-data.geojson").query('name != "Antarctica"')

world\_data.plot(figsize=(8,8))

plt.show()

world\_data.head()

crisis\_data = pd.read\_csv(r"C:\Users\dhars\Desktop\TRIMESTER 1\Z-Major Project T1\Dataset\20160923\_global\_crisis\_data.csv")

# Minor data cleansing

crisis\_data = crisis\_data.query('Year > 1800').replace({np.nan:0})

crisis\_data['Banking Crisis '] = crisis\_data['Banking Crisis '].apply(

lambda d: int(d)

)

crisis\_data['Systemic Crisis'] = crisis\_data['Systemic Crisis'].apply(

lambda d: int(d)

)

crisis\_data['Currency Crises'] = crisis\_data['Currency Crises'].apply(

lambda d: int(d)

)

crisis\_data['Inflation Crises'] = crisis\_data['Inflation Crises'].apply(

lambda d: int(d)

)

crisis\_data.head()

crisis\_data.describe()

crisis\_data.nunique()

# Performing outliers test

numerical\_cols = crisis\_data.select\_dtypes(include=['float64', 'int64']).columns.tolist()

def detect\_outliers(df, features):

outlier\_indices = []

for col in features:

Q1 = df[col].quantile(0.25)

Q3 = df[col].quantile(0.75)

IQR = Q3 - Q1

outlier\_step = 1.5 \* IQR

outlier\_list\_col = df[(df[col] < Q1 - outlier\_step) | (df[col] > Q3 + outlier\_step)].index

outlier\_indices.extend(outlier\_list\_col)

outlier\_indices = list(set(outlier\_indices))

return outlier\_indices

outliers = detect\_outliers(crisis\_data, numerical\_cols)

outliers\_data = crisis\_data.loc[outliers]

outliers\_data

# Calculate the number of missing values per column

missing\_values = crisis\_data.isnull().sum()

print(missing\_values)

numerical\_columns = ['Case', 'Year']

n\_cols = len(numerical\_columns)

n\_rows = 1 # Since we have only two columns, we need only one row

# Set up the matplotlib figure

fig, axes = plt.subplots(nrows=n\_rows, ncols=n\_cols, figsize=(6 \* n\_cols, 4 \* n\_rows))

# Create a boxplot for each of the specified columns

for i, col in enumerate(numerical\_columns):

sns.boxplot(x=crisis\_data[col], ax=axes[i])

axes[i].set\_title(f'Boxplot for {col}')

# Adjust layout

plt.tight\_layout()

plt.show()

# Histogram of the 'Year' column

plt.figure(figsize=(10, 6))

sns.histplot(crisis\_data['Year'].dropna(), bins=50, kde=True)

plt.title('Distribution of Years in Dataset')

plt.xlabel('Year')

plt.ylabel('Frequency')

plt.show()

# Count plot of the top 10 countries by the number of entries

top\_countries = crisis\_data['Country'].value\_counts().head(70).index

plt.figure(figsize=(20, 14))

sns.countplot(y=crisis\_data['Country'], order=top\_countries)

plt.title('Number of Entries for Each Country')

plt.xlabel('Count')

plt.ylabel('Country')

plt.show()

# Selected columns for the correlation matrix

selected\_columns = ['Year', 'Gold Standard', 'exch\_usd', 'Domestic\_Debt\_In\_Default',

'GDP\_Weighted\_default', 'Inflation, Annual percentages of average consumer prices',

'Independence', 'Banking Crisis ', 'Systemic Crisis', 'Currency Crises', 'Inflation Crises']

# Replace spaces with NaN and convert columns to float

for col in selected\_columns:

crisis\_data[col] = pd.to\_numeric(crisis\_data[col].replace(' ', np.nan), errors='coerce')

# Drop rows with NaN values in selected columns

crisis\_data = crisis\_data.dropna(subset=selected\_columns)

# Calculate the correlation matrix

correlation\_matrix = crisis\_data[selected\_columns].corr()

# Renaming 'Inflation, Annual percentages of average consumer prices' to 'Inflation Annual Percentages' for the heatmap

correlation\_matrix = correlation\_matrix.rename(index={'Inflation, Annual percentages of average consumer prices': 'Inflation Annual Percentages'},

columns={'Inflation, Annual percentages of average consumer prices': 'Inflation Annual Percentages'})

# Plotting the heatmap

plt.figure(figsize=(12, 10))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Heatmap')

plt.show()

crisis\_summary = crisis\_data.groupby(['CC3']).agg({'Banking Crisis ':'sum', 'Systemic Crisis':'sum','Currency Crises':'sum', 'Inflation Crises':'sum' })

crisis\_summary

crisis\_world\_summary = world\_data.merge(crisis\_summary, left\_on='iso\_a3', right\_on='CC3', how='left')

crisis\_world\_summary.replace({np.nan:0}, inplace=True)

crisis\_world\_summary.head()

top\_limit = 20

title = f'<div style="font-weight: bold; font-size: 16pt;">Top {top\_limit} Countries with most Banking Crisis events</div>'

subtitle = f'<div style="font-size: 12pt;">1800 to Present</div>'

display(HTML(title))

display(HTML(subtitle))

crisis\_world\_summary[['name','continent', 'Banking Crisis ']] \

.sort\_values(['Banking Crisis '], ascending=[False]) \

.head(top\_limit)

top\_limit = 20

title = f'<div style="font-weight: bold; font-size: 16pt;">Top {top\_limit} Countries with most Systemic Crisis events</div>'

subtitle = f'<div style="font-size: 12pt;">1800 to Present</div>'

display(HTML(title))

display(HTML(subtitle))

crisis\_world\_summary[['name','continent', 'Systemic Crisis']] \

.sort\_values(['Systemic Crisis'], ascending=[False]) \

.head(top\_limit)

top\_limit = 20

title = f'<div style="font-weight: bold; font-size: 16pt;">Top {top\_limit} Countries with most Currency Crisis events</div>'

subtitle = f'<div style="font-size: 12pt;">1800 to Present</div>'

display(HTML(title))

display(HTML(subtitle))

crisis\_world\_summary[['name','continent', 'Currency Crises']] \

.sort\_values(['Currency Crises'], ascending=[False]) \

.head(top\_limit)

top\_limit = 20

title = f'<div style="font-weight: bold; font-size: 16pt;">Top {top\_limit} Countries with most Inflation Crisis events</div>'

subtitle = f'<div style="font-size: 12pt;">1800 to Present</div>'

display(HTML(title))

display(HTML(subtitle))

crisis\_world\_summary[['name','continent', 'Inflation Crises']] \

.sort\_values(['Inflation Crises'], ascending=[False]) \

.head(top\_limit)

import matplotlib.pyplot as plt

fig, axes = plt.subplots(4, 1, figsize=(20, 20))

# Banking Crisis (Changed color to purple)

ax1 = axes[0]

ax1.set\_title('Global Banking Crisis Summary (1800 to Present)', fontsize=14)

ax1.set\_facecolor('lightblue')

crisis\_world\_summary.plot(

ax=ax1,

cmap='Purples', # Changed cmap color to purple

edgecolor='black',

column='Banking Crisis ',

legend=True,

k=5,

scheme='fisher\_jenks',

legend\_kwds=dict(loc='lower left', fmt='{:,.0f}')

)

# Systemic Crisis

ax2 = axes[1]

ax2.set\_title('Global Systemic Crisis Summary (1800 to Present)', fontsize=14)

ax2.set\_facecolor('lightblue')

crisis\_world\_summary.plot(

ax=ax2,

cmap='Blues',

edgecolor='black',

column='Systemic Crisis',

legend=True,

k=5,

scheme='fisher\_jenks',

legend\_kwds=dict(loc='lower left', fmt='{:,.0f}')

)

# Currency Crises

ax3 = axes[2]

ax3.set\_title('Global Currency Crisis Summary (1800 to Present)', fontsize=14)

ax3.set\_facecolor('lightblue')

crisis\_world\_summary.plot(

ax=ax3,

cmap='Greens',

edgecolor='black',

column='Currency Crises',

legend=True,

k=5,

scheme='fisher\_jenks',

legend\_kwds=dict(loc='lower left', fmt='{:,.0f}')

)

# Inflation Crises

ax4 = axes[3]

ax4.set\_title('Global Inflation Crisis Summary (1800 to Present)', fontsize=14)

ax4.set\_facecolor('lightblue')

crisis\_world\_summary.plot(

ax=ax4,

cmap='Reds',

edgecolor='black',

column='Inflation Crises',

legend=True,

k=5,

scheme='fisher\_jenks',

legend\_kwds=dict(loc='lower left', fmt='{:,.0f}')

)

plt.show()

crisis\_summary = crisis\_data \

.groupby(['Year']) \

.agg({'Banking Crisis ':'sum', 'Systemic Crisis':'sum','Currency Crises':'sum', 'Inflation Crises':'sum'}) \

.sort\_values(['Year'], ascending=[True])

crisis\_summary

crisis\_cumulative\_summary = crisis\_summary[['Banking Crisis ', 'Systemic Crisis','Currency Crises','Inflation Crises']] \

.agg({'Banking Crisis ':'cumsum', 'Systemic Crisis':'cumsum','Currency Crises':'cumsum', 'Inflation Crises':'cumsum'})

crisis\_cumulative\_summary

import matplotlib.pyplot as plt

fig, axs = plt.subplots(4, 1, figsize=(20, 16))

# First Part: Individual subplots for each crisis type by year

axs[0].set\_title('Banking Crisis by Year (1800 to Present)', fontsize=16)

crisis\_summary['Banking Crisis '].plot(kind='line', color='orange', ax=axs[0], grid=True)

axs[1].set\_title('Systemic Crisis by Year (1800 to Present)', fontsize=16)

crisis\_summary['Systemic Crisis'].plot(kind='line', color='blue', ax=axs[1], grid=True)

axs[2].set\_title('Currency Crises by Year (1800 to Present)', fontsize=16)

crisis\_summary['Currency Crises'].plot(kind='line', color='red', ax=axs[2], grid=True)

axs[3].set\_title('Inflation Crises by Year (1800 to Present)', fontsize=16)

crisis\_summary['Inflation Crises'].plot(kind='line', color='green', ax=axs[3], grid=True)

# Second Part: Cumulative summary

fig, ax2 = plt.subplots(1, 1, figsize=(20, 8))

ax2.set\_title('Banking, Systemic, Currency and Inflation Crises Cumulative (1800 to Present)', fontsize=16)

crisis\_cumulative\_summary.plot(kind='line', color=['orange', 'blue', 'red', 'green'], ax=ax2, grid=True)

ax2.legend(loc='upper left')

plt.show()

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report

# Prepare your data for 'Banking Crisis'

banking\_crisis\_target = crisis\_data['Banking Crisis '].apply(pd.to\_numeric, errors='coerce') # Convert to numeric

banking\_crisis\_features = crisis\_data[['Systemic Crisis', 'Gold Standard', 'exch\_usd', 'Inflation, Annual percentages of average consumer prices', 'Independence', 'Currency Crises', 'Inflation Crises']].apply(pd.to\_numeric, errors='coerce')

# Drop rows with NaN values

non\_na\_indices = banking\_crisis\_features.dropna().index

banking\_crisis\_features = banking\_crisis\_features.loc[non\_na\_indices]

banking\_crisis\_target = banking\_crisis\_target.loc[non\_na\_indices]

# Split data into training and testing sets for 'Banking Crisis'

X\_train\_bc, X\_test\_bc, y\_train\_bc, y\_test\_bc = train\_test\_split(banking\_crisis\_features, banking\_crisis\_target, test\_size=0.2, random\_state=42)

# Create and train the logistic regression model for 'Banking Crisis'

model\_bc = LogisticRegression()

model\_bc.fit(X\_train\_bc, y\_train\_bc)

# Make predictions for 'Banking Crisis'

predictions\_bc = model\_bc.predict(X\_test\_bc)

# Evaluate the model for 'Banking Crisis'

print("Banking Crisis Model Accuracy:", accuracy\_score(y\_test\_bc, predictions\_bc))

print("Banking Crisis Model Classification Report:\n", classification\_report(y\_test\_bc, predictions\_bc))

# Prepare your data for 'Systemic Crisis'

systemic\_crisis\_target = crisis\_data['Systemic Crisis'].apply(pd.to\_numeric, errors='coerce') # Convert to numeric

systemic\_crisis\_features = crisis\_data[['Banking Crisis ', 'Gold Standard', 'exch\_usd', 'Inflation, Annual percentages of average consumer prices', 'Independence', 'Currency Crises', 'Inflation Crises']].apply(pd.to\_numeric, errors='coerce')

# Drop rows with NaN values

non\_na\_indices = systemic\_crisis\_features.dropna().index

systemic\_crisis\_features = systemic\_crisis\_features.loc[non\_na\_indices]

systemic\_crisis\_target = systemic\_crisis\_target.loc[non\_na\_indices]

# Split data into training and testing sets for 'Systemic Crisis'

X\_train\_sc, X\_test\_sc, y\_train\_sc, y\_test\_sc = train\_test\_split(systemic\_crisis\_features, systemic\_crisis\_target, test\_size=0.2, random\_state=42)

# Create and train the logistic regression model for 'Systemic Crisis'

model\_sc = LogisticRegression()

model\_sc.fit(X\_train\_sc, y\_train\_sc)

# Make predictions for 'Systemic Crisis'

predictions\_sc = model\_sc.predict(X\_test\_sc)

# Evaluate the model for 'Systemic Crisis'

print("Systemic Crisis Model Accuracy:", accuracy\_score(y\_test\_sc, predictions\_sc))

print("Systemic Crisis Model Classification Report:\n", classification\_report(y\_test\_sc, predictions\_sc))

# Prepare your data for 'Currency Crises'

currency\_crisis\_target = crisis\_data['Currency Crises'].apply(pd.to\_numeric, errors='coerce') # Convert to numeric

currency\_crisis\_features = crisis\_data[['Banking Crisis ','Systemic Crisis', 'Gold Standard', 'exch\_usd', 'Inflation, Annual percentages of average consumer prices', 'Independence', 'Inflation Crises']].apply(pd.to\_numeric, errors='coerce')

# Drop rows with NaN values

non\_na\_indices = currency\_crisis\_features.dropna().index

currency\_crisis\_features = currency\_crisis\_features.loc[non\_na\_indices]

currency\_crisis\_target = currency\_crisis\_target.loc[non\_na\_indices]

# Split data into training and testing sets for 'Currency Crises'

X\_train\_bc, X\_test\_bc, y\_train\_bc, y\_test\_bc = train\_test\_split(currency\_crisis\_features, currency\_crisis\_target, test\_size=0.2, random\_state=42)

# Create and train the logistic regression model for 'Currency Crises'

model\_bc = LogisticRegression()

model\_bc.fit(X\_train\_bc, y\_train\_bc)

# Make predictions for 'Currency Crisis'

predictions\_bc = model\_bc.predict(X\_test\_bc)

# Evaluate the model for 'Currency Crisis'

print("Currency Crisis Model Accuracy:", accuracy\_score(y\_test\_bc, predictions\_bc))

print("Currency Crisis Model Classification Report:\n", classification\_report(y\_test\_bc, predictions\_bc))

# Prepare your data for 'Inflation Crises'

inflation\_crisis\_target = crisis\_data['Inflation Crises'].apply(pd.to\_numeric, errors='coerce') # Convert to numeric

inflation\_crisis\_features = crisis\_data[['Banking Crisis ','Systemic Crisis', 'Gold Standard', 'exch\_usd', 'Inflation, Annual percentages of average consumer prices', 'Independence', 'Currency Crises']].apply(pd.to\_numeric, errors='coerce')

# Drop rows with NaN values

non\_na\_indices = currency\_crisis\_features.dropna().index

inflation\_crisis\_features = inflation\_crisis\_features.loc[non\_na\_indices]

inflation\_crisis\_target = inflation\_crisis\_target.loc[non\_na\_indices]

# Split data into training and testing sets for 'Inflation Crises'

X\_train\_bc, X\_test\_bc, y\_train\_bc, y\_test\_bc = train\_test\_split(inflation\_crisis\_features, inflation\_crisis\_target, test\_size=0.2, random\_state=42)

# Create and train the logistic regression model for 'Inflation Crises'

model\_bc = LogisticRegression()

model\_bc.fit(X\_train\_bc, y\_train\_bc)

# Make predictions for 'Inflation Crisis'

predictions\_bc = model\_bc.predict(X\_test\_bc)

# Evaluate the model for 'Inflation Crisis'

print("Inflation Crisis Model Accuracy:", accuracy\_score(y\_test\_bc, predictions\_bc))

print("Inflation Crisis Model Classification Report:\n", classification\_report(y\_test\_bc, predictions\_bc))

from statsmodels.tsa.arima.model import ARIMA

import matplotlib.pyplot as plt

import pandas as pd

import warnings

warnings.filterwarnings('ignore')

# Load the dataset

dataset\_path = r'C:\Users\dhars\Desktop\TRIMESTER 1\Z-Major Project T1\Dataset\20160923\_global\_crisis\_data.csv'

crisis\_data = pd.read\_csv(dataset\_path)

# Data cleaning

crisis\_data = crisis\_data.drop(0) # drop the first row if it is not part of the data

crisis\_data['Year'] = pd.to\_datetime(crisis\_data['Year'], format='%Y') # convert 'Year' to datetime

crisis\_data.set\_index('Year', inplace=True) # set 'Year' as the index

# Correct column names based on the actual column names in the dataset

crisis\_columns = [

'Banking Crisis ', # Notice the space at the end

'Systemic Crisis',

'Currency Crises',

'Inflation Crises'

]

# Convert crisis columns to numeric and handle missing values

for col in crisis\_columns:

crisis\_data[col] = pd.to\_numeric(crisis\_data[col], errors='coerce')

# Aggregate the data to have one observation per year

crisis\_data\_aggregated = crisis\_data.groupby(crisis\_data.index).sum()

# Define a function to perform ARIMA forecasting and plotting

def forecast\_arima(series, order, title):

model = ARIMA(series, order=order)

model\_fit = model.fit()

forecast = model\_fit.get\_forecast(steps=12)

forecast\_index = pd.date\_range(series.index[-1], periods=12, freq='A') # Annual frequency

plt.figure(figsize=(12, 3))

plt.plot(series, label='Historical Data')

plt.plot(forecast\_index, forecast.predicted\_mean, label='Forecast', color='red')

plt.fill\_between(forecast\_index,

forecast.conf\_int().iloc[:, 0],

forecast.conf\_int().iloc[:, 1], color='pink', alpha=0.3)

plt.title(title)

plt.legend()

plt.show()

# Forecast each type of crisis using the ARIMA model

arima\_orders = {

'Banking Crisis ': (1, 1, 0),

'Systemic Crisis': (1, 1, 0),

'Currency Crises': (1, 1, 0),

'Inflation Crises': (1, 1, 0)

}

for crisis\_type, order in arima\_orders.items():

forecast\_arima(crisis\_data\_aggregated[crisis\_type], order, f"{crisis\_type.strip()} Forecast")

import warnings

warnings.filterwarnings('ignore')

# Define a function to generate and plot residual diagnostics for ARIMA models

def plot\_residuals(series, order, title):

# Fit the ARIMA model

model = ARIMA(series, order=order)

model\_fit = model.fit()

# Get residuals

residuals = model\_fit.resid

# Create subplots for residual diagnostics

plt.figure(figsize=(15, 10))

# Residuals vs. Time Plot

plt.subplot(2, 2, 1)

plt.plot(residuals)

plt.title('Residuals vs. Time')

# Kernel Density Plot of Residuals

plt.subplot(2, 2, 2)

residuals.plot(kind='kde')

plt.title('Kernel Density Plot of Residuals')

# ACF Plot of Residuals

plt.subplot(2, 2, 3)

pd.plotting.autocorrelation\_plot(residuals)

plt.title('ACF Plot of Residuals')

# Histogram of Residuals

plt.subplot(2, 2, 4)

plt.hist(residuals, bins=20)

plt.title('Histogram of Residuals')

plt.suptitle(title, fontsize=16)

plt.tight\_layout()

plt.show()

# Define ARIMA order parameters for each crisis type

arima\_orders\_corrected = {

'Banking Crisis ': (1, 1, 0), # Replace with appropriate values

'Systemic Crisis': (1, 1, 0), # Replace with appropriate values

'Currency Crises': (1, 1, 0), # Replace with appropriate values

'Inflation Crises': (1, 1, 0) # Replace with appropriate values

}

# Plot residuals for each type of crisis using the ARIMA models

for crisis\_type, order in arima\_orders\_corrected.items():

plot\_residuals(crisis\_data\_aggregated[crisis\_type], order, f"Residual Diagnostics for {crisis\_type.strip()}")

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf

from statsmodels.tsa.seasonal import seasonal\_decompose

import warnings

warnings.filterwarnings('ignore')

# Load the dataset

dataset\_path = r'C:\Users\dhars\Desktop\TRIMESTER 1\Z-Major Project T1\Dataset\20160923\_global\_crisis\_data.csv'

crisis\_data = pd.read\_csv(dataset\_path)

# Convert crisis columns to numeric and handle non-numeric values

crisis\_columns = ['Banking Crisis ', 'Systemic Crisis', 'Currency Crises', 'Inflation Crises']

for col in crisis\_columns:

crisis\_data[col] = pd.to\_numeric(crisis\_data[col], errors='coerce')

crisis\_data.fillna(0, inplace=True) # Assuming no crisis is represented as missing values

# Filter data for years from 1800 onwards

crisis\_data = crisis\_data[crisis\_data['Year'] >= 1800]

# Aggregate data for each crisis type by year

crisis\_data\_aggregated = {

'Banking Crisis': crisis\_data.groupby('Year')['Banking Crisis '].sum(),

'Systemic Crisis': crisis\_data.groupby('Year')['Systemic Crisis'].sum(),

'Currency Crises': crisis\_data.groupby('Year')['Currency Crises'].sum(),

'Inflation Crises': crisis\_data.groupby('Year')['Inflation Crises'].sum()

}

# Function to plot seasonal decomposition and ACF/PACF

def plot\_seasonal\_decomposition(crisis\_type, series):

# Seasonal Decomposition

result = seasonal\_decompose(series, model='additive', period=12)

fig = result.plot()

plt.suptitle(f'Seasonal Decomposition - {crisis\_type}', fontsize=14)

# Adjusting layout to prevent overlapping

fig.tight\_layout(rect=[0, 0.03, 1, 0.95]) # Adjust these values as needed

# ACF and PACF Plots

plt.figure(figsize=(10, 5))

plt.subplot(121)

plot\_acf(series, ax=plt.gca(), title=f'ACF - {crisis\_type}')

plt.subplot(122)

plot\_pacf(series, ax=plt.gca(), title=f'PACF - {crisis\_type}')

plt.tight\_layout() # Adjust layout here as well

plt.show()

# Generate plots for each crisis type

for crisis\_type, series in crisis\_data\_aggregated.items():

plot\_seasonal\_decomposition(crisis\_type, series)

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.statespace.sarimax import SARIMAX

import warnings

warnings.filterwarnings('ignore')

# Load the dataset

dataset\_path = r'C:\Users\dhars\Desktop\TRIMESTER 1\Z-Major Project T1\Dataset\20160923\_global\_crisis\_data.csv'

crisis\_data = pd.read\_csv(dataset\_path)

# Convert crisis columns to numeric and handle non-numeric values

crisis\_columns = ['Banking Crisis ', 'Systemic Crisis', 'Currency Crises', 'Inflation Crises']

for col in crisis\_columns:

crisis\_data[col] = pd.to\_numeric(crisis\_data[col], errors='coerce')

crisis\_data.fillna(0, inplace=True) # Assuming no crisis is represented as missing values

# Filter data for years from 1800 onwards

crisis\_data = crisis\_data[crisis\_data['Year'] >= 1800]

# Aggregate data for each crisis type by year and set 'Year' as index

crisis\_data\_aggregated = {

'Banking Crisis': crisis\_data.groupby('Year')['Banking Crisis '].sum(),

'Systemic Crisis': crisis\_data.groupby('Year')['Systemic Crisis'].sum(),

'Currency Crises': crisis\_data.groupby('Year')['Currency Crises'].sum(),

'Inflation Crises': crisis\_data.groupby('Year')['Inflation Crises'].sum()

}

for crisis\_type in crisis\_data\_aggregated:

crisis\_data\_aggregated[crisis\_type].index = pd.to\_datetime(crisis\_data\_aggregated[crisis\_type].index, format='%Y')

# Define SARIMA orders for each crisis type (replace with your determined values)

sarima\_orders = {

'Banking Crisis': {'order': (1, 0, 1), 'seasonal\_order': (1, 1, 1, 12)},

'Systemic Crisis': {'order': (1, 0, 1), 'seasonal\_order': (1, 1, 1, 12)},

'Currency Crises': {'order': (1, 0, 1), 'seasonal\_order': (1, 1, 1, 12)},

'Inflation Crises': {'order': (1, 0, 1), 'seasonal\_order': (1, 1, 1, 12)}

}

# Fit and forecast with SARIMA for each crisis type

for crisis\_type, params in sarima\_orders.items():

print(f"Processing {crisis\_type}...")

series = crisis\_data\_aggregated[crisis\_type]

# Fit SARIMA model

model = SARIMAX(series, order=params['order'], seasonal\_order=params['seasonal\_order'])

results = model.fit()

# Example of forecasting the next 12 periods

forecast = results.get\_forecast(steps=12)

forecast\_values = forecast.predicted\_mean

forecast\_conf\_int = forecast.conf\_int()

print(f"Forecast for {crisis\_type}:")

print(forecast\_values)

# Optional: Plotting the forecast with confidence intervals

plt.figure(figsize=(18, 3))

plt.plot(series, label='Observed')

plt.plot(forecast\_values, label='Forecast', color='red')

plt.fill\_between(forecast\_values.index,

forecast\_conf\_int.iloc[:, 0],

forecast\_conf\_int.iloc[:, 1],

color='pink', alpha=0.3)

plt.title(f"Forecast for {crisis\_type}")

plt.legend()

plt.show()

import matplotlib.pyplot as plt

from statsmodels.graphics.tsaplots import plot\_acf

from statsmodels.stats.diagnostic import acorr\_ljungbox

# After fitting the SARIMA model for each crisis type

for crisis\_type, params in sarima\_orders.items():

print(f"Processing {crisis\_type}...")

series = crisis\_data\_aggregated[crisis\_type]

# Fit SARIMA model

model = SARIMAX(series, order=params['order'], seasonal\_order=params['seasonal\_order'])

results = model.fit()

# Plotting residuals

plt.figure(figsize=(10, 4))

plt.plot(results.resid)

plt.title(f'Residuals for {crisis\_type}')

plt.ylabel('Residual')

plt.xlabel('Year')

plt.axhline(y=0, color='r', linestyle='-')

plt.show()

# Autocorrelation plot of residuals

plt.figure(figsize=(10, 4))

plot\_acf(results.resid, lags=30)

plt.title(f'Autocorrelation of Residuals for {crisis\_type}')

plt.show()

# Ljung-Box test

lb\_test = acorr\_ljungbox(results.resid, lags=[10], return\_df=True)

print(f"Ljung-Box test for {crisis\_type}:")

print(lb\_test)

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.statespace.sarimax import SARIMAX

import warnings

warnings.filterwarnings('ignore')

# Load the dataset

dataset\_path = r'C:\Users\dhars\Desktop\TRIMESTER 1\Z-Major Project T1\Dataset\20160923\_global\_crisis\_data.csv'

crisis\_data = pd.read\_csv(dataset\_path)

# Convert relevant columns to numeric, assuming non-numeric entries indicate no crisis

crisis\_columns = ['Banking Crisis ', 'Systemic Crisis', 'Currency Crises', 'Inflation Crises']

for col in crisis\_columns:

crisis\_data[col] = pd.to\_numeric(crisis\_data[col], errors='coerce').fillna(0)

# Filter for years from 1800 onwards and aggregate data by year

crisis\_data = crisis\_data[crisis\_data['Year'] >= 1800]

crisis\_data['Year'] = pd.to\_datetime(crisis\_data['Year'], format='%Y')

crisis\_data.set\_index('Year', inplace=True)

# Aggregate data for each crisis type by year

crisis\_data\_aggregated = {

'Banking Crisis ': crisis\_data['Banking Crisis '].resample('YS').sum(),

'Systemic Crisis': crisis\_data['Systemic Crisis'].resample('YS').sum(),

'Currency Crises': crisis\_data['Currency Crises'].resample('YS').sum(),

'Inflation Crises': crisis\_data['Inflation Crises'].resample('YS').sum()

}

# Placeholder SARIMA orders (you should determine these from your analysis)

sarima\_orders = {

'Banking Crisis ': {'order': (1, 1, 1), 'seasonal\_order': (1, 1, 1, 12)},

'Systemic Crisis': {'order': (1, 1, 1), 'seasonal\_order': (1, 1, 1, 12)},

'Currency Crises': {'order': (1, 1, 1), 'seasonal\_order': (1, 1, 1, 12)},

'Inflation Crises': {'order': (1, 1, 1), 'seasonal\_order': (1, 1, 1, 12)}

}

# Forecasting up to the year 2100 for each crisis type

for crisis\_type, params in sarima\_orders.items():

print(f"Processing {crisis\_type}...")

series = crisis\_data\_aggregated[crisis\_type]

# Fit the SARIMA model

model = SARIMAX(series, order=params['order'], seasonal\_order=params['seasonal\_order'])

results = model.fit()

# Calculate the number of steps to forecast until the year 2100

last\_year = series.index[-1].year

steps\_to\_2050 = 2050 - last\_year

# Multi-step forecast

future\_years = pd.date\_range(start=series.index[-1] + pd.offsets.DateOffset(years=1),

end=pd.Timestamp(year=2050, month=1, day=1),

freq='YS')

forecast\_values = results.get\_forecast(steps=len(future\_years)).predicted\_mean

forecast\_values.index = future\_years

# Plotting the forecast

plt.figure(figsize=(15, 3))

plt.plot(series.index, series, label='Observed')

plt.plot(forecast\_values.index, forecast\_values, label='Forecast', color='red')

plt.title(f"Multi-Step Forecast for {crisis\_type} up to 2050")

plt.legend()

plt.show()

import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import LSTM, Dense

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

# Select one crisis type for demonstration. Here, using 'Banking Crisis' as an example

data = crisis\_data\_aggregated['Banking Crisis '].values

data = data.reshape(-1, 1)

# Scale the data

scaler = MinMaxScaler(feature\_range=(0, 1))

data\_scaled = scaler.fit\_transform(data)

# Create the dataset for LSTM

def create\_dataset(dataset, look\_back=1):

X, Y = [], []

for i in range(len(dataset) - look\_back - 1):

a = dataset[i:(i + look\_back), 0]

X.append(a)

Y.append(dataset[i + look\_back, 0])

return np.array(X), np.array(Y)

# Split into train and test sets

look\_back = 1

train\_size = int(len(data\_scaled) \* 0.67)

test\_size = len(data\_scaled) - train\_size

train, test = data\_scaled[0:train\_size,:], data\_scaled[train\_size:len(data\_scaled),:]

X\_train, Y\_train = create\_dataset(train, look\_back)

X\_test, Y\_test = create\_dataset(test, look\_back)

# Reshape input to be [samples, time steps, features]

X\_train = np.reshape(X\_train, (X\_train.shape[0], 1, X\_train.shape[1]))

X\_test = np.reshape(X\_test, (X\_test.shape[0], 1, X\_test.shape[1]))

# Create and fit the LSTM model

model = Sequential()

model.add(LSTM(4, input\_shape=(1, look\_back)))

model.add(Dense(1))

model.compile(loss='mean\_squared\_error', optimizer='adam')

model.fit(X\_train, Y\_train, epochs=100, batch\_size=1, verbose=2)

# Make predictions

train\_predict = model.predict(X\_train)

test\_predict = model.predict(X\_test)

# Invert predictions

train\_predict = scaler.inverse\_transform(train\_predict)

Y\_train = scaler.inverse\_transform([Y\_train])

test\_predict = scaler.inverse\_transform(test\_predict)

Y\_test = scaler.inverse\_transform([Y\_test])

# Calculate root mean squared error

train\_score = np.sqrt(mean\_squared\_error(Y\_train[0], train\_predict[:,0]))

print('Banking Crisis - Train Score: %.2f RMSE' % (train\_score))

test\_score = np.sqrt(mean\_squared\_error(Y\_test[0], test\_predict[:,0]))

print('Banking Crisis - Test Score: %.2f RMSE' % (test\_score))

# Select 'Systemic Crisis' data

data = crisis\_data['Systemic Crisis'].values

data = data.reshape(-1, 1)

# Scale the data

scaler = MinMaxScaler(feature\_range=(0, 1))

data\_scaled = scaler.fit\_transform(data)

# Create the dataset for LSTM

def create\_dataset(dataset, look\_back=1):

X, Y = [], []

for i in range(len(dataset) - look\_back - 1):

a = dataset[i:(i + look\_back), 0]

X.append(a)

Y.append(dataset[i + look\_back, 0])

return np.array(X), np.array(Y)

# Split into train and test sets

look\_back = 1

train\_size = int(len(data\_scaled) \* 0.67)

test\_size = len(data\_scaled) - train\_size

train, test = data\_scaled[0:train\_size,:], data\_scaled[train\_size:len(data\_scaled),:]

X\_train, Y\_train = create\_dataset(train, look\_back)

X\_test, Y\_test = create\_dataset(test, look\_back)

# Reshape input to be [samples, time steps, features]

X\_train = np.reshape(X\_train, (X\_train.shape[0], 1, X\_train.shape[1]))

X\_test = np.reshape(X\_test, (X\_test.shape[0], 1, X\_test.shape[1]))

# Create and fit the LSTM model

model = Sequential()

model.add(LSTM(4, input\_shape=(1, look\_back)))

model.add(Dense(1))

model.compile(loss='mean\_squared\_error', optimizer='adam')

model.fit(X\_train, Y\_train, epochs=100, batch\_size=1, verbose=2)

# Make predictions

train\_predict = model.predict(X\_train)

test\_predict = model.predict(X\_test)

# Invert predictions

train\_predict = scaler.inverse\_transform(train\_predict)

Y\_train = scaler.inverse\_transform([Y\_train])

test\_predict = scaler.inverse\_transform(test\_predict)

Y\_test = scaler.inverse\_transform([Y\_test])

# Calculate root mean squared error

train\_score = np.sqrt(mean\_squared\_error(Y\_train[0], train\_predict[:,0]))

test\_score = np.sqrt(mean\_squared\_error(Y\_test[0], test\_predict[:,0]))

print('Systemic Crisis - Train Score: %.2f RMSE' % (train\_score))

print('Systemic Crisis - Test Score: %.2f RMSE' % (test\_score))

# Select 'Currency Crises' data

data = crisis\_data['Currency Crises'].values

data = data.reshape(-1, 1)

# Scale the data

scaler = MinMaxScaler(feature\_range=(0, 1))

data\_scaled = scaler.fit\_transform(data)

# Create the dataset for LSTM

def create\_dataset(dataset, look\_back=1):

X, Y = [], []

for i in range(len(dataset) - look\_back - 1):

a = dataset[i:(i + look\_back), 0]

X.append(a)

Y.append(dataset[i + look\_back, 0])

return np.array(X), np.array(Y)

# Split into train and test sets

look\_back = 1

train\_size = int(len(data\_scaled) \* 0.67)

test\_size = len(data\_scaled) - train\_size

train, test = data\_scaled[0:train\_size,:], data\_scaled[train\_size:len(data\_scaled),:]

X\_train, Y\_train = create\_dataset(train, look\_back)

X\_test, Y\_test = create\_dataset(test, look\_back)

# Reshape input to be [samples, time steps, features]

X\_train = np.reshape(X\_train, (X\_train.shape[0], 1, X\_train.shape[1]))

X\_test = np.reshape(X\_test, (X\_test.shape[0], 1, X\_test.shape[1]))

# Create and fit the LSTM model

model = Sequential()

model.add(LSTM(4, input\_shape=(1, look\_back)))

model.add(Dense(1))

model.compile(loss='mean\_squared\_error', optimizer='adam')

model.fit(X\_train, Y\_train, epochs=100, batch\_size=1, verbose=2)

# Make predictions

train\_predict = model.predict(X\_train)

test\_predict = model.predict(X\_test)

# Invert predictions

train\_predict = scaler.inverse\_transform(train\_predict)

Y\_train = scaler.inverse\_transform([Y\_train])

test\_predict = scaler.inverse\_transform(test\_predict)

Y\_test = scaler.inverse\_transform([Y\_test])

# Calculate root mean squared error

train\_score = np.sqrt(mean\_squared\_error(Y\_train[0], train\_predict[:,0]))

test\_score = np.sqrt(mean\_squared\_error(Y\_test[0], test\_predict[:,0]))

print('Currency Crises - Train Score: %.2f RMSE' % (train\_score))

print('Currency Crises - Test Score: %.2f RMSE' % (test\_score))

# Select 'Inflation Crises' data

data = crisis\_data['Inflation Crises'].values

data = data.reshape(-1, 1)

# Scale the data

scaler = MinMaxScaler(feature\_range=(0, 1))

data\_scaled = scaler.fit\_transform(data)

# Create the dataset for LSTM

def create\_dataset(dataset, look\_back=1):

X, Y = [], []

for i in range(len(dataset) - look\_back - 1):

a = dataset[i:(i + look\_back), 0]

X.append(a)

Y.append(dataset[i + look\_back, 0])

return np.array(X), np.array(Y)

# Split into train and test sets

look\_back = 1

train\_size = int(len(data\_scaled) \* 0.67)

test\_size = len(data\_scaled) - train\_size

train, test = data\_scaled[0:train\_size,:], data\_scaled[train\_size:len(data\_scaled),:]

X\_train, Y\_train = create\_dataset(train, look\_back)

X\_test, Y\_test = create\_dataset(test, look\_back)

# Reshape input to be [samples, time steps, features]

X\_train = np.reshape(X\_train, (X\_train.shape[0], 1, X\_train.shape[1]))

X\_test = np.reshape(X\_test, (X\_test.shape[0], 1, X\_test.shape[1]))

# Create and fit the LSTM model

model = Sequential()

model.add(LSTM(4, input\_shape=(1, look\_back)))

model.add(Dense(1))

model.compile(loss='mean\_squared\_error', optimizer='adam')

model.fit(X\_train, Y\_train, epochs=100, batch\_size=1, verbose=2)

# Make predictions

train\_predict = model.predict(X\_train)

test\_predict = model.predict(X\_test)

# Invert predictions

train\_predict = scaler.inverse\_transform(train\_predict)

Y\_train = scaler.inverse\_transform([Y\_train])

test\_predict = scaler.inverse\_transform(test\_predict)

Y\_test = scaler.inverse\_transform([Y\_test])

# Calculate root mean squared error

train\_score = np.sqrt(mean\_squared\_error(Y\_train[0], train\_predict[:,0]))

test\_score = np.sqrt(mean\_squared\_error(Y\_test[0], test\_predict[:,0]))

print('Inflation Crises - Train Score: %.2f RMSE' % (train\_score))

print('Inflation Crises - Test Score: %.2f RMSE' % (test\_score))

import numpy as np

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

# Load and preprocess the dataset

crisis\_data = pd.read\_csv(r"C:\Users\dhars\Desktop\TRIMESTER 1\Z-Major Project T1\Dataset\20160923\_global\_crisis\_data.csv")

# Selecting relevant columns

features = crisis\_data[['Banking Crisis ', 'Systemic Crisis', 'Currency Crises', 'Inflation Crises']]

features=features.dropna()

features.head(20)

# Now scale the features

# Scaling the features

scaler = MinMaxScaler(feature\_range=(0, 1))

scaled\_features = scaler.fit\_transform(features)

# Function to create dataset for LSTM

def create\_dataset(dataset, look\_back=1):

X, Y = [], []

for i in range(len(dataset) - look\_back):

a = dataset[i:(i + look\_back), :]

X.append(a)

Y.append(dataset[i + look\_back, :])

return np.array(X), np.array(Y)

# Create dataset for LSTM

look\_back = 10

X, y = create\_dataset(scaled\_features, look\_back)

X

# Building the LSTM model

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=(look\_back, X.shape[2])))

model.add(LSTM(units=50,return\_sequences=False))

model.add(Dense(4)) # Output layer with 4 units for 4 features

# Compiling the LSTM

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Fitting the LSTM to the dataset

model.fit(X, y, epochs=10,batch\_size=228)

import numpy as np

import pandas as pd

from keras.models import Sequential

from keras.layers import LSTM, Dense

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

# Assuming 'X' and 'y' are already defined and preprocessed

# X should be a 3D array for LSTM (samples, time steps, features)

# y should be a 2D array (samples, crisis types)

# Define the LSTM model architecture

look\_back = X.shape[1] # This is the number of time steps you're considering for each sample

n\_features = X.shape[2] # This is the number of features in each time step

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=(look\_back, n\_features)))

model.add(LSTM(units=50, return\_sequences=False))

model.add(Dense(4)) # Assuming you have 4 types of crises as the target variables

# Compile the LSTM model

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Fit the LSTM model to the dataset

model.fit(X, y, epochs=10, batch\_size=228)

# Predict on the dataset (assuming you want to predict on the entire set for visualization)

predictions = model.predict(X)

# Evaluate the model

mse = mean\_squared\_error(y, predictions, multioutput='raw\_values')

rmse = np.sqrt(mse)

print("RMSE for each crisis type:")

print(rmse)

# Create 8 subplots for the actual and predicted values of each crisis type

fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(14, 18), constrained\_layout=True)

fig.suptitle('Predictions vs Actual Values for Each Crisis Type')

# Names of the crisis types for titles

crisis\_types = ['Banking Crisis', 'Systemic Crisis', 'Currency Crises', 'Inflation Crises']

for i in range(4):

# Actual values plot

axes[i, 0].plot(y[:, i], label='Actual Values', color='blue')

axes[i, 0].set\_title(f'Actual Values - {crisis\_types[i]}')

axes[i, 0].legend()

# Predicted values plot

axes[i, 1].plot(predictions[:, i], label='Predictions', color='orange')

axes[i, 1].set\_title(f'Predicted Values - {crisis\_types[i]}')

axes[i, 1].legend()

plt.show()

import numpy as np

import pandas as pd

from keras.models import Sequential

from keras.layers import LSTM, Dense

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

# Assuming 'X' and 'y' are already defined and preprocessed

# X should be a 3D array for LSTM (samples, time steps, features)

# y should be a 2D array (samples, crisis types)

# Define the LSTM model architecture

look\_back = X.shape[1] # This is the number of time steps you're considering for each sample

n\_features = X.shape[2] # This is the number of features in each time step

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=(look\_back, n\_features)))

model.add(LSTM(units=50, return\_sequences=False))

model.add(Dense(4)) # Assuming you have 4 types of crises as the target variables

# Compile the LSTM model

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Fit the LSTM model to the dataset

model.fit(X, y, epochs=10, batch\_size=228)

# Predict on the dataset (assuming you want to predict on the entire set for visualization)

predictions = model.predict(X)

# Evaluate the model

mse = mean\_squared\_error(y, predictions, multioutput='raw\_values')

rmse = np.sqrt(mse)

print("RMSE for each crisis type:")

print(rmse)

# Create 4 subplots for the actual and predicted values of each crisis type

fig, axes = plt.subplots(nrows=4, ncols=1, figsize=(14, 18), constrained\_layout=True)

fig.suptitle('Predictions vs Actual Values for Each Crisis Type')

# Names of the crisis types for titles

crisis\_types = ['Banking Crisis', 'Systemic Crisis', 'Currency Crises', 'Inflation Crises']

for i in range(4):

# Actual values plot

axes[i].plot(y[:, i], label='Actual Values', color='blue')

# Predicted values plot

axes[i].plot(predictions[:, i], label='Predictions', color='orange')

axes[i].set\_title(f'{crisis\_types[i]}')

axes[i].legend()

plt.show()

import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import LSTM, Dense

from keras.callbacks import EarlyStopping

import warnings

warnings.filterwarnings('ignore')

# Load the dataset

data = pd.read\_csv(r"C:\Users\dhars\Desktop\TRIMESTER 1\Z-Major Project T1\Dataset\20160923\_global\_crisis\_data.csv")

# Filter the data for the years 2010 to 2016

data\_filtered = data[(data['Year'] >= 2010) & (data['Year'] <= 2016)]

# Replace NaN values with forward fill and then backward fill

data\_filtered.fillna(method='ffill', inplace=True)

data\_filtered.fillna(method='bfill', inplace=True)

# Randomly select a subset of this data for training

random\_sample = data\_filtered.sample(n=200, random\_state=42)

# Selecting relevant columns for prediction

predictors = ['exch\_usd', 'Inflation, Annual percentages of average consumer prices']

targets = ['Banking Crisis ', 'Systemic Crisis', 'Currency Crises', 'Inflation Crises']

# Check if all predictor columns are present in the data

for col in predictors:

if col not in random\_sample.columns:

raise ValueError(f"Column {col} not found in the data.")

# Normalize the input features

scaler = MinMaxScaler(feature\_range=(0, 1))

scaled\_predictors = scaler.fit\_transform(random\_sample[predictors])

# Reshape input for LSTM model

X = np.array(scaled\_predictors).reshape((random\_sample.shape[0], 1, len(predictors)))

y = random\_sample[targets]

# Split data into training and testing sets

train\_size = int(len(X) \* 0.8)

X\_train, X\_test = X[:train\_size], X[train\_size:]

y\_train, y\_test = y[:train\_size], y[train\_size:]

# LSTM model

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1], X\_train.shape[2])))

model.add(LSTM(units=50, return\_sequences=False))

model.add(Dense(units=25))

model.add(Dense(y\_train.shape[1]))

# Compile the model

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Early stopping

early\_stop = EarlyStopping(monitor='val\_loss', patience=10)

# Fit the model

model.fit(X\_train, y\_train, epochs=100, batch\_size=64, validation\_data=(X\_test, y\_test), callbacks=[early\_stop], verbose=1)

# Prepare future data for prediction (years 2017 to 2036)

future\_years = np.arange(2017, 2037)

scaled\_future\_years = scaler.transform(np.column\_stack((np.zeros(len(future\_years)), future\_years)))

X\_future = np.array(scaled\_future\_years).reshape((len(future\_years), 1, len(predictors)))

# Predict future crises

future\_predictions = model.predict(X\_future)

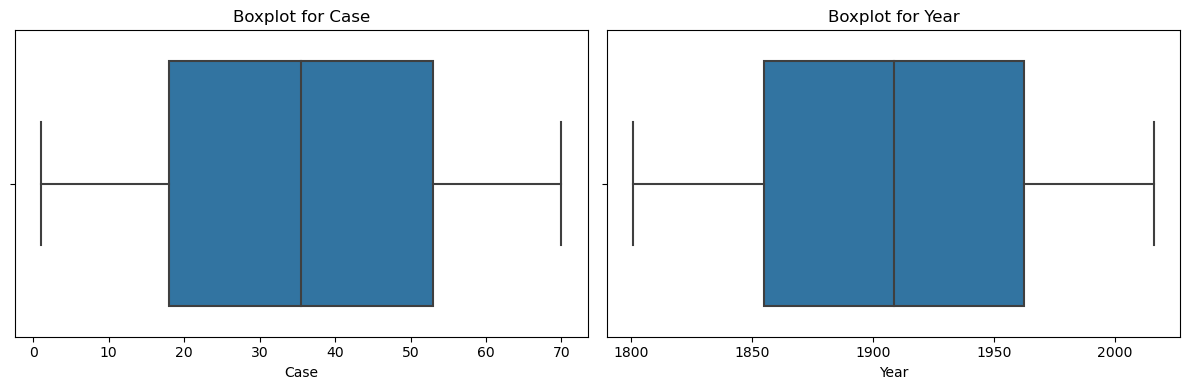
# Create a DataFrame for the predictions

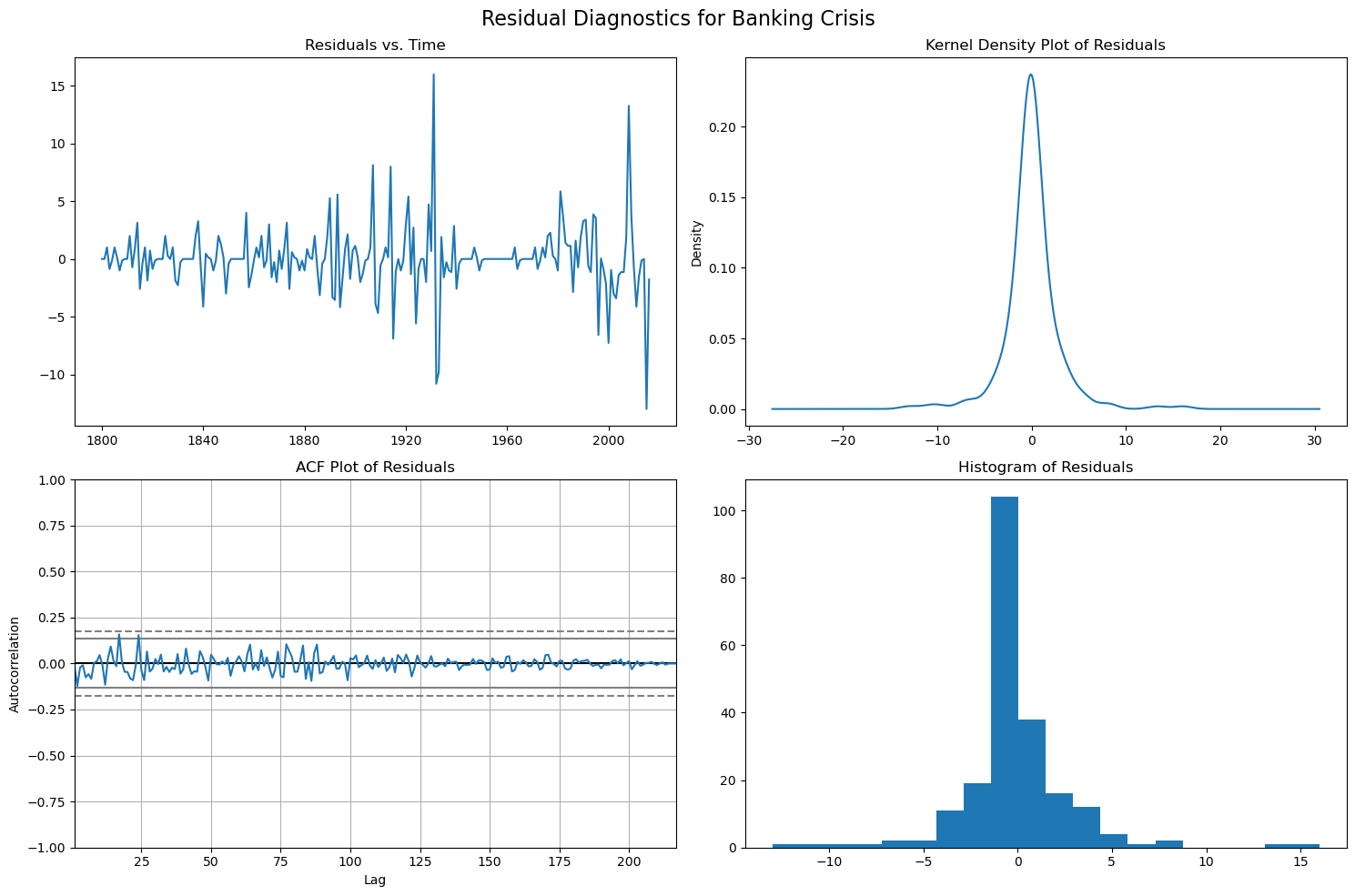
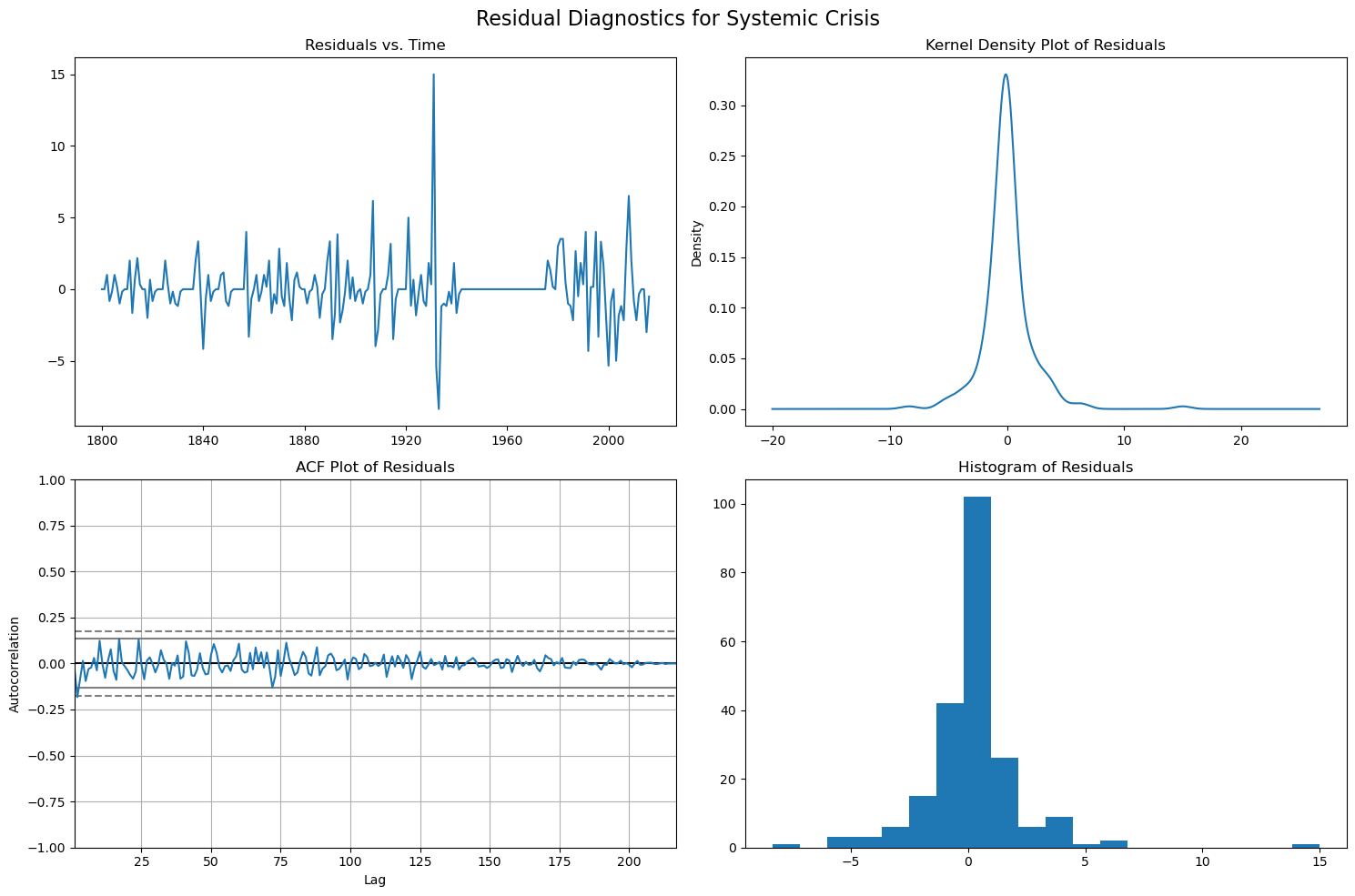
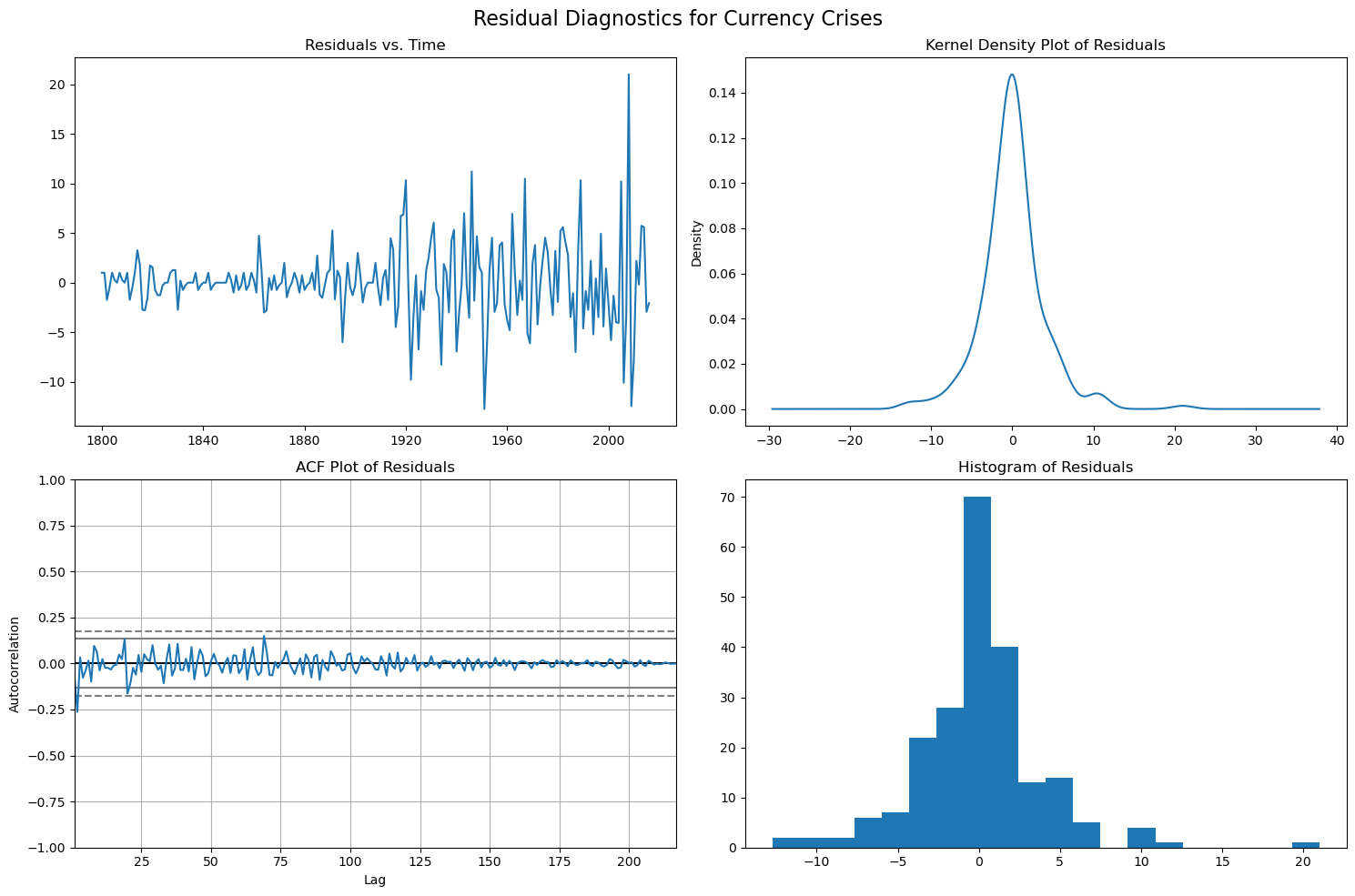
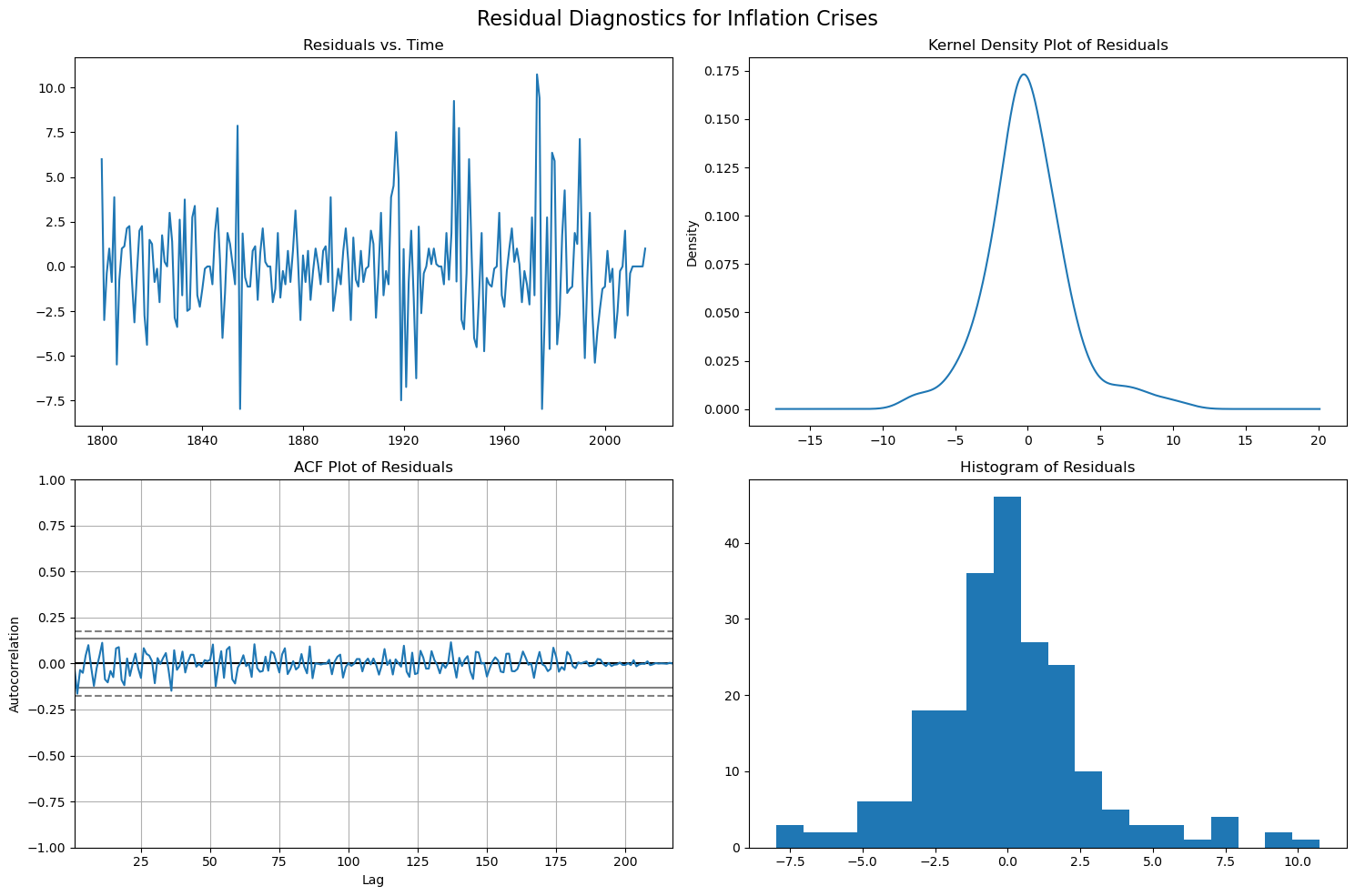
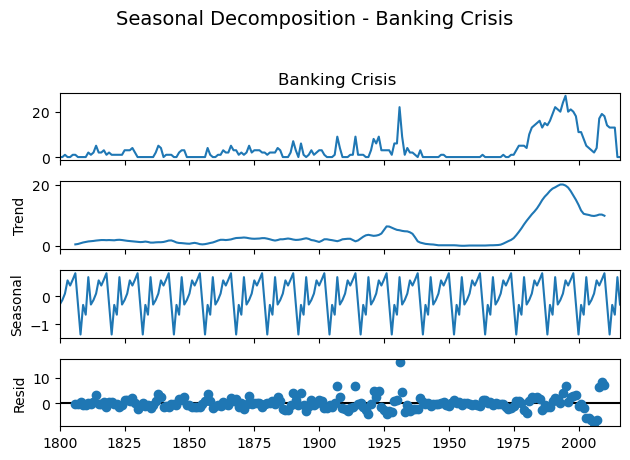
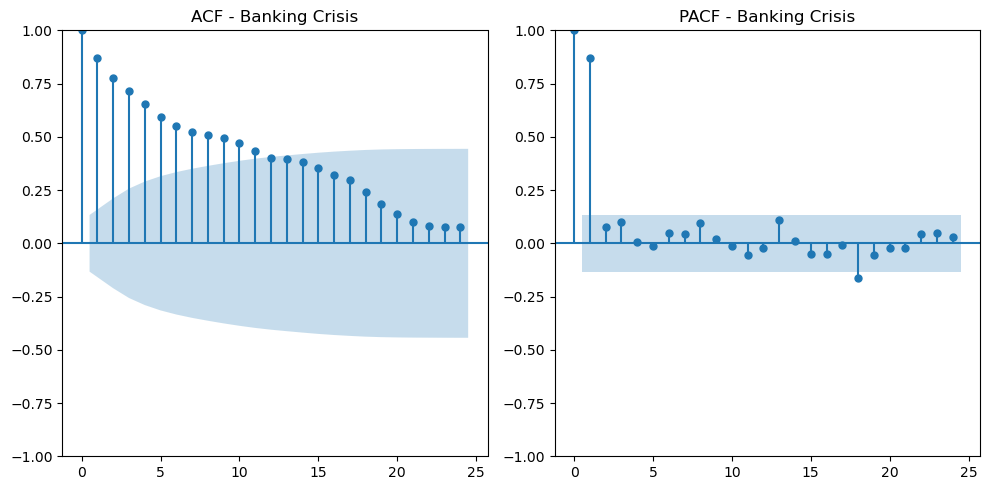
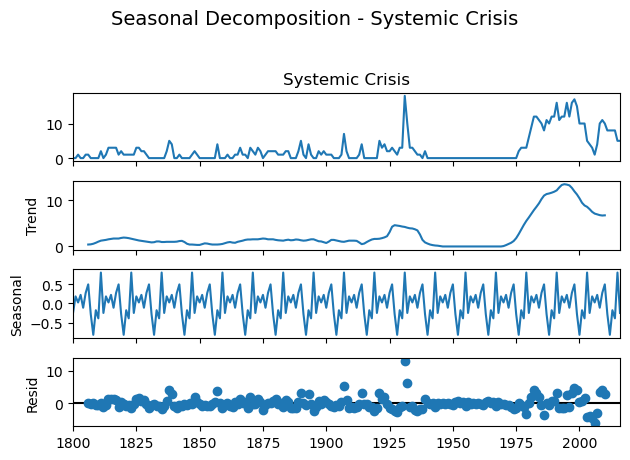
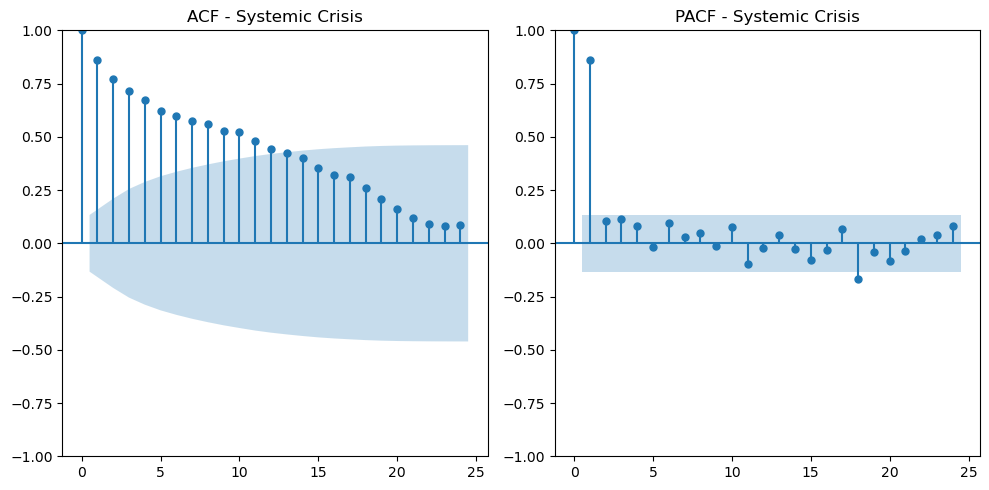
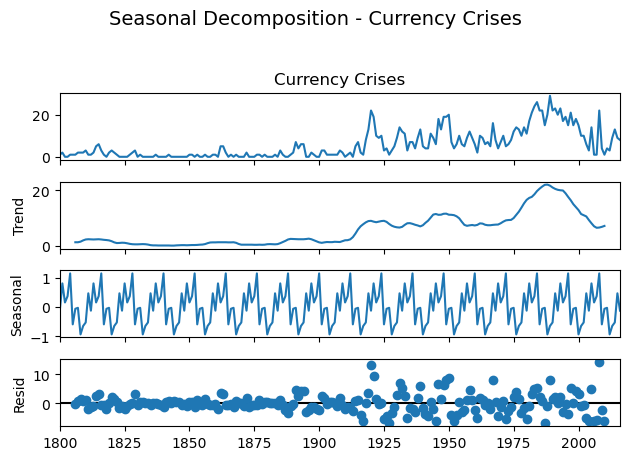
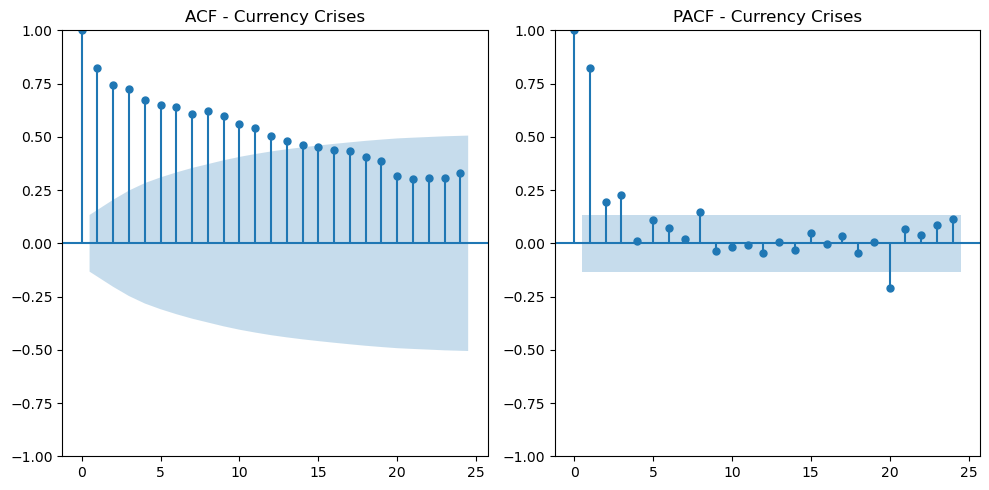
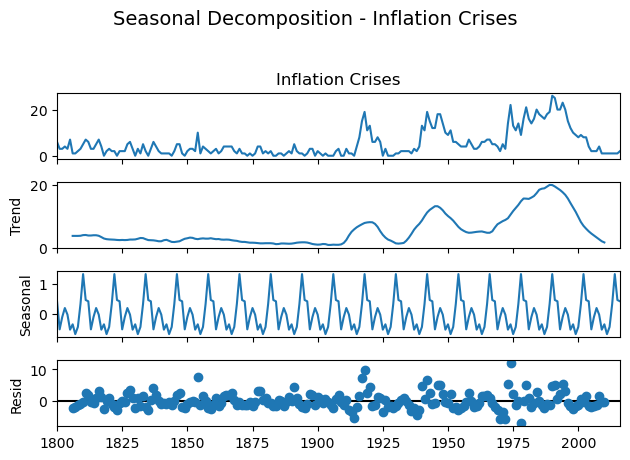
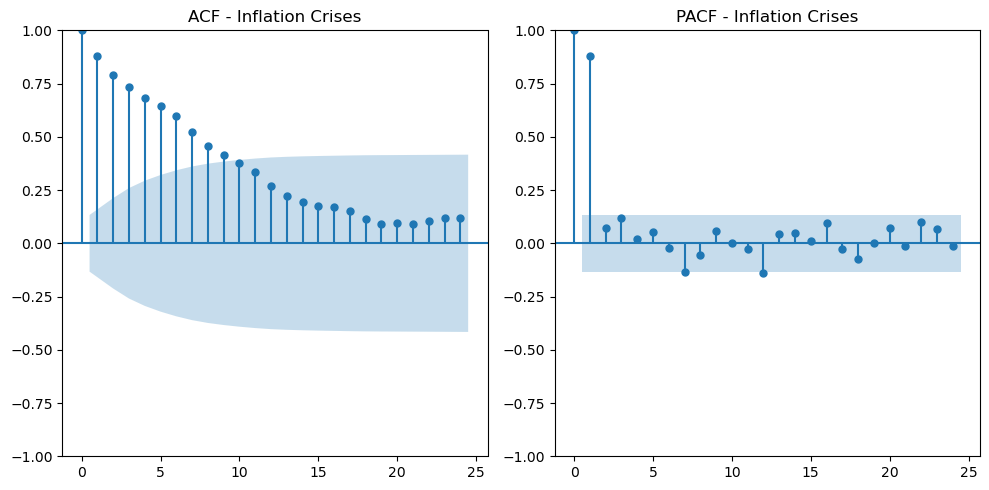
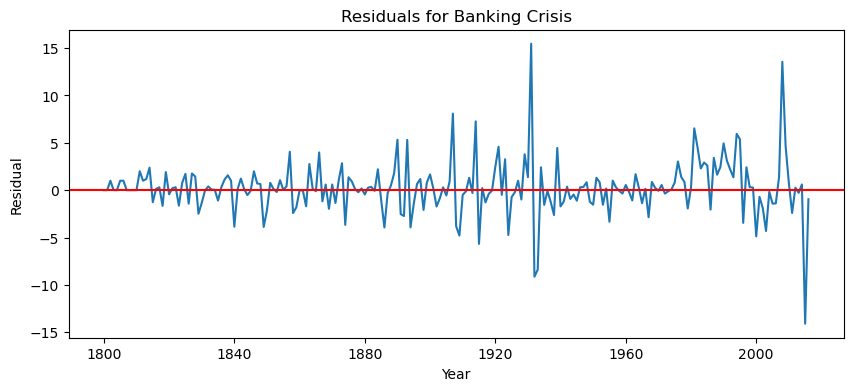
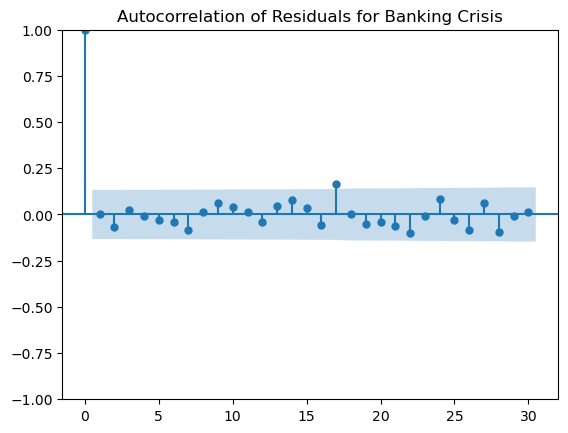
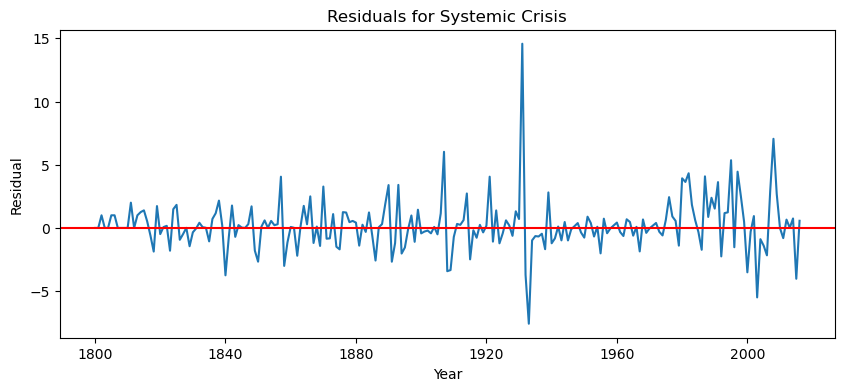
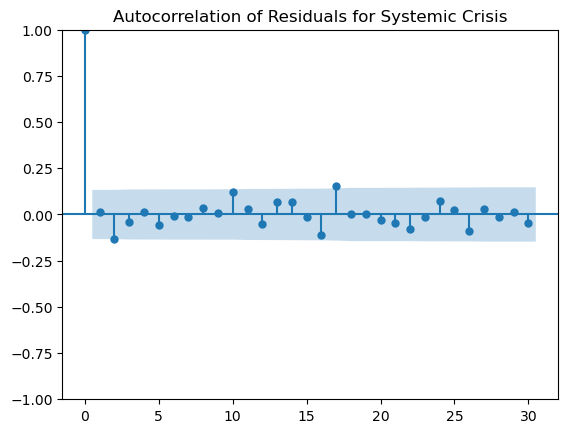
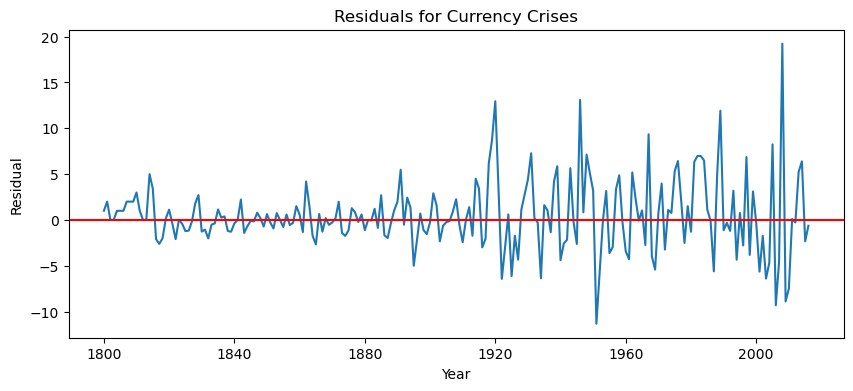
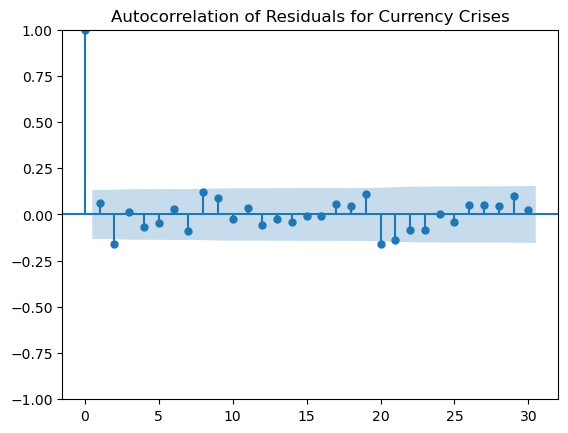
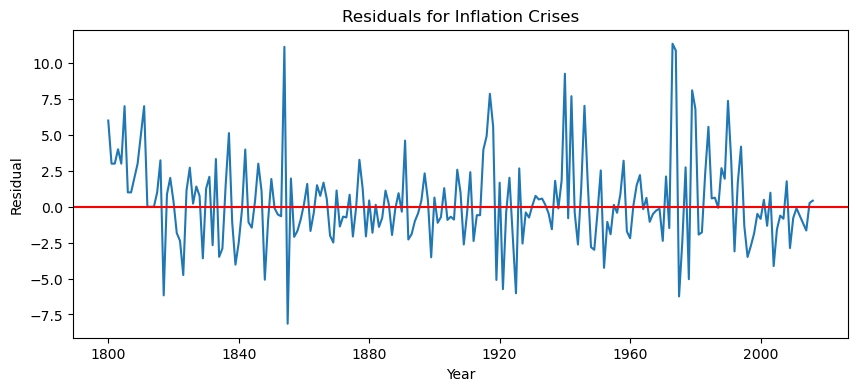
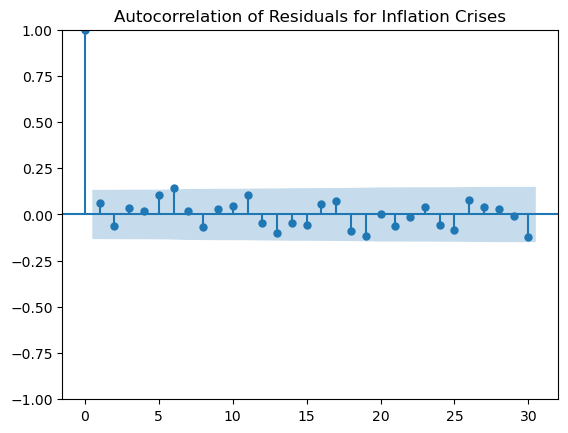
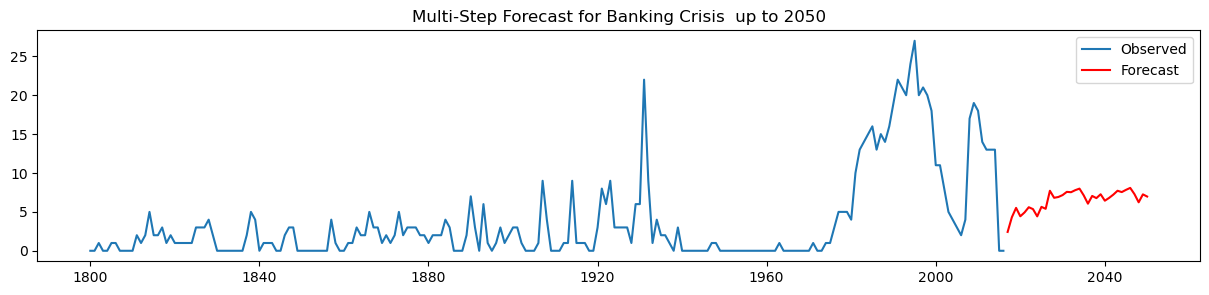
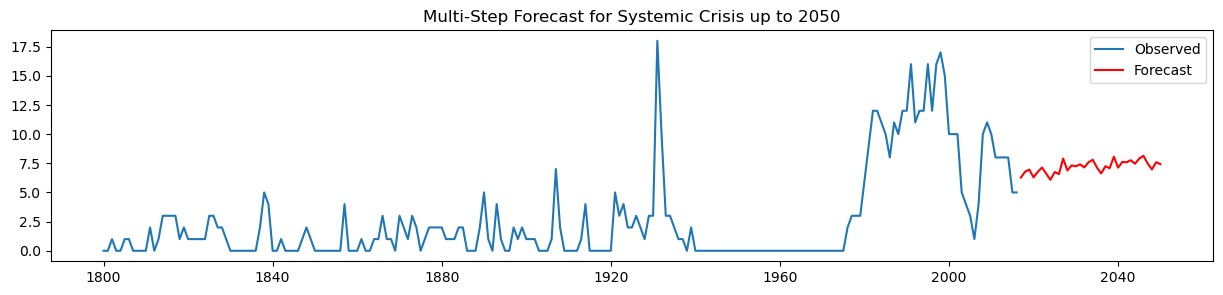
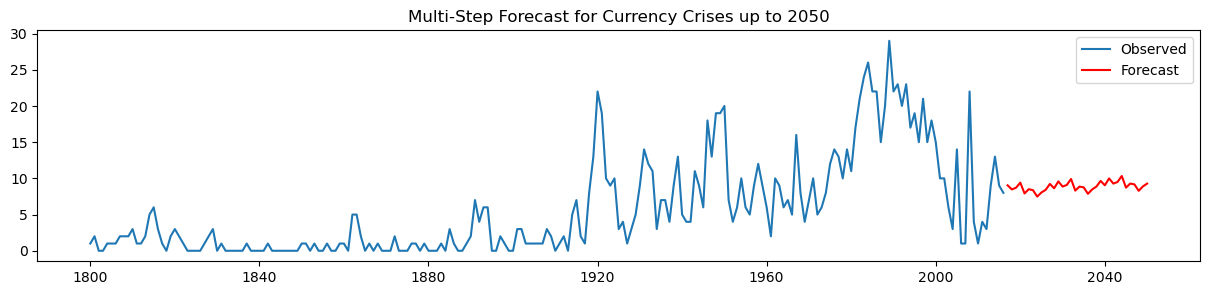
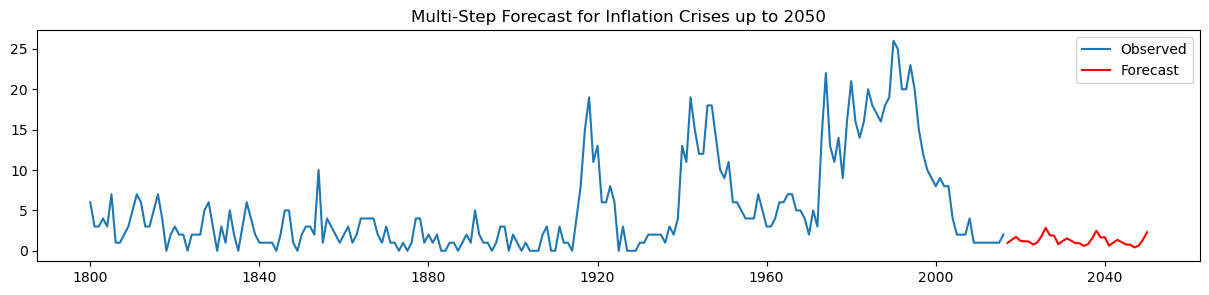
predictions\_df = pd.DataFrame(future\_predictions, columns=targets)

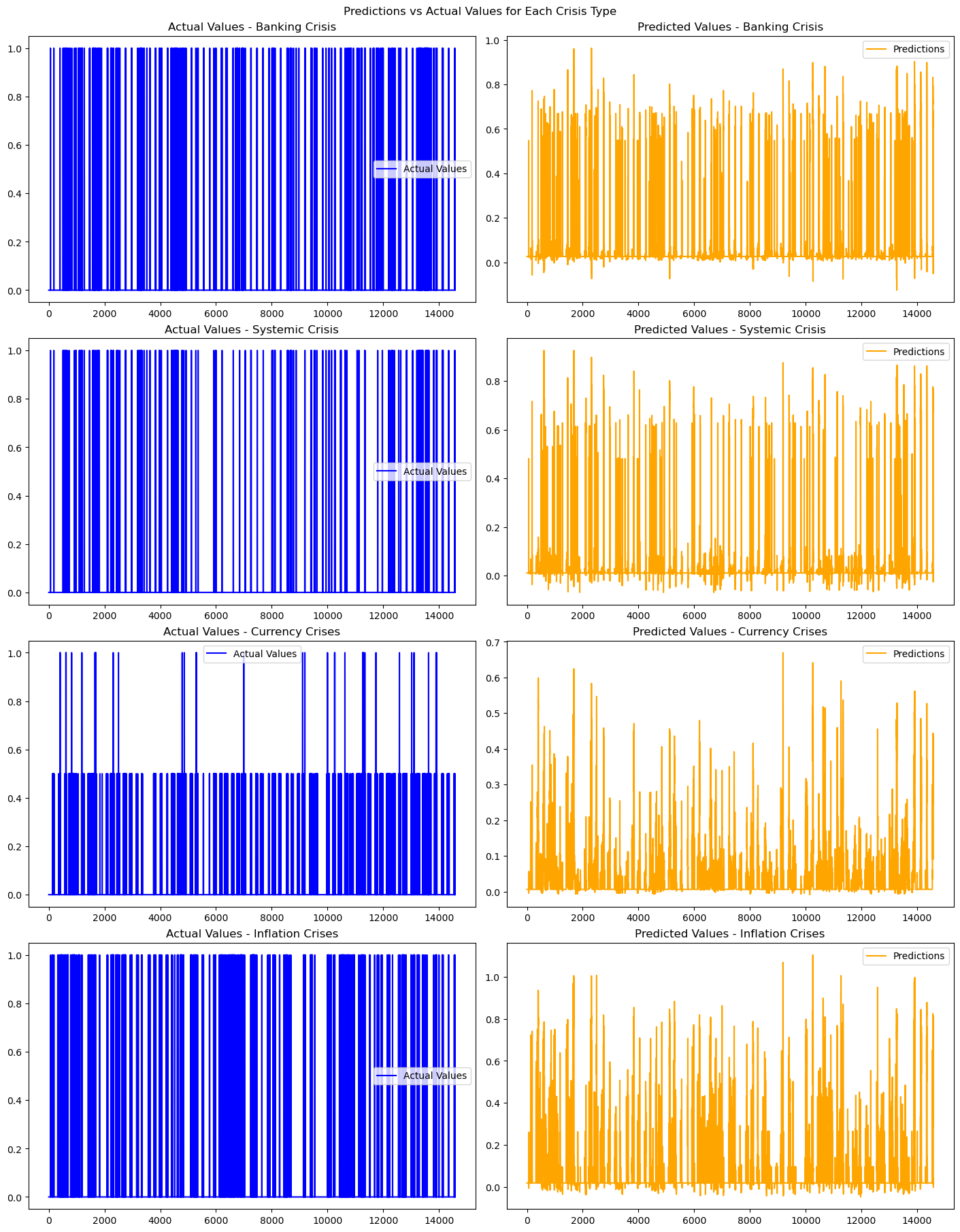
predictions\_df['Year'] = future\_years

print(predictions\_df)

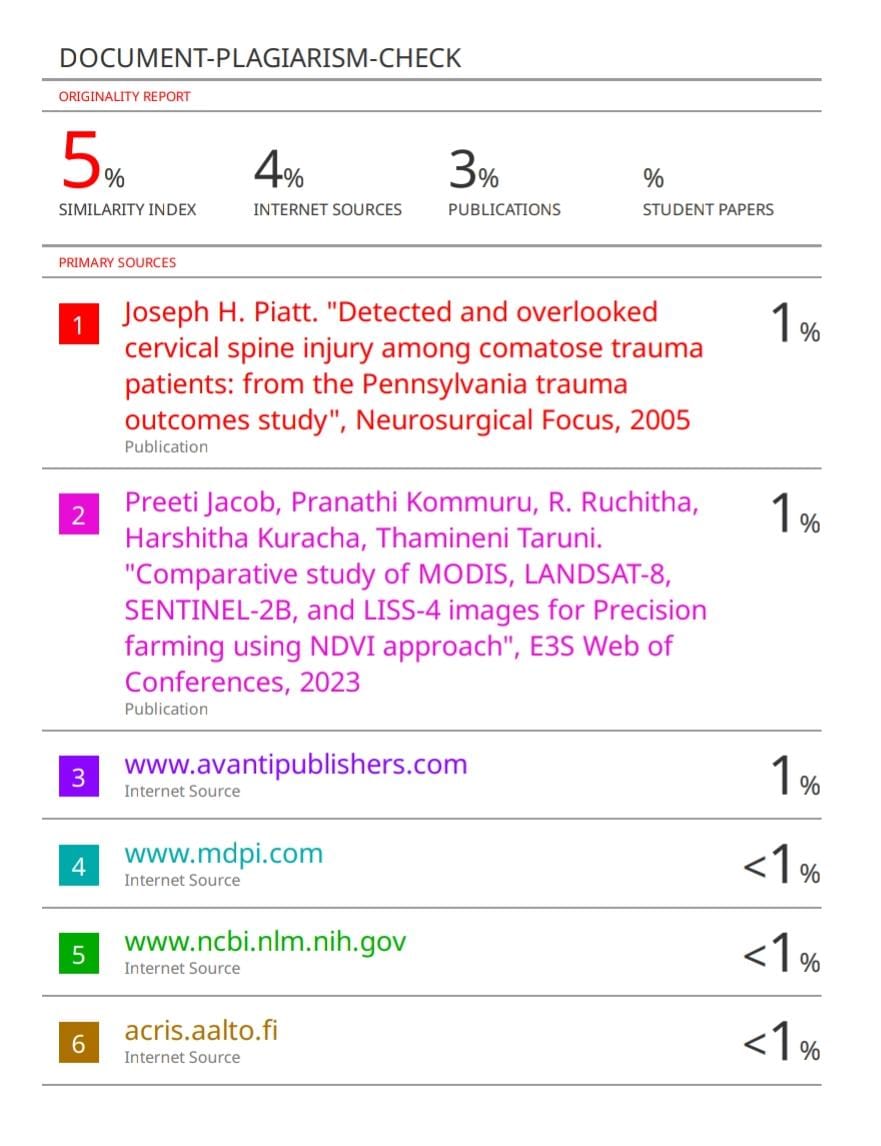
**SCREENSHOTS:**





**9.2 PLAGIARISM REPORT**



**10 REFERENCES**

[1] Harvard Business School. (2018). *2018 Financial Risk and Regulation Survey: Behavioral Finance & Financial Stability*.

[2] Tekin, H. (2020). *How optimal cash changed by the global financial crisis?*. Economics and Business Letters, 9(2), 114-123.

[3] World Bank. (2020). *Global Financial Development Report 2019/2020: Bank Regulation and Supervision a Decade after the Global Financial Crisis*.

[4] Lajili, K., Li, T. M., Chourou, L., Dobler, M., & Zéghal, D. (2023). *Corporate risk disclosures in turbulent times: An international analysis in the global financial crisis*. Journal of International Financial Management & Accounting.

[5] Shehzad, K., Xiaoxing, L., & Kazouz, H. (2020). *COVID-19’s disasters are perilous than Global Financial Crisis: A rumor or fact?*. Finance Research Letters, 36.

[6] Lolea, I.-C., Petrariu, I.-R., & Giurgiu, A. (2021). *ARIMA vs. MACHINE LEARNING IN TERMS OF EQUITY MARKET FORECASTING*. The Annals of the University of Oradea. Economic Sciences, XXX(2), 299-308.

[7] Tang, Y. M., Chau, K.-Y., Li, W., & Wan, T. W. (2020). *Forecasting Economic Recession through Share Price in the Logistics Industry with Artificial Intelligence (AI)*. Computation, 8(3).

[8] Liu, J., Yan, R., & Zhang, Y. (2023). *Stock Market Forecasting Using the ARIMA GARCH and Random Forest Model During The Russia–Ukraine War*. Applied Economics and Policy Studies book series (AEPS), Springer.

[9] Zulkefly, R. N., Karim, A. A., Rahman, A. A., & Sarmidi, T. (2020). *Financial inclusiveness and economic growth: New evidence using a threshold regression analysis*. Economic Research-Ekonomska Istraživanja, 33(1), 1465-1484.

[10] Hossain, M. J. (2023). *Comparative Analysis of ARIMA SARIMAX and Random Forest Models for Forecasting Future GDP in Relation to Unemployment Rate*.

**11 WORKLOG**

