UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE POLITICHE, ECONOMICHE E SOCIALI

Master's Degree Program in
Data Science and Economics

NOWCASTING US GDP GROWTH: AN AUTOMATED MACHINE LEARNING APPROACH

Supervisor: Prof. Matteo Bodini
Co-supervisor: Prof. Fabrizio Iacone

Author:
Filippo Guardassoni

Academic Year 2022/2023

Acknowledgements

I am deeply grateful to my family, friends, and professors for their unwavering support throughout my academic journey. Your belief in me and your encouragement have been instrumental in my success.

I also want to acknowledge the professors and colleagues who supported me in writing my thesis and whose work has inspired and informed my thesis.

This journey has been challenging and rewarding, and I thank everyone who has contributed to its successful conclusion.

Abstract

This research paper presents a comprehensive evaluation of Automated Machine Learning (AutoML) techniques in the domain of economic nowcasting. AutoML, which automates the machine learning pipeline, is examined for its potential to outperform traditional statistical models in predicting economic indicators, specifically Gross Domestic Product (GDP) growth. The study compares various AutoML frameworks, including H2O, Auto-Sklearn, AutoGluon, TPOT, and AutoKeras, against established statistical models like Dynamic Factor Models (DFM) and ARIMA. The comparison encompasses different evaluation periods, capturing both periods of economic stability and turbulence, such as the COVID-19 recession. Key findings reveal that AutoML models consistently outperform traditional statistical models, particularly Extra Trees and XGBoost, showcasing their potential for enhancing the accuracy of economic nowcasting. Feature importance analysis highlights the distinct patterns in variable contributions among these models, shedding light on their interpretability. The research also emphasizes the importance of systematically selecting features and fine-tuning hyperparameters to improve AutoML model performance further. The Diebold-Mariano test is employed to determine the statistical significance of these improvements, providing valuable insights into the models' forecasting capabilities. In conclusion, this study underscores the promising role of AutoML in advancing the field of economic nowcasting. The results suggest that well-tailored AutoML models can offer superior predictive performance, challenging the dominance of traditional statistical models. These findings contribute to the ongoing discourse on the adoption of cutting-edge machine learning techniques in economic forecasting.

Table of Contents

Chapter 1 – Introduction

Gross Domestic Product ("GDP" hereafter) measures the monetary value of final goods and services produced in a country given a specific period of time. GDP is composed of goods and services produced for sale in the market, however it also includes some non-market production, such as defense and education ("*Gross Domestic Product: An Economy's All*", International Monetary Fund). GDP is most commonly calculated on an annual basis, but it can be measured on a quarterly frequency in the biggest economies. The GDP can be found in two forms: nominal and real. The main difference between the two is that the real GDP is calculated using inflation-adjusted data since it is based on the monetary values. Nominal GDP risks to not reflect the proper growth of the quantity or the quality of goods and services ("*Gross Domestic Product: An Economy's All*", International Monetary Fund). On the other hand, the GDP growth rate compares year-over-year (or quarter) change in a country's economic output to measure how fast an economy is growing. It is usually expressed as a percentage rate. GDP growth rates provide an alternative perspective that can be valuable in various contexts. For instance, policymakers often pay close attention to GDP growth rates, especially when they are increasing. This could signal economic overheating, prompting central banks to consider raising interest rates as a measure to maintain stability. However, it's crucial to note that GDP alone does not reflect the overall quality of life or the well-being of a nation. It offers a specific economic indicator but does not encompass all aspects of a population's standard of living or happiness. As a broad measure of overall domestic production, GDP is considered as the single most important indicator to capture economic activity, health and growth, and is used by policymakers, researchers and the general public. In order to be a reliable indicator is often revised before being released due to its complex and subjective nature. GDP is calculated merging data submitted by different organizations within a country. As this process is slow and complicated, it usually results in a delayed release of GDP figures (especially quarterly) which makes it challenging to assess economic activity in real-time (i.e., synchronously). This lag depends on the country, but in the United States, advanced estimates for an elapsed quarter are not released for at least one month afterwards, with final estimates not appearing until three months afterwards (Federal Reserve Bank of San Francisco, 2005).

In order to thin this gap, a technique called Nowcasting is employed. Nowcasting "is the estimation of the current, or near to it either forwards or backwards in time, state of a target variable using information that is available in a timelier manner" (Hopp, 2021). In essence, series that are available in a timelier manner than GDP can be used to estimate a model using historical data, when data for both GDP and these independent series are available. This model can then be used to obtain estimates for GDP well before advanced estimates are available, even while the quarter for prediction is ongoing. Nowcasting models have been developed to mitigate some of these uncertainties, and they have been widely used by forecasters at many central banks and other institutions (Giannone, 2008; Banbura, 2013; Jansen, 2016; Bloor, 2009). Nowcasting models typically involve time-series regression models. However, recent studies revealed that Machine Learning ("ML" hereafter) algorithms are potential alternatives (Dauphin, Dybczak, Maneely, Sanjani, Suphaphiphat, Wang, and Zhang, 2022). ML models are particularly suited for handling datasets where the number of potential regressors is large. In particular, the Long Short-Term Memory ("LSTM" hereafter) neural network is possibly identified as one of the most accurate algorithms for nowcasting (Hopp, 2021). Nevertheless, the application of ML techniques to nowcasting problems is a relatively recent task, hence it is still in an early stage (Hopp, 2022). The process of applying traditional machine learning methods time-consuming, resource-intensive, and challenging; domain knowledge is also required to a certain extent (Gijsbers, LeDell, Thomas, Poirier, Bischl and Vanschoren, 2019). Therefore, the search for the right algorithm might be incompatible with the need of timely disposable data. A further step towards a better solution is the application of Automated Machine Learning ("AutoML" hereafter).

AutoML is the process of automating the tasks of applying machine learning to real-world problems. AutoML potentially includes every stage from beginning with a raw dataset to building a machine learning model ready for deployment. AutoML was proposed as an artificial intelligence-based solution to the growing challenge of applying machine learning (Thornton, Hutter, Hoos, & Leyton-Brown, 2013). The high degree of automation in AutoML aims to allow non-experts to make use of machine learning models and techniques without requiring them to become experts in machine learning. Automating the process of applying machine learning end-to-end additionally offers the advantages of producing simpler solutions, faster creation of those solutions, and models that often outperform hand-designed models (Alsharef, Sonia, Kumar, Iwendi, 2022).

To the best of knowledge up to the time of writing this paper, and following an extensive review of prior literature, no articles were identified pertaining to the application of AutoML techniques to GDP nowcasting problems. Thus, the goal of this paper is to apply and evaluate different AutoML frameworks in order to predict the US GDP growth using explanatory variables from the Federal Reserve of Economic Data (FRED) as specified in (Bok, 2018). This dataset is one of the most used in this field and it features a long publication history starting in 1947 (McCracken and Ng, 2016). After an extensive Exploratory Data Analysis ("EDA" hereafter), 9 frameworks were analyzed, and then tested over two different evaluation periods, also called out-of-sample datasets. In addition, a real-time setting of the nowcasting exercise is simulated through 3 different vintages: one-, two- and three-month data availability. While several models were generated, LSTM and Ensemble Algorithm (such as Extra Trees) were the best ones. The packages were able to achieve a very small loss and high accuracy on the dataset with AutoKeras and TPOT being the top performers. The significance of this research lies in its contribution to understanding the role of AutoML in contemporary data science and machine learning practices taking nowcasting as a specific example. As data continues to proliferate across sectors, the ability to automate the modeling process becomes increasingly valuable. This study can inform practitioners, researchers, and policymakers about the efficacy of AutoML in various contexts.

This thesis is organized into several chapters, each addressing specific facets of AutoML. Chapter 2 provides a comprehensive literature review, summarizing the key concepts, methodologies, and tools in the AutoML landscape. Chapter 3 outlines the research methodology, detailing the datasets, evaluation metrics, and experimental setups used in this study. Chapters 4 and 5 present the empirical results and discussions, offering insights into the performance and interpretability of AutoML models. Finally, Chapter 6 concludes the thesis, summarizing the key findings and their implications for the broader field of machine learning.

Chapter 2 – Literature Review

This chapter presents a comprehensive analysis of existing research and studies relevant to the topic. It critically evaluates key findings, methodologies, and gaps in the current literature. This chapter contextualizes the research and highlights its significance within the broader academic discourse.

2.1 Nowcasting

Nowcasting, originally coined and applied in meteorology in the 1980s, refers to a forecasting technique that predicts the near future by using current available data. The first attempt to adapt this technique to the economic field is encapsulated in the idea formulated by Sargent and Sims (1977) that the salient features of business cycle fluctuations can be captured by a small number of factors that could be estimated from data. Stock and Watson (1989) were among the first to construct indices of business cycle using factor models, albeit being relatively simple, extracted a single common factor from a small set of monthly core indicators using maximum likelihood. Nowcasting became truly popular in the economic literature in the mid-2000s after the publication of Giannone's research in 2005. Prior to this, in 2003, Mariano and Murasawa developed a coincident business cycle index based on real-time data to estimate the macroeconomic situation, which was similar to economic nowcasting.

Over time, factor models used in economic nowcasting have undergone significant improvements and expansions to accommodate larger datasets with a higher number of variables. Giannone et al. (2005) and Watson (2005) demonstrated that the factor model approach could be applied to high-dimensional datasets, and they proposed formulating factor models using state-space specifications to reduce the dimensionality of large economic systems, thereby extracting unknown common factors. While the large number of variables does not inherently restrict the feasibility of estimation (Doz et al., 2012), it increases the time required for estimation. Consequently, researchers had to be selective in choosing which variables to include in the model. As factor models continued to evolve, they became widely used tools for monitoring economic conditions. For instance, Giannone, Reichlin, and Small (2008) implemented a model for nowcasting US GDP at the Board of Governors of the Federal Reserve, which was based on a project that began in 2003. Variants of this model were subsequently developed for different economies and implemented in other central banks, such as the European Central Bank (ECB, 2008) and the International Monetary Fund (IMF) (Matheson, 2011; Taheri Sanjani, 2013). Continuous refinement and improvement of the factor model methodology have been ongoing. D'Agostino et al. (2011) and Giannone et al. (2013) published a series of papers in which they further developed the Dynamic Factor Model (DFM) methodology and enhanced the prediction properties of these models by improving the Kalman-Filter estimation method and optimizing the data selection process. These advancements in factor models have contributed to more accurate and efficient nowcasting of key economic indicators, enabling policymakers and institutions to make more informed decisions in real-time based on timely and reliable data-driven estimates. Matheson et al. (2014) developed monthly indicators for tracking short-run trends in real GDP growth and conducted nowcasting analysis for 32 advanced and emerging-market economies.

Additionally, other methodologies like MIDAS (applied by Kuzin et al., 2009 and Marcellino and Schumacher, 2010 in forecasting the German economic activity), MF-VAR (Kuzin et al., 2009), Bayesian VAR (Kuzin et al., 2009), and various machine learning algorithms have been employed for economic nowcasting in different studies, each having its own characteristics and suitability for dealing with data challenges in the nowcasting context.

2.2 Time-Series Prediction Using Machine Learning

As a result of the enhancement in efficiency, accuracy and customizability, machine learning techniques have become popular among economists (Shen, Zhang, Lu, Xu, Xiao, 2020) (Livieris, Pintela, 2020) (Du Li, Yang, Horng, 2020). Several studies, including Varian (2014), Belloni et al. (2014), and Storm (2020), explored tools for manipulating and analyzing big data in economics. Amat et al. (2018) demonstrated that machine learning outperforms traditional methods in predicting floating exchange rates. Bajari et al. (2015) found that machine learning methods perform better than standard panel data models in estimating consumer demand. Chakraborty and Joseph (2017) discussed machine learning's application in central banking and presented case studies using it.

Kohlscheen (2021) used regression trees and random forests to examine inflation drivers. Basuchoudhary et al. (2014) applied various machine learning algorithms to predict long-term economic growth and the likelihood of recessions, identifying significant variables. Athey (2019) compared machine learning with traditional econometrics and highlighted its advantages in policy analysis. Overall, these studies showed that machine learning has the potential to significantly impact empirical work and improve economic predictions.

On the other hand, Artificial Neural Networks (ANNs) have seen substantial advancements in various fields, however their impact on economics has been limited until recent years (Németh, Hadházi, 2023). Despite, only few studies of nowcasting have used ANNs, several others have applied them to forecast macroeconomic and financial variables. The first instances of these trials date back in the early 2000s. Kuan and Liu (1995) used ANNs for financial forecasting, in particular exchange rates. Tkacz (2001) forecasted Canadian GDP growth rate using ANNs with lagged GDP and financial variables. Heravi et al. (2004) applied ANNs to predict monthly industrial production for European economies using only lagged values. The cited works demonstrated the use of AI and machine learning, particularly artificial neural networks (ANNs), for stock market time series prediction. Li et al. (2010) found that ANNs are effective in financial economics due to their learning, generalization, and nonlinear behaviour properties. Mitra et al. (2011) surveyed machine learning methods for processing textual input from news stories to predict abnormal behaviour in stock time series based on sentiment scores. Changqing et al. (2016) reviewed advancements in non- linear and non-stationary time series forecasting models and noted the increasing interest in nonparametric models for forecasting in real-world manufacturing and health informatics applications. Peng et al. (2015) used text mining with neural networks to improve stock movement prediction by extracting information from financial news. Tiffin (2016) applied machine learning algorithms in case of Lebanon, a country with long delays in the publication of its GDP. Sentiment analysis techniques with deep network models were applied by Prosky et al. (2017) to enhance time series stock trend prediction. Various works, including McNally et al. (2018), Phaladisailoed et al. (2018), Alsharef, Sonia, Aggarwal (2021), compared deep learning with linear models in predicting time series data, such as cryptocurrency and stock market prices, showing that deep-learning- based models outperformed linear models. Torres and Qiu (2018) applied recurrent neural networks to daily data of several crypto-currencies, exchange rates, commodities and stocks between 2013 and 2017. Bolhuis and Rayner (2020) tested alternative machine learning algorithms to nowcast the Turkish GDP growth. Loermann and Maas (2019) conducted a comparison between Multi-Layer Perceptron (MLP) models and dynamic factor models (DFMs) for nowcasting and forecasting accuracy. They found that MLP outperformed DFMs and achieved results comparable to those of the Survey of Professional Forecasters. Hopp (2021) investigated a more complex neural network architecture called Long Short-Term Memory (LSTM) for nowcasting global merchandise export values and volumes, as well as global services exports. The study showed that LSTMs provided more accurate nowcasts compared to DFMs. Most recently, Németh and Hadházi (2023) tested furtherly LSTM to forecast US GDP growth. The study conducted a nowcasting competition between different statistical models and found that all models outperformed the naive benchmark. Some neural network architectures performed better than traditional time series approaches, particularly the CNN. The results indicated that long-term memory was not crucial for GDP nowcasting.

2.3 Automated Machine Learning

As seen, Machine Learning (ML) has gained significant importance, finding applications in various domains such as automatic speech recognition, self-driving cars, and predictive maintenance in Industry 4.0. ML has showcased impressive capabilities, even surpassing human performance in complex tasks (Silver et al., 2017). Building sophisticated ML pipelines for such applications requires a highly trained team of human experts. This represents the point at which complexity can escalate. In fact, this process is time consuming, costly, and often involves trial and error iterations (Zöller, Huber, 2021). Many machine learning algorithms require a significant level of expertise in domains such as preprocessing, feature selection, and hyperparameter optimization to achieve satisfactory results in forecasting tasks (Quemy, 2020). More importantly, individuals with proficiency in both machine learning and domain knowledge are relatively scarce (Dahl, 2020) making seeking, for example, time series forecasting solutions very expensive for the organizations (Alsharef, Kumar, Iwendi, 2022).

AutoML aims to automate this process and enhance the efficiency of building ML applications (Mohr, Wever, Hüllermeier, 2018). Automated machine learning (AutoML) offers solutions for constructing and executing machine learning pipelines with reduced reliance on human intervention (Manikantha, Jain, 2021). For example, ML experts can benefit from automation, especially in tasks like hyperparameter optimization (HPO), making the development process more efficient. In addition, AutoML empowers practitioners making it significantly simpler and accessible to develop and deploy ML applications as almost no manual work is necessary (Hu, Huang, 2017).

AutoML is not a new trend and has been evolving since the 1990s. Commercial solutions began offering automatic HPO for specific classification algorithms via grid search in the 1990s (Dinsmore, 2016). Early adaptations of grid search using a best-first approach were available since 1995 (Kohavi & John, 1995). Efficient HPO strategies emerged in the early 2000s, demonstrating superior results in less time for specific settings, such as tuning support-vector machines (SVMs) (Momma & Bennett, 2002; Chapelle et al., 2002; Chen et al., 2004). In 2004, the first approaches for automatic feature selection were published (Samanta, 2004). Full model selection aims to automate complete ML pipelines, selecting preprocessing, feature selection, and classification algorithms simultaneously, while tuning their hyperparameters (Escalante et al., 2009). As Guyon et al., 2008 specified, this domain-agnostic method exhibited potential in various datasets.

From 2011 onwards, Bayesian optimization gained popularity for hyperparameter tuning (Bergstra et al., 2011; Snoek et al., 2012) and model selection (Thornton et al., 2013). In 2015, the first method for automatic feature engineering without domain knowledge was proposed (Kanter & Veeramachaneni, 2015). The ability to construct variable-shaped pipelines emerged in 2016 (Olson & Moore, 2016). AutoML garnered substantial media attention in 2017 and 2018 with the release of commercial solutions by global players (Golovin et al., 2017; Clouder, 2018; Baidu, 2018; Das et al., 2020). Research in AutoML made remarkable progress during this period, resulting in significant improvements in performance. Recent advancements have considerably reduced the runtime of AutoML procedures, making it a valuable and efficient tool for practitioners (Hutter et al., 2018b).

The application of AutoML can be seen as tasks: regression, classification, time-series and unsupervised ML. When the task is forecasting time series, the possible methods are categorized into three groups: linear modeling, machine learning, in particular, Deep Learning (“DL” hereafter), and Automated Machine Learning (AutoML). Linear models are simple and fast to execute but may have lower prediction accuracy (Liu, Hou, Liu, 2017). DL models can perform better, but like other machine learning methods, they require expertise to fine-tune hyperparameters and set up the model. Additionally, the number of hyperparameters to optimize can be large, and the process might be time consuming. AutoML aims to address this issue by automatically finding an appropriate ML model and optimizing it (Tornede, Tornede, Wever, Hüllermeier, 2021). As a result, in the last couple of years, there is a growing demand for frameworks that automate the machine learning pipeline for forecasting time series (Dahl, 2020). However, establishing a mechanism that automates the whole ML process for forecasting is not yet a developed area of study, and also contains limitations and peculiarities that should be handled in special ways (Paldino, De Stefani, De Caro, Bontempi, 2021). Even in this relatively embryonal stage, the application of AutoML to time series specific problem is a big step in the field. This is also due to the fact that literature and documentation for newer developed forecasting methods are somewhat scarce (many of these algorithms are created by companies which do not disclose how they work in order to make profit).

Researchers have proposed AutoML solutions which demonstrated accurate forecasts without human input and outperformed various machine learning benchmarks such as the TSPO framework designed by Dahl (2020). The TSPO system demonstrated superior performance compared to various machine learning in 9 out of 12 randomly selected time series datasets from different domains. An automated time series regression challenge (AutoSeries) was organized to push forward research on automated time series, where competitors delivered efficient models compared to the baseline benchmark model (LightGBM) (Xu et al., 2021). Paldino et al. (2021) evaluated four AutoML frameworks (AutoGluon, H2O, TPOT, Auto-sklearn) against traditional forecasting techniques (naïve, exponential smoothing, Holt-Winter’s) on various time series forecasting challenges. However, the results indicated that AutoML techniques are not yet mature enough to effectively address time series forecasting problems, prompting researchers to encourage more rigorous validation and further work in this field. Javeri et al. (2021) presented a data augmentation method to enhance the performance of neural networks on intermediate length time series. By combining data augmentation with AutoML techniques, they achieved significant accuracy improvements in predicting a COVID-19 dataset with three neural network-based models. Nevertheless, full automation of the entire machine learning process for forecasting is still an evolving field of research, and time series data present unique challenges that require specialized handling due to constraints and oddities like trend, seasonality, outliers, drifts, and abrupt changes.

2.4 Contribution Of This Thesis

This thesis is a proof of concept (POC), also known as proof of principle, and it is a demonstration or experiment conducted to verify the feasibility and practicality of a particular idea, concept, or technology. It is a preliminary evidence or demonstration that a proposed solution or innovation can work effectively in a real-world setting. The main purpose of a proof of concept is to test the viability of a concept before investing significant resources in its full implementation.

In fact, this paper represents the first attempt to apply AutoML techniques to nowcasting of US GDP. This paper wants to be a turning point in applying machine learning techniques to predict time series data following the approach of Németh and Hadházi (2023), who investigated nowcasting with neural networks. More precisely, this paper represents a pioneering endeavor, being the first to apply the amalgamation of the two techniques (AutoML and Nowcasting) in the context of predicting GDP growth. As such, it serves as a seminal work, laying the groundwork and setting a precedent for future studies in this area.

Chapter 3 – Data

In this chapter, the paper begins with a comprehensive description of the data, covering its sources, key variables, and overall structure. Following this, an Exploratory Data Analysis (EDA) is explained to gain insights into underlying patterns and relationships within the dataset. The data is then preprocessed to address data quality issues and handle missing values.

3.1 Dataset

For the sake of replicability, the dataset used in this study is derived from two distinct underlying data sources, but both datasets are accessible through the same website, which is overseen and maintained by the reputable institution, the Federal Reserve of St. Louis. In fact, the Federal Reserve Economic Data ("FRED" hereafter) is an online database consisting of hundreds of thousands of economic data time series from scores of national, international, public, and private sources.

3.1.1 Target Variable

The target variable of this empirical analysis is the quarterly GDP growth, that is the percent change (relative to the preceding period) of the US Gross Domestic Product. The original source of the series is the U.S. Bureau of Economic Analysis and is available from 1947:Q2 to 2023:Q1. It contains 304 observations.

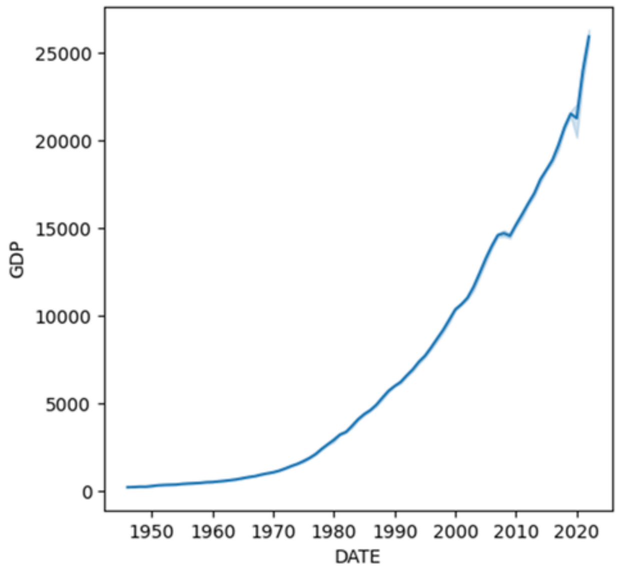


Figure 1: Quarterly GDP Line Plot.

3.1.2 Input Variables

The input features utilized in this study are sourced from the FRED-MD database, a monthly database for Macroeconomic Research established by the Federal Reserve Bank of St. Louis. Detailed information about this database can be found in McCracken and Ng's work of 2016. The FRED-MD database serves as a comprehensive repository of macroeconomic data, specifically tailored for the analysis of large-scale datasets. The utilization of the FRED-MD database is increasingly prevalent in economic applications because it not only encompasses datasets that have been previously employed in economic research but also introduces three crucial enhancements. Firstly, it is updated in real-time through the FRED database, ensuring the availability of the most current data for analysis. Secondly, it is made publicly accessible, facilitating replication of empirical studies by other researchers. Lastly, it relieves the burden on researchers of handling data changes and revisions, as these tasks are efficiently managed by the data desk at the Federal Reserve Bank of St. Louis.

It comprises 134 monthly time series, categorized into eight distinct groups, encompassing a wide range of indicators and features, all of which are publicly available and consistently updated on a monthly basis. These 134 time-series encompass various macroeconomic aspects, and can be grouped in the following categories:

1. Output and Income;
2. Labor Market;
3. Housing;
4. Consumption, Orders and Inventories;
5. Money and Credit;
6. Interest and Exchange Rates;
7. Prices;
8. Stock Market Data.

Thereby it offers an extensive array of variables for analysis and research purposes. complete list of the data and the series-related data transformations are described in detail in Appendix as well as the recent modifications. Monthly indicator data are available from 1960:M1 to 2023:M6, so we have 760 observations for the potential regressors at the moment in which this research is conducted.

3.2 Exploratory Data Analysis

The exploratory data analysis (EDA) provides valuable insights into the underlying patterns and characteristics of the dataset before delving into more advanced analyses. In this chapter, the data is thoroughly examined and visualized, uncovering potential trends and relationships.

3.2.1 Visualizing the Time Series

As in this research, a great number of time series are used, visualizing the raw data can help to understand the temporal patterns, trends, and fluctuations within a dataset over time. Line plots are used to achieve this. In line plots, the time points are plotted on the x-axis, and the corresponding values of the variable are plotted on the y-axis. The line connects the data points, revealing the trend and direction of change over time. A rising or falling trend in the line plot indicates the presence of an upward or downward trend in the data, respectively.

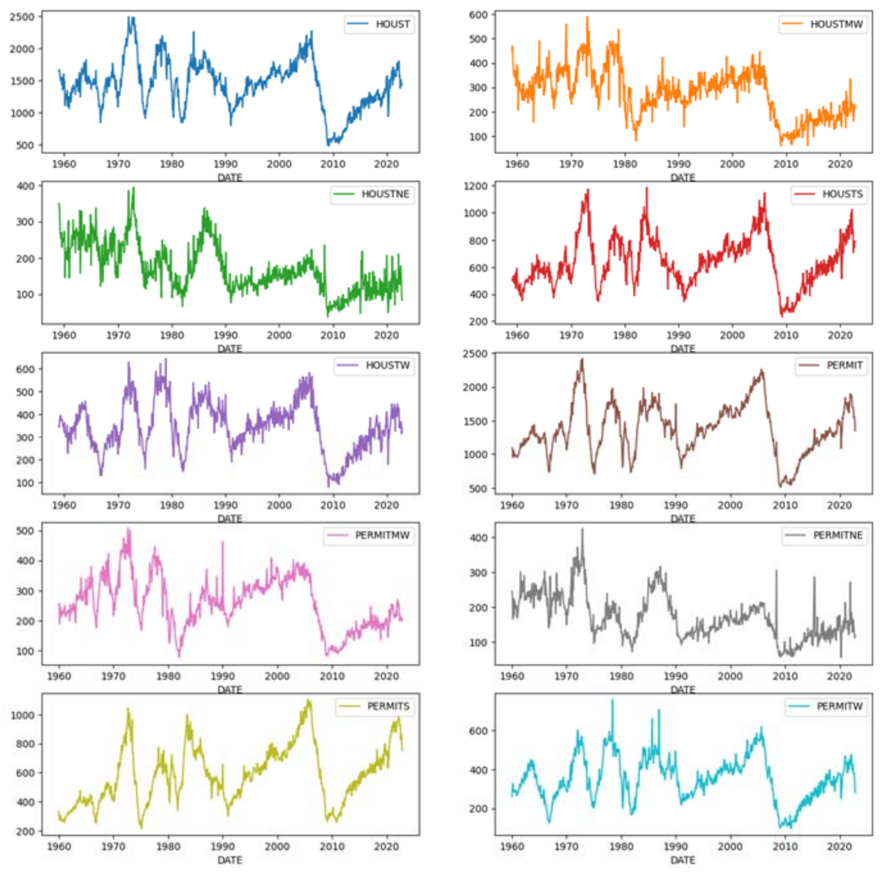


Figure 2: Monthly Line Plots of Housing Category (Group 3).

For example, line plots are particularly useful for a preliminary understanding of patterns within the category and also possible correlation the variables. In fact, the line plots for group 3 (Housing) clearly show patterns which consist in rising and declining about every

5 years with a critical downward in 2010. The graphs are similar to one another, and this could suggest the presence of high correlation between them. Findings from other groups are presented in the Appendix.

3.2.2 Checking Autocorrelation

Autocorrelation, also known as serial correlation, is a statistical concept used in time series analysis to measure the correlation between a time series and its lagged versions. In simpler terms, it quantifies the relationship between a data point at a particular time and its past observations. Aside the assumptions for linear models, autocorrelation helps to identify patterns and dependencies in the time series data. A significant autocorrelation at lag 1 indicates that the current data point is related to the previous data point. Furthermore, in a stationary time series, the autocorrelation between consecutive observations decays quickly to zero. On the other hand, the partial autocorrelation measures the direct influence of the k-th lag on the present value of the time series. The combination of the two might reveal that a series is white noise, and as such, it does not contain any information that could be used by the models.

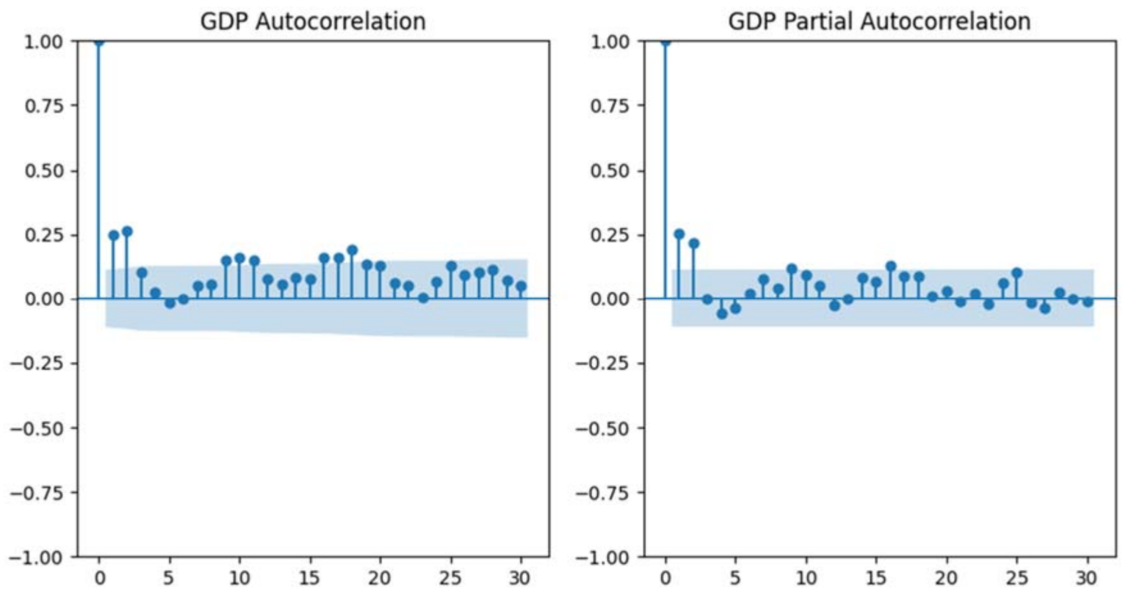


Figure 3: GDP Autocorrelation and Partial Autocorrelation Plots.

For example, the GDP autocorrelation suggests that the time series presents some kind of seasonality. Seasonality might be present due to several reasons such as economic activities seasonality, calendar effects, government policies and business cycles. The autocorrelation plots for the other variables are presented in the Appendix.

3.2.3 Checking Multicollinearity

Multicollinearity is a statistical phenomenon that occurs when two or more independent variables in a regression model are highly correlated with each other. In other words, multicollinearity exists when there is a strong linear relationship between two or more predictor variables, making it difficult for the model to distinguish the individual effects of each variable on the dependent variable. Multicollinearity is important in time series analysis for several reasons but the most important one regards the model interpretability and causal inference. In this context, the only focus is the prediction. Therefore, this analysis could have relevance in future works where feature engineering and selection might be performed.

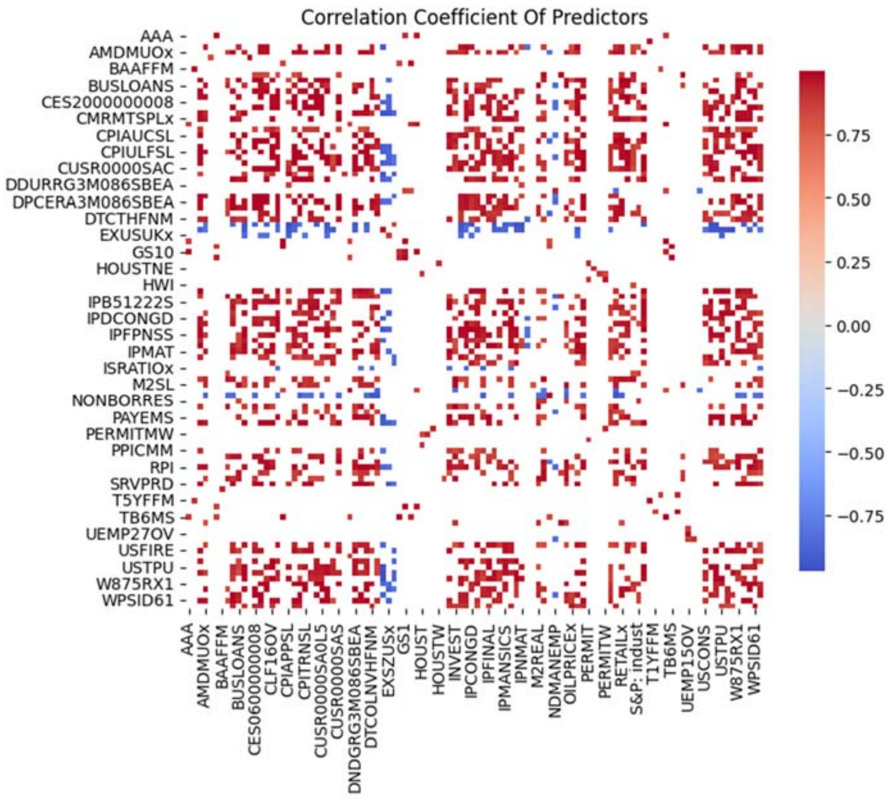


Figure 4: Absolute Highest Correlation Plot.

The correlation matrix plot above shows only the most relevant correlated relationships in an absolute point of view (i.e., the positive highest values and the negative highest values). For example, CMRMTSPLx, Real Manufacturing and Trade Industries Sales is negatively correlated with EXSZUSx, the Switzerland / U.S. Foreign Exchange Rate, suggesting an inverse relationship between the two variables. A stronger U.S. dollar makes U.S. products more expensive for Swiss consumers, reducing demand and impacting sales. Additionally, a stronger dollar makes imports cheaper, leading to increased reliance on foreign goods and decreased demand for domestic products. Companies heavily reliant on foreign markets may experience reduced profits due to exchange rate fluctuations. Economic conditions also influence the exchange rate and demand for exports. But again, this goes out-of-scope in the context of this analysis.

3.3 Preprocessing

As the datasets are imported raw, even though the series could be used right out-of-the-box, for this application, some preprocessing is needed. Preprocessing is a crucial step in machine learning that involves transforming raw data into a format suitable for training machine learning models. It is essential because raw data often contains noise, missing values, irrelevant information, and inconsistencies that can negatively impact the performance and accuracy of machine learning algorithms. Furthermore, as the work involves time series, transformations are also crucial. In fact, previous empirical studies found that on average better predictive performance can be expected when data is stationary and seasonally adjusted (Nemeth and Hadházi, 2023). In particular, the steps followed are presented below.

3.3.1 Imputing Missing Values

Missing values refer to the absence of data for one or more variables in a dataset. Handling missing values is an essential step in data preprocessing and analysis because they can affect the accuracy and validity of the results as well as different algorithms cannot handle missing, and therefore the search space of the best algorithms would be restricted. In this case, the main explanation for missing values is that some series have been being collected in more recent times compared to others. Imputing series that started too recently would involve artificially generating more than half of the records, which could render the imputed information meaningless because it would introduce not true underlying patterns and trend. In this situation, excluding these series from the analysis altogether seems to be the more appropriate approach in order to maintain the validity of the analysis.

For those series which have few missing data, K-Nearest Neighbors (KNN) imputer is used. This machine learning technique used to fill missing values has the advantage of dealing with datasets that have missing values in a non-random pattern, where the missingness might be related to the values of other variables. In particular, by considering missing values refer to the absence of data for one or more variables in a dataset. Handling missing values is an essential step in data preprocessing and analysis because they can affect the accuracy and validity of the results as well as different algorithms cannot handle missing, and therefore the search space of the best algorithms would be restricted. In this case, the main explanation for missing values is that some series have been being collected in more recent times compared to others. Imputing series that started too recently would involve artificially generating more than half of the records, which could render the imputed information meaningless because it would introduce not true underlying patterns and trend. In this situation, excluding these series from the analysis altogether seems to be the more appropriate approach in order to maintain the validity of the analysis.

For those series which have few missing data, K-Nearest Neighbors (KNN) imputer is used. This machine learning technique used to fill missing values has the advantage of dealing with datasets that have missing values in a non-random pattern, where the missingness might be related to the values of other variables. In particular, by considering the values of the nearest neighbors, the imputed values can better reflect the underlying structure of the data. Missing values refer to the absence of data for one or more variables in a dataset. Handling missing values is an essential step in data preprocessing and analysis because they can affect the accuracy and validity of the results as well as different algorithms cannot handle missing, and therefore the search space of the best algorithms would be restricted. In this case, the main explanation for missing values is that some series have been being collected in more recent times compared to others. Imputing series that started too recently would involve artificially generating more than half of the records, which could render the imputed information meaningless because it would introduce not true underlying patterns and trend. In this situation, excluding these series from the analysis altogether seems to be the more appropriate approach in order to maintain the validity of the analysis.

For those series which have few missing data, K-Nearest Neighbors (KNN) imputer is used. This machine learning technique used to fill missing values has the advantage of dealing with datasets that have missing values in a non-random pattern, where the missingness might be related to the values of other variables. In particular, by considering the values of the nearest neighbors, the imputed values can better reflect the underlying structure of the data.

3.3.2 Transforming Time Series to Stationary

The initial phase involves conducting an augmented Dickey-Fuller test to assess the stationarity of the series. The augmented Dickey-Fuller test is a statistical test used to determine if a time series is stationary or non-stationary. Stationarity is an important property in time series analysis, as it implies that the statistical properties of the series, such as mean and variance, do not change over time. In the context of the augmented Dickey-Fuller test, the null hypothesis assumes that the time series has a unit root, meaning it is non-stationary. The alternative hypothesis suggests that the series is stationary. The test computes a test statistic, and based on its value, a p-value is calculated. If the p-value is below a chosen significance level (commonly 0.05), the null hypothesis is rejected, indicating that the series is stationary. On the other hand, if the p-value is higher than the significance level, the null hypothesis cannot be rejected, and the series is considered non-stationary. The augmented Dickey-Fuller test is a widely used tool in econometrics and time series analysis to assess the stationarity of data and make informed decisions regarding data transformations or model selection.

If a series is found to be non-stationary, it undergoes a chosen transformation based on the approach proposed by McCracken (2021) which can be found in the Appendix.

Table 1: Augmented Dicky-Fuller Test Results for GDP Growth.

	Value
Test Statistic	-8.512509
P-value	0.0
Number of lags	1
Number of observations	302
Reject (signif. level 0.05)	True
Critical value 1%	-3.45219
Critical value 5%	-2.871158
Critical value 10%	-2.571895

The table above shows that GDP growth is stationary as we can fairly reject the null hypothesis. In fact, if we look at the line plot for the GDP growth and its distribution, we can confirm that indeed the series is stationary as the mean and the variance appears to be constant to a certain degree over time.

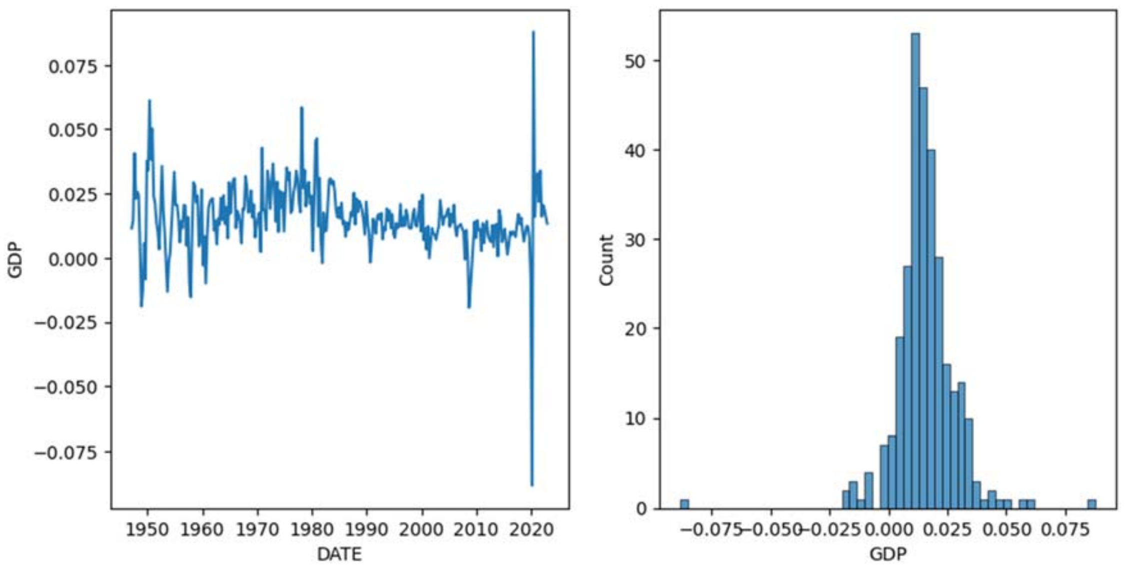


Figure 5: On the left, Line Plot of GDP Growth. On the right, Distribution Plot of GDP Growth.

3.3.3 Deseasonalizing Time Series

Subsequently, the series is decomposed and deseasonalized using either an additive or multiplicative model. Deseasonalization is a process in time series analysis that involves removing the seasonal component from a time series, leaving behind the underlying trend and irregular components. Seasonal components refer to the patterns or fluctuations that occur at regular intervals within a time series, such as daily, monthly, or yearly patterns. The purpose of deseasonalization is to focus on the underlying trend and irregular behavior of the data, which can be useful for various analytical purposes, such as forecasting, trend analysis, and identifying long-term patterns.

Deseasonalization can be performed using either an additive or a multiplicative model, depending on the nature of the seasonal component in the time series data. In an additive model, the seasonal component is added to the trend and irregular components. This means that the seasonal fluctuations have a constant amplitude throughout the time series. Mathematically, an additive model can be represented as:

$$Time\ Series = Trend + Seasonal + Irregular$$

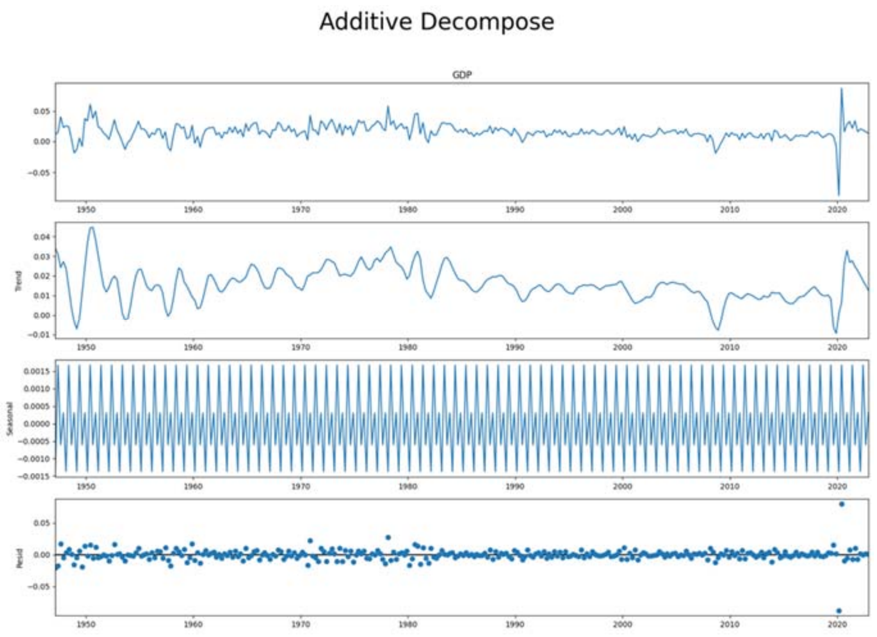


Figure 6: Additive Decomposition Plots.

In a multiplicative model, the seasonal component is multiplied by the trend and irregular components. This means that the amplitude of the seasonal fluctuations changes proportionally with the trend. Multiplicative models are more appropriate when the seasonal patterns exhibit a changing amplitude over time. Mathematically, a multiplicative model can be represented as:

$$Time\ Series = Trend * Seasonal * Irregular$$

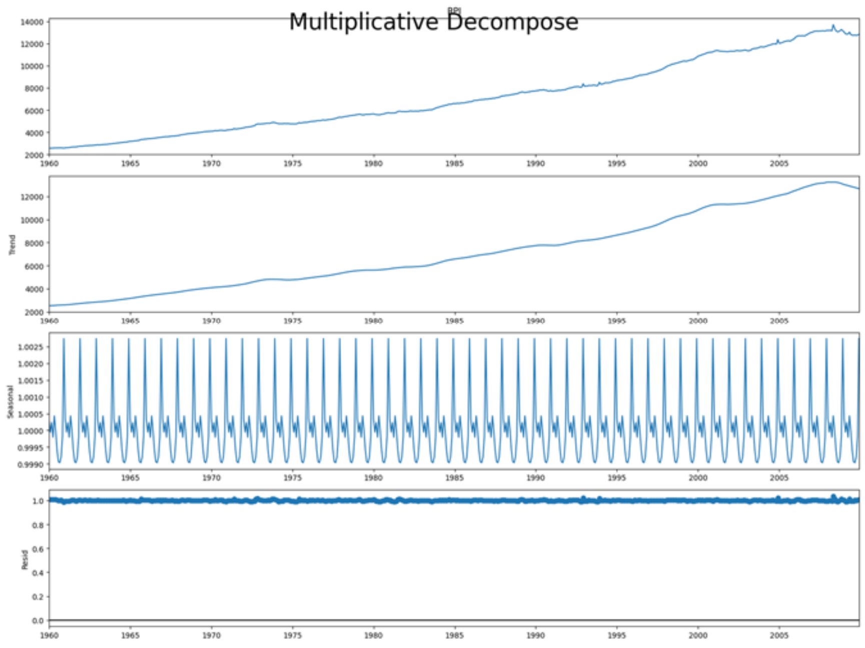


Figure 7: Multiplicative Decomposition Plots.

As a further step, in order to gain deeper insights into the underlying patterns of GDP, the Kalman filter is applied to eliminate noise resembling a smoothing technique. The smoothed line can be seen in the plot below.

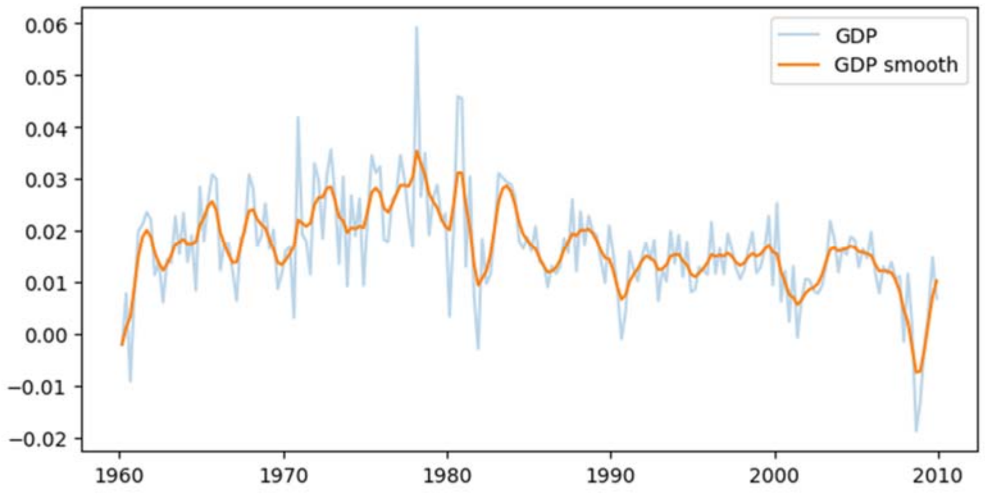


Figure 8: Line Plot of GDP Growth Smoothed With Kalman Filter.

3.3.4 Removing Outliers

Removing outliers is a crucial step in data preprocessing to ensure the accuracy and reliability of the analysis. Outliers are data points that significantly deviate from the majority of the data and can distort statistical measures and model performance. There are various methods to detect and remove outliers and amongst all, the most famous is the Z-Score or Standard Deviation method. However, in this case, the Interquartile Range (IQR) method is used. It is important to note that the decision to remove outliers should be made carefully and in consideration of the specific context and domain knowledge. Outliers may sometimes represent valid extreme values or important anomalies in the data that should not be discarded without proper justification. Additionally, removing too many outliers can lead to the loss of important information and negatively impact the analysis. Therefore, a balance should be struck between removing outliers for data integrity and preserving valuable information.

In fact, in this study, only the most extreme values that constitute a threat to the learning of the algorithms are removed. The IQR used for identifying outliers was [0.01, 0.99] another valid approach would be to multiply by 1.5 (or greater) the IQR [0.25, 0.75] to get the lower boundary and upper boundary. Subsequently, the outliers are removed, and the missing values are interpolated using the linear method. Linear interpolation is a data interpolation method used to estimate values between two known data points. It assumes a linear relationship between the data points and fills in the missing values along the straight line connecting these points.

3.3.5 Normalizing Time Series

Normalization is a data preprocessing technique used in machine learning to scale numerical features to a standard range, typically between 0 and 1 or -1 and 1. It is an essential step to ensure that all features contribute equally to the learning process and to avoid certain features dominating the model due to their larger magnitudes. Normalization is particularly useful when dealing with features that have different units or scales, as it brings all features to a common scale, making them comparable.

The most common method, and the one used in this context, for normalization is min-max scaling, which scales the features by subtracting the minimum value and dividing by the range (maximum value minus minimum value) of each feature. This ensures that all values are scaled between 0 and 1. It is important to note that normalization should be done separately for training and test datasets to avoid data leakage and ensure that the scaling is based solely on the training data.

Chapter 4 – Methodology

This chapter outlines the research design and approach used to address the research objectives. In particular, the research design, data analysis techniques and frameworks employed in our nowcasting experiment are presented.

4.1 Automated Machine Learning Modes and Frameworks

AutoML refers to the automated process of solving machine learning tasks with minimal or no manual intervention (Hu, Huang, 2017). Its purpose is to enable non- experts to apply machine learning techniques for specific tasks without requiring prior technical or domain knowledge (Feurer, Eggensperger, Falkner, Lindauer, Hutter, 2018). The main goal of AutoML is to fully automate the steps of model selection, hyperparameter optimization, and feature selection, which were previously tackled separately but are now integrated into fully automated approaches (Mohr, Wever, Hüllermeier, 2018).

AutoML frameworks are able to handle the complete data pipeline and optimize it. An AutoML data pipeline is a comprehensive and automated workflow that handles the entire process of data preparation, feature engineering, model selection, hyperparameter optimization, and model evaluation for a machine learning task. The data pipeline consists of different steps:

1. Data Preprocessing;
2. Feature Engineering;
3. Feature Selection;
4. Model Selection;
5. Hyperparameter Optimization;
6. Model Training and Evaluation;
7. Model Deployment.

In this research, the focus is on automating and optimizing three crucial components of the machine learning pipeline: model selection, hyperparameter optimization, and model training and evaluation. These are critical steps in building effective predictive models and are automated through the use of AutoML techniques. By automating these key steps, the research aims to simplify and streamline the nowcasting process, making it more efficient and accessible to users without requiring them to have in-depth expertise in machine learning. The automated approach reduces the need for manual intervention, allowing researchers to focus more on interpreting the results and gaining insights from the nowcasting predictions.

4.1.1 Model Selection

In the context of machine learning, model selection involves the process of identifying the most suitable model from a set of candidate models for a given dataset. The primary objective is to find the model that achieves the highest accuracy when trained and tested on the dataset. AutoML takes on the task of model selection without requiring human intervention. It automatically explores and evaluates various models using the same input data and then selects the model that demonstrates the best performance. This automated approach allows AutoML to iterate through different models and choose the one that best fits the data, streamlining the process of model selection and saving valuable time and effort (Waring, Lindvall, Umeton, 2020).

4.1.2 Hyperparameter Optimization

Hyperparameter optimization (HPO) is a fundamental aspect of machine learning model development, as the choice of hyperparameters can significantly impact the performance and generalization of the model (Mantovani, Horváth, Cerri, Vanschoren, de Carvalho, 2016). HPO involves finding the best combination of hyperparameter values that result in the optimal model performance for a given task and dataset. Properly tuned hyperparameters can lead to improved accuracy, faster convergence during training, and better resistance to overfitting (Yu, Zhu, 2020). Traditional approaches to hyperparameter optimization involved human experts manually tweaking hyperparameters based on their knowledge and experience. While this approach can be effective to some extent, it may not always yield the most optimal results, especially with the increasing complexity of modern machine learning models. Additionally, manual hyperparameter tuning can be time consuming and resource-intensive, hindering the scalability and efficiency of the model development process.

Automated hyperparameter optimization (HPO) has emerged as a powerful solution to address these challenges. HPO algorithms automate the process of exploring the hyperparameter space, searching for the best combination of hyperparameters without requiring extensive human intervention. These algorithms leverage various search strategies, sampling techniques, and optimization methodologies to efficiently navigate the high-dimensional hyperparameter space and find promising configurations. Among the popular automated HPO methods are Grid Search and Random Search. Grid Search exhaustively explores predefined hyperparameter combinations (Elmasdotter, Nyströmer, 2018) (Liashchynskyi, 2019) (Lerman, 1980), while Random Search randomly samples hyperparameters to evaluate different configurations (Bergstra, Bengio, 2012) . While both approaches have their merits, they can be computationally expensive and may not be suitable for large-scale hyperparameter search. Bayesian optimization has gained significant attention in recent years due to its efficiency and effectiveness in finding optimal hyperparameters (Snoek, Larochelle, Adams, 2012). This approach leverages a probabilistic model to model the performance landscape and intelligently select hyperparameter configurations for evaluation. Bayesian optimization can outperform Grid Search and Random Search by efficiently exploring the hyperparameter space, quickly converging to promising regions, and making informed decisions based on past evaluations. Another notable technique is Genetic Algorithms, inspired by the principles of natural selection and evolution. Genetic Algorithms maintain a population of potential hyperparameter configurations, iteratively evolving and selecting the most promising candidates based on their fitness function, which measures the model's performance (Goldberg, 1989). This evolutionary process mimics the survival of the fittest and has shown success in optimizing complex hyperparameter spaces. By leveraging the power of automated HPO, models can achieve higher accuracy, robustness, and adaptability, enhancing their performance across various tasks and datasets.

4.1.3 AutoML Frameworks

In recent years, the field of AutoML has witnessed significant advancements with the development of various systems that integrate model selection, hyperparameter optimization, and feature engineering into a unified framework. Some of the notable AutoML frameworks that have emerged include H2O, Auto-Sklearn, AutoGluon, TPOT, Auto-Weka, TSPO, AutoKeras and EvalML, among others. Each of these frameworks brings its unique set of features, methodologies, and algorithms to automate the different components of the machine learning process. In the following paragraphs, we will provide a comprehensive review of these AutoML frameworks, exploring their strengths, limitations, and areas of applicability. Through this comparison, we aim to shed light on the diverse options available for automating machine learning and assist researchers and practitioners in selecting the most suitable AutoML framework for their specific needs and use cases. It has to be noted that each framework requires the data to be in a specific format and has specific dependencies, therefore it might be challenging to make all packages co-exist in the same environment. In this instance, these challenges were manually solved installing the packages from their sources and transforming the dataset each time.

4.1.3.1 PyCaret

PyCaret is a popular AutoML library that has gained significant attention in the machine learning community. It is an open-source and low-code Python library designed to simplify the end-to-end machine learning workflow. PyCaret offers a wide range of functionalities, making it a comprehensive solution for automating various tasks in machine learning, such as data preprocessing, model selection, hyperparameter tuning, and model evaluation. One of the key features of PyCaret is its easy-to-use API, which allows users to perform complex machine learning tasks with just a few lines of code. It supports both supervised and unsupervised learning tasks, making it versatile for different types of problems. PyCaret also provides support for various types of data, including structured, unstructured, and time series data.

PyCaret integrates with popular machine learning libraries, such as scikit-learn, XGBoost, LightGBM, CatBoost, and many others, enabling users to take advantage of a wide range of algorithms and models. It automates the process of feature engineering, handling missing values, and dealing with categorical variables, reducing the need for manual preprocessing steps. The library offers a range of evaluation metrics and visualization tools to assess model performance and make informed decisions. Additionally, PyCaret allows users to compare multiple models, select the best-performing one, and export the final model for deployment. PyCaret also provides support for ensemble methods, allowing users to combine multiple models to improve overall predictive performance. It simplifies the process of ensembling, making it accessible to users without extensive knowledge of ensemble techniques.

In particular, PyCaret has three modules or use-cases: Supervised ML, Unsupervised ML and Time Series. The main difference between them is the algorithms considered in the model selection. In this application, the Time Series module was used as it applies time series specific algorithms such as ARIMA, Theta, ETS and etc.... This module also allows to construct and visualize different experiments with tailored parameters as well as generate predictions along with a line plot.

These are the usual steps in setting up PyCaret:

1. Setup the experiment through the Functional API or the Object-Oriented API;
2. Compare the models inside the specific module;
3. Create the final model with the selected model (usually the best one);
4. Optimize the selected metric by tuning the model hyperparameters;
5. Optionally run diagnostics in order to analyze the model;
6. Finalize the selected model training it on the entire dataset;
7. Make predictions.

4.1.3.2 AutoGluon

AutoGluon used by Erickson (2020) is an advanced open-source AutoML library developed by the MXNet community. It is designed to automate and simplify the end-to-end machine learning process, making it accessible to users with varying levels of expertise. AutoGluon is built on top of MXNet, a deep learning framework, and supports both traditional machine learning algorithms and deep learning models. One of the key strengths of AutoGluon is its ability to handle a wide range of tasks, including classification, regression, and time series forecasting. It offers a high-level API that allows users to easily train and deploy complex machine learning models without the need for extensive coding.

AutoGluon is known for its exceptional performance and efficiency in hyperparameter optimization. It employs advanced optimization techniques, such as Bayesian optimization and gradient-based methods, to automatically find the best hyperparameter settings for the models. This helps users achieve optimal performance without the need for manual tuning. The library supports ensemble methods, enabling users to create powerful model ensembles with just a few lines of code. AutoGluon intelligently combines multiple models to improve predictive accuracy and generalization. It automatically handles time series upon transforming the input into the accepted format, AutoGluon Time Series Dataframe. AutoGluon also offers a user-friendly dashboard and visualization tools that allow users to easily inspect and analyze the results of the AutoML process. A particular useful feature is the possibility to visualize a leaderboard of models for each prediction. This helps users gain insights into the models and make informed decisions. Moreover, AutoGluon is scalable and can be efficiently run on both local machines and cloud environments. It is well-suited for large-scale machine learning tasks, making it ideal for handling complex and extensive datasets.

4.1.3.3 AutoKeras

AutoKeras released by Jin H, Song Q and Hu X (2019) is an open-source AutoML framework developed by researchers of the DATA Lab at the University of Texas at

Austin. It is built on top of the Keras deep learning library and TensorFlow, making it capable of handling both traditional machine learning tasks and deep learning applications.

The main objective of AutoKeras is to automate the process of model selection and hyperparameter tuning, allowing users to build high-performing machine learning models with minimal manual effort. It is designed to be user-friendly, making it accessible to users with limited expertise in machine learning. AutoKeras uses a neural architecture search (NAS) algorithm to automatically explore and discover the best neural network architecture for a given dataset and task. The NAS process involves efficiently searching through a large space of possible neural network architectures to find the optimal one for the specific problem at hand. One of the key features of AutoKeras is its support for automatic hyperparameter tuning. It leverages Bayesian optimization techniques to automatically find the best hyperparameter settings for the neural network, such as the number of layers, the number of neurons in each layer, learning rate, and batch size. This helps users achieve optimal model performance without the need for manual tuning. AutoKeras also supports automatic data preprocessing and feature engineering. It can handle common data preprocessing tasks, such as handling missing values and scaling features, to ensure that the data is properly prepared for model training. Additionally, AutoKeras supports various types of ML tasks, including image classification, text classification, and structured data regression. It can automatically select appropriate models and architectures for each task, making it versatile and suitable for a wide range of applications. In this case, the Time Series Forecaster is used. Furthermore, AutoKeras provides a user-friendly API and a simple command-line interface, making it easy for users to interact with the framework and train models without writing complex code.

4.1.3.4 Fast and Lightweight AutoML (FLAML)

FLAML (Fast and Lightweight AutoML) is an open-source AutoML framework designed for fast and efficient model selection and hyperparameter optimization. It is developed by Microsoft Research and offers a lightweight yet powerful solution for automating the machine learning pipeline.

One of the key features of FLAML is its speed and efficiency. It is built to handle large datasets and complex machine learning tasks with minimal computational resources. FLAML achieves this efficiency through several techniques, including early stopping, dynamic early stopping, and adaptive resource allocation. These techniques allow FLAML to quickly explore different models and hyperparameter configurations and stop the search early if a promising solution is found. FLAML supports a wide range of machine learning tasks, including classification, regression, and recommendation systems. It includes a variety of state-of-the-art algorithms, such as random forest, gradient boosting, and deep learning models, making it versatile and suitable for diverse applications. The framework provides automatic model selection, where it intelligently chooses the best algorithm and model based on the input data and task. Additionally, FLAML conducts hyperparameter optimization using advanced techniques like Bayesian optimization and Simulated Annealing. This ensures that the selected model is fine-tuned for optimal performance on the specific dataset. Furthermore, FLAML is easy to use, with a simple API that allows users to quickly set up and run the AutoML process. It provides detailed logs and performance metrics, enabling users to track the progress and results of the automated search. FLAML is compatible with popular machine learning libraries like scikit-learn and XGBoost, making it seamlessly integrate into existing workflows and projects, and as an interesting feature, the search space can be customized excluding specific models.

4.1.3.5 H2O

H2O created by LeDell and Poirier (2020) is a popular open-source AutoML platform developed by H2O.ai. It is designed to streamline the process of building and deploying machine learning models at scale. H2O provides a user-friendly interface and supports a wide range of algorithms and data types, making it a versatile tool for data scientists and analysts.

One of the key features of H2O is its ability to handle big data efficiently. It can process large datasets in a distributed manner, utilizing multiple nodes in a cluster for parallel processing. This allows H2O to handle massive datasets that may not fit into the memory of a single machine, making it suitable for big data applications. H2O offers automatic model selection and hyperparameter optimization, allowing users to simply input their data and let the platform find the best-performing model with optimal hyperparameters. It supports various machine learning algorithms, including random forest, gradient boosting, deep learning, and more. The platform also provides model interpretability tools, allowing users to understand how models make predictions and gain insights into the relationships between features and outcomes. This is crucial for building trust in the model's decisions and ensuring transparency in the model's behavior. H2O supports various programming languages such as Python, R, and Java, making it accessible to users from different technical backgrounds. It also offers APIs for integration with other popular machine learning libraries like scikit-learn and XGBoost. Another advantage of H2O is its scalability and performance. It is designed to efficiently utilize the hardware resources available, enabling faster model training and inference times. This scalability makes H2O suitable for both small-scale projects and large-scale enterprise applications.

One major drawback of H2O is that it is not completely open source and free. H2O offers a 360-degrees ai platform which upon a subscription can be very powerful. Unfortunately, in the “free” version, there is no original support for time series data, therefore, the regression module is used removing the time from the inputs. Another drawback is that the package support only the specific H2O framework format which is built onto pandas.

4.1.3.6 Tree-Based Pipeline Optimization Tool (TPOT)

TPOT (Tree-based Pipeline Optimization Tool) created by Trang, Fu and Moore (2020) is an automated machine learning (AutoML) library developed by Epistasis Lab at the University of Pennsylvania. It is designed to automatically create and optimize machine learning pipelines for a given dataset, without requiring extensive manual intervention from users. TPOT is built on top of scikit-learn.

TPOT employs a genetic programming approach to evolve a population of pipelines that consist of various data preprocessing steps and machine learning models. It uses tree-based genetic operators such as mutation and crossover to create new pipelines by combining and modifying existing ones. The pipelines are evaluated based on their performance on a given objective function, typically accuracy or mean squared error, and the best-performing pipelines are retained for the next generation. The library supports a wide range of data preprocessing steps, such as feature selection, scaling, imputation, and transformation, as well as various machine learning models, including decision trees, random forests, support vector machines, and more. TPOT automatically selects the most appropriate preprocessing steps and models based on the dataset, thus saving users from the tedious process of trying different combinations manually. TPOT also performs hyperparameter optimization to fine-tune the parameters of the selected models, further improving their performance. It uses a technique called genetic programming with Pareto optimization to efficiently search the hyperparameter space and find the best combination of parameters. One of the advantages of TPOT is its ability to handle both classification and regression tasks, making it versatile for various types of predictive modeling problems. Additionally, TPOT provides built-in support for multiclass classification and handles imbalanced datasets by applying techniques such as oversampling and undersampling. TPOT is implemented in Python and can be easily integrated into existing machine learning workflows. It provides a simple and intuitive API, allowing users to quickly get started with AutoML without the need for advanced coding skills.

Compared to the other TPOT is more complicated to run. In fact, genetic algorithms require different initial parameters such as generations, population size, mutation rate and crossover rate. These parameters must be initially set by the user and there is no standard in the literature other than empirical results from previous studies in the related field which implies a trial-and-error procedure to fine tune the values. In addition, TPOT is not meant to run for few minutes. If TPOT is not run long enough, it may not find the best pipeline, if at all. The typical TPOT runs can be time consuming, especially for large and complex datasets. The time taken to finish a TPOT run can range from hours to days, depending on the size of the dataset and the complexity of the problem. However, TPOT provides the option to interrupt the run and review the best results so far.

4.1.3.7 EvalML

EvalML is an automated machine learning (AutoML) library developed by Alteryx, designed to streamline, and simplify the process of building and evaluating machine learning models. It is built on top of scikit-learn, a popular machine learning library in Python, and offers a high-level API that allows users to quickly build and evaluate models without extensive manual coding.

EvalML is capable to automatically handle data preprocessing tasks as well as missing data, encode categorical features, and perform feature engineering without requiring explicit user intervention. This is particularly useful for users who are not familiar with the intricacies of data preprocessing, as EvalML takes care of these steps behind the scenes. Furthermore, EvalML supports a wide range of machine learning models, including classification, regression, and time series models. It uses a pipeline-based approach to build models, allowing users to easily combine different preprocessing steps and models to create complex pipelines. The library also automatically performs hyperparameter optimization to fine-tune the models and improve their performance. Another important feature of EvalML is its ability to handle imbalanced datasets, which are common in many real-world applications. It uses techniques such as oversampling, undersampling, and class weighting to address the class imbalance problem and ensure that the models are trained on balanced data. EvalML provides built-in support for model evaluation and performance metrics, making it easy for users to assess the quality of the models. It also supports time series cross validation, which is essential for evaluating the performance of models on time dependent data.

EvalML is great for time series analysis as it supports three time series specific problem types, regression, multiclass and binary. It also supports a considerable number of metrics which can be used to visualize the ranking of all models. Furthermore, EvalML gives the best pipeline used by the best model with all the specific parameters. In addition to its user-friendly API, EvalML offers extensive documentation and tutorials to help users get started with AutoML and understand the best practices for building and evaluating models. It is well-maintained and actively developed, with regular updates and improvements based on user feedback and the latest advancements in the field of machine learning.

4.1.3.8 Mljar

Mljar created by Plonska and Plonski (2021) is an AutoML platform that aims to simplify the process of building and deploying machine learning models. It is designed to automate various stages of the machine learning workflow, including data preprocessing, feature engineering, model selection, hyperparameter tuning, and model evaluation.

Users can upload their dataset and select the target variable they want to predict, and Mljar will handle the rest of the process, automatically suggesting and training various machine learning algorithms. Mljar offers a wide range of machine learning algorithms, including decision trees, random forests, gradient boosting, support vector machines, and neural networks. It uses a combination of Bayesian optimization and hill climbing algorithms to efficiently search through the hyperparameter space and find the best-performing model for the given dataset. Another important aspect of Mljar is its focus on transparency and interpretability. It provides users with detailed explanations of the model selection and hyperparameter tuning process, making it easier to understand how the final model was chosen. This is especially important in applications where model interpretability is crucial, such as in healthcare or finance. Mljar also allows users to deploy their trained models to production environments easily. It provides APIs and integrations with popular deployment platforms, making it seamless to put the models into action. In fact, Mljar is a little bit different from other packages in the sense that it offers numerous predefined features and settings that makes it highly customizable but still relatively simple. For example, the training part is divided in predefined steps which are common in the process of searching the best ML pipeline. It starts with simple algorithms, then it uses more complicated models with standard hyperparameters, and successively it tunes these parameters. In addition, Mljar provides three different AutoML modes: explain, perform and compete. Explain is to be used when the user wants to explain and understand the data. Perform is to be used when the user wants to train a model that will be used in real-life use cases. Compete to be used for machine learning competitions which focuses on performance.

4.1.3.9 Other ML Packages

AutoML has gained significant attention and popularity in recent years. As a result of the broad application of this technique, researchers and companies have been actively developing their own AutoML frameworks and packages to cope with different needs and specific use cases.

This surge in AutoML development has led to a wide variety of solutions available in the market. Each framework may offer unique features, optimization algorithms, and model architectures, providing users with a diverse set of options to choose from based on their requirements and preferences. In this study, most of the most popular packages are tested, but other are still available such as Auto-Sklearn (Feurer et al., 2015), TransmogrifAI by Salesforce, Auto-Weka (Lars Kotthoff, Chris Thornton, Frank Hutter, 2017), Auto-Pytorch (Lucas Zimmer, Marius Lindauer, Frank Hutter, 2020), AutoTS (Chunnan Wang, Xingyu Chen, Chengyue Wu, Hongzhi Wang, 2022) and AutoViML.

4.2 Most Used Algorithms

In this subchapter, the focus is on introducing and discussing some of the most commonly used algorithms employed for nowcasting US GDP growth by AutoML packages.

4.2.1 Statistical Base Models

Two primary models will be presented, namely the ARIMA model and the DFM (Dynamic Factor Model) proposed by Giannone et al. (2008) along with the simplest method, the naïve forecaster. These models will serve as statistical benchmarks against which the performance of other advanced models will be evaluated.

4.2.1.1 Autoregressive Integrated Moving Average (ARIMA)

The ARIMA (Autoregressive Integrated Moving Average) model is a widely used time series forecasting method that captures the autoregressive and moving average properties of the data. It is particularly suitable for stationary time series and is capable of handling trend and seasonality in the data. The ARIMA model involves three main components, namely autoregressive (AR), differencing (I), and moving average (MA), which are combined to create a forecasting model. The formulation of the ARIMA model and its application to nowcast US GDP growth.

The model is relatively easy to interpret, as presented below:

$$y_t = \Phi_1 y_{t-1} + \dots + \Phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

φ represents the autocorrelation values and θ the moving average parameters. ε_t is the residual term. The order of the autoregressive part p and the order of the moving average part q will be based on the Bayesian Information Criterion (BIC). Note that no constant is needed since the data has been standardized. The AR component, *p*, captures the relationship between the current observation and its lagged values, while the MA component, q, models the relationship between the current observation and its past forecast errors. The differencing component is used to make the time series stationary, which means it removes any trend or seasonality present in the data.

The ARIMA model is a powerful tool for time series forecasting, and its application to nowcast US GDP growth allows for the extraction of short-term patterns and trends from historical data. However, it's essential to keep in mind that the ARIMA model assumes linearity and stationarity in the data, which may not always hold true for economic time series. Therefore, the performance of the ARIMA model will be compared to other advanced models, such as machine learning algorithms and AutoML frameworks, to determine the most accurate and reliable method for nowcasting US GDP growth.

4.2.1.2 Dynamic Factor Model (DFM)

The Dynamic Factor Model is a statistical method that utilizes latent factors to capture the common underlying patterns in a large set of economic indicators. It is a powerful tool for nowcasting as it effectively combines information from multiple time series, allowing for improved forecasting accuracy. The DFM proposed by Giannone et al. (2008) has been extensively used for macroeconomic forecasting and is considered one of the state-of-the-art methods for nowcasting in real-time.

It assumes the presence of unobservable latent factors that influence the entire group of variables, representing shared patterns or trends. These latent factors drive the behavior of the observed time series. Each observed time series is related to the latent factors through factor loadings. The factor loadings represent the strength and direction of the relationship between each time series and the underlying factors. They determine how much each time series is influenced by the common trends represented by the latent factors. Additionally, the model accounts for individual variability or noise in each series using idiosyncratic errors. These errors capture the individual variability or noise specific to each time series that is not explained by the common factors. Notably, the DFM allows for dynamic relationships, enabling factor loadings and errors to change over time and capture evolving patterns in the data.

The formulation of the DFM can be represented as follows:

$$Y(t) = \mathit{\Lambda} F(t) + \varepsilon(t)$$

Y(t) is a vector of observed economic indicators at time t. Λ is the factor loading matrix, representing the relationship between the latent factors and the observed indicators. F(t) is a vector of latent factors at time t, capturing the common movements among the observed indicators. ε(t) is the vector of idiosyncratic errors, representing the unique movements of each observed indicator that are not captured by the common factors.

The DFM assumes that the latent factors follow a dynamic process, allowing them to change over time. This dynamic nature enables the model to capture shifts and changes in the underlying economic conditions that may affect the observed indicators.

4.2.1.3 Naïve Forecaster

The naïve forecaster is one of the simplest methods to predict future values, and it is based on naive assumptions about past trends continuing. Though very simple, it can yield reasonable results when applied to economic scenarios. However, it has limitations, particularly when dealing with time series data in dynamic and seasonal environments as it assumes that the future values will be the same as the most recent observed value, which may not be accurate in cases where the underlying patterns are not constant. A partial solution to this problem is a slightly more complicated model called Seasonal Naïve Forecaster. In the seasonal naive forecaster, the forecast for a future time point is based on the value observed in the same season of the previous year.

4.2.2 Machine Learning (ML) Models

In this subsection, the focus will be on discussing Machine Learning (ML) models proposed to outperform the statistical benchmarks, namely the ARIMA model and the Dynamic Factor Model (DFM), in nowcasting US GDP growth. ML models offer a more flexible and data-driven approach to nowcasting, leveraging their ability to capture complex patterns and nonlinear relationships present in the economic data.

4.2.2.1 Random Forest (RF)

Random Forest (RF) is a powerful machine learning algorithm that belongs to the family of ensemble methods and was proposed by Breiman (2001). Ensemble methods combine multiple individual models to improve the overall predictive performance and generalization ability. In the case of Random Forest, these individual models are Decision Trees.

Decision trees, as described by Gordon et al. (1984) in their book and depicted in the figure below, are powerful algorithms that partition the predictor space into distinct and non-overlapping regions, achieved by minimizing an objective function called the residual sum of squares (RSS). According to Breiman et al. (2017), decision tree models can effectively capture complex and non-linear relationships between independent variables and the dependent variable.

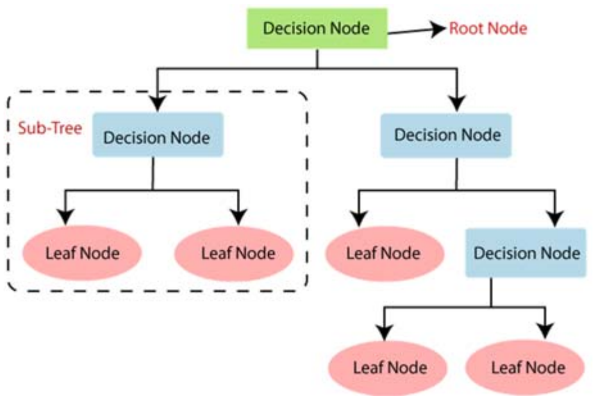


Figure 9: Simplified Illustration of Decision Trees.

Indeed, while decision trees are easy to interpret and provide a clear structure, they can suffer from limitations in prediction accuracy. First of all, decision trees can be computationally expensive to train, particularly on large datasets, and they are prone to overfitting, which means they may perform well on the training data but poorly on unseen data. Additionally, decision trees can get stuck in local optima, meaning they may not reach the optimal solution during the learning process. These shortcomings highlight the importance of using ensemble tree models like Random Forest, which can overcome some of these issues.

The key idea behind Random Forest is to create multiple decision trees, each trained on a random subset of the data and a random subset of the features. This process is called bootstrap sampling and feature randomization. A tree structure will be created by dividing the variable space F by selecting the most important variable j and its optimal split value is considering the objective function and splits the space accordingly. By doing so, each tree in the forest becomes slightly different, introducing diversity among the trees. This diversity is crucial because it helps to reduce the risk of overfitting, which can occur when individual decision trees memorize the training data too well and perform poorly on new, unseen data. The training process of a single decision tree involves recursively partitioning the data into subsets based on the values of different features. At each node of the tree, the algorithm selects the best feature to split the data, based on criteria such as Gini impurity for classification tasks or mean squared error for regression tasks. This process continues until a stopping criterion is met, such as a maximum depth for the tree or a minimum number of data points in a leaf node. This process can be described by these the following equations:

$$R_1(j,s) \ = \ \{X|X_j \leq s\} \, and \ R_2(j,s) \ = \ \{X|X_j > s\}, \qquad R_1, R_2 \ \in \ \mathrm{F}.$$

$$\min_{j,s}[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_2)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2$$

Where the first equation represents the pair of regions R1 and R2, obtained by splitting the variable j at value s. X are then assigned to R1 or R2, depending on its value. The variable j and value s are determined by the second equation. C is a constant.

Once all the individual trees are trained, the final prediction from the Random Forest is obtained by aggregating the predictions of each tree. For classification tasks, the most popular class predicted by the trees is taken as the final output (voting). For regression tasks, the average of the predicted values from all the trees is used as the final prediction.

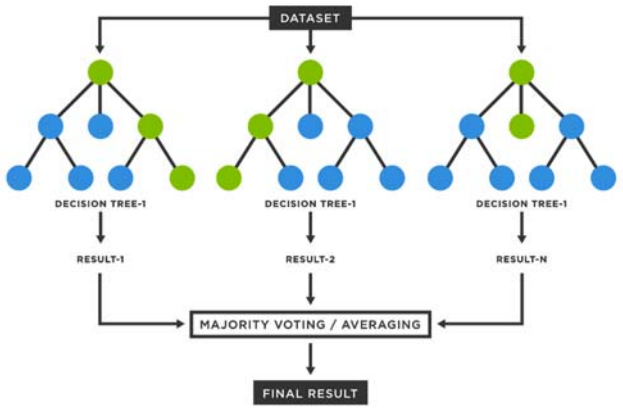


Figure 10: Simplified Illustration of Random Forest.

Random Forest is a machine learning algorithm that is well-suited for nowcasting due to its ability to handle time series data and capture complex relationships between variables. In fact, one of the key advantages of Random Forest is its ability to model non-linear relationships which characterize economic data. In nowcasting, dealing with a large number of potential predictor variables is common. Random Forest can provide estimates of feature importance, indicating which variables are most relevant for making predictions. Random Forest can handle missing data without requiring imputation, which simplifies the preprocessing steps and saves time during the nowcasting process. Moreover, Random Forest is robust to outliers and noisy data, which are common challenges in economic datasets. Additionally, the ensemble approach of Random Forest leads to improved generalization and reduced overfitting, which is crucial when working with economic data, which may have limited samples and inherent noise. Scalability is another advantage of Random Forest, making it suitable for analyzing extensive economic datasets with a high frequency of updates. The algorithm can handle large datasets efficiently and is capable of producing predictions quickly, making it suitable for real-time nowcasting applications where up-to-date information is crucial.

A very similar ensemble method to Random Forest is Extra Trees. Extra Trees stands for Extremely Randomized Trees. They differ as the latter extra randomness to the decision tree building process. While Random Forests already use bootstrapping (randomly sampling the data with replacement) and a random subset of features for each split, Extra Trees goes a step further by randomly selecting the splits at each node, rather than searching for the best one. This additional randomness results in Extra Trees having higher bias, but lower variance compared to Random Forests. In other words, Extra Trees may make more errors on the training data but can be less sensitive to small variations in the data. This can make them more robust, especially when dealing with noisy or limited datasets. Another advantage of Extra Trees is their computational efficiency. Because they don't perform an exhaustive search for the best split at each node, they are faster to train, making them a good choice for large datasets or situations where computational resources are limited.

4.2.2.2 Extreme Gradient Boosting (XGBoost)

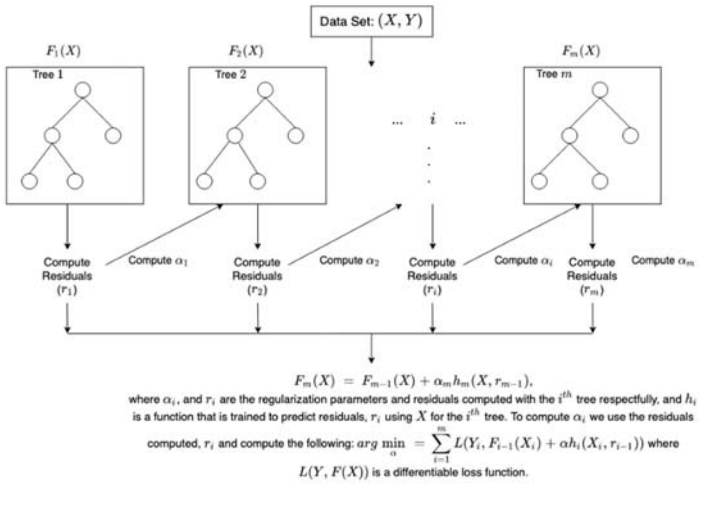
XGBoost (Extreme Gradient Boosting) is an advanced machine learning algorithm used for supervised learning tasks, particularly in regression and classification problems. XGBoost is an open-source package originally created by Chen & Guestrin (2016), and it is an advanced application of the gradient boosting machine introduced by Friedman (2001). Chen & Guestrin (2016) refer to XGBoost as an optimized version of the gradient tree boosting model, benefiting from hardware and software optimizations. This results in superior performance with fewer computing resources and reduced processing time compared to other ensemble tree models.

Unlike Random Forest (RF), which uses bagging, XGBoost employs boosting. The key difference is that boosting learns from the errors of previous trees, while bagging uses independent parallel trees. Boosting creates a committee of "weak" learners to generate a powerful estimation, where trees continuously learn from the errors of previous ones until further improvements are not possible. XGBoost employs gradient boosting, utilizing a gradient descent algorithm to minimize the loss while adding trees. Gradient descent is a technique that iteratively adjusts parameters to find the vector of coefficients that minimizes the loss function. By applying gradient boosting, XGBoost builds a forest of trees additively, leading to an optimal set of regression trees. According to Chen & Guestrin (2016), XGBoost builds trees by minimizing the loss function:

$$\mathcal{L}(\emptyset) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

$$\text{where } \Omega(f_k) = \gamma T + \frac{1}{2}\|w\|^2$$

Ω is the regularization function that penalizes the regression trees' complexity, which avoids overfitting. γ represents the penalty to encourage pruning, a technique that reduces the size of regression by removing sections of the tree that only contribute slightly. T represents the number of leaves per tree, and w is the sum of the output of the corresponding leaves. ||w||^2 represents the L2-norm of leaf scores. Furthermore, since XGBoost makes use of boosting, the algorithm incorporates the ability to learn iteratively from the previous tree's errors.



$$F_m(X) = F_{m-1}(X) + \alpha_m h_m(X, r_{m-1}),$$

where α_i, and r_i are the regularization parameters and residuals computed with the i^{th} tree respectfully, and h_i is a function that is trained to predict residuals, r_i using X for the i^{th} tree. To compute α_i we use the residuals computed, r_i and compute the following: $arg \min_{\alpha} = \sum_{i=1}^{m} L(Y_i, F_{i-1}(X_i) + \alpha h_i(X_i, r_{i-1}))$ where $L(Y, F(X))$ is a differentiable loss function.

Figure 11: Schematic of XGBOOST.

XGBoost is suited for nowcasting for several reasons. In fact, XGBoost is known for its exceptional predictive performance, especially in complex and high-dimensional datasets like macroeconomic time series. It is also able to handle non-linearity and missing data. XGBoost is also flexible and scalable (can be parallelized).

4.2.2.3 Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)

A Recurrent Neural Network (RNN) is a type of artificial neural network designed to handle sequential data and time series. Unlike traditional feedforward neural networks, RNNs have a feedback loop that allows them to persist information across time steps (Dematos et al., 1996). This feedback mechanism enables RNNs to capture temporal dependencies and patterns in sequential data, making them well-suited for tasks such as natural language processing, speech recognition, and time series forecasting. The basic building block of an RNN is the "recurrent" or "hidden" state, which maintains a memory of the past inputs and influences the current output. At each time step, the RNN takes an input, processes it along with the previous hidden state, and produces an output and a new hidden state. This process is repeated for each time step, allowing the RNN to consider the entire history of the sequence when making predictions. One of the main advantages of RNNs is their ability to handle input sequences of variable lengths. They can adapt to sequences of different lengths by processing each time step individually, making them flexible for various real-world applications. However, traditional RNNs have limitations in capturing long-term dependencies due to the vanishing or exploding gradient problem, limiting their usefulness in the nowcasting application (Ekman, 2021).

In this research, the focus is on exploring more advanced variants of recurrent neural networks (RNNs) that incorporate a gating mechanism called the Long Short-Term Memory (LSTM) network which was published by Hochreiter and Schmidhuber in 1997. Unlike traditional RNNs, LSTM is designed to address the challenge of capturing long-term dependencies in sequential data, making it particularly suitable for time series forecasting tasks, including GDP nowcasting. This study is particularly focused on investigating the performance and suitability of LSTM for nowcasting GDP growth while comparing it to other modeling techniques.

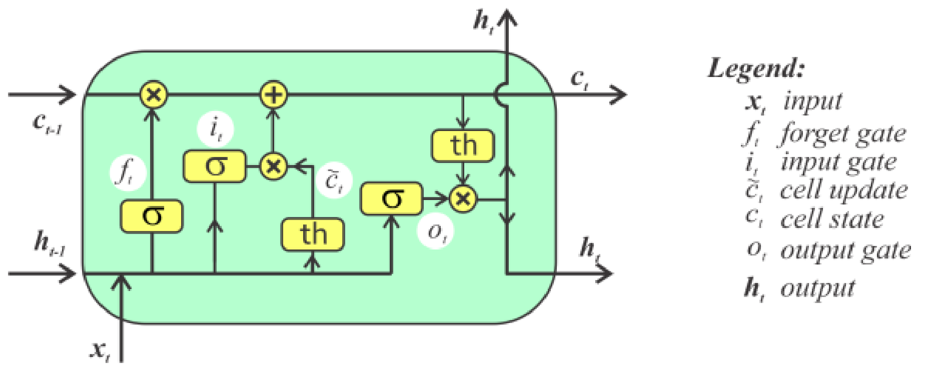


Figure 12: Schematic of LSTM.

In the LSTM network, each unit takes three inputs: X_t, which is the input at the current time step; h_{t-1}, which is the output from the previous LSTM unit; and c_{t-1}, which represents the memory from the previous unit, considered to be a crucial input. The unit then generates two outputs: h_t, which is the output of the current unit, and c_t, which represents the memory of the current unit. The decision-making process of this single LSTM unit involves considering the current input, the output from the previous unit, and the memory from the previous unit. Using this information, the unit generates a new output and modifies its memory.

The first straight line in the diagram is the memory layer. In an LSTM unit, the input includes the old memory represented as a vector. The first operation it undergoes is the forget gate $\otimes$, which is an element-wise multiplication with another vector. The forget gate allows the model to decide what information from the old memory, c_{t-1}, needs to be retained and what should be forgotten. The memory flow will go through the $\oplus$ operator which means piece-wise summation. New memory and the old memory will merge by this operation. In particular, how much new memory should be merged is controlled by the second $\otimes$ operator. If the forget gate vector contains values that are close to 0, it indicates that the model wants to forget most of the old memory, effectively discarding previous information. On the other hand, if the forget gate vector contains values close to 1, it means that the model wants to retain the old memory without forgetting anything. By adjusting the forget gate, the LSTM unit can selectively retain or discard information from the old memory based on the current input and the task at hand. After these operations are completed, the old memory c_{t-1} is changed to c_t.

Onto the second layer, the first element in the LSTM unit is referred to as the forget valve. It is controlled by a simple one-layer neural network. This neural network takes as inputs the previous output, h_{t-1}, which is the output of the previous LSTM block, the current input, X_t, which is the input for the current LSTM block, the previous memory, c_{t-1}, which is the memory of the previous block, and a bias vector, $_0$. The neural network employs a sigmoid activation function, and its output is a vector representing the forget valve. This forget valve vector is applied to the old memory, c_{t-1}, using element-wise multiplication. The second valve is called memory valve which controls how much the new memory should influence the old memory. The new memory itself, however, is generated by another neural network. It is also a one-layer network but uses hyperbolic tangent (tanh) as the activation function. The output of this network will element-wise multiple the new memory valve and add to the old memory to form the new memory.

The last step is to generate the output of the network. An output valve is controlled by the new memory. This valve controls how much new memory should output to the next LSTM unit.

4.2.3 Time Series Specific Models

In this subsection, the focus is on introducing and discussing some of the most relevant time series models commonly employed by AutoML packages. These models are specifically designed to handle the unique characteristics and challenges of time series data. AutoML frameworks leverage these models to automate the process of building and selecting the best-performing model for time series forecasting tasks.

4.2.3.1 Models with Conditional Deseasonalization and Detrending

The first model is the Linear Model with Conditional Deseasonalization and Detrending. The first model is the Linear Model with Conditional Deseasonalization and Detrending. This method aims to address the seasonal and trend components present in time series data, making it more suitable for capturing the underlying patterns. The process involves first identifying the seasonality and trend in the time series using techniques such as seasonal decomposition or moving averages. Once the seasonal and trend components are isolated, they can be removed from the original series, leaving behind the deseasonalized and detrended data. With the deseasonalized and detrended series, a linear model is then applied to make predictions based on the remaining time independent patterns. This approach can be effective for time series data with clear seasonal and trend patterns, as it attempts to capture the stationary aspects of the data for forecasting purposes. However, it may not be as effective for time series with complex non-linear relationships or irregular patterns.

The second model is the Bayesian Ridge with Conditional Deseasonalize and Detrending. This method aims to combine Bayesian regularization techniques with the removal of seasonal and trend components from the time series data. To begin, the seasonal and trend components are identified and removed from the original time series using appropriate methods like seasonal decomposition or detrending techniques. This step helps in isolating the stationary part of the data. Next, the Bayesian Ridge model is applied to the deseasonalized and detrended series. The Bayesian Ridge regression is a probabilistic linear regression model that uses Bayesian inference to estimate the model's parameters. It is particularly useful for situations with limited data, as it incorporates a regularization term to prevent overfitting. By combining the Bayesian Ridge model with conditional deseasonalization and detrending, this approach aims to capture the time series' stationary patterns while mitigating the risk of overfitting. It can be particularly effective for time series data with complex relationships and noise, making it suitable for forecasting tasks where uncertainty and regularization are important considerations. However, as with any modeling approach, its effectiveness should be validated and compared to other techniques based on the specific characteristics of the data at hand.

4.2.3.2 Theta Forecaster

The Theta forecaster is another relevant time series model commonly utilized by AutoML packages. This forecasting method is specifically designed to handle time series data that exhibit strong seasonality. The Theta model of Assimakopoulos & Nikolopoulos (2000) is a simple method for forecasting the involves fitting two θ-lines, forecasting the lines using a Simple Exponential Smoother, and then combining the forecasts from the two lines to produce the final forecast. In other word, it is based on the idea of using a seasonal naive forecast combined with a damping parameter (theta) to adjust the seasonality as time progresses.

The model is implemented in steps:

1. Test for seasonality;
2. Deseasonalize if seasonality detected;
3. Estimate α by fitting a SES model to the data and b_0 by OLS:

$$X_t = a_0 + b_0(t-1) + \epsilon_t$$

$$\tilde{X}_t = (1-\alpha)X_t + \alpha\tilde{X}_{t-1}$$

4. Forecast the series:

$$\hat{X}_{T+h|T} = \frac{\theta - 1}{\theta}\hat{b}_0\left[h - 1 + \frac{1}{\hat{\alpha}} - \frac{(1-\hat{\alpha})^T}{\hat{\alpha}}\right] + \tilde{X}_{T+h|T}$$

5. Reseasonalize if the data was deseasonalized.

The Theta forecaster is particularly useful when dealing with time series data that exhibit consistent and repetitive seasonal patterns. It can provide reliable short-term forecasts by taking advantage of the underlying seasonality without requiring complex model training. However, its performance might be limited when dealing with highly irregular or non-seasonal data, as it heavily relies on the observed seasonal patterns.

4.2.3.3 ETS (Error, Trend, Seasonality) and Exponential Smoothing

The ETS (Error, Trend, Seasonality) model is another important time series forecasting method commonly employed by AutoML packages. It is designed to capture the underlying patterns in time series data, including trend and seasonality components, while also considering the random error or noise present in the data.

The ETS model is based on three main components. The Error (E) component accounts for the random fluctuations or noise present in the time series data. It represents the variability that cannot be explained by the trend or seasonality components. The Trend (T) component captures the long-term movement or pattern observed in the data. It represents the overall direction in which the time series is moving over time, such as upward, downward, or stable. The Seasonality (S) component accounts for the repetitive patterns observed in the data at fixed intervals, such as daily, weekly, or monthly cycles.

It represents the regular fluctuations in the data due to seasonal effects. The ETS model combines these three components to make predictions for future time points. The model is capable of adapting to changes in the trend and seasonality over time, making it suitable for time series data with varying patterns.

Exponential smoothing is an ETS method that is based on the principle of giving more weight to recent observations and gradually decreasing the weights as the observations become older. It is a simple yet effective method for handling time series data with trend and seasonality. There are different variants of exponential smoothing, such as Simple Exponential Smoothing (SES), Holt's Linear Exponential Smoothing (Holt's), and Holt-Winters Exponential Smoothing (Holt-Winters). Each variant incorporates different levels of complexity and handles various characteristics of the time series data.

4.2.3.4 Recursive Tabular

The Recursive Tabular Model is specifically designed to handle time series data, where past observations are used to predict future values. In the Recursive Tabular Model, the dataset is organized in a tabular format, where each row represents a specific time point, and each column corresponds to different features or variables. The target variable to be predicted is typically one of the columns.

The model works in a recursive manner, where it iteratively forecasts future values based on past observations. The process involves the following steps:

1. Train the Model: Initially, the model is trained on the historical time series data. It uses a portion of the data to learn the patterns, relationships, and dependencies between the features and the target variable;
2. Forecast Future Values: After training, the model makes predictions for the next time point in the time series. It utilizes the trained knowledge to forecast the value at the next time step;
3. Update Data: Once the forecasted value is obtained, it is added to the dataset as a new observation, extending the time series. This new observation is then used in the next iteration to predict the value for the subsequent time point;
4. Repeat: The process of forecasting, updating the data, and making subsequent predictions is repeated for the desired number of future time points.

By continuously updating the dataset and using the most recent forecasts as inputs for future predictions, the Recursive Tabular Model dynamically adapts to changes in the time series and considers any observed shifts or patterns in the data.

4.3 Nowcasting Experiment

This section examines the inherent challenges of nowcasting. It investigates the complexities of handling real-time data, managing intricate features, and addressing the time-sensitive nature of economic indicators.

Transitioning to the results of the study, the rationale behind selecting specific datasets is detailed, and the metrics used to evaluate model performance are outlined.

4.3.1 Problems of Nowcasting

As said above, GDP is typically calculated on an annual basis, but in most market economies it is also measured on a quarterly frequency. The first measurements are usually available only a few months after the end of the current quarter, which makes it difficult to assess economic activity in real-time.

Such analysis faces the following three problems (Giannone et al., 2008):

1. Combining monthly predictor variables and a quarterly target variable;
2. Handling a large number of potential regressors;
3. Monthly data are released at different times (unsynchronized) within a quarter.

4.3.1.1 Mixed Frequency Problem

The first problem arises when there is mixed frequency data, or when all independent variables and the dependent variable are not recorded with the same periodicity. Our full dataset ranges from 1960:Q1 to 2022:Q3, so that it contains feature observations for every month and target observations for every third month (i.e., for every quarter).

This gives rise to the question on how to combine the diverse granularity of the data. Different methods were proposed in recent years, some of them can be found in relatively recent literature such as Zheng and Rossiter (2006). The most common which is used in this study is averaging or simple bridge equation. Taking an indicator as example, it is relatively straightforward, and consists in taking the average of the three months for the quarter of interest of such indicator. Of course, there are much more complex procedures that can be more suitable in this context, however averaging gave consistent results across previous studies with the advantage of simple implementation.

4.3.1.2 Edges Problem

The second is the heterogeneous publication schedules of independent variables, frequently referred to as "ragged edges". In other words, there are differences in missing variables at the end of series due to different publication schedules for each. Any nowcasting methodology must provide provisions for incomplete or partially complete data, as varying availability of latest data is the reality of most data sets of economic series. In this case, GDP is a collection of several variables from different sources. These sources are often public institutions which collects data independently. As a result, due to the time and complexity required for data collection and statistical data processing as well as the intrinsic various availability of the underlying data, the release of GDP data is often delayed within the month of interest. Different strategies can be implemented to resolve this issue such as averaging, ARMA model imputation and EM imputation.

As this paper is focused on testing whether or not AutoML would be applicable and leading to significant results in an economic nowcasting exercise, introducing a further component would add noise to the original purpose of the study. So, we followed a different path. Regarding the target variable, we assume that quarterly measured GDP observations are available for months M3, M6, M9, and M12. Originally, the FRED data releases associate these quarterly measurements with timestamps M1, M4, M7, and M10. However, using these timestamps would be highly counter-intuitive in this type of analysis. Following the literature, we assume that quarterly GDP measurements are published at the beginning of the following quarter. In reality, the first measurements are usually available only a few months after the end of the current quarter. Regarding the predictors, in this study we assumed that all the variables are released promptly at the same time within a quarter at the beginning of each month. In this way, the ragged edges problem is nullified.

4.3.1.3 Curse Of Dimensionality Problem

Finally, there is the issue of the “curse of dimensionality”, which renders many classical econometric methods less effective in the nowcasting context and hinders the application of “big data” to the field (Buono et al. 2017). The problem stems from the nature of many economic variables, where they may have few observations relative to the potential pool of explanatory variables or features. In fact, to predict the GDP growth, more than 130 variables are used with a relatively short number of records spanning from 1960 to 2009 (200 quarterly observation). This problem is solved using models that automatically deals with this problem and applying 3-fold cross validation without shuffling in order to mitigate overfitting.

4.3.2 Training Set, Evaluation Periods And Vintages

Instead of the simple approach splitting the dataset into training, validation and test subset, two separate test periods were used for assessing the performance of the models, each representing a significant period in the US economic history. This approach resembles the approach used by Hopp, 2021.

In particular, all models are trained on a fixed-size, static training window, which ranges from 1960:Q1 to 2019:Q4. Thus, it contains 200 observations for quarterly GDP growth and 600 observations for the monthly indicators (features) which are then compressed into quarterly frequency.

The first evaluation period ranges from 2010:Q1 to 2019:Q4, so it ends before the economic consequences of the COVID-19 pandemic would hit the economy, but it captures the impact of the great recession of 2008 (it includes 40 observations for the target variable and 120 measurements for the monthly indicators). Our second evaluation period ranges from 2010:Q1 to 2022:Q3, so it also includes the COVID-19 crisis, although it captures its impact as it is abrupt and sudden, the real long-term consequences have yet to be seen. We evaluate the results from two test (evaluation) periods in the empirical analysis (contains 51 target and 153 feature observations). This helped to ensure the robustness of the results, testing the models across very disparate eras with large economic shocks as this allow to test if the model is able to learn the truly underlying patterns, and can adapt to very different input values. The training periods for

In addition to the two separate evaluation periods, we simulated the nowcasting exercise in a real-life scenario (which would be to generate predictions as data becomes available) creating three vintages per quarter, thus depending on the information set available for our models. Hence, three forecasts per GDP growth data point are generated. The 1-month nowcasting scenario presumes that the monthly indicators' value is available up through the first month within a quarter. In the 2-month scenario, the information set also contains feature observations released until the second month within the current quarter. Finally, in the 3-month scenario, we use all monthly indicator data available at the end of the current quarter. Here, the same thought process for the mixed-frequency problem can be applied, thus averaging was used.

4.3.2.1 Common Settings For AutoML Packages

When testing out different packages, it is vital in order to ensure robustness and consistency to the analysis to use the same settings and parameters whenever can be applicable. In general, in the packages where a time series specific module was implemented was used, otherwise the regression module was chosen. Of course, the main implication of this choice would be that the algorithm search space differs being "broader" in the first case as also time series models are considered. Nevertheless, we expect the different packages to converge in a similar result preferring machine learning models which were proven to be more effective than statistical models in previous studies.

The following settings were commonly applied:

1. Budget time is 1200 seconds;
2. Cross Validation is 3 folds with no shuffling;
3. Whenever applicable, Forecast Horizon is 3;
4. No restrictions on hyperparameters;
5. No restrictions on the search space.

As for the metric with which models from different packages are compared, Root Mean Squared Error was chosen as it is commonly employed for such purposes in most studies across different field and was supported by all packages used. It has to be noted that as this metric needs to be minimized, some packages use the negative version, but taking the absolute results in the same value.

Chapter 5 – Results

In the upcoming chapter, the focus shifts to the results and outcomes of the exploration into various machine learning and automated techniques applied to the task of nowcasting U.S. GDP growth. This section delves into extensive experiments and analyses, revealing the performance, accuracy, and practical utility of these methods in real-world forecasting scenarios.

5.1 Best Models For Each Framework

This section presents a comprehensive examination of the outcomes obtained from the best-performing models from each framework. The results shed light on the strengths and limitations of each model, offering valuable insights for effective forecasting strategies.

As mentioned earlier, the goal is to mimic the nowcasting process by using a rolling window approach based on when information becomes available. Following the training phase on the training dataset, the top-performing model within each package is selected to generate predictions for the corresponding time window. Subsequently, the RMSE is computed for each nowcasting window. The table below displays the results for the first evaluation period, from 2010 to 2019.

With the exception of the PyCaret framework, the other packages showed relatively similar performance. On average, they achieved an RMSE of approximately 0.2181. Notably, two packages, AutoKeras and TPOT, performed better overall. Among them, AutoKeras stood out as the top performer, surpassing the average RMSE score with an average value of 0.0631. It's interesting to observe that in the simulated nowcasting scenario, where more information becomes available from month to month, one might expect that the models' accuracy would progressively improve.

However, the reality is more nuanced. While it is theoretically plausible that the Root Mean Squared Error (RMSE) should decrease over time as more data becomes available, this isn't always the case. In fact, in most of the models, the RMSE is barely decreasing and in for the LSTM model is particularly noteworthy as it increases.

Table 2: Results of selected best models from each package for 2010-2019.

		1-month	*2-month*	*3-month*
Package	Best Model	RMSE	RMSE	RMSE
PyCaret	Linear w/ Cond. Deseas. & Detr.	5.6398	5.6278	4.3971
AutoGluon	Recursive Tabular	0.2822	0.2822	0.2822
AutoKeras	LSTM	**0.0557**	**0.0661**	**0.0676**
FLAML	Extra Trees	0.2491	0.222	0.2104
H20	GBM	0.2432	0.2191	0.2170
TPOT	Extra Trees	0.1537	0.1555	0.1463
EvalML	Extra Trees	0.2798	0.2801	0.2743
Mljar	Random Forest	0.2633	0.2636	0.2613

Even for the second evaluation period from 2010 to 2022, it can be seen that the results are consistent with the previous findings. Notably, among the frameworks evaluated, AutoKeras identified LSTM as the best-performing model. Additionally, it's worth highlighting that PyCaret demonstrated a substantial improvement when using the 3-month average as compared to utilizing only the 1-month information. This suggests that AutoKeras, leveraging the LSTM model, displayed superior predictive capabilities in the nowcasting experiment (as the results are really close this could be extended to TPOT).

Table 3: Results of selected best models from each package for 2010-2022.

		1-month	*2-month*	*3-month*
Package	Best Model	RMSE	RMSE	RMSE
PyCaret	Linear w/ Cond. Deseas. & Detr.	9.8018	4.3971	0.8676
AutoGluon	AutoETS	0.1981	0.1981	0.1981
AutoKeras	LSTM	**0.0210**	**0.0170**	**0.0216**
FLAML	Extra Trees	0.1285	0.1285	0.1285
H20	GBM	0.1598	0.1598	0.1598
TPOT	Extra Trees	0.0315	0.0387	0.0573
EvalML	Extra Trees	0.1558	0.1580	0.1484
Mljar	Random Forest	0.1408	0.1398	0.1267

5.2 Diebold and Mariano Test

The Diebold-Mariano (DM) test was intended for comparing forecasts, however in the empirical literature is extensively used to assess whether one forecasting model performs significantly better than another, especially in the context of time series forecasting. The DM test and the RMSE serve different purposes in the evaluation of forecasting models. In fact, the DM test is a statistical hypothesis test that helps determine whether there is a meaningful difference in forecasting accuracy between two models while the RMSE is a performance metric. It assesses whether the difference in performance is statistically significant. This is crucial because models might have slightly different performance due to randomness in data, and you want to be confident that the difference is not due to chance. The test helps in making informed decisions about which model to use for forecasting during model selection. Specifically, the test evaluates the null hypothesis $H0$, which states that the mean of the loss differential of Model A is lower than or equal to that of Model B. In simpler terms:

- Null Hypothesis ($H0$): The forecasting accuracy of Model A is as good as or better than that of Model B.
- Alternative Hypothesis: Rejecting the null hypothesis ($H0$) suggests that the forecasts of Model B are significantly more accurate than those of Model A.

In essence, the test determines whether there is a statistically significant difference in forecasting performance between the two models. If the null hypothesis is rejected, it indicates that Model B is the superior model in terms of accuracy. If the null hypothesis is not rejected, there is no significant evidence to claim that Model B outperforms Model A.

Table 4: DM Test of AutoKeras against other frameworks for 2010-2019 predictions.

		PyCaret	*AutoGluon*	*FLAML*	*H2O*	*TPOT*	*EvalML*	*Mljar*
1-month	*DM Statistics*	-8.5595	1.3444	1.4262	2.0319	-2.5186	1.4236	2.0474
	P-Value	**8.62e-11**	0.9067	0.9191	0.9755	**0.0080**	0.9187	0.9763
2-month	*DM Statistics*	-8.8795	1.3444	1.4150	2.5864	-2.5730	1.3359	2.0118
	P-Value	**3.31e-11**	0.9067	0.9175	0.9932	**0.0070**	0.9053	0.9744
3-month	*DM Statistics*	-9.4573	1.3444	1.6294	2.4856	-2.1444	1.5696	2.1982
	P-Value	**6.06e-12**	0.9067	0.9444	0.9913	**0.0191**	0.9377	0.9830

For the first evaluation period from 2010 to 2019, when AutoKeras is compared with the other models, the null hypothesis is not rejected meaning that there is no significant proof that the LSTM model performs statistically better. However, other two scenarios can be distinguished.

The first scenario is where the DM test yielded a significant p-value, but the RMSE of the alternative model is very high. This is the case of PyCaret. In this situation, the significant p-value suggests that there is evidence to support the claim that the alternative model performs differently from the benchmark model. However, the high RMSE of the alternative model indicates that it is not providing accurate predictions. This scenario can be challenging because it implies that while there is a statistical difference in performance, the alternative model is not a viable choice for making accurate predictions. It's crucial to consider the practical implications and consequences of choosing a model with high RMSE, even if it shows a statistically significant difference in performance.

The second scenario is where the DM test yielded a significant p-value, and the RMSE of the alternative model is very low. This is the case of TPOT. In this situation, it suggests that although there is a statistically significant difference between the models, both models are performing exceptionally well in terms of prediction accuracy. This can be seen as a positive outcome because it implies that you have multiple reliable models to choose from, each of which is providing accurate forecasts.

Table 5: DM Test of AutoKeras against other frameworks for 2010-2022 predictions.

		PyCaret	*AutoGluon*	*FLAML*	*H2O*	*TPOT*	*EvalML*	*Mljar*
1-month	*DM Statistics*	-1.2383	1.4234	1.8794	1.6792	1.8239	2.1036	2.1463
	P-Value	**3.76e-17**	0.9196	0.9670	0.9503	0.9629	0.9798	0.9816
2-month	*DM Statistics*	7.2487	1.3785	1.8270	1.7036	1.6012	2.0076	2.0866
	P-Value	**1.22e-09**	0.9129	0.9632	0.9527	0.9422	0.9750	0.9790
3-month	*DM Statistics*	-6.3030	1.3971	1.8491	1.8448	0.9478	2.0995	2.1251
	P-Value	**3.69e-08**	0.9157	0.9648	0.9645	0.8261	0.9796	0.9807

In the second evaluation period from 2010 to 2022, the overall results remain consistent with the first evaluation period, except for one notable difference. In this period, the hypothesis where TPOT is considered the alternative model is rejected. However, it's essential to consider the context and practical implications of this result. While the test suggests no significant difference in performance between TPOT and Model A during this period, it doesn't necessarily mean that TPOT is a poor model. It's possible that both models perform well, but any differences in accuracy are not statistically significant based on the available data. The detailed plots pertaining the multivariate DM test can be found in the Appendix.

5.3 Multiple-Line Plots

A time series multiline plot, also known as a multiline time series chart or simply a multiline plot, is a type of data visualization used to display multiple time-dependent data series on a single graph. It is a way to visualize how multiple variables or measures change over time and how they may relate to each other. The solid blue line in the plot represents the observed real GDP growth data. In contrast, the dotted lines on the plot represent the forecasted or predicted values. These lines represent the values of GDP growth that were estimated or predicted by a forecasting model. The dotted lines show how the model expects GDP growth to change over time based on its analysis of historical data and other relevant factors.

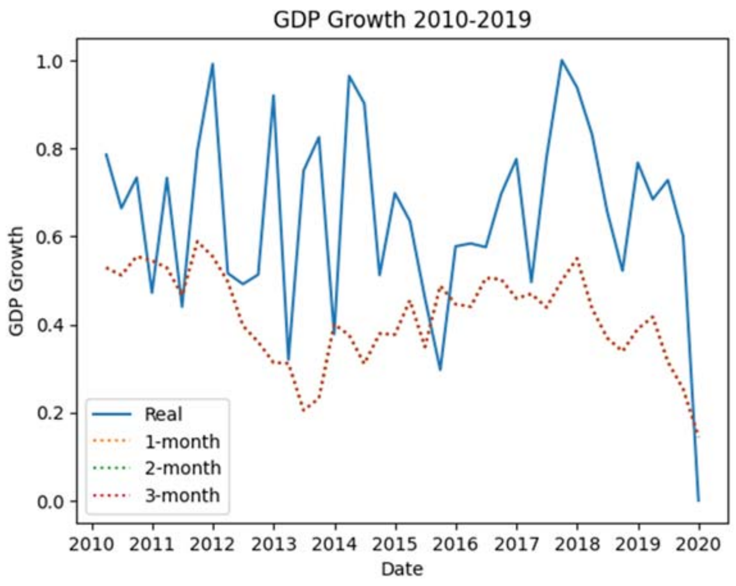


Figure 13: Multi-line Plot for 2010-2019.

This visual comparison helps evaluate the accuracy and reliability of the forecasting model in capturing the trends and fluctuations in GDP growth. In particular, it can be seen that for both the evaluation periods, the AutoKeras LSTM is pessimistic in most cases. It follows rather accurately the trends. However, the growth is in almost all instances less than the actual (both positive and negative growth).

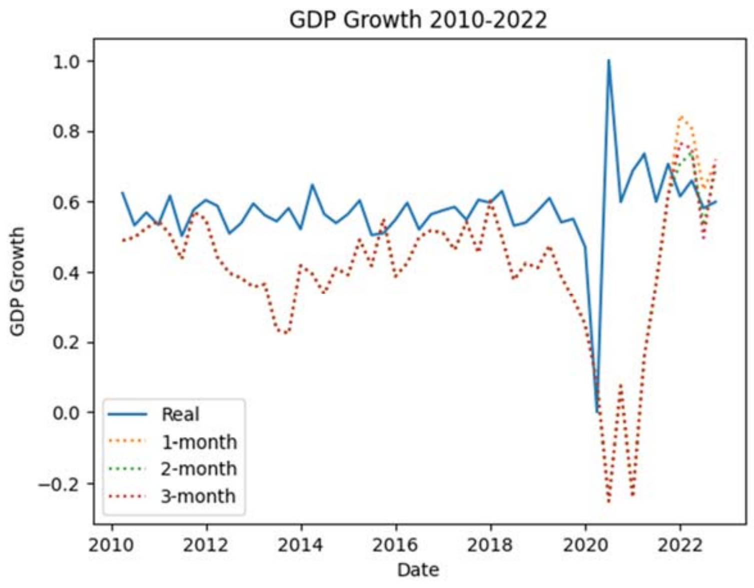


Figure 14: Multi-line Plot for 2010-2022.

Chapter 6 – Conclusions

In this chapter, the final insights and implications drawn from the study are presented. The key findings are summarized, their significance in the context of the research objectives is discussed, and valuable recommendations for future work in this field are provided. This chapter concludes the research, providing a comprehensive understanding of the outcomes and their broader implications.

6.1 Conclusions

This study has undertaken an exploration of automated machine learning (AutoML) frameworks and their application to the challenging task of nowcasting U.S. GDP growth. The investigation involved a comprehensive evaluation of several leading AutoML packages, including AutoKeras, TPOT, PyCaret, and others. In this sense, this paper serves the purpose of proof of concept as it represents an attempt to verify the feasibility and the utility of this application.

Through experimentation and analysis, several key findings emerged. Firstly, AutoML has showcased its prowess in democratizing machine learning, making it accessible to both experts and non-experts alike. Indeed, with just a few lines of code, AutoML can produce noteworthy results that stand up to comparison with outcomes from substantially more extended studies that relied heavily on domain expertise. It significantly streamlines the model selection, hyperparameter tuning, reducing the need for extensive manual intervention. Furthermore, the study revealed noteworthy variations in the performance of different AutoML frameworks. AutoKeras and TPOT emerged as standout performers, consistently delivering superior predictive accuracy compared to the average RMSE. The statistical analysis indicates that there is no significant difference among the results produced by different AutoML packages, suggesting a high level of consistency in their agnostic processes. This finding implies that while it's advisable to utilize a variety of tools as a best practice, the results obtained from one package can often serve as a reasonably accurate representation of the others. However, it's worth noting that these packages are constructed differently, and while the results are generally close, there may still be variations.

In light of these results, it's essential to emphasize the importance of not only adopting AutoML but also carefully selecting the specific framework that best aligns with the unique requirements of the forecasting task. The performance differences observed against PyCaret underscore the significance of this selection process, as the choice of the AutoML framework can have a substantial impact on the forecasting outcomes. The deployment of Diebold-Mariano tests allowed for robust statistical evaluations of the model performance differences. These tests, coupled with RMSE assessments, enabled drawing statistically significant conclusions regarding the effectiveness of various AutoML packages. One intriguing observation was the sensitivity of model performance to the availability of additional information. As more data was progressively introduced, the improvement in predictive accuracy did not always follow a linear trajectory. This finding underscores the importance of understanding the nuances of the forecasting scenario and tailoring the modeling approach accordingly.

Regarding the model selection, LSTM proved to be the best machine learning algorithm for this application. Nevertheless, LSTM did not exhibit significant relevance in GDP growth nowcasting when statistically compared to other models like Extra Trees. This observation aligns with the findings in the study conducted by Nemeth and Hadházi. It is worth noticing that the LSTM generated by AutoKeras achieved a much smaller RMSE ,and thus, much more accurate predictions.

In conclusion, this study underscores the pivotal role of AutoML in advancing predictive modeling for complex economic tasks like GDP nowcasting. It showcases the potential of AutoKeras and TPOT as leading contenders in the field, offering valuable insights for practitioners and researchers alike. As the field of AutoML continues to evolve, it promises to play an increasingly vital role in augmenting predictive analytics across diverse domains.

6.2 Limitations and Future Developments

The quality and accessibility of economic data directly shape GDP growth nowcasting. Any issues related to data discrepancies, inaccuracies, or delays can significantly impact the precision of nowcasting models. In fact, The findings of this study are inevitably influenced by the quality and accessibility of economic data, which is subject to certain limitations that warrant consideration. Firstly, data discrepancies, arising from various sources and collection methods, pose a notable challenge. In addition, data release delays can significantly affect the timeliness of nowcasting. Future research should investigate methods to mitigate these data limitations. This includes thorough data cleaning, validation procedures, and the integration of alternative data sources. Addressing these data limitations requires the implementation of rigorous data preprocessing techniques. Data cleaning and validation procedures are essential to identify and rectify discrepancies or errors in the data. Moreover, the incorporation of alternative data sources, such as high-frequency indicators or sentiment data, can help mitigate the impact of data limitations and enhance the robustness of nowcasting models. While this specific study did not delve into the data preprocessing aspect, AutoML packages are equipped to handle various aspects of data preprocessing (in particular, feature engineering), which can significantly streamline the model development process.

In this study, a variety of AutoML techniques were explored to develop predictive models for GDP nowcasting. While these techniques cover a substantial portion of the machine learning landscape, it's crucial to acknowledge that they represent just a subset of the potential model complexities that can be investigated. Furthermore, this study represents a proof of concept, therefore the AutoML packages were not trained to their full potential having restricted some parameters such as the quality of the considered models, the time and other package-specific parameters. Ideally, the packages should be trained for long periods of time and on powerful machines. In addition, the primary evaluation metrics used in this study were RMSE and Diebold-Mariano tests. While these are informative, future research should consider a broader array of evaluation metrics to capture different facets of nowcasting performance, including measures of uncertainty and prediction intervals.

AutoML models, especially deep learning models like LSTMs, are often viewed as black boxes, making it difficult to understand their decision-making processes. Subsequent research should prioritize methods that enhance model interpretability without compromising predictive power. This is crucial for policymakers who need insights into what drives model predictions. Perhaps, the biggest limitation of AutoML is indeed the lack of transparency of these packages. This could lead to several problems along the line.

For example, without transparency, it can be challenging to identify and mitigate biases in AutoML models. Biases may exist in the training data, and if not addressed, they can lead to unfair or discriminatory outcomes. Understanding model decisions is essential for addressing bias issues effectively. Additionally, the closure of these packages could make the implementation of an experiment challenging as often each framework has a pre-defined pipeline to follow and do not allow for customization.

Lastly, investigating feature importance using libraries like Shap involves a deeper examination of how individual features, such as variables or attributes in a dataset, contribute to the predictions made by machine learning models. Shap (SHapley Additive exPlanations) is a popular tool for this purpose.

References

Michael W. McCracken, a. S. (n.d.). *FRED-MD: A Monthly Database for Macroeconomic Research.* Retrieved from Federal Reserve Bank of St. Louis: https://doi.org/10.20955/wp.2015.012

Adya, M. &. (1998). How e ective are neural networks at forecasting and prediction? a review and evaluation. *Journal of Forecasting.*

Babii, A. G. (2021). Machine learning time series regressions with an application to nowcasting. *Journal of Business Economic Statistics.*

Binner, J. B. (n.d.). *A comparison of linear forecasting models and neural networks: an application to euro in ation and euro divisia.*

Bok, B. C. (2018). Macroeconomic nowcasting and forecasting with big data. *Annual Review of Economics.*

Bouwman, K. E. (2011). Forecasting with real-time macroeconomic data: The ragged-edge problem and revisions. *Journal of Macroeconomics.*

Breiman, L. (2001). *Machine Learning.*

Breiman, L. F. (2017). *Classification and regression trees.*

Chen, T. &. (2016). *Xgboost: A scalable tree boosting system.*

Diebold, F. X. (1995). Comparing predictive accuracy. *Journal of Business Economic Statistics,.*

Doz, C. G. (2011). A two-step estimator for large approximate dynamic factor models based on kalman ltering. *Journal of Econometrics.*

York, F. R. (2021). *Nowcasting report: Methodology.*

Giannone, D. R. (2008). Nowcasting: The real-time informational content of macroeconomic data. *Journal of Monetary Economics.*

Hochreiter, S. &. (1997). Long short-term memory. *Neural computation.*

Hopp, D. (2021). *Economic nowcasting with long short-term memory arti cial neural networks (lstm).*

Jansen, W. J. (2016). Forecasting and nowcasting real gdp: Comparing statistical models and subjective forecasts. *International Journal of Forecasting.*

Qureshi, S. C. (2020). *Forecasting canadian gdp growth using xgboost.*

Richardson, A. v. (2021). Nowcasting GDP using machine-learning algorithms: A real-time assessment. *International Journal of Forecasting*.

Schumacher, C. &. (2008). Real-time forecasting of german gdp based on a large factor model with monthly and quarterly data. *International Journal of Forecasting*.

Stevanovic, D. S. (2019). *How is Machine Learning Useful for Macroeconomic Forecasting?*

Chung, J. G. (2014). *Empirical evaluation of gated recurrent neural networks on sequence modeling.*

Harvey, A. C. (1990). *Forecasting, structural time series models and the Kalman filter.*

Loermann, J. &. (n.d.). *Nowcasting us gdp with artificial neural networks.*

Forecasting, s. t. (2010). *Factor MIDAS for nowcasting and forecasting with ragged-edge data: A model comparison for German GDP.*

Tkacz, G. (2001). Neural network forecasting of Canadian GDP growth. *International Journal of Forecasting*.

Torres, D. G. (2018). *Applying recurrent neural networks for multivariate time series forecasting of volatile financial data.*

Wallis, K. F. (1986). Forecasting with an econometric model: The 'ragged edge'problem. *Journal of Forecasting*.

Banbura, M. G. (n.d.). *Handbook of Economic Forecasting.*

Antolin-Diaz, J. T. (2020). *Advances in Nowcasting Economic Activity: Secular Trends, Large Shocks and New Data.*

Camacho, M. Y. (2015). "Can We Use Seasonally Adjusted Variables in Dynamic Factor Models. *Studies in Nonlinear Dynamics & Econometrics*.

Jean-François Dauphin, K. D. (2022). *A Scalable Approach Using DFM, Machine Learning and Novel Data, Applied to European Economies.*

Ahmad Alsharef, K. A. (2022). *Review of ML and AutoML Solutions to Forecast Time-Series Data.*

Jin H, S. Q. (2019). *Auto-keras: an efficient neural architecture search system.*

Alteryx. (2021). *EvalML.* Retrieved from https:// evalml. alter yx. com/ en/ stable/

al, E. N. (2020). *Autogluon-tabular: Robust and accurate automl for structured data.*

LeDell E, P. S. (2020). *H2O automl: scalable automatic machine learning.*

SMJ, D. (2020). *an autoML approach to time series forecasting.*

Xu Z, T. W.-W. (2021). *AutoML meets time series regression design and analysis of the autoseries challenge.In: Joint European conference on machine learning and knowledge discovery in databases.*

Hu Y-J, H. S.-W. (2017). *Challenges of automated machine learning on causal impact analytics for policy evaluation.*

Liashchynskyi P, L. P. (2019). *Grid search, random search, genetic algorithm: a big comparison for NAS.*

DE, G. (1989). *Processing Systems 25 (NIPS 2012) 72. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning.*

al, F. M. (2019). *Auto-sklearn: efficient and robust automated machine learning. Automated machine learning.*

Olson, R. S. (2016). *TPOT: A tree-based pipeline optimization tool for automating machine learning.*

Qi W, X. C. (2021). *AutoGluon: a revolutionary framework for landslide hazard analysis.*

Marc-Andr e Z□oller, M. F. (2021). *Benchmark and Survey of Automated Machine Learning Frameworks.*

Balaji, A. &. (2018). *Benchmarking Automatic Machine Learning Frameworks.*

Dinsmore, T. (2016). *Automated Machine Learning: A Short History.*

Ali, M. (2020). *PyCaret: An open source, low-code machine learning library in Python.*

Chi Wang, Q. W. (2021). *FLAML: A Fast and Lightweight AutoML Library .*

Plonski, A. P. (2021). *MLJAR: State-of-the-art Automated Machine Learning Framework for Tabular Data.*

Appendix A

A.1 FRED-MD

Group 1: Output and Income.

	ID	tcode	FRED	Description
1	1	5	RPI	Real Personal Income
2	2	5	W875RX1	Real personal income ex transfer receipts
3	6	5	INDPRO	IP Index
4	7	5	IPFPNSS	IP: Financial Products and Nonindustrial Supplies
5	8	5	IPFINAL	IP: Final Products (Market Group)
6	9	5	IPCONGD	IP: Consumer Goods
7	10	5	IPDCONGD	IP: Durable Consumer Goods
8	11	5	IPNCONGD	IP: Nondurable Consumer Goods
9	12	5	IPBUSEQ	IP: Business Equipment
10	13	5	IPMAT	IP: Materials
11	14	5	IPDMAT	IP: Durable Materials
12	15	5	IPNMAT	IP: Nondurable Materials
13	16	5	IPMANSICS	IP: Manufacturing (SIC)
14	17	5	IPB51222s	IP: Residential Utilities
15	18	5	IPFUELS	IP: Fuels
16	19	1	NAPMPI	ISM Manufacturing: Production Index
17	20	2	CUMFNS	Capacity Utilization: Manufacturing

Group 2: Labor Market.

	ID	tcode	FRED	Description
1	21	2	HWI	Help-Wanted Index for United States
2	22	2	HWIURATIO	Ratio of Help Wanted/No. Unemployed
3	23	5	CLF160OV	Civilian Labor Force
4	24	5	CE160V	Civilian Employment
5	25	2	UNRATE	Civilian Unemployment Rate
6	26	2	UEMPMEAN	Average Duration of Unemployment (Weeks)
7	27	5	UEMPLT5	Civilians Unemployed – Less Than 5 Weeks
8	28	5	UEMP5TO14	Civilians Unemployed for 5-14 Weeks
9	29	5	UEMP15OV	Civilians Unemployed – 15 Weeks and Over
10	30	5	UEMP15T26	Civilians Unemployed for 15 – 26 Weeks
11	31	5	UEMP27OV	Civilians Unemployed for 27 Weeks and Over
12	32	5	CLAIMSx	Initial Claims
13	33	5	PAYEMS	All Employees: Total nonfarm
14	34	5	USGOOD	All Employees: Goods-Producing Industries
15	35	5	CES1021000001	All Employees: Mining and Logging: Industries
16	36	1	USCONS	All Employees: Construction
17	37	2	MANEMP	All Employees: Manufacturing
18	38	5	DMANEMP	All Employees: Durable Goods
19	39	5	NDMANEMP	All Employees: Nondurable Goods
20	40	5	SRVPRD	All Employees: Service-Providing Industries
21	41	5	USTPU	All Employees: Trade, Transportation and Utilities
22	42	5	USWTRADE	All Employees: Wholesale Trade
23	43	5	USTRADE	All Employees: Retail Trade
24	44	5	USFIRE	All Employees: Financial Activities
25	45	5	USGOVT	All Employees: Government
26	46	1	CES0600000007	Avg Weekly Hours: Goods-Producing
27	47	2	AWOTMAN	Avg Weekly Overtime Hours: Manufacturing
28	48	1	AWHMAN	Avg Weekly Hours: Manufacturing
29	49	1	NAPMEI	ISM Manufacturing: Employment Index
30	127	6	CES0600000008	Avg Hourly Earnings: Goods-Producing
31	128	6	CES2000000008	Avg Hourly Earnings: Construction

32	129	6	CES3000000008	Avg Hourly Earnings: Manufacturing

Group 3: Housing.

	ID	tcode	FRED	Description
1	50	4	HOUST	Housing Starts: Total New Privately Owned
2	51	4	HOUSTNE	Housing Starts: Northeast
3	52	4	HOUSTMW	Housing Starts: Midwest
4	53	4	HOUSTS	Housing Starts: South
5	54	4	HOUSTW	Housing Starts: West
6	55	4	PERMIT	New Private Housing Permits (SAAR)
7	56	4	PERMITNE	New Private Housing Permits: Northeast (SAAR)
8	57	4	PERMITMW	New Private Housing Permits: Midwest (SAAR)
9	58	4	PERMITS	New Private Housing Permits: South (SAAR)
10	59	4	PERMITW	New Private Housing Permits: West (SAAR)

Group 4: Consumption, Orders and Inventories.

	ID	tcode	FRED	Description
1	3	5	DPCERA3M086SBEA	Real personal consumption expenditures
2	4	5	CMRMTSPLx	Real Manu. and Trade Industries Sales
3	5	5	RETAILx	Retail and Food Services Sales
4	60	1	NAPM	ISM: PMI Composite Index
5	61	1	NAPMNOI	ISM: New Orders Index
6	62	1	NAPMSDI	ISM: Supplier Deliveries Index
7	63	1	NAPMII	ISM: Inventories Index
8	64	5	ACOGNO	New Orders for Consumer Goods
9	65	5	AMDMNOx	New Orders for Durable Goods
10	66	5	ANDENOx	New Orders for Nondefense Capital Goods
11	67	5	AMDMUOx	Unfilled Orders for Durable Goods
12	68	5	BUSINVx	Total Business Inventories
13	69	2	ISRATIOx	Total Business: Inventories to Sales Ratio

14	130	2	UMSCENTx	Consumer Sentiment Index

Group 5: Money and Credit.

	ID	tcode	FRED	Description
1	70	6	M1SL	M1 Money Stock
2	71	6	M2SL	M2 Money Stock
3	72	5	M2REAL	Real M2 Money Stock
4	73	6	AMBSL	St. Louis Adjusted Monetary Base
5	74	6	TOTRESNS	Total Reserves of Depository Institutions
6	75	7	NONBORRES	Reserves of Depository Institutions
7	76	6	BUSLOANS	Commercial and Industrial Loans
8	77	6	REALLN	Real Estate Loans at All Commercial Banks
9	78	6	NONREVSL	Total Nonrevolving Credit
10	79	2	CONSPI	Nonrevolving consumer credit to Personal Income
11	131	6	MZMSL	MZM Money Stock
12	132	6	DTCOLNVHFNM	Consumer Motor Vehicle Loans Outstanding
13	133	6	DTCTHFNM	Total Consumer Loans and Leases Outstanding
14	134	6	INVEST	Securities in Bank Credit at All Commercial Banks

Group 6: Interest and Exchange Rates.

	ID	tcode	FRED	Description
1	84	2	FEDFUNDS	Effective Federal Funds Rate
2	85	2	CP3Mx	3-Month AA Financial Commercial Paper Rate
3	86	2	TB3MS	3-Month Treasury Bill
4	87	2	TB6MS	6-Month Treasury Bill
5	88	2	GS1	1-Year Treasury Rate
6	89	2	GS5	5-Year Treasury Rate
7	90	2	GS10	10-Year Treasury Rate
8	91	2	AAA	Moody’s Seasoned Aaa Corporate Bond Yield
9	92	2	BAA	Moody’s Seasoned Baa Corporate Bond Yield
10	93	1	COMPAPFFx	3-Month Commercial Paper Minus FEDFUNDS
11	94	1	TB3SMFFM	3-Month Treasury C Minus FEDFUNDS
12	95	1	TB6SMFFM	6-Month Treasury C Minus FEDFUNDS
13	96	1	T1YFFM	1-Year Treasury C Minus FEDFUNDS
14	97	1	T5YFFM	5-Year Treasury C Minus FEDFUNDS
15	98	1	T10YFFM	10-Year Treasury C Minus FEDFUNDS
16	99	1	AAAFFM	Moody’s Aaa Corporate Bond Minus FEDFUNDS
17	100	1	BAAFFM	Moody’s Baa Corporate Bond Minus FEDFUNDS
18	101	5	TWEXMMTH	Trade Weighted U.S. Dollar Index: Major Currencies
19	102	5	EXSZUSx	Switzerland / U.S. Foreign Exchange Rate
20	103	5	EXJPUSx	Japan / U.S. Foreign Exchange Rate
21	104	5	EXUSUKx	U.S. / U.K. Foreign Exchange Rate
22	105	5	EXCAUSx	Canada / U.S. Foreign Exchange Rate

Group 7: Prices.

	ID	tcode	FRED	Description
1	106	6	WPSFD49207	PPI: Finished Goods
2	107	6	WPSFD49502	PPI: Finished Consumer Goods
3	108	6	WPSID61	PPI: Intermediate Materials
4	109	6	WPSID62	PPI: Crude Materials
5	110	6	OILPRICEx	Crude Oil, spliced WTI and Cushing
6	111	6	PPICMM	PPI: Metals and metal products
7	112	1	NAPMPRI	ISM Manufacturing: Prices Index
8	113	6	CPIAUCSL	CPI: All Items
9	114	6	CPIAPPSL	CPI: Apparel
10	115	6	CPITRNSL	CPI: Transportation
11	116	6	CPIMEDSL	CPI: Medical Care
12	117	6	CUSR0000SAC	CPI: Commodities
13	118	6	CUSR0000SAD	CPI: Durables
14	119	6	CUSR0000SAS	CPI: Service
15	120	6	CPIULFSL	CPI: All Items less Food
16	121	6	CUSR0000SA0L2	CPI: All Items less Shelter
17	122	6	CUSR0000SA0L5	CPI: All Items less Medical Care
18	123	6	PCEPI	Personal Cons. Expend.: Chain Index
19	124	6	DDURRG3M086SBEA	Personal Cons. Expend.: Durable Goods
20	125	6	DNDGRG3M086SBEA	Personal Cons. Expend.: Nondurable Goods
21	126	6	DSERRG3M086SBEA	Personal Cons. Expend.: Services

Group 8: Stock Market.

	ID	tcode	FRED	Description
1	80	5	S&P 500	S&P's Common Stock Price Index: Composite
2	81	5	S&P: indust	S&P's Common Stock Price Index: Industrials
3	82	2	S&P div yield	S&P's Composite Common Stock: Dividend Yield

4	83	5	S&P PE ratio	S&P’s Composite Common Stock: Price-Earnings Ratio

A.2 TCODE

The TCODE column denotes the following data transformation for a series x:

1. No transformation

2. Δx_t

3. $\Delta^2 x_t$

4. $\log(x_t)$

5. $\Delta\log(x_t)$

6. $\Delta^2\log(x_t)$

7. $\Delta(\frac{x_t - x_{t-1}}{x_{t-1}})$

The FRED column gives mnemonics in FRED followed by a short description. Some series require adjustments to the raw data available in FRED. These variables are tagged by an asterisk to indicate that they have been adjusted and thus differ from the series from the source. For a detailed summary of the adjustments see McCracken and Ng, 2016.

Appendix B

B.1 Line Plots

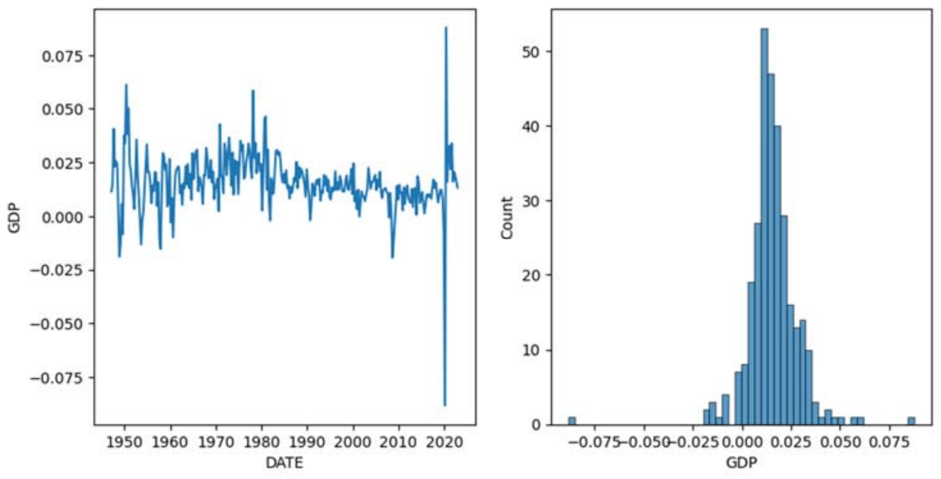


Figure 15: GDP Line Plot and Distribution.

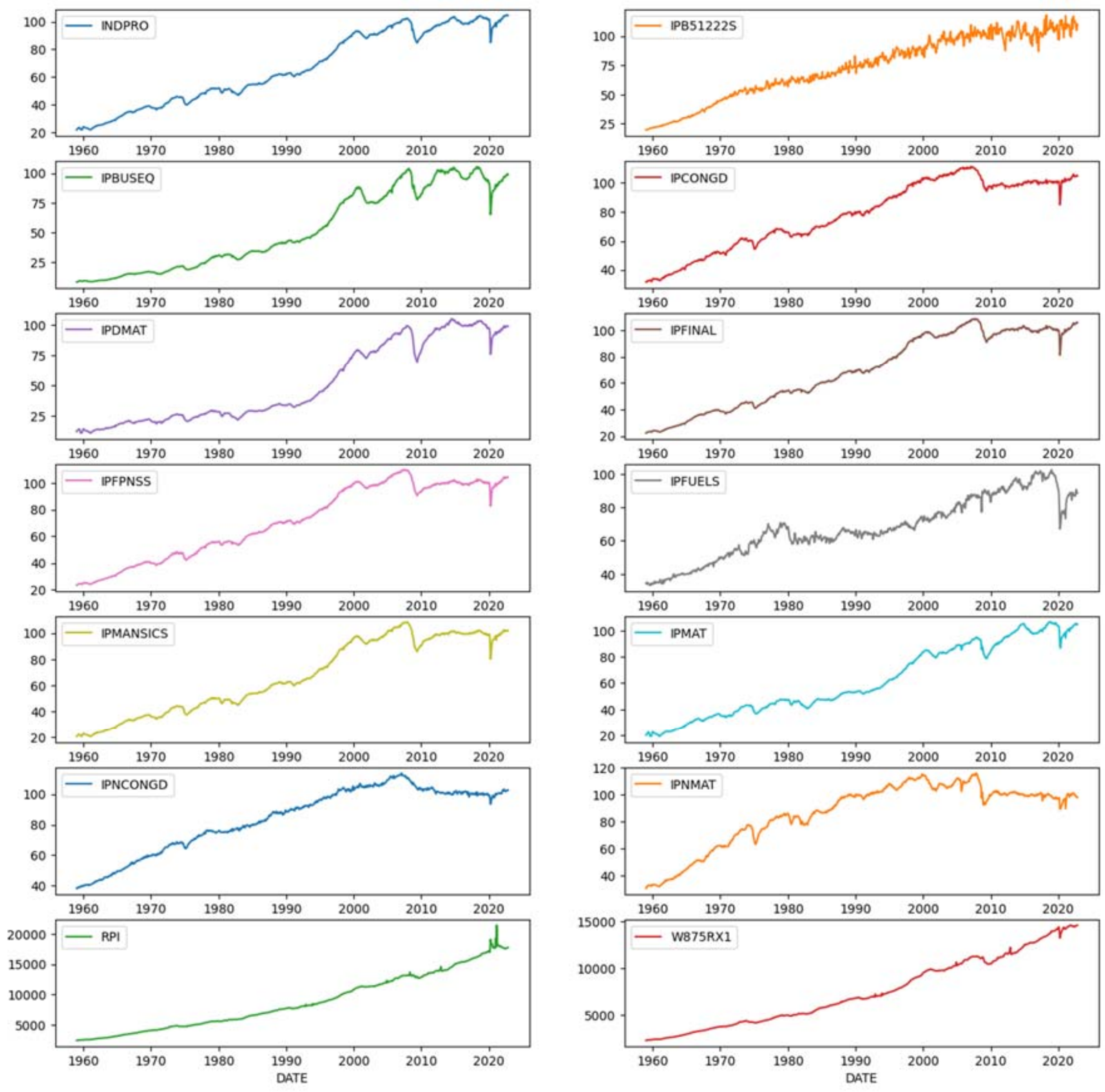


Figure 16: Line plots for Group 1.

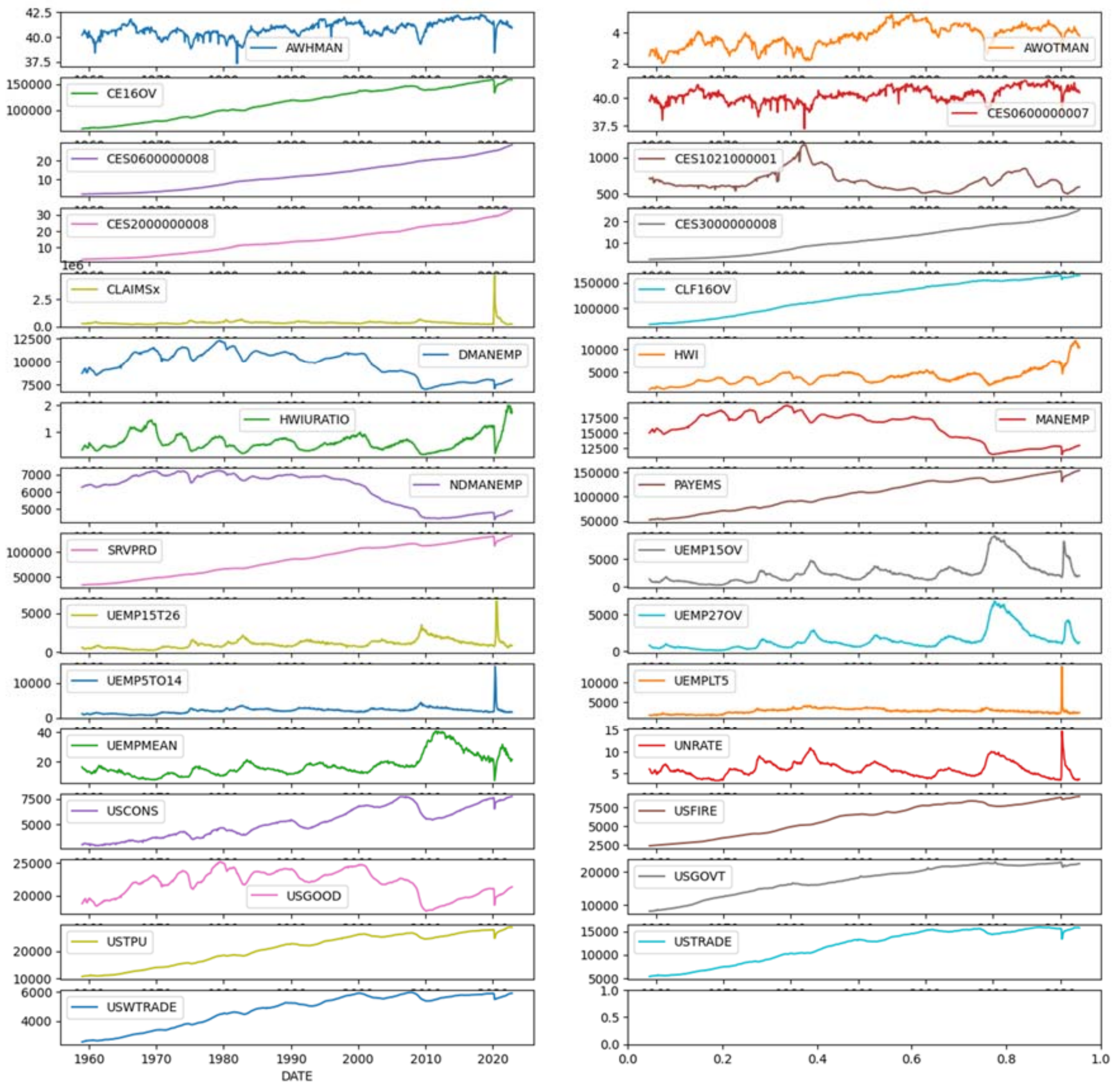


Figure 17: Line plots for Group 2.

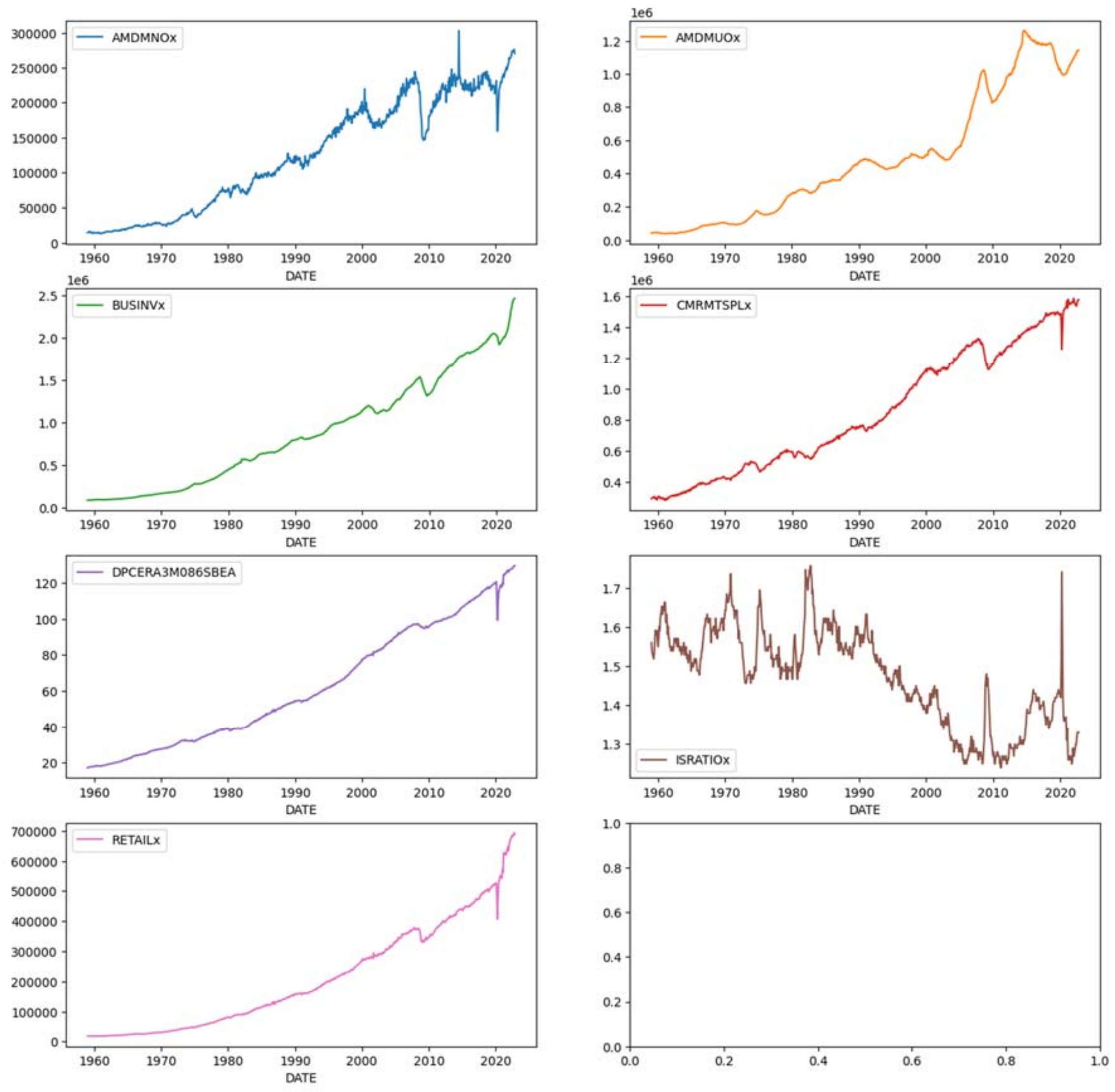


Figure 18: Line plots for Group 4.

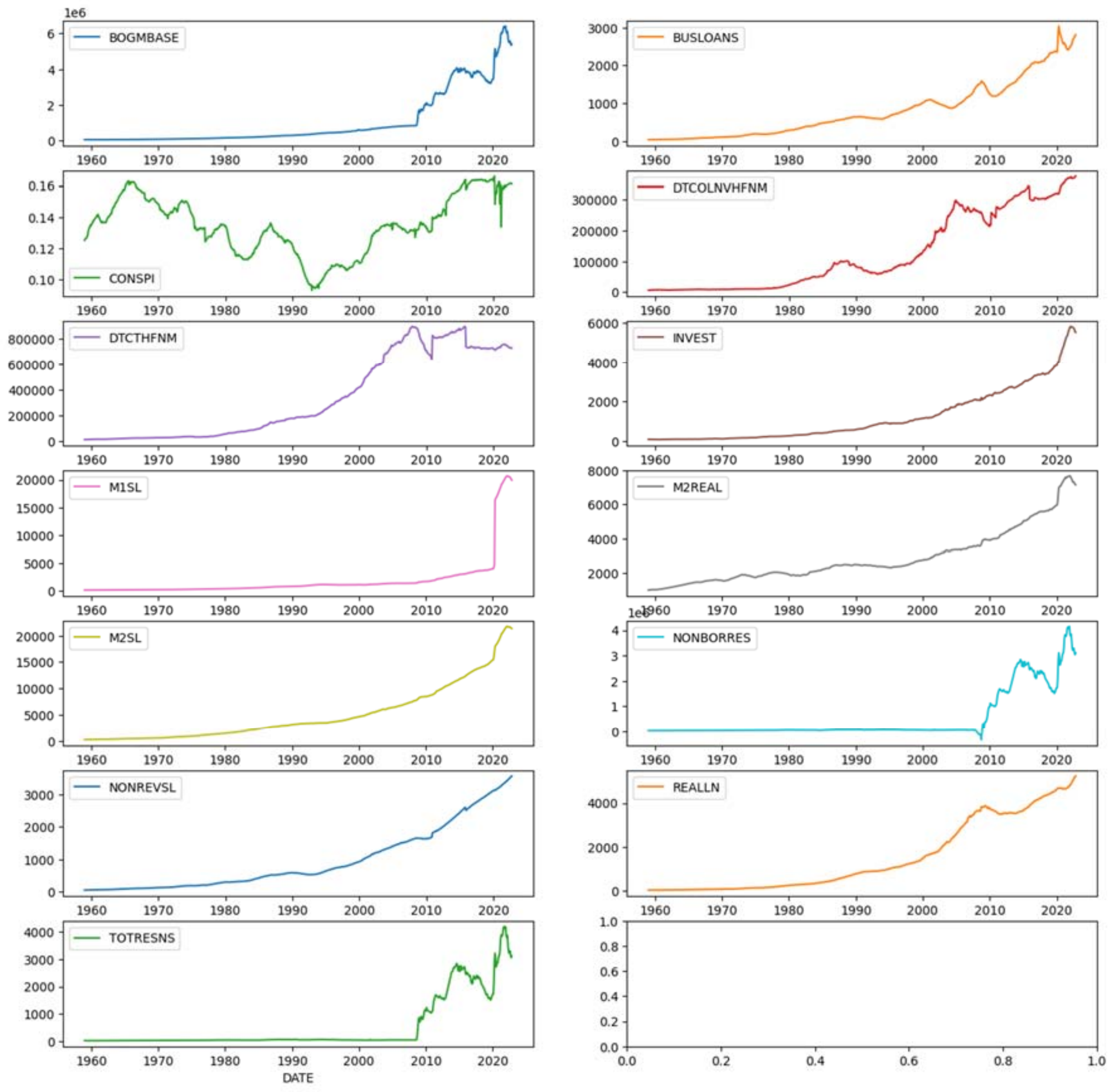


Figure 19: Line plots for Group 5.

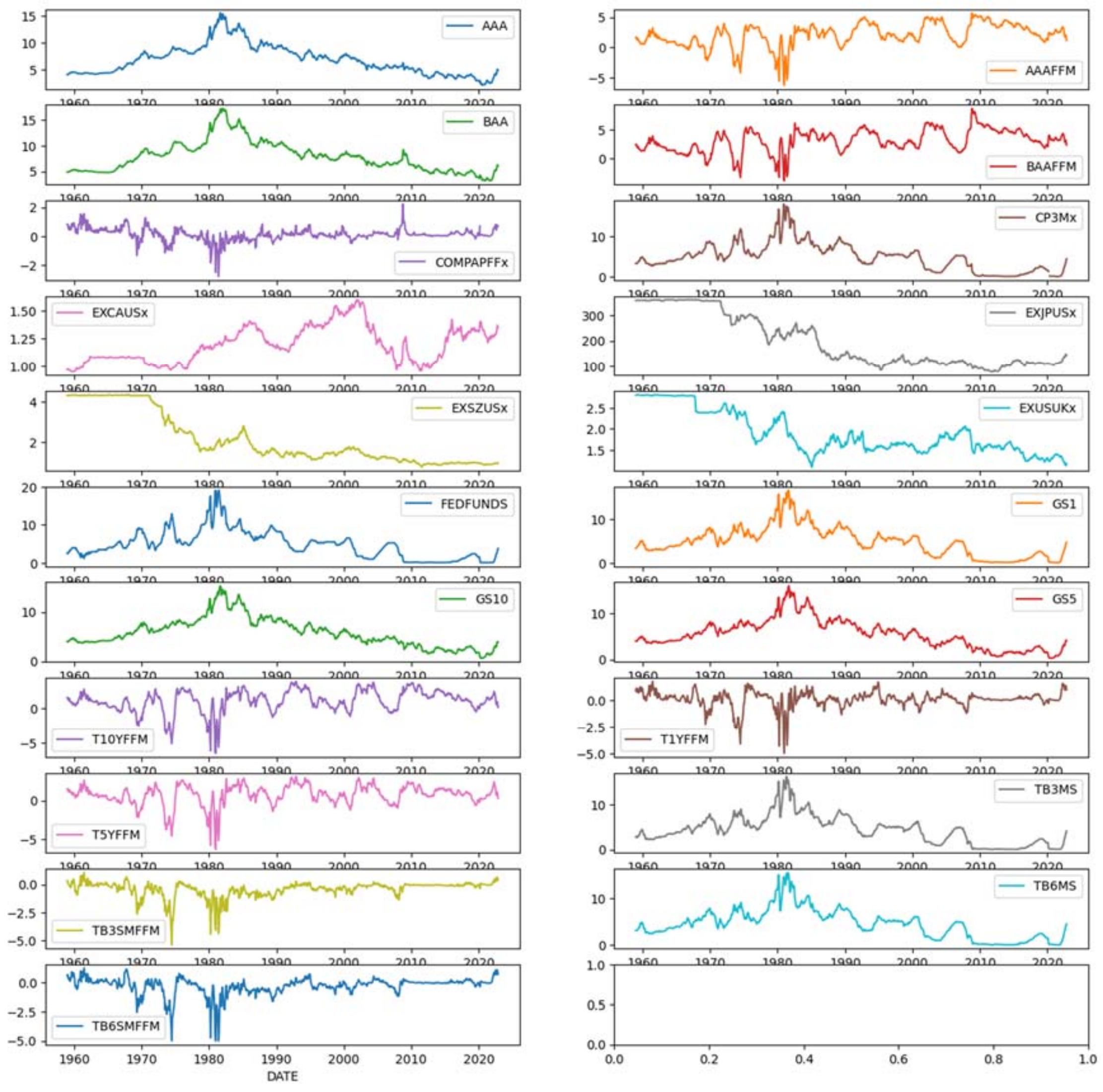


Figure 20: Line plots for Group 6.

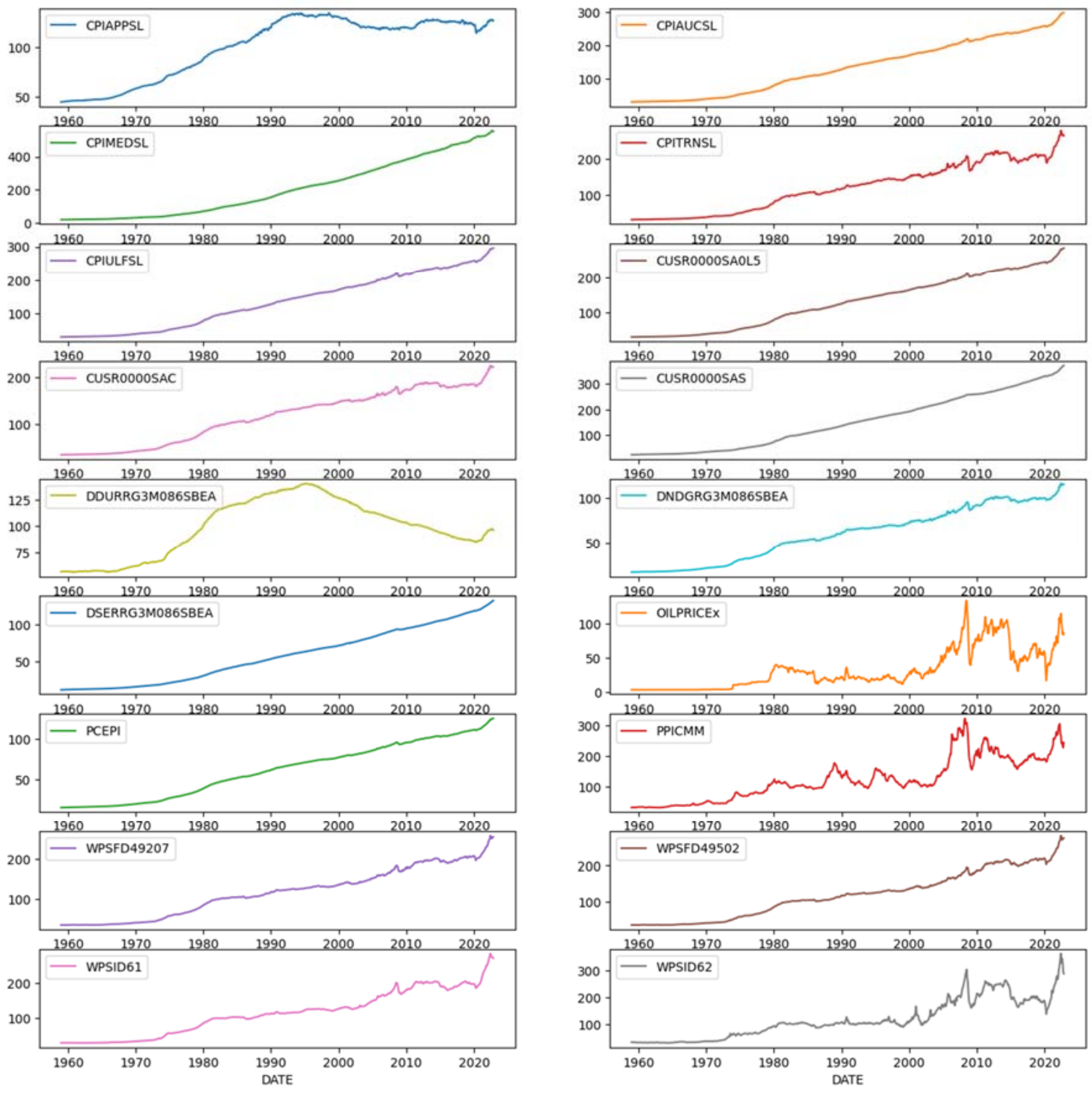


Figure 21: Line plots for Group 7.

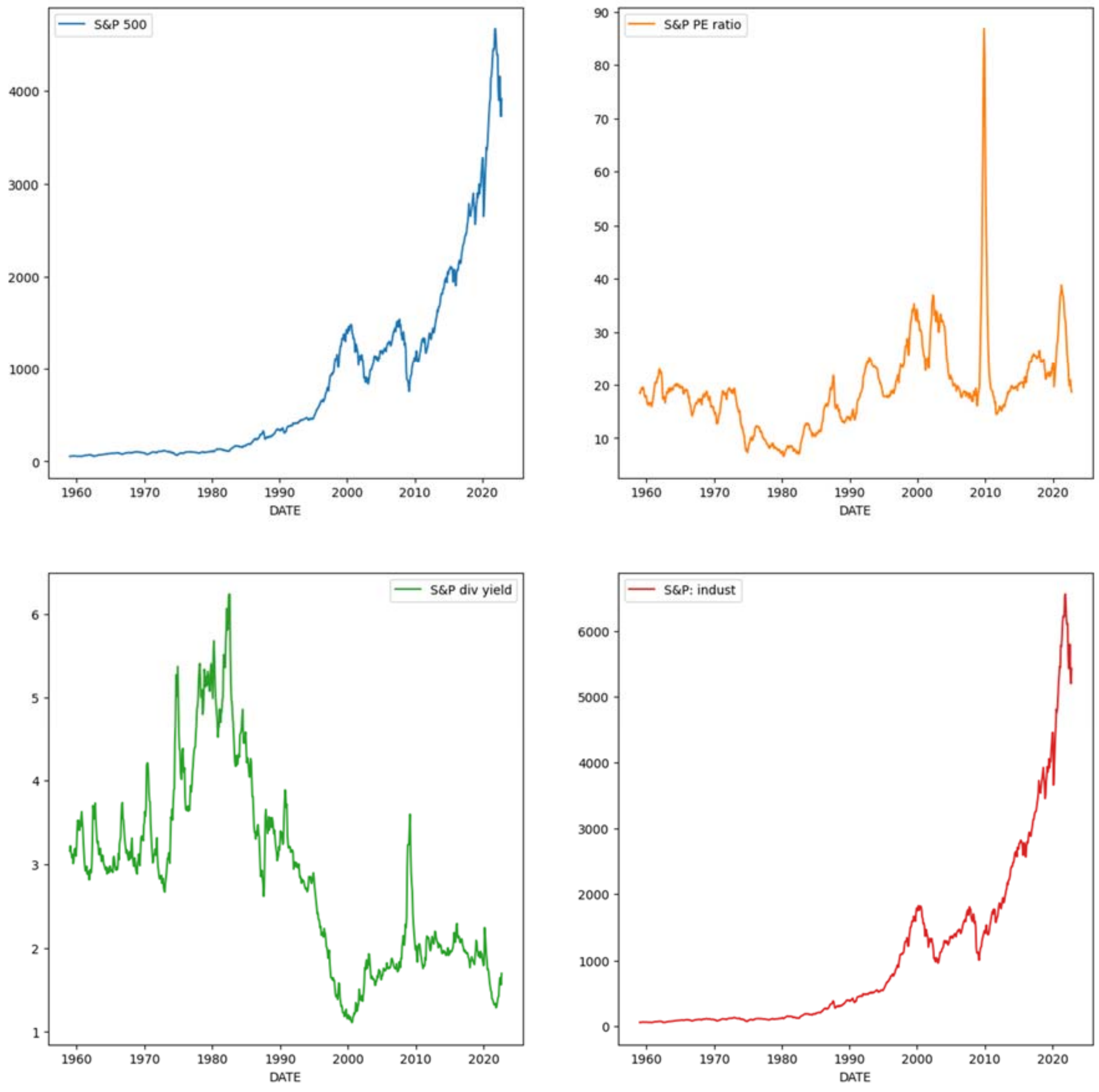


Figure 22: Line plots for Group 8.

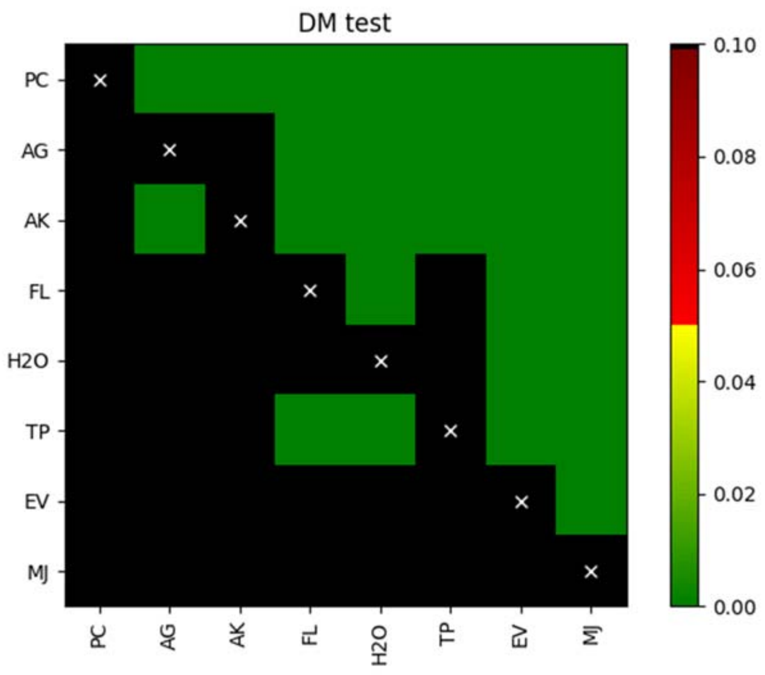


Figure 23: Multivariate Diebold-Mariano Test Plot, 1-month, 2010-2019.

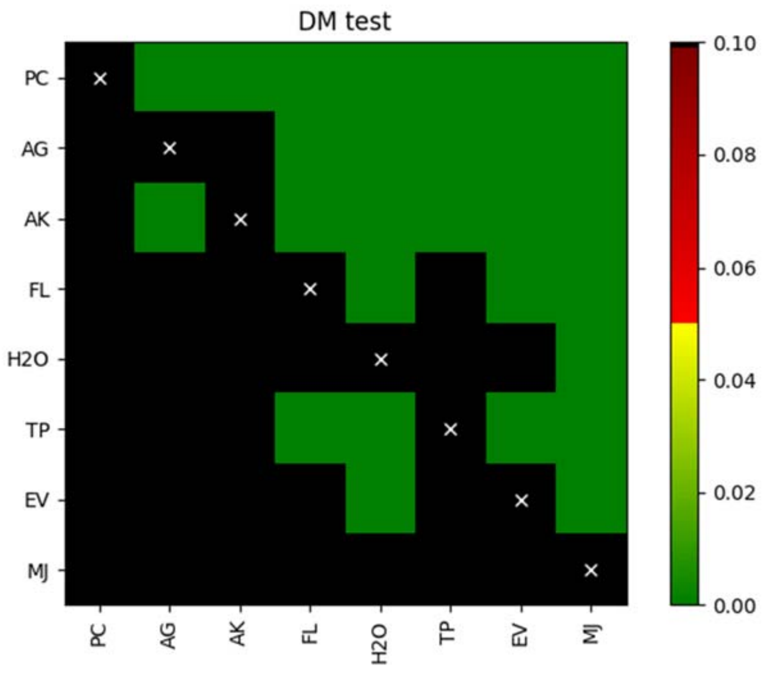


Figure 24: Multivariate Diebold-Mariano Test Plot, 2-month, 2010-2019.

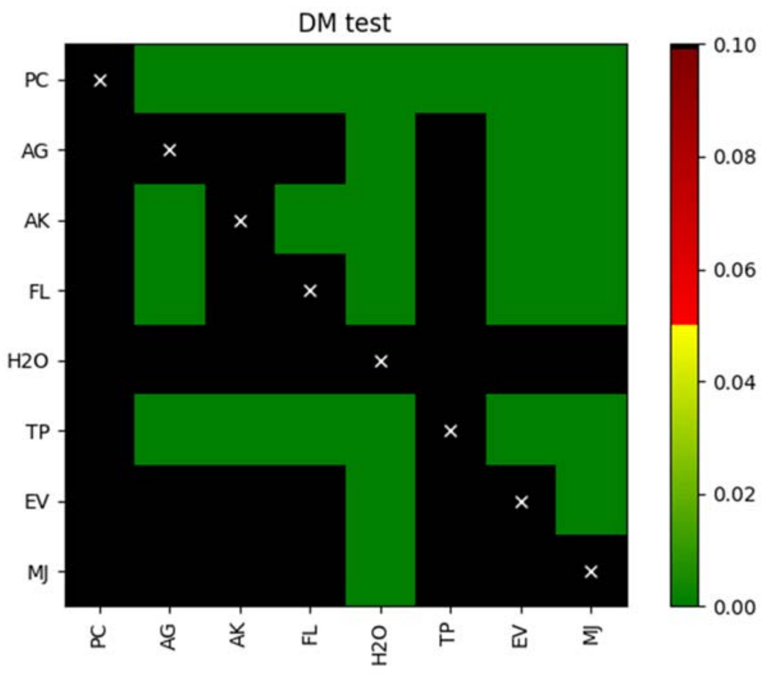


Figure 25: Multivariate Diebold-Mariano Test Plot, 3-month, 2010-2019.

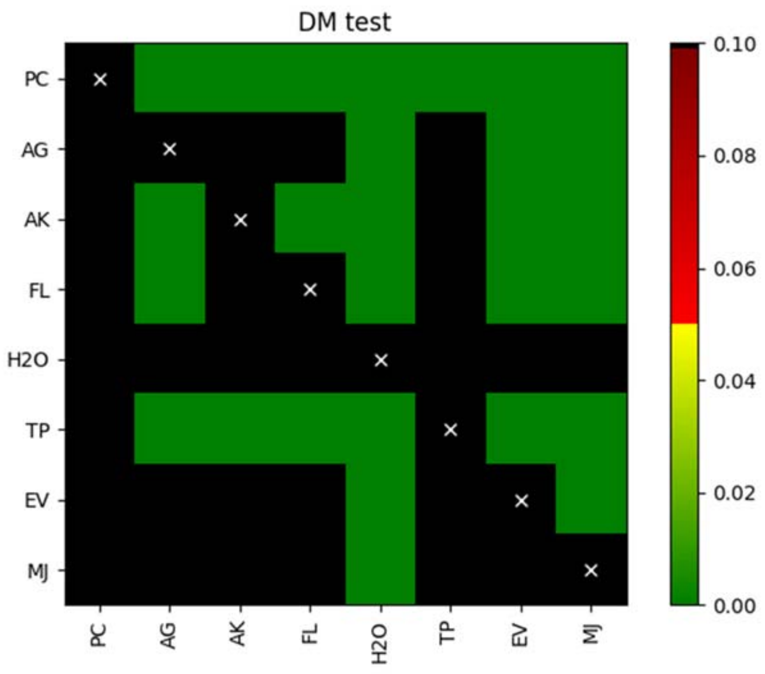


Figure 26: Multivariate Diebold-Mariano Test Plot, 1-month, 2010-2022.

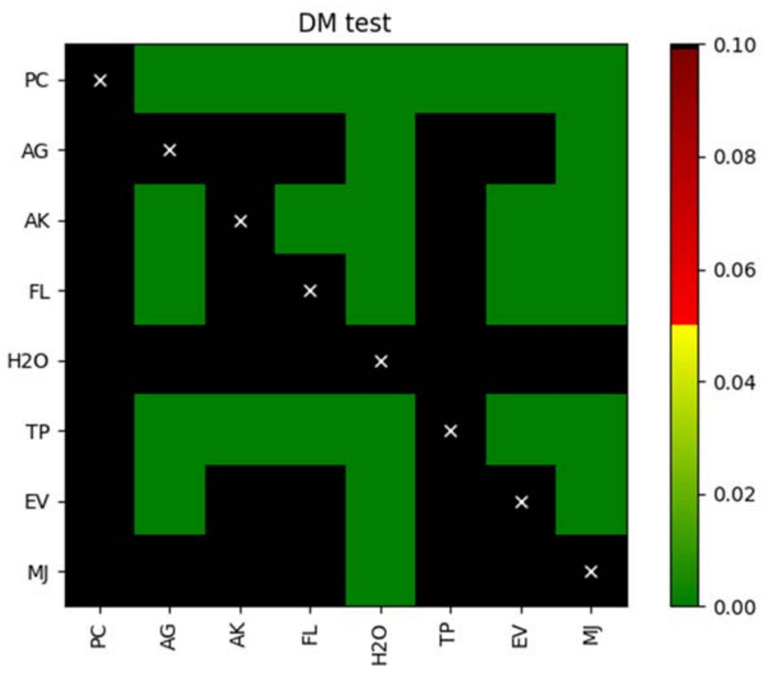


Figure 27: Multivariate Diebold-Mariano Test Plot, 2-month, 2010-2022.

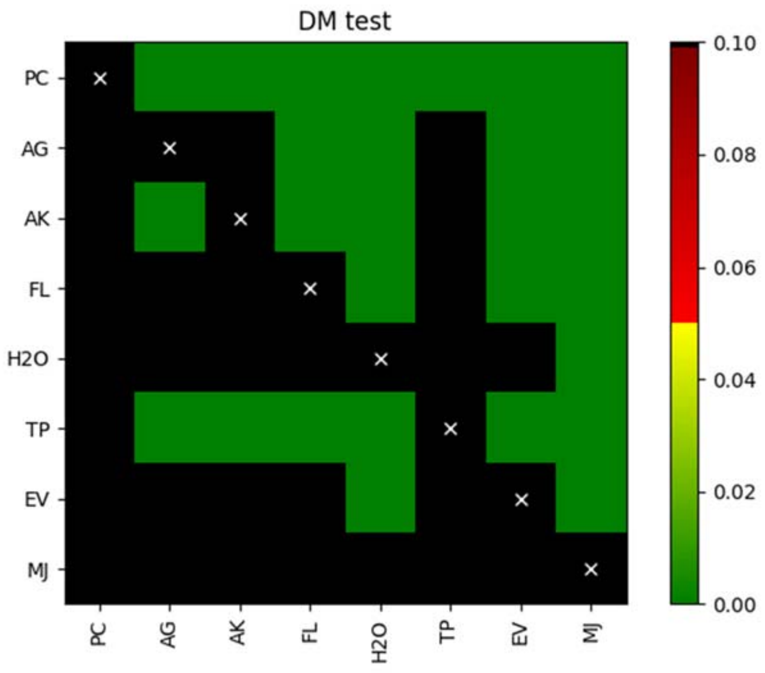


Figure 28: Multivariate Diebold-Mariano Test Plot, 3-month, 2010-2022.