Erasmus University Rotterdam

Erasmus School of Economics

Master Econometrics & Management Science

Quantitative Finance - Master Thesis

Nowcasting US GDP growth using Machine Learning: a real-time application

Author:

Bob Droogh, 511639

Supervisor:

dr. Andreas Pick

Co-reader:

Nick Koning MSc.

February 15, 2022

Abstract

Economic variables, such as US GDP growth, are often released with substantial lags and revised multiple times. Nowcasting GDP growth, which is done using statistical models, mitigates this problem. This research shows that Machine Learning models can outperform popular statistical models in a real-time empirical nowcasting application. More specifically, Random Forest, eXtreme Gradient Boosting, Long Short-Term Memory neural network and Support Vector Machines obtain a lower prediction error than the Dynamic Factor Model and the AutoRegressive Integrated Moving Average model while producing point forecasts using regression. The models' drivers are discovered using feature importance, partial dependence and SHapley Additive exPlanations values to review the ML models' explainability.

Keywords: Nowcasting, Real-time, FRED-MD, Random Forest, XGBoost, LSTM neural network, Support Vector Regression, Feature importance, Partial dependence, SHAP values.

List of Abbreviations

ANN	Artificial Neural Network
ADF-test	Augmented Dickey & Fuller (1979) test
ARIMA	AutoRegressive Integrated Moving Average
BIC	Bayesian Information Criterion
DFM	Dynamic Factor Model
DM-test	Diebold & Mariano (1995) test
FRED-MD	Federal Reserve Economic Data Monthly Database
GDP	Gross Domestic Product
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
NN	Neural Network
OLS	Ordinary Least Squares
RF	Random Forest
RNN	Recurrent Neural Network
RMSFE	Root Mean Squared Forecast Error
SHAP	SHapley Additive exPlanations
SVM	Support Vector Machines
SVR	Support Vector Regression
XGBoost	eXtreme Gradient Boosting

Contents

1 Introduction **1**

2 Data **5**

3 Methodology **7**
3.1 Statistical models 7
3.1.1 AutoRegressive Integrated Moving Average (ARIMA) 7
3.1.2 Dynamic Factor Model (DFM) 7
3.2 Machine Learning (ML) models 9
3.2.1 Random Forest (RF) 9
3.2.2 XGBoost 11
3.2.3 Long Short-Term Memory neural network (LSTM) 13
3.2.4 Support Vector Regression (SVR) 16
3.3 Model estimation and hyperparameter optimization 18
3.4 Performance measures 19
3.4.1 Mean Absolute Error (MAE) 19
3.4.2 Root Mean Squared Forecast Error (RMSFE) 20
3.4.3 Diebold-Mariano test (DM-test) 20
3.5 Explainability 20
3.5.1 Feature Importance 20
3.5.2 Partial dependence 21
3.5.3 SHAP values 21

4 Results **23**
4.1 Prediction accuracy 23
4.1.1 Full estimation period 23
4.1.2 Pre, during, and post financial crisis 25
4.2 Explainability results 27
4.2.1 Feature importance 27
4.2.2 Partial dependence 30
4.2.3 SHAP values 31

5 Conclusion **33**

6 Discussion **34**

7 Acknowledgements **35**

References **36**

A Appendix **41**
A.1 FRED-MD 41
A.2 Hyperparameters 43
A.3 Feature Importance 46

List of Tables

1 Kernel formulas $K(x, y)$ (Gunn (1998)). 18
2 Out-of-sample performance 2006/Q1 - 2019/Q4 23
3 Forecasting errors 2006/Q1 - 2019/Q4 24
4 Diebold & Mariano (1995) test 25
5 Out-of-sample performance 2006/Q1 - 2014/Q1 27
6 Description of the FRED-MD 41
7 Hyperparameters RF 43
8 Hyperparameters XGBoost 43
9 Hyperparameters LSTM 44
10 Hyperparameters SVR 44
11 RF Relative Feature Importance 46
12 XGBoost Relative Feature Importance 47
13 LSTM Relative Feature Importance 48
14 SVR Relative Feature Importance 49

List of Figures

1 Actual US GDP growth 2006/Q1 - 2020/Q4. 6
2 Essence of a decision tree, using the Iris data set (Loh (2014)). 9
3 LSTM memory cell (Olah (2015)) 14
4 SVR's ϵ-insensitivity tube (Lins et al. (2010)) 17
5 SHAP values (Lundberg & Lee (2017)) 22
6 Out-of-sample predictions compared to actual US GDP growth. 26
7 Waterfall plots presenting feature importance. 28
8 Features importance through the estimation period. 29
9 Partial dependence plots. 31
10 SHAP summary plot. 32

1 Introduction

Financial institutions face the problem of real-time policy-making with incomplete statistical information. Important economic variables are released with large time lags, especially if the native frequency of the variable is low (Schumacher & Breitung (2008); Bańbura et al. (2013); Jansen et al. (2016)). Nowcasting does mitigate this problem, defined as 'forecasting the present' by Giannone et al. (2008). Nowcasting in real-time comes with various issues since it depends on other, more frequently released economic statistics. Hundreds of economic predictors possibly contain valuable information, resulting in issues as high-dimensionality, mixed frequencies and 'ragged' data sets.

Nowadays, central banks and other financial institutions nowcast macroeconomic variables via statistical models. This paper researches whether Machine Learning (ML) models can outperform these statistical models, and if so, which model does achieve the highest nowcasting accuracy in a real-time empirical setting. Two statistical models and four ML models are reviewed nowcasting US GDP growth, by using vintages that mimic the nowcasting process with incomplete statistical information. Random Forest (RF), eXtreme Gradient Boosting (XGBoost), Long Short-Term Memory (LSTM) neural network and Support Vector Regression (SVR) are benchmarked against the Dynamic Factor Model (DFM) and the AutoRegressive Integrated Moving Average (ARIMA) model.

The empirical results indicate that each ML model can outperform both statistical models. It cannot be determined which ML model obtains the highest nowcasting accuracy individually. Support Vector Regression (SVR) attains the lowest Mean Absolute Error (MAE) on average, while XGBoost averages the lowest Root Mean Squared Forecasting Error (RMSFE). Different ML models obtain the lowest MAE and RMSFE at different moments during the estimation period, but only slightly. Similarly, the results indicate that changing economic situations influence the nowcasting performance of individual models. Also, the results state that a simple, equally-weighted combination of the ML models' predictions produces the most accurate nowcasts, outperforming each individual ML model and statistical benchmark by quite a margin.

Next to the accuracy, the explainability of the models is reviewed by the use of feature importance, partial dependence and SHapley Additive exPlanations (SHAP) values. Schumacher & Breitung (2008), and Makridakis et al. (2018) touch on the problem that ML models generally are hard to interpret. ML models are often interpreted as black-box procedures, such that they are not suitable for practitioners who need to know how forecasts are established and how they can be influenced. This interpretability problem

has been outlined by James et al. (2013), Stevanovic et al. (2019) and Fan et al. (2021) as well, among others. Only a few studies try to address this issue by analyzing variable importance to increase the model's explainability. This study discovers the drivers behind the ML models' predictions and tries to motivate how the nowcast estimations for macroeconomic variables are established.

Gross Domestic Product (GDP) growth is such a macroeconomic variable of main importance for monetary policy assessment, according to Bok et al. (2018), calling it 'the most representative of all variables.' An initial US GDP growth estimate is commonly released broadly one month after the end of the particular quarter, with revisions in the subsequent months afterward (Bok et al. (2018); Babii et al. (2021)). In order to resolve this, the DFM is used frequently (Loermann & Maas (2019); Federal Reserve Bank of New York (2021)). This model, or a derived version of it, can be perceived as the workhorse model for institutions to assess the current state of the economy. The DFM is able to cope with the problems of high dimensionality, mixed frequencies and ragged edges of data sets, making it an appealing statistical model to use. The ARIMA model does not have to cope with these issues regarding the data since it only uses previous values of the dependent variable to construct predictions.

Present academic ML forecasting literature contains the problem that most published studies provide forecasts with satisfactory results, as stated by Makridakis et al. (2018), however, without comparing them with statistical benchmarks. Literature reviewing multiple ML models nowcasting macroeconomic variables is scarce, especially in real-time applications. This real-time empirical research will add to this literature by reviewing regression via four promising ML models. Hence, the question of whether ML models can outperform the statistical models in such a real-time application will be answered.

The first issue that comes with real-time nowcasting, high dimensionality, has been researched in numerous studies, and various ML techniques have been proposed. Multiple ML models have been able to use many predictor series in economic forecasting applications. Early applications were mainly neural networks (NNs), summarized in an article by Adya & Collopy (1998). They reviewed over 40 NN studies in economic forecasting and concluded that 81% of the studies find significantly better forecasts obtained by NNs relative to statistical techniques. Gradually, more studies were conducted trying to outperform statistical methods using other ML models. Kim (2003) applied Support Vector Machines (SVM) to predict the stock price index, proposing it as a promising alternative to neural networks and case-based reasoning. Binner et al. (2005) compared the predictive performance of NNs to ARIMA and VAR models, predicting the inflation rate of the Euro. The NN outperformed both statistical models. Chakraborty & Joseph (2017) presented

an extensive comparison of popular ML methods, such as artificial NNs, tree-based models and SVM, among others. The ML models were reviewed in three case studies relevant to central banks and regulators. They concluded that ML methods generally outperform traditional modeling approaches in (short-term) prediction applications. However, questions regarding causal inference were left open. Bolhuis & Rayner (2020) also use multiple ML methods to outperform statistical methods forecasting Turkish GDP growth, finding similar results.

Based on this previous research, it can be concluded that ML methods can outperform statistical methods in (short-term) economic forecasting applications. However, much fewer studies also incorporate the problem of mixed frequencies in the used data sets. Valuable predictor series are published quarterly, monthly or even at a higher frequency, while the dependent variable is often released quarterly. Proposed solutions, according to Schumacher (2016), are often based on bridge equations or via mixed data sampling (MIDAS). Bridge equations are mainly based on aggregation, averaging, for example. MIDAS, as explained by Ghysels et al. (2004), uses statistical phenomena as autocorrelation and regression.

Hence, statistical methods are helpful to solve the problem of mixed frequencies and can even be used in combination with ML methods. Babii et al. (2021) use regularization in combination with MIDAS in a real-time GDP nowcasting application to solve this issue. Hopp (2021) finds that the Long Short-Term Memory (LSTM) model, a model within the class of recurrent neural networks (RNNs), is particularly well-suited in economic time series. Notably, the model can function well in the case of mixed frequency data due to its architecture, at the cost of interpretability, a common problem for neural networks. Generally, other ML methods are combined with bridge equations or MIDAS to fix the data's mixed frequency problem.

The third issue that comes into play in nowcasting assessments with real-time data is the non-synchronously publication of the information set, resulting in a 'ragged' data set. Richardson et al. (2021) have published a nowcasting assessment using real-time vintages recently. In the paper, a variety of ML models are compared against the DFM of Stock & Watson (2002). They found that ML models, especially if their outputs are combined, can beat the DFM quite well in GDP growth nowcasting accuracy. Unfortunately, the results are briefly explained only, leaving the main finding that ML methods perform relatively well unexplained. They solve ragged vintages by using the estimates of New Zealand's national bank and not necessarily by statistical methods. Hopp (2021) solves the problem by using ARIMA models and the Kalman filter. Similarly, Loermann & Maas (2019) uses univariate ARIMA models to forecast the missing values to fill the ragged edges.

The remainder of this paper will be structured as follows. Section 2 presents the data used for the creation of vintages, implying a real-time empirical application. Section 3 contains the methodology, explaining the statistical and ML methods, hyperparameter optimization, performance metrics and methods for model interpretability computation in detail. Section 4 discusses the results and reviews the differences between the models. Finally, Section 5 concludes the paper, and Section 6 discusses possible flaws and initiates directions for further research.

2 Data

The vintages are created via the Federal Reserve Economic Data Monthly Database (FRED-MD) by Mccracken & Ng (2015). This database, presented in Table 6 of the Appendix, provides 128 series of critical economic variables regarding the US in a monthly frequency. FRED-MD is created to establish empirical macroeconomic research using big data, relieving researchers from composing and updating data sets themselves. The data sets are updated in real-time by the FRED database. The use of the FRED-MD increases the replicability of this research, eases the search for an extensive and complete data set, and provides the opportunity to compare obtained results with other research.

The vintages consist of $N = 123$ monthly macroeconomic series, mostly starting from 1960-Q1 until 2020-Q4. Some higher-frequency series have been aggregated up to a monthly frequency by averaging. The series have been classified into eight groups: (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 and (8) Stock market. FRED-MD previously contained 135 series. However, the US Institute of Supply Management (ISM) asked to remove seven series in 2016. Further, five series have been removed due to too many missing data points. Releases, release dates, and release sources are added manually to recreate real-time vintages. The release dates refer to January 2021 to provide intuition about the non-synchronicity of the releases.

The data has been imported as raw data, and its stationarity is obtained by using the transformation codes that are provided by McCracken (2021). The Augmented Dickey-Fuller (ADF) test by Dickey & Fuller (1979) has been conducted to ensure the series' stationarity. Further, outliers have been detected in the data set and are removed. The min-max [0,1] standardization method has been used to standardize the data. Vintages are created by using the release dates referencing January 2021, in Table 6, and the assumption has been made that each month before and after follows a similar publication schedule. This assumption has been adopted to create real-time vintages with ragged edges. Transformation, outlier removal and standardization have been conducted for each vintage individually. Bouwman & Jacobs (2011) emphasize the need for the actual forecast performance, considering the noisy characteristics of preliminary data and the effect of revisions. To obtain the actual forecasting performance, three vintages per quarter are created. For example, if the researcher wants to nowcast the US GDP growth for Q1, vintages are created with the available information on the first of February, March and April. Hence, three forecasts per GDP growth data point are obtained.

To tackle the problem of ragged vintages, standard imputation methods are used. Due to publication lags, the vintages contain series that are not released yet, resulting in missing values at the end of the series. The missing values due to publication lags or outlier removal can be dissolved by imputing forecasts. AutoRegressive Integrated Moving Average (ARIMA) models are used for imputation. Bouwman & Jacobs (2011) cover this subject in their article and find that multivariate models struggle to outperform univariate models, like ARIMA models. The ragged edge problem will be resolved using ARIMA models in combination with Kalman (1960)'s filter. The parameters of the ARIMA models will be picked based on the Bayesian information criterion (BIC) value.

Below, in Figure 1, the actual US GDP quarter-to-quarter growth data has been plotted. The GDP growth values assumed as actual values are obtained from the FRED's US GDP estimates. It is on a quarterly scale since a GDP point prediction is presented every quarter. It shows the aberrant observations around the year 2020, most likely caused by the impact of the COVID-19 pandemic. Because of its rare cause, data from 2020/Q1 and after will be removed, as indicated by the red line of Figure 1.

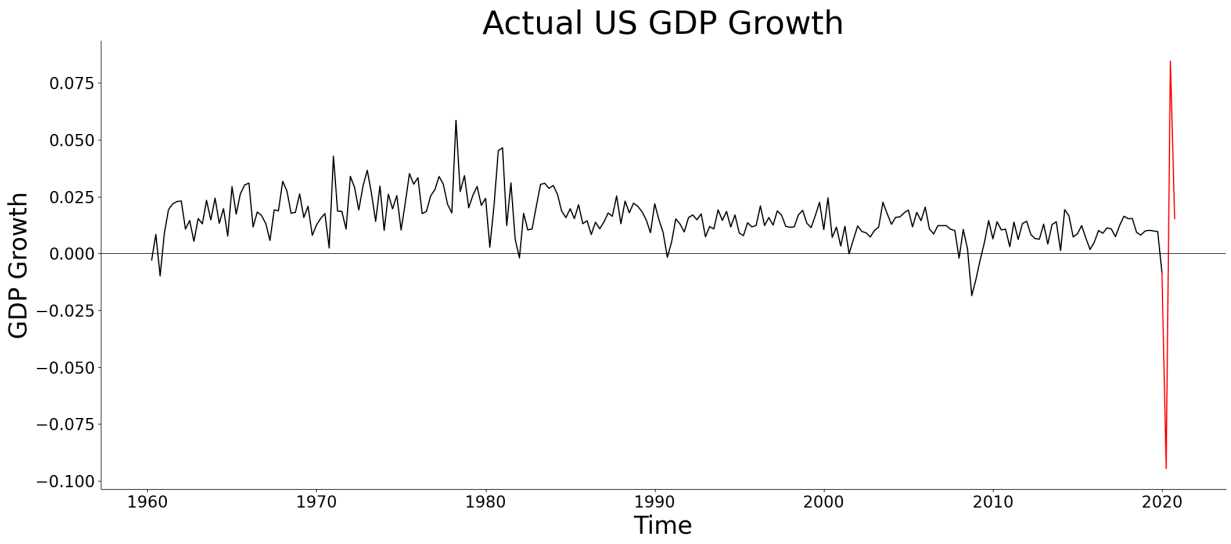


Figure 1: Actual US GDP growth 2006/Q1 - 2020/Q4.

3 Methodology

The models used to nowcast US GDP growth will be discussed in this section. The ARIMA model and the DFM of Giannone et al. (2008) will be presented to serve as statistical benchmarks. As ML models, regression via RF, XGBoost, LSTM and SVR will be reviewed. The estimation of the ML models' (hyper)parameters will be explained, and the metrics to quantify the models' performance will be introduced. Finally, the methods to increase the ML models' interpretability will be proposed.

3.1 Statistical models

The statistical models serve as a benchmark to determine whether the proposed ML models can outperform the statistical ones.

3.1.1 AutoRegressive Integrated Moving Average (ARIMA)

A univariate AutoRegressive Integrated Moving Average (ARIMA) model, as presented below in Equation 1, will be used as a benchmark. Autoregressive models can always be fitted to a time series and will usually provide a decent baseline for short-term prediction (Elliott et al. (2006)). The model is relatively straightforward and could function as a serious competitor because it only uses previous US GDP growth data. Hence, it does not have to deal with the issues of high dimensionality, mixed frequencies or a ragged data set. The model is relatively easy to interpret, as presented below:

$$y_t = \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} + \varepsilon_t. \tag{1}$$

ϕ represent 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, as explained in Section 2, no constant is needed since the data has been standardized.

3.1.2 Dynamic Factor Model (DFM)

As pointed out already in Section 1, Dynamic Factor Models (DFMs) are particularly suitable for nowcasting macroeconomic variables in real-time. The particular DFM that will be used is a popular one by Giannone et al. (2008), the first formal, consistent statistical framework combining big data and filtering (Bok et al. (2018)). This model consists of a two-step estimation method. The first step considers parameter estimation, using an Ordinary Least Squares (OLS) regression on principal components. The second step produces the expected values of the common factors. This method is suitable for real-time nowcasting, in particular, according to Bok et al. (2018), mainly because these models

cast in a state-space form, and inference can be performed well using Kalman filtering techniques. Hence, these techniques can provide a framework to handle the problems of mixed frequencies and non-synchronicity because the Kalman filter replaces missing monthly indicator observations with optimal predictions. Also, the DFM provides a parsimonious framework since it summarizes the information of a large amount of series into only a few latent common factors. Below, the two-step method of Giannone et al. (2008) is summarized. A more mathematically detailed approach can be found in Doz et al. (2011).

The first step uses principal component analysis (PCA) on the regressors $\boldsymbol{X}$ $(T \times N)$. The r obtained principal components are used as approximations for the factors $\hat{\boldsymbol{F}}$ $(T \times r)$. The following equation gives a matrix representation of this diffusion:

$$\boldsymbol{X} = \hat{\boldsymbol{F}}\boldsymbol{\Lambda}' + \boldsymbol{\zeta}, \qquad \boldsymbol{\zeta} \sim N(0, \Sigma_\zeta), \tag{2}$$

in which the $(T \times r)$ matrix $\boldsymbol{\Lambda}$ contains the factor loadings. The factor loadings are obtained via an OLS regression of $\boldsymbol{X}$ on the principal components. $\boldsymbol{\zeta}$ $(T \times N)$ contains the idiosyncratic disturbances specifically for each variable, which are modeled as Gaussian autoregressive processes. These disturbances arise from measurement error. The factors follow a p-lag vector autoregressive (VAR(p)) process, presented in the following equation:

$$\boldsymbol{\Psi}(L)\boldsymbol{F}_t = \boldsymbol{B}u_t, \qquad u_t \sim N(0, I_q), \tag{3}$$

where q denotes the number of shocks to the factors. $\boldsymbol{B}$ $(r \times q)$ contains the dynamics of u_t, which is a q-dimensional white noise process. $\boldsymbol{\Psi}(L)$ $(p \times r)$ contains the lag polynomials of the factors corresponding the VAR(p) process. The assumption has been adopted that ζ_t and u_t are uncorrelated cross-sectionally. In-sample data has been used to define the number of factors r, the number of shocks q and the p of the VAR process. The values that minimizes the BIC value have been selected, resulting in $r = 6$, $q = 6$ and $p = 1$ for most vintages.

Equations 2 and 3 form the state-space representation necessary for the Kalman filter. The first step is conducted on a balanced panel instead of the ragged data set. In the second step, the Kalman smoother has been run using the ragged dataset, where the Kalman filter forecasts using the common factors to produce observations that were missing. Finally, an OLS regression of the US GDP growth on the estimated factors is performed. The prediction can be computed as:

$$\hat{y} = \alpha + \beta'\hat{F}, \tag{4}$$

where $\hat{y}$ is the prediction, α a constant and β the coefficients following from the OLS regression.

3.2 Machine Learning (ML) models

The ML models proposed to outperform the statistical models will be discussed in detail in this subsection. The models of interest are the Random Forest (RF), eXtreme Gradient Boosting (XGBoost), Long Short-Term Memory neural network (LSTM) and Support Vector Regression (SVR).

3.2.1 Random Forest (RF)

Random Forest (RF) is a type of a tree ensemble model proposed by Breiman (2001). The idea of RF is based on creating numerous uncorrelated smaller decision trees based on random sets of regressors and averaging their outputs. Decision trees, as explained by Gordon et al. (1984) in their book, and presented in Figure 2, divide the regressor space into mutually exclusive regions by minimizing an objective function, the residual sum of squares (RSS). According to Breiman et al. (2017), decision tree models can capture the nonlinear relationship between independent variables and the dependent variable, which makes them appealing in this application.

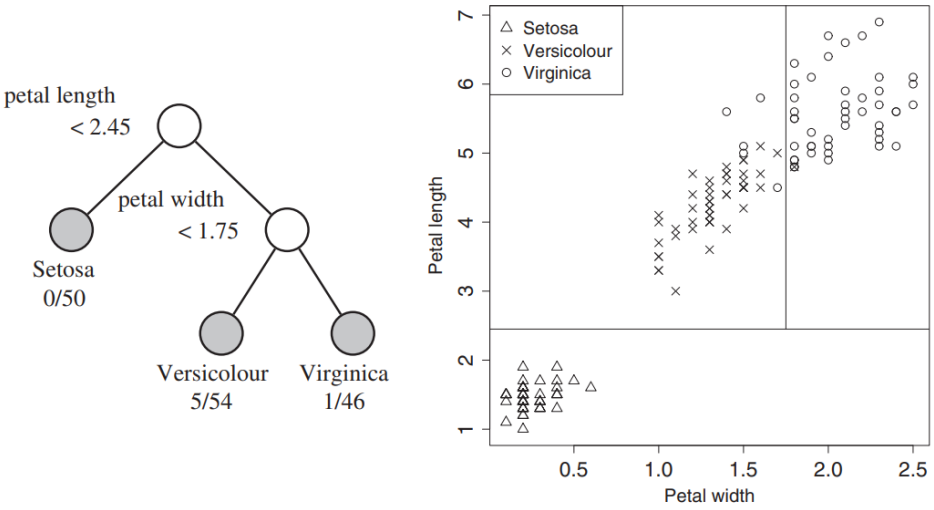


Figure 2: Essence of a decision tree, using the Iris data set (Loh (2014)).

Decision trees are easy to interpret, but they make poor predictions compared to ensemble tree models. In general, decision trees are computationally expensive to train, are often overfit, and can get stuck in local optima. To solve these issues, RF can be used. The idea of RF is based on bagging numerous smaller decision trees. Bagging can be explained as bootstrap aggregation, aggregating random sampling with replacement. This implies that the model builds many trees independently and averages their predictions. The essential idea of bagging, according to Hastie et al. (2008), is to average noisy, unbiased models to reduce variation heavily. The problem of overfitting is diminished because of the implied

additional randomness due to selecting random samples of independent variables for each split. Also, it solves the problem of getting stuck in local minima since RF does not rely much on possible dominant regressors.

The generalization error for RF, defined as the out-of-sample prediction accuracy by the model, converges to a limit as the number of trees becomes large, depending on the strength of the individual trees and the correlation between individual trees (Breiman (2001)). By using numerous trees, it follows from the Strong Law of Large Numbers that RF converges to a limiting value of generalization error. Hence, adding more trees cannot lead to overfitting. Also, Breiman (2001) finds that the lowest upper bound for the models' generalization error is obtained if the trees' dependence comes close to zero. Using trees that are as independent as possible lowers the models' prediction error.

A tree structure will be created by dividing the variable space $\mathcal{F}$ by selecting the most important variable j and its optimal split value s considering the objective function and splits the space accordingly. The most important variable j will be selected from the subset of available variables based on the given objective. This process repeats until a stopping criterium has been reached, and the variable space has been divided as optimal as possible. This process is mathematically expressed in the following equations:

$$R_1(j,s) = \{X|X_j \leq s\} \quad \text{and} \quad R_2(j,s) = \{X|X_j > s\}, \quad R_1, R_2 \in \mathcal{F}. \tag{5}$$

$$\min_{j,s} \Big[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \Big]. \tag{6}$$

Equation 5 represents the pair of regions R_1 and R_2, obtained by splitting the variable j at value s. X will then be assigned to R_1 or R_2, depending on its value. The variable j and value s will be determined based on Equation 6. Note that c is a constant, optimized by $\hat{c} = \text{ave}(y_i|x_i \in R_m)$, which is just the average of y_i in region R_m.

In the case of tree regression, the value proposed as the final value is the average value of the leaf node the data point gets assigned to. This leaf node is reached after the data point has run through the entire tree. Each regression tree obtains an estimation, and RF's final prediction will be an equally-weighted prediction of these estimations, as presented below:

$$\hat{y}_i = \phi(x_i) = \sum_{b=1}^{B} f_b(x_i), \qquad f_b \in \mathcal{B}. \tag{7}$$

The hyperparameters are tuned to make each tree as strong as possible by making use of k-fold cross-validation combined with a grid search, as explained in Section 3.3. The search ranges and corresponding optimal hyperparameters are presented in Table 7 of

the Appendix. The parameters estimated by training the RF are the variable order and thresholds used to split each node. For the execution of this algorithm, the Python function *RandomForestRegressor*[1] of the *sklearn* package based on Breiman (2001) and created by Liaw & Wiener (2018) is used, which performs well according to Hastie et al. (2008).

3.2.2 XGBoost

Extreme Gradient Boosting, better known as XGBoost, is also a decision tree ensemble method. It is originally created by Chen & Guestrin (2016) as an open-source package[2]. XGBoost is an advanced application of the gradient boosting machine of Friedman (2001). In fact, it is assumed by Chen & Guestrin (2016) as the optimized version of the gradient tree boosting model because of its optimization in both hardware and software. The technique yields superior results using fewer computing resources and takes the shortest time compared to other ensemble tree models.

Compared to RF, XGBoost uses boosting instead of bagging. The difference is that boosting uses previous trees to learn from, while bagging uses independent, parallel trees. Boosting is based on forming a committee of 'weak' learners to produce a powerful estimation. Trees will learn from the previous trees' errors until no further improvements can be made. XGBoost makes use of gradient boosting since it uses a gradient descent algorithm to minimize the loss while adding trees. Gradient descent is a method that consists of obtaining a vector of coefficients that minimize a loss function, which is obtained by tweaking parameters iteratively (Qureshi et al. (2020)). This way, XGBoost builds a forest of trees additively to ultimately provide an optimal set of regression trees.

According to Chen & Guestrin (2016), XGBoost builds trees by minimizing the loss function $\mathcal{L}(\phi)$ in Equation 8 below. Note that the procedure is summarized in this paper, for the full derivations one should consider the full paper of Chen & Guestrin (2016).

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k). \tag{8}$$

$$\Omega(f_k) = \gamma T + \frac{1}{2}||w||^2. \tag{9}$$

In this loss function, l represents a twice-differentiable loss function calculating the difference between the prediction $\hat{y}_i$ and target y_i of the dependent variable, in this regression application implemented as a squared error function. Ω is the regularization function

[1] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
[2] https://github.com/dmlc/xgboost

that penalizes the regression trees' complexity, which avoids overfitting. If one would set $\Omega = 0$, Equation 8 would be equal to a traditional tree gradient boosting function. 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 L^2-norm of leaf scores. Further, since XGBoost makes use of boosting, the algorithm incorporates the ability to learn iteratively from the previous tree's errors. The i^{th} prediction at the t^{th} iteration is computed by minimizing $\hat{\mathcal{L}}$ with respect to f_t in:

$$\hat{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t). \tag{10}$$

This tree ensemble model includes functions that cannot be optimized using traditional optimization methods in Euclidean space. Therefore, a greedy way of improving by adding f_t works as alternative. By applying the second-order Taylor expansion to $l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))$, and mathematical derivations from Chen & Guestrin (2016), Equation 10 can be written as:

$$\hat{\mathcal{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T, \tag{11}$$

in which $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ represent the first and second order derivatives of the loss function. The first term of Equation 11 indicates that the loss has been reduced after adding $f_t(x_i)$, and the second term indicates the cost γT of increasing the tree's complexity. Hence, Equation 11 shows the change in the loss function by changing the structure of tree q. It is not possible to directly obtain a smaller loss by calculating all possible structures q. This is because it is necessary to use a greedy algorithm, starting with a single leaf and iteratively adding branches to the tree. Hence, a formula that indicates the gain of adopting an additional split at node j to decrease the tree's loss is needed, presented as:

$$\text{gain } = \frac{1}{2} \left[\frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma. \tag{12}$$

I_L and I_R are sets of instances to which data point i could belong to after splitting the leaf at node j. The first two terms represent the score once the data point is assigned to the left or right branch, and the third term denotes the score before the split. Hence, Equation 12 can be used to calculate options for splits in order to improve the regression tree. After concluding that no further improvements can be made, predictions are made through a weighted average of the regressions of all trees f_b. Similar as Equation 7, this

is presented in Equation 13 below:

$$\hat{y}_i = \phi(x_i) = \sum_{b=1}^{B} f_b(x_i), \qquad f_b \in \mathcal{B}. \tag{13}$$

The algorithm that has been used for execution is the *XGBRegressor* from the Python package *xgboost*[3], based on Chen & Guestrin (2016). The algorithm's hyperparameters are tuned as explained in Section 3.3. Table 8 in the Appendix presents the optimized hyperparameters with additional explanation.

3.2.3 Long Short-Term Memory neural network (LSTM)

The Long Short-Term Memory neural network (LSTM) of Hochreiter & Schmidhuber (1997) is an extension of the recurrent neural network (RNN) architecture, which introduces a temporal component to artificial neural networks (ANNs). The LSTM architecture seems to be better suited in nowcasting applications than a more traditional feedforward architecture because of its ability to store long-range time dependency information, as used in the work of Loermann & Maas (2019), for example.

ANNs, in short, are made from layers composed of neurons. Commonly, an ANN contains an input layer, one or more hidden layers and one output layer. Neurons take a weighted sum of all previous neurons' output as input and run this through a nonlinear activation function. The first layer gets its input from the data set, and the output layer outputs an estimation through a predefined cost function. The assigned weights between neurons are set randomly at the start and are tweaked by training the algorithm using sets of data. These weight adjustments are based on calculated gradients and adjusted using an optimizer to obtain an output with a smaller error after each training round.

RNNs differ from ANNs since RNNs introduce a feedback loop. This allows for a notion of time in the sense that the input at t depends on $t-1$, which can be useful in time series applications. The outputs can be fed back into the network, incorporating the ability to train the model using gradient descent. Unfortunately, a common problem for RNNs in such an application is that the gradient can either explode or vanish. This results in losing previous information, depicted as the short-term memory problem (Bengio et al. (1994)).

The LSTM architecture has been designed to solve this problem and makes it possible to remember information over longer periods. This is achieved by Hochreiter & Schmidhuber (1997), who introduced a *memory cell* that contains three gates, as presented in Figure 3. Gates let information through, optionally, and are usually composed of a sigmoid neural

[3]https://xgboost.readthedocs.io/en/latest/python/index.html

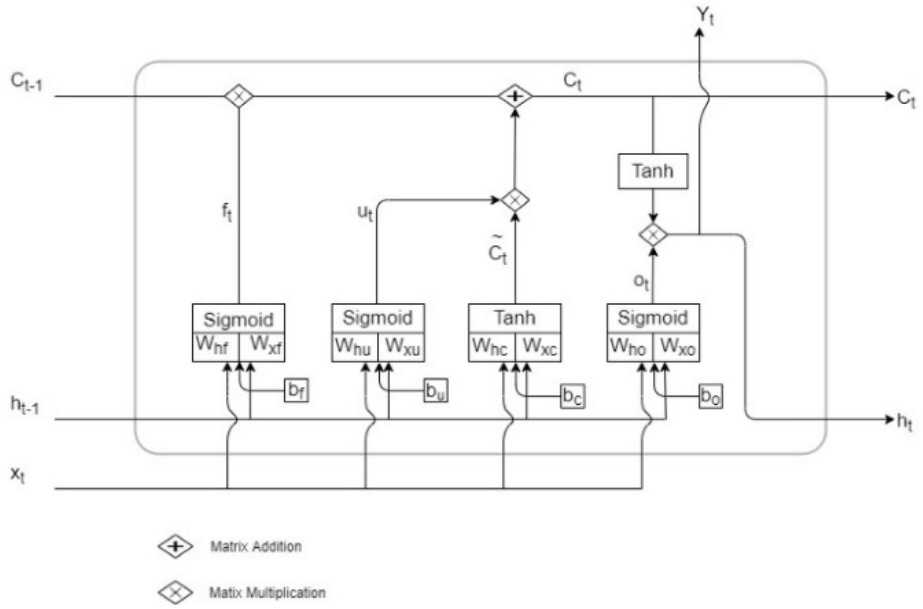


Figure 3: LSTM memory cell (Olah (2015))

net layer σ and a pointwise multiplication operation $(\cdot)$. This memory cell makes it possible to store information from the past, and the gates make this information accessible at relevant moments in time. Further, LSTM models are especially suited for applications including many regressors, taking care of the curse of dimensionality. In general, neural networks are robust to multicollinearity (Veaux et al. (1993)). The working of the LSTM model will be explained, however, a more detailed (mathematical) explanation can be found in Hochreiter & Schmidhuber (1997) and Olah (2015).

The data set has been transformed into a 3-dimensional data set $\boldsymbol{X}_t$ $(\tilde{T} \times t \times N)$, incorporating time lags t of each variable. $\tilde{T}$ is the batch size, defined as the amount of the total samples T are included per training round. The input x_t $(\tilde{T} \times N)$ is fed into the first layer of the LSTM network, which is presented in Figure 3. The above horizontal line in the cell represents the *cell state*, starting at 0, which preserves long-term information. C_{t-1} $(\tilde{T} \times H)$ puts the cell state of the previous iteration in, an important variable that makes LSTM valuable in this context. H is defined as the number of neurons in hidden layers. Further, three different gates can be distinguished, making it possible to add or remove information to the cell state by simple linear interactions. Gates are normally composed of a sigmoid neural net layer and a pointwise multiplication, but instead, the Rectified Linear (ReLu) activation function has been used. This ReLu function is a simple function and ideal for regression applications and replaces the sigmoid function assumed as default in Figure 3. An activation function decides how the weighted sum of the input is transformed into an output from the nodes in a layer of the network.

The first gate (left), the so-called the *forget gate*, determines what information gets removed from the cell state. It considers either h_{t-1} $(\tilde{T} \times H)$, the output of the previous layer, and x_t, and conducts the ReLu activation function $a(x) = \max[0, x]$. The function outputs a value between 0 and a positive value x that will be multiplied with C_{t-1}, such that if the function states 0, the cell state C_{t-1} will be set equal to zero. This can be expressed as:

$$f_t = a((x_t \cdot \boldsymbol{W}_{xf}) + (h_{t-1} \cdot \boldsymbol{W}_{hf}) + b_f), \tag{14}$$

in which $\boldsymbol{W}$ $(N \times H)$ represents a matrix of weights and b $(H \times 1)$ represents a vector with bias terms. All are adjusted by training the network. The second gate, the *input gate*, consists of two segments. The first segment runs a, in this case, ReLu function with input values:

$$u_t = a((x_t \cdot \boldsymbol{W}_{xu}) + (h_{t-1} \cdot \boldsymbol{W}_{hu}) + b_u), \tag{15}$$

with similar parameters as in Equation 14. The second segment runs a tanh function, which is a hyperbolic tangent function: $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. This tanh function outputs values between -1 and 1, and computes a vector of new values $\tilde{C}_t$ as below:

$$\tilde{C}_t = tanh((x_t \cdot \boldsymbol{W}_{xC}) + (h_{t-1} \cdot \boldsymbol{W}_{hC}) + b_C). \tag{16}$$

Then, u_t gets multiplied elementwise $(\circ)$ with $\tilde{C}_t$ and added to the cell state value C_{t-1}. The new cell state value C_t can be computed as:

$$C_t = f_t \circ C_{t-1} + u_t \circ \tilde{C}_t. \tag{17}$$

The third gate, the *output gate*, determines the output of the memory cell. The output will be based on a filtered version of C_t. The same ReLu activation function will be used to compute o_t based on h_{t-1} and x_t:

$$o_t = a((x_t \cdot \boldsymbol{W}_{xo}) + (h_{t-1} \cdot \boldsymbol{W}_{ho}) + b_o). \tag{18}$$

C_t will be run through a tanh function, such that the output value y_t can be computed as:

$$y_t = h_t = o_t \circ tanh(C_t). \tag{19}$$

The Adam optimizer with a learning rate of 0.001 has been used to optimize values for the weight matrix $\boldsymbol{W}$ and bias vector b. No further transformations of the input data have been necessary since the vintages have been standardized using min-max (0,1)-standardization. Neural networks prefer data that is scaled between 0 and 1. The results of the model's hyperparameterization can be found in Table 9 in the Appendix. The six hyperparameters that are optimized are of great importance for the quality of the model.

However, more hyperparameters could have been optimized. Due to the computational burden of optimizing the LSTM model, the choice has been made to limit the number of hyperparameters in the optimization and only optimize the most influential ones. The hyperparameters are tuned as explained in Section 3.3 and presented in Table 9. The package for the composition of the LSTM that has been used is the Python library *Keras*[4].

3.2.4 Support Vector Regression (SVR)

Support Vector Regression (SVR) by H. Drucker & Vapnik (1997) is a domain of the Support Vector Machines (SVM), originally founded by Cortes & Vapnik (1995). SVM, according to Hastie et al. (2008), produce nonlinear boundaries by constructing a linear boundary in a large, transformed version of the variable space to classify points to their respective classes in a m-dimensional space. It uses hyperplanes as decision boundaries. To find the optimal position of the hyperplane, it maximizes the margins between the hyperplane and the support vectors. Support vectors are the closest data points of both groups. The technique can be used for both classification and regression purposes.

SVR is formulated as an optimization problem, balancing model complexity and prediction error. Since SVR transforms the variable space in a higher-dimensional space, the curse of dimensionality does not harm the model. SVR's optimization does not depend on the dimensions of the input (Drucker et al. (1997)), and can capture nonlinear relations. SVR conducts nonlinear regression by doing linear regression in a higher-dimensional space. It makes SVR appealing for this kind of applications.

The mathematics necessary for SVR start with a linear regression function, which is presented as:

$$f(x) = w^T(x) + b, \tag{20}$$

where b denotes the bias and intercept term of the equation, and x the regressors' values of a data point. w can be interpreted as the regression coefficients vector. SVR tries to minimize the coefficient vector w. The loss function that is used is the ϵ-insensitive loss function proposed by Cortes & Vapnik (1995). This loss function incorporates a maximum error margin ϵ, implying a tube in which observations do not incorporate any loss. This is visualized in Figure 4. Data points that lay in the tube will not be considered.

There may be no possible line that separates all observations correctly, and therefore,

[4]https://www.tensorflow.org/api$_d$*ocs/python/tf/keras/layers/LSTM*

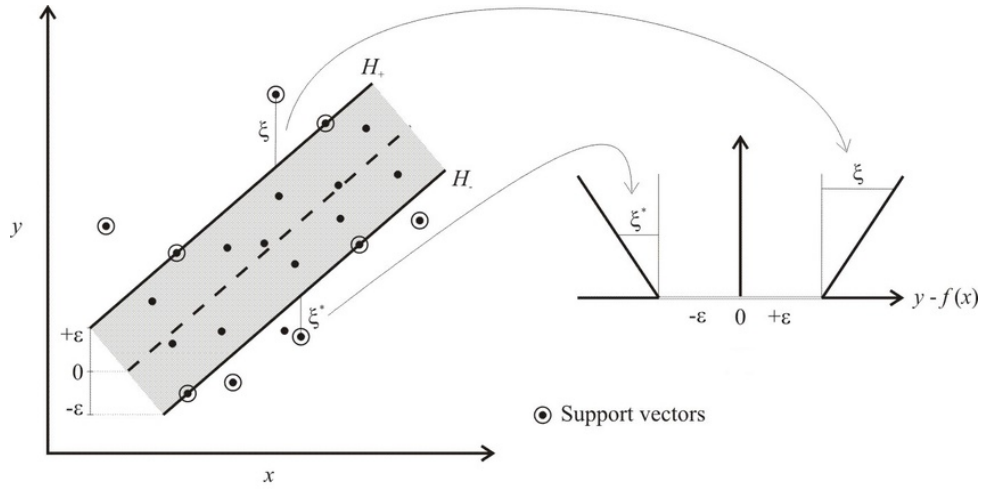


Figure 4: SVR's ϵ-insensitivity tube (Lins et al. (2010))

slack variables ξ_i and ξ_i^* are introduced. SVR tries to minimize the following equation:

$$\Phi\left(\mathbf{w}, \xi^*, \xi\right) = \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{l}\xi_i + \sum_{i=1}^{l}\xi_i^*\right), \tag{21}$$

with the constraints $|y_i - w_i x_i| \leq \epsilon + |\xi_i|$. In Equation 21, C is the misclassification tolerance parameter (Gunn (1998)), which can be interpreted as the regularization parameter. C adjusts how hard or soft the large margin classification should be. SVR tries to maximize the margin between the samples of classes, usually using only the samples that lie the closest to other classes. This could lead to overfitting since noisy samples affect the separation to be suboptimal for most data. Certain samples are allowed to be inside the margin with a softer margin classification, implied by a higher C value. By doing so, the overall fit of the model might be better than with a harder margin classification. However, a lower C value penalizes samples inside the margins less. Hence, finding the right value C leads to the optimal fitting of the hyperplane to the data and penalization of the number of samples inside the margins.

To obtain the most optimal hyperplane for separation, computations must be done in a high-dimensional space. This is hard and computationally expensive. Instead, the Kernel trick will be used. This trick does the calculations based on dot products while the data has been mapped to a higher dimension. This results in a more efficient and less computationally expensive transformation. Also, as mentioned before, this provides a way to address the curse of dimensionality, useful in this application. A Lagrangian has been used and transformed to a dual problem, as derived in Gunn (1998). With α

representing the Lagrange multipliers, the derived equation is:

$$\max_{\alpha,\alpha'} W(\alpha,\alpha^*) = \max_{\alpha,\alpha'} \left\{ \begin{array}{c} \sum_{i=1}^{l} \alpha_i^* (y_i - \epsilon) - \alpha_i (y_i + \epsilon) \\ -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(x_i, x_j) \end{array} \right\}, \tag{22}$$

with constraint $0 \leq \alpha_i \leq C \ \forall i$ and $\sum_{i=1}^{l}(\alpha_i^* - \alpha_i) = 0$. The Kernel function $K(x_i, x_j)$ replaces the dot product, taking care of the nonlinear mapping, while the constraints remain unchanged. This Kernel function allows for nonlinear regression. Several kernels exist, but the most commonly used ones that imply appealing results are the Linear, Gaussian RBF and Polynomial kernels. Hence, these kernels will be considered. The functions are as follows:

Table 1: Kernel formulas $K(x, y)$ (Gunn (1998)).

Linear	RBF	Polynomial
$K(x,y) = x^T y + c$	$K(x,y) = \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right)$	$K(x,y) = (\gamma x^T y + c)^d$

x and y represent variable vectors from some input space. c is a constant, and d is the degree of the Polynomial kernel. The kernel choice will be determined by hyperparameter tuning, as explained and presented in Table 10 in the Appendix. Finally, the regression function is given by:

$$\hat{y} = f(\mathbf{x}) = \sum_{\text{SVs}} (\bar{\alpha}_i - \bar{\alpha}_i^*) K(x_i, x), \tag{23}$$

computing the output value as estimated by SVR. The algorithm that has been used for computation is the Python function *SVR* from the *sklearn* package which is based on Platt (1999). The algorithm's hyperparameters are tuned as explained in Section 3.3. Table 10 in the Appendix presents the optimal hyperparameters with additional explanation.

3.3 Model estimation and hyperparameter optimization

The data set has been split into two sets. The first set, data until January 1 2006, is used to train and validate the models. To obtain the optimal hyperparameters, k-fold cross-validation is used in combination with a grid search. $k = 5$ is picked, a general used value for this method (Tsamardinos et al. (2014)). This implies that the first data set has been split into five sets, such that four sets are used to train the model, and one set is used to validate the model's performance. This has been done five times, each time with a different validation set, and the average performance is reported.

In the grid search, this k-fold cross-validation method has been conducted for each combination of the values in the ranges presented in Tables 7, 8, 9 and 10 of the appendix, to find the optimal hyperparameters. The ranges indicate the extensiveness of the grid

search. Due to a large number of hyperparameters per model, optimization can result in a computational burden. To solve this, cloud computing via Microsoft Azure[5] has been conducted. For the LSTM model, randomized grid searching has been implemented. Training a single LSTM neural network could take minutes, taking too many resources to test every combination possible.

The second data set has been used to test the out-of-sample nowcast performance of the models. Three vintages per GDP growth data point have been created. The models are trained on the available data of the vintage, using the hyperparameters that have been optimized. The trained models are used to estimate the US GDP growth. This prediction is compared to the actual value. Then, a new vintage is presented to the models, and they have been trained again, such that one more observation can be used. The trained models have estimated the next value, and so on. These steps are conducted for each vintage, and the performance has been measured.

3.4 Performance measures

Performance measures are incorporated to quantify the nowcasting accuracy of the models that have been proposed. The Mean Absolute Error (MAE) and Root Mean Squared Forecast Error (RMSFE) will be introduced as performance measures, two of the most commonly used measures for continuous variables. To examine if the ML models' performances are significantly different from the statistical models, the test of Diebold & Mariano (1995) will be conducted.

3.4.1 Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) quantifies the nowcasting accuracy of a model. It is defined as:

$$MAE = \frac{\sum_{t=1}^{T} |y_t - \hat{y}_t|}{T}, \tag{24}$$

where y_t represents the assumed actual value and $\hat{y}_t$ the estimated value of US GDP growth, respectively. T represents the number of forecasts. The MAE is a simple, but clear accuracy measure.

[5]https://azure.microsoft.com/nl-nl/

3.4.2 Root Mean Squared Forecast Error (RMSFE)

The Root Mean Squared Forecast Error (RMSFE) is another measure to quantify the nowcasting accuracy. It is defined as:

$$RMSFE = \sqrt{\frac{\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}{T}}, \tag{25}$$

where, again, y_t represents the actual value and $\hat{y}_t$ the estimated value of US GDP growth. T represents the number of forecasts. Compared to the MAE, the RMSFE gives more weight to forecasts that are far off due to the fact that the error is squared.

3.4.3 Diebold-Mariano test (DM-test)

To determine whether the ML models certainly outperform the statistical models, the Diebold & Mariano (1995) test (DM-test) has been conducted to investigate if the differences in the models' forecasting accuracy are significant. If the forecast errors are defined as $e_{i,t+1|t} = y_{t+1} - \hat{y}_{i,t+1|t}$, the DM-test determines if the sample mean of the loss differential $d_{t+1} = e_{i,t+1|t} - e_{j,t+1|t}$ differs from zero for models i and j. If this is the case, models i and j do have significant differences in forecasting error. The test statistic is defined as:

$$DM = \frac{\overline{d}}{\sqrt{V(\hat{d}_{t+1})/P}} \overset{a}{\sim} N(0,1), \quad \text{with} \quad V(\hat{d}_{t+1}) = \frac{1}{P-1}\sum_{t=T}^{T+P-1}(d_{t+1} - \overline{d})^2. \tag{26}$$

P presents the amount of one-step-ahead forecasts and $\overline{d}$ the sample mean of these forecasts. $V(\hat{d_{t+1}})$ is the estimate of the variance of d_{t+1}. The two-sided test assumes similar accuracy for both models, hence $H_0 : E[d_{t+1}] = 0$.

3.5 Explainability

To increase the interpretability of the models, feature importance, partial dependence and SHAP values will be computed. Practitioners are, besides nowcasting accuracy, interested in the models' interpretability during the selection of ML models.

3.5.1 Feature Importance

Feature importance describes the importance of a predictor variable on the nowcasting prediction. According to Molnar (2019), feature importance can be defined as the increase in the model's prediction error once the variable value is permuted. This type of local measurement has been implied by Breiman (2001), introducing the idea with his RF ideas accordingly. With local, the measurement of the contribution of the variable for a specific observation is meant. It has been developed by Fisher et al. (2019) into a model-agnostic

version. In the second part of the result section, Section 4.2, feature importance will be used to identify the drivers of the models. The empirical values of the feature importances are computed via an equal technique as SHAP values are computed.

Feature importance allows to understand the relationship between the regressors and the dependent variable. It provides some intuition about the relationship between the data and the model since it does show the importance of a variable for the model. It does not necessarily reflect the predictive value of a variable on the model's output. According to Saarela & Jauhiainen (2021), the individual contribution of the variable for the model will be measured, regardless of the shape or direction of its effect.

3.5.2 Partial dependence

Partial dependence, as defined by Friedman (2001), among others, can be used to interpret models that work like a black-box, especially with a large number of predictor variables. Partial dependence captures the nature of the marginal influence of the variable on the prediction output, whether this relationship is linear, monotonic, or of a more complex nature. The marginal effect of the regressor will be derived from its SHAP value and its effect on the estimation output will be plotted in so-called partial dependence plots.

Partial dependence is different from feature importance since it does show the complexity of the relationship between the regressor and dependent variable, whether it is a monotonic, linear or more complex relationship. It provides some insight into the way that the model does use the variable value to produce output and what kind of relationships the model estimates between the variables and the dependent variable.

3.5.3 SHAP values

A recently developed method to increase the interpretability of ML models is the use of SHapley Additive exPlanations (SHAP) values by Lundberg & Lee (2017). SHAP values are based on game theory, where SHAP values quantify each player's contribution to the game. In this context, variables are considered as players and the model's prediction as the game.

Following Lundberg & Lee (2017), SHAP values can be computed as:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f_{S \cup \{i\}} \left(x_{S \cup \{i\}} \right) - f_S \left(x_S \right) \right], \tag{27}$$

in which F contains all variables and S the subset of variables of a given trained model. $f_S(x_S)$ contains the model's output with input regressors S. Hence, ϕ_i quantifies the

SHAP value of variable i. The first term of Equation 27 calculates the weight for the marginal contributions of variable i to model $f_S(x_S)$. The second term represents the specific marginal contribution of variable i to model f_S, computed by models that are repeatedly trained with and without variable i. Hence, SHAP values are the weighted average of all marginal contributions of a variable.

Usually, models cannot be evaluated by just a subset of variables. SHAP solves this by using conditional expectations to approximate $f_S(x_S)$ by $\mathbb{E}[f(x)|x_S]$, such that Equation 27 can be stated as:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left(\mathbb{E}\left[f(X) \mid X_{S \cup \{i\}} = x_{S \cup \{i\}} \right] - \mathbb{E}\left[f(X) \mid X_S = x_s \right] \right). \tag{28}$$

SHAP values assign the contribution of each variable to the model's prediction per data point. A schematic visualization of this relation is presented below, in Figure 5, obtained from Lundberg & Lee (2017). In this figure, the SHAP values for five variables are presented for one data point.

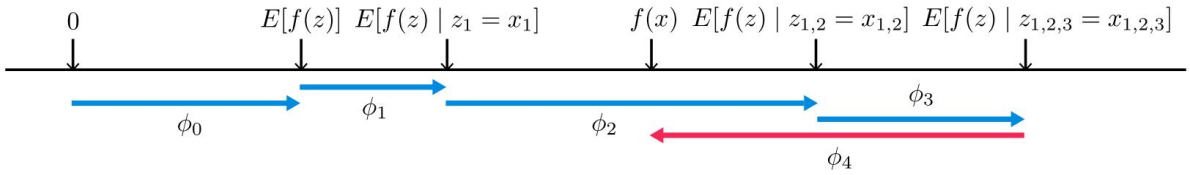


Figure 5: SHAP values (Lundberg & Lee (2017))

However, SHAP values have a disadvantage as well. Because it is necessary to compute all possible combinations of players, the calculation becomes a computational burden once many regressors are presented. Considering N variables, 2^N calculations must be computed per observation. To solve this, Lundberg & Lee (2017) proposed to estimate the SHAP values in their paper with a complexity of $O(TLN)$, with T number of trees, L the maximum amount of leaves in any tree, and $O(\cdot)$ as Big O notation. The latter describes the asymptotic upper limit regarding the computation complexity of the function. By this alternative algorithm, implemented in the *shap*[6] library in Python e.g., the computation time has been reduced from exponential to (low order) polynomial.

[6] https://shap.readthedocs.io/en/latest/index.html

4 Results

In this chapter, the results of the statistical and ML models will be presented. The nowcasting accuracy will be compared and reviewed for different time frames and economic situations. Further, the explainability results will be discussed.

4.1 Prediction accuracy

4.1.1 Full estimation period

The out-of-sample nowcasting performance of all models for 2006/Q1 - 2019/Q4 are presented in Table 2. Generally, it can be concluded that the ML models can outperform the statistical models. The results indicate that the SVR does obtain the highest point forecasting accuracy among the ML models for the combined MAE metric. XGBoost obtains the highest accuracy considering the RMSFE metric, on average. An equally-weighted average of the four ML models (WA) receives the best value for both metrics. Further, the results indicate that it yields to pick a different model for different estimation moments during the quarter.

Table 2: Out-of-sample performance 2006/Q1 - 2019/Q4

	1^{st} month		2^{nd} month		3^{rd} month		Combined	
	MAE	RMSFE	MAE	RMSFE	MAE	RMSFE	MAE	RMSFE
Statistical models								
ARIMA	0.478	0.683	0.478	0.683	0.478	0.683	0.478	0.683
DFM	0.572	0.758	0.533	0.732	0.507	0.671	0.537	0.721
ML models								
RF	**0.468**	0.672	0.479	0.670	0.439	0.646	0.462	0.663
XGBoost	0.502	0.681	**0.459**	**0.617**	0.426	0.615	0.462	**0.638**
LSTM	0.495	0.688	0.466	0.644	0.443	**0.597**	0.468	0.644
SVR	0.472	**0.670**	**0.459**	0.655	**0.419**	0.611	**0.450**	0.646
WA	0.470	0.661	0.437	0.625	0.413	0.593	0.440	0.627

Note: The values are multiplied with 10^2 for reading convenience.

Table 2 includes three different estimation moments per quarter. For example, if the US GDP growth value of Q1 will be estimated, 1^{st} month refers to the information set that is available at the end of January. The RF and SVR models are the most accurate during the first estimation moment. XGBoost is the most accurate ML model during the second estimation method, LSTM and SVR during the third estimation moment. The WA performs best on average, obtaining the lowest value for both MAE and RMSFE. This finding is in line with Timmermann (2006), who finds that simple forecasting combinations that

ignore correlations are useful, and even dominate individual forecasting models or combination schemes that use theoretically optimal combination weights in empirical studies. A possible explanation could be that individual models suffer more from structural breaks (Aiolfi et al. (2011)) or model misspecification (Timmermann (2006)), compared to combined forecasts.

Table 3 below shows the mean μ and standard deviation σ of the forecasting errors of each model, compared to the actual US GDP growth values. The values are computed as $\hat{y} - y$, in which $\hat{y}$ corresponds to the model's estimation and y to the actual value.

Table 3: Forecasting errors 2006/Q1 - 2019/Q4

	μ	σ
Statistical models		
ARIMA	0.100	0.676
DFM	0.402	**0.599**
ML models		
RF	0.247	0.615
XGBoost	0.120	0.627
LSTM	**0.002**	0.644
SVR	0.052	0.644

Note: The mean (μ) and standard deviation (σ) are multiplied with 10^2 for reading convenience.

Table 3 shows that the DFM overestimates the value on average by quite a margin, in particular. This also applies to the RF model, such that the RF model also overestimates the value on average. Notable is the fact that the LSTM model averages close to zero forecasting error. Further, it seems that models that overestimate more on average, tend to have a relatively smaller standard deviation compared to models that have a mean of forecasting errors closer to zero.

In Section 3.4.3, Diebold & Mariano (1995)'s test (DM-test) has been introduced. The DM-test investigates whether models' differences in forecasting accuracy are significant. A two-sided test with null hypothesis $H_0 : E[d_{t+1}] = 0$ is used to verify whether the forecasting errors of the ML models and statistical models are significantly different. The test statistics and corresponding p-values are reported in Table 4. It can be concluded that all ML models nowcast significantly different compared to the DFM, using $\alpha = 0.05$. The ARIMA model is assumed to be the most simple model but scores relatively well. ARIMA nowcasts significantly different compared to XGBoost and SVR, but not significantly different from the RF and LSTM. Hence, it can be concluded that all ML models significantly outperform the DFM and that XGBoost and SVR significantly outperform ARIMA.

Table 4: Diebold & Mariano (1995) test

	RF		XGBoost		LSTM		SVR	
	Test stat.	p-value	Test stat.	p-value	Test stat.	p-value	Test stat.	p-value
ARIMA	-1.044	0.297	-2.322	0.020	-1.643	0.100	-2.148	0.032
DFM	-2.003	0.045	-2.274	0.023	-2.431	0.015	-2.056	0.040

Note: Results are based on a two-sided test $H_0 : \mathbb{E}[d_{t+1}] = 0$. $\alpha = 0.05$ is assumed for significance.

As mentioned already, the MAE and RMSFE metrics do not consider the same model as the 'best' model at each estimation moment. The RMSFE metric emphasizes larger errors compared to the MAE. Models that are constantly not too far off receive a relatively lower RMSFE value, while models with only a few large errors receive a relatively higher value. To gain more insight into the behavior of the models, Figure 6 visualizes the out-of-sample predictions compared to the actual values. The plots show point nowcasts in a monthly frequency since three estimations per quarter are obtained.
Considering Figure 6, it is clear to see why ARIMA scores well in terms of the RMSFE metric. Its predictions are constantly in between spikes, and the model does not adjust to larger quarter-to-quarter differences and stays therefore between 'safe' margins. DFM does not adjust correctly to the values, and its predictions are too high continuously, as already concluded from Table 3. However, it is the only model that at least reacts to the sharp decrease of the US GDP growth in 2008.

The graph shows that RF cannot adjust to present circumstances since its predictions show similar behaviour as the ARIMA model by staying in the middle of peaks and drops. It does not react much. An equal pattern could be observed for the other three ML models. XGBoost can produce some peaks that stand out, but are not in line with a great result. As it seems, the ML models are not able to react to larger decreases in US GDP growth. A possible explanation could be that 95.8% of the quarter-to-quarter US GDP growth data is positive, such that the models cannot be trained properly to account for this. For statistical models, there is no need for a similar extent for training compared to the ML models, such that they suffer less from the lack of recession data.

4.1.2 Pre, during, and post financial crisis

As observed in previous Subsection 4.1.1, it seems that the state of the economy does influence the nowcasting accuracy of the proposed models. Accounting for business cycle patterns might improve the nowcasting performance of the models. It can be observed from Figure 6, for example, that only the DFM seems to react to the decrease in US GDP growth around 2008.

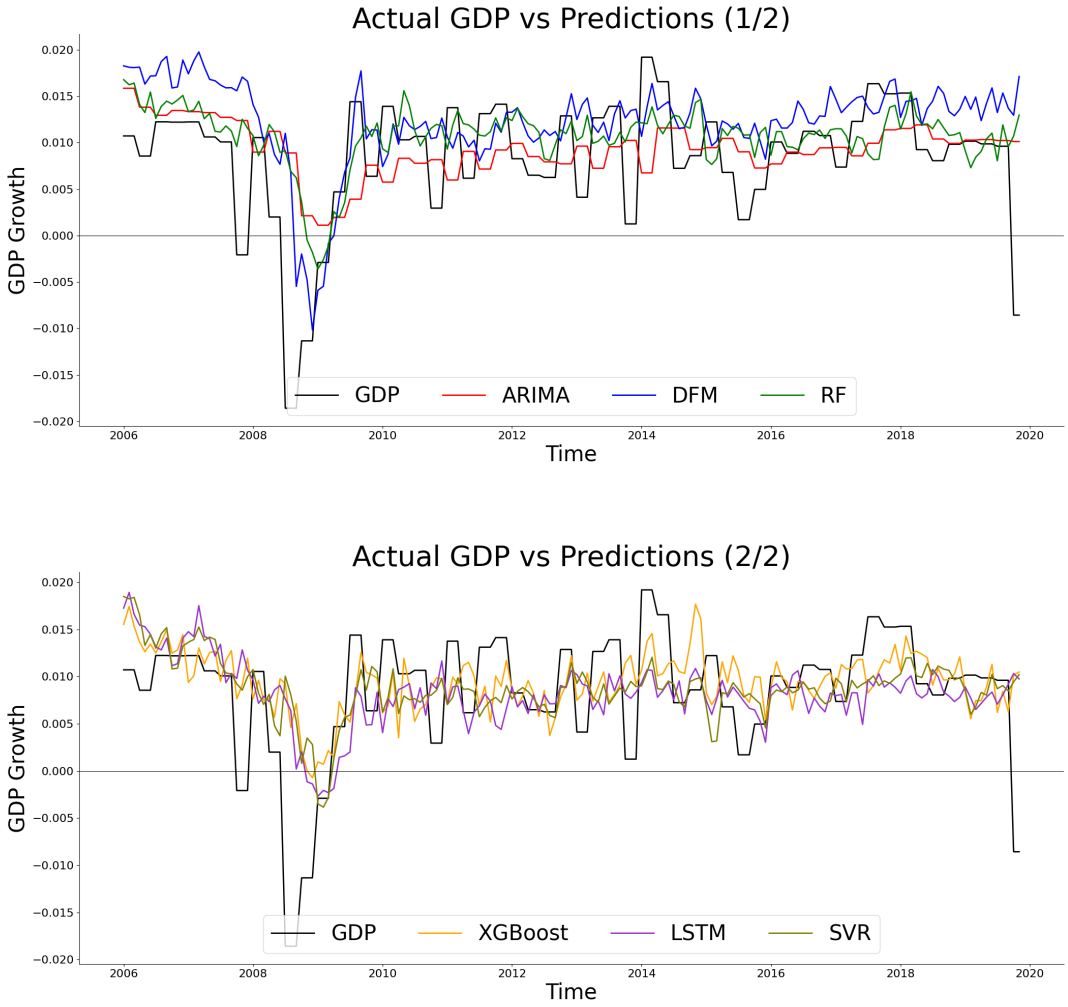


Figure 6: Out-of-sample predictions compared to actual US GDP growth.

The out-of-sample nowcasting performance period of 2006/Q1 - 2019/Q4 contains one major financial crisis: the financial crisis starting in 2007. This financial crisis has mainly been caused by cheap credit and lax lending standards fueling a housing bubble in the US. When the bubble burst, homeowners owed more on their mortgages than their homes were worth (Duca et al. (2010)). Financial institutions were left holding trillions of dollars worth of investments in subprime mortgages.

Financial markets around the world were showing signs that the reckoning was overdue for cheap credit, though it took until 2007Q4 to have its impact on US GDP growth considering the data set. Therefore, the pre-financial crisis period has been depicted as 2006Q1 - 2007Q3. The period that refers to the actual period of the financial crisis is 2007Q4 - 2009Q1, since US GDP growth increased from then. For the post-financial crisis period, 2009Q2 - 2014Q1 has been determined. The exact post-financial crisis period is hard to identify, but 2014Q1 has been picked because it contains the last large spike after

the crisis period affirming almost complete recovery.

Table 5: Out-of-sample performance 2006/Q1 - 2014/Q1

	Pre		During		Post		Combined	
	MAE	RMSFE	MAE	RMSFE	MAE	RMSFE	MAE	RMSFE
Statistical models								
ARIMA	0.267	0.319	1.170	1.442	0.507	0.593	0.577	0.783
DFM	0.656	0.670	0.976	1.300	**0.423**	0.521	0.573	0.753
ML models								
RF	0.283	0.344	1.028	1.341	**0.423**	**0.498**	**0.504**	0.709
XGBoost	**0.254**	**0.312**	1.021	1.289	0.446	0.509	0.510	**0.693**
LSTM	0.334	0.414	0.948	**1.228**	0.461	0.547	0.522	0.701
SVR	0.331	0.418	**0.959**	1.317	0.437	0.504	0.509	0.711
WA	0.275	0.357	0.981	1.284	0.417	0.482	0.490	0.684

Note: The values are multiplied with 10^2 for reading convenience. The headings refer to pre (2006Q1 - 2007Q3), during (2007Q4 - 2009Q1) and post (2009Q4 - 2014Q1) financial crisis periods.

Table 5 shows that XGBoost performed best pre-financial crisis. During the crisis, LSTM and SVR performed best. RF performed best in the post-financial crisis period. Combined, RF and XGBoost performed best. Notable is the fact that WA again performs best overall, but performs worst pre and during the financial crisis compared to individual models. It can be concluded that the models, in general, have a hard time predicting US GDP growth in crisis periods. Further, Table 5 clearly shows that it yields to pick a different model regarding the current economic situation.

4.2 Explainability results

In this section, multiple methods to explain the ML models' workings and results are presented. More insight into why certain models can outperform others will be gained by feature importance, partial dependence and SHAP values.

4.2.1 Feature importance

The focus will be shifted to the effect of individual variables on the nowcasting predictions. Feature importance will be used to quantify the importance of individual variables on the prediction outcome. The 20 most influential regressors are presented in so-called waterfall plots, in Figure 7. Besides, the mean and volatility of these and each other regressors' relative importance through the prediction period is presented in Tables 11, 12, 13 and 14 of the Appendix. Below, the bars visualize the relative marginal effect (%) of each variable based on the same technique as for SHAP values, as explained in Section 3.5.3.

The dotted lines present the cumulative effect of the variables. The plots are generated considering the average feature importance of the models through the estimation period.

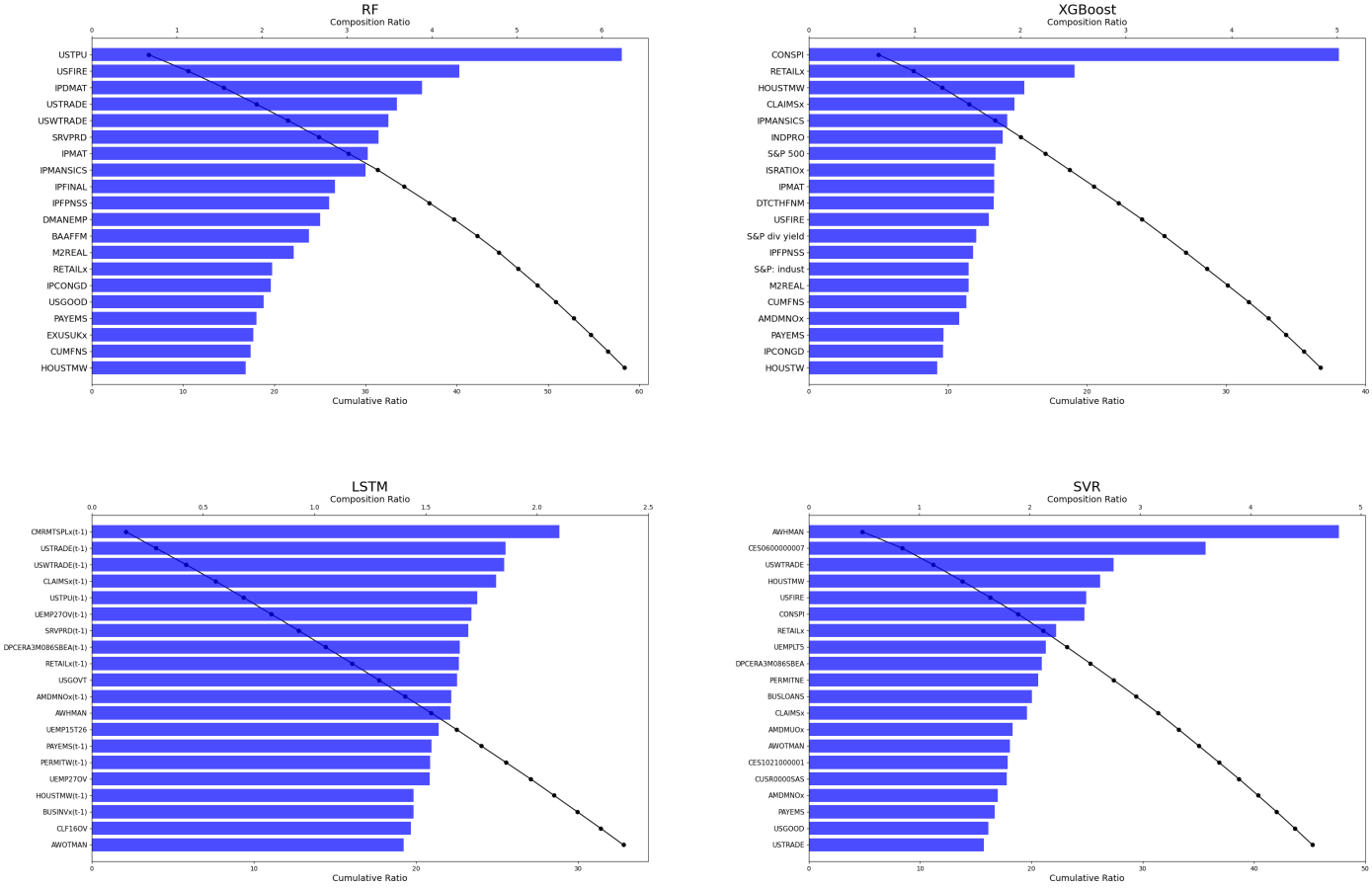


Figure 7: Waterfall plots presenting feature importance.

It can be observed that the 20 most critical features make up about 60% of the total influence for RF. This is quite large, considering the total of 123 independent variables. As explained in Section 3.2.1, RF is based on bagging numerous smaller decision trees. During the formation of these decision trees, random samples of features are selected for each split. The most important are selected first and are therefore relatively of larger importance for the model. This might be the reason that the few important variables selected first do have a larger relative influence on the output value. This is not the case for XGBoost since only about 37% is explained by the top 20 most critical variables. This is much less compared to RF. Also, the most important variable is relatively more dominant, having double the influence as the second most important variable. The main difference between RF and XGBoost is that XGBoost uses boosting. This means that XGBoost picks one regressor for the first split and then probably uses less correlated variables for the next split. RF builds trees from a selected set of variables, resulting in a different pattern.

The waterfall plot considering LSTM shows a different pattern. The cumulative feature importance of the top 20 most influential variables covers only about 32%, and most variables represent a similar influence on the output value. Especially the latter is different from the other ML models. LSTM is, and NNs, in general, can combat multicollinearity. Since many regressors in the data set are highly correlated, it could appear that models wrongly estimate the marginal effect of a regressor. Also, it can be observed that relatively much of the top 20 critical variables are variables with a one-period lag. This might suggest that the LSTM model can benefit from its memory ability. The waterfall plot linked to SVR does not contain any distinguishable characteristics. The first 20 most critical variables cover about 45 % of the total relative feature importance.

To further research the models' feature importance, the most influential variables per model will be researched through the out-of-sample estimation period. Figure 8 presents the relative feature importance (%) plotted for all nowcasting estimations through time. Only five of the most influential variables are presented to keep good readability. The less influential variables follow a similar pattern, such that it is not necessary to present them all.

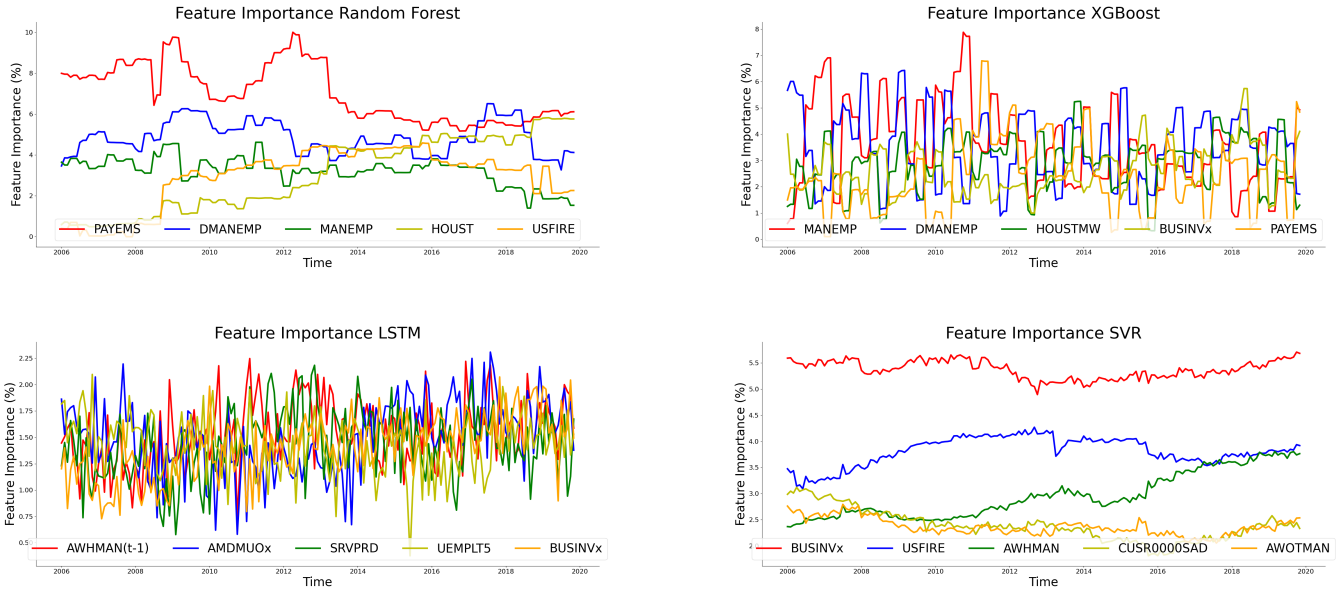


Figure 8: Features importance through the estimation period.

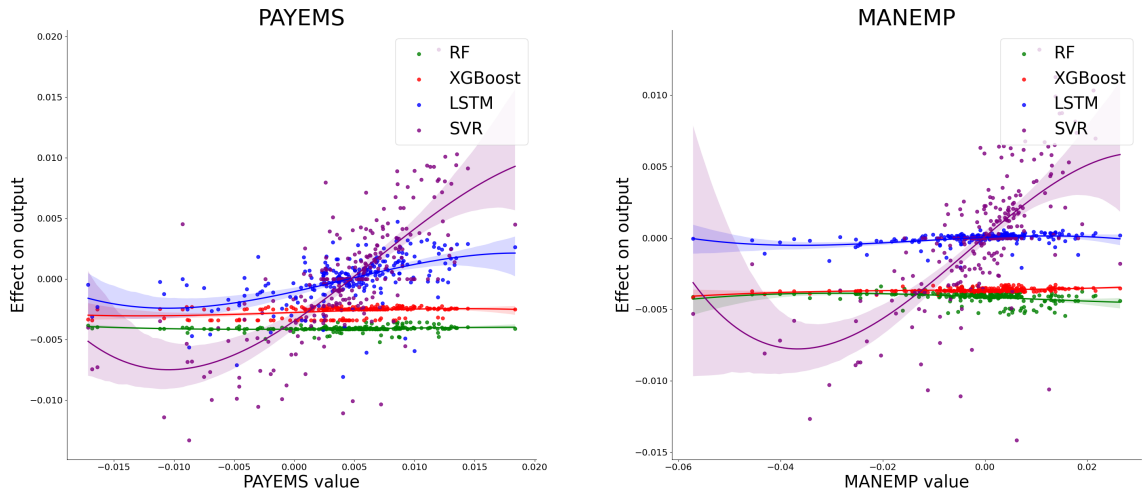
The plots of Figure 8 indicate that each ML model has its typical pattern of relative feature importance through time. For RF, the feature importance changes gradually through time. The HOUST feature, for example, does only slightly influence the out-of-sample prediction at the start of the period. In the end, this variable becomes almost equal to the most influential one. XGBoost allows for large fluctuations of relative feature importance. The resulting pattern is volatile and is often linked to seasonality. LSTM considers

most features similar and the feature importance shows a volatile pattern. The changes are spiky but not of great magnitude. The feature importance of the SVR shows a constant pattern throughout the estimation period. The relative feature importance does not change much in value or order over time.

It can be concluded from Figure 8 that SVR's estimation output depends similarly on its features throughout the estimation period. The order and magnitude seem to stay constant, such that it is straightforward which features contribute the most to the model to produce point forecasts. RF shows a similar pattern, with slightly more volatility. The behavior of feature importance through time of XGBoost and LSTM makes it challenging to discover the drivers behind the models' output.

4.2.2 Partial dependence

Partial dependence shows how a particular regressor is related to the dependent variable. The partial dependence plots presented in Figure 9 show the relationship between the values of four relatively essential variables and the estimation output of each ML model. The dots represent the variable's value and its corresponding effect on the prediction output. A curve with a confidence interval has been fitted to identify the shape of the relationship of the variable's value and its effect on the estimation.



From the plots, it can be concluded that variable values have a generally constant relationship for the impact on the model output considering RF and XGBoost. As it seems for RF and XGBoost, variable values do not change the effect of a variable on the model's output. This is probably due to the fact that tree-based ensemble models do use splits to divide variable space in order to compute prediction output, such that the use and ranking of the variable are more important compared to the variable values. The re-

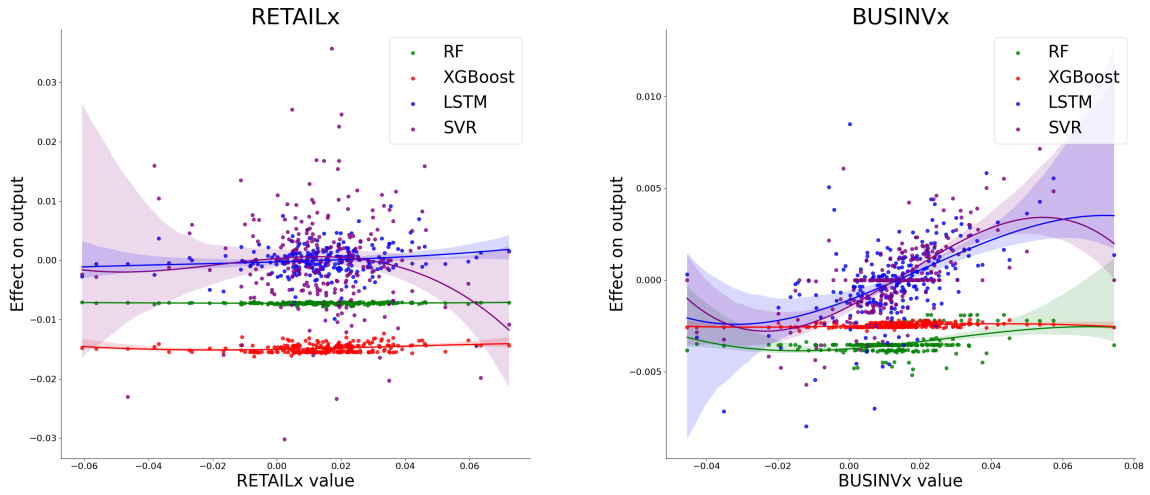


Figure 9: Partial dependence plots.

lationship between the variables' values and their effect on the model's output is more complex considering LSTM and SVR. LSTM has a relatively more constant relationship, however, the scatterplots show some dispersion. This indicates a complex relationship between the variables' values and their effect on the model's output. SVR shows a very complex relationship as well, which can hardly be captured by a 3^{rd}-order polynomial curve, as visualized. The dispersion is quite large. It seems that both LSTM and SVR use underlying nonlinear relationships in order to produce model output.

4.2.3 SHAP values

To present another insight on the impact of various variables on the point predictions, summary plots of SHAP values are presented in Figure 10. The plots deliver a systematic view of the positive or negative effects of the variables on the point prediction. Each dot represents a monthly observation over the estimation period 2006/Q1 - 2019/Q4. Note that the variable order does not necessarily equal the feature importance order of Figures 7 and 8. Before, the feature importance driving forecasting has been computed per nowcast through the out-of-sample period. Now, the feature importance has been visualized at the end of the period using the complete information set.

The SHAP summary plots of RF and XGBoost show some similarities. The SHAP values of both models are more compact compared to the LSTM and SVR plots. Further, RF and XGBoost have relatively many dots colored the same on both sides of the y-axis and many dots of both colors on the same side of the y-axis. This is quite unexpected because the same variable value might have a different effect on the prediction at each point in time. However, this is in line with what Figure 9 concluded earlier. It indicates that the

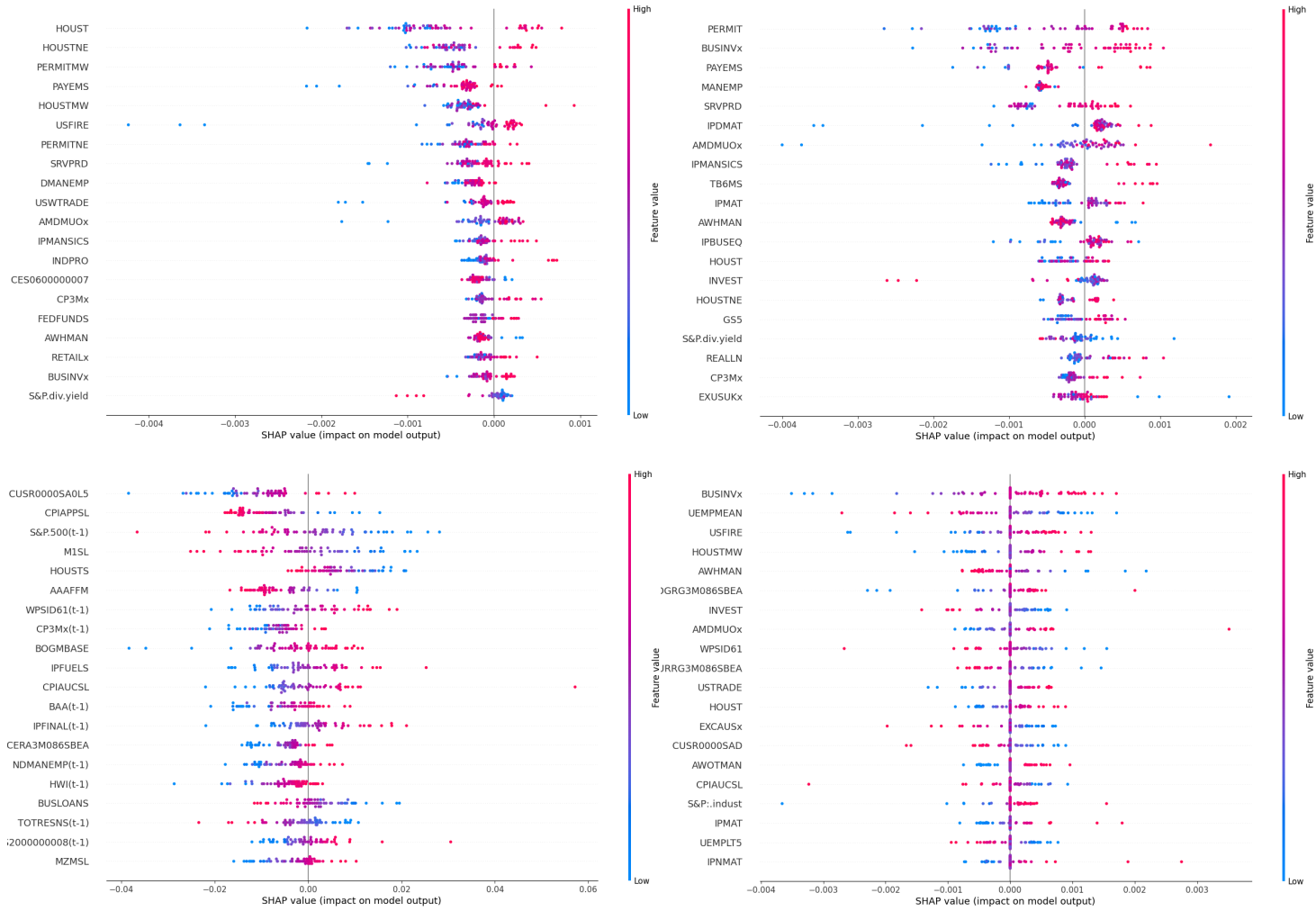


Figure 10: SHAP summary plot.

same variable value can have both a negative and positive effect on the model's prediction for tree-based ensemble models. This makes it hard to develop intuition about the effect of a variable value because it can either have a positive or negative influence.

The plots referring to LSTM and SVR show less compact SHAP values. The SHAP values for LSTM are somewhat compacter than SVR. For both models, the dots are more similarly colored at one side of the y-axis, which is more as expected. An odd finding in Figure 10 for SVR is the amount of dots centered at the y-axis for each variable. After consulting each observation individually, it turns out that SVR does not use each variable value at each observation if SVR predicts. For example, the variable value of RPI has only been used from five observations to have a partial effect on the prediction. An explanation might be that the SVR does not use those variables values at all since the values could lay within the ϵ-tube, as explained by Section 3.2.4.

5 Conclusion

The central question in this paper that has been answered is whether ML models can outperform statistical models when it comes to nowcasting in a real-time empirical application. The results from Section 4.1 indicate that each ML model can outperform the statistical models. It has been determined that each ML model outperforms the DFM significantly. Two out of the four ML models also yield significantly better prediction results than the ARIMA model. Further, SVR obtains the highest nowcasting accuracy among each model considering the MAE metric, while XGBoost receives the highest score considering the RMSFE measure. However, an equally-weighted forecasting combination of the four ML models' predictions receives the best results for both metrics. Also, it turns out that it yields to pick a different model at different estimation moments during the quarter, and regarding the current economic situation.

Another critical problem in present literature that has been considered is that ML models are often interpreted as black-box procedures. The drivers behind the ML models' predictions have been discovered by feature importance, partial dependence and SHAP values. The results following from Section 4.2 indicate that the ML models have a different pattern regarding the marginal relationship of variable values on the prediction output. For RF, fewer variables drive a larger part of the model's prediction compared to the other ML models. This feature importance only changes gradually throughout the estimation period. It turns out that for tree-based ensemble ML models the variable value is of less importance but that the feature order and split value have a more prominent effect on the output value. This applies for XGBoost also, however, more variables drive less of the model's output compared to RF. The pattern of feature importance is different, especially throughout the estimation period.

LSTM and SVR show a different relationship regarding the influence of variable values and their effect on the models' output. LSTM's variables are of almost equal importance for the model's output. The marginal contribution of the variable values has a complex relationship with the model's output, considering the large dispersion in the partial dependence plots. SVR shows similar characteristics, but a larger part of what the model's output drives consists of fewer variables. For SVR, this complex relationship seems to be partly caused by the use of the ϵ-insensitivity tube, which omits numerous observations in order to produce model output.

6 Discussion

Multiple statistical and ML models have been selected and formed in this research to review their nowcasting accuracy in a real-time empirical application. Also, methods to mitigate the problem of interpretability have been presented. This discussion will review ways for improvement and further research.

A direct suggestion for improvement is a more refined grid search for the ML models' hyperparameter optimization. In particular, optimizing the LSTM neural network's hyperparameters costs many resources to optimize. Even with the help of cloud computing. For each other model, more hyperparameters of less importance could be optimized as well. This could lead to a slightly better fit to the data, possibly resulting in a higher nowcasting accuracy.

Another improvement that could be suggested is the introduction of other ML models. These ML models could be more state-of-the-art or more advanced versions of the used ones. It is believed that the used models are the most evident and relevant in this application. However, newer or more advanced ML models could be added, such as the sg-LASSO-MIDAS model of Babii et al. (2021), for example, or a Bayesian approach for the LSTM network. Incorporating more models might lead to finding models with a higher forecasting accuracy.

Further, the results indicated that a simple equally-weighted forecast combination does outperform the individual ML models with quite a margin. It is out of scope for this paper, but to increase the nowcasting accuracy, other methods to combine forecasts could be tested to optimize weight schemes. This might lead to forecasting combinations that increase performance.

Finally, methods to change the input of the ML models are not considered. Literature suggests that changing the input of the ML models can improve nowcasting accuracy. Soybilgen & Yazgan (2021), for example, first extract dynamic factors out of the data set and feed these factors into tree-based ensemble machine learning models. Another method, as used by Qureshi et al. (2020), is based on removing the least critical features step by step until only a limited amount of features are left. These kind of methods are not considered. However, estimation accuracy might be increased by using these type of methods.

7 Acknowledgements

I would like to thank dr. Andreas Pick for his constructive criticism and guidance during this project. The collaboration has been pleasant, especially considering the circumstances during the writing process. COVID-19 has been present throughout the whole process which made physical conversation and collaboration inapplicable. Still, I have felt fully supported during the process.

Also, I would like to thank the second assessor Nick Koning MSc. for his judgement in order to evaluate this paper.

References

Adya, M., & Collopy, F. (1998). How effective are neural networks at forecasting and prediction? a review and evaluation. *Journal of Forecasting*, *17*(5-6), 481–495.

Aiolfi, M., Capistrán, C., & Timmermann, A. (2011). Forecast combinations. *Oxford Handbooks Online*, 355 – 388.

Babii, A., Ghysels, E., & Striaukas, J. (2021). Machine learning time series regressions with an application to nowcasting. *Journal of Business Economic Statistics*, 1–33.

Bańbura, M., Giannone, D., Modugno, M., & Reichlin, L. (2013). Now-casting and the real-time data flow. *Handbook of Economic Forecasting*, 195–237.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Binner, J., Bissoondeeal, R., Elger, T., Gazely, A., & Mullineux, A. (2005). A comparison of linear forecasting models and neural networks: an application to euro inflation and euro divisia. *Applied Economics*, *37*(6), 665–680.

Bok, B., Caratelli, D., Giannone, D., Sbordone, A. M., & Tambalotti, A. (2018). Macroeconomic nowcasting and forecasting with big data. *Annual Review of Economics*, *10*(1), 615–643.

Bolhuis, M., & Rayner, B. (2020). *Deus ex machina? a framework for macro forecasting with machine learning* (IMF Working Paper No. 45).

Bouwman, K. E., & Jacobs, J. P. (2011). Forecasting with real-time macroeconomic data: The ragged-edge problem and revisions. *Journal of Macroeconomics*, *33*(4), 784–792.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(3), 261–277.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (2017). Regression trees. In *Classification and regression trees* (1st ed.). Routledge.

Chakraborty, C., & Joseph, A. (2017, Nov). Machine learning at central banks. *SSRN Electronic Journal*.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, *74*(366), 427.

Diebold, F. X., & Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business Economic Statistics*, *13*(3), 253.

Doz, C., Giannone, D., & Reichlin, L. (2011). A two-step estimator for large approximate dynamic factor models based on kalman filtering. *Journal of Econometrics*, *164*(1), 188-205.

Drucker, H., Burges, C., Kaufman, L., Smola, A., & Vapnik, V. (1997, 01). Support vector regression machines. *Adv Neural Inform Process Syst*, *28*, 779-784.

Duca, J., Muellbauer, J., & Murphy, A. (2010). Housing markets and the financial crisis of 2007–2009: Lessons for the future. *Journal of Financial Stability*, *6*(4), 203-217.

Elliott, G., Granger, C. W. J., & Timmermann, A. (2006). *Handbook of economic forecasting* (Vol. 1). Elsevier North Holland.

Fan, F., Xiong, J., Li, M., & Wang, G. (2021). *On interpretability of artificial neural networks: A survey.*

Federal Reserve Bank of New York. (2021). *Nowcasting report: Methodology.*

Fisher, A., Rudin, C., & Dominici, F. (2019). *All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously.*

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5).

Ghysels, E., Santa-Clara, P., & Valkanov, R. (2004). *The midas touch: Mixed data sampling regression models* (CIRANO Working Papers). CIRANO.

Giannone, D., Reichlin, L., & Small, D. (2008). Nowcasting: The real-time informational content of macroeconomic data. *Journal of Monetary Economics*, *55*(4), 665–676.

Gordon, A. D., Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and regression trees. *Biometrics*, *40*(3), 874.

Gunn, S. (1998, May). *Support vector machines for classification and regression* (ISIS Technical Report). University of Southampton.

Hastie, T., Friedman, J., & Tibshirani, R. (2008). *The elements of statistical learning: data mining, inference, and prediction.* Springer.

H. Drucker, L. K. A. S., C. Burges, & Vapnik, V. (1997). Support vector regression machines. *Advances in neural information processing systems*, *9*(3), 155–161.

Hochreiter, S., & Schmidhuber, J. (1997, 12). Long short-term memory. *Neural computation*, *9*, 1735-80.

Hopp, D. (2021, March). *Economic nowcasting with long short-term memory artificial neural networks (lstm)* (Research Paper No. 62). UNCTAD Research.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: with applications in R*. Springer.

Jansen, W. J., Jin, X., & Winter, J. M. D. (2016). Forecasting and nowcasting real gdp: Comparing statistical models and subjective forecasts. *International Journal of Forecasting*, *32*(2), 411–436.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, *82*(Series D), 35–45.

Kim, K.-J. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*(1-2), 307–319.

Liaw, A., & Wiener, M. (2018). Breiman and cutler's random forests for classification and regression.

Lins, I., Moura, M., Silva, M., Droguett, E., Veleda, D., Araujo, M., & Jacinto, C. M. (2010, 01). Sea surface temperature prediction via support vector machines combined with particle swarm optimization..

Loermann, J., & Maas, B. (2019). *Nowcasting us gdp with artificial neural networks* (MPRA Paper No. 5459). Munich Personal RePEc Archive.

Loh, W.-Y. (2014, 06). Fifty years of classification and regression trees. *International Statistical Review*, *82*.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. Curran Associates Inc.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, *13*(3).

McCracken, M. W. (2021). *Appendix - fred-md.*

Mccracken, M. W., & Ng, S. (2015). *FRED-MD: A monthly database for macroeconomic research* (Working Paper No. 2015-012). Federal Reserve Bank of St. Louis.

Molnar, C. (2019). *Interpretable machine learning: a guide for making black box models interpretable*. Lulu.

Olah, C. (2015). *Understanding lstm networks.* Retrieved from `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers* (pp. 61–74). MIT Press.

Qureshi, S., Chu, B. M., & Demers, F. S. (2020, August). *Forecasting canadian gdp growth using xgboost* (Carleton Economics Working Paper No. CEWP 20-14). Charleton University.

Richardson, A., van Florenstein Mulder, T., & Vehbi, T. (2021). Nowcasting GDP using machine-learning algorithms: A real-time assessment. *International Journal of Forecasting*, *37*(2), 941-948.

Saarela, M., & Jauhiainen, S. (2021, February 03). Comparison of feature importance measures as explanations for classification models. *SN Applied Sciences*, *3*(2), 272.

Schumacher, C. (2016). A comparison of midas and bridge equations. *International Journal of Forecasting*, *32*(2), 257–270.

Schumacher, C., & Breitung, J. (2008). Real-time forecasting of german gdp based on a large factor model with monthly and quarterly data. *International Journal of Forecasting*, *24*(3), 386–398.

Soybilgen, B., & Yazgan, E. (2021). Nowcasting us gdp using tree-based ensemble models and dynamic factors. *Computational Economics*, *57*(1), 387–417.

Stevanovic, D., Surprenant, S., & Coulombe, P. G. (2019, October). *How is Machine Learning Useful for Macroeconomic Forecasting?* (CIRANO Working Papers No. 2019s-22). CIRANO.

Stock, J. H., & Watson, M. W. (2002). Forecasting using principal components from a large number of predictors. *Journal of the American Statistical Association*, *97*(460), 1167–1179.

Timmermann, A. (2006). Forecast Combinations. In G. Elliott, C. Granger, & A. Timmermann (Eds.), *Handbook of Economic Forecasting* (Vol. 1, p. 135-196). Elsevier.

Tsamardinos, I., Rakhshani, A., & Lagani, V. (2014, 05). Performance-estimation properties of cross-validation- based protocols with simultaneous hyper-parameter optimization. In (Vol. 24).

Veaux, R. D., Psichogios, D., & Ungar, L. (1993). A comparison of two nonparametric estimation schemes: Mars and neural networks. *Computers Chemical Engineering*, *17*(8), 819–837.

A Appendix

A.1 FRED-MD

Table 6: Description of the FRED-MD

	Description	FRED code	Group	Release	Release date	Released by	t code
1	Real Personal Income	RPI	1	Personal Income and Outlays	February 26	BEA	5
2	Real personal income ex transfer receipts	W875RX1	1	Personal Income and Outlays	February 26	BEA	5
3	IP Index	INDPRO	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
4	IP: Final Products and Nonindustrial Supplies	IPFPNSS	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
5	IP: Final Products (Market Group)	IPFINAL	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
6	IP: Consumer Goods	IPCONGD	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
7	IP: Durable Consumer Goods	IPDCONGD	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
8	IP: Nondurable Consumer Goods	IPNCONGD	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
9	IP: Business Equipment	IPBUSEQ	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
10	IP: Materials	IPMAT	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
11	IP: Durable Materials	IPDMAT	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
12	IP: Nondurable Materials	IPNMAT	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
13	IP: Manufacturing (SIC)	IPMANSICS	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
14	IP: Residential Utilities	IPB51222s	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
15	IP: Fuels	IPFUELS	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	5
16	Capacity Utilization: Manufacturing	CUMFNS	1	Industrial Production and Capacity Utilization - G.17	February 17	FRB	2
17	Help-Wanted Index for United States	HWI	2	Job Openings and Labor Turnover Survey (JOLTS)	March 11	BLS	2
18	Ratio of Help Wanted/No. Unemployed	HWIURATIO	2	Job Openings and Labor Turnover Survey (JOLTS)	March 11	BLS	2
19	Civilian Labor Force	CLF16OV	2	Current Employment Statistics (CES)	February 5	BLS	5
20	Civilian Employment	CE16OV	2	Current Employment Statistics (CES)	February 5	BLS	5
21	Civilian Unemployment Rate	UNRATE	2	Current Employment Statistics (CES)	February 5	BLS	2
22	Average Duration of Unemployment (Weeks)	UEMPMEAN	2	Current Employment Statistics (CES)	February 5	BLS	2
23	Civilians Unemployed - Less Than 5 Weeks	UEMPLT5	2	Current Employment Statistics (CES)	February 5	BLS	5
24	Civilians Unemployed for 5-14 Weeks	UEMP5TO14	2	Current Employment Statistics (CES)	February 5	BLS	5
25	Civilians Unemployed - 15 Weeks and Over	UEMP15OV	2	Current Employment Statistics (CES)	February 5	BLS	5
26	Civilians Unemployed for 15-26 Weeks	UEMP15T26	2	Current Employment Statistics (CES)	February 5	BLS	5
27	Civilians Unemployed for 27 Weeks and Over	UEMP27OV	2	Current Employment Statistics (CES)	February 5	BLS	5
28	Initial Claims	CLAIMSx	2	Unemployment Insurance Weekly Claims Report	February 6	ETA	5
29	All Employees: Total nonfarm	PAYEMS	2	Current Employment Statistics (CES)	February 5	BLS	5
30	All Employees: Goods-Producing Industries	USGOOD	2	Current Employment Statistics (CES)	February 5	BLS	5
31	All Employees: Mining and Logging: Mining	USMINE	2	Current Employment Statistics (CES)	February 5	BLS	5
32	All Employees: Construction	USCONS	2	Current Employment Statistics (CES)	February 5	BLS	5
33	All Employees: Manufacturing	MANEMP	2	Current Employment Statistics (CES)	February 5	BLS	5
34	All Employees: Durable goods	DMANEMP	2	Current Employment Statistics (CES)	February 5	BLS	5
35	All Employees: Nondurable goods	NDMANEMP	2	Current Employment Statistics (CES)	February 5	BLS	5
36	All Employees: Service-Providing Industries	SRVPRD	2	Current Employment Statistics (CES)	February 5	BLS	5
37	All Employees: Trade, Transportation and Utilities	USTPU	2	Current Employment Statistics (CES)	February 5	BLS	5
38	All Employees: Wholesale Trade	USWTRADE	2	Current Employment Statistics (CES)	February 5	BLS	5
39	All Employees: Retail Trade	USTRADE	2	Current Employment Statistics (CES)	February 5	BLS	5
40	All Employees: Financial Activities	USFIRE	2	Current Employment Statistics (CES)	February 5	BLS	5
41	All Employees: Government	USGOVT	2	Current Employment Statistics (CES)	February 5	BLS	5
42	Avg Weekly Hours : Goods-Producing	CES0600000007	2	Current Employment Statistics (CES)	February 5	BLS	1
43	Avg Weekly Overtime Hours : Manufacturing	AWOTMAN	2	Current Employment Statistics (CES)	February 5	BLS	2
44	Avg Weekly Hours : Manufacturing	AWHMAN	2	Current Employment Statistics (CES)	February 5	BLS	1
45	Avg Hourly Earnings : Goods-Producing	CES0600000008	2	Current Employment Statistics (CES)	February 5	BLS	6
46	Avg Hourly Earnings : Construction	CES2000000008	2	Current Employment Statistics (CES)	February 5	BLS	6
47	Avg Hourly Earnings : Manufacturing	CES3000000008	2	Current Employment Statistics (CES)	February 5	BLS	6
48	Housing Starts: Total New Privately Owned	HOUST	3	New Residential Construction	February 18	USCB	4
49	Housing Starts, Northeast	HOUSTNE	3	New Residential Construction	February 18	USCB	4
50	Housing Starts, Midwest	HOUSTMW	3	New Residential Construction	February 18	USCB	4
51	Housing Starts, South	HOUSTS	3	New Residential Construction	February 18	USCB	4
52	Housing Starts, West	HOUSTW	3	New Residential Construction	February 18	USCB	4
53	New Private Housing Permits (SAAR)	PERMIT	3	New Residential Construction - Permits	February 24	USCB	4
54	New Private Housing Permits, Northeast (SAAR)	PERMITNE	3	New Residential Construction - Permits	February 24	USCB	4
55	New Private Housing Permits, Midwest (SAAR)	PERMITMW	3	New Residential Construction - Permits	February 24	USCB	4
56	New Private Housing Permits, South (SAAR)	PERMITS	3	New Residential Construction - Permits	February 24	USCB	4
57	New Private Housing Permits, West (SAAR)	PERMITW	3	New Residential Construction - Permits	February 24	USCB	4
58	Real personal consumption expenditures	DPCERA3M086S	4	Personal Income and Outlays	February 26	BEA	5
59	Real Manu. and Trade Industries Sales	CMRMTSPLx	4	Supplemental Estimates, Underlying Detail Tables	February 26	BEA	5
60	Retail and Food Services Sales	RETAILx	4	Monthly Retail Trade and Food Services	March 16	USCB	5
61	New Orders for Consumer Goods	ACOGNO	4	Manufacturer's Shipments, Inventories and Orders	February 25	USCB	5
62	New Orders for Durable Goods	AMDMNOx	4	Manufacturer's Shipments, Inventories and Orders	February 25	USCB	5
63	New Orders for Nondefense Capital Goods	ANDENOx	4	Manufacturer's Shipments, Inventories and Orders	February 25	USCB	5
64	Unfilled Orders for Durable Goods	AMDMUOx	4	Manufacturer's Shipments, Inventories and Orders	February 25	USCB	5
65	Total Business Inventories	BUSINVx	4	Manufacturing and Trade Inventories and Sales	March 16	USCB	5
66	Total Business: Inventories to Sales Ratio	ISRATIOx	4	Manufacturing and Trade Inventories and Sales	March 16	USCB	2
67	Consumer Sentiment Index	UMCSENTx	4	Surveys of Consumers	February 1	UMich	2
68	M1 Money Stock	M1SL	5	H.6 Money Stock Measures	February 23	FRB	6
69	M2 Money Stock	M2SL	5	H.6 Money Stock Measures	February 23	FRB	6
70	Real M2 Money Stock	M2REAL	5	Real Money Stock Measures	February 23	FRED	5
71	Monetary Base	BOGMBASE	5	H.6 Money Stock Measures	February 23	FRB	6
72	Total Reserves of Depository Institutions	TOTRESNS	5	H.6 Money Stock Measures	February 23	FRB	6
73	Reserves Of Depository Institutions, Nonborrowed	NONBORRES	5	H.6 Money Stock Measures	February 23	FRB	7
74	Commercial and Industrial Loans	BUSLOANS	5	H.8 Assets and Liabilities of Commercial Banks	February 12	FRB	6
75	Real Estate Loans at All Commercial Banks	REALLN	5	H.8 Assets and Liabilities of Commercial Banks	February 12	FRB	6
76	Total Nonrevolving Credit	NONREVSL	5	G.19 Consumer Credit	March 5	FRB	6
77	Nonrevolving consumer credit to Personal Income	CONSPI	5	G.19 Consumer Credit	March 5	FRB	2
78	MZM Money Stock	MZMSL	5	Money Zero Maturity	February 23	FRED	6
79	Consumer Motor Vehicle Loans Outstanding	DTCOLNVHFNM	5	G.20 Finance Companies	February 26	FRB	6
80	Total Consumer Loans and Leases Outstanding	DTCTHFNM	5	G.20 Finance Companies	February 26	FRB	6

	Description	FRED code	Group	Release	Release date	Released by	t code
81	Securities in Bank Credit at All Commercial Banks	SBCACBW027SBOG	5	H.8 Assets and Liabilities of Commercial Banks	February 12	FRB	6
82	Effective Federal Funds Rate	FEDFUNDS	6	H.15 Selected Interest Rates	February 1	FRB	2
83	3-Month AA Financial Commercial Paper Rate	CP3Mx	6	H.15 Selected Interest Rates	February 1	FRB	2
84	3-Month Treasury Bill:	TB3MS	6	H.15 Selected Interest Rates	February 1	FRB	2
85	6-Month Treasury Bill:	TB6MS	6	H.15 Selected Interest Rates	February 1	FRB	2
86	1-Year Treasury Rate	GS1	6	H.15 Selected Interest Rates	February 1	FRB	2
87	5-Year Treasury Rate	GS5	6	H.15 Selected Interest Rates	February 1	FRB	2
88	10-Year Treasury Rate	GS10	6	H.15 Selected Interest Rates	February 1	FRB	2
89	Moody's Seasoned Aaa Corporate Bond Yield	AAA	6	Moody's Daily Corporate Bond Yield Averages	February 1	Moody's	2
90	Moody's Seasoned Baa Corporate Bond Yield	BAA	6	Moody's Daily Corporate Bond Yield Averages	February 1	Moody's	2
91	3-Month Commercial Paper Minus FEDFUNDS	CPFF	6	Interest Rate Spreads	February 1	FRED	1
92	3-Month Treasury C Minus FEDFUNDS	TB3SMFFM	6	Interest Rate Spreads	February 1	FRED	1
93	6-Month Treasury C Minus FEDFUNDS	TB6SMFFM	6	Interest Rate Spreads	February 1	FRED	1
94	1-Year Treasury C Minus FEDFUNDS	T1YFFM	6	Interest Rate Spreads	February 1	FRED	1
95	5-Year Treasury C Minus FEDFUNDS	T5YFFM	6	Interest Rate Spreads	February 1	FRED	1
96	10-Year Treasury C Minus FEDFUNDS	T10YFFM	6	Interest Rate Spreads	February 1	FRED	1
97	Moody's Aaa Corporate Bond Minus FEDFUNDS	AAAFFM	6	Interest Rate Spreads	February 1	FRED	1
98	Moody's Baa Corporate Bond Minus FEDFUNDS	BAAFFM	6	Interest Rate Spreads	February 1	FRED	1
99	Trade Weighted U.S. Dollar Index	TWEXAFEGSMTHx	6	H.10 Foreign Exchange Rates	February 1	FRB	5
100	Switzerland / U.S. Foreign Exchange Rate	EXSZUSx	6	H.10 Foreign Exchange Rates	February 1	FRB	5
101	Japan / U.S. Foreign Exchange Rate	EXJPUSx	6	H.10 Foreign Exchange Rates	February 1	FRB	5
102	U.S. / U.K. Foreign Exchange Rate	EXUSUKx	6	H.10 Foreign Exchange Rates	February 1	FRB	5
103	Canada / U.S. Foreign Exchange Rate	EXCAUSx	6	H.10 Foreign Exchange Rates	February 1	FRB	5
104	PPI: Finished Goods	WPSFD49207	7	Producer Price Index	February 17	BLS	6
105	PPI: Finished Consumer Goods	WPSFD49502	7	Producer Price Index	February 17	BLS	6
106	PPI: Intermediate Materials	WPSID61	7	Producer Price Index	February 17	BLS	6
107	PPI: Crude Materials	WPSID62	7	Producer Price Index	February 17	BLS	6
108	Crude Oil, spliced WTI and Cushing	OILPRICEx	7	Producer Price Index	February 17	BLS	6
109	PPI: Metals and metal products:	PPICMM	7	Producer Price Index	February 17	BLS	6
110	CPI : All Items	CPIAUCSL	7	Consumer Price Index	February 10	BLS	6
111	CPI : Apparel	CPIAPPSL	7	Consumer Price Index	February 10	BLS	6
112	CPI : Transportation	CPITRNSL	7	Consumer Price Index	February 10	BLS	6
113	CPI : Medical Care	CPIMEDSL	7	Consumer Price Index	February 10	BLS	6
114	CPI : Commodities	CUSR0000SAC	7	Consumer Price Index	February 10	BLS	6
115	CPI : Durables	CUUR0000SAD	7	Consumer Price Index	February 10	BLS	6
116	CPI : Services	CUSR0000SAS	7	Consumer Price Index	February 10	BLS	6
117	CPI : All Items Less Food	CPIULFSL	7	Consumer Price Index	February 10	BLS	6
118	CPI : All items less shelter	CUUR0000SA0L2	7	Consumer Price Index	February 10	BLS	6
119	CPI : All items less medical care	CUSR0000SA0L5	7	Consumer Price Index	February 10	BLS	6
120	Personal Cons. Expend.: Chain Index	PCEPI	7	Personal Income and Outlays	February 26	BEA	6
121	Personal Cons. Exp: Durable goods	PCEDG	7	Personal Income and Outlays	February 26	BEA	6
122	Personal Cons. Exp: Nondurable goods	PCEND	7	Personal Income and Outlays	February 26	BEA	6
123	Personal Cons. Exp: Services	PCES	7	Personal Income and Outlays	February 26	BEA	6
124	S&P's Common Stock Price Index: Composite	S&P 500	8	Standard & Poor	February 1	Dow Jones	5
125	S&P's Common Stock Price Index: Industrials	S&P: indust	8	Standard & Poor	February 1	Dow Jones	5
126	S&P's Composite Common Stock: Dividend Yield	S&P div yield	8	Standard & Poor	February 1	Dow Jones	2
127	S&P's Composite Common Stock: Price-Earnings Ratio	S&P PE ratio	8	Standard & Poor	February 1	Dow Jones	5
128	VXO	VXOCLSx	8	CBOE Market Statistics	February 1	CBOE	1

Note: The descriptions, groups, FRED and transformation codes are obtained from McCracken (2021). The groups are: (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 and (8) Stock market. The releases are from: US Bureau of Economic Analysis (BEA), US Board of Governors of the Federal Reserve System (FRB), US Bureau of Labor Statistics (BLS), US Employment and Training Administration (ETA), US Census Bureau (USCB), University of Michigan (UMich) and Federal Reserve Economic Data (FRED).

A.2 Hyperparameters

Random Forest

Table 7: Hyperparameters RF

Parameter	Description	Range	Optimized value
n_estimators	number of trees in forest	1 - 1000	46
max_depth	max. depth in each tree	1 - None	9
max_features	features considered each split	Auto/Sqrt	Sqrt
bootstrap	for sampling data points	True/False	False
min_samples_split	min. data points for new node	1 - 20	2
min_samples_leaf	min. data points per leaf node	2 - 20	2

n_estimators determines the amount of trees equal to 46. This means that the optimal prediction will be based on the average of these 46 trees' predictions. *max_depth* limits the maximum depth of each tree to 9, which is the maximum amount of nodes between the first to a leaf node. *max_features* represents the size of the random subset of features, which in this case equals $\sqrt{N}$. This ensures additional randomness and solves the problem of getting stuck in local optima. When training, each tree learns from a random sample of the dataset, and not using bootstrapping means that the samples are drawn without replacement, such that the same tree does not see the same sample multiple times. *min_samples_split* represents the minimum number of samples required to split a node. By increasing this number, the tree gets more constrained. *min_samples_leaf* is of a similar fashion, but then for leaf nodes.

Extreme Gradient Boosting

Table 8: Hyperparameters XGBoost

	Description	Range	Optimized value
num_boost_round	number of learning trees	1 - 10000	420
max_depth	max. depth in each tree	None - 100	None
subsample	fraction of data per tree (rows)	0 - 1	0.2
colsample_bytree	fraction of data per tree (cols)	0 - 1	0.2
ETA	learning rate	0 - 1	0.03
min_child_weight	min. data points for new node	1 - 20	4

The amount of serial trees *num_boost_round* equals 420. *subsample* presents the fraction of training data used as a random sample, and *colsample_bytree* is the subsample ratio of columns used when constructing each tree. Subsampling occurs once for every tree constructed. *ETA* defines the learning rate. After each boosting step, the tree obtains weights for each feature. The learning rate shrinks the new weights to make the boosting

process a bit more conservative. *min_child_weight* gives the minimum amount of data points necessary for a new split.

Long Short-Term Memory neural network

Table 9: Hyperparameters LSTM

	Description	Range	Optimized value
n_timesteps	amount of lags	1 - 50	1
batchsize	data points per training round	1 - 500	100
epochs	number of training episodes	1 - 1000	100
n_neurons	number of hidden neurons	10 - 100	47
n_layers	number of hidden layers	1 - 5	2
dropout	fraction of nodes to dropout	0 - 1	0.1

n_timesteps refers to the memory of the network, indicating the number of periods of past regressors and past target values on which the estimation will depend on. The value one indicates that the model does not depend much on previous values of the regressors and only considers values from the previous period. *batchsize* presents the number of data samples that the LSTM network get fed with per round. The amount of times a model is trained is indicated by *epochs*. The model contains *n_neurons* per layer, and the amount of layers is presented by *n_layers*. Hence, the architecture of the models consists of one input layer, two hidden layers consisting of 47 neurons per layer, and one output layer. *dropout* refers to the fraction of nodes dropped per training epoch, introducing noise on which the network can improve.

Support Vector Regression

The $Kernel$ that is used for regression is the Polynomial kernel. *degree* equals 4 and the $coef\_0$ equals 0.2. Both values are specific for the Polynomial kernel. $coef\_0$ scales the data. Due to the *degree*, small data points are transformed to much smaller data points, while larger data points transform to almost infinity. $coef\_0$ makes sure that this distinction does not get too dominant. ϵ scales the ϵ-tube. C is the regularization

Table 10: Hyperparameters SVR

	Description	Range	Optimized value
$Kernel$	Kernel function	Poly/RBF	Poly
degree	Polynomial degree (d)	0 - 5	4
$coef\_0$	scales the data (c)	0 - 10	0.04
ϵ	scales the ϵ-tube	0 - 1	0.03
C	regularization parameter	0 - 30	10.2
γ	inverse of radius support vectors	0 - 1	0.01

parameter that penalizes using the L_2-norm, and its function has been explained already in Section 3.2.4. γ represents the inverse of the radius of the samples selected as support vectors, as presented in Table 1.

A.3 Feature Importance

Random Forest

Table 11: RF Relative Feature Importance

	Feature	Importance (%)	St. dev (%)		Feature	Importance (%)	St. dev (%)		Feature	Importance (%)	St. dev (%)
1	PAYEMS	6.55	1.30	42	TWEXAFEGSMTHx	0.67	0.05	83	UEMP5TO14	0.35	0.02
2	DMANEMP	4.52	0.63	43	UNRATE	0.65	0.04	84	CUSR0000SAS	0.35	0.02
3	MANEMP	3.00	0.61	44	IPDCONGD	0.63	0.04	85	UEMP15OV	0.34	0.02
4	HOUST	2.84	0.54	45	CONSPI	0.61	0.04	86	UEMP27OV	0.34	0.02
5	USFIRE	2.71	0.31	46	BUSLOANS	0.61	0.04	87	T5YFFM	0.34	0.02
6	HOUSTNE	2.70	0.32	47	ANDENOx	0.57	0.04	88	DSERRG3M086SBEA	0.33	0.02
7	USTPU	2.56	0.26	48	CES1021000001	0.54	0.04	89	WPSFD49502	0.33	0.02
8	TB6MS	2.20	0.21	49	HWI	0.53	0.04	90	AAAFFM	0.33	0.02
9	AMDMUOx	2.00	0.21	50	HOUSTS	0.53	0.04	91	UEMP15T26	0.33	0.02
10	INDPRO	1.85	0.20	51	AAA	0.53	0.04	92	CPIAPPSL	0.32	0.02
11	USTRADE	1.82	0.18	52	S&P:.indust	0.51	0.04	93	EXJPUSx	0.32	0.02
12	SRVPRD	1.80	0.16	53	DPCERA3M086SBEA	0.51	0.03	94	AWOTMAN	0.32	0.02
13	HOUSTMW	1.77	0.16	54	USCONS	0.50	0.03	95	DDURRG3M086SBEA	0.31	0.02
14	IPMAT	1.76	0.15	55	S&P.500	0.49	0.03	96	UEMPMEAN	0.31	0.02
15	USGOOD	1.74	0.14	56	MZMSL	0.48	0.03	97	T10YFFM	0.31	0.02
16	BUSINVx	1.60	0.13	57	HOUSTW	0.47	0.03	98	TOTRESNS	0.31	0.02
17	IPDMAT	1.59	0.13	58	TB6SMFFM	0.47	0.03	99	NONBORRES	0.30	0.02
18	GS1	1.55	0.13	59	EXUSUKx	0.47	0.02	100	EXCAUSx	0.30	0.02
19	IPBUSEQ	1.47	0.12	60	CUSR0000SAD	0.47	0.02	101	CLF16OV	0.30	0.02
20	CP3Mx	1.43	0.11	61	ACOGNO	0.46	0.02	102	DNDGRG3M086SBEA	0.30	0.02
21	IPMANSICS	1.33	0.10	62	M1SL	0.44	0.02	103	UMCSENTx	0.30	0.02
22	CLAIMSx	1.19	0.09	63	T1YFFM	0.44	0.02	104	CMRMTSPLx	0.30	0.02
23	RETAILx	1.18	0.08	64	CE16OV	0.43	0.02	105	NONREVSL	0.29	0.02
24	USWTRADE	1.23	0.08	65	CES2000000008	0.42	0.02	106	CPIAUCSL	0.29	0.02
25	CES0600000007	1.18	0.08	66	BOGMBASE	0.42	0.02	107	CUSR0000SA0L2	0.29	0.02
26	TB3MS	1.18	0.07	67	M2REAL	0.42	0.02	108	IPB51222S	0.28	0.02
27	HWIURATIO	1.16	0.07	68	UEMPLT5	0.42	0.02	109	CPIULFSL	0.27	0.02
28	CUMFNS	1.15	0.07	69	WPSFD49207	0.41	0.02	110	WPSID61	0.26	0.02
29	GS5	1.15	0.06	70	ISRATIOx	0.41	0.02	111	WPSID62	0.25	0.02
30	FEDFUNDS	1.13	0.06	71	COMPAPFFx	0.41	0.02	112	DTCOLNVHFNM	0.24	0.02
31	PERMITMW	1.05	0.06	72	GS10	0.41	0.02	113	IPNCONGD	0.24	0.02
32	IPFPNSS	1.05	0.06	73	IPFUELS	0.41	0.02	114	PCEPI	0.24	0.02
33	AWHMAN	1.04	0.06	74	IPCONGD	0.40	0.02	115	DTCTHFNM	0.22	0.02
34	S&P.div.yield	1.01	0.06	75	S&P.PE.ratio	0.40	0.02	116	REALLN	0.22	0.02
35	NDMANEMP	0.97	0.06	76	TB3SMFFM	0.39	0.02	117	OILPRICEx	0.22	0.02
36	USGOVT	0.97	0.05	77	EXSZUSx	0.38	0.02	118	CPITRNSL	0.21	0.02
37	PERMITNE	0.90	0.05	78	BAA	0.37	0.02	119	PERMITS	0.21	0.02
38	PERMIT	0.83	0.04	79	CPIMEDSL	0.37	0.02	120	CUSR0000SAC	0.21	0.02
39	INVEST	0.73	0.04	80	CES0600000008	0.37	0.02	121	M2SL	0.20	0.02
40	IPNMAT	0.70	0.04	81	W875RX1	0.36	0.02	122	PPICMM	0.20	0.02
41	IPFINAL	0.69	0.04	82	RPI	0.36	0.02	123	CUSR0000SA0L5	0.18	0.02

Extreme Gradient Boosting

Table 12: XGBoost Relative Feature Importance

	Feature	Importance (%)	St. dev (%)		Feature	Importance (%)	St. dev (%)		Feature	Importance (%)	St. dev (%)
1	MANEMP	3.59	0.97	42	EXSZUSx	0.77	0.04	83	CUSR0000SA0L5	0.47	0.03
2	DMANEMP	3.53	0.60	43	IPNMAT	0.75	0.04	84	BOGMBASE	0.45	0.03
3	HOUSTMW	2.87	0.40	44	S&P.div.yield	0.75	0.04	85	PCEPI	0.45	0.03
4	BUSINVx	2.60	0.33	45	IPFUELS	0.74	0.04	86	CPIMEDSL	0.44	0.03
5	PAYEMS	2.43	0.32	46	INVEST	0.73	0.04	87	CPIULFSL	0.44	0.03
6	TB6MS	2.39	0.27	47	S&P.500	0.73	0.04	88	T5YFFM	0.44	0.03
7	IPDMAT	2.35	0.26	48	COMPAPFFx	0.71	0.04	89	UEMP15T26	0.44	0.03
8	SRVPRD	2.24	0.22	49	PERMITMW	0.70	0.04	90	HOUSTS	0.43	0.03
9	CP3Mx	1.94	0.19	50	TB3SMFFM	0.68	0.04	91	IPCONGD	0.43	0.02
10	IPMAT	1.93	0.16	51	UEMP27OV	0.68	0.04	92	NONBORRES	0.42	0.02
11	USGOOD	1.70	0.16	52	M1SL	0.68	0.04	93	M2SL	0.42	0.02
12	INDPRO	1.58	0.16	53	ACOGNO	0.66	0.04	94	BAA	0.41	0.02
13	AMDMUOx	1.57	0.15	54	AAA	0.65	0.04	95	DSERRG3M086SBEA	0.41	0.02
14	RETAILx	1.42	0.14	55	EXUSUKx	0.65	0.04	96	WPSFD49207	0.40	0.02
15	CONSPI	1.40	0.13	56	HOUSTNE	0.65	0.04	97	NONREVSL	0.40	0.02
16	USFIRE	1.36	0.13	57	CES1021000001	0.63	0.04	98	UEMP15OV	0.39	0.03
17	CMRMTSPLx	1.34	0.13	58	IPMANSICS	0.63	0.04	99	PERMITNE	0.38	0.03
18	PERMIT	1.26	0.12	59	UMCSENTx	0.63	0.04	100	IPNCONGD	0.38	0.03
19	CLAIMSx	1.14	0.11	60	AWOTMAN	0.60	0.04	101	WPSFD49502	0.38	0.02
20	HWIURATIO	1.14	0.11	61	IPB51222S	0.60	0.03	102	IPFINAL	0.38	0.02
21	AWHMAN	1.12	0.11	62	UNRATE	0.60	0.03	103	EXCAUSx	0.37	0.02
22	ISRATIOx	1.12	0.11	63	W875RX1	0.59	0.03	104	WPSID61	0.37	0.02
23	REALLN	1.07	0.11	64	CPIAPPSL	0.59	0.03	105	DTCOLNVHFNM	0.37	0.02
24	IPDCONGD	1.06	0.10	65	PERMITS	0.59	0.03	106	DNDGRG3M086SBEA	0.37	0.02
25	USWTRADE	1.06	0.10	66	CES0600000007	0.59	0.03	107	AAAFFM	0.36	0.02
26	USTRADE	1.05	0.09	67	MZMSL	0.59	0.03	108	CES0600000008	0.36	0.02
27	GS5	1.03	0.08	68	CLF16OV	0.59	0.03	109	HWI	0.36	0.02
28	BUSLOANS	1.01	0.07	69	ANDENOx	0.57	0.03	110	CPIAUCSL	0.34	0.02
29	USGOVT	1.01	0.07	70	T1YFFM	0.57	0.03	111	WPSID62	0.32	0.02
30	DDURRG3M086SBEA	1.00	0.06	71	CES2000000008	0.57	0.03	112	PPICMM	0.32	0.02
31	HOUSTW	0.98	0.06	72	CUSR0000SAS	0.56	0.03	113	CUSR0000SA0L2	0.32	0.02
32	CUSR0000SAD	0.96	0.06	73	USCONS	0.56	0.03	114	GS10	0.31	0.03
33	HOUST	0.96	0.06	74	CE16OV	0.54	0.03	115	TB6SMFFM	0.30	0.03
34	IPBUSEQ	0.95	0.06	75	UEMP5TO14	0.51	0.03	116	TWEXAFEGSMTHx	0.28	0.03
35	RPI	0.91	0.05	76	IPFPNSS	0.50	0.03	117	TOTRESNS	0.28	0.03
36	NDMANEMP	0.86	0.06	77	DTCTHFNM	0.50	0.03	118	OILPRICEx	0.28	0.03
37	DPCERA3M086SBEA	0.85	0.06	78	TB3MS	0.50	0.03	119	EXJPUSx	0.27	0.03
38	FEDFUNDS	0.83	0.05	79	M2REAL	0.50	0.03	120	CPITRNSL	0.26	0.03
39	UEMPLT5	0.80	0.05	80	S&P.PE.ratio	0.49	0.03	121	UEMPMEAN	0.25	0.02
40	USTPU	0.79	0.05	81	S&P:.indust	0.48	0.03	122	CUSR0000SAC	0.25	0.03
41	GS1	0.78	0.04	82	CUMFNS	0.47	0.03	123	T10YFFM	0.23	0.03

Long Short-Term Memory neural network

Table 13: LSTM Relative Feature Importance

	Feature	Importance (%)	Cum (%)		Feature	Importance (%)	Cum (%)		Feature	Importance (%)	Cum (%)
1	AWHMAN(t-1)	1.55	0.18	84	UEMP5TO14(t-1)	0.40	0.02	167	EXCAUSx(t-1)	0.23	0.01
2	AMDMUOx	1.50	0.15	85	USGOOD(t-1)	0.39	0.02	168	IPDMAT(t-1)	0.23	0.01
3	SRVPRD	1.44	0.12	86	USTRADE(t-1)	0.39	0.02	169	EXUSUKx(t-1)	0.23	0.01
4	UEMPLT5	1.43	0.11	87	PAYEMS(t-1)	0.39	0.02	170	CPIULFSL(t-1)	0.23	0.01
5	BUSINVx	1.41	0.10	88	T10YFFM(t-1)	0.38	0.01	171	M2REAL(t-1)	0.23	0.01
6	CES0600000007(t-1)	1.40	0.09	89	ACOGNO	0.38	0.01	172	NDMANEMP(t-1)	0.22	0.01
7	BUSLOANS(t-1)	1.31	0.08	90	AWOTMAN	0.38	0.01	173	UEMP15OV(t-1)	0.22	0.01
8	INVEST	1.25	0.07	91	IPDMAT	0.37	0.01	174	CP3Mx(t-1)	0.22	0.01
9	BUSLOANS	1.23	0.07	92	COMPAPFFx	0.36	0.01	175	MZMSL	0.22	0.01
10	PAYEMS	1.17	0.07	93	HWIURATIO(t-1)	0.36	0.01	176	BAA(t-1)	0.22	0.01
11	CUSR0000SAD	1.15	0.07	94	UEMP27OV	0.36	0.01	177	OILPRICEx	0.22	0.01
12	IPNCONGD(t-1)	1.04	0.06	95	PERMITNE	0.36	0.01	178	GS1	0.21	0.01
13	IPMAT	0.99	0.06	96	S&P:.indust(t-1)	0.36	0.01	179	CUSR0000SA0L5	0.21	0.01
14	PERMITMW	0.98	0.06	97	PCEPI	0.36	0.01	180	UEMP15T26(t-1)	0.21	0.01
15	HOUSTMW	0.93	0.05	98	GS10	0.35	0.01	181	PCEPI(t-1)	0.20	0.01
16	DDURRG3M086SBEA	0.92	0.05	99	GS1(t-1)	0.35	0.01	182	CES0600000008(t-1)	0.20	0.01
17	CONSPI	0.92	0.05	100	DMANEMP	0.35	0.01	183	CPIAUCSL	0.20	0.01
18	AWHMAN	0.89	0.05	101	IPMAT(t-1)	0.35	0.01	184	BOGMBASE	0.20	0.01
19	CONSPI(t-1)	0.88	0.05	102	PERMITMW(t-1)	0.35	0.01	185	DTCTHFNM	0.20	0.01
20	IPCONGD(t-1)	0.87	0.05	103	IPFPNSS	0.34	0.01	186	CPIAUCSL(t-1)	0.20	0.01
21	PERMITS(t-1)	0.87	0.05	104	MANEMP(t-1)	0.34	0.01	187	IPBUSEQ(t-1)	0.20	0.01
22	CUSR0000SAD(t-1)	0.86	0.05	105	CLAIMSx(t-1)	0.34	0.01	188	USGOVT(t-1)	0.20	0.01
23	USFIRE(t-1)	0.83	0.04	106	TWEXAFEGSMTHx(t-1)	0.34	0.01	189	USCONS(t-1)	0.20	0.01
24	UNRATE	0.82	0.04	107	IPCONGD	0.34	0.01	190	CPITRNSL	0.20	0.01
25	IPFUELS	0.80	0.04	108	IPNMAT(t-1)	0.34	0.02	191	AAA(t-1)	0.20	0.01
26	INVEST(t-1)	0.77	0.04	109	DTCOLNVHFNM(t-1)	0.34	0.02	192	OILPRICEx(t-1)	0.20	0.01
27	EXSZUSx	0.77	0.04	110	TB6MS(t-1)	0.34	0.01	193	IPDCONGD(t-1)	0.19	0.01
28		0.76	0.04	111	S&P.PE.ratio	0.33	0.01	194	TB3MS(t-1)	0.19	0.01
29	S&P.div.yield	0.74	0.04	112	IPNMAT	0.33	0.01	195	FEDFUNDS(t-1)	0.19	0.01
30	CE16OV(t-1)	0.74	0.03	113	S&P:.indust	0.33	0.01	196	W875RX1(t-1)	0.19	0.01
31	USFIRE	0.74	0.03	114	TB6SMFFM	0.33	0.02	197	S&P.PE.ratio(t-1)	0.19	0.01
32	IPNCONGD	0.72	0.03	115	T10YFFM	0.33	0.02	198	CUSR0000SA0L2(t-1)	0.19	0.01
33	SRVPRD(t-1)	0.69	0.03	116	CES1021000001	0.33	0.02	199	UEMP5TO14	0.18	0.01
34	HOUST	0.68	0.03	117	EXJPUSx	0.32	0.02	200	CPIMEDSL	0.18	0.01
35	UEMPMEAN	0.68	0.03	118	T1YFFM(t-1)	0.32	0.02	201	WPSFD49207(t-1)	0.18	0.01
36	CLF16OV(t-1)	0.68	0.03	119	WPSFD49207	0.32	0.02	202	UNRATE(t-1)	0.18	0.01
37	UEMPMEAN(t-1)	0.67	0.03	120	HOUSTNE(t-1)	0.32	0.02	203	T1YFFM	0.18	0.01
38	CUSR0000SAS	0.66	0.03	121	GS5	0.32	0.02	204	RETAILx(t-1)	0.18	0.01
39	UEMP15T26	0.65	0.03	122	UMCSENTx	0.32	0.02	205	RPI(t-1)	0.18	0.01
40	INDPRO	0.65	0.03	123	HOUSTMW(t-1)	0.31	0.02	206	GS5(t-1)	0.18	0.01
41	AMDMUOx(t-1)	0.64	0.03	124	CUSR0000SA0L2	0.31	0.02	207	AAA	0.18	0.01
42	USGOVT	0.64	0.03	125	HOUSTS(t-1)	0.31	0.01	208	DSERRG3M086SBEA	0.17	0.01
43	CMRMTSPLx(t-1)	0.64	0.03	126	CUMFNS	0.31	0.01	209	IPBUSEQ	0.17	0.01
44	HOUSTW	0.63	0.03	127	IPFINAL	0.31	0.01	210	DDURRG3M086SBEA(t-1)	0.17	0.01
45	AAAFFM	0.63	0.03	128	CUMFNS(t-1)	0.30	0.01	211	COMPAPFFx(t-1)	0.17	0.01
46	USWTRADE	0.62	0.03	129	TB6MS	0.30	0.01	212	WPSFD49502(t-1)	0.17	0.01
47	CPIAPPSL	0.61	0.03	130	AAAFFM(t-1)	0.30	0.01	213	RPI	0.17	0.01
48	ISRATIOx	0.61	0.03	131	INDPRO(t-1)	0.30	0.01	214	CP3Mx	0.17	0.01
49	HWIURATIO	0.59	0.03	132	BUSINVx(t-1)	0.29	0.01	215	CUSR0000SAC(t-1)	0.17	0.01
50	USTRADE	0.59	0.03	133	HOUST(t-1)	0.29	0.01	216	CE16OV	0.17	0.01
51	PERMITNE(t-1)	0.58	0.03	134	IPMANSICS	0.29	0.01	217	W875RX1	0.16	0.01
52	CES0600000007	0.58	0.02	135	HWI	0.29	0.01	218	FEDFUNDS	0.16	0.01
53	USCONS	0.57	0.02	136	CES2000000008	0.29	0.01	219	DPCERA3M086SBEA(t-1)	0.16	0.01
54	HOUSTS	0.57	0.02	137	CMRMTSPLx	0.29	0.01	220	TB3MS	0.16	0.01
55	DPCERA3M086SBEA	0.57	0.02	138	ANDENOx(t-1)	0.28	0.01	221	EXUSUKx	0.16	0.01
56	UEMP15OV	0.54	0.02	139	S&P.500	0.28	0.01	222	CUSR0000SAC	0.16	0.01
57	RETAILx	0.53	0.02	140	NDMANEMP	0.28	0.02	223	EXCAUSx	0.16	0.01
58	M2REAL	0.53	0.02	141	IPDCONGD	0.28	0.01	224	DSERRG3M086SBEA(t-1)	0.16	0.01
59	HOUSTW(t-1)	0.53	0.02	142	CES2000000008(t-1)	0.28	0.01	225	AWOTMAN(t-1)	0.16	0.01
60	CUSR0000SAS(t-1)	0.52	0.02	143	CUSR0000SA0L5(t-1)	0.28	0.01	226	WPSFD49502	0.16	0.01
61	EXSZUSx(t-1)	0.52	0.02	144	T5YFFM	0.28	0.01	227	WPSID62(t-1)	0.15	0.01
62	TWEXAFEGSMTHx	0.52	0.02	145	IPFPNSS(t-1)	0.28	0.01	228	WPSID61(t-1)	0.15	0.01
63	IPFUELS(t-1)	0.52	0.02	146	USWTRADE(t-1)	0.27	0.01	229	DNDGRG3M086SBEA(t-1)	0.15	0.01
64	USTPU	0.49	0.02	147	IPMANSICS(t-1)	0.27	0.01	230	MZMSL(t-1)	0.15	0.01
65	UEMPLT5(t-1)	0.49	0.02	148	CES0600000008	0.27	0.01	231	PPICMM	0.14	0.01
66	NONREVSL	0.49	0.02	149	S&P.500(t-1)	0.27	0.01	232	M1SL(t-1)	0.14	0.01
67	USGOOD	0.48	0.02	150	M1SL	0.26	0.01	233	CPITRNSL(t-1)	0.14	0.01
68	IPFINAL(t-1)	0.48	0.02	151	TB3SMFFM(t-1)	0.26	0.01	234	WPSID61	0.14	0.01
69	DTCTHFNM(t-1)	0.48	0.02	152	CLF16OV	0.26	0.01	235	REALLN(t-1)	0.14	0.01
70	PERMIT	0.47	0.02	153	UMCSENTx(t-1)	0.26	0.01	236	BAA	0.13	0.01
71	S&P.div.yield(t-1)	0.46	0.02	154	ANDENOx	0.26	0.01	237	NONREVSL(t-1)	0.13	0.01
72	HOUSTNE	0.46	0.02	155	TB3SMFFM	0.26	0.01	238	CES1021000001(t-1)	0.13	0.01
73	PERMIT(t-1)	0.45	0.02	156	USTPU(t-1)	0.25	0.01	239	DTCOLNVHFNM	0.12	0.01
74	EXJPUSx(t-1)	0.44	0.02	157	IPB51222S(t-1)	0.25	0.01	240	WPSID62	0.12	0.01
75	PERMITS	0.44	0.02	158	UEMP27OV(t-1)	0.25	0.01	241	BOGMBASE(t-1)	0.12	0.01
76	ACOGNO(t-1)	0.44	0.02	159	CLAIMSx	0.25	0.01	242	NONBORRES	0.12	0.01
77	CPIMEDSL(t-1)	0.43	0.02	160	DNDGRG3M086SBEA	0.24	0.01	243	M2SL(t-1)	0.12	0.01
78	T5YFFM(t-1)	0.42	0.02	161	PPICMM(t-1)	0.24	0.01	244	M2SL	0.10	0.01
79	REALLN	0.42	0.02	162	ISRATIOx(t-1)	0.24	0.01	245	NONBORRES(t-1)	0.09	0.01
80	DMANEMP(t-1)	0.42	0.02	163	HWI(t-1)	0.24	0.01	246	TOTRESNS	0.07	0.01
81	MANEMP	0.42	0.02	164	GS10(t-1)	0.24	0.01	247	TOTRESNS(t-1)	0.07	0.01
82	CPIAPPSL(t-1)	0.40	0.02	165	TB6SMFFM(t-1)	0.24	0.01				
83	CPIULFSL	0.40	0.02	166	IPB51222S	0.23	0.01				

Support Vector Regression

Table 14: SVR Relative Feature Importance

	Feature	Importance (%)	St. dev (%)		Feature	Importance (%)	St. dev (%)		Feature	Importance (%)	St. dev (%)
1	BUSINVx	5.38	0.17	42	INDPRO	0.98	0.04	83	CUSR0000SA0L5	0.47	0.03
2	USFIRE	3.81	0.27	43	HWIURATIO	0.97	0.04	84	BOGMBASE	0.45	0.03
3	AWHMAN	2.96	0.39	44	EXUSUKx	0.94	0.04	85	PCEPI	0.45	0.03
4	CUSR0000SAD	2.39	0.25	45	HOUSTW	0.92	0.04	86	CPIMEDSL	0.44	0.03
5	AWOTMAN	2.37	0.14	46	IPNMAT	0.92	0.04	87	CPIULFSL	0.44	0.03
6	CP3Mx	2.02	0.14	47	USGOOD	0.91	0.05	88	T5YFFM	0.44	0.03
7	AMDMUOx	2.01	0.13	48	UMCSENTx	0.88	0.05	89	UEMP15T26	0.44	0.03
8	DTCOLNVHFNM	1.99	0.11	49	TWEXAFEGSMTHx	0.86	0.05	90	HOUSTS	0.43	0.03
9	IPMAT	1.98	0.11	50	GS10	0.80	0.05	91	IPCONGD	0.43	0.02
10	USGOVT	1.89	0.11	51	COMPAPFFx	0.79	0.05	92	NONBORRES	0.42	0.02
11	USTRADE	1.85	0.11	52	S&P.PE.ratio	0.77	0.05	93	M2SL	0.42	0.02
12	DPCERA3M086SBEA	1.85	0.11	53	ANDENOx	0.68	0.05	94	BAA	0.41	0.02
13	UEMPLT5	1.79	0.10	54	IPFUELS	0.66	0.05	95	DSERRG3M086SBEA	0.41	0.02
14	AAAFFM	1.76	0.10	55	PCEPI	0.64	0.05	96	WPSFD49207	0.40	0.02
15	HOUSTMW	1.76	0.11	56*	ACOGNO	0.61	0.04	97	NONREVSL	0.40	0.02
16	EXJPUSx	1.73	0.10	57	NDMANEMP	0.61	0.04	98	UEMP15OV	0.39	0.03
17	IPNCONGD	1.73	0.07	58	CMRMTSPLx	0.59	0.04	99	PERMITNE	0.38	0.03
18	DDURRG3M086SBEA	1.70	0.07	59	UEMPMEAN	0.58	0.03	100	IPNCONGD	0.38	0.03
19	T1YFFM	1.57	0.06	60	CUSR0000SA0L2	0.54	0.03	101	WPSFD49502	0.38	0.02
20	REALLN	1.51	0.05	61	CUSR0000SA0L5	0.52	0.03	102	IPFINAL	0.38	0.02
21	INVEST	1.42	0.05	62	UNRATE	0.52	0.03	103	EXCAUSx	0.37	0.02
22	IPDCONGD	1.36	0.06	63	TB6SMFFM	0.51	0.03	104	WPSID61	0.37	0.02
23	S&P.div.yield	1.30	0.05	64	AAA	0.51	0.03	105	DTCOLNVHFNM	0.37	0.02
24	CES0600000007	1.30	0.05	65	CPIULFSL	0.49	0.03	106	DNDGRG3M086SBEA	0.37	0.02
25	BUSLOANS	1.29	0.05	66	CE16OV	0.48	0.03	107	AAAFFM	0.36	0.02
26	DMANEMP	1.28	0.04	67	CES1021000001	0.48	0.03	108	CES0600000008	0.36	0.02
27	RETAILx	1.27	0.04	68	WPSID61	0.48	0.03	109	HWI	0.36	0.02
28	CES2000000008	1.27	0.04	69	MZMSL	0.46	0.02	110	CPIAUCSL	0.34	0.02
29	PAYEMS	1.25	0.03	70	PERMITNE	0.42	0.02	111	WPSID62	0.32	0.02
30	HOUST	1.24	0.03	71	HWI	0.41	0.02	112	PPICMM	0.32	0.02
31	CONSPI	1.23	0.04	72	UEMP15OV	0.40	0.02	113	CUSR0000SA0L2	0.32	0.02
32	S&P:.indust	1.21	0.04	73	UEMP5TO14	0.40	0.02	114	GS10	0.31	0.03
33	MANEMP	1.19	0.04	74	CPIAPPSL	0.39	0.02	115	TB6SMFFM	0.30	0.03
34	USCONS	1.16	0.05	75	DTCTHFNM	0.39	0.02	116	TWEXAFEGSMTHx	0.28	0.03
35	S&P.500	1.09	0.05	76	M2REAL	0.38	0.02	117	TOTRESNS	0.28	0.03
36	CPIAUCSL	1.08	0.05	77	IPFINAL	0.37	0.02	118	OILPRICEx	0.28	0.03
37	HOUSTS	1.07	0.05	78	NONBORRES	0.37	0.02	119	EXJPUSx	0.27	0.03
38	SRVPRD	1.06	0.05	79	FEDFUNDS	0.37	0.02	120	CPITRNSL	0.26	0.03
39	TB3SMFFM	1.04	0.05	80	T5YFFM	0.35	0.02	121	UEMPMEAN	0.25	0.02
40	DNDGRG3M086SBEA	1.03	0.05	81	TB6MS	0.34	0.02	122	CUSR0000SAC	0.25	0.03
41	PERMITMW	0.99	0.05	82	IPB51222S	0.32	0.02	123	T10YFFM	0.23	0.03