Unit Roots in Macroeconomic Time Series: A Comparison of Classical, Bayesian and Machine Learning Approaches

Yamin Ahmad[a,*], Adam Check[b], Ming Chien Lo[c]

[a] *University of Wisconsin - Whitewater, Department of Economics, 809 W. Starin Road, Whitewater WI 53190, USA*

[b] *University of St. Thomas, Department of Economics, 2115 Summit Avenue, St. Paul, MN 55105, USA*

[c] *Metropolitan State University, Department of Economics and Finance, College of Management, 1501 Hennepin Avenue, Minneapolis, MN 55403, USA*

Abstract

We compare the effectiveness of Classical, Bayesian, and Machine Learning (ML) methods for predicting the presence of a unit root in univariate time-series models. Framing the issue as a classification problem, we demonstrate how ML may be used to uncover structural features of a macroeconomic time series with small data. We evaluate these approaches and find that ML outperforms the Classical and Bayesian tests using prediction accuracy, and appears to be the most flexible for classifying unit roots. Using US macroeconomic data, we find broad consensus among the approaches for predicted nonstationary series, with some disagreement for predicted stationary series.

JEL Classification: C22, C49, C53, C55, E1

Keywords: k-Nearest Neighbors, Random Forest, Supervised Learning, Support Vector Machines

Word Count: 12,733

*Corresponding author

Email addresses: ahmady@uww.edu (Yamin Ahmad), ajc@stthomas.edu (Adam Check), Ming.lo@metrostate.edu (Ming Chien Lo)

1. Introduction

This paper compares a number of Classical, Bayesian and Machine Learning methods for accurately predicting the presence of a unit root in a univariate time-series. While the unit root testing literature is well-developed for both Classical and Bayesian methods, properties of Machine Learning (henceforth ML) methods in detecting a unit root have not been widely studied. We use the issue of identifying a unit root as an illustration to demonstrate how Machine Learning may be used to uncover structural features in macroeconomic time series with small data. With the availability of big data, economists have begun to utilize sparse modeling techniques, such as LASSO and other forms of regularized regressions, for prediction problems and for causal inference.[1] However, applications of these kinds of ML techniques in addressing macroeconomic questions have been rare, in part because time series data is smaller in nature.[2]

We make three contributions to the literature in this paper. First, we propose and demonstrate how to employ some popular ML methods that have had success with prediction problems involving big data to macroeconomic time series data, where we have relatively small samples. Second, we ask whether ML methods may be used to detect the presence of a unit root in a macroeconomic time series, without making any assumptions about the underlying data generating process (DGP). Third, we investigate how predictions from a ML approach that are agnostic about the underlying model structure and DGP, compare to predictions from both Classical and Bayesian methods in accurately classifying whether a particular time series has a unit root or not. We try to determine which of the methods that we consider can best distinguish between a

[1] Some examples include research where ML has been used to tackle problems related to causal inference (Athey et al., 2017; Wager and Athey (2018)), optimal policy (Belloni et al., 2017 ; Athey and Imbens, 2017; Athey, 2015), and even for panel data (Athey et al., 2018). Athey and Imbens (2019) and Mullainathan and Spiess (2017) provide a good introduction to the types of ML methods relevant for economists for anyone considering starting out in this area.

[2] One recent application in macroeconomics is Giusto and Piger (2017) who use vector quantization to identify turning points in business cycles.

unit root and stationary series using a combination of measures like accuracy, precision and recall.

We use supervised learning and frame the problem as a classification problem, where we try to classify a number of time series as one that either contains a unit root or as a stationary process, in a similar spirit to that of Stock (1994). We use a Monte Carlo approach to generate a number of unit root and local-to-unity processes and train the ML algorithms using a training sample. We then use a hold-out set to compare standard unit root tests, like the Augmented Dickey-Fuller (ADF) and KPSS tests, and the Bayesian Marriott and Newbold (1998) test, against a number of nonparametric ML algorithms such as k-nearest neighbors (kNN), support vector machines (SVM), and random forest (RF). We evaluate these approaches using accuracy as the key metric, and assess their ability to correctly classify the time series either as one that contains a unit root or one that is truly stationary.

Our results show that the ML approaches dominate the other approaches when using accuracy as a metric for prediction when there are approximately equal numbers of time series that either contain a unit root or are stationary in a given dataset. Moreover, the ML approach comes in a close second when faced with class imbalance, where there is a greater proportion of the series within the dataset that either contain a unit root or are stationary. It also performs well, relative to the other approaches, in smaller time series data. When there are a large number of unit root series in the simulated data, the classical ADF and the Bayesian tests predict the best in long time series data, followed closely by the ML approaches. Conversely, when a large number of series are stationary, the KPSS test performs the best in predicting these, followed by the ML approaches. Overall, the ML methods appear to be the most flexible in the simulations where these proportions may not be known, and among these the SVM and kNN methods generally predict the best.

In an empirical application, we find broad consensus on a number of individual time series data that are predicted to be nonstationary using these approaches. Once the data is transformed to render it approximately stationary, disagreement occurs between these approaches, with the ML approach indicating that the transformations have rendered nearly all the series stationary, while the Classical and Bayesian tests indicate that a number of resulting series may still contain a unit root. However, there is still consensus here as these approaches jointly predict that approximately 50% of the transformed series are stationary.

The remainder of the paper is structured as follows. Section 2 outlines the research design and methodological approach proposed in this paper. Section 3 reports the main results, while section 4 applies the methods to predict unit roots in actual data. Section 5 discusses some of the advantages and disadvantages of the proposed methodology of applying ML methods to macroeconomic data and then section 6 concludes.

2. Methodology

Consider the following data generating process (DGP):

$$y_t = \rho y_{t-1} + \varepsilon_t \tag{1}$$

where ε_t are iid errors drawn from a Gaussian distribution: $\varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ and $|\rho| \leq 1$. We use Monte Carlo simulations to generate a draw which yields a sequence of values for $\{y_t\}$. A parameter γ determines whether the a particular sequence of y_t follows a random walk or an AR(1) process so that ultimately a proportion γ of the series in the dataset will contain a random walk. Initially, we set $\gamma = 0.5$, although later, we also consider the cases where $\gamma \in \{0.1, 0.25, 0.75, 0.9\}$ represents the fraction of the simulated series in the dataset that are a random walk; the remainder of the draws leads to a stationary AR(1) process for y_t from a given distribution for ρ described below. In order to allow for a clear comparison among the three approaches, we simulate data

using a specification without a trend. In some of our preliminary exercises, we found that an obvious trend in the simulated data would give the machine learning approach a clear advantage that defies our purpose. Complications may also arise in unit root tests when trends are involved - the performance of a unit root test can be sensitive to any specification error (classical) and the number and variety of models included (Bayesian).

Stationarity of the series y_t is governed by the parameter $\rho : |\rho| < 1$. In our simulations that follow, we consider two set of ranges for ρ when the process y_t is stationary. The first case, which we subsequently refer to as the 'Wide' case, is one where y_t is a fairly persistent process where $\rho \sim \mathcal{U}[0.90, 1.0)$. Here, one-fifth of the simulated AR series comes from random draw from a uniform distribution from five sub-ranges: $b \leq \rho \leq b + 0.02$, where $b \in \{0.90, 0.92, 0.94, 0.96, 0.98\}$. The second case we consider, which we subsequently refer to as the 'Narrow' case, is where y_t is a local-to-unity process and as such ρ takes on values between: $\rho \sim \mathcal{U}[0.98, 1.0)$. In this case, we draw uniformly from within the range. Finally, without loss of generality, we let $\sigma_\varepsilon = 1$.

The methodological approach is fairly simple since the goal is to evaluate the different approaches on their ability to predict whether a particular simulated sequence of $\{y_t\}$ contains a unit root. Given that the series are being simulated, we know the true state of the world, and can assess the accuracy of the different approaches in terms of their ability to detect the presence of a unit root, when one is present, as well as a stationary series when it is indeed the case. We use Monte Carlo simulations to generate simulated series, where the DGP is given by equation (1). We create a hold-out set of 10,000 simulated series, which we use to compare the predictions from the different approaches. In addition, we create an independent set of 100,000 simulated series which we use to train the ML algorithms. We do these simulations for $\{y_t\}_{t=1}^{T}$ ranging in length from $T = \{50, 100, 250, 500\}$ observations.

The essence of the identification problem for detecting the presence of a unit root is to be able to (correctly) distinguish between processes where $\rho = 1$ vs $\rho < 1$. This is nearly impossible from an observational perspective, and statistical tests such as classical unit root tests have demonstrated that they have low power against a unit root null. One central question that we investigate in this paper is how the ML approach fares relative to Bayesian and Classical methods. In what follows next, we briefly outline the commonly used unit root tests that we employ to detect the presence of a unit root in the simulated series using a number of Classical, Bayesian and ML approaches.

2.1. Classical Unit Root Tests

We consider two classical-type tests that differ in their null hypothesis:- the Augmented Dickey-Fuller (ADF) test (Dickey and Fuller, 1979), and the KPSS test (Kwiatkowski et al., 1992).

Augmented Dickey-Fuller (ADF) Test

As a first step, we apply the ADF equation with no specification error:

$$\Delta y_t = \phi y_{t-1} + \varepsilon_t \tag{2}$$

on the simulated random walk series for $T = \{50, 100, 250, 500\}$. The null hypothesis here, $H_0 : \phi = 0$, is one that assumes the process y_t contains a unit root. Under the null hypothesis, the distribution follows a Dickey-Fuller distribution (without a constant term). For each value of T, we construct the distribution for the ADF statistic using 10,000 simulated draws, which we use to extract 200 critical value for the p-values ranging from 0.005 to 1.000 with a stepsize of 0.005. Then, we apply the ADF test using these critical values for a given T on all holdout sets of the same length that

include a mix of random walk and AR series.

This ADF specification has no constant term and has no time trend, which is consistent with our data generating process. However, we also consider an alternative specification that includes a constant term:

$$\Delta y_t = \alpha + \phi y_{t-1} + \varepsilon_t$$

because in practice it is highly likely that an agnostic researcher would include a constant term. Like our earlier exercises, we simulate the distribution of the ADF statistic for this version of the test using our simulated random walk, this time including a drift term. After that, we apply this version of the ADF tests on the holdout sets and compare the test statistics against the simulated critical values.

KPSS Test

Kwiatkowski et al. (1992) consider cases where the null hypothesis is either a trend-stationary process or a level-stationary process. Given our DGP, the level-stationary null hypothesis is appropriate. We consider three ranges of AR values in our simulations. The first two are the aforementioned 'narrow range': $(0.9800 \leq \rho \leq 0.9999)$, and 'wide' range: $(0.9000 \leq \rho \leq 0.9999)$. We also consider a low range: $(0.50 \leq \rho \leq 0.80)$ as a robustness check. Accordingly, we simulate three sets of distributions of the KPSS test statistic, where each set has four distributions for each value of T. This gives a total of twelve subsets consisting of a permutation of the range of ρ, and the value of T. The value of ρ in each simulation is randomly drawn from the respective range.

Under the null hypothesis, $y_t = \rho y_{t-1} + \varepsilon_t$ where $\rho < 1$. Kwiatkowski et al. (1992) define $e_t = y_t - \bar{y}$. Although we assume zero unconditional mean in our data generating process, we demean our simulated series when deriving the KPSS test statistics, which

is constructed based on the equation (13) in Kwiatkowski et al. (1992) as follows:

$$KPSS = T^{-2} \sum_{t=1}^{T} \frac{S_t^2}{s^2}$$

where $S_t = \sum_{i=1}^{t} e_t$ and $s^2 = \sum_{t=1}^{T} e_t^2/T$. Since our DGP has no autocorrelation within the simulations, we ignore the second term in equation (10) of Kwiatkowski et al (1992) which requires researchers to assume a specific value for truncation to limit the length of the autocorrelation structure. Such truncations are not needed here. Applying this statistics to the 10,000 series in each of the twelves subsets of simulated AR(1) series, we generate a distribution for the respective range of ρ and T. When applying the KPSS test to the holdout sets, we make use of the specific distribution of the test statistics, matching the range of ρ and T.

2.2. Bayesian Unit Root Tests

The presence of a unit root is not as problematic in the Bayesian framework as it is in the frequentist framework (- see for example Sims and Uhlig, 1991). Most Bayesian testing relies on model comparison instead of null hypothesis significance testing, where Bayesian model comparison is based on each model's marginal likelihood — a statistic that summarizes the weight of the evidence for a model after observing the data. To perform Bayesian unit root testing, we follow Marriott and Newbold (1998) and compare two models: (1) a stationary autoregressive model and (2) a unit root model.

Bayesian unit root testing is sensitive to the prior placed on the autoregressive parameter. While the autoregressive parameter is present in the autoregressive model, it is not present in the unit root model. Therefore, this parameter is subject to "Lindley's paradox" — placing a diffuse prior on the autoregressive parameter will result in near certain preference for the unit root model. Many authors have investigated different types of priors on the autoregressive parameter in order to determine a prior distri-

bution that is both realistic and also performs well in practice. In pioneering work, Berger and Yang (1994) and Uhlig (1994) found that it is important for the prior on the autoregressive parameter to place non-negligible probability on near-unit-root values of the autoregressive coefficients. We follow this advice and use a Beta(5,0.5) prior on the autoregressive coefficient.[3]

Prior to estimation, we place equal prior probability on the autoregressive and unit root models. That is, if the autoregressive model is denoted as M_0 and the unit root model is denoted as M_1, we set:

$$p(M_0) = 0.5$$
$$p(M_1) = 0.5$$

After observing the data and estimating both models, we compute the posterior probability of the unit root model as:

$$p(M_1|Y) = \frac{p(M1)p(Y|M_1)}{p(M0)p(Y|M_0) + p(M1)p(Y|M_1)} \tag{3}$$
$$p(M_1|Y) = \frac{p(Y|M_1)}{p(Y|M_0) + p(Y|M_1)} \tag{4}$$

where we have gone from equation (3) to (4) using the fact that the model priors are all equal. We declare the presence of a unit root if the posterior probability of the unit root model is greater than 0.5. Note that this does not necessarily mean that the DGP has a unit root — just that the marginal likelihood favors one.

[3]Since the variance of the error term is included in both models, Lindley's paradox does not apply to it and we place a non-informative prior on it. In our simulations we set the variance of the error to 1, but we still use the non-informative prior. We believe this more accurately reflects the prior state of knowledge of researchers who are investigating real data, but using this mis-specified prior means that the Bayesian approach is not guaranteed to be the best classifier on the simulated data.

2.3. Machine Learning Approach

The term 'Machine learning' is used to refer to a collection of different algorithms that have been developed for the purpose of prediction. We implement supervised learning, which involves a set of labelled observations and outcomes (classifications), $(X, \mathbb{Y})$, that are used to train the algorithms. Once trained, these algorithms are then applied to an independent dataset, $\left(\tilde{X}, \tilde{\mathbb{Y}}\right)$, and used to predict the outcomes $\widehat{\tilde{\mathbb{Y}}}$ there. We use Monte Carlo simulations to train the ML algorithms, training the ML algorithms to recognize the structural feature of interest in a given data generating process. Using an analogy of a photograph of a cat, training involves teaching the algorithms to recognize all the photos that have a cat and those that do not have a cat. We can then show the algorithm a new photo (- a new macroeconomic time series) and see the outcome of its predictions.

The overall ML methodology is fairly straightforward. A training sample is used by the ML algorithms (or 'learners') to learn about the features of the data. Once trained, it can then be used to make predictions on a hold-out dataset. For classification problems, the predictions that we make are whether a particular observation in the hold-out set belongs to a particular class, along with associated probability of being in that class. Our proposed approach in this paper is to frame the unit root problem as a classification problem, similar in spirit to the approach taken by Stock (1994), and use ML to classify a particular time series. Given the values of ρ we consider, we have a binary classification problem where $\rho = 1$ or not. This may be considered to be analogous to the cat/no-cat classification problem of trying to determine whether there is a cat in a particular image or a photograph.[4]

[4]Due to space considerations, we provide an overview of the approach used here for those unfamiliar with ML, with a brief description of the algorithms used. Additional details may be found in the accompanying technical appendix.

It is useful at this point to note two things. First, for our analysis of macroeconomic time series, we define an observation as a pair consisting of an individual simulated series, along with its class label. This has implications for when we do cross-validation to tune key hyperparameters in the machine learning algorithms. When we implement cross-validation, our 'sample' splits are across the simulations, rather than within a specific simulation. This means that the fundamental properties of a particular simulated time series remains intact, from the underlying autocorrelation structure of the elements in the series, to the governing moments. This in turn allows us to pursue ML as a tool to investigate and compare the predictive properties of the approach, from training the different algorithms, to performing out of sample predictions.

Second, a key assumption that validates our proposed approach is that both the holdout sample and the training sample are drawn from the same data generating process. Were this not to be the case, then the efficiency gains from the algorithms would be reduced if a different data generating process were used to generate the hold-out sample. For our purposes here, this is clearly evident since we are using Monte Carlo methods to simulate both the training and hold-out series through equation (1).[5] However, this is no different than if we were to estimate the data as an autoregressive process in the Classical framework and test a null hypothesis, or specify the set of models that we compare in the Bayesian framework.

Since we simulate both the training and hold-out datasets, we know the true classes of the series y_t in the hold-out set, $\tilde{\mathbb{Y}}$. Thus we can compare the class predictions from the different ML algorithms and see the performance of the algorithms relative to the predictions from the Classical and Bayesian approaches. We consider three of the most popular nonparametric ML approaches that have been used for classification problems

[5]Later in section 4, we apply the method to data, and the maintained assumption is that if the underlying series has a unit root, then without loss of generality, equation (1) is a good approximation to that process.

(Wu et al., 2008): k-Nearest Neighbors (kNN), Support Vector Machines (SVM) and Random Forest (RF), and provide a perfunctory description of each approach and refer the reader to the accompanying technical appendix for additional details.

k-Nearest Neighbors: Given a set of q query observations from the hold-out set, $\tilde{\mathbb{Y}}_z$, the prediction problem is to classify $\tilde{\mathbb{Y}}_z$ as either a unit root process, or a stationary process. The kNN algorithm minimizes the distance between the query observation and the k nearest neighbors in the training sample. We use Euclidean distance as our measure of distance and optimally determine the value of k in the training sample using 2.5 percent of the training sample as potential nearest neighbors. We select a value k^* that yields the smallest cross-validated loss. Once trained, we implement this method to obtain class predictions for the hold-out set based on the probability of each class predicted by the kNN model, repeating for all values of $T = \{50, 100, 250, 500\}$. The optimal values for k^* are reported in Table 1 along with the corresponding values for the training error (- the average loss for the 9 folds used to train the algorithm) and the cross-validated loss. The cross-validated errors are marginally higher than the in-sample training error, and they decrease as the length of the time series T increases.

				Training Error				Cross-Validated Loss			
T	γ	ρ	k^*	original	m4	m12	m48	original	m4	m12	m48
50	0.5	$[0.9, 1)$	210	0.3096	0.3139	0.3143	0.3143	0.312	0.3167	0.3169	0.3169
100	0.5	$[0.9, 1)$	79	0.2286	0.2298	0.2298	0.2298	0.2326	0.2345	0.2341	0.2341
250	0.5	$[0.9, 1)$	71	0.1425	0.1376	0.1375	0.1375	0.1448	0.1409	0.1409	0.1409
500	0.5	$[0.9, 1)$	23	0.0898	0.0865	0.0864	0.0864	0.0948	0.0917	0.0916	0.0916
50	0.5	$[0.98, 1)$	968	0.4409	0.4458	0.4454	0.4454	0.4421	0.448	0.4482	0.4482
100	0.5	$[0.98, 1)$	489	0.3980	0.4007	0.4008	0.4008	0.3993	0.4031	0.4029	0.4029
250	0.5	$[0.98, 1)$	91	0.3048	0.308	0.3079	0.3079	0.3106	0.3158	0.3156	0.3156
500	0.5	$[0.98, 1)$	94	0.2298	0.2281	0.2281	0.2281	0.2326	0.2331	0.2327	0.2327

Table 1: Optimal values of k for kNN algorithm.
Notes:
(i) γ represents the proportion of simulations that contain a random walk; the rest $(1 - \gamma)$ have values of ρ that fall in the respective ranges indicated in the column ρ.
(ii) Training error corresponds to the average loss across all training folds used to train the kNN algorithm. It is the in-sample loss.
(iii) Cross-validated loss corresponds to the average loss across the (out-of-sample) training folds being used for cross validation.
(iv) 'Original' refers to the actual values used in the series; mX refers to the transformed series, where each observation has been transformed into the mean and the remaining first $(X - 1)$ central moments of the series. $X = \{4, 12, 48\}$.

Support Vector Machines: Support vector machines (SVM) classify objects in a hold-out set by finding a separating hyperplane within a training set that can separate objects into their correct classes. Given a set of observations in the training set, $(\mathbf{x}^i, \mathbf{Y}^i)$, we use a vector of weights or coefficients, w, and a constant b to define the hyperplane $w'\mathbf{x}^i + b = 0$. This hyperplane then separates the region $w'\mathbf{x}^i + b > 0$ (i.e. where $\rho = 1$), from $w'\mathbf{x}^i + b < 0$ (i.e. $\rho < 1$). We cross-validate the model and determine the optimal weights for prediction that minimizes the MSPE. Once $\hat{w}$ and $\hat{b}$ are determined, they can be used to classify observations in the hold-out set. We train the SVM model for each value of T, and then use them to make predictions.

Random Forests: Random forests (RF) are an extension of Classification and Regression Trees (CART), an algorithm that utilizes decision trees. The overall approach involves growing decision trees that split the data using rules assigned to the covariates. Random forests involve a collection of these trees that are then used for prediction, and in general, they vote for the most popular class. Athey et al. (2019) interpret random forest as yielding a kernel weighting function that assigns weights to predictions from trees in the neighborhood of the query observation. Trees yielding predictions that are close to the query are weighted more heavily than those further away. We refer the reader to Breiman (2001) for a more detailed explanation of random forests

For all the ML methods above, we construct class probabilities using the distance between the query and training observations. For each value of T, we use a logistic function to obtain the probability of a particular class $\tilde{\mathbb{Y}}$:

$$P(\tilde{\mathbb{Y}}) = \frac{1}{1 + e^{-\sum_j w_j \mathbb{I}(\widehat{\tilde{\mathbf{Y}}}_v^j = \tilde{\mathbf{Y}}_v^j)}} \tag{5}$$

Finally, given a decision threshold, c^*, we assign a class label based on whether $P(\tilde{\mathbb{Y}}) \geq c^*$. This decision threshold is initially set to $c^* = 0.5$. Later in section 3.4 we examine what happens when we vary this threshold.

3. Results

3.1. Comparing Predictions From the Classical, Bayesian and ML Approaches

Classical, Bayesian and ML approaches cannot be compared apples-to-apples because the approach to prediction is different in each. For example, prediction from classical tests are the resulting inference from the outcome of a hypothesis test, while the Bayesian approach assigns a probability that a particular model is true. The ML methods we consider also assign a probability that classifies these series. However, the ML approach is agnostic about the underlying DGP, and the probabilities are dependent upon the various states of the world that the 'Learner' (- the particular learning algorithm used) has been exposed to. In this sense, the ability of the ML approach proposed here to address macroeconomic questions will hinge crucially on the number of states of the world that it is exposed to, in order to be able to make good predictions.

We use *accuracy*, *precision* and *recall* to compare the predictions from the different approaches. These are defined as:

$$Accuracy = \frac{tr(C)}{n} = \frac{TP + TN}{n}; \quad \text{Precision} \quad = \frac{TP}{TP + FP}; \quad \text{Recall} \quad = \frac{TP}{TP + FN}$$

where n is the total number of observations in our holdout set - 10,000 in our case; TP (TN) are true positive (negative) outcomes while FP (FN) are false positive (negative) outcomes respectively. Accuracy reflects the proportion of correctly classified observations, relative to the total number of observations in the hold-out set, and is a good measure when there are approximately equiproportional classes ($\gamma = 0.5$) in the data. We focus on precision and recall when $\gamma \neq 0.5$. Precision reports the proportion of cases where our methods correctly predict a random walk (or stationary process) relative to the total number of random walk (stationary) predictions. Ideally, we would like to have both precision and recall equal to 1. However, as precision increases, there is typically a tradeoff with recall.

3.2. Equiproportionate Classes in the Data

Table 2 reports the results for accuracy, where the classical tests are conducted at a 5% level of significance and predictions from the Bayesian approach are based upon classifications of posterior model probabilities greater than or equal to a threshold of 0.5. The machine learning algorithms are trained on data using a training sample where half the observations consist of a random walk process ($\gamma = 0.5$) and the other half is a stationary process. We use these same trained models for prediction, even where there is class imbalance and the data contains different proportions of random walk and stationary series. Panel A reports accuracy results for the case where stationary

Panel A: Wide Range $[0.9 \leq \rho < 1)$								
T	γ	ADF	ADF-μ	KPSS	Bayesian	kNN	SVM	RF
50	0.5	0.5631	0.5131	0.5572	0.6000	0.6909	0.6915	0.6891
100	0.5	0.6653	0.5484	0.5931	0.6715	0.7677	0.7712	0.7582
250	0.5	0.8418	0.7186	0.6664	0.8063	0.8593	0.8655	0.8479
500	0.5	0.9091	0.8454	0.7198	0.8784	0.9018	0.9086	0.8975

Panel B: Narrow Range $[0.98 \leq \rho < 1)$								
T	γ	ADF	ADF-μ	KPSS	Bayesian	kNN	SVM	RF
50	0.5	0.508	0.504	0.515	0.549	0.554	0.549	0.560
100	0.5	0.516	0.501	0.532	0.525	0.606	0.604	0.602
250	0.5	0.559	0.509	0.558	0.549	0.681	0.690	0.686
500	0.5	0.665	0.550	0.595	0.593	0.769	0.773	0.760

Table 2: Accuracy results for different methodologies with a fraction γ of the sample containing a random walk process.
Notes:
(i) The values reported here show the fractions of the simulations that were either correctly classified either as a true RW or a true AR process.
(ii) γ represents the proportion of simulations that contain a random walk; the rest $(1-\gamma)$ have values of ρ that fall in the respective ranges indicated in the panels.
(iii) The ADF columns report results of the ADF test at a 5% level of significance. ADF is the Dickey-Fuller test without a constant term in the regression; ADF-μ consists of the case where the ADF test is misspecified as it includes a constant term;
(iv) KPSS is the Kwiatkowski et al. (1992) test conducted at a 5% level of significance.
(v) kNN are the classification results based on the k-Nearest Neighbors algorithm;
(vi) SVM are the classification results based on the support vector machines algorithm;
(vii) RF are the classification results based on the random forest algorithm.
(viii) For the ML and Bayesian methods, the threshold for classification here are based on a posterior probability ≥ 0.5.

processes include values of $\rho \in [0.9, 1)$, along with random walk processes where $\rho = 1$. First, as expected, we see that accuracy increases as the length of the time series,

T, increases for all methods employed. Moreover, all the methods have a value for accuracy greater than 0.5, which is what we would expect from a naive prediction (- the proverbial monkey randomly classifying each series as either stationary or a random walk). The predictions from the Bayesian approach are more accurate than both the ADF and KPSS approaches when T is small. This flips to some extent as T gets bigger as the accuracy of the ADF test improves when it is correctly specified (i.e. results in the ADF column relative to the ADF-μ column). However, the machine learning methods dominate all the classical and Bayesian approaches in the wide range nearly over all time series lengths, with accuracy ranging from nearly 70% when $T = 50$ to 91% when $T = 500$. Only the traditional ADF test (correctly specified to not include a constant term) fairs marginally better than the ML algorithms, also with an accuracy of 91% when $T = 500$, although the value is the same as the SVM method to 3 decimal places. Among the ML algorithms, the support vector machines appears to predict marginally better than the other two algorithms.

When we focus our attention to local-to-unity processes for the narrow range of values for ρ in panel B, we see a similar pattern of results emerge. The predictions from the ML algorithms clearly dominate the others, although the accuracy levels are not as high as in the wider range of values for ρ. Accuracy ranges from 56% when $T = 50$ to 77% when $T = 500$ for these ML methods. The mis-specified ADF test fairs the worst in this range, with accuracy equal to that of the naive predictor for the majority of the values of T. The Bayesian approach has accuracy levels similar to that of the ML methods when $T = 50$, although this does not improve much as T gets larger.[6]

[6]We also consider an alternative case where we set $\gamma = 0$ and consider a range of values: $[0.5 \leq \rho \leq 0.8]$. We then evaluate the predictions and determine the extent to which they indicate that the series are stationary. While we do not report the results here, we find that the ML methods have 100% accuracy with all predictions indicating a stationary process; the classical and Bayesian approaches have high levels of accuracy, that increase to 100% as T gets larger.

3.3. Precision, Recall and ROC Curves

In this section, we vary the proportion of random walk and stationary series within our hold-out sample. Given the tradeoff between precision and recall, we opt to see how these methods perform when we value both measures equally. One way to do this is to combine the two and compute their harmonic mean - the F_1 statistic:

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The values for the F_1 score for the random walk and stationary cases are reported in table 3. The results here show that the ADF test and the ML methods do equally well. In particular, the ML does better in predicting random walk processes when there are few random walk processes overall, i.e. for low values of γ; the ADF test and Bayesian tests do well in detecting random walk processes when the datasets contain a large number of the time series that are random walks. Similarly, for the stationary case, the KPSS test does well in detecting stationary processes when there are a large number of stationary series within a dataset, particularly when the value of ρ falls in the narrow range. This makes intuitive sense given the nature of these statistical approaches. The ML methods come in close behind them in both these scenarios.

Overall, we conclude that the ML approach here appears to be the most flexible in terms of prediction, since it yields the highest value of the F_1 score the most times across the range of values for γ in table 3. Finally, we note that no particular ML method consistently performs the best across the different values of γ. Instead, among the three ML methods that we consider, the kNN and the SVM methods generally appear to have the higher predictive power.

An alternative way to visualize the tradeoff between precision and recall is to plot the Receiver Operating Characteristic (ROC) curve. The ROC curve plots recall against the FPR (False Positive Rate = $\frac{FP}{FP+TN}$), depicting how the number of correctly classi-

Panel A: Wide AR Range
$[0.9 \leq \rho < 1)$

			Random Walk Predictions							Stationary Predictions						
T	γ	$1-\gamma$	ADF	ADF-m	KPSS	Bayesian	kNN	SVM	RF	ADF	ADF-m	KPSS	Bayesian	kNN	SVM	RF
100	0.1	0.9	0.254	0.199	0.291	0.265	0.411	0.405	0.385	0.418	0.196	0.217	0.350	0.373	0.381	0.380
100	0.25	0.75	0.503	0.427	0.342	0.506	0.625	0.625	0.604	0.518	0.239	0.447	0.503	0.614	0.618	0.607
100	0.5	0.5	0.739	0.678	0.358	0.724	0.758	0.765	0.756	0.534	0.243	0.702	0.593	0.776	0.777	0.761
100	0.75	0.25	0.884	0.849	0.356	0.844	0.810	0.819	0.818	0.557	0.262	0.866	0.630	0.847	0.842	0.824
100	0.9	0.1	0.941	0.932	0.373	0.897	0.826	0.835	0.841	0.549	0.258	0.937	0.641	0.872	0.866	0.851
500	0.1	0.9	0.609	0.447	0.519	0.518	0.651	0.632	0.594	0.738	0.676	0.295	0.702	0.648	0.706	0.665
500	0.25	0.75	0.807	0.700	0.616	0.756	0.832	0.826	0.799	0.852	0.781	0.551	0.823	0.815	0.848	0.824
500	0.5	0.5	0.912	0.860	0.635	0.886	0.901	0.911	0.900	0.906	0.827	0.773	0.869	0.902	0.907	0.895
500	0.75	0.25	0.950	0.933	0.657	0.942	0.930	0.946	0.938	0.920	0.845	0.902	0.888	0.937	0.930	0.919
500	0.9	0.1	0.965	0.960	0.659	0.961	0.942	0.958	0.950	0.928	0.848	0.948	0.893	0.945	0.937	0.927

Panel B: Narrow AR Range
$[0.98 \leq \rho < 1)$

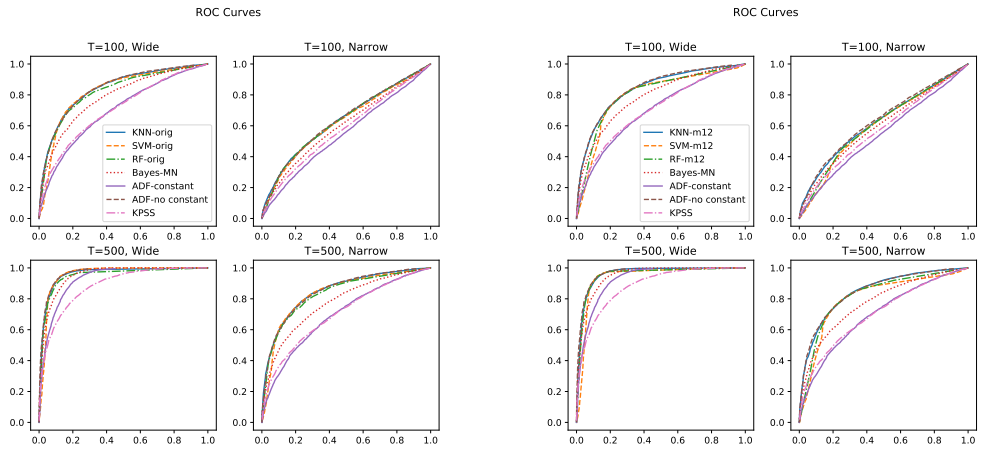
			Random Walk Predictions							Stationary Predictions						
T	γ	$1-\gamma$	ADF	ADF-m	KPSS	Bayesian	kNN	SVM	RF	ADF	ADF-m	KPSS	Bayesian	kNN	SVM	RF
100	0.1	0.9	0.188	0.183	0.157	0.192	0.263	0.258	0.245	0.106	0.065	0.190	0.160	0.228	0.230	0.226
100	0.25	0.75	0.401	0.400	0.165	0.402	0.411	0.411	0.417	0.170	0.109	0.411	0.257	0.441	0.439	0.438
100	0.5	0.5	0.661	0.654	0.186	0.643	0.538	0.551	0.568	0.152	0.101	0.671	0.290	0.656	0.646	0.631
100	0.75	0.25	0.845	0.841	0.189	0.814	0.582	0.602	0.633	0.150	0.110	0.846	0.311	0.772	0.756	0.737
100	0.9	0.1	0.926	0.924	0.192	0.885	0.626	0.647	0.672	0.165	0.110	0.931	0.325	0.832	0.816	0.787
500	0.1	0.9	0.252	0.197	0.291	0.216	0.427	0.411	0.388	0.428	0.198	0.216	0.328	0.367	0.387	0.386
500	0.25	0.75	0.506	0.421	0.343	0.453	0.632	0.634	0.631	0.496	0.217	0.446	0.347	0.606	0.619	0.611
500	0.5	0.5	0.739	0.679	0.361	0.699	0.756	0.766	0.759	0.532	0.247	0.703	0.371	0.782	0.779	0.761
500	0.75	0.25	0.880	0.850	0.373	0.858	0.802	0.823	0.822	0.558	0.242	0.865	0.392	0.857	0.848	0.838
500	0.9	0.1	0.940	0.931	0.363	0.936	0.819	0.842	0.846	0.556	0.238	0.938	0.382	0.886	0.873	0.855

Table 3: Values of the F_1 score for random walk and stationary predictions using different methodologies with a fraction γ of the sample containing a random walk process.
Notes: (i) The F_1 scores reported here show the harmonic mean of the precision and recall values for the random walk predictions, equally weighting each.
(ii) γ represents the actual proportion of simulations that contain a random walk; the rest $(1-\gamma)$ have values of ρ that fall in the respective ranges indicated in the panels.
(iii) The ADF columns report results of the ADF test at a 5% level of significance. ADF is the Dickey-Fuller test without a constant term in the regression; ADF-μ consists of the case where the ADF test is misspecified as it includes a constant term;
(iv) KPSS is the Kwiatkowski et al. (1992) test conducted at a 5% level of significance.
(v) kNN are the classification results based on the k-Nearest Neighbors algorithm;
(vi) SVM are the classification results based on the support vector machines algorithm;
(vii) RF are the classification results based on the random forest algorithm.
(viii) For the ML and Bayesian methods, the threshold for classification here are based on a posterior probability ≥ 0.5.

fied unit root predictions varies with the number of incorrectly classified AR predictions as we vary the classification threshold used to determine a positive outcome (i.e. a prediction of a unit root). Up until this point, we have set the classification threshold, $c^* = 0.5$ for the Bayesian and ML methods, and a 5% level of significance for the Classical tests. Here we allow it to vary.[7] If we were to set the threshold classification rate, $c^* = 1$, then we would be at the lower left point on the graph, i.e. (0,0). This is because we would not be able to identify any data points as a unit root, leading to no true positive or false positive outcomes. As we decrease c^*, we would begin to classify data points, leading to correct predictions (i.e. TP outcomes) as well as false positive outcomes. At a threshold of $c^* = 0$, we would end up at the upper right corner of the ROC curve with $TPR = FPR = 1$. Thus changes in c^* leads to movements along the ROC curve. The goal of an ideal classifier is to be in the upper-left corner, i.e. the (0,1) point in the ROC space. This would indicate perfect recall, with no false positive outcomes. If γ is small so that there are a large number of observations in the negative class, i.e. the majority of observations are truly stationary, a large change in the amount of false positives only leads to a small change in the FPR for the ROC curve. However, precision would change a lot more since it compares false positives to true positives by construction, rather than true negative outcomes (see Davis and Goadrich, 2006). The opposite would be true if γ were large and there were a large number of random walk series in the data, i.e. the FPR would change a lot, and precision would change only a little.

Figure 1a plots the ROC curves for the different approaches considered in this paper. For a given value of recall depicted on the y-axis, the approaches that do the best are the ones closest to the axis, i.e. with the smallest FPR. For predicting a unit root, we observe that the ML methods and the correctly specified ADF test (ADF no-constant)

[7]For our convenience, c^* in the Bayesian and ML context can be viewed as equivalent to the level of significance for the Classical tests.



(a) Original simulated data.

(b) Machine Learning methods using moments.

Figure 1: Receiver Operating Characteristic (ROC) Curves of ML vs Classical vs Bayesian Methods for predicting a unit root in the data.

dominate.[8] This is true for both the narrow and wide range of values of ρ considered, as well as for small and large values of T.

3.4. Robustness Check

For robustness, we allow for the features in the ML algorithms to be a transformation of the data. This check serves two purposes. First, it serves to validate the ML methodology being applied here to the shorter macroeconomic time series data: if we can treat each time series as a single observation or a set of draws from a particular DGP, then the individual values of a particular series is irrelevant. The informational content of that series will be reflected in the structural features of the data generating process, from the underlying autocorrelations in the data, to the moments of that series. Thus, we should get similar rates of prediction using transformations of the data as we do by looking at the particular realizations of that time series. If the rates of prediction are markedly different, then that may reflect a methodological error in

[8]It should be noted that while the ADF test does well and is close to the convex hull, this is only if we are willing to accept non-standard p-values as a criteria for making predictions. As such, if the goal is to maximize total accuracy, then it would mean that researchers would have to ultimately use a non-standard significance value in their tests.

the approach taken. Second, it may provide us with some insight as to the relevant number of moments that contain the information needed to help in the determination as to whether a particular process contains a unit root or whether it truly is a stationary process.

We implement this by transforming each time series into its mean and remaining $(n-1)$ central moments. We use these as the relevant features with which to train the algorithms. We do this transformation for both the training and hold-out datasets. Once trained, we make predictions as in the earlier section. The results for accuracy and the F_1 scores are reported in table 4. For a given range of values of ρ (i.e. either within the 'wide' or 'narrow' ranges), numbers in red indicate higher levels of accuracy or F_1 scores relative to the other ML methods in the respective row. Values that are bolded indicate the highest values relative to the corresponding machine learning results in tables 2 - 3; values that are bolded and in blue represent the highest values across all methods, including the classical and Bayesian approaches.

Panel A reports the results for accuracy when each of the simulated series are transformed into the first n moments of the data. For the most part, the accuracy of predictions using the transformed data are largely in line with the values found earlier using the untransformed data. However, as T gets large, the SVM approach with a larger number of moments ($n = 48$) performs better than the others, including the classical and Bayesian approaches for both the narrow and the wide range of values for ρ.

When we examine the F_1 scores for a random walk in panel B, we once again see that the rates of prediction are broadly in line with what was found earlier in table 3, when jointly considering precision and recall. The SVM approach with a large number of moments ($n = 48$) seems to do a better job of prediction here relative to the other ML methods for the wide range of values for ρ, particularly for large values of T.

Panel A: Accuracy

			Wide AR Range - $[0.9 \leq \rho < 1)$									Narrow AR Range - $[0.98 \leq \rho < 1)$								
			kNN			SVM			RF			kNN			SVM			RF		
T	γ	$1-\gamma$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$
50	0.5	0.5	0.681	0.683	0.683	0.664	0.679	0.681	0.681	0.681	0.682	0.555	0.556	0.556	0.534	0.546	0.547	0.555	0.554	0.554
100	0.5	0.5	0.763	0.763	0.763	0.761	0.764	0.766	0.762	0.764	0.764	0.603	0.602	0.602	0.572	0.591	0.598	0.601	0.600	0.600
250	0.5	0.5	0.866	0.865	0.865	0.863	0.866	**0.867**	0.865	0.864	0.865	0.679	0.678	0.678	0.675	0.680	0.681	0.684	0.686	0.685
500	0.5	0.5	0.906	0.906	0.906	0.910	0.910	**0.911**	0.910	0.907	0.909	0.769	0.769	0.769	0.765	0.772	**0.773**	0.770	0.770	0.771

Panel B: Random Walk

			kNN			SVM			RF			kNN			SVM			RF		
T	γ	$1-\gamma$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$
100	0.1	0.9	0.393	0.393	0.393	0.391	0.396	0.398	0.395	0.390	0.391	0.246	0.247	0.247	0.219	0.239	0.242	0.246	0.242	0.242
100	0.25	0.75	0.618	0.616	0.616	0.613	0.620	0.621	0.620	0.620	0.620	0.407	0.408	0.408	0.398	0.400	0.400	0.402	0.402	0.402
100	0.5	0.5	0.761	0.761	0.761	0.760	0.762	0.762	0.759	0.763	0.761	0.535	0.534	0.534	**0.571**	0.535	0.530	0.533	0.529	0.529
100	0.75	0.25	**0.826**	0.824	0.824	0.825	0.823	0.821	0.822	0.826	0.823	0.587	0.584	0.584	**0.665**	0.596	0.585	0.577	0.575	0.575
100	0.9	0.1	0.843	0.842	0.842	0.845	0.841	0.840	0.839	0.843	0.840	0.622	0.621	0.621	**0.710**	0.636	0.622	0.621	0.615	0.615
500	0.1	0.9	0.617	0.617	0.617	0.627	0.634	0.624	0.609	0.624	0.613	0.398	0.400	0.400	0.394	0.403	0.403	0.390	0.396	0.398
500	0.25	0.75	0.821	0.821	0.821	0.824	0.825	0.822	0.819	0.822	0.819	0.626	0.627	0.627	0.620	0.630	0.631	0.627	0.627	0.628
500	0.5	0.5	0.908	0.908	0.908	0.912	0.912	**0.913**	0.913	0.909	0.911	0.766	0.765	0.765	0.764	0.768	**0.770**	0.769	0.769	0.768
500	0.75	0.25	0.945	0.945	0.945	0.948	0.947	0.949	0.949	0.944	0.948	0.826	0.825	0.825	0.826	0.825	0.825	**0.829**	0.828	0.825
500	0.9	0.1	0.959	0.959	0.959	0.961	0.959	0.962	0.962	0.956	0.961	0.846	0.846	0.846	0.848	0.846	0.846	**0.853**	0.849	0.846

Panel C: Stationary Process

			kNN			SVM			RF			kNN			SVM			RF		
T	γ	$1-\gamma$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$	$m4$	$m12$	$m48$
100	0.9	0.1	0.379	0.381	0.381	**0.383**	0.379	0.379	0.374	0.379	0.377	0.222	0.221	0.221	0.214	0.219	0.221	0.224	0.224	0.224
100	0.75	0.25	**0.620**	0.617	0.617	0.615	0.615	0.615	0.617	0.620	0.619	0.437	0.436	0.436	0.404	0.427	0.434	0.429	0.428	0.428
100	0.5	0.5	0.764	0.765	0.765	0.762	0.767	0.769	0.765	0.766	0.766	0.653	0.652	0.652	0.573	0.635	0.648	0.651	0.652	0.652
100	0.25	0.75	0.832	0.831	0.831	0.828	0.835	0.836	0.835	0.832	0.834	0.768	0.769	0.769	0.673	0.750	0.764	0.768	0.770	0.770
100	0.1	0.9	0.853	0.853	0.853	0.851	0.857	0.858	0.856	0.851	0.853	0.821	0.820	0.820	0.710	0.801	0.817	0.821	0.823	0.823
500	0.9	0.1	0.710	0.708	0.708	0.718	0.712	**0.726**	0.724	0.693	0.718	0.392	0.392	0.392	0.388	0.390	0.391	**0.395**	0.392	0.389
500	0.75	0.25	0.843	0.844	0.844	0.850	0.848	**0.853**	0.850	0.842	0.849	0.616	0.615	0.615	0.613	0.616	0.617	0.616	0.615	0.613
500	0.5	0.5	0.903	0.903	0.903	0.908	0.908	**0.908**	0.907	0.905	0.906	0.772	0.772	0.772	0.766	0.775	0.776	0.771	0.772	0.774
500	0.25	0.75	0.927	0.927	0.927	0.929	**0.930**	0.927	0.925	0.929	0.926	0.840	0.840	0.840	0.834	0.842	0.843	0.837	0.838	0.841
500	0.1	0.9	0.932	0.932	0.932	0.935	0.937	0.933	0.929	0.935	0.930	0.863	0.863	0.863	0.857	0.865	0.865	0.855	0.859	0.863

Table 4: Results for predictions with Machine Learning algorithms using moments of the data as features with a fraction γ of the sample containing a random walk process. Notes: (i) Panel A reports the accuracy results. Panels B and C report the F_1 scores, which show the harmonic mean of the precision and recall values for the random walk predictions, equally weighting each.
(ii) γ represents the actual proportion of simulations that contain a random walk; the rest $(1-\gamma)$ have values of ρ that fall in the respective ranges indicated in the panels.
(iii) mX represents the number of moments, where $X = \{4, 12, 48\}$
(iv) The highest value among the ML methods within the respective ranges for ρ are represented in red for each value of T and γ in each row. Bolded values indicate that these are the highest values relative to the corresponding machine learning results using untransformed data in tables 2, and 3; values in blue represent the highest values across all methods, including the classical and Bayesian approaches.
(v) kNN are the classification results based on the k-Nearest Neighbors algorithm;
(vi) SVM are the classification results based on the support vector machines algorithm;
(vii) RF are the classification results based on the random forest algorithm.
(viii) For the ML methods, the threshold for classification here are based on a posterior probability ≥ 0.5.

It performs better than the earlier ML predictions with untransformed data, or even across all methods. It also does well in the narrow range across different values of T, particularly as the proportion of unit root processes in the data increase (i.e. for high values of γ). Only the RF approach with fewer moments yield higher F_1 scores in the narrow range when nearly all the series in the data contain a unit root.

In panel C, we see once again that the SVM approach predicts better than the other ML approaches when it comes to stationary processes as well. In fact, for large T, the SVM approach with $n = 48$ does very well in correctly predicting stationary processes particularly when there are relatively few stationary processes in the sample, even outperforming the classical and Bayesian approaches. Overall, the results here show that the ML models trained using moments as features do as well as those using untransformed simulated data, and often perform better.

4. Empirical Results

In this section, we apply the different approaches to macroeconomic data and compare the predictions. We use data from the FRED-MD and FRED-QD databases, along with eighteen daily exchange rate data series, two commodity price series, and a stock price series.[9] We examine the T most recent set of observations where $T = \{100, 500\}$ for two versions of the data. The first is the original (untransformed) data that may contain stationary series, along with nonstationary data that may be trend stationary, difference stationary, or be rendered stationary after an appropriate transformation. Such transformations are suggested in the 'TCODE' column in the FRED-MD and QD databases to render them equivalent to the series considered in Stock and Watson (2012). The second version of the data is one where we apply the transformations suggested in the FRED-MD and QD databases to that data, along with other appropriate

[9]Details on the data may be found in the accompanying technical appendix and on the authors' website post publication.

transformations to render the remaining 21 series approximately stationary, so that they are either an I(1) or an I(0) process. For these 21 high frequency data series, we take the log of each series and then de-mean them.[10] Ultimately, this yields 395 and 148 untransformed data series for $T = \{100, 500\}$ respectively, and results in 416 and 169 data series for the transformed series given the two types of transformations that we consider for the 21 high frequency series.

It is useful at this juncture to note that the point of this empirical exercise is not to ascertain the optimal way to characterize the DGP for each of the data series. A unit root process may not be the best representation of the DGP for any of the series in our dataset, particularly in the untransformed data. Instead, the point of this empirical exercise is to adopt an agnostic approach and apply the different methodologies examined in the paper to see whether they predict a unit root for any of the series, and to compare the predictions from the different approaches given our earlier findings with simulated data. We do this for two reasons. First, if any of the data may be represented as a nonstationary process, then the prediction of a unit root process for an untransformed data series may help to provide evidence that the data should be modeled as a nonstationary process, and transformations that render the data stationary may be appropriate. Second, we want to try and implement transformations of the data that will yield an approximate unit root process - a process that may ultimately either be I(0) or I(1) - so that we can compare the predictions from all these different methods. That is why we also make predictions on the transformed data: it provides us some insight into whether these transformations are successful in rendering the data stationary.

While prediction from the Bayesian and ML approaches is fairly straightforward, an

[10]For the transformed dataset, we consider two sets of transformations. The first is where we detrend and de-mean the log of each series. The second is where we de-mean the first difference of the log of each series.

additional complication arises for the classical tests. Typically inference and prediction in classical tests are based upon the result of a hypothesis test. Once we determine the value of the relevant test statistic, for a 'given level of significance', we either fail to reject the null hypothesis, or find evidence in favor of the alternative. What is well known, and corroborated by our earlier analysis using the ROC curves, is that a tradeoff exists: as the true positive rate increases, so does the false positive rate. Consequently in order to be able to compare predictions from the classical tests as apples-to-apples with the predictions from the Bayesian and ML approaches for actual data, we use the critical values simulated from our Monte Carlo to test whether there is evidence of nonstationarity in the empirical series. This means that our tests do not assume any autocorrelation structure. Given the agnostic nature of our study, we opt not to use the conventional 5% level of significance. Instead, we use the critical values for the corresponding optimal p-value based on the ROC analysis.[11]

For each series, we first fit the data to an AR(1) model and obtain the value for the autoregressive parameter. Then, we extract the critical value from the particular ADF or KPSS distribution ($T = \{100, 500\}$; wide/narrow cases). The distributions for the KPSS are based on the null of a stationary process, which is categorized as either wide or narrow in our simulations. Because the ADF is based on the null of a random walk, this categorization does not apply. Nevertheless, for both ADF and KPSS, the optimal p-value varies and depends on the degree of persistence of the series.

[11] The optimal p-values are those that achieve the highest accuracy given our 50% AR(1) and 50% random walk simulated sample. These are indicated in the table below:

	T = 100, Wide	T = 100, Narrow	T = 500, Wide	T = 500, Narrow
ADF (no constant)	0.235	0.495	0.065	0.26
ADF (w/ constant)	0.42	0.605	0.095	0.375
KPSS	0.275	0.3	0.22	0.245

For example, consider a series with $T = 100$ that has an estimated AR parameter of 0.95.[12] We apply the three classical tests and derive three test statistics: (1) ADF without constant, (2) ADF with constant, and (3) KPSS. Then, we select the two specific ADF distributions for $T = 100$, and the KPSS distribution given $T = 100$ and the "wide" range. Tests are completed by comparing the test statistics and the critical value picked, based on the optimal p-value from the table in footnote 11. We

Data	Untransformed (Unit Root)		Transformed (Stationary)	
T	100	500	100	500
Panel A: Number of predictions				
Classical (C)	327 (0.83)	133 (0.90)	158 (0.38)	74 (0.44)
Bayesian (B)	343 (0.87)	123 (0.83)	158 (0.38)	86 (0.51)
Machine Learning (ML)	316 (0.80)	102 (0.69)	397 (0.95)	166 (0.98)
N	395	148	416	169
Panel B: Agreement on predictions across approaches				
All three	267 (0.71)	86 (0.61)	130 (0.32)	61 (0.36)
C&B	44 (0.12)	37 (0.26)	6 (0.01)	2 (0.01)
C&M	14 (0.04)	9 (0.06)	22 (0.05)	10 (0.06)
B&M	16 (0.04)	0 (0.00)	20 (0.05)	23 (0.14)
C only	2 (0.01)	1 (0.01)	0 (0.00)	1 (0.01)
B only	16 (0.04)	0 (0.00)	2 (0.00)	0 (0.00)
M only	19 (0.05)	7 (0.05)	225 (0.56)	72 (0.43)
Total	378	140	405	169

Table 5: Number of series with predictions based on majority rule within a given approach (i.e. Classical, Bayesian and Machine Learning) in the data. Numbers for untransformed (transformed) column represent predictions for unit root (stationary) processes. Numbers in parentheses indicate proportions. N represents the number of series in the dataset.

report the results in table 5 and in the Venn diagrams in figure 2. In table 5, we broadly classify predictions as being nonstationary based on majority rule predictions

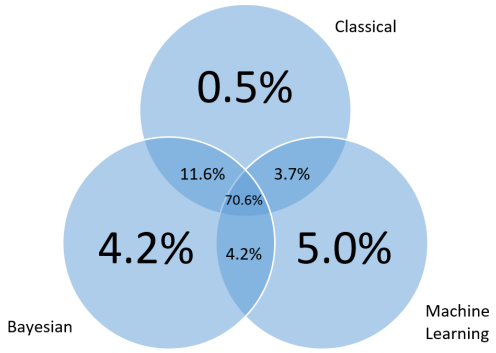
[12] Admittedly, there are AR estimates below 0.9. The p-values derived from our simulations would not be optimal in this situation.

of a unit root from a given approach (i.e. from within the Classical, Bayesian and ML approaches). There are six classical tests and twenty-four ML algorithms applied to the data sets. We assume the ML collectively identify a series to be a unit root or stationary process when more than twelve of them does so.[13] This is a simple majority rule. It is more nuanced when it comes to the classical tests. Since our optimal p-value is above 0.05, the classical tests are less likely to reject the null hypothesis than when they are conventionally used. However, the four ADF and the two KPSS tests have opposite null hypotheses. In order to account for both types, we avoid the simple majority rule and allow for a 50-50 equal weight between the two, or a weight of 12.5% on each of the four ADF tests and a weight of 25% on each of the two KPSS tests. When the ADF tests and the KPSS tests do not agree, this modified rule reduces results being dominated by the ADF which hold the majority.

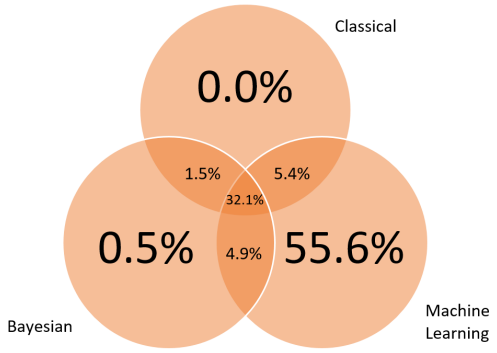
We find that the three approaches broadly predict similar numbers of nonstationary series within the untransformed dataset. Here, where we might expect a larger proportion of the dataset to potentially contain nonstationary series a priori, these approaches predict that somewhere between 80% - 87% of the macroeconomic time series within the dataset may contain a unit root for $T = 100$, and somewhere between 69% - 90% for $T = 500$.

When comparing the predictions across the approaches, we see in panel B of table 5 that all three approaches jointly predict that 267 of the 395 untransformed data series are nonstationary when $T = 100$, and approximately 86 out of the 148 for $T = 500$. We see that there is broad agreement in the widespread predictions of nonstationary behavior for the untransformed data in figures 2a and 2c. Approximately 71% of the series classified as nonstationary are done so jointly by the three approaches for $T = 100$ and approximately 61% for $T = 500$. When also including pairwise agreements, we find in

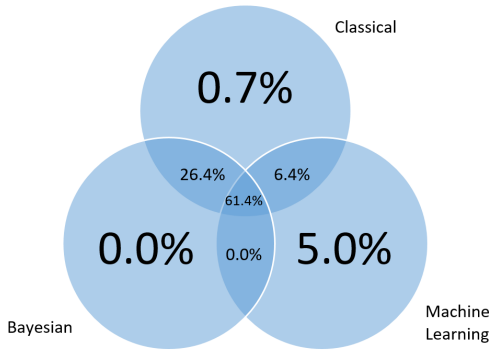
[13] This is similar to implementing an ensemble approach with a set of weak learners.



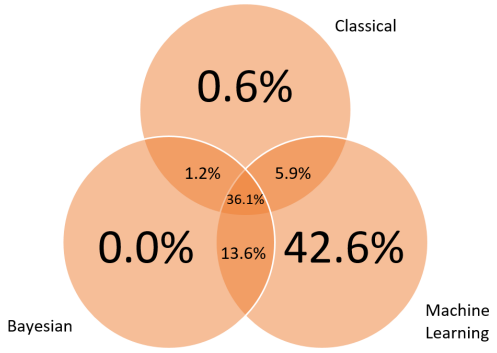
(a) Unit root; $T = 100$, Untransformed data.



(b) Stationary; $T = 100$, Transformed data.



(c) Unit root; $T = 500$, Untransformed data.



(d) Stationary; $T = 500$, Transformed data.

Figure 2: Agreement on predictions in the data by the Classical, Bayesian and Machine Learning approaches.

total that approximately 91% - 94% of the time series considered are nonstationary, a result that is consistent with the findings of Nelson and Plosser (1982). Using $T = 500$ as an example for the ML approach, the largest posterior probabilities for a unit root mostly fall in the range of 0.72 - 0.77 for the RF and SVM methods, whereas the largest probability for a unit root process using the kNN approach has a marginally lower prediction probability of around 0.6. The posterior probabilities for the Bayesian approach are considerably higher and closer to 1. The posterior probabilities are higher for both the Bayesian and ML in the $T = 500$ case, relative to the $T = 100$ case.

When looking at the transformed dataset, where the transformations are purported to yield a stationary series, we find that there is a sizable disagreement concerning the two transformed data sets. The ML algorithms overwhelmingly identify the majority of the series to be stationary for both $T = 100$ and $T = 500$, while neither the Classical nor

the Bayesian identify half as much. Although small sample bias in Classical tests may contribute to the divergence for $T = 100$, it is hard to extend this explanation to the $T = 500$ data set. In particular, majority rule prediction for the ML approach predicts that only 5% of the transformed series contain a unit root for $T = 100$ and only 2% for $T = 500$. The Classical and Bayesian approaches predict a much higher portion of the series contain a unit root, indicating that approximately 62% of the series in the data contain a unit root for $T = 100$, to somewhere between 49% - 56% for $T = 500$. There appears to be a lot of agreement between the classical and Bayesian predictions, where the unit root predictions coincide 75.5% of the time for the $T = 100$ case and 84.7% of the time for $T = 500$ case. However, there is little overlapping agreement between the ML and either the Classical or Bayesian predictions in the transformed case for predicting unit roots, since the ML approach predicts that the vast majority (95% - 98% for $T = 100$ and $T = 500$ respectively) of the series are stationary.

This can be seen by focusing on unit root predictions for the Classical and Bayesian approaches the $T = 500$ case as an example. We see a lot of agreement between the Classical and Bayesian approaches. With that being said, ML identification is weak and the corresponding posterior probabilities for the ML methods for the transformed data are mostly in the range of 0.3 to 0.6. In essence, the ML methods are indicating there are a few series that are a coin-toss: they are unable to sufficiently distinguish between the processes for them given the data. For the remainder of the series, they place a lower probability of a unit-root type process, while the posterior probabilities for the Bayesian approach indicate a strong preference for a model with a unit root. At the same time, the Bayesian and Classical approaches jointly agree on 84.7% of the series that they continue to classify as nonstationary after the respective transformations.

When we only consider the predictions for stationary processes among the three approaches, we see that there is quite a lot of agreement. Once again, focusing on the

$T = 500$ case as an illustration, we can be seen in panel B of table 5 and in figure 2d that the three approaches jointly agree on 36.1% of the series as being stationary. There is also some agreement between the Bayesian and ML approaches, and the Classical and ML approaches, but little to no agreement between the Bayesian and Classical approaches only.

In our earlier results, we found that if there was class imbalance, then in the case where there were a large number of unit root processes, the ADF and Bayesian approaches correctly yielded the highest predictions of a unit root, with the ML approach coming in very close behind. Conversely, when the data was stationary, the KPSS test correctly yielded the highest predictions of a stationary process with the ML approach coming in a close second. This helps to reconcile the results for the untransformed data, where we see broad agreement for individual series that those macroeconomic data are not stationary.

In the case of the transformed dataset, this is harder to do, particularly for those series where there is disagreement between the different approaches for the unit root prediction. The series in our Monte Carlo study are specified in an AR(1) format. The data generating process of the empirical data is less clear. Any transformation of these series may make the matter more complicated, and as such, the differences between predictions are difficult to reconcile and interpret. We do find a fair amount of agreement among the three approaches, where they jointly agree that approximately half of the series in the dataset are rendered stationary after their respective transformations. However, this also means that approximately half of the transformed series may be behaving in some 'non-stationary' fashion.

As we acknowledge earlier, a unit root process may not be an appropriate characterization of the data generating process, even after the transformations. Thus we view the disagreements between the approaches in a positive light, since they indicate to

us that each of these series predicted to be nonstationary would need to be examined more closely in order to be able to speak to this when disagreement occurs. Since the ML algorithms do not involve any model specification, it is possible for them to pick up properties among the transformed data that the Classical and the Bayesian approaches might miss. For these latter approaches, we would have to correctly (or incorrectly) specify the data generating process post-transformation, and mis-specification may be an issue. At the same time, disagreement may occur if the ML methods are picking up something else - for example the effects of outliers or 'influential' observations, or some higher order complexity not readily apparent in our depiction of the DGP. We view these results as raising a question for future research about the transformed data series.

5. Brief Discussion

In this section, we briefly discuss some of the advantages of the ML approach, along with the limitations that we ran into in implementing our proposed approach to apply ML to macroeconomic time series data. One key advantage of the machine learning methods that have been proposed in the literature is their ability to capture the complexity of an underlying system. The ML approach uses large amounts of data to help the algorithm learn the mapping from the set of features or covariates to the observation at hand. This mapping may be simple, nonlinear, or highly complex. It is the training of the algorithms that uncovers this mapping, since the ML approach is agnostic by its very nature - we do not 'tell' the algorithms what we believe the relationship to be. Instead the data informs the algorithm. This is where big data is important. In general, the greater the amount of data used to train the algorithms, the better the predictions.

The mapping we investigate in this paper is relatively simple on the scale of things, with a low degree of complexity. After all, we are trying to recover one structural parameter

in the underlying time series within a linear model, and we do this fairly successfully with only 100000 observations used to train the algorithms. Additional data to train the models would then help with the predictions since we are able to better gauge the true generalization error by using additional training data and cross-validation to avoid overfitting. Although the bias is never fully eliminated, the cross validated loss would be lower with additional observations in the training set. This leads to one limitation of the ML methodology, which is the computational cost. For some of the ML methods, it takes a lot of time to train the ML models. For example, the SVM algorithm took approximately 8 to 10 hours to train for our set of $T = 500$ time series model, and this was only for one parameter.[14] Although we could have trained the ML algorithms with 1 billion observations instead of 100000, the time taken to do so would have been substantially higher. However, once the models have been trained, predictions take very little time. Another limitation of the ML approach is that the size of the training sample has to grow in magnitude of order $O((Tm)^r)$ as the number of parameters, r, increases. This also increases the computational burden.

One additional limitation of our proposed methodology here is that the ML algorithms are trained for a time series of a specific length. Currently, we train the ML algorithms for a specific value of T, i.e for $T = \{50, 100, 250, 500\}$. So if we have initially trained a ML model for $T = 500$, and if we now have a time series of length $T = 501$ to make predictions on, we would have to re-train the algorithms with observations in the training sample of length $T = 501$. The reason for this is because an observation in our methodology consists of an entire time series along its class label, i.e. $(\mathbf{x}, \mathbf{Y})$. When we train the algorithms, we use draws from the same DGP. However, this is not a serious limitation since as we can always transform the each time series into its n moments and use that to make predictions as in section 3.4.

[14]This was done in Matlab with an Intel®Xeon®E3-1535M CPU with two cores running on a Windows 10 operating system with 16GB of RAM.

At the end of the day, the key advantage of the ML approach is that it is agnostic about the underlying data generating process, as compared to the other econometric methods. We do not have to specify a model with the nonparametric methods used here - we simply show the algorithms which observations in the training sample have unit roots and which do not. These are then used to make predictions and inferences on an independent dataset. Consequently, the ML approach may accurately reflect the relative ignorance of the researcher. If the research question being investigated can be cast as a prediction problem, the methodology is easy to implement, and as our results show, appear to be very flexible. These benefits make the approach appealing despite the limitations indicated above.

6. Conclusions

In this paper, we are able to demonstrate how machine learning methods may be applied to address macroeconomic questions by using these methods on a classic problem: the issue of unit roots in macroeconomic time series. We frame the problem of identifying a unit root within a particular time series as a classification problem, and examine the predictive accuracy of Classical, Bayesian and Machine Learning methods in uncovering them using supervised learning. We use a Monte Carlo approach to train the machine learning algorithms on an independently simulated dataset, which we then use to predict make predictions on a hold-out set. These predictions are compared to those from the Classical and Bayesian methods in a simple horse-race fashion, examining their predictive accuracy.

The machine learning methods used in this paper perform relatively better than the Classical and Bayesian methods when the length of the time series is smaller, and when the there are approximately equal proportions of random walk processes and stationary processes in the dataset. The ADF and Bayesian tests tend to predict better in longer time series data, and when the majority of time series in a given data set are a random

walk. Similarly, the KPSS test performs the best when the majority of time series are stationary. In both these cases, the ML methods perform nearly as well as the leading predictors.

Aside from the relative predictive performance of these approaches, this project helps time series econometricians understand a new approach in the application of ML on to a familiar issue in the discipline. The methodology we propose here may be applied to other macroeconomic questions that seek to uncover the value of structural parameters from the data. Examples include identifying structural breaks in time series, to determining the type of break present, or the presence of threshold effects in the data among others. We leave this to future research.

References

Athey, S. (2015). Machine learning and causal inference for policy evaluation. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 5–6. ACM.

Athey, S., Bayati, M., Doudchenko, N., Imbens, G., and Khosravi, K. (2018). Matrix completion methods for causal panel data models. Technical report, National Bureau of Economic Research.

Athey, S., Imbens, G., Pham, T., and Wager, S. (2017). Estimating average treatment effects: Supplementary analyses and remaining challenges. *American Economic Review*, 107(5):278–81.

Athey, S. and Imbens, G. W. (2017). The state of applied econometrics: Causality and policy evaluation. *Journal of Economic Perspectives*, 31(2):3–32.

Athey, S. and Imbens, G. W. (2019). Machine learning methods that economists should know about. *Annual Review of Economics*, 11.

Athey, S., Tibshirani, J., Wager, S., et al. (2019). Generalized random forests. *The Annals of Statistics*, 47(2):1148–1178.

Belloni, A., Chernozhukov, V., Fernández-Val, I., and Hansen, C. (2017). Program evaluation and causal inference with high-dimensional data. *Econometrica*, 85(1):233–298.

Berger, J. O. and Yang, R.-Y. (1994). Noninformative priors and bayesian testing for the AR(1) model. *Econometric Theory*, 10(3-4):461–482.

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

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240.

Dickey, D. A. and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366a):427–431.

Giusto, A. and Piger, J. (2017). Identifying business cycle turning points in real time with vector quantization. *International Journal of Forecasting*, 33(1):174–184.

Kwiatkowski, D., Phillips, P. C., Schmidt, P., and Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics*, 54(1-3):159–178.

Marriott, J. and Newbold, P. (1998). Bayesian comparison of arima and stationary arma models. *International Statistical Review*, 66(3):323–336.

Mullainathan, S. and Spiess, J. (2017). Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, 31(2):87–106.

Nelson, C. R. and Plosser, C. R. (1982). Trends and random walks in macroeconmic time series: some evidence and implications. *Journal of Monetary Economics*, 10(2):139–162.

Sims, C. A. and Uhlig, H. (1991). Understanding unit rooters: A helicopter tour. *Econometrica*, pages 1591–1599.

Stock, J. H. (1994). Deciding between i (1) and i (0). *Journal of Econometrics*, 63(1):105–131.

Stock, J. H. and Watson, M. W. (2012). Disentangling the channels of the 2007-2009 recession. *Brookings Papers on Economic Activity, May*, pages 81–135.

Uhlig, H. (1994). On jeffreys' prior when using the exact likelihood function. *Econometric Theory*, 10:633–644.

Wager, S. and Athey, S. (2018). Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37.