Stacking Regression for Time-Series, with an Application to Forecasting Quarterly US GDP Growth

Erkal Ersoy, Haoyang Li, Mark E. Schaffer(✉), and Tibor Szendrei

Edinburgh Business School, Heriot-Watt University, Edinburgh, UK
m.e.schaffer@hw.ac.uk

Abstract. Machine learning methods are being increasingly adopted in economic forecasting. Many learners are available, and a practical issue now presents itself: which one(s) to use? The answer we suggest is 'stacking regression' (Wolpert, 1992), an ensemble method for combining predictions of different learners. We show how to use stacking regression in the time series setting. Macroeconomic and financial time series data present their own challenges to forecasting (extreme values, regime changes, etc.), and this presents challenges to stacking as well. Our findings suggest that using absolute deviations for scoring the base learners performs well compared to stacking on mean squared error. We illustrate this with a Monte Carlo exercise and an empirical application: forecasting US GDP growth around the Covid-19 pandemic.

Keywords: Stacking regression · machine learning · forecasting · robust statistics

1 Introduction

Machine learning methods are being imported in applied econometrics in a variety of settings. These methods provide powerful tools for prediction and forecasting. This poses a new problem for applied econometricians: too much choice. There are many machine learning estimators available. Which learner should they use? Model selection methods typically select a model and then conduct inference based on the assumption that the model actually generated the data. Their inference can only be trusted if the 'best' model selected happens to be a close approximation to the true data generating process. In practice, however, it is more likely that the best model captures some aspects of the truth, while other models capture other aspects. By conditioning only on the best model, model selection methods ignore all the evidence contained in the alternatives and can lead to misleading results in the sense of being either systematically

Invited paper for the Sixth International Econometrics Conference of Vietnam, Banking University of Ho Chi Minh City, Vietnam, 9–11 January 2023. All errors are our own.

N. Ngoc Thach et al. (Eds.): *Optimal Transport Statistics for Economics and Related Topics*, SSDC 483, pp. 131–149, 2024.
https://doi.org/10.1007/978-3-031-35763-3_7

wrong or overfitting the data. Whenever quantities that are not model-specific are of interest, therefore, it makes more sense to create a mixture of the different models rather than select a 'best' model (Steel, 2020). To this end we consider 'stacking regression' (Wolpert, 1992), an ensemble method for combining predictions of different learners, as a way to mix information contained in the different models. Stacking regression is, in effect, a generalization of cross-validation.

The dependent data setting has its own peculiarities: financial and macroeconomic data are typically serially correlated, extreme values are an issue, etc. Tuning methods have to avoid data leakage (letting information from the future leak into the model training process). This applies to stacking regression on two fronts: (1) the training of the different learners needs to avoid data leakage from the future into the individual learners; (2) the stacking procedure that combines the predictions of these learners has to avoid data leakage. We outline in this paper how this is done in practice.

Lastly, this paper examines alternative scoring approaches, i.e., how in practice weights are assigned to the different learners to obtain the stacked forecast. The most common scoring approach in both cross-validation and stacking is mean-squared prediction error (MSPE). In the time-series setting, however, the mean absolute prediction error (MAPE) is an attractive alternative. Macro and financial time series commonly present researchers with problems such as extreme values and regime changes, and working with absolute rather than squared deviations has been shown in other contexts to add robustness to the analysis. Stacking on mean squared has been used in some empirical applications (Pavlyshenko, 2018; Ribeiro et al., 2019; Ribeiro and dos Santos Coelho, 2020; da Silva et al., 2020), but we are unaware of any systematic exploration of stacking on median. We examine whether the robustness of the median extends to stacking regression. Using Monte Carlo experiments and an empirical application focused on US GDP around the COVID-19 pandemic, we find that stacking on the regression has attractive features for practitioners.

The paper is organised as follows. We first briefly summarize how cross-validation is done in the time-series setting, and then introduce stacking regression. In our applications we use 5 different 'base learners' that are combined to obtain a stacking forecast, and the next section describes them in brief: lasso, ridge, elastic net, support vector machine, and random forest. The next section presents the results of a Monte Carlo exercise that shows that stacking regression using MAPE for scoring compares favourably to stacking regression when MSPE is used for evaluation. We then present the results of a practical application – forecasting US GDP growth – and again show that stacking regression using MAPE for scoring performs well. The final section concludes.

2 Time Series Cross-Validation

Cross-validation (CV) allows researchers to choose a model specification – typically, a tuning parameter (e.g., the penalization parameter for lasso) – based on predictive performance, and it is often used to avoid modelling difficulties

like overfitting and selection bias. As part of the process, the dataset is split into two sections: the 'training' sample, which is used to fit the model, and the 'validation' or 'holdout' sample, which is used to assess predictive performance. Mean squared prediction error, MSPE, is a common choice of metric for this.

This resampling technique iteratively trains and tests a model using different portions of the data to tune the parameter of the base learner. The goal is to find the model that has the best out-of-sample predictive performance and can generalize to other samples from the same population. Since stacking can be regarded as a logical extension of cross-validation, we briefly go through CV before formally introducing the implementation of stacking.

In the case of independent data, 'K-fold' cross-validation is the most commonly used approach. In K-fold cross-validation, the data are randomly split into K portions or 'folds'. At each iteration, one fold is treated as the validation set while the remaining $K-1$ folds are treated as the training set to fit the model for some value of the tuning parameter. After each fold is used as the validation set once (and only once), the predictive performance of the model is estimated by averaging the MSPE over all the validation sets. Given a range of values for the tuning parameters, the model with the best predictive performance is selected as the final model (Fig. 1).

The iterative nature of cross-validation makes it computationally intensive: the model needs to be repeatedly estimated and its performance checked across different folds and across a grid of values for the tuning parameter.

Cross-validation with dependent data, i.e., in a time series setting, adds further complications because of the need to ensure that the validation data are independent of the training data. The key issue with K-fold cross-validation in the context of time series prediction is data leakage. When data are dependent, the information from the validation set can leak into the training set, leading to overfitting and hence poor generalization.

With the exception of very specific cases where K-fold cross-validation may be appropriate, researchers should typically use time series cross-validation, a version of 'non-dependent cross validation' (Bergmeir et al., 2018) where cross-validation is set up to account for the nature of the dependence that may see

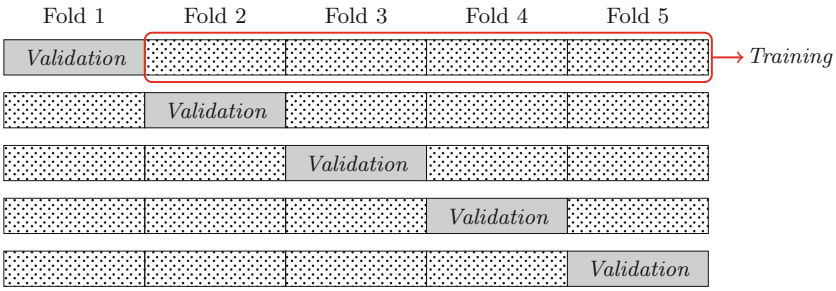


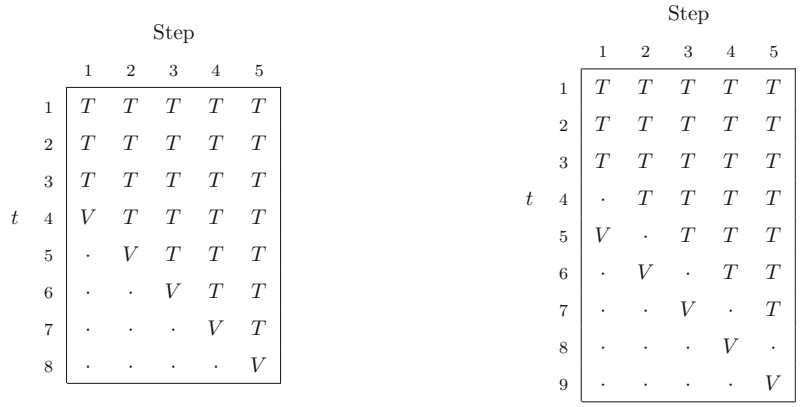
Fig. 1. K-fold cross-validation for cross sectional data. ($K = 5$)

dependent observations omitted from the validation sets. In simple terms, time series cross-validation ensures that the training and validation take place with h-step-ahead forecasts. For example, 1-step-ahead cross-validation (Hyndman and Athanasopoulos, 2018) fits a model on t observations and assesses predictive performance based on the forecast for time $t + 1$.

More generally, consider a series of validation sets, each of which includes one observation at $t + 1$. The corresponding training set of each validation set would then consist of observations through time t, all of which, by definition, will have occurred before $t + 1$. After the predictions at current iteration are made, the validation set moves forward by one and the current observation is added to the training set to form the new training set for the next iteration. Thus, future observations are never used to forecast previous ones; data never leaks into the training process from the future. After going through all the predetermined validation sets, the model with the best predictive performance is selected as the final model. This one-step-ahead expanding window approach is demonstrated in Fig. 2(a).

This process can be generalized for h-step-ahead CV and the training window can be fixed instead of expanding. A rolling window fixes the size of the training set by deleting the most distant observation when a new observation is added, while an expanding window simply adds the new observation to the current training set. Therefore, the rolling window always has a fixed training size predetermined by the researcher and the expanding window includes a growing number of observations in the training set. The rolling window is useful when the series is volatile or the forecasting depends largely on the most recent history, while the expanding window is more appropriate when the series has a stable trend or seasonal pattern.

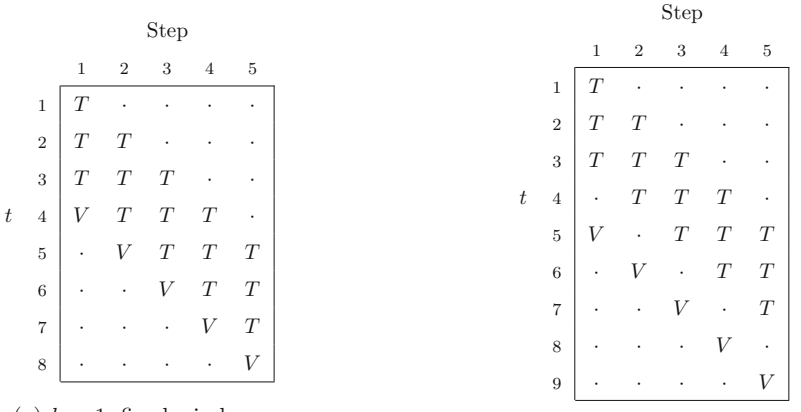
Figure 2 and Fig. 3 show examples of one-step- and two-step-ahead forecasts with expanding and fixed windows, e.g., Fig. 3(b) demonstrates 2-step-ahead CV with a fixed window, where 'T' and 'V' refer to the training and validation samples, respectively. The first step is identical across Fig. 2 and Fig. 3: observations 1 to 3 constitute the training set and observation 4 is used for validation while the remaining observations are unused as indicated by a dot ('.'). In step 2, the training set becomes larger in the expanding window setup such that it consists of observations 1 through 4 (Fig. 2(a)) whereas the size of the training set is fixed in Fig. 3(a) such that it consists of observations 2 through 5. Considering the focus here is on macroeconomic data, and GDP in particular, we opt for an expanding window approach for the base learners in this paper. For the Monte Carlo experiments we will use 1-step ahead forecasts, while for the empirical application we will use 1 and 4 step ahead forecasts.



(a) $h = 1$, expanding window

(b) $h = 2$, expanding window

Fig. 2. Rolling h-step ahead cross-validation with expanding training window. 'T' and 'V' denote that the observation is included in the training and validation sample, respectively. A dot ('·') indicates that an observation is excluded from both training and validation data.



(a) $h = 1$, fixed window

(b) $h = 2$, fixed window

Fig. 3. Rolling h-step ahead cross-validation with fixed training window.

3 Stacking

Machine learning methods have become popular in time series applications, too. With 'wide' databases becoming available for different applications, there has been an influx of applied papers utilising a plethora of different methods for fit and variable selection (Goulet Coulombe et al., 2022; Kohns and Bhattacharjee, 2022; Massacci and Kapetanios, 2023). Given the breadth of choice, one question to ask is which model to use. In some cases we have prior information which can

help us make a decision, e.g., we believe the problem at hand is sparse and linear, and so we prefer the lasso. But often we don't have this information. Stacking regression (Wolpert, 1992) provides a potential solution: rather than select a 'best' model, mix the different models in a principled manner. In essence, the idea is that our models describe reality given some simplification. Not all models simplify the problem at hand along the same 'axis' (i.e., the models have different assumptions). In such cases, we can find an optimal convex mixture of the models to give an overall better prediction. Importantly, since stacking is a generalization of cross-validation, this convex combination of models is theoretically founded (for a textbook treatment, see Hastie et al. (2009)).

With stacking regression, we combine predictions from multiple learners into a meta model. The initial set of models consists of 'base learners' or 'Level 0 models'. The stacking method combines the predictions of the base learners into a 'meta model' or 'Level 1 model'. Assume we have M base learners and denote by $\hat{f}_m(\boldsymbol{x}_i)$ the prediction for observation i of base learner m after tuning. Formally, stacking can be represented as:

$$\begin{aligned} \hat{\boldsymbol{w}} &= \underset{w_1,\ldots,w_M}{\arg\min} \sum_{i=1}^{n} \left(y_i - \sum_{m=1}^{M} w_m \hat{f}_m(\boldsymbol{x}_i) \right)^2 \\ \text{s.t. } & w_m \geq 0, \\ & \sum_m |w_m| = 1 \end{aligned} \tag{1}$$

Note that this set of equations essentially describes a constrained least squares problem, where we constrain the weights to be non-negative and their sum to be unity. These constraints lead to better performance and facilitate the interpretation of stacking as a weighted average of base learners (Hastie et al., 2009).

Stacking on mean squared is the most common approach here, just as it is the most common choice of scoring method in standard cross-validation. However, time series applications are notorious for various breaks occurring in the data. Extreme events are known to have a large influence on models focused on the mean (Rousseeuw and Hubert, 2011). Importantly, from a forecasting perspective, conditional mean models usually yield subpar forecasts right after crisis episodes, which is exactly the moment policymakers and stakeholders need accurate information. The median has a better breakdown point than the mean (Huber and Ronchetti, 2009; Rousseeuw and Hubert, 2011), which makes it a more attractive choice for modelling during uncertain times. Recasting the equation to stack on the median yields:

$$\begin{aligned} \hat{\boldsymbol{w}} &= \underset{w_1,\ldots,w_M}{\arg\min} \sum_{i=1}^{n} \left| y_i - \sum_{m=1}^{M} w_m \hat{f}_m(\boldsymbol{x}_i) \right| \\ \text{s.t. } & w_m \geq 0, \\ & \sum_m |w_m| = 1 \end{aligned} \tag{2}$$

Note how the only difference between Eq. (1) and (2) is that the objective function in the latter minimises absolute deviations. From an application standpoint this can be solved using Koenker and Ng (2005) instead of a constrained least squares. To avoid confusion with learners and scoring methods, we refer to stacking on the median as "Stacking (L1)" and stacking on mean squared as "Stacking (L2)" in the figures and tables.

4 Base Learners

In this section we briefly describe the base learners we use in the paper: lasso, ridge, elastic net, support vector machine, and random forest.

4.1 Lasso

When dealing with high-dimensional data, researchers often have to circumvent overfitting problem. Including too many irrelevant variables in the regression model can result in poor out of sample generalization. Tibshirani (1996) introduced the lasso ('least absolute shrinkage and selection operator') to improve the prediction accuracy and interpretability of models in such case by performing both regularization and variable selection. Consider a sample with n observations and p covariates. Let y_i be the outcome and X_i be the vector of regressors for the i^{th} observation. The lasso solves the optimization problem:

$$\hat{\boldsymbol{\beta}}(\lambda) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(y_i - X_i' \boldsymbol{\beta}\right)^2 + \lambda ||\boldsymbol{\beta}||_1 \right\}, \tag{3}$$

where $\boldsymbol{\beta}$ is the coefficient vector, and λ is the regularization parameter that controls the overall penalty level. A higher λ means a stronger penalty on the magnitude of all coefficients. At one extreme, lasso estimates approach those of OLS as λ goes to zero. At the other end, all coefficients are shrunk to zero when λ is large enough. In practice, λ is usually tuned through cross-validation, while the search range needs to be predetermined by the researcher. Including non-zero coefficients for covariates will increase the score of the loss function. Consequently, all coefficients will shrink towards zero, while the coefficients of those covariates who contribute little or nothing to the outcome will be shrunk to exactly zero. The covariates are typically standardized so that the solution does not depend on the measurement scale. The L_1 norm $||\boldsymbol{\beta}||_1$ makes lasso a quadratic programming problem, hence there is no closed form solution and the computation can be slow. Lasso is an appropriate choice for both prediction and variable selection when the model is sparse.

4.2 Ridge

Ridge regression (Tikhonov, 1963; Hoerl and Kennard, 1970) was the most popular technique for improving predictive performance prior to lasso. It resembles

lasso but has one key difference: the L_2 norm is used for the penalization term. In particular, ridge solves the problem:

$$\hat{\boldsymbol{\beta}}(\lambda) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(y_i - X_i' \boldsymbol{\beta}\right)^2 + \lambda ||\boldsymbol{\beta}||_2^2 \right\} \tag{4}$$

Ridge also intends to reduce prediction error by shrinking all coefficients towards zero, but no coefficients will be shrunk to exactly zero, which means that it does not perform variable selection. Additionally, no requirement for the sparsity assumption makes ridge attractive when the model is dense. Unlike the lasso, ridge is also computationally efficient since it has a closed form solution:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X'X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X'y},$$

where $\boldsymbol{X}$ is the $n \times p$ design matrix, $\boldsymbol{I}$ is a $p \times p$ identity matrix, and $\boldsymbol{y}$ is the $n \times 1$ vector of outcome. In general, the solution is still well defined when $\boldsymbol{X'X}$ is rank deficient provided that λ is sufficiently large.

4.3 Elastic Net

Elastic net regularization (Zou and Hastie, 2005) is simply a linear combination of L_1 and L_2 penalties of lasso and ridge. Specifically, it solves the problem:

$$\hat{\boldsymbol{\beta}}(\lambda) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(y_i - X_i' \boldsymbol{\beta}\right)^2 + \lambda_1 ||\boldsymbol{\beta}||_1 + \lambda_2 ||\boldsymbol{\beta}||_2^2 \right\} \tag{5}$$

Note that lasso has several limitations in practice. Firstly, for high-dimensional data where the number of covariates p is larger than the number of observations n, lasso selects at most n covariates before it saturates. Secondly, lasso tends to select only one covariate from a group of highly correlated covariates and discards the others even they all contribute to the outcome. Thirdly, the solutions of lasso are not always unique and re-ordering the covariates may end up with different estimates. The elastic net overcomes the limitations by adding a quadratic term to the penalty while still preserving the advantages of lasso. The L_2 penalty makes the loss function above strongly convex and hence has a unique solution. A common practice of reparameterization is to set:

$$\begin{aligned} \lambda_1 &= \alpha\lambda \\ \lambda_2 &= (1-\alpha)\lambda \end{aligned} \tag{6}$$

where λ controls the overall penalty level and α controls the balance between lasso and ridge. A higher α indicates a higher weight on lasso and more coefficients will be shrunk to zero. The reparameterization is useful in the sense that it allows us to fix α and selects a single parameter λ instead of tuning λ_1 and λ_2 separately.

4.4 Support Vector Machine

The linear *Support Vector Machine (SVM)* (Boser et al., 1992) solves a classification problem by finding a decision function, $f(X)$, based on a set of n observations X_i with labels $y_i \in \{1, -1\}$, that divides all observations into two classes. From the training set this algorithm estimates the parameters of the decision function $f(X)$ through a learning process. Then the classification of a new observation is predicted according to the decision function. Each data point X_i can be viewed as a p-dimensional vector. Consider a $p-1$ dimensional *hyperplane* defined by $\{X : f(X) = X'\boldsymbol{\beta} + \beta_0 = 0\}$, linear SVM chooses the best hyperplane that maximizes the distance (or margin) from it to the nearest data point in each class. The hyperplane is a geometric representation of the decision function $f(X)$ with a p-dimensional norm vector, $\boldsymbol{\beta}$, and a bias term, $\beta_0 \in \mathbb{R}$. Linear SVM's training outcome is a classification rule, $G(X)$, depending on the side of the hyperplane that an unclassified observation lands on. In particular, $G(X) = \text{sign}[f(X)] = \text{sign}[X'\boldsymbol{\beta} + \beta_0]$.

If there exists two parallel hyperplanes that separate the two classes of training set, linear SVM maximises the *margin*, $M = \frac{2}{||\boldsymbol{\beta}||}$, between the planes by solving the minimization problem:

$$\begin{aligned} &\underset{\boldsymbol{\beta},\beta_0}{\text{minimize}}\ ||\boldsymbol{\beta}||_2^2 \\ &\text{s.t. } y_i(X_i'\boldsymbol{\beta} + \beta_0) \geq 1\ \forall i \in \{1, 2, ..., n\} \end{aligned} \tag{7}$$

However, data are sometimes not linearly separable. This can be incorporated in the optimization function by including a hinge loss function $\xi_i = \max(0, 1 - y_i(X_i'\boldsymbol{\beta} + \beta_0))$, which is proportional to the distance from the margin if the point is misclassified and takes the value of zero if the point lies on the correct side. We can now modify the optimization problem above as:

$$\begin{aligned} &\underset{\boldsymbol{\beta},\beta_0,\xi}{\text{minimize}}\ ||\boldsymbol{\beta}||_2^2 + C\sum_i^n \xi_i \\ &\text{s.t. } y_i(X_i'\boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i,\ \xi_i \geq 0\ \forall i \in \{1, 2, ..., n\}, \end{aligned} \tag{8}$$

where C is the *cost parameter* that penalizes the amount of observations inside the margin. A larger value of C will make the optimization choose a smaller *margin* and hence increasing the overfitting. The Lagrange dual function can be written as:

$$\begin{aligned} \mathcal{L}(\alpha_1, \alpha_2, ..., \alpha_n) &= \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i y_i (X_i' X_i) y_j \alpha_j \\ \text{s.t. } \alpha_i \geq 0, &\sum_i^n \alpha_i y_i = 0\ \forall i \in \{1, 2, ..., n\} \end{aligned} \tag{9}$$

where α_i are Lagrange multipliers. The function can be efficiently solved by quadratic programming algorithms, yielding $\hat{\boldsymbol{\beta}} = \sum_{i=1}^n \hat{\alpha}_i y_i X_i$. Notably, $\alpha_i \geq 0$

only if the point X_i lies on the boundary of the *margin*. such points are called *support vectors*.

A regression version of SVM, *support vector regression (SVR)*, was proposed by Drucker et al. (1996). Training SVR means solving the problem:

$$\begin{aligned} &\underset{\boldsymbol{\beta},\beta_0}{\text{mininize}}\ \frac{1}{2}||\boldsymbol{\beta}||_2^2 \\ &\text{s.t. } |y_i - X_i'\boldsymbol{\beta} - \beta_0| \leq \epsilon\ \forall i \in \{1, 2, ..., n\}, \end{aligned} \tag{10}$$

where ϵ is a tuneable parameter that serves as a threshold. The distance between any prediction and the true value should be within the range ϵ.

4.5 Random Forests

The *random forests* (Breiman, 2001) algorithm applies 'bagging' (bootstrap aggregation) to *decision tree learners*. Although tree learners are invariant to transformations of features and hence robust to inclusion of irrelevant features, they tend to overfit the training sets and suffer from high variance. In principle, tree-based algorithms split the training set into subsets based on thresholds of selected features with the purpose to minimize the prediction error. The process is recursively repeated on each derived subset until a subset (node) with the minimum amount of observations is reached, which is usually set to five for regression problem. However, the minimum size can be tuned to alleviate overfitting.

Given a training set $\boldsymbol{X} = \{X_1, X_2, ..., X_n\}$ with outcomes $\boldsymbol{y} = \{y_1, y_2, ..., y_n\}$, random forests repeatedly draw a random sample of size n with replacement from the set B times and fits trees to the samples. A tree learner T_b is trained on each sample $\boldsymbol{Z_b} = (\boldsymbol{y_b}, \boldsymbol{X_b})$. After training, the prediction for an observation with features X can be made by averaging over all the individual trees:

$$\hat{T}^B(X) = \frac{1}{B}\sum_{b=1}^{B} T_b(X) \tag{11}$$

The bootstrapping procedure improves the model performance in the sense that it reduces the variance without increasing the bias. The predictions of a single tree could be sensitive to outliers, but the average of trees will be less affected, as long as trees are uncorrelated. Although bootstrap sampling decorrelates the trees by training them on different samples, they can still be highly correlated if several strong features are mostly or always selected. Random forests addresses this problem by including another type of bagging: a modified tree learning algorithm that selects a random subset of the features at each split is used. The process is called *feature bagging*. Typically, the number of randomly selected features is set to $m = \frac{p}{3}$ for regression, where p is the total number of features. Additionally, the optimal number of trees B can be tuned by finding the one that minimizes the *out-of-bag error (OOB)*, which is the mean prediction

error on each observation X_i, using only the trees that do not include X_i in their bootstrap sample.

In this paper, rather than explicitly tune the random forest specifications by selecting the number of features or tree depth using cross-validation, we specify a small number of different random forest specifications as base learners. In this sense, the stacking algorithm tunes the random forest specification as part of the overall stacking procedure when it assigns weights to the different random forest learners.

5 Monte Carlo

We investigate the finite-sample performance of stacking using simulated data. Stacking is compared with its components: lasso, ridge, elastic net, support vector machine, and random forest. The Monte Carlo designs are explained in Sect. 5.1, the normalization of data is presented in Sect. 5.2, and evaluation criterion are discussed in Sect. 5.3. Finally, we compare the results of different learners in Sect. 5.4.

5.1 Setup

The dependent variable y_t is generated as:

$$y_t = \alpha y_{t-1} + \boldsymbol{\beta} X_{t-1} + \epsilon_t, \ \ for\ t = 1, 2, ..., n, \tag{12}$$

where X_{t-1} is the vector of all variables of length p and ϵ_t is the error term. X_t follows multivariate normal distribution $N_p(0, \Sigma)$, where Σ is the $p \times p$ covariance matrix with element $\Sigma_{ij} = 0.2^{|i-j|}$. Considering that the performances of learners such as lasso can be largely different based on whether the model is sparse or dense, we set $\boldsymbol{\beta} = \{1, 0.5, 0.2, 0, 0, 0, ...\}$ to simulate the sparse model and $\boldsymbol{\beta} = \{\frac{1}{\sqrt{1}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}}, ...\}$ to represent the dense model. The following three types of data generating processes are used:

DGP I (Autoregressive distributed lag model): α is fixed to be 0.5. The error term is distributed as $\epsilon_t \sim N(0, 1)$.

DGP II (ARDL model with a fat tail): Since extreme events are likely to happen in many applications, it is of interest to know how the performances change with more outliers. Now the error term follows a t distribution with degrees of freedom set to 3.

DGP III (Non-stationary model): The dependent variable y_t is generated as in DGP I, but now α is set to be 1 such that the model is non-stationary.

We run $R = 100$ simulations for each DGP with $p = n = 200$. Since a 'bad' starting point may over-sample points occurring with low probability before it reaches the equilibrium distribution, we also include 50 burn-in periods at the beginning of each DGP. 30% of the data (60 observations) are treated as validation sets, on which stacking regression is done. For each simulation, the

last observation is treated as the testing set, i.e. this observation is not used for training the base learners, nor for the stacking regression. Instead, the test set is used to evaluate the performance of the different estimators.

5.2 Data Normalization

Normalization is important for distance-based machine learning algorithms such as SVM. A distance summarizes the relative difference between two vectors. Numerical values may have different scales, which can greatly affect the calculation of distance measures. In particular, features with relatively larger scales will have stronger impacts on the distance even they actually contribute less to the dependent variable. To avoid this issue, all the features are normalized to have mean zero and unit variance. Note that scaling the whole sample up front will lead to data leakage when the data are dependent. Therefore, normalization needs to be conducted for each training set individually within the corresponding cross-validation split. Once the stacking regressors are trained, we then normalize the whole training sample at once, and the corresponding normalization factors are applied to normalize the hold-out sample. In essence, we run all our base learners on the normalized data without causing any data leakage issue.

5.3 Performance Measures

Root mean squared prediction error (RMSPE) is used to compare the finite-sample performance of two types of stacking and its base learners. Specifically,

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

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

where T is the number of predictions, i.e. the size of the testing set. y_t and $\hat{y}_t$ are the out-of-sample realized and predicted values respectively.[1]

5.4 Results

Table 1 shows the results from MC experiments. Overall, both fat tails and non-stationarity will lead to higher RMSPE of all the learners. Lasso always performs the best among all the base learners when the model is sparse. While ridge or elastic net has the smallest mean RMSPE when the model is dense. As expected, both stacking methods follow closely the best base learners in different situations, and sometimes even outperform all the base learners. Even though the difference

[1] Note that $T = 1$ in our MC setting, leading to $RMSPE = MAPE$. As such we will only focus on RMSPE when discussing the MC results. We define both measures here because in the empirical application, T equals to 30, leading to $RMSPE \neq MAPE$.

Table 1. mean RMSPE of Monte Carlo Experiments

	Sparse			Dense		
	DGP I	DGP II	DGP III	DGP I	DGP II	DGP III
lasso	1.543	2.765	2.434	3.453	9.748	18.218
ridge	2.494	4.155	11.347	2.561	10.155	29.241
EN	1.550	2.778	2.575	3.435	9.700	18.197
svm	2.598	4.446	14.268	3.842	11.806	36.933
rf10	2.398	3.666	6.261	8.335	12.809	22.939
rf50	2.246	3.173	5.675	7.681	13.448	21.803
rf100	2.248	3.281	5.485	7.430	12.998	20.761
rf200	2.219	3.351	5.301	7.342	13.221	21.300
Stacking (L1)	1.611	2.686	2.487	2.808	10.078	17.064
Stacking (L2)	1.611	2.693	2.469	2.822	9.873	17.284

is small, stacking on median seems to perform slightly better than stacking on mean in general. It is worth noting that stacking on median has a smaller mean RMSPE than stacking on mean in DGP II where the model is sparse and has a fat tail, since the median method is more robust to outliers. However, the contrary is true when the model is dense, suggesting that the degree of sparsity may also influence the relative performance of two stacking methods. A further investigation is beyond the scope of the present paper.

6 Empirical Application

For this empirical application we use McCracken and Ng (2020), a database of US macroeconomic variables at the quarterly frequency. Kohns and Szendrei (2020) have shown that one can use the median as an adequate measure of fit on this database. Here, we take the mantle forward with the performance of the models when stacking on the median and compare the performance with stacking on the mean square. Our application includes the Covid-19 period, which is notoriously difficult to incorporate in forecasting models (Primiceri and Tambalotti, 2020; Ioannidis et al., 2022). In this section, we use the same base learners described in the Monte Carlo section above.

To ensure a rich selection of variables, we follow Kohns and Szendrei (2020) and start our empirical exercise from 1970Q1, which means we have 228 variables. We use the final 30 observations for testing purposes. Importantly, this means that the Covid-19 period is included in the testing set as well as data from 'normal' times. We perform 1 quarter ahead and 1 year (4 quarters) ahead direct forecasts.

The results for the static forecast exercises are presented in Fig. 4 for the 1-quarter-ahead, and Fig. 5 for the 1-year-ahead forecast horizon. The figures show the root mean squared prediction error (RMSPE) and the mean absolute

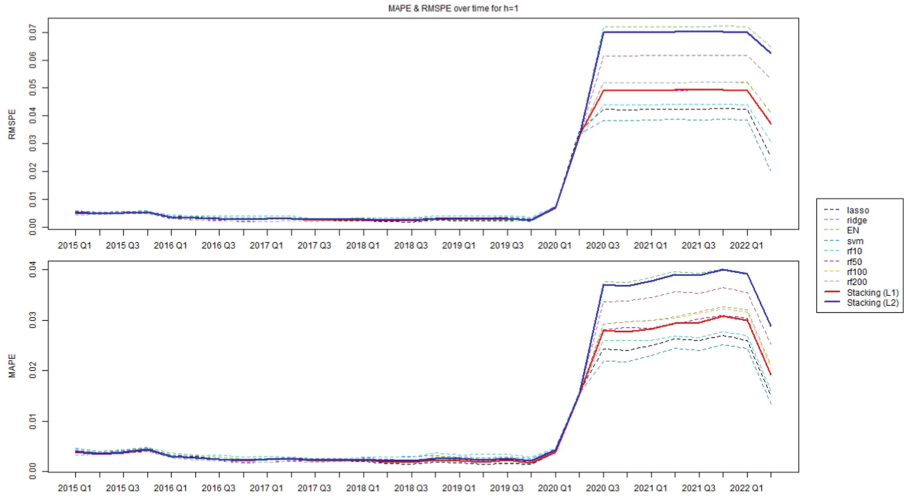


Fig. 4. Forecast results for 1 quarter ahead

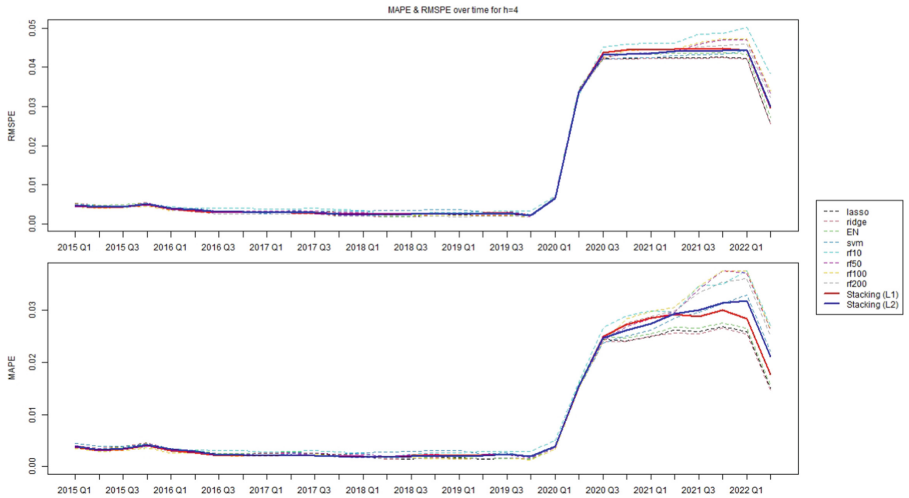


Fig. 5. Forecast results for 1 year ahead

prediction error (MAPE) with a moving window of 2 years. Note that our evaluation window is always backwards looking. The figures reveal that the base learners that work particularly well in 'normal' times (e.g., elastic net and ridge regression), show worse performance during the Covid-19 period, while some models perform better around periods of crisis (e.g., lasso). This highlights that choosing a model that performs adequately across all time periods is difficult.

Looking at the stacked regression performance in the short forecast horizon, we can see that during normal times stacking on the mean and median

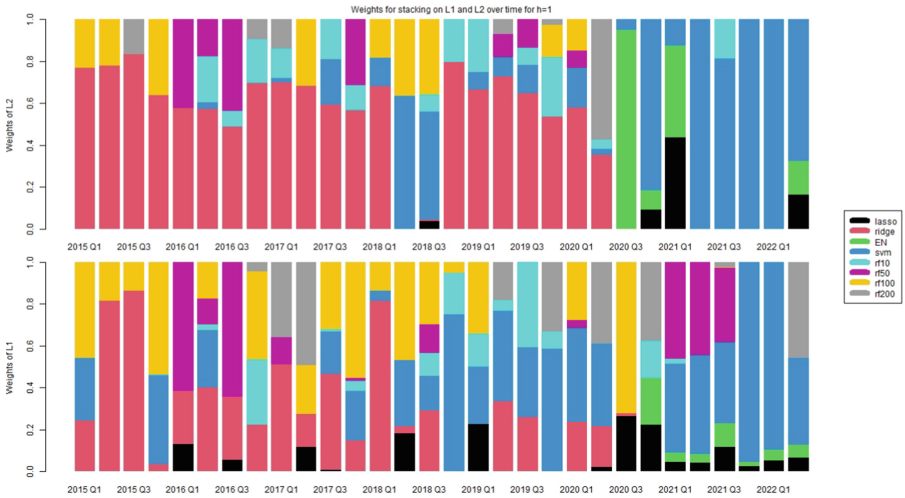


Fig. 6. Weights of the different stacking methods (1 quarter ahead)

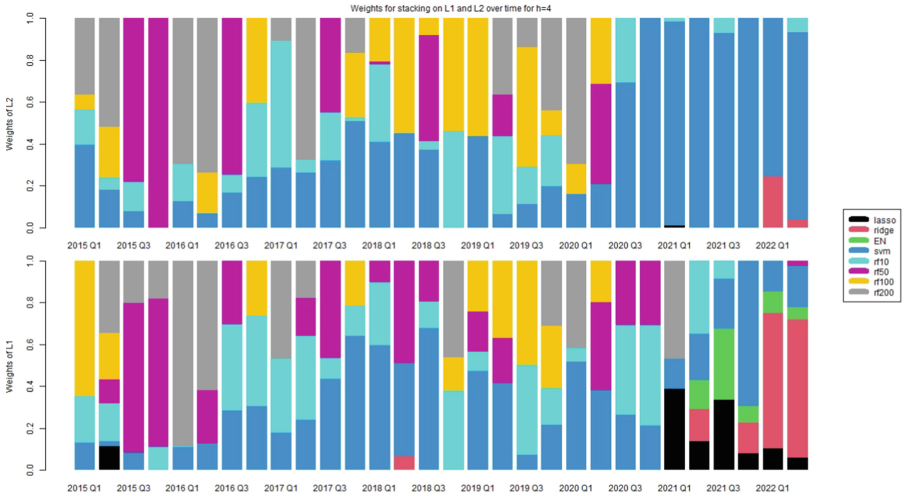


Fig. 7. Weights of the different stacking methods (1 year ahead)

offers comparable performance. Nevertheless, during the crisis episode, the performance of the two stacking methods deviates. During the Covid-19, stacking on the median offers far better performance than stacking on the mean. This is not surprising given the fact that the median is more robust to rank preserving shocks (Huber and Ronchetti, 2009). The results are similar for the longer forecast horizon: stacking on the median performs admirably.

Interestingly, the performance of stacking on the median is not impacted by the choice of the horizon: RMSPE and MAPE are almost identical throughout the time-frame. Looking at the weights in Figs. 6 and 7 reveals why this might be the case. In essence, stacking on the median is more likely to mix information from more base learners. Given that longer forecast horizons have more uncertainty associated with it, a stacking method that is more likely to incorporate information from a more diverse set of models is likely to fare better.

Comparing the weights across the forecast horizons also reveals how different types of base learners are preferred at the different forecast horizons, with the SVM being the only base learner that is included frequently for both horizons. At the shorter forecast horizon, ridge regression is more dominant especially for stacking on mean square, while for the longer forecast horizons random forests are far more prevalent. Random forests being selected at the longer forecast horizon is likely because although random forests overfit in-sample, this has little to no consequences out-of-sample.[2] Importantly, we can see from the weights that although there are multiple random forests among the base learners, stacking regression always assigned non-zero weights to other base learners as well. This further highlights a key advantage of stacking: we are not limited to mixing only one type of model. In this instance, information from the elastic net learner is mixed into the stacking regression, which leads to better performance, especially during the crisis episodes.

Our results point towards running both types of stacking methods at all times and relying on stacking during crisis times due to its robustness. It is difficult to know ex ante (and sometimes even in real time) whether one is in a crisis, which makes it difficult to choose between the two stacking methods. Figures 8 and 9 show that the fitted values of the two stacking methods are very close during normal times, but deviate from each other during crisis periods. The tendency for the two sets of fitted values to deviate during crises episodes is not too surprising given the robustness of the median to outliers (Huber and Ronchetti, 2009). As such, one can opt to favor stacking on the median, when the fitted values deviated from each other. We leave for future research the question of at what point deviations between the fitted values should be considered significant from a policy maker perspective.

[2] See Goulet Coulombe (2020) for further discussion and an explanation of why random forests tend to perform relatively well in a forecasting setting.

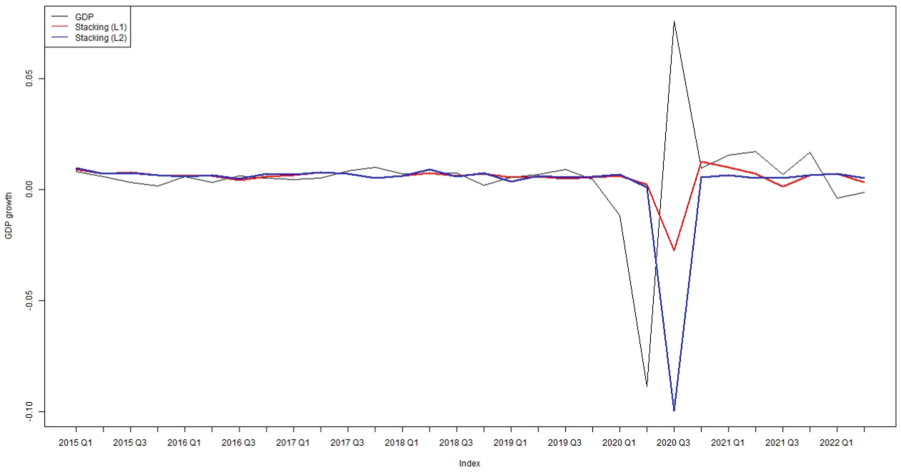


Fig. 8. Fitted values and GDP at the 1 quarter ahead horizon

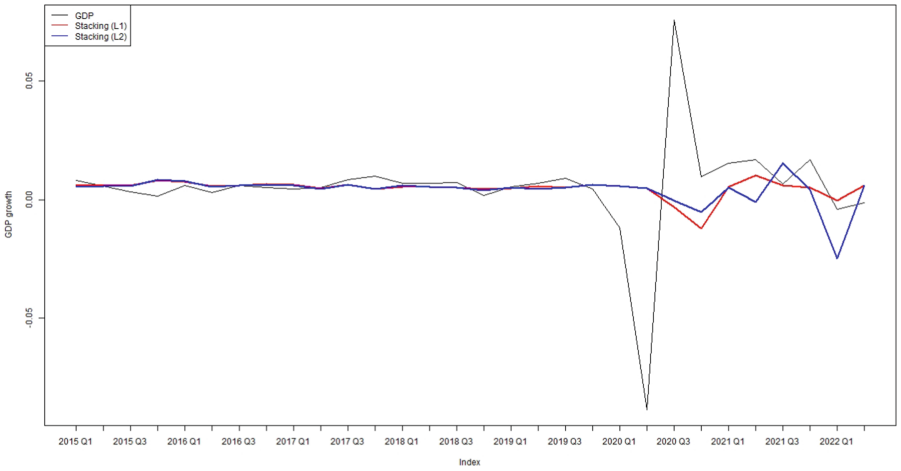


Fig. 9. Fitted values and GDP at the 1 year ahead horizon

7 Conclusion

The key goal of stacking regression is to obtain an optimal mix of models that can lead to better fit than one particular model. Stacking regression has been popular for cross-section data and in this paper we outlined how to apply the method to time-series data. We note that extreme observations are not infrequent in time-series settings (e.g., macroeconometrics and finance), and this can have detrimental effects on models focused on optimizing the squared residual. To remedy this we propose 'stacking on the median', since the median is more robust to outliers Rousseeuw and Hubert (2011).

In the Monte Carlo exercise we find that stacking on the median performs admirably, even beating stacking on the mean squared error. These results are corroborated by the empirical application focused on US GDP forecasts around the global pandemic. Rather than exclusively stacking on the median, we propose that policymakers consider running the two methods simultaneously, as the fitted values deviate during uncertain times. This way the policymaker will have information on not just what the forecasted value is, but also when an extreme event has occurred.

References

Bergmeir, C., Hyndman, R.J., Koo, B.: A note on the validity of cross-validation for evaluating autoregressive time series prediction. Comput. Stat. Data Anal. **120**, 70–83 (2018)

Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 144–152 (1992)

Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)

Da Silva, R.G., Ribeiro, M.H.D.M., Fraccanabbia, N., Mariani, V.C., dos Santos Coelho, L.: Multi-step ahead bitcoin price forecasting based on vmd and ensemble learning methods. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2020)

Drucker, H., Burges, C.J., Kaufman, L., Smola, A., Vapnik, V.: Support vector regression machines. Adv. Neural Inf. Process. Syst. **9** (1996)

Goulet Coulombe, P.: To bag is to prune (2020). arXiv preprint arXiv:2008.07063

Goulet Coulombe, P., Leroux, M., Stevanovic, D., Surprenant, S.: How is machine learning useful for macroeconomic forecasting? J. Appl. Econom. **37**(5), 920–964 (2022)

Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. SSS, Springer, New York (2009). https://doi.org/10.1007/978-0-387-84858-7

Hoerl, A.E., Kennard, R.W.: Ridge regression: applications to nonorthogonal problems. Technometrics **12**(1), 69–82 (1970)

Huber, P.J., Ronchetti, E.M.: Robust statistics. Wiley Series in Probability and Mathematical Statistics (2009)

Hyndman, R.J., Athanasopoulos, G.: Forecasting: Principles and Practice (2 ed.) (2018)

Ioannidis, J.P., Cripps, S., Tanner, M.A.: Forecasting for COVID-19 has failed. Int. J. Forecast. **38**(2), 423–438 (2022)

Koenker, R., Ng, P.: Inequality constrained quantile regression. Sankhyā Indian J. Stat. 418–440 (2005)

Kohns, D., Bhattacharjee, A.: Nowcasting growth using google trends data: a Bayesian structural time series model. Int. J. Forecast. (2022)

Kohns, D., Szendrei, T.: Horseshoe prior Bayesian quantile regression (2020). arXiv preprint arXiv:2006.07655

Massacci, D., Kapetanios, G.: Forecasting in factor augmented regressions under structural change. Int. J. Forecast. (2023)

McCracken, M., Ng, S.: FRED-QD: a quarterly database for macroeconomic research. Technical report, National Bureau of Economic Research (2020)

Pavlyshenko, B.: Using stacking approaches for machine learning models. In: 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), pp. 255–258. IEEE (2018)

Primiceri, G.E., Tambalotti, A.: Macroeconomic forecasting in the time of COVID-19. Manuscript, Northwestern University, pp. 1–23 (2020)

Ribeiro, M.H.D.M., dos Santos Coelho, L.: Ensemble approach based on bagging, boosting and stacking for short-term prediction in agribusiness time series. Appl. Soft Comput. **86**, 105837 (2020)

Ribeiro, M.H.D.M., Ribeiro, V.H.A., Reynoso-Meza, G., dos Santos Coelho, L.: Multiobjective ensemble model for short-term price forecasting in corn price time series. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2019)

Rousseeuw, P.J., Hubert, M.: Robust statistics for outlier detection. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **1**(1), 73–79 (2011)

Steel, M.F.: Model averaging and its use in economics. J. Econ. Lit. **58**(3), 644–719 (2020)

Tibshirani, R.: Regression shrinkage and selection via the lasso. J. Roy. Stat. Soc. Ser. B (Methodol.) **58**(1), 267–288 (1996)

Tikhonov, A.N.: On the solution of ill-posed problems and the method of regularization. In: Doklady Akademii Nauk, vol. 151, pp. 501–504. Russian Academy of Sciences (1963)

Wolpert, D.H.: Stacked generalization. Neural Netw. **5**(2), 241–259 (1992)

Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. R. Stat. Soc. Ser. B Stat. Methodol. **67**(2), 301–320 (2005)