

Hans Hookoomsing

M00839393

2024 June 21

CST3990

Undergraduate Individual Project

Final Report

Suraj Juddoo (Supervisor)

# Abstract

Table of Contents

[Abstract 2](#_Toc168920885)

[Acknowledgements 3](#_Toc168920886)

[Introduction 3](#_Toc168920887)

[Literature Review 3](#_Toc168920888)

[Requirements Specifications 3](#_Toc168920889)

[Analysis & Designs 3](#_Toc168920890)

[Implementation 3](#_Toc168920891)

[Data Processing 3](#_Toc168920892)

[Obtain stock data and sort by date 5](#_Toc168920893)

[Obtain Macroeconomic Data 6](#_Toc168920894)

[Merge the Stock Data and Macroeconomic Data 7](#_Toc168920895)

[Standardize the data using RobustScaler 10](#_Toc168920896)

[Create Sequences and Split the data into features and labels 11](#_Toc168920897)

[Split the data into training, validating and testing sets 13](#_Toc168920898)

[The purpose of the different Data processing steps 15](#_Toc168920899)

[Testing & Results 16](#_Toc168920900)

[Testing the Collected Data 16](#_Toc168920901)

[Code, explanation and test results for test performed on stock dataset 16](#_Toc168920902)

[Code, explanation and test results for test performed on Macroeconomic dataset 19](#_Toc168920903)

[Code, explanation and test results for the test performed on the standardized dataset 20](#_Toc168920904)

[Code, explanation and test results for the test performed after sequencing 25](#_Toc168920905)

[Code, explanation and test results for the test performed after the train-val-test split 25](#_Toc168920906)

[Evaluation 25](#_Toc168920907)

[Conclusion 25](#_Toc168920908)

[User Guide 25](#_Toc168920909)

[Bibliography 25](#_Toc168920910)

[Appendix 26](#_Toc168920911)

[Link to Macroeconomic datasets 26](#_Toc168920912)

# Acknowledgements

# Introduction

# Literature Review

# Requirements Specifications

# Analysis & Designs

# Implementation

## Data Processing

For this project, data was collected from two main sources, the Federal Reserve Bank of St. Louis where macroeconomic data was obtained using the FRED API (*St. Louis Fed Web Services | St. Louis Fed*, no date) and from Yahoo Finance where stock data of 8 companies operating in different sectors of the economy were obtained. Those corporations, their respective industries and exchanges are listed in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Index** | **Ticker** | **Full Name** | **Exchange** | **Industry** |
| 1 | F | Ford Motor Company | |  | | --- | | NYSE | | Automotive |
| 2 | JPM | JPMorgan Chase & Co | NYSE | Commercial/Investment Banking |
| 3 | MS | Morgan Stanley | NYSE | Investment Banking |
| 4 | WFC | Wells Fargo & Company | NYSE | Technology |
| 5 | AAPL | Apple Inc | NASDAQ | Technology |
| 6 | AMZN | Amazon.com Inc | NASDAQ | Retail/ Technology |
| 7 | MSFT | Microsoft Corporation | NASDAQ | Technology |
| 8 | CSCO | Cisco Systems Inc. | NASDAQ | Technology |

Table 1 List of Companies and Their Sectors

Furthermore, GDP figures, inflation and interest rates were obtained from the Federal Reserve Bank of St Louis and the table below provides a concise overview of the key information for each dataset used in your analysis. The GDP data is reported quarterly, while the Effective Federal Funds Rate (EFFR) is reported daily and the Sticky Price Consumer Price Index less Food and Energy (CORESTICKM159SFRBATL) is reported monthly. In addition, the links to those datasets are available in the Appendix section of this report.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset Name** | **Description** | **Frequency** | **Units** | **Additional Information** |
| |  | | --- | |  |  |  | | --- | | GDP |  |  | | --- | |  |  |  | | --- | |  | | Gross Domestic Product | |  | | --- | | Quarterly | | Billions of Dollars | Seasonally Adjusted Annual Rate |
| EFFR | Effective Federal Funds Rate | Daily | Percent | Not Seasonally Adjusted |
| CORESTICKM159SFRBATL | Sticky Price Consumer Price Index less Food and Energy | Monthly | Percent Change from a Year Ago | Not Seasonally Adjusted |

Table 2 List of Macroeconomic datasets and their characteristics

Data processing could be described as the manipulation and preparation of data in order to derive information or fulfil specific tasks (Talend, no date). In order to prepare the stock and macroeconomic data to be used in the LSTM model, the following steps have been undertaken using the Python programming language and libraries such as Pandas and Matplotlib, among others:

Obtain stock data and sort by date 

Figure 1 Code to download Stock data using the yfinance API

#### Code Explanation

|  |  |
| --- | --- |
| **Step** | **Description** |
| Directory Setup | Specifies the directory where the resulting CSV file will be saved. |
| Tickers List | Lists the stock tickers for companies on the NYSE and NASDAQ exchanges. |
| Date Range Specification | Defines the start and end dates for downloading stock data (January 4, 2001, to December 31, 2023). |
| Data Download | Uses the yfinance library to download the closing prices for the specified stocks over the defined date range. |
| Date Column Transformation | Converts the index to a column so that dates become part of the DataFrame columns. |
| Data Melting | Transforms the DataFrame from wide format (each stock's closing prices as columns) to long format (each row represents a stock's closing price on a specific date). |
| Data Sorting | Sorts the transformed DataFrame by date to ensure chronological order |
| Data Saving | Saves the sorted, transformed data to a CSV file in the specified directory |

Obtain Macroeconomic Data

Figure 2 Code to download Macroeconomic data using the FRED API

#### Code Explanation

|  |  |
| --- | --- |
| **Step** | **Description** |
| Initialize the Fred API | Initializes the Fred API using the provided API key ('9390c3b48fba8e99df33843071015842'). |
| Define Date Range | Specifies the start and end dates for fetching data from FRED (1999-12-01 to 2024-01-01). |
| Fetch Data from FRED | Retrieves three types of macroeconomic data from FRED: GDP ('GDP'), interest rates ('EFFR'), and inflation rates ('CORESTICKM159SFRBATL'). |
| Convert Data to DataFrames | Converts each series of fetched data into separate Pandas DataFrames and sets the index name to 'Date'. |
| Save Data to CSV Files | Saves each DataFrame to a CSV file in the specified output folder ('/Users/hanshookoomsing/Documents/undergraduate\_project/LSTM\_TensorFlow'). |

Merge the Stock Data and Macroeconomic Data

Figure 3 Part 1 of Code to merge stock and macroeconomic data

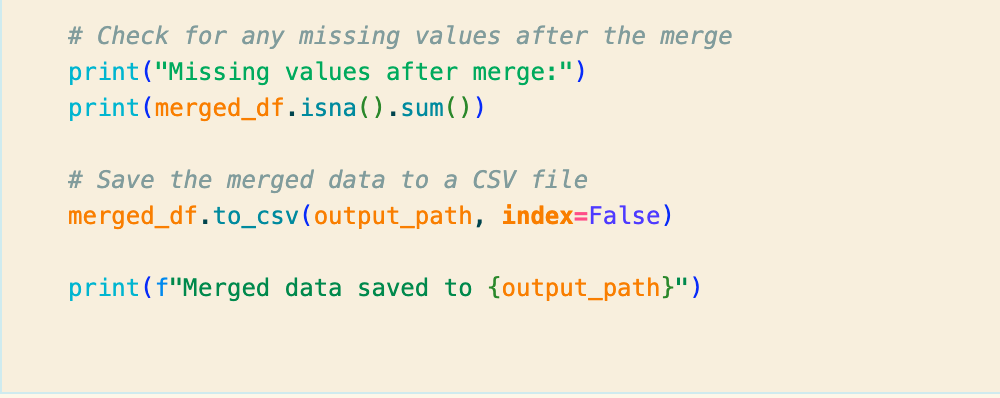


Figure 4 Part 2 of Code to merge stock and macroeconomic data

#### Code Explanation

|  |  |
| --- | --- |
| **Step** | **Description** |
| File Paths | Defines the file paths for the interest rates, inflation rates, and the output file for the merged data. |
| Load Datasets | Loads the interest rates and inflation rates data from CSV files into Pandas DataFrames. |
| Check DataFrames | Prints the first few rows of each DataFrame to understand their structure. |
| Forward Fill Inflation Rates | Sets the 'Date' column as the index for the inflation rates DataFrame, resamples it to daily frequency, and forward fills missing values. |
| Check Forward Filled Data | Prints the first few rows of the inflation rates DataFrame after forward filling to verify the changes |
| Merge DataFrames | Merges the interest rates and forward-filled inflation rates DataFrames on the 'Date' column. |
| Check for Missing Values | Checks for any missing values in the merged DataFrame after the merge. |
| Save Merged Data | Saves the merged DataFrame to a CSV file. |
| Print Confirmation | Prints a message confirming that the merged data has been saved. |

Table 3 Code Explanation



Figure 5 Part 3 where GDP data is linearly interpolate

#### Code Explanation

|  |  |
| --- | --- |
| **Step** | **Description** |
| File Paths | Defines the file paths for the input GDP data and the output file for the interpolated daily GDP data. |
| Load GDP Dataset | Loads the GDP dataset from a CSV file into a Pandas DataFrame |
| Check DataFrame Structure | Prints the first few rows of the GDP DataFrame to understand its structure |
| Resample and Interpolate GDP | Sets the 'Date' column as the index, resamples the GDP data from quarterly to daily frequency, and performs linear interpolation to fill in the gaps. |
| Check Interpolated Data | Prints the first few rows of the GDP DataFrame after resampling and interpolation to verify the changes |
| Save Interpolated Data | Saves the interpolated daily GDP data to a CSV file. |
| Print Confirmation | Prints a message confirming that the daily GDP data has been saved |

Table 4 Code Explanation

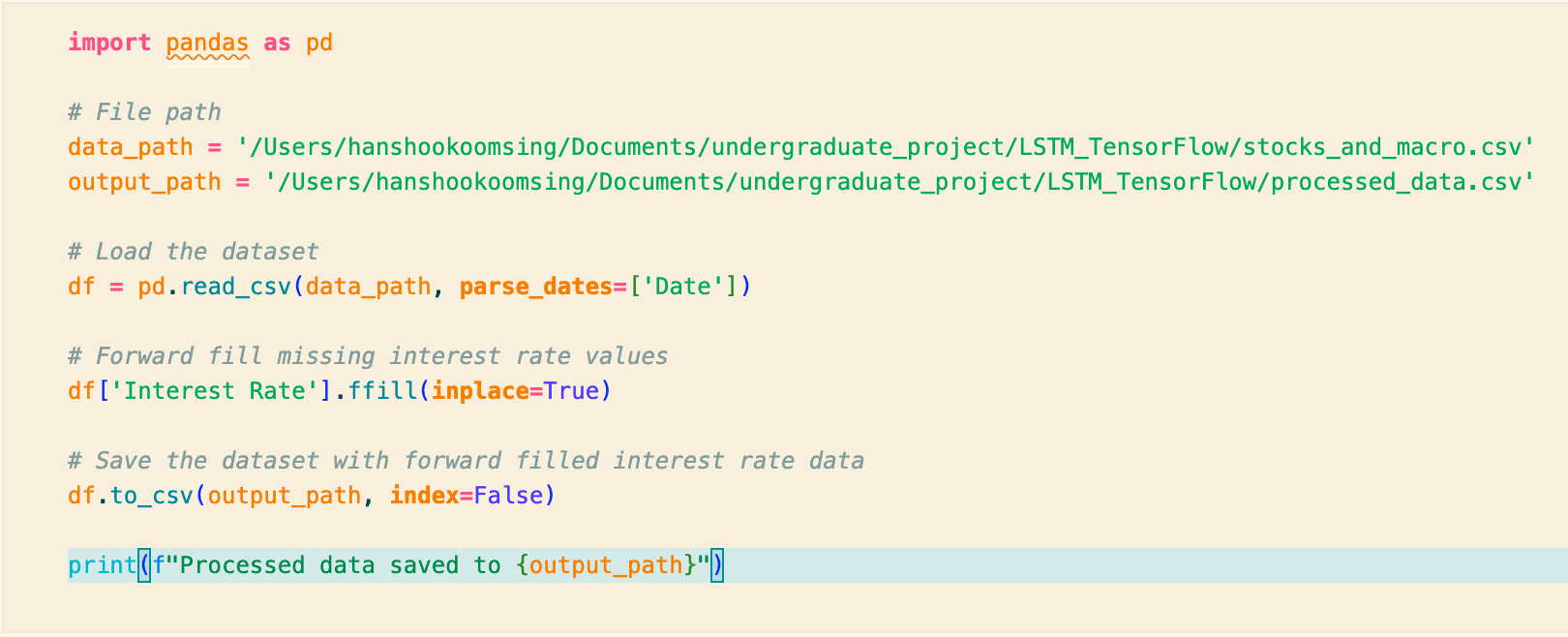


Figure 6 Code to forward-fill interest rate data

#### Code Explanation

|  |  |
| --- | --- |
| **Step** | **Description** |
| File Paths | Defines the file paths for the input data (stocks and macroeconomic data) and the output file for the processed data. |
| Load Dataset | Loads the combined dataset from a CSV file into a Pandas DataFrame. |
| Forward Fill Interest Rates | Forward fills missing values in the 'Interest Rate' column to ensure no gaps in the data. |
| Save Processed Data | Saves the processed DataFrame with forward-filled interest rate data to a CSV file. |
| Print Confirmation | Prints a message confirming that the processed data has been saved |

Table 5 Code Explanation

Standardize the data using RobustScaler

Figure 7 Code to standardise the dataset using RobustScaler

#### Code Explanation

|  |  |
| --- | --- |
| **Step** | **Description** |
| Load Dataset | Loads the dataset from the specified file path (/Users/hanshookoomsing/Documents/undergraduate\_project/LSTM\_TensorFlow/processed\_data.csv). |
| Separate Numeric Columns | Identifies and separates numeric columns from non-numeric columns in the dataset. |
| Initialize RobustScaler | Initializes the RobustScaler from the sklearn.preprocessing module. |
| Scale Numeric Data | Fits the RobustScaler on the numeric data and transforms it to scale the numeric values. |
| Convert to DataFrame | Converts the scaled numeric data back into a DataFrame with the same column names. |
| Combine Data | Combines the scaled numeric data with the non-numeric data into a single DataFrame. |
| Save Scaled Data | Saves the combined DataFrame to a new CSV file (/Users/hanshookoomsing/Documents/undergraduate\_project/LSTM\_TensorFlow/standardized.csv). |
| Display Scaled Data | Displays the first few rows of the scaled data DataFrame for verification. |

Table 6 Code Explanation

### Create Sequences and Split the data into features and labels



Figure 8 Part 1 of code to create sequences



Figure 9 Part 2 of code to create sequences

#### Code Explanation

|  |  |
| --- | --- |
| **Step** | **Description** |
| Load Dataset | Loads the standardized dataset from the specified file path (/Users/hanshookoomsing/Documents/undergraduate\_project/LSTM\_TensorFlow/standardized.csv). |
| Convert Date Column | Ensures the 'Date' column is converted to date format. |
| Sort Data by Date | Sorts the data by the 'Date' column to ensure chronological order. |
| Set Sequence Length | Defines the sequence length to be used for creating sequences (28 days). |
| Create Sequences Function | Defines a function create\_sequences that generates sequences, labels, and dates from the data. |
| Generate Sequences per Stock | Iterates over each unique stock in the dataset, creates sequences using the defined function, and stores them. |
| Combine Sequences | Combines sequences, labels, and dates from all stocks into single arrays (X, y, and dates). |
| Save Sequences and Labels | Saves the combined sequences, labels, and dates to .npy files in the specified directory. |
| Verify Saved Files | Prints a confirmation message indicating that the sequences, labels, and dates have been saved successfully. |

Table 7 Code Explanation

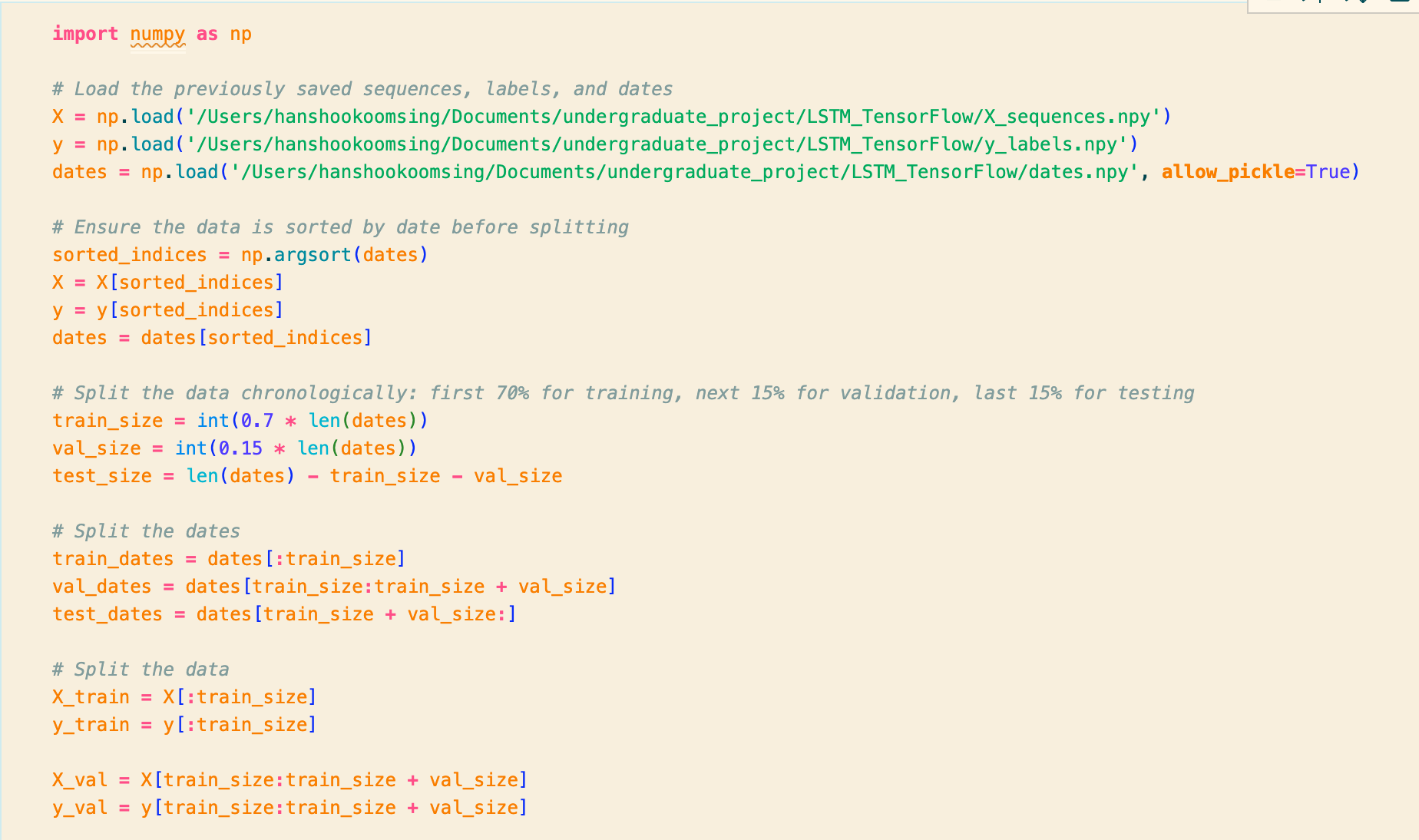
Split the data into training, validating and testing sets

Figure 10 Part 1 of code to split data into training, validation and test datasets

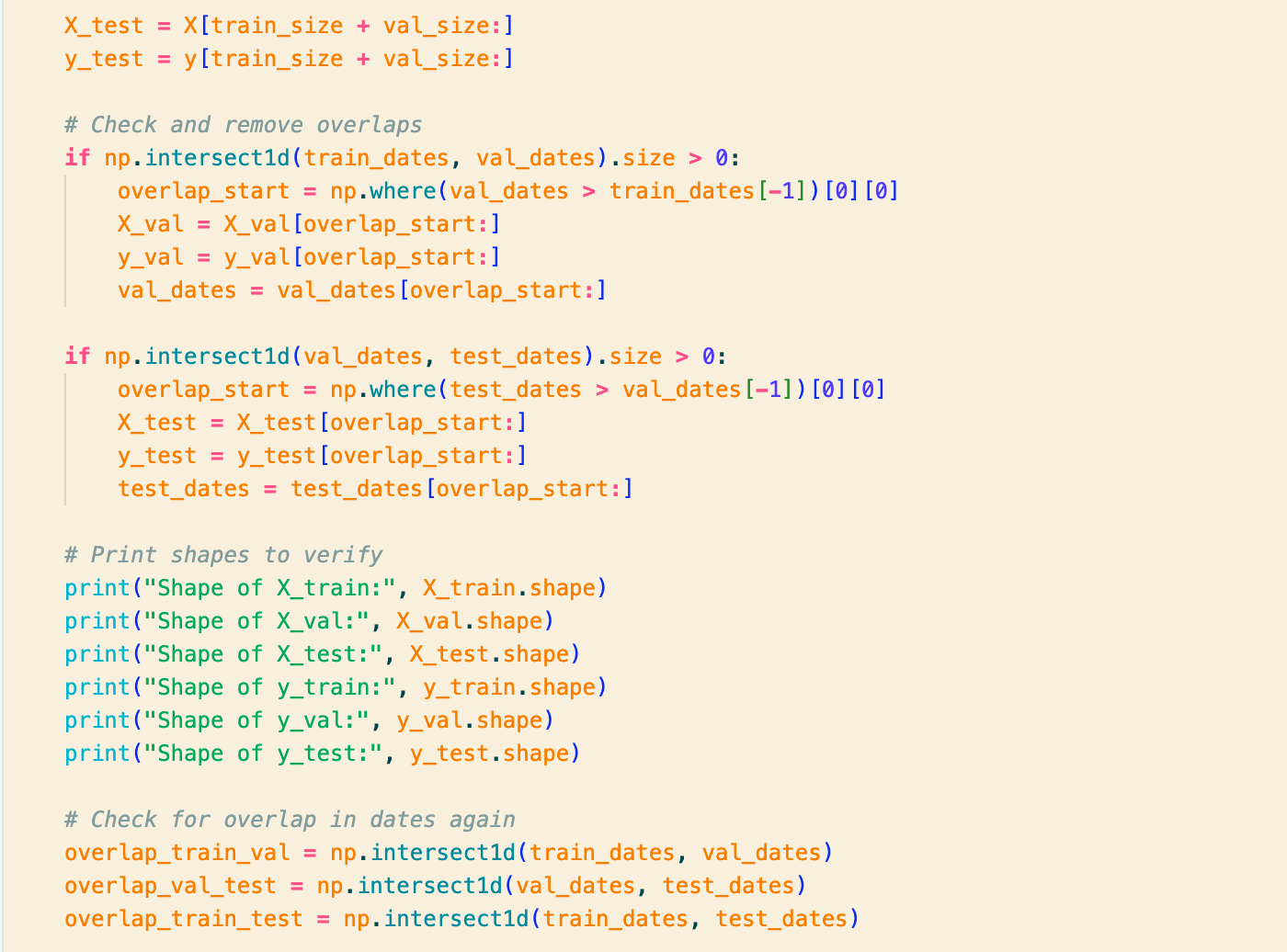


Figure 11 Part 2 of code to split data into training, validation and testing datasets

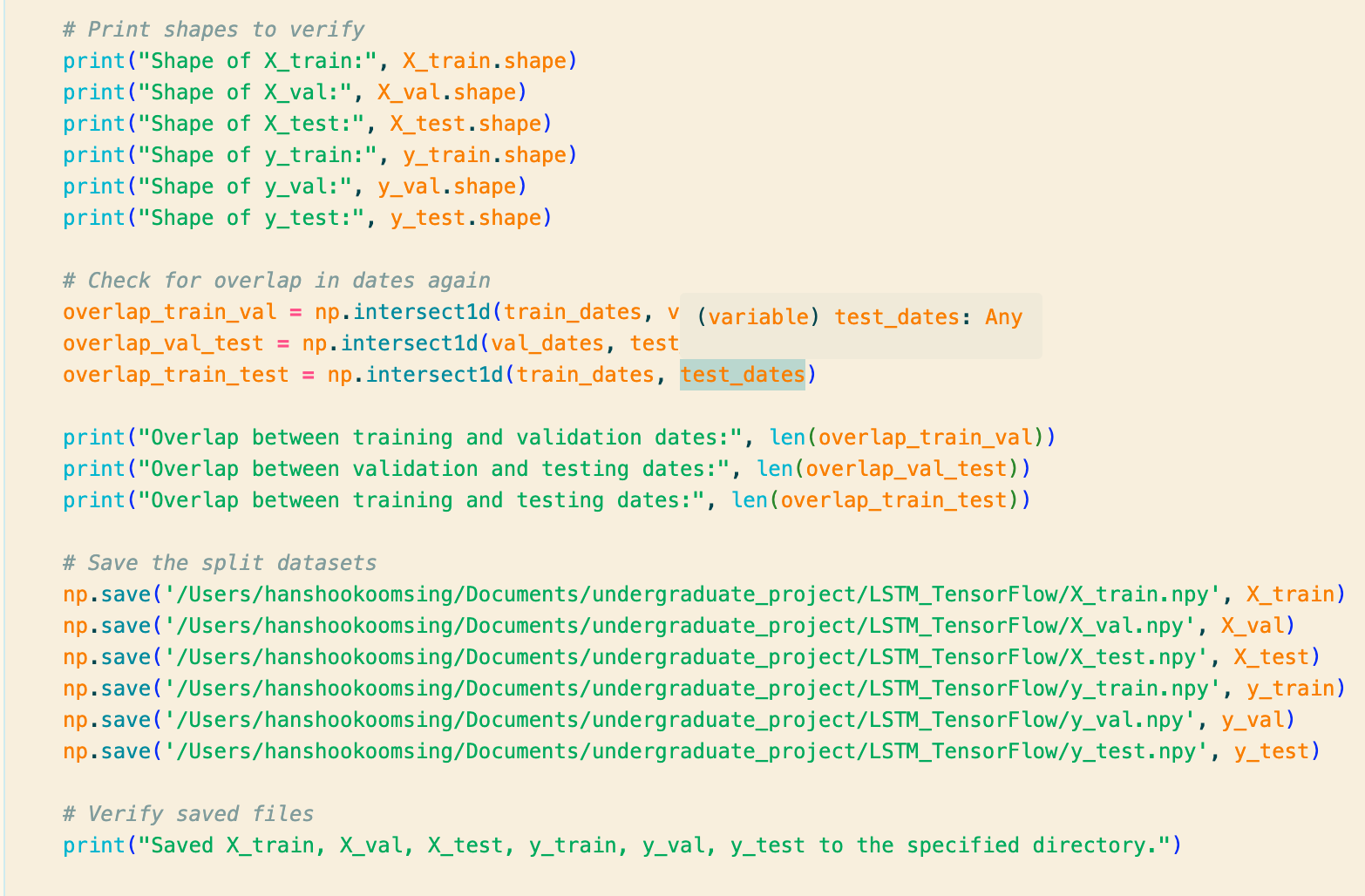


Figure 12 Part 3 of code used to split data into training, validation and testing datasets

#### Code Explanation

|  |  |
| --- | --- |
| **Step** | **Description** |
| Load Sequences, Labels, Dates | Loads the previously saved sequences (X), labels (y), and dates (dates) from .npy files |
| Sort Data by Date | Sorts the sequences, labels, and dates by the date to ensure chronological order before splitting. |
| Determine Split Sizes | Calculates the sizes for training (70%), validation (15%), and testing (15%) splits based on the total number of dates. |
| Split Dates | Splits the dates into training, validation, and testing sets based on the calculated sizes |
| Split Data | Splits the sequences and labels into training, validation, and testing sets according to the date splits. |
| Check and Remove Overlaps | Checks for overlaps between the date ranges of the splits and removes any overlaps to ensure distinct splits. |
| Print Shapes | Prints the shapes of the training, validation, and testing sets to verify the splits. |
| Check Overlaps Again | Checks for any remaining overlaps between the date ranges of the splits and prints the results |
| Save Split Datasets | Saves the split datasets (X\_train, X\_val, X\_test, y\_train, y\_val, y\_test) to .npy files in the specified directory. |
| Verify Saved Files | Prints a confirmation message indicating that the split datasets have been saved successfully. |

Table 8 Code Explanation

## The purpose of the different Data processing steps

|  |  |  |
| --- | --- | --- |
| **Step** | **Purpose** | **Reason** |
| Forward Filling Interest Rates | In order to ensure a continuous time series without missing data points (Narang, 2024). | Interest rates are reported irregularly; forward filling fills gaps with the last available value (Narang, 2024b). |
| Linearly Interpolating GDP Data | In order to create a continuous daily dataset from quarterly data (*Using linear interpolation to construct new data points | Apple Developer Documentation*, no date). | Linear interpolation estimates daily values between quarterly data points, aligning with other daily datasets (*Using linear interpolation to construct new data points | Apple Developer Documentation*, no dateb). |
| Creating Sequences | In order to capture temporal dependencies in the data for time-series analysis and predictive modeling (Wei, 2023). | Sequences allow the model to learn patterns and trends over time, improving prediction accuracy (Wei, 2023b). |
| Splitting Data into Features and Labels | In order to prepare the dataset for supervised learning (*Understanding features vs labels in a dataset*, no date). | Features are used by the model to make predictions; labels are used to train and evaluate the model (*Understanding features vs labels in a dataset*, no dateb). |
| Splitting into Training, Validation, and Testing Datasets | In order to evaluate the model's performance and ensure it generalizes well to new data (Khanna, 2024). | Training set is used to train the model, validation set to tune hyperparameters, and testing set to evaluate final model performance (Khanna, 2024b). |
| Using Robust Scaler | In order to standardize the data by removing the median and scaling according to the interquartile range (*RobustScaler*, no date). | The Robust Scaler is less sensitive to outliers than other scaling methods, ensuring a more reliable standardization (*RobustScaler*, no dateb). |
| Standardizing Data | In order to normalize the data to improve the model's performance and training stability (Brownlee, 2019). | Standardized data ensures that features have similar scales, which helps in faster convergence and better performance of machine learning algorithms (Brownlee, 2019b). |

Table 9 This table explains why the various steps in the data processing process was carried out

# Testing & Results

## Testing the Collected Data

### Code, explanation and test results for test performed on stock dataset



Figure 13 Code to perform tests on the stock data

#### Code Explanation

|  |  |
| --- | --- |
| **Loads the Data** | Reads the CSV file containing stock data. |
| **Ensures Correct Data Types** | Converts the 'Date' column to date format and the 'Close' column to numeric format, addressing any non-numeric values by coercion. |
| **Saves the Data** | Updates the CSV file with the corrected data types. |
| **Checks for Missing Values** | Identifies any missing values in each column. |
| **Verifies Date Alignment** | Checks if each date has the same number of entries, confirming that all stocks have data for each date. |
| **Counts Entries per Year** | Counts the number of stock entries for each year and prints the result. |

Table 10 This table explains the code to perform tests on the stock file

#### Test Results



Figure 14 Results of test performed on stock dataset

#### Explanation of Test Results

|  |  |
| --- | --- |
| **Description** | **Details** |
| Missing Values per Column | No missing values in any columns (Date, Stock, Close). This indicates complete data entries. |
| Date Alignment | All dates are aligned. This confirms that entries across different stocks are comparable on the same days. |
| Number of Stock Entries per Year | Varies year by year, generally close to 2016 entries per year, reflecting the number of trading days and stocks tracked. Different numbers may reflect variations in trading days due to holidays or leap years. |

Table 11 Explanation of test results for test done on stock data

|  |  |
| --- | --- |

### Code, explanation and test results for test performed on Macroeconomic dataset

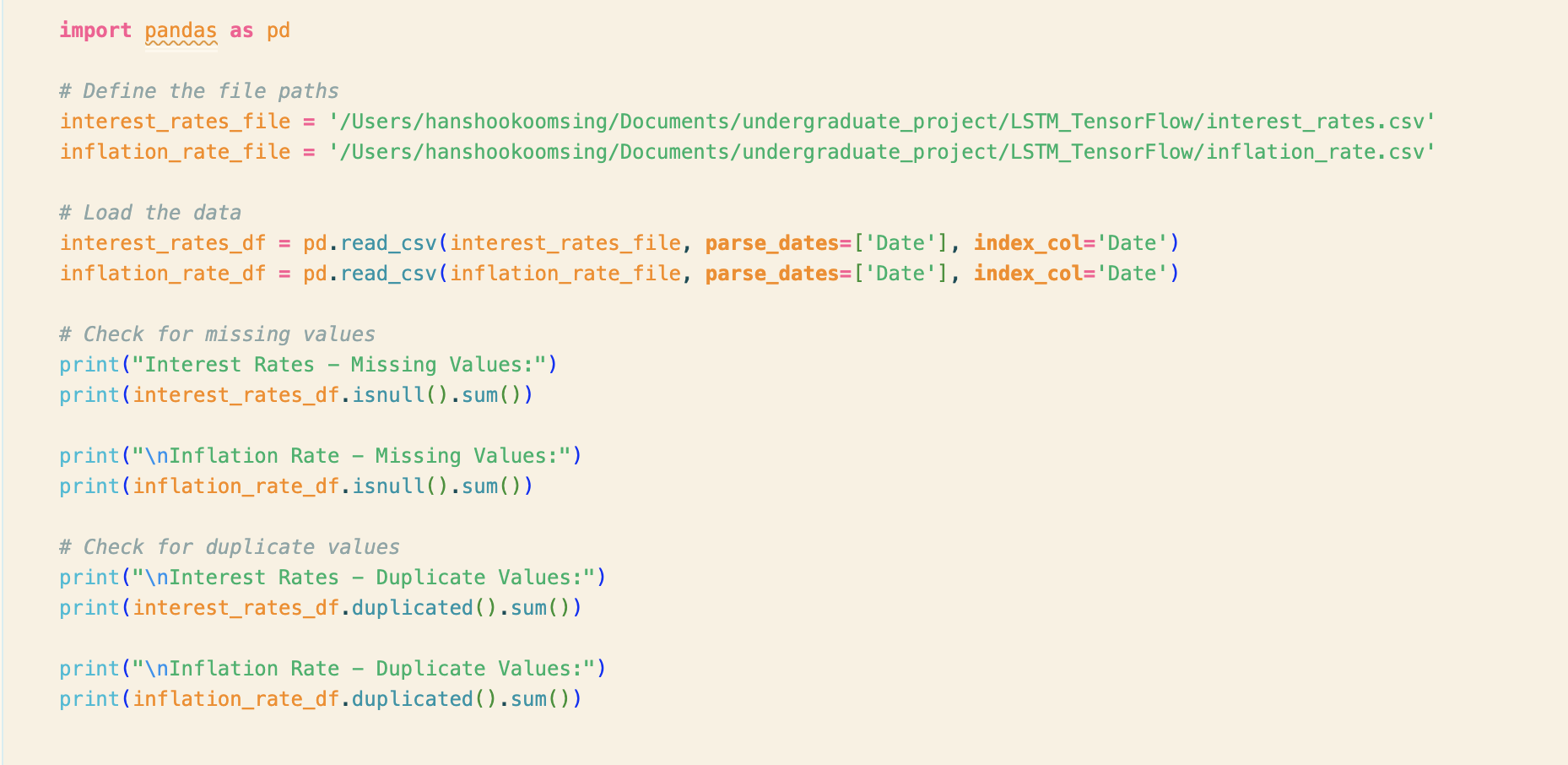


Figure 15 Code to test Macroeconomic Data

#### Test Results

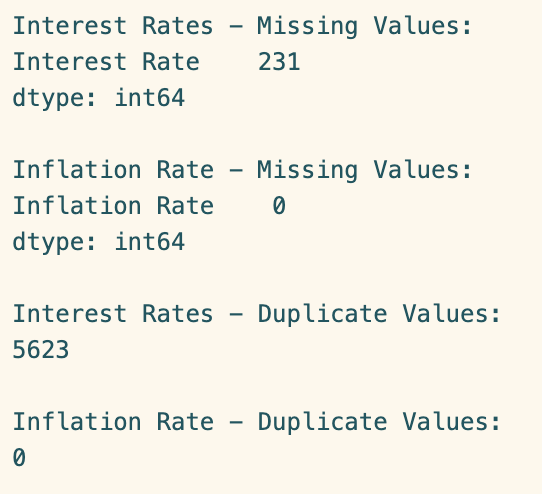


Figure 16 Test Results

#### Explanation of Test Results

|  |  |  |
| --- | --- | --- |
| **Data Quality Metric** | **Interest Rates** | **Inflation Rates** |
| Missing Values | 231 missing entries. May require imputation or exclusion. | No missing entries; data is complete. |
| Duplicate Values | 5623 duplicate entries, indicating possible errors or repetitions. | No duplicate entries; data collection is clean. |

Table 12 Explanation of test results for test done on macroeconomic data

### Code, explanation and test results for test performed after interpolating GDP Data



Figure 17 Part 1 of Test Code

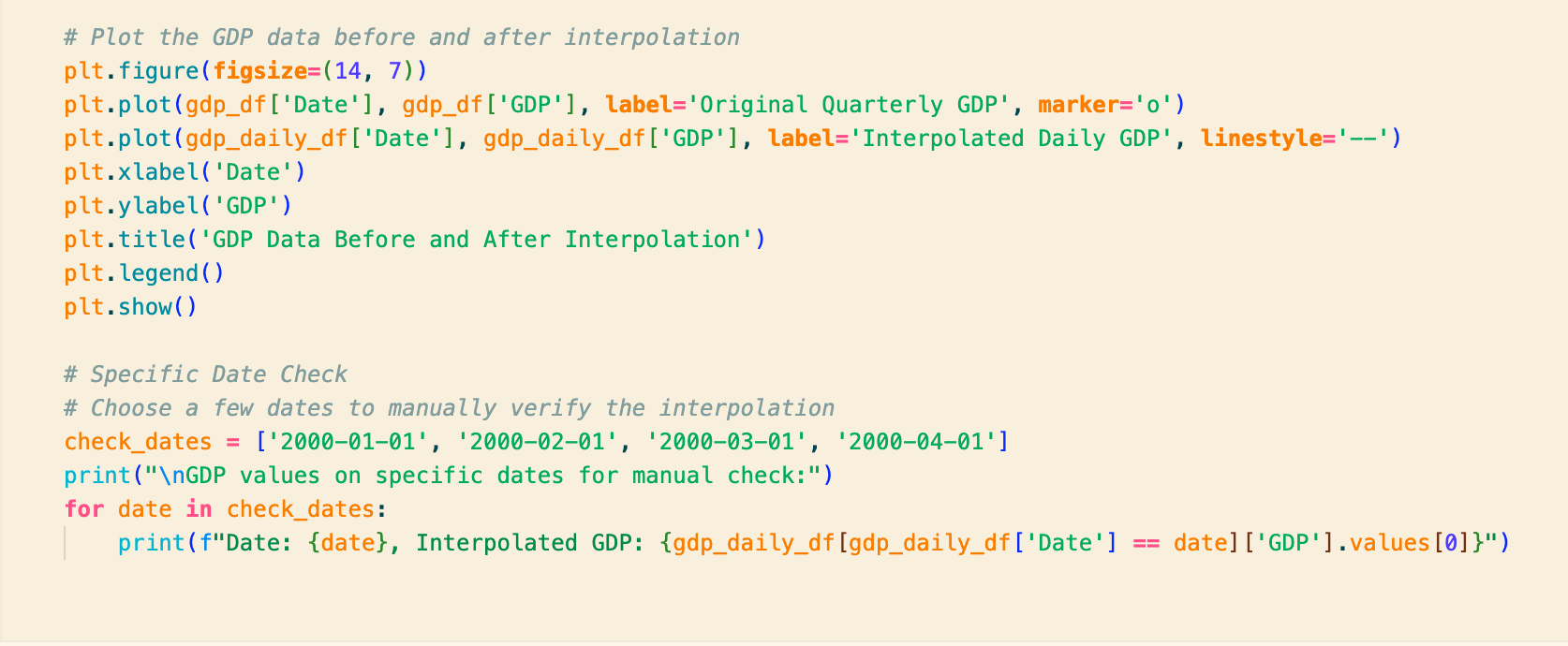


Figure 18 Part 2 of Test Code

### Code, explanation and test results for the test performed on the standardized dataset



Figure 19 Part 1 of test



Figure 20 Part 2 of Test

#### Code Explanation

|  |  |
| --- | --- |
| **Aspect** | **Details** |
| Data Loading | Two CSV files are loaded: processed\_data.csv and standardized.csv, which contain the original and scaled datasets respectively. |
| Numeric Columns | Identifies columns with numeric data types in the original dataset for analysis. |
| Scaling Test | Calculates and prints the median and interquartile range (IQR) of scaled numeric data. Medians should be close to 0, and IQRs should be close to 1 after standardization. Results confirm this, showing that the scaling process was correctly applied. |
| Data Visualization | Plots histograms for each numeric column to visually compare distributions of original and scaled data. |
| Data Format Check | Prints data types of columns in both datasets. Results confirm that data types remained unchanged between the original and scaled data. |

Table 13 Code Explanation for test done on standardized dataset

#### Test Results

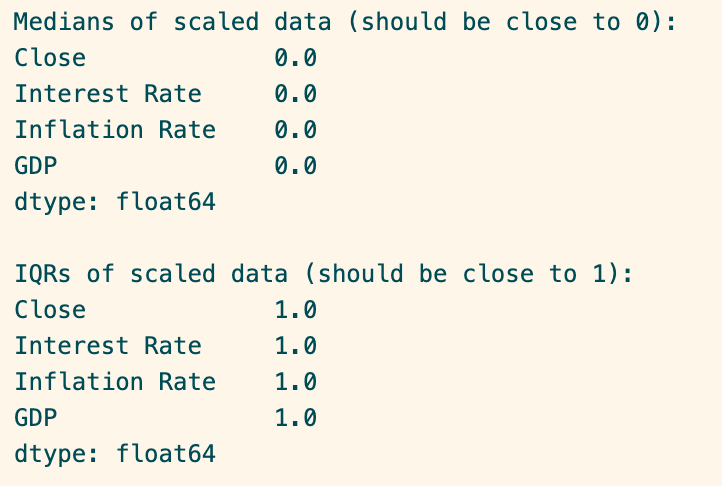


Figure 21 Part 1 of Test Results

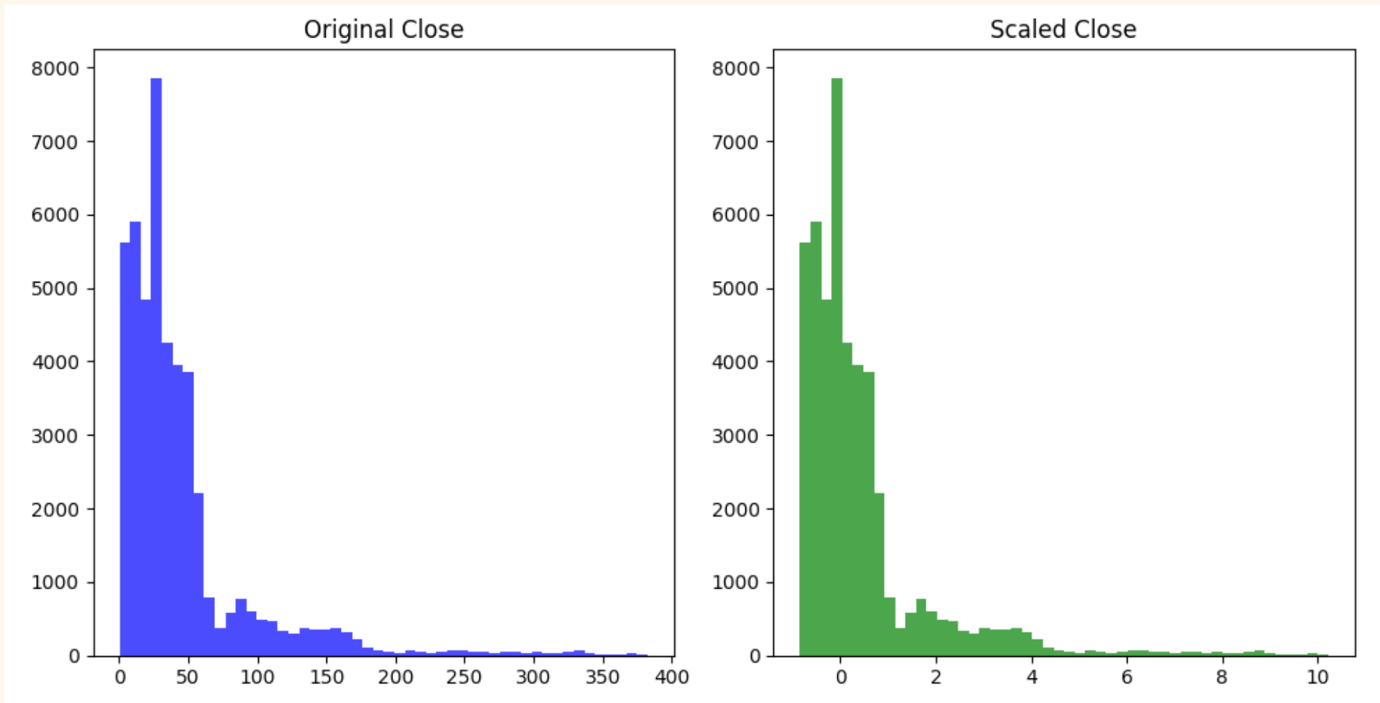


Figure 22 Part 2 of Test Results

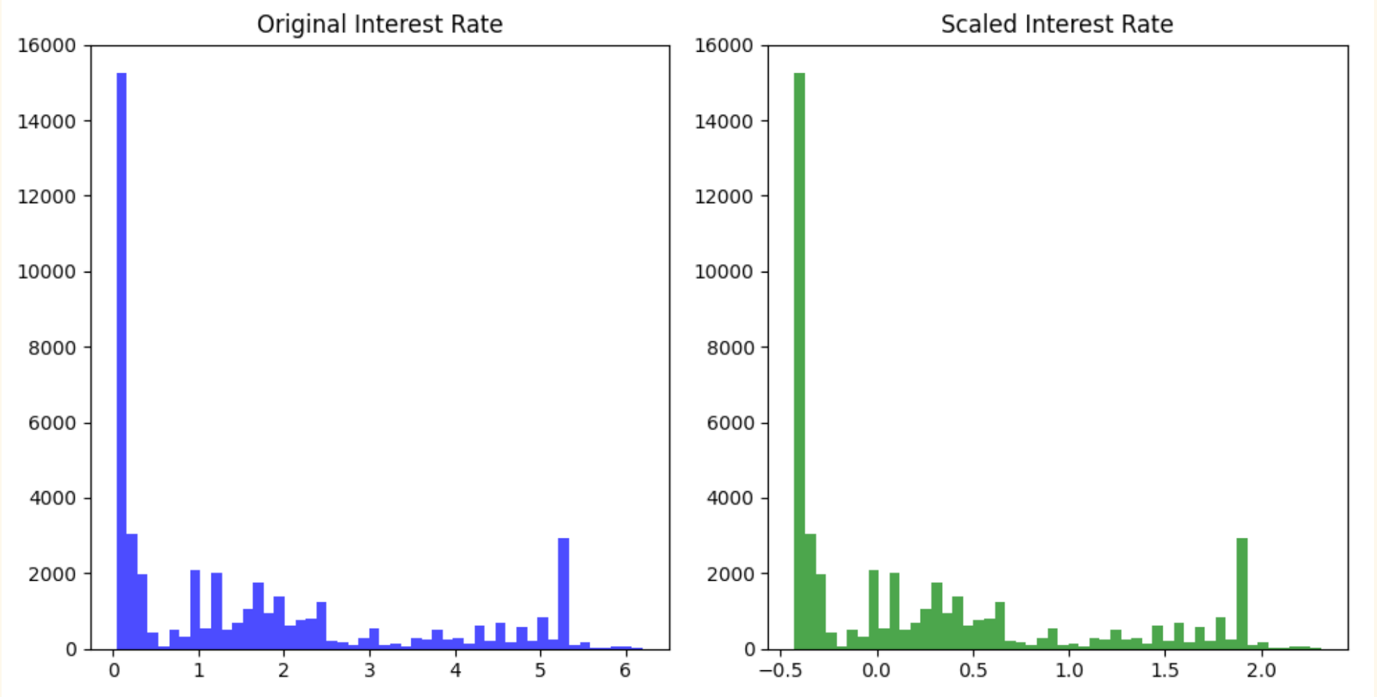


Figure 23 Part 3 of Test Results

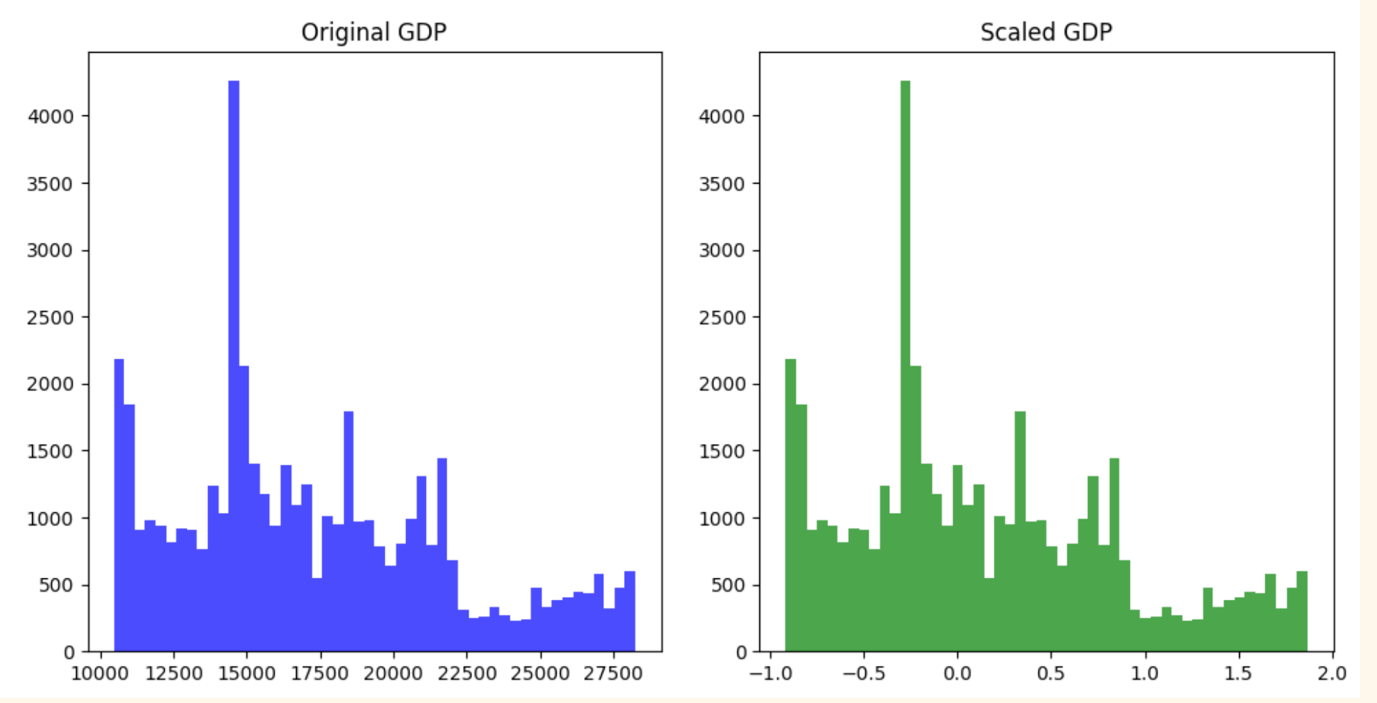


Figure 24 Part 4 of Test Results

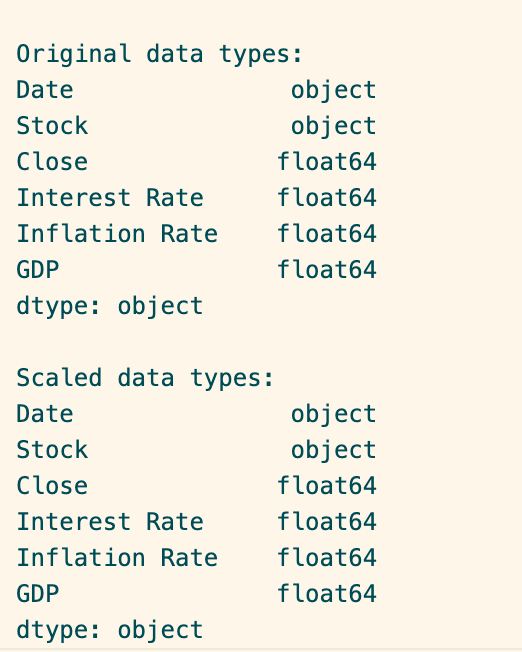


Figure 25 Part 5 of Test Results

#### Explanation of Test Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Metric** | | **Graph Analysis** | **Statistical Results** | |  | | --- | | **Data Types** | |
| Close | The histogram of the original Close prices shows a wide range of values with a significant concentration at lower values. Post-scaling, the distribution centers around 0 and is more compact, reflecting successful standardization | Median: 0.0, IQR: 1.0 (Scaled data medians and IQRs confirm effective normalization where the median is centered at 0 and the spread (IQR) is close to 1.) | Original and Scaled: float64 |
| Interest Rate | Original Interest Rate data is mostly concentrated at lower rates with some outliers. The scaled data appears mostly centered around 0 with a few outliers, showing the typical effects of standardization on skewed data. | Median: 0.0, IQR: 1.0 | Original and Scaled: float64 |
| GDP | GDP data in its original form shows multiple peaks, suggesting different groups or ranges of GDP values. The scaled version reduces this to a somewhat normal distribution centered around zero, indicating successful scaling. | Median: 0.0, IQR: 1.0 | Original and Scaled: float64 |

Table 14 Graph Analysis and Statistical Results Explanation

### Code, explanation and test results for the test performed after the train-val-test split



Figure 26 Part 1 of Test Code

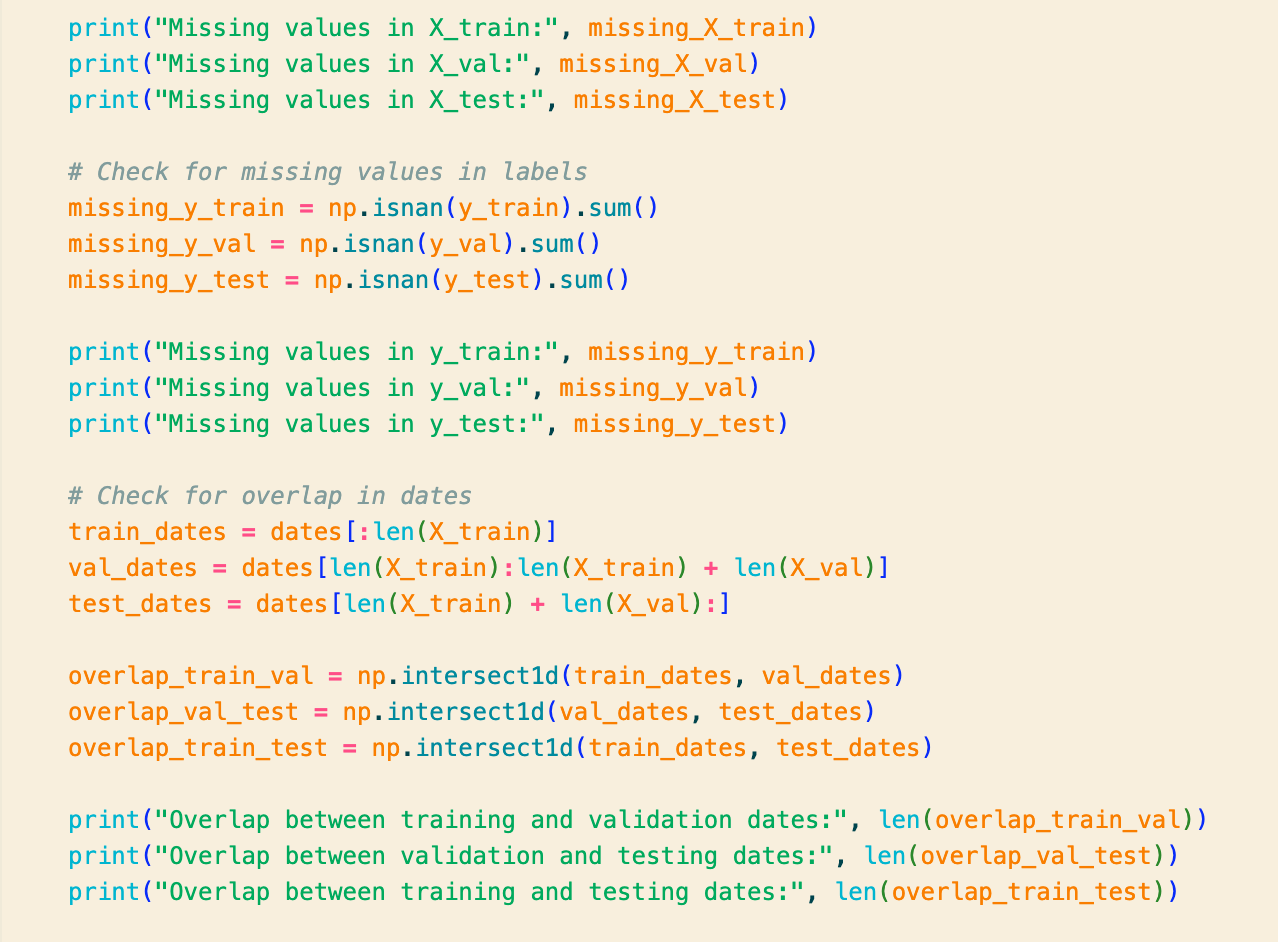


Figure 27 Part 2 of Test Code

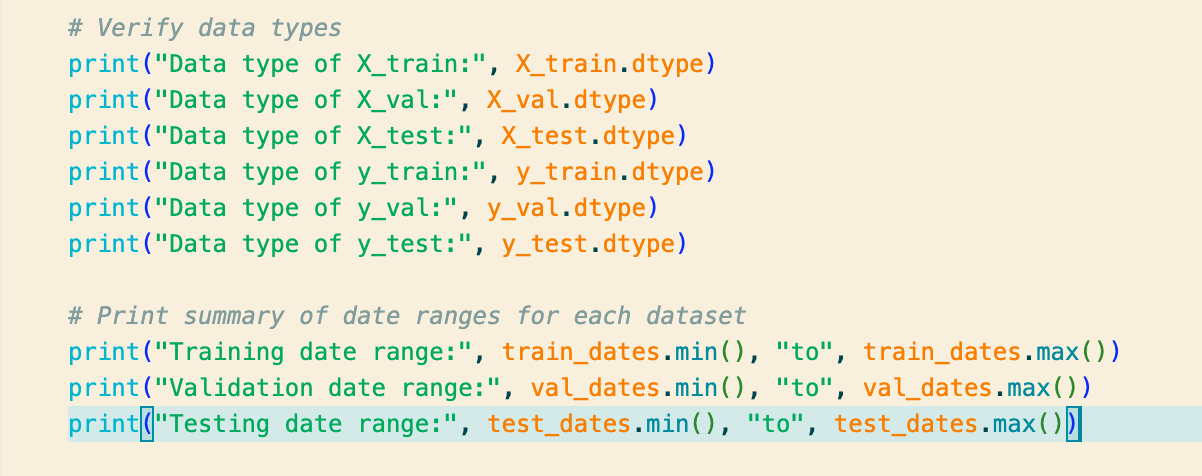


Figure 28 Part 3 of Test Code

#### Code Explanation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Step** | | |  | | --- | | **Action** | | |  | | --- | | **Purpose** | |
| Load Datasets | Loads X\_train, X\_val, X\_test, y\_train, y\_val, y\_test, and dates from .npy files. | To prepare the feature and label datasets for training, validation, and testing, as well as the corresponding dates. |
| Sort Data by Date | Dates are sorted, and sorted indices are applied to ensure all data splits are time-ordered. | Ensures that data is processed in a chronological order, which is critical for time series analysis. |
| Verify Shapes | Prints the shapes of all datasets. | Confirms the dimensions of each dataset to ensure they are correctly partitioned and loaded |
| Check Missing Values | Checks and prints the count of missing values in X\_train, X\_val, X\_test, y\_train, y\_val, y\_test. | Identifies any missing data that could impact model training or require preprocessing. |
| Check Date Overlaps | Checks for overlapping dates between training, validation, and testing datasets | In order to ensure that there is no data leakage between splits and that each set is distinct |
| Verify Data Types | Prints the data types of all datasets. | Ensures consistency in data types for model input requirements |
| Print Date Ranges | Prints the date ranges for training, validation, and testing sets. | Provides an overview of the temporal coverage for each dataset to understand the period each dataset represents. |

Table 15 Code explanation for the test done after training, test, split

#### Test Results

#### Explanation of Test Results

|  |  |
| --- | --- |
| **Parameter** | **Explanation** |
| Shape | Each dataset dimension is listed, showing the number of samples, the length of each sequence (28 days), and the number of features (4 features per day). |
| Missing Values | Indicates that there are no missing values in any of the datasets, which is crucial for maintaining the integrity and performance of the models trained on this data. |
| Data Types | All datasets consist of float64 data types, ensuring that numerical operations are consistent and precise across the training, validation, and testing phases |
| Date Ranges | Displays the temporal coverage for each dataset, crucial for understanding the period represented and ensuring no temporal leakage in model training and evaluation. |
| Overlap with Next Set | Highlights the presence of an overlap between the training and validation datasets (1 day), which is not enough in order to create a significant data leakage |

Table 16 Explanation of test results for test performed after the train, test, validating data split

Evaluation

# Conclusion

# User Guide

# Bibliography

Talend (no date) *What is Data Processing? Definition and Stages - Talend Cloud Integration*. <https://www.talend.com/resources/what-is-data-processing/#:~:text=Data%20processing%20occurs%20when%20data,end%20product%2C%20or%20data%20output>.

Narang, M. (2024) *Top 4 techniques for handling missing values in Machine Learning*. <https://blog.paperspace.com/top-4-techniques-for-handling-the-missing-values-in-machine-learning/#:~:text=Forward%20fill%20(ffill)%20and%20backward,fill%20in%20the%20missing%20values.&text=Forward%20fill%20replaces%20missing%20values%20with%20the%20previous%20non%2Dmissing%20value>.

Narang, M. (2024b) *Top 4 techniques for handling missing values in Machine Learning*. <https://blog.paperspace.com/top-4-techniques-for-handling-the-missing-values-in-machine-learning/#:~:text=Forward%20fill%20(ffill)%20and%20backward,fill%20in%20the%20missing%20values.&text=Forward%20fill%20replaces%20missing%20values%20with%20the%20previous%20non%2Dmissing%20value>.

*Using linear interpolation to construct new data points | Apple Developer Documentation* (no date). <https://developer.apple.com/documentation/accelerate/using_linear_interpolation_to_construct_new_data_points>.

*Using linear interpolation to construct new data points | Apple Developer Documentation* (no dateb). <https://developer.apple.com/documentation/accelerate/using_linear_interpolation_to_construct_new_data_points>.

Wei, S. (2023) 'Method to create input output sequences of time series for machine learning,' *Medium*, 4 July. <https://medium.com/@shouke.wei/method-to-create-input-output-sequences-of-time-series-for-machine-learning-dad638479a4e#:~:text=By%20defining%20appropriate%20sequence%20lengths,in%20time%20series%20forecasting%20tasks>.

Wei, S. (2023) 'Method to create input output sequences of time series for machine learning,' *Medium*, 4 July. <https://medium.com/@shouke.wei/method-to-create-input-output-sequences-of-time-series-for-machine-learning-dad638479a4e#:~:text=By%20defining%20appropriate%20sequence%20lengths,in%20time%20series%20forecasting%20tasks>.

Khanna, S. (2024) *A comprehensive Guide to Train-Test-Validation split in 2024*. <https://www.analyticsvidhya.com/blog/2023/11/train-test-validation-split/#:~:text=The%20train%2Dtest%2Dvalidation%20split%20helps%20assess%20how%20well%20a,to%20generalize%20to%20new%20instances>.

Khanna, S. (2024b) *A comprehensive Guide to Train-Test-Validation split in 2024*. <https://www.analyticsvidhya.com/blog/2023/11/train-test-validation-split/#:~:text=The%20train%2Dtest%2Dvalidation%20split%20helps%20assess%20how%20well%20a,to%20generalize%20to%20new%20instances>.

*RobustScaler* (no date). <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html>.

*RobustScaler* (no dateb). <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html>.

Brownlee, J. (2019) 'How to Normalize and Standardize Time Series Data in Python', Machine Learning Mastery. Available at: <https://machinelearningmastery.com/normalize-standardize-time-series-data-python/>

Brownlee, J. (2019b) 'How to Normalize and Standardize Time Series Data in Python', Machine Learning Mastery. Available at: <https://machinelearningmastery.com/normalize-standardize-time-series-data-python/>

# Appendix

## Link to Macroeconomic datasets

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Dataset Code** | **Units** | **Link** |
| |  | | --- | | Gross Domestic Product | | GDP | Billions of Dollars | <https://fred.stlouisfed.org/series/GDP> |
| Effective Federal Funds Rate | EFFR | Percent | <https://fred.stlouisfed.org/series/EFFR> |
| Sticky Price Consumer Price Index less Food and Energy | CORESTICKM159SFRBATL | Percent Change from Year Ago | <https://fred.stlouisfed.org/series/CORESTICKM159SFRBATL#0> |

Table 17 Links to the Macroeconomic datasets where they can be downloaded using the FRED API