Is Black-Box Modeling the Future of Macroeconomic Forecasting?

Sahil Nisar (sn3028@nyu.edu)

Suniya Raza (sr5748@nyu.edu)

Vinicius Moreira (vgm236@nyu.edu)

Graduate School of Arts and Science (GSAS) at New York University (NYU)

19 West 4th Street

New York, NY 10003

2022

This paper was presented at the Special Projects In Economic Research, advised by Prof. Irasema Alonso in May 2022 at the Graduate School of Arts and Science program of Master of Arts in Economics.

Summary

In this paper, we aim to provide a comparison of machine learning and more traditional econometric techniques to forecast US GDP. Machine learning (ML) algorithms have become increasingly popular specially over the last two decade, and their application has been wide ranging. This popularity has meant that it was about time before the ML algorithms were used to forecasting macro variables. Through this paper, we have used key ML algorithms such as K-Nearest Neighbor, Random Forest, Dynamic Factor Model and Vector Support Machines, Elastic Net and Decision trees to forecast the US GDP. More importantly, we have also used the traditional time series methods such as ARIMA, OLS and VAR to draw comparison between the ML and these methods. Based on our findings, ML models perform better than the traditional time series methods such as VAR and ARIMA but perform worse than basic estimation method such as OLS and more complex econometric model such as Dynamic Factor Model. This study is only limited to forecasting of the US GDP and other results might be different specially for high frequency, widely available variables. Within the scope of this study, ML algorithms have shown only incremental impact for macroeconomic forecasting, much less so than in other areas.

1. Introduction

Forecasting macroeconomic and financial variables has always been a challenge, especially since this requires forecasters to make discretionary choices about the data and methodology. Even though most choices have their basis in economic intuition and judgment, there is a high possibility that the assumptions made could be flawed. This concern has been aggravated especially due to the inability of traditional methods to accurately perform during unprecedented times such as the recent COVID-19 pandemic. The predictors and models that have proved to be useful for predictions in normal times have failed to generate appropriate insights during the time of crises.

Machine Learning (ML) techniques can automate most of the choices regarding model selection and specifications, allowing the algorithm to learn from the data without any programmer explicitly specifying steps in the model. The ability of Machine Learning techniques to identify the non-linearities that exist in data series as well as how variables interact with each other has contributed to the predictive power of these models. In today's world, machine learning algorithms are successfully and widely used for sales forecasting, predicting stock prices, designing banking softwares to detect frauds and by search engines to make personalized recommendations based on search history. The discretionary nature of traditional forecasting models as well as the high predictive power of ML techniques has led macroeconomic forecasters to turn their attention to the development and application of Machine Learning techniques to forecast the macroeconomic variables. Moreover, with increased availability of "big data" in economics, the development of machine learning techniques for macroeconomic forecasting can be promising.

While the formulation of the machine learning techniques dates to the 1950's, its application to macroeconomics is still in the development phase. This has a lot to do with the fact that the predictive power of the ML algorithms is contingent on the availability of data. The frequency with which most of the key macroeconomic variables are reported is montly and quarterly, which limits the number of observations available to make good predictions using the ML algorithms. To put things in perspective, the data on stock prices is updated in real time, meaning that the available observations are in abundance. However, with the evolution of time, enough data has accumulated to make use of ML for macroeconomic forecasting.

In this paper, we aim to provide a primer of different models followed by a comparative analysis of the traditional time series models with the machine learning models to forecast the US Gross Domestic Product (GDP) growth rate. GDP is the total amount of goods and services produced within the confines of the country in the specified period. It is generally considered to be a good proxy for the wealth of the economy and hence drives a lot of decision-making processes (Provost and Fawcett, 2013). This makes GDP forecasting a key and a relevant issue in macroeconomics. In this research, we aim to study varying classical econometric models and machine learning algorithms, applying them to forecast US real GDP growth rate. The goal is to assess the performance of each of these models and understand which model provides the best and robust forecast in the specified setting.

For the classical time series models, we have mainly used the Auto-Regressive (AR), Moving-Average (MA), Auto-Regressive-Integrated-Moving-Average (ARIMA) and Seasonal ARIMAX (SARIMAX), Ordinary Least Squares (OLS), Ridge/Lasso/Elastic net regressions and the Dynamic Facator Model. For the Machine Learning models, we have used the K-Nearest Neighbor (KNN), Random Forest, Decision Trees and Neural Networks. Finally, to assess the performance of the models we have used the Root Mean Squared Error (RMSE). We are particularly interested in observing the accuracy of the estimates provided by each of our models for the specified forecasting horizon.

The rest of the paper is organized as follows: Section 2 dives deeper into the existing literature available on the application of machine learning in macro forecasting, Section 3 provides some insights into the summary statistics of the data used, Section 4 lays out our forecasting methods and specifications, Section 5 demonstrates our empirical results and analysis, Section 6 compares and discuss these models, and Section 6 concludes.

Further details and code of the models and transformation used can be found in this GitHub link: https://github.com/vgm236/Special_Project_Moreira_Raza-Nisar.

2. Literature Review

The intersection of Machine Learning (ML) with econometrics has recently gained prominence in macroeconomic application, especially due to the increase in the availability of large datasets. Even though only recently there has been a surge in the number of studies that apply ML techniques for macroeconomic forecasting (Surprenant et al, 2019), machine learning does have a long history in macro-econometrics.

A 1993 study tested the adequacy of using linear models for forecasting time series by using a popular ML model, Neural Networks to check for neglected non-linearities in time series. The results showed that neural network test performs as well or better than standard tests and helps prove the presence of non-linearities in data series (Lee et al. (1993)). Swanson and White also use Artificial Neural Networks and other linear models to evaluate how useful these approaches are to predict nine macroeconomic variables (Swanson and White (1997)).

A 1999 working paper has also tested the forecasting performance of neural network model in predicting GDP at 1-quarter and 4-quarter forecast horizons, with only the 4-quarter horizon results showing statistically significant forecasting accuracy over linear model counterparts (Tkacz and Hu, 1999-2003). Similarly, another paper focused on using neural networks to forecast Canada's real GDP growth and concluded that for both in-sample and out-of-sample periods, this model performed better than the traditional linear regression models. However, in this case the author was unable to prove that the improvement in forecasting accuracy was statistically significant (Gonzalez, 2000-2007).

In recent years, the interest in neural networks has increased because of its potential to identify and reproduce linear and non-linear relationships among a set of variables. A recent paper by researchers at the Federal Reserve Bank of Kansas City has explored the potential of deep neural networks in improving the forecasting accuracy of macroeconomic variables, despite the existence of limited data. The results show that the model outperforms benchmark models at forecasting civilian unemployment at short horizons (Cook et all, 2017). In our research, we aim to explore this

ML technique using recent data of GDP growth rates, especially in context to shocks such as the financial crises and the COVID-19 pandemic.

Another technique, the non-parametric K-Nearest Neighbor (KNN) is one of the oldest and most researched ML models and is known for its consistency property (Stone, 1977). KNN has the ability to identify repeated patterns within time series and is widely used in financial time series modeling (Ban et all, 2013). KNN has gained popularity in predicting macroeconomic variables, since it retains its consistency properties even in cases of limited past information i.e., few data points. As compared to other artificial intelligence methods, which require a minimum number of observations to perform adequately, KNN application is not severely hindered by this limitation (Diebold and Nason, 1990). A recent research paper compares GDP forecasts from KNN model with results from basic time series models such as Autoregressive (AR), Autoregressive Integrated Moving Average (ARIMA), Seasonal AR and ARIMA and AR and ARIMA with covariates. The study concludes that KNN captures the self-predictive ability of GDP better than other traditional models (Maccarrone et all, 2021). These results are promising, especially since the data used covers the financial crisis and pandemic time period. Another paper compares KNN technique with two other ML techniques, the Random Forest and Extreme Gradient Boosting for predicting inflation in Costa Rica and concludes that KNN outperforms these models based on forecast accuracy (Rodriguez-Vargas, 2020).

Random Forest algorithms are widely used in medical research and biological studies. This technique is known for its prediction accuracy and robustness since it can handle a very large number of input variables without overfitting. However, it is not as widely used in macroeconomic forecasting yet. Baiu and D'Elia (2011) have used Random Forest model to forecast the quarterly GDP growth rate for the Euro region and concluded that while RF fails to outperform standard AR model in predicting growth rate, it can be used as a tool to select the most relevant predictive variables.

Principal Component Analysis (PCA) is a machine learning technique used specifically when there is a large amount of data available. Large datasets not only make computation hard but also make it difficult to interpret. PCA tends to reduce the dimensionality of such datasets, while at the same time ensuring that the information loss is minimum. The way it achieves that is by creating new uncorrelated variables in order to attain maximum variance. PCA essentially reduces the problem to solving eigenvalues and eigenvectors. Stock and Watson have proved that under the general conditions on the error terms, principal components of X (explanatory variables) are a consistent estimator of the true latent factors. In their research, Stock and Watson tried to forecast the Federal Reserve Board's Index of Industrial Production using the PCA approach and found that the model outperforms univariate AR and VAR models (Stock and Watson, 2002).

Alternatively, another variation of the PCA is the Dynamic Factor Model (DFMs). DFM has gained prominence over the last decade mainly because of their ability to model sequential datasets consistently, specially when the number of series exceeds the number of time series observations. DFM was first introduced by Geweke (1977) as an addendum to time series factor models that were mainly developed for cross-sectional data. The idea of DFM is similar to the PCA, except that they introduce the dynamic part to it through a Kalman Filter. Sargent and Sims (1977) work has extensively examined the macroeconomic variables using the dynamic factor model and have found that the models have the potential to explain a large variation in them (Stock and Watson, 2010).

Classical time series forecasting models such as AR, MA, ARIMAX and SARIMAX have always remained the go to models for the forecasters mainly because they are structured. With the advancement of machine learning techniques, the classical time series forecasting models are now becoming benchmark models to which the other non-traditional models of forecasting are compared. Classical time series models mainly rely on assumptions about the distribution of the error terms and linear relationship between the dependent and independent variables. Some complex time series models modify these assumptions somewhat, but the structure is still imposed. ARIMAX model accounts for the stationarity of the data and makes use of other explanatory variables and SARIMAX accounts for the seasonality on top of ARIMAX.

One of the advantages of classical time series models over machine learning models is that they provide a useful interpretation because of the parametrization as opposed to ML. On the other hand, multivariate time series models tend to increase the complexity mainly because of the non-linearity. According to Banerjee and Marcellino (2006), univariate linear models tend to be often more robust than their multivariate counterparts. However, this doesn't imply that multivariate models are not useful. For the purpose of our analysis, we will be using the combination of both univariate and multivariate time series models. The idea is to use the classical econometric models as a benchmark because they have been regarded as the gold standard for forecasting and have been widely used for over many years for macroeconomic forecasting.

3. Data and Summary Statistics

3.1 Data description

The source of the data used in this paper is the Fred, a database created and maintained by the Research Department at the Federal Reserve Bank of St. Louis. The database consists of hundreds of thousands of economic data time series from scores of national, international, public, and private sources. Our models will be univariate and multi-variate, in which case we will use the monthly variables for the US, and their transformations, to predict the US GDP growth. We got eleven monthly data series and one quarterly data series, the GDP. All of these series are seasonally adjusted.

The series labeled "payroll" is the Total Nonfarm Payroll, a measure of the number of U.S. workers in the economy that excludes proprietors, private household employees, unpaid volunteers, farm employees, and the unincorporated self-employed. This measure accounts for approximately 80 percent of the workers who contribute to Gross Domestic Product (GDP). The Bureau of Labor Statistics (BLS) adjusts the data to offset the seasonal effects to show non-seasonal changes. The series labeled "job_openings" is the Total Nonfarm Job Openings, which measures all jobs that are not filled on the last business day of the month. A job is considered open if a specific position exists and there is work available for it, the job can be started within 30 days, and there is active recruiting for the position. Total Nonfarm Job Openings are measured by the Job Openings and Labor Turnover Survey (JOLTS) and published by the Bureau of Labor Statistics (BLS).

The series named "quits" measures the employees who left voluntarily, with the exception of retirements or transfers to other locations. This numbers is divided by the total employment to return the quits rate. For "ip" we collect the industrial production of the US, the Board of Governors

of the Federal Reserve System. It measures real output for all facilities located in the United States manufacturing, mining, and electric, and gas utilities (excluding those in U.S. territories). The series is determined from 312 individual series based on the 2007 North American Industrial Classification System (NAICS) codes. On "vehicle_sales" is the vehicle sales data for the US, also seasonally adjusted, and published by the U.S. Bureau of Economic Analysis. The seasonally adjusted series includes auto (all passenger cars, including station wagons) and truck sales data.

We classify as "retail" the Advance Retail and Food Services Sales, deflated using the Consumer Price Index for All Urban Consumers. As the other series, it is also seasonally adjusted. Also, for "wholesale_inventories" which measures the inventories of wholesalers, not including retailers. The estimates are based on data from three surveys: the Monthly Retail Trade Survey, the Monthly Wholesale Trade Survey, and the Manufacturers' Shipments, Inventories, and Orders Survey. We use sentiment data as well, with the "consumer_sentiment" data is the Consumer Sentiment indicator measured by the University of Michigan. To calculate the Index of Consumer Sentiment (ICS), the University computes the relative scores (the percent giving favorable replies minus the percent giving unfavorable replies, plus 100) for each of the five index questions, including current financial conditions, expectations of financial conditions, expectations of business conditions, expectations of unemployment and conditions to buy households items.

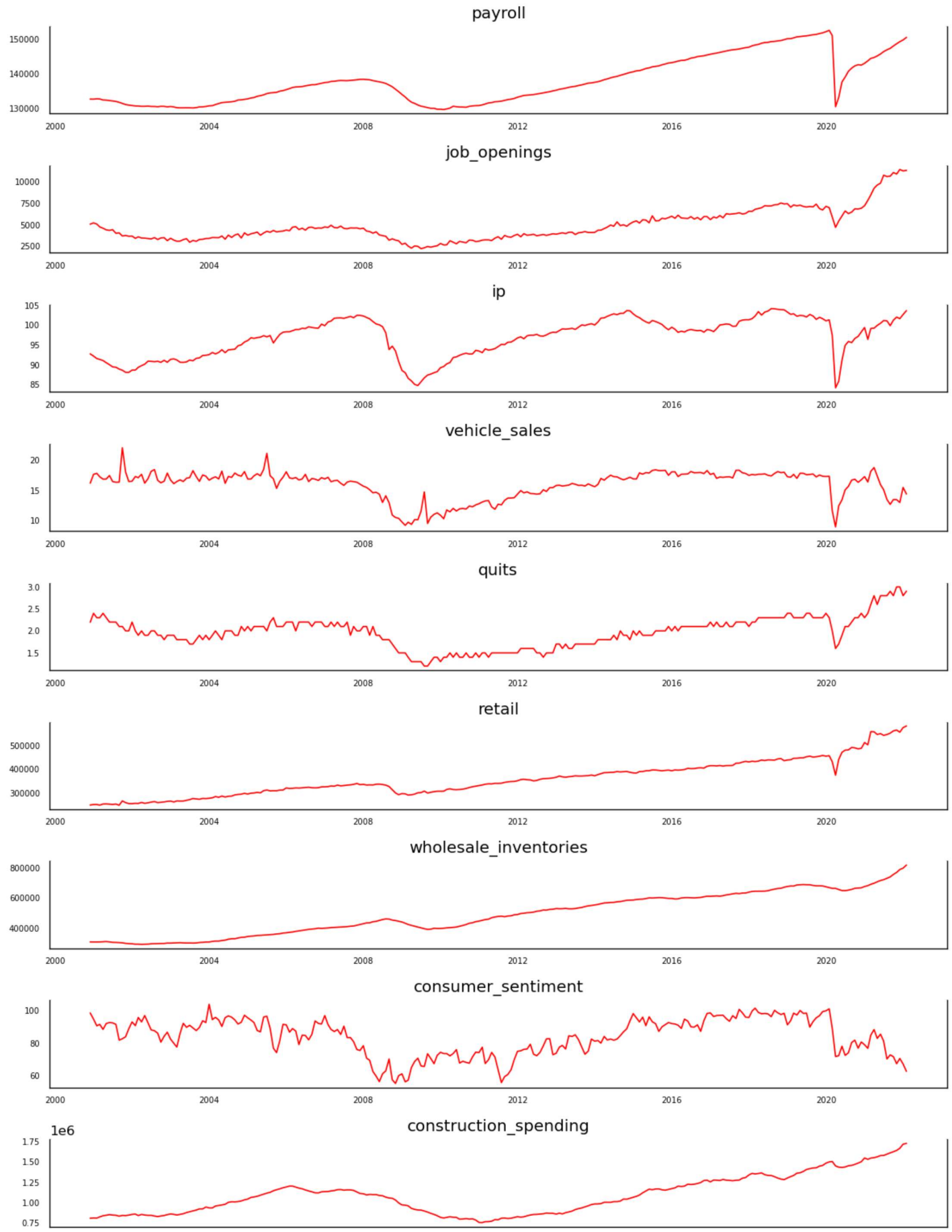
We label as "construction_spending" measures the total dollar value of construction work done in the United States (including 50 states and the District of Columbia). Composite estimates are based on mail-out/mail-back and interview surveys of selected construction projects and building owners, and estimates developed or compiled from other Census Bureau, federal agency, and private data sources. For "new_orders" indicates the new orders of manufacturing products. This data comes from the Manufacturers' Shipments, Inventories, and Orders (M3) survey, which provides broad-based monthly statistical data on current economic conditions and indications of future production commitments in the manufacturing sector. Finally, for New Privately-Owned Housing we label as "housing_starts." As provided by the Census, start occurs when excavation begins for the footings or foundation of a building. All housing units in a multifamily building are defined as being started when this excavation begins. Beginning with data for September 1992, estimates of housing starts include units in structures being totally rebuilt on an existing foundation.

The only originally quarterly data used is the GDP series, which we unsurprisingly label as "gdp" for all models in this study. Gross domestic product (GDP), the featured measure of U.S. output, is the market value of the goods and services produced by labor and property located in the United States.For more information, see the Guide to the National Income and Product Accounts of the United States (NIPA) and the Bureau of Economic Analysis.

3.2 Summary Statistics

The data is seasonally adjusted, and therefore won't require transformations of such kind. However, all the the original monthly data are in levels and an investigation of stationarity is required. We show below the summary statistics of the data series and a chart for each of them. Then, we use the Augmented Dickey-Fuller test to check for stationarity in the set of data. None of the eleven series used pass the test at a 10% level. In this sense, a transformation of these series is required. We

perform the same transformation for all of them: we will use log differences - the continuous equivalent of a percentage change - from now on.



	payroll	job_openings	ip	vehicle_sales	quits	retail	wholesale_inventories	con
count	255.000000	255.000000	255.000000	255.000000	255.000000	255.000000	255.000000	
mean	137896.482353	4881.929412	96.870191	15.949416	1.964314	361238.380392	491485.733333	
std	6681.601392	1838.401365	4.894543	2.317720	0.352970	76374.464473	138074.360927	
min	129698.000000	2232.000000	84.201800	8.961000	1.200000	249845.000000	293159.000000	
25%	131969.000000	3553.000000	92.746900	15.032000	1.700000	303957.000000	376982.500000	
50%	136520.000000	4373.000000	98.285400	16.761000	2.000000	341787.000000	476187.000000	
75%	143151.000000	5856.500000	100.949750	17.508000	2.200000	406128.000000	606267.000000	
max	152504.000000	11448.000000	104.165900	22.055000	3.000000	582698.000000	818209.000000	

	ADF Fuller test
payroll	-0.918527
job_openings	0.904821
quits	-1.083210
retail	1.564151
ip	-1.957387
vehicle_sales	-2.592540
wholesale_inventories	1.113354
consumer_sentiment	-2.279565
construction_spending	0.974425
new_orders	-1.831629
housing_starts	-1.427559
10% threshold	-2.573096

Since we need to forecast a quarterly data, the GDP, we then have to turn our monthly proxies into quarterly data. In this sense, to turn the monthly series into quarterly, we perform the following

transformations. First, we calculate the quarterly average of each variable and create a quarterly series for them. After this proceeding is done, we perform the log differences for the quarterly series. For GDP, we only need to run the log differences as the series is already quarterly by construction.

4. Forecasting methods

In this section, we briefly describe the methodology behind each of the models used. After that discussion, we then delve into the modeling process and the empiric results.

4.1 Univariate Models

Trailing 3-period average:

This simple estimator plays the role of a naïve benchmark. It is an average of three periods, which can be 3-months or 3-quarters, depending on the variable used in our analysis.

$$\hat{\mu}_{t+1} = \frac{1}{T}\sum_{t=1}^{t=3}\hat{\mu}_t$$
$$\text{where } \hat{\mu}_t \text{is the independent variable}$$

Exponential Smoothing:

A weighted average of lagged values, with weights decaying exponentially the longer the lag. Exponential smoothing takes into account all past data, whereas moving average only takes into account k past data points.

$$X_{t+1} = \alpha X_t + \alpha(1-\alpha)X_{t-1} + \alpha(1-\alpha)^2 X_{t-2} + \dots$$

where $0 \leq \alpha \leq 1$ is the smoothing parameter. You choose how many lags to use. We will use four lags here.

ARIMA:

A stochastic process $\{X_t\}$ is called an *autoregressive moving average process*, or ARMA(p, q), if it can be written as

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}$$

where $\{\epsilon_t\}$ is white noise.

In what follows we **always assume** that the roots of the polynomial $\phi(z)$ lie outside the unit circle in the complex plane. This condition is sufficient to guarantee that the ARMA(p, q) process is covariance stationary. In fact, it implies that the process falls within the class of general linear processes. We define an ARIMA(p, d, q) model as the mixture of an AR(p) and MA(q) model with differencing (to help make the process stationary)

4.2 Linear Regression and Machine Learning Models

Simple Linear Regression (OLS):

The most common technique to estimate a linear relationship between variables is Ordinary Least Squares (OLS). OLS model is solved by finding the parameters that minimize the sum of squared residuals.

The model can be defined, in the matrix form, as:

$$y = X\beta + u$$

To solve for the unknown parameter β, we want to minimize the sum of squared residuals

$$\min_{\hat{\beta}} \hat{u}'\hat{u}$$

Rearranging the first equation and substituting into the second equation, we can write

$$\min_{\hat{\beta}} \ (Y - X\hat{\beta})'(Y - X\hat{\beta})$$

Solving this optimization problem gives the solution for the $\hat{\beta}$ coefficients

$$\hat{\beta} = (X'X)^{-1}X'y$$

Ridge / Lasso / Elastic Net:

These models are very closely related to traditional OLS, but they focus on regularization of parameters to avoid overfitting. The Lasso model generates predictions using but optimizes over a slightly different loss function:

$$\min_{\hat{\beta}} \ (Y - X\hat{\beta})'(Y - X\hat{\beta}) + \alpha\hat{\beta}$$

where α is the regularization parameter. The additional term penalizes large coefficients and in practice, effectively sets coefficients to zero for features that are not informative about the target. Ridge regressions places a particular form of constraint on the parameters β, which is chosen to minimize the penalized sum of squares:

$$\min_{\hat{\beta}} \ (Y - X\hat{\beta})'(Y - X\hat{\beta}) + \lambda\hat{\beta}'\hat{\beta}$$

This means that if the β take on large values, the optimization function is penalized, but not zero (only reducing the impact of "irrelevant" features of the model). The elastic net algorithm uses a weighted combination of Ridge and Lasso forms of regularization.

4.3 More Complex Econometric Methods

Vector autoregressions (VARs):

The Vector Autoregression models (VARs) are an extension of the AR model described before, in a multi-variate form used when two or more time series can influence each other. This parametric model is built as an equation that includes the variable's lagged (past) values, the lagged values of the other variables in the model, and an error term.

An important part of the VAR models, as it is for the AR version, is the order of the model - the number of lags one wants to include in its model. In the example below we represent as an infinity of lags. However, on the data analysis it would be required that we perform a lag criterion test, such as AIC, PIC and SIC.

In the example of equation below, we describe an VAR model where the X_t an the independent variable, which we are interested in modeling, X_{t-T} are the lags of the independent variable. In the meantime, Y_{t-T} is a matrix that represents all the lagged explanatory variables used in the model. Again $\{\epsilon_t\}$ is white noise.

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \epsilon_t$$

Dynamic Factor Model:

In a dynamic factor model, we model a potentially large number of macroeconomic series as being driven by a much smaller number of latent factors, which are estimated, which involves a principal component analysis. Principal component analysis is an unsupervised algorithm, based on feature correlation, used for dimensionality reduction. The premise is simply to take data of higher dimensions, and reduce to a lower dimension. Oftentimes, in higher dimensional data, it isn't possible to create visual representations of relationships between variables. Through applying PCA, it then becomes possible to reduce the dimensions of the data and display variable relationships. This tool also allows easier visualization and noise filtering, among other applications. The PCA must be used when three conditions apply: 1. Reduce the number of variables 2. Ensure that each variable is independent of one another 3. Assume that the interpretation of the independent variables is less important

For our specific case, we use a state space model to try and explain the GDP data we observe by using only one unobserved factor, extracted from a series of monthly data.

Define our state space model as:

$$\begin{aligned} \lambda_t &= A\lambda_{t-1} + \eta_t \\ p_{i,t} &= G_i\lambda_t + \varepsilon_{i,t} \\ \eta_t &\sim N(0, I) \\ \varepsilon_t &\sim N(0, \Sigma) \end{aligned}$$

Note: If we stack our $p_{i,t}$ values, we wind up with

$$P_t = G\lambda_t + \varepsilon_t$$

where $G \equiv \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_I \end{bmatrix}$, $\varepsilon_t \equiv \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_I \end{bmatrix}$

4.4 Nonlinear Algorithms

Random Forest:

Random forest is a supervised machine learning algorithm, using ensemble learning methods i.e. combining the output of multiple decision trees to reach a single result. For classification problems, the random forest algorithm takes majority vote and the output is the class selected by most trees. For Regression problems, the output is the average of all predictions. Random Forest applies the technique of Bootstrap Aggregation, also known as Bagging, to tree learners. For instance, if a training set:

$X = x_1, x_2, \dots x_{n'}$ has responses $Y = y_1, y_2, \dots y_{n'}$ then the bagging ensemble method repeatedly selects a random sample, with replacement, from the training set and fits trees to these samples, such that: Let B be the number of times a random sample is selected, For $b = 1, \dots, B$ 1. Sample, with replacement, n training examples from X, Y. Let these be called X_b, Y_b 2. Train a classification or regression tree f_b on X_b, Y_b Once the model has been trained using the samples, predictions for the test data or the unseen x', in case of regression problem, will be made by averaging the predicitions from all individual regression trees on x':, $\hat{f} = 1/B \sum_{b=1}^{B} \hat{f}(x')$

For classification problems, the majority vote will be used. The biggest advantage of using bootstrapping procedure in Random Forest is that this method decreases the variance of the model, without increasing the bias and improves the performance of the model. While the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees. However, bootstrap sampling is a way of de-correlating the trees by using different training sets. Apart from using bootstrap aggregation, random forest algorithm uses another type of bagging scheme, which is a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the feautures. In an ordinary bootstrap sample, if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated. However, random selection of inputs tackles this problem and increases the accuracy of the model.

Gradient Boosted Decision Trees:

Gradient Boosting is a machine learning technique, which uses the Boosting method to combine weak learners into a single strong learner in an iterative fashion. In other words, unlike the Random Forest algorithm, the decision trees in gradient boosting are built additively i.e one after another. In the case of many supervised learning problems, there is an output variable y and a vector of input variables x, related to one another through a probabilistic distribution. In order to find some function $\hat{F}(x)$ that best approximates the output variable from the values of input variables, the loss function $L(y, F(x))$ has to be minimised, such that: $\hat{F}(x) = arg_F min \mathbb{E}_{x,y}[L(y, F(x))]$ To achieve this, the gradient boosting method assumes a real-valued y and seeks an approximation $\hat{F}(x)$ in the form of a weighted sum of functions $h_i(x)$ from some class H, called base or weak learners such that: $\hat{F}(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + const$ The alogorithm starts with a constant function $F_0(x)$ and incrementally expands it such that: $F_0(x) = arg_\gamma min \sum_{i=1}^{n} L(y_i, \gamma)$ $F_m(x) = F_{m-1}(x) + arg_{h_m \in H} min[\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h_m(x_i))]$ where $h_m \in H$ By iterating on $F_m(x)$, the algorithm finds the local minimum of the loss function.

K-Nearest Neighbors:

K Nearest Neighbour is a simple univariate algorithm that stores all the available cases and classifies the new data or case based on a similarity measure. It is mostly used to classifies a data point based on how its neighbours are classified. Similarity is defined according to a distance metric between two data points. Distance could be Euclidean, Manhattan, Minkowski, or Hamming distance. We use the Euclidean approach, which is the most widely used distance measure to calculate the distance between test samples and trained data values. - Euclidean Distance

$$d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \cdots + |x_{ip} - x_{jp}|^2}$$

- Minkowski Distance

$$d(i,j) = (|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \cdots + |x_{ip} - x_{jp}|^q)^{1/q}$$

- Manhattan Distance

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ip} - x_{jp}|$$

From a given k, one can choose the closest neighbors of that given data. For a data record t to be classified, its k nearest neighbours are retrieved, and this forms a neighbourhood of t. One needs to choose an appropriate value for k, and the success of classification is very much dependent on this value. There are no pre-defined statistical methods to find the most favorable value of K. The only option is to choose different k and select the one with the lowest error. However, we won't proceed like this here to avoid overfitting.

Support Vector Regression:

In ε-SV regression (Vapnik 1995), the goal is to find a function f (x) that has at most ε deviation from the actually obtained targets yi for all the training data, and at the same time is as flat as possible. In other words, we do not care about errors as long as they are less than ε, but will not accept any deviation larger than this. This may be important if you want to be sure not to lose more than ε money when dealing with exchange rates, for instance.

For example, the case of linear functions f, taking the form:

$f(x) = < w, x > + b$ with $w \epsilon X, b \epsilon \mathbb{R}$

where $<.,.>$ denotes the dot product in X. Flatness in the case of the equation above means that one seeks a small w. One way to ensure this is to minimize the norm, i.e. $||w||2 = < w, w >$.

Therefore, the optimization problem is, in a simpler, more intuive version:

minimize:

$1/2 * ||w||^2$

subject to:

$y_i - < w, x_i > - b \leq \epsilon$

$< w, x_i > + b - y_i \leq \epsilon$

Neural Networks

Neural networks are also commonly known as deep learning algorithms. They are computing systems inspired by the biological neural networks which are part of human brains. They were designed to discover and recognize pattern not only in numerical data but also in images, sound, text or time-series. In simple terms, neurons are the building blocks of the neural network algorithms. They work just like a function taking in input and producing output. Firstly, each input is multiplied by a weight:

$x_1 -> x_1 * w_1$

$x_2 -> x_2 * w_2$

Then all the weighted inputs are added together with a bias b:

$(x_1 * w_1) + (x_2 * w_2) + b$

Finally, the sum is passed through an activation function:

$y = f(x_1 * w_1 + x_2 * w_2 + b)$

A commonly known activation function is the sigmoid function. Sigmoid function only outputs numbers in the range (0,1). Neural Network are nothing, but the bunch of neurons connected. There are different types of neural networks such as Feedforward, Long Short-Term Memory (LSTM) and Recurrent Neural Network. The organization of neurons is what makes each of these Neural Networks different. Goal for each of these algorithms is to recognize patterns in the data, image or any object. Neurons are generally organized into multiple layers. These layers can be categorized into the input, output and the hidden layers. The layers used for the purpose of this paper are Dense and LSTM.

Neural Network - Dense Layers:

These are layers that are deeply connected with their preceding layers which means that the neurons of the layers are connected to every neuron of its preceding layer(cite). This kind of layering is often used in the Artificial Neural Networks. The way dense layers work is that the model receives output from every neuron of its preceding layer, where the neurons of the dense layer perform matrix-vector multiplication. Matrix vectormultiplication is a procedure where the row vector of the output from the preceding layers is equal to the column vector of the dense layer. The general rule of matrix-vector multiplication is that the row vector must have as many columns as possible like the column vector. The general formula for a matrix-vector product is

where $Ax = \begin{bmatrix} a_{11}\ a_{12}\ \dots\ a_{1n} \\ a_{21}\ a_{22}\ \dots\ a_{2n} \\ \vdots \\ a_{m1}\ a_{m2}\ \dots\ a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

Where A is a (M XN) matrix and X is a 1 X N) matrix. Values under this matrix are the trained parameters of the preceding layers also can be updated by the backpropagation. Backpropagation is a common algorithm used for the training of the feedforward neural networks. It calculates the

gradient of the loss function with respect to the weights of the network for single input or output. From the above intuition, we can say that the output coming from the dense layer will be an N-dimensional vector. We can see that it is reducing the dimension of the vectors. So basically, a dense layer is used for changing the dimension of the vectors by using every neuron

5. Empirical Results

In this section, we apply the methods described before to the US data we discussed before. In summary, whenever applicable, we will use monthly US economic data to predict the US quarterly GDP growth.

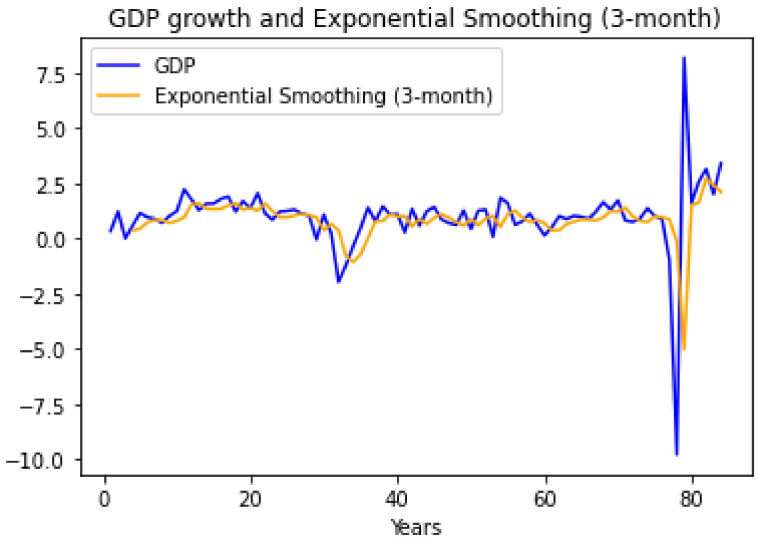
For each and every model, we follow similar procedures. We estimate them in-sample with a sample from 2000 to 2021. Then, whenever possible (e.g. there's no out-of-sample for trailing average), we estimate them out-of-sample with the coefficients/training of the model from 2000 to 2010 and the forecast out-of-sample from 2011 to 2021. This step avoids the overfitting problem and make our model comparison more reliable. Then, in the following section, we combine all of our findings to compare them.

5.1 Trailing 3-period average (application):

Trailing 3-period average is a very naive method to forecast GDP and therefore a good starting point. This model simply takes the average of the last 3-periods to produce next period's forecasts. As expected, this measure tracks GDP with lags and it is not highly useful in predicting it.

5.2 Exponential Smoothing (application):

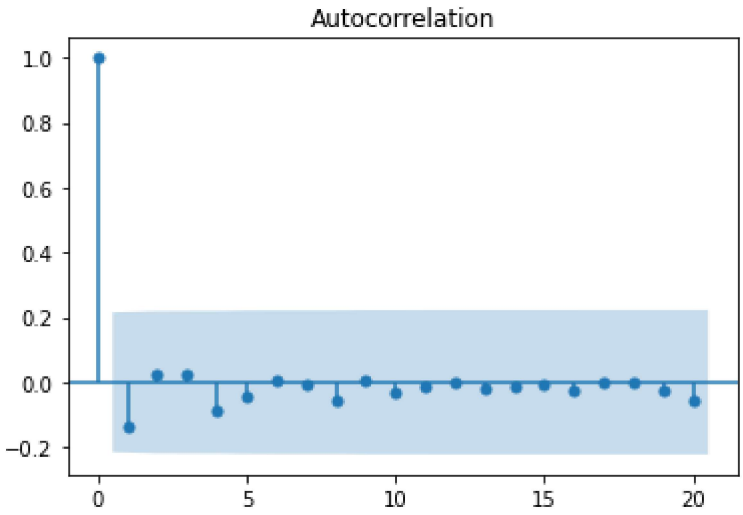
Exponential Smoothing is a time series forecasting technique for the univariate data. It is similar to Box-Jenkins ARIMA or the trailing average family of methods in a sense that the prediction is the weighted sum of past observations, but the only difference is that the model uses an exponentially decreasing weight for past observations. The problem of the trailing average persists, and in fact for our sample the RMSE is higher.



5.3 ARIMA models (application):

ARIMA models are the gold standard of time series forecasting. These models not only take into account the variable's own lags but also the lags of the error terms and the stationarity of the data. The hyparameters for ARIMA model is p,d,q where p is the number of autoregressive terms. d is the difference required to make the series stationary and q is the number of lagged forecast errors.

First, to check for stationarity of the GDP series, Partial Auto-correlation functions are used. The GDP data, when looking at its first difference (our main variable), does not show any meaningful presence of autocorrelation. Still, we use for this exercise and ARIMA (3,0,0) -- note that any MA term when combined with a similar AR term would have unitary root in this case.



It is interesting to note that when estimated through the whole sample, including the COVID-19 period, the AR(1) coefficient is close to zero. However, when estimating from 2000 to 2010, the AR(1) coefficient is around 0.6. This larger coefficient makes the estimates more volatile in the COVID-19 period, with lags.

ARMA Model Results

Dep. Variable:	GDP	**No. Observations:**	83

Model:	ARMA(3, 0)	**Log Likelihood**	-156.068
Method:	css-mle	**S.D. of innovations**	1.586
Date:	Thu, 12 May 2022	**AIC**	322.136
Time:	20:09:37	**BIC**	334.230
Sample:	03-31-2001	**HQIC**	326.995
	- 09-30-2021		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.9633	0.158	6.087	0.000	0.653	1.273
ar.L1.GDP	-0.1354	0.110	-1.235	0.217	-0.350	0.080
ar.L2.GDP	0.0086	0.112	0.077	0.939	-0.211	0.228
ar.L3.GDP	0.0253	0.111	0.228	0.820	-0.192	0.243

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-2.0722	-2.4724j	3.2260	-0.3610
AR.2	-2.0722	+2.4724j	3.2260	0.3610
AR.3	3.8040	-0.0000j	3.8040	-0.0000

ARMA Model Results

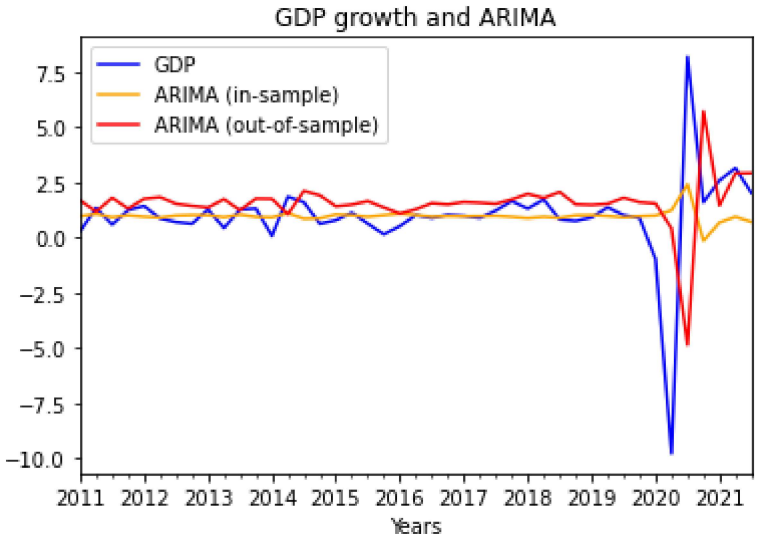
Dep. Variable:	GDP	**No. Observations:**	40
Model:	ARMA(3, 0)	**Log Likelihood**	-38.105
Method:	css-mle	**S.D. of innovations**	0.623
Date:	Thu, 12 May 2022	**AIC**	86.210
Time:	20:09:37	**BIC**	94.654
Sample:	03-31-2001	**HQIC**	89.263
	- 12-31-2010		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.9374	0.270	3.469	0.001	0.408	1.467
ar.L1.GDP	0.5970	0.158	3.767	0.000	0.286	0.908
ar.L2.GDP	0.0062	0.193	0.032	0.974	-0.371	0.384
ar.L3.GDP	0.0502	0.163	0.307	0.758	-0.270	0.370

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.4157	-0.0000j	1.4157	-0.0000
AR.2	-0.7696	-3.6716j	3.7514	-0.2829

AR.3	-0.7696	+3.6716j	3.7514	0.2829



5.4 OLS models (application):

OLS models are also very primary and traditional models. They are primarily used when the data is cross-sectional. Their use in time series is more limited because of potential violations of standard OLS assumptions. However, they still are widely used and serve as the benchmark models and can be used to compare the forecasts across models.

In this initial step, we show how the in-sample regression looks like. The adjusted r-squared is relatively high with payrolls, IP, retail sales showing the highest predictive powers to GDP. Other variables, such as Vehicle Sales and Housing starts enter with the wrong signal, which would recommend their removal. We keep them for the exercise, particularly as we add Elastic Net models that are intended to reduce problems such as these.

When we run the regression only for the sample 2000-2010, the r-squared is similar, but the importance of variables change somewhat. For example, payrolls show lower correlation in that period than in the whole sale, whereas variables such as wholesale inventories and construction spending gain importance.

OLS Regression Results

Dep. Variable:	GDP	**R-squared:**	0.861
Model:	OLS	**Adj. R-squared:**	0.840
Method:	Least Squares	**F-statistic:**	138.1
Date:	Thu, 12 May 2022	**Prob (F-statistic):**	2.43e-43
Time:	20:09:38	**Log-Likelihood:**	-74.847
No. Observations:	83	**AIC:**	173.7
Df Residuals:	71	**BIC:**	202.7
Df Model:	11		
Covariance Type:	HAC		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.7831	0.105	7.485	0.000	0.574	0.992
payroll	3.2475	0.350	9.269	0.000	2.549	3.946
job_openings	-0.0778	0.037	-2.082	0.041	-0.152	-0.003
ip	0.4823	0.190	2.538	0.013	0.103	0.861
vehicle_sales	-0.0002	0.051	-0.004	0.997	-0.102	0.102
quits	0.0945	0.038	2.502	0.015	0.019	0.170
retail	0.1745	0.090	1.948	0.055	-0.004	0.353
wholesale_inventories	-0.0400	0.180	-0.222	0.825	-0.399	0.319
consumer_sentiment	0.0122	0.025	0.497	0.620	-0.037	0.061
construction_spending	0.0055	0.081	0.068	0.946	-0.157	0.168
new_orders	0.0019	0.041	0.046	0.964	-0.080	0.083
housing_starts	-0.0067	0.020	-0.330	0.742	-0.047	0.034

Omnibus:	45.662	**Durbin-Watson:**	1.745
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	207.746
Skew:	1.613	**Prob(JB):**	7.73e-46
Kurtosis:	10.047	**Cond. No.**	20.2

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC) using 1 lags and without small sample correction

Note that the in-sample regression does a very good job in tracking the GDP behavior. However, when estimated for a train sample and forecasted into the future, its predictive power falls significantly, particularly during the COVID-19 period.

OLS Regression Results

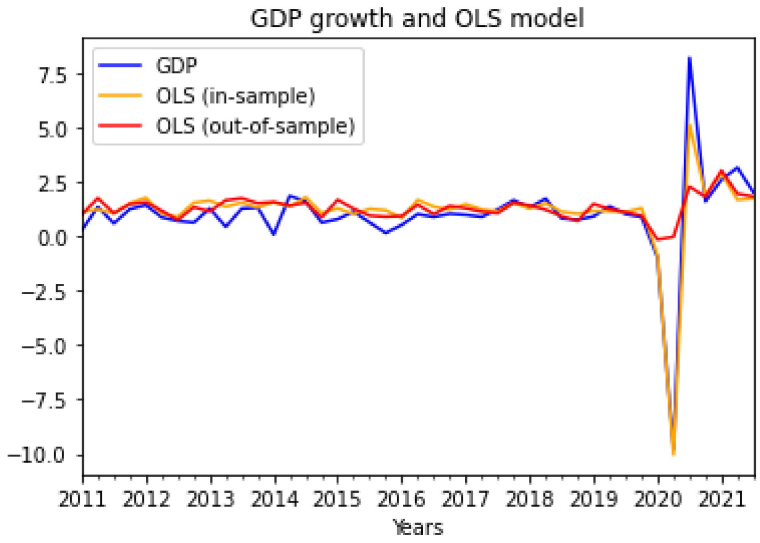
Dep. Variable:	GDP	**R-squared:**	0.853
Model:	OLS	**Adj. R-squared:**	0.796
Method:	Least Squares	**F-statistic:**	42.65
Date:	Thu, 12 May 2022	**Prob (F-statistic):**	1.54e-14
Time:	20:09:38	**Log-Likelihood:**	-9.7532
No. Observations:	40	**AIC:**	43.51
Df Residuals:	28	**BIC:**	63.77
Df Model:	11		
Covariance Type:	HAC		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.8303	0.080	10.366	0.000	0.666	0.994
payroll	-0.0451	0.778	-0.058	0.954	-1.640	1.549
job_openings	0.0057	0.028	0.204	0.840	-0.052	0.063
ip	0.3213	0.122	2.626	0.014	0.071	0.572
vehicle_sales	-0.0471	0.049	-0.952	0.349	-0.148	0.054
quits	0.1144	0.034	3.356	0.002	0.045	0.184
retail	0.2695	0.139	1.933	0.063	-0.016	0.555
wholesale_inventories	0.2607	0.106	2.457	0.020	0.043	0.478
consumer_sentiment	0.0128	0.016	0.805	0.427	-0.020	0.045
construction_spending	0.3316	0.079	4.202	0.000	0.170	0.493
new_orders	0.0405	0.036	1.121	0.272	-0.033	0.114
housing_starts	-0.0271	0.025	-1.075	0.292	-0.079	0.025

Omnibus:	0.100	**Durbin-Watson:**	2.300
Prob(Omnibus):	0.951	**Jarque-Bera (JB):**	0.154
Skew:	-0.103	**Prob(JB):**	0.926
Kurtosis:	2.777	**Cond. No.**	75.6

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC) using 1 lags and without small sample correction

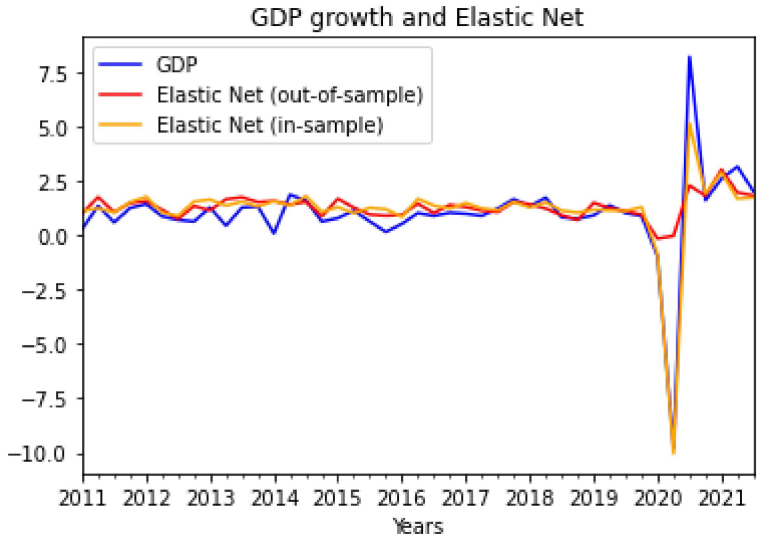
The results of this model are more intuitive. The COVID-19 shock generate significant impact on the coefficients, as one would expect. In this sense, the in-sample model performs significantly better than the out-of-sample one. This fact puts in evidence the Lucas critique problem of trusting into aggregate coefficients, when shocks can behavior.



5.5 Elastic Net models (application):

Elastic net models are penalized linear regression models. They are validated by use of L1 and L2 penalties during the training. The L1 penalty basically refers to inclusion of absolute value of coefficient as a bias. A model with just L1 is also referred to as the lasso regression. On the other hand, the L2 regularization adds the "squared magnitude" of coefficients as penalty term to the cost function. This is also referred to as ridge regression. The Elastic net models use both L1 and L2 penalties.

In this case, it will solve the problems we had before of including coefficients with the wrong signal or giving too much importance to one variable. As we learned from the OLS model, the degree of relevance of these variables can change meaningfully depending on the sample used. In any case, again, the in-sample results show a better accuracy than out-of-sample, similarly to the OLS model.



5.6 VAR models (application):

Vector Autoregressive Models (VAR) used for the multivariate time series forecasting. The way they are structured is that each variable is the linear function of its own past lags and the lags of other

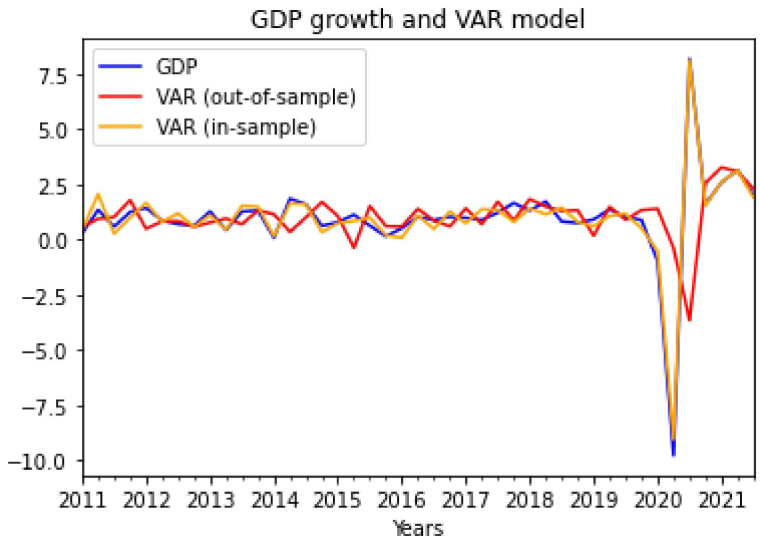
variables, a multivariate version of the AR models. VAR models are gold standards for the time series forecasting and are used widely to forecast macroeconomic variables.

The first step in this model is to try to came up with the lags to be used. There are many different option of tests, four of which we present in the table below. Three out of four of those tests suggest that five lags is the most appropriate ordering. In this sense, our model will have five lags. For the restricted sample, the tests (not shown) suggest that two lags are enough.

VAR Order Selection (* highlights the minimums)

	AIC	BIC	FPE	HQIC
0	1.780	2.143*	5.931	1.925
1	-1.327	3.386	0.2753	0.5594
2	-1.390	7.674	0.3298	2.238
3	-1.881	11.53	0.4109	3.489
4	-3.414	14.35	0.4637	3.698
5	-9.967*	12.15	0.02956*	-1.113*

One of the distinct advantages of this model is its ability to deliver impulse response functions in an easy and intuitive way. The main disadvantage is that models like these usually require a large dataset to avoid losing too many degrees of freedom in estimating the impacts. For the sake of space, we won't provide the summary table of this method.



5.4 Dynamic Factor Model (application):

Dynamic Factor Model is an advanced machine learning algorithm used for time series forecasting. Dynamic-factor models are flexible models for multivariate time series in which the observed endogenous variables are linear functions of exogenous covariates and unobserved factors, which have a vector autoregressive structure. The unobserved factors may also be a function of exogenous covariates.

The main advantage of this method is that it does not need the full updated dataset to make a prediction. For example, for a prediction of GDP in 1Q22, one needs all the covariates for 1Q22 in an OLS model. In the Dynamic Factor Model, the Kalman Filter generates a prediction for each variable for the period of forecasting. In this sense, the Dynamic Factor model generates a prediction within a prediction to be able to forecast even when the full data set is unavailable.

This first table summarizes the result of the dynamic factor with three orders of lags. All variables enter with the same signal, which suggest that all of them "work" on the same direction (e.g. payrolls and vehicle_sales move together, and so does all variables).

Dynamic Factor Results

Dep. Variable:	"payroll", and 10 more	**No. Observations:**	255
Model:	Dynamic Factor Model	**Log Likelihood**	-3449.177
	+ 1 factors in 1 blocks	**AIC**	6972.354
	+ AR(1) idiosyncratic	**BIC**	7103.381
Date:	Thu, 12 May 2022	**HQIC**	7025.058
Time:	20:09:40	**EM Iterations**	68
Sample:	0		
	- 255		
Covariance Type:	Not computed		

Observation equation:

Factor loadings:	**0**	**idiosyncratic: AR(1)**	**var.**
payroll	-0.43	-0.11	0.31
job_openings	-0.25	-0.46	0.64
ip	-0.44	0.01	0.24
vehicle_sales	-0.31	-0.29	0.67
quits	-0.24	-0.45	0.66
retail	-0.43	-0.05	0.44
wholesale_inventories	-0.08	0.60	0.64
consumer_sentiment	-0.14	-0.04	0.94
construction_spending	-0.19	0.40	0.76
new_orders	-0.26	-0.42	0.68
housing_starts	-0.28	-0.43	0.66

Transition: Factor block 0

	L1.0	**L2.0**	**L3.0**	**error variance**
0	0.21	-0.17	-0.02	3.15

Warnings:
[1] Covariance matrix not calculated.

The first principal component using the dynamic part (AR(3)) is generate as a Dynamic Factor Model (DFM) variable without taking into consideration their effects on GDP. That's where this model mainly differs from OLS: OLS builds a weighted average (coefficients) of the variables to predict GDP. The Dynamic Factor Model first aggregates the set of variables in only one (the principal factor) and only then we run a OLS with the DFM vs. the GDP to predict GDP growth.

Comparing the models using the periods of 2000 to 2022 (first table below) and 2000 to 2010 (second table below), we see that in both cases the dfm enters with a negative coefficient. Indeed, we saw before that the principal components of activity data had all negative coefficients. Now, with the regression coefficient as negative, it means that each of those variables together has a positive effect on GDP (minus times minus). This is true for both samples. Note, however, that the coefficient with the larger sample is around -1.5 whereas in the restricted sample it is around -0.9.

OLS Regression Results

Dep. Variable:	GDP	**R-squared:**	0.476
Model:	OLS	**Adj. R-squared:**	0.469
Method:	Least Squares	**F-statistic:**	73.53
Date:	Thu, 12 May 2022	**Prob (F-statistic):**	5.50e-13
Time:	20:09:40	**Log-Likelihood:**	-130.07
No. Observations:	83	**AIC:**	264.1
Df Residuals:	81	**BIC:**	269.0
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
dfm	-1.5685	0.183	-8.575	0.000	-1.932	-1.205
const	0.9935	0.129	7.707	0.000	0.737	1.250

Omnibus:	82.444	**Durbin-Watson:**	2.653
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1593.869
Skew:	-2.668	**Prob(JB):**	0.00
Kurtosis:	23.794	**Cond. No.**	1.42

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

OLS Regression Results

Dep. Variable:	GDP	**R-squared:**	0.632
Model:	OLS	**Adj. R-squared:**	0.623
Method:	Least Squares	**F-statistic:**	65.33
Date:	Thu, 12 May 2022	**Prob (F-statistic):**	8.87e-10
Time:	20:09:40	**Log-Likelihood:**	-28.157
No. Observations:	40	**AIC:**	60.31
Df Residuals:	38	**BIC:**	63.69
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
dfm	-0.9063	0.112	-8.083	0.000	-1.133	-0.679
const	1.1064	0.081	13.584	0.000	0.942	1.271

Omnibus:	0.029	**Durbin-Watson:**	1.807
Prob(Omnibus):	0.986	**Jarque-Bera (JB):**	0.178
Skew:	0.054	**Prob(JB):**	0.915
Kurtosis:	2.691	**Cond. No.**	1.49

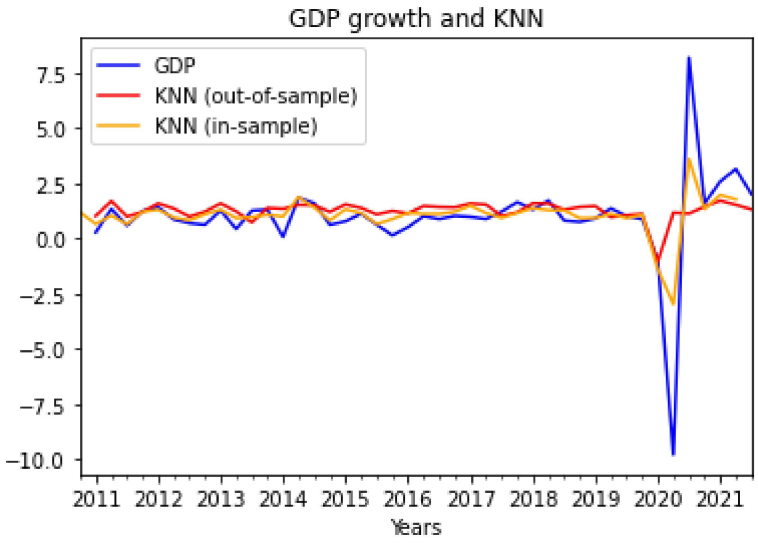
Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

The main feature here is the following. Differently from other models, the improvement between in and out-of-sample versions is not as large as in other cases. However, it is notable how this mode performs better than most out-of-sample, something which we will discuss in more details when we put all those models together.

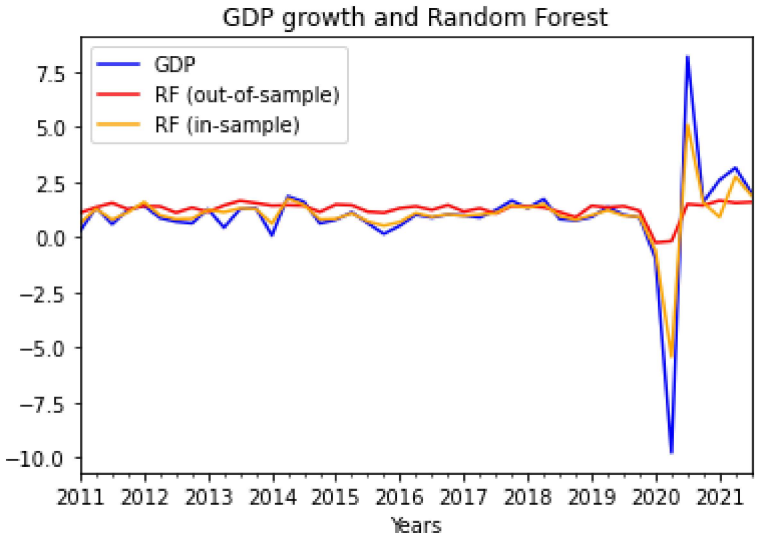
5.7 K-Nearest Neighbor (application):

K-Nearest Neighbor is a simple, east to implement, supervised machine learning algorithm and is widely used to solve regression and classification problem. This model assumes that the similar things exist in proximity and based on that make predictions.



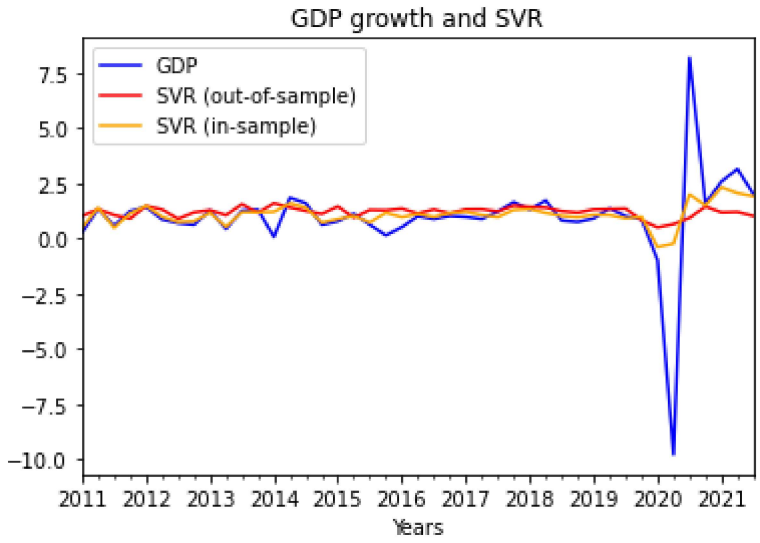
5.8 Random Forest (application):

Random Forest is another supervised machine learning algorithm that is widely used in classification and regression problems. The way it works is by building decision trees using different samples and then taking their average locally.



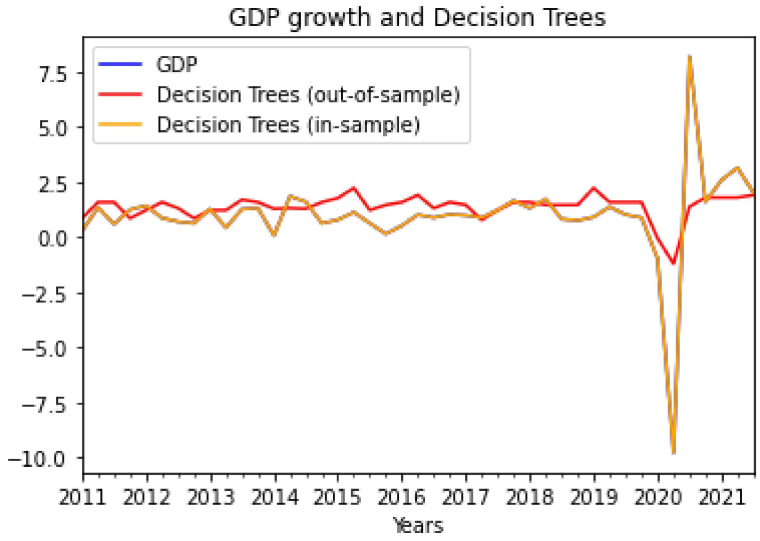
5.9 Support Vector Regression (application):

Support Vector Regression is an advanced supervised machine learning algorithm used to predict mainly the discrete values. It uses the same principles as the Support Vector Machines. It tries to find the line of best time and it does that in the hyperplane that has maximum number of points.



5.10 Gradient Boosted Decision Trees (application):

In this case, the in-sample model is meaningless as the decision trees can forecast precisely the behavior of GDP in-sample without errors. The out-of-sample performance is therefore more informative of the capacity of this model in forecasting economic information.



5.11 Neural Network (Application)

We apply the deep learning technique to firstly train the data on the data set and then we use it to forecast in-sample as well as out-of-sample data. This algirithm does a good job in out-of-sample forecasts, especialy during COVID years since it predicts the dip and rise in GDP growth, if not by the same magnitude but in the correct direction. The below graph shows the comparison:

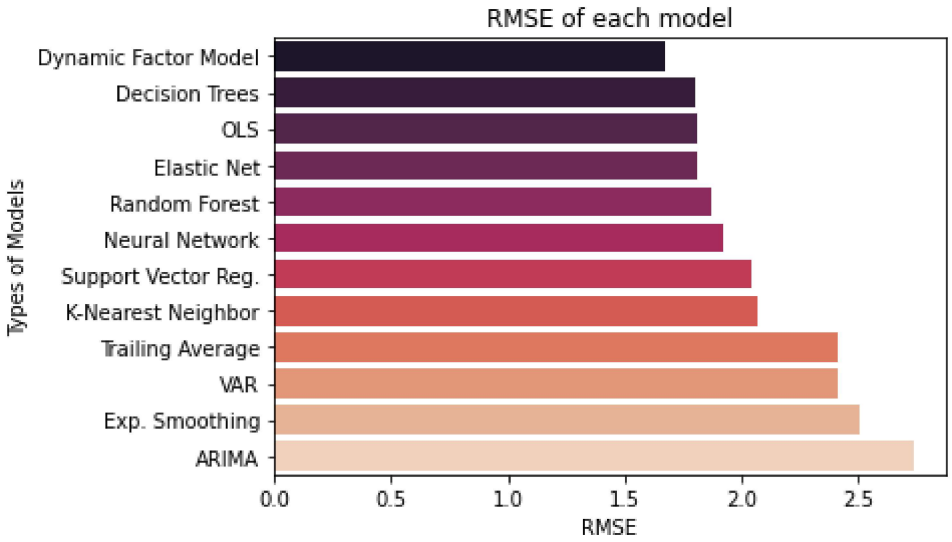
6. Comparison of the Models

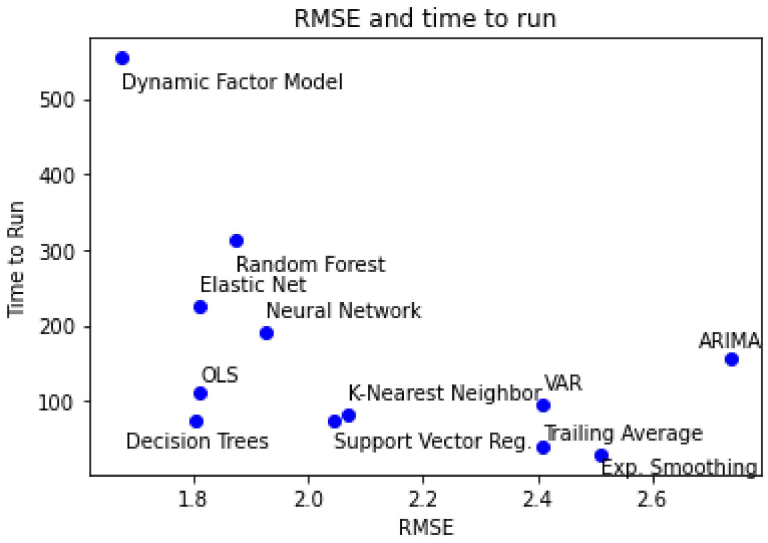
To compare how each model performs in terms of forecasting accuracy, we analyze the out of sample Root-Mean Squared Error of each model as well as the time it takes to run them. The plot below outlines RMSE of the models, with Dynamic Factor Model having the lowest RMSE -

indicating highest forecasting accuracy and the ARIMA model having the highest RMSE - indicating the worst forecasting accuracy. The second chart shows the comparison between the RMSE and the time it takes to run a model. Even though the dynamic factor model is more accurate, it tends to take more time to run. In our analysis, the DFM took five times more than the OLS. In this sense, the best 'bang-for-the-buck' models are the OLS and Random Forest, both models having low RMSE and low run time as well. This might not be a big problem for the models we built here - all models take miliseconds to run. However, in more complex models, the cost in terms of time may be higher.

If we compare machine learning models with other traditional econometric methods, the analysis suggests that simple models such as the OLS and the DFM perform as well and often even better than ML methods. In fact, the OLS performs better than many of the "machine learning" models such as Random Forest, K-Nearest Neighbor or Support Vector Regression. It performs almost the same as the Decision Trees and only underperforms meaningfully the Dynamic Factor Model. This analysis suggests that for macroeconomics, when the number of variables and their length is quite limited, simpler models may prove to be as useful than more complex ones. The Decision Tree ML model helps in forecasting whereas other ML methods perform significantly worse.

However, with regards to the forecasting accuracy of Machine Learning models, even if the RMSE of these models is not as low as that of OLS, it is still much better than that of ARIMA and VAR models, both of which are amongst the main econometric models used for multivariate analysis and forecasting, and the performance do not fall that much behind the most advanced traditional models.





7. Conclusion

This paper aims to explore whether machine learning techniques could be the future of macroeconomic forecasting and replace the traditional econometric models as the principle models. Historically, in case of turbulence and shocks, traditional methods have not performed accurately in predicting the movement of economic variables. With the development of new and improved machine learning algorithms and their proven accuracy in predicting market and financial data, it must be analysed whether these perform accurately in case of highly dynamic and low-frequency economic data.

This paper compared multiple time series models with machine learning models to forecast the US GDP. Using RMSE as a metric for forecasting accuracy, our results show that traditional advanced econometric models tend to outperform ML methods, but the latter also frequently show relatively low root mean squared errors and do not underperform by a large margin. This is an important takeaway, especially since we are considering COVID-19 years in our forecasting period. Decision trees, a very popular ML technique, has the same RMSE as ordinary least square (OLS) model, a benchmark estimation method used widely in econometric analysis. Only the Dynamic Factor Model, a mainstream multivariate timeseries model outperforms all other models in the analysis. ML techniques such as KNN, Neural Network and Random Forest, all are more accurate than the popular time-series models VAR and ARIMA.

In light of these findings, ML techniques have some potential in macroeconomic forecasting, especially as more data becomes available. However, based on these results, we are not able to hypothesise whether these techniques shall replace the mainstream econometric methods that have been used since decades. Firstly, no ML technique does better than the benchmark model of OLS. Secondly, ML models may perform better than traditional econometric models of VAR and ARIMA in terms of forecasting accuracy. However, these models do not have high interpretability which makes them less useful for policy-makers who use forecasts to develop policy reforms and strategies. It is important to note that scope of this study is just limited to the forecasting of the US GDP, a very dynamic and complex indicator, and further investigation to other macroeconomic variables is required.

8. References

Ban, T., Zhang, R., Pang, S., Sarrafzadeh, A., and Inoue, D. (2013). "Referential Knn Regression for Financial Time Series Forecasting," in International Conference on Neural Information Processing (Springer), 601–608. doi:10.1007/978-3-642-42054-2_75

Biau, O. and D'Elia, A., (2011), "Euro area GDP forecasting using large survey datasets".

Cook, Thomas R., and Aaron Smalter Hall. "Macroeconomic Indicator Forecasting with Deep Neural Networks." Federal Reserve Bank of Kansas City, Research Working Paper 17-11, September. Available at External Link https://doi.org/10.18651/RWP2017-11

Coulombe, Leroux, Stevanovic and Surprenant, "How is Machine Learning Useful for Macroeconomic Forecasting", 2019. https://economics.sas.upenn.edu/system/files/2019-03/GCLSS_MC_MacroFcst.pdf

Diebold, F. X., and Nason, J. A. (1990). Nonparametric Exchange Rate Prediction? J. Int. Econ. 28, 315–332. doi:10.1016/0022-1996(90)90006-8

FRED, federal reserve economic data. St. Louis, MO: Federal Reserve Bank of St. Louis, 1997. Software, E-Resource. https://lccn.loc.gov/98802805.

Gonzalez, "Neural Networks for Macroeconomic Forecasting: A Complementary Approach to Linear Regression Models" 2000-20007. https://publications.gc.ca/Collection/F21-8-2000-7E.pdf

Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K. (2003). KNN Model-Based Approach in Classification. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds) On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. OTM 2003. Lecture Notes in Computer Science, vol 2888. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-39964-3_62

Lee, T. H., White, H., and Granger, C. W. J. (1993). "Testing for neglected nonlinearity in time series models. A comparison of neural network methods and alternative tests". Journal of Econometrics, 56(3):269–290.

Maccarrone, Morelli and Spadaccini, "GDP Forecasting: Machine Learning, Linear or Autoregression?". 2021. https://www.frontiersin.org/articles/10.3389/frai.2021.757864/full#B5

Rodríguez-Vargas, A. (2020). "Forecasting Costa Rican Inflation With Machine Learning Methods". Latin Am. J. Cent. Banking. 1, 100012. doi:10.1016/j.latcb.2020.100012

Stone, C. J. (1977). "Consistent Nonparametric Regression". Ann. Stat. 5, 595–620. doi:10.1214/aos/1176343886

Swanson, N. R. and White, H. (1997). "A Model Selection Approach To Real-Time Macroeconomic Forecasting Using Linear Models And Artificial Neural Networks". The Review of Economics and Statistics, 79(4):540–550.

Tkacz and Hu, "Forecasting GDP Growth Using Artificial Neural Networks". Bank of Canada, 1999-2003. https://www.bankofcanada.ca/wp-content/uploads/2010/05/wp99-3.pdf