Roots from Trees – A Machine Learning Approach to Unit Root Detection

A Preprint

Gary Cornwall *
Office of the Chief Economist
Bureau of Economic Analysis
Suitland, MD USA
Gary.Cornwall@bea.gov

Jeffrey Chen
Bennett Institute for Public Policy
University of Cambridge
Stockholm, Sweden
contact@jeffchen.org

Beau Sauley
Department of Economics and Finance
Murray State University
Murray, KY USA
bsauley@murraystate.edu

June 30, 2021

Abstract

In this paper we draw inspiration from the ensemble forecasting and model averaging literature and use a gradient boosting algorithm to exploit variation between test statistics used to determine if a series contains a unit root. The result is a *pseudo*-composite ML-based test for unit roots which is approximately seventeen percentage points more accurate than the next best traditional test. This increase in accuracy comes from a thirty-seven percentage point increase in sensitivity (empirical power). Through a train-validation framework this method allows for control over Type I error rates and the gains in power come with little variation in specificity (empirical size). Additionally, the proposed method requires fewer assumptions regarding the originating data generating process and closes off an additional error path for unit root testing, that of model misspecification.

*The authors would like to acknowledge Peter C.B. Phillips, Tara Sinclair, Marina Gindelsky, Scott Wentland, Jeff Mills, and Olivier Parent for their helpful comments. The views expressed here are those of the authors and do not represent those of the U.S. Bureau of Economic Analysis or the U.S. Department of Commerce.

Keywords Integrated Processes · Forecast or Model Averaging · Time Series · Machine Learning

1 Introduction

The identification of time-series which contain a unit root has important implications for data users and researchers. Granger and Newbold (1974) demonstrated this through the use of Monte Carlo simulation and showed that failure to identify the presence of a unit root can lead to 'nonsense regressions'. Over the next five decades there was a great deal of research into test statistics which, under various conditions, improved the ability for a researcher to identify time-series containing a unit root (see Dickey and Fuller (1981), Phillips and Perron (1988), Schmidt and Phillips (1992), Elliott et al. (1992), Zivot and Andrews (2002), and Jansson and Nielsen (2012) among many others). Despite the bevy of options available to researchers interested in time-series analysis, there remains a great deal of skepticism regarding available tests stemming primarily from evidence of low power when the root of an autoregressive polynomial is near, but not equal to, one (Ng and Perron, 2001).[2]

In this paper, we propose a solution which draws on inspiration from the model and forecast averaging literature (see Elliott and Timmermann (2004); Timmermann (2006), combined with modern computational power and classification algorithms to form a *pseudo*-composite unit root test. We show – under a simulation environment common to the unit root literature – that our proposed method is more powerful at differentiating near unit roots from unit roots. This increase in power is approximately 37% and comes from the exploitation of both variation between the test statistics themselves, and those provided by other common time-series characteristics (Wang et al., 2006, 2009).

The testing method proposed herein takes advantage of a well-specified hypothesis. In the case of unit roots, we typically define the hypotheses as $H_0 : \phi = 1$ and $H_1 : |\phi| < 1$ where ϕ is the parameter of interest in a lag polynomial. From these definitions, one can generate an arbitrarily large set of time-series that meet the conditions of each hypothesis. In other words, it is possible to construct a *digital twin* of each the null and alternative distributions. Importantly, in the unit root literature, we rely primarily upon simulated times series to establish the null distribution so there is both an established methodology and extensive use cases for such simulations (MacKinnon, 1996, 2010).

In many cases, subsequent research has taken an original unit root test and developed new test statistics which relax assumptions, change the object of measurement, or seek to make an existing statistic more robust. Oftentimes the new test has some attractive feature that improves upon the original design, *e.g.,* more power at a lower number of observations. However, in some cases, this tinkering means the two tests can disagree on the disposition of the null as we will show in Section 4. This happens because classification

[2]The detection of a unit root is a critical function in the modeling of time-series data. As a result, more accurate tests for this data feature are important to statistical agencies such as the U.S. Bureau of Economic Analysis, U.S. Census Bureau, and Bureau of Labor Statistics among many others.

from the first test is unlikely to be a perfect subset of the second. This means that, depending on which test a practitioner uses, there may be a reasonable disagreement to be had regarding that aspect of the data. This reasonable disagreement can be disastrous for not only the practitioner, but also the scientific community which builds upon that result. We will show in Section 4 that there is a great deal of variation between these test statistics which means that this "reasonable disagreement" is likely to occur between tests and practitioners or researchers. As a result, the conventional wisdom has become "When in doubt, assume a unit root." which may or may not be appropriate depending on the context.

The remainder of this paper is structured as follows. In Section 2 we describe the unit root problem and analyze the accuracy of current hypothesis tests. In Section 3 we show how common tests for a unit root are equivalent to a weak learner. This equivalence facilitates our use of an ensemble learner such as the gradient boosted machine to exploit variation between test statistics. Section 4 lays out an ML-based unit root test which delivers marked improvement in test accuracy. Section 5 applies these advancements to macroeconomic time-series that have long been debated in the literature. Section 6 concludes and offers avenues for additional research.

2 Unit Roots

Of the statistical tests in time-series econometrics perhaps none are more visible nor have attracted as much research over the previous five decades as that of the unit root. Its importance is made obvious by Granger and Newbold (1974), which outlines the concept of spurious regression, showing that regressions involving random walks can create relationships between otherwise independent series. Subsequently, a great deal of research capital has been spent on developing and improving tests for unit roots and stationarity with some of the more well-known including Dickey and Fuller (1981), Phillips and Perron (1988), Kwiatkowski et al. (1992), Elliot et al. (1996), Ng and Perron (2001), and Bai and Ng (2004) among many others.

Herein we focus on three unit root structures that are used throughout the related literature. Let y_t be an autoregressive time-series generated such that one of the following is true,

$$y_t = \gamma + \phi y_{t-1} + \delta t + \epsilon_t, \tag{1}$$

$$y_t = \gamma + \phi y_{t-1} + \epsilon_t, \tag{2}$$

$$y_t = \phi y_{t-1} + \epsilon_t, \tag{3}$$

indexed by $t = 1, \ldots, T$. In Equation 1 we have included a drift term, γ, as well as a time trend, δt, and in all cases $\epsilon_t \sim WN$. Equations 2 and 3 omit the time trend and both the time trend and drift term respectively.

Equation 3 is the simplest case and often is the first used to introduce the concept of unit roots in econometric texts. Alternatively, one could write this process using the lag operator as $(1 - \phi L)y_t = \epsilon_t$ such that

$Ly_t = y_{t-1}$. We will assume that $\epsilon_t \sim N(0, \sigma_t^2)\ \forall t$ and that $\sigma_1^2 = \ldots = \sigma_T^2$. The polynomial, $(1 - \phi L)$ has a root of $1/\phi$, and if $|\phi| < 1$ then y_t is considered stationary. This process is one which does not depend on t, and over the long run has a vanishing memory. If $\phi = 1$ then the series is a random walk and contains a unit root, a potentially dangerous attribute that needs to be identified.

When transformed into a difference equation this leads to a hypothesis whereby the null is $H_0 : \phi = 1$ and the alternative is $H_1 : |\phi| < 1$. This structure is used in Dickey and Fuller (1981), Phillips and Perron (1988), and Elliot et al. (1996) among others. Since many of these tests are effective in rejecting the null when $|\phi| \to 0$ we will focus our efforts on a very narrow subset of the domain such that our null remains $H_0 : \phi = 1$ while the alternative is $H_1 : |\phi| \in (.90000, .99999)$. It is at this interface between unit roots and *near* unit roots that a test's worth is proven.

Testing for a unit root requires one to make two decisions. First, one must choose a data generating process that best represents the data. This choice dictates the distribution of the null and resulting critical values for the test itself. It should be apparent that there is ample room for human error – if an alternative structure is a better approximation of the true data generating process – then a practitioner can make an error by choosing the wrong framework for the null distribution. Second, once a test scenario is established, the practitioner must choose some level of long-run Type I error rate, α, they are willing to accept with the most common choices $\alpha \in \{0.01, 0.05, 0.10\}$. This two-choice framework leads to intra-test variation. The choice of Equation 1 with $\alpha_1 = 0.05$ may lead to a different result than using Equation 2 with $\alpha_2 = 0.05$. Moreover, depending on the selected test (*e.g.,* Augmented Dickey-Fuller versus Phillips-Perron) there can be between test variation, that is one test may reject the null while the other fails-to-reject it.

We exploit this variation by employing a classification algorithm as a mapping function in place of the traditional null distribution. Limiting design to the three cases outlined above results in a clean simulated environment that is well known to the unit root literature and allows us to evaluate not only our classification results and ensure the data being generated fulfills some, if not all, of the null assumptions.[3] While there are a bevy of tests to choose from, we will use nine different tests for unit roots, many of which have the structure outlined above with respect to the null and alternative. These tests include the ADF (Dickey and Fuller, 1981), PP (Phillips and Perron, 1988), KPSS (Kwiatkowski et al., 1992), PGFF (Pantula et al., 1994), Breit (Breitung, 2002), (Breitung and Taylor, 2003), ERS-d and ERS-p (Elliot et al., 1996), URSP (Schmidt and Phillips, 1992), and URZA (Zivot and Andrews, 2002).

[3]The simulation environment can be arbitrarily expanded to include other DGPs that are pertinent to the problem at hand. This study's environment reflects most of the DGPs that are relied upon in the literature when evaluating univariate unit roots.

2.1 Accuracy of Current Tests

Simulation studies have become rather ubiquitous in the unit root literature, in part because the null distribution for many of the tests has no analytic form. As a result, critical values are often obtained through simulation (see MacKinnon (1996, 2010) for example) and are thus dependent on the assumptions applied to the simulation environments under which they were constructed. In order to assess the performance of current unit root tests, we simulate a large number of time-series, apply tests to these simulations, then compare their ability to detect unit roots.

To that end we will generate $M = 500,000$ series, of which $350,000$ and $75,000$ are reserved for training and validating ML algorithms, respectively. The remaining $75,000$ series will serve as the basis of accuracy comparisons for all subsequent sections. To begin let $\pi_u \sim U(0,1)$ such that,

$$U = \begin{cases} +1: \ \pi_u \geq 0.50 \\ -1: \ \pi_u < 0.50 \end{cases}, \tag{4}$$

where $U = +1$ denotes a series with an Unit Root and $U = -1$ a Near Unit Root. Conditional upon the series containing a Unit Root we generate data from Equation 1 , 2 , and 3 where $\phi = 1$. If $U = -1$ then we draw ϕ uniformly over the interval $(0.9000, 0.9999)$.[4] For all simulated series $\epsilon_t \sim N(0,1)$.[5] Since the frequency of a series is orthogonal to its unit root status, we assume all series are of monthly frequency.[6] Each series is variable in length with the number of years drawn uniformly over the interval $(5, 50)$. From each series we calculate the appropriate test statistic (*e.g.* Augmented Dickey-Fuller) and record the test's conclusion assuming $\alpha = 0.05$ for all tests.[7]

[Table 1 about here.]

To evaluate efficacy of each test, we compare its performance based on standard measures from the machine learning literature, all of which are calculable from a 2×2 confusion matrix (Hastie et al., 2009; Kuhn et al., 2013; Tharwat, 2018). As outlined in Table 1, a confusion matrix produces the basic building blocks for a rich assortment of accuracy measures that emphasize different concepts, such as accuracy (ACC), sensitivity (SEN), specificity (SPE), positive predictive value (PPV), negative predictive value (NPV), F-Measure (various weights), and the Matthew's Correlation Coefficient (MCC). Each measure allows a practitioner to optimize

[4] We have examined other distributions which govern the draw of ϕ in the case of a Near Unit Root including $U(0.00, 0.99)$, $B(\alpha = 8, \beta = 1)$, and $B(\alpha = 2, \beta = 2)$. We have also varied relative weights given to Unit Roots and Near Unit Roots by changing the cutoff for π_u. All additional results and code will be available on our website.

[5] We have also varied the distribution of the error term by using $U(-1, 1)$ with similar results.

[6] We have varied the frequency in earlier iterations of this process but have left it as monthly in order to limit parameters in the simulated environment.

[7] If given a choice of lag lengths for model selection, such as in the case of the Augmented Dickey-Fuller statistic, we choose based on Schwarz Information Criterion (also known as Bayesian Information Criterion or BIC) (Schwarz et al., 1978).

a model's performance for qualities that are pertinent to the application at hand. In Table 2 we lay out the calculations for each of these measures. In this case, and we will assume going forward, a series with a unit root in truth is a "positive case", thus a true positive implies that the test identifies a series with a unit root as a series with a unit root.

[Table 2 about here.]

Among the most commonly evaluated are sensitivity (also known as recall) which is a measure of positive accuracy relative to the Type II errors, and specificity, which is a measure of negative accuracy relative to the Type I errors.[8] In short, a high sensitivity implies a low number of Type II errors, and a high specificity implies a low number of Type I errors. Both measures are over the interval $(0, 1)$ with a "perfect" classifier resulting in both a sensitivity and specificity of 1. Positive predicted value, also known as *precision*, is a measure of how many predicted positives were in fact true positives. The F-Measure, F_β, is the weighted harmonic mean of both PPV and sensitivity, with weights determined by β. A value of $\beta < 1$ indicates that more emphasis is placed on precision, while a value of $\beta > 1$ favors sensitivity. For our purposes we will assume $\beta = 1$ and thus we place equal weight upon both precision and sensitivity. An F_β of 1 means that a classifier has perfect sensitivity and precision. Finally, the MCC (Matthews, 1975; Tharwat, 2018) is a measure of correlation between the true and predicted classifications and lies in the interval $(-1, 1)$. A value of 1 indicates perfect prediction while a value of -1 corresponds to total disagreement; an MCC of zero is random selection. Oftentimes the MCC is considered to be a superior measure of classification ability since it is relatively invariant to changes in the data distribution.

When the battery of nine unit root tests are applied to the $M = 75,000$ series, we observe clear strengths and weaknesses – suggesting that before one places trust on any test, its qualities should be carefully examined. As is apparent in Table 3, the ADF, ERS-d and ERS-p tests achieve the highest performance, achieving accuracy over 76%. While convenient for narrative purposes, overall accuracy obscures the realities of a test's performance, thus one needs to evaluate each of the classification measures. The sensitivity for these tests is less than 60% while the specificity is over 90%, indicating that current tests are more likely to identify a unit root correctly than a near unit root. This lower performance for near unit roots points to "leakage" in the tests – current tests incorrectly classify near unit roots as unit roots to a large degree. This is consistent with previous research showing tests for unit roots tend to have low power (Ng and Perron, 2001; Kennedy, 2008) and as a result higher sensitivity would be preferable.

[Table 3 about here.]

The quality of tests vary across different conditions, such as values of ϕ and the sample size. We can measure tests' power – in this case, equivalent to sensitivity – by comparing the empirical power. The ADF test,

[8] In fact, it is trivial to show that specificity, in the context of a hypothesis test is equal to $1 - \alpha$ where $\alpha \in \{0.01, 0.05, 0.10\}$ is the long-run Type I error rate typically chosen by researchers in the Neyman-Pearson framework.

conditional upon the DGP outlined in Equation 3, is among the most powerful test statistics over the simulated sample (45% sensitivity – see Table 7 in the Appendix). In Figure 1 we have plotted traditional power curves for four of the unit root tests under examination. Here we have slightly expanded the interval of ϕ such that $\phi \in (.85, 1)$, moved in steps of .005, and completed $20,000$ iterations at each step. Data was generated from Equation 3 directly with $T \in \{25, 50, 100, 250\}$, $\epsilon_t \sim N(0, 1)$, and $y_1 = 0$. As may be expected, test performance tends to be weak in small samples (*e.g.* $T = 25$ does not exceed Power of 0.25) and grows with more degrees of freedom. While these figures are supportive of the ADF test as the more powerful option of the four, it is helpful to remember that each test may be designed with some slight differences in mind. For example, some tests are focused on achieving high performance in the presence of a moving-average term rather than the DGP we have specifically used.

[Figure 1 about here.]

One other piece of insight which can be drawn from Figure 1 and Table 3 is that there must be some variation in labeling between test statistics. That is, despite the lower power of the Phillips-Perron test under this DGP, in smaller samples, it is unlikely that the set of series identified as containing a unit root by the Phillips-Perron test is a perfect subset of the Augmented Dickey-Fuller test. This same argument could be made for the other tests as well. If this is the case, and we will show you that it is shortly, then using any single test will leave information on the table. Exploiting this between-test variation is one of the main cases for using classification algorithms such as the gradient boosted machine as mapping functions in place of the traditional null hypothesis, a subject we will delve into more thoroughly in Section 4.

3 The Unit Root Test as a Weak Learner

As well-defined hypothesis tests have all the trappings of a binary classification problem. In this section, we draw an equivalence between the two techniques.

The typical classification model is developed and applied in three steps: *training, validation, and testing.* In the training step, a model learns from a sample of input features (i.e. right-hand side variables) in order to classify instances by one of two classes of a binary target (i.e. dependent variable). For example, let $\mathcal{D} = \{(x_1, y_1), \ldots, (x_M, y_M)\}$ be a set of training data, indexed by $m = (1, \ldots, M)$. It is assumed these observations are independent and identically distributed, with each x_m a realization of random variable $\mathcal{X}$ having support $\mathbb{R}$.[9] Following convention we will denote the collection of observations as a vector with the capital such that $\mathcal{D} = (X, Y)$. A classifier, h, is a mapping, $h : \mathcal{X} \to \{-1, +1\}$, which returns the predicted

[9]Note that for simplicity we are assuming that there is but one observed x for each observation. It is trivial to assume that x_m is a K-dimensional vector of observed features such that $X \in \mathcal{X} \subset \mathbb{R}^K$.

class, y, for each m, conditional upon x. For example, we can consider the following classifier,

$$y_m = \begin{cases} +1 \text{ if } x_m \in \zeta \\ -1 \text{ if } x_m \in \zeta^c \end{cases}, \tag{5}$$

where $\zeta \subset \mathcal{X}$ such that $\zeta \cup \zeta^c = \mathcal{X}$.[10] Both simple and complex classifiers can partition $\mathcal{D}$ into classifications $y = +1$ and $y = -1$ conditional upon $\mathcal{X}$. Interestingly, this is nearly identical to the definition of a "statistical test" as provided by Neyman and Pearson (1933).

The objective of the validation step is to evaluate the performance of candidate models on an unbiased data set that is independent of the training sample. Model validation designs can take on any number of forms. A simple design involves splitting a sample into three randomly assigned partitions – a training set for fitting a model, a validation set to identify the best candidate model, and a test set to evaluate the best model's performance. In each step, the model validation design ensures identical and independently distributed samples, thereby eliminating bias in performance estimates. The design can also be evolved to meet the specific requirements of the application at hand. For example, the training sample could serve as its own validation sample through k-folds cross validation, allowing the validation set to be used to calibrate a decision threshold which reflects preferred relative costs between Type I and II errors, or one based upon a desired long run specificity (Type I error rate). The test sample remains as the final evaluation.

Only once a model achieves a desired level of accuracy when applied to the out-of-sample test can it advance to the second stage of the classification process– *scoring*. A model is said to score new, previously unseen instances by mapping them to the hypothesis space, thereby resulting in a probability of belonging to the positive class ($Pr(y_m = +1|X)$). To convert a probability into a prediction of class membership requires one to make an explicit choice of a classification threshold. In a balanced two-class classifier, a threshold can be selected to explicitly reflect the Type I and Type II costs. With this in mind, we can consider traditional hypothesis tests, such as the ADF and KPSS, as effectively simple binary classification scoring models. Given striking resemblance, we map the similarities between hypothesis tests and classification problems in Table 3.

[Table 4 about here.]

It is clear that both classification problems, as outlined in the machine learning literature, and hypothesis tests have a great deal in common. The primary differences lie in the mapping function and the resulting choice of threshold. Let us begin with perhaps the simplest learning algorithm, the decision stump. A decision stump can be thought of as a severely pruned decision tree (Oliver and Hand, 1994), with its purpose to minimize the risk function $R(h) = \mathbb{P}(Y = -1)R_1(h) + \mathbb{P}(Y = +1)R_2(h)$ where $R_1(h) = \mathbb{P}(h(X) \neq Y|Y = -1)$ and $R_2(h) = \mathbb{P}(h(X) \neq Y|Y = 1)$ denote a Type I (false positive) and Type II (false negative) error respectively

[10] The use of a binary classification system is merely for its simplicity and readability. In practice $y \in Y$ where Y has some finite cardinality, k.

(Tong et al., 2016). One way to minimize this risk function is to evaluate the conditional probability density functions $f(x|y = +1)$ and $f(x|y = -1)$ from the training set. Assuming that the training sample is representative of the process, then – in a binary classification system – it can be shown that the minimized combination of Type I and II errors occurs where these two densities intersect (see Schapire and Freund (2013) pp. 28 for a discussion).[11]

[Figure 2 about here.]

Let us examine the Augmented Dickey-Fuller test statistic as an illustrative example in Figure 2. In the top panel, we have plotted the ADF statistics, marking the decision threshold with a typical critical value of $\alpha = 0.05$ (-1.95). The density of all collected ADF statistics is represented by the dashed line, $f(x)$. Since we have simulated this data we know what the true value of ϕ is, and thus we know the true disposition of the series and can outline the resulting conditional distributions $f(x|y = +1)$ when the null is false (*i.e.* the series is stationary), and $f(x|y = -1)$ when the null is true (*i.e.* the series contains a unit root). Here, we expect a long run Type I error rate of 5% as indicated by our choice of α (shaded area labeled e_1), and an indeterminate number of Type II errors (shaded area labeled e_2).

On the other hand, the decision stump has chosen the intersection point between the two conditional densities such that $\zeta \in (-\infty, x_0 \approx -1.03]$ and $\zeta^c \in (x_0 \approx -1.03, \infty)$ which leads to a larger Type I error rate but a smaller Type II error rate. As mentioned earlier, this is the point in which the combination of Type I and Type II errors, conditional upon a $0-1$ loss function, is minimized. This means that the overall accuracy of the decision stump is strictly greater than that of the hypothesis test at the chosen $\alpha = 0.05$ for the hypothesis test. Note that both methods rely upon the support of $\mathcal{X}$, which in this case is $\mathbb{R}$. Since x_α and x_0 are both chosen over the same support, then it must be the case that for some $\alpha = \alpha'$ the cutoffs chosen are equivalent, $x_\alpha = x_0$. In our example above the decision stump corresponds to a hypothesis test in which $\alpha \approx 0.273$, meaning the hypothesis test $g(x)$, and the decision stump, $h(x)$, are equivalent conditional upon $\alpha = \alpha'$ (Neyman and Pearson, 1933). More plainly, a hypothesis test such as the Augmented Dickey Fuller test is equivalent to a weak learner.

Before we discuss how the Augmented Dickey-Fuller test – and indeed any of the unit root tests – can benefit from modern machine learning techniques, we would like to take a moment and show not only an equivalence between the test statistic and decision stump at $\alpha = \alpha'$, but also that, for any choice of α, a boosted statistic can recover the full size-power trade off of a hypothesis test.

Suppose one would like to apply a supervised learning algorithm to a hypothesis testing problem, but restrict the long-run Type I error rate to 5%. This task can be accomplished through boosting algorithms (Schapire and Freund, 2013). Boosting extends our single decision stump by combining two or more stumps to improve

[11] In practice the intersection of the conditional densities is cumbersome computationally and the decision threshold is made via Gini Impurity or some other information based metric. We find this particular explanation useful to provide intuition and allow for consistency with common graphics shown in most texts covering hypothesis testing.

classification of the phenomenon of interest – giving the technique the latitude to map the hypothesis space. Let us consider the case of Adaptive Boosting (AdaBoost), a supervised learning algorithm where the learning takes place through an iterative process. In a nutshell, the Adaboost algorithm trains a weak learner (almost always a decision stump), re-weighting the sample based on the mistakes of that learner, then training another stump on a re-weighted sample. [12] This has been shown to be surprisingly accurate and relatively robust to overfitting, see Friedman et al. (2001),Schapire and Freund (2013), and Kuhn et al. (2013) for a more robust discussion regarding the diversity in algorithms, strengths, weaknesses, and implementation.[13]

In Figure 3 we present how boosting converges towards the performance of the ADF statistic with increased number of iterations. The first plot depicts the performance of a single decision stump, which neatly intersects the ADF. When AdaBoost is trained with two iterations, the ROC curve intersects the ADF at two points, that is we have recovered two values of α' for which $\alpha = \alpha'$. In each successive figure, we increase the number of iterations the boosting algorithm is allowed to evaluate. By 50 iterations, the ROC curve for the boosted test statistic has converged to that of the null distribution. It should be clear that any weighted average of cutoffs produced by the boosting algorithm will produce some level of $\alpha' = \alpha$ and thus a boosted test statistic is equivalent to a standard hypothesis test for any choice of long-run Type I errors that may be made. Thus, if we would like to restrict our Type I errors to $\alpha = 0.05$ we can then simply choose a probability threshold such that Specificity over the validation set is $1 - \alpha = 0.95$.

[Figure 3 about here.]

It is important to note that any model or test – no matter how simple or sophisticated – is only as good as the data on which it is trained and validated. For an algorithm to retain its predictive qualities "in the wild", the training data must reflect the conditions that will be encountered when applied in real world use. In many cases the qualities of a phenomenon are overly expansive and impossible to capture in a neat empirical definition, thus the data and – in turn the test – run the risk of biasing from the true class distribution. However, if the definition of a phenomenon is well-specified, one can simulate from the definition (*i.e.* the DGP) so an algorithm can have ample opportunity to thoroughly inspect and distinguish between the null and the alternative. Unit roots fit into this latter case; we can be certain that our training set accurately reflects the hypothesis test under consideration. Increasing the size of this training set will only improve the accuracy of our threshold choice.

[12] Note that because the weak learner used in the AdaBoost algorithm is the decision stump does not prohibit the boosting of multiple features. At each iteration a different feature may provide a better classifying decision stump based on the re-weighted sample.

[13] For our purposes, in R version 4.0.2 – "Taking Off Again" – we rely on the packages "Rweka" and "ada" to implement our decision stumps and boosting algorithms respectively.

4 An ML-Based Unit Root Test

In many respects, the hypothesis test's ability to achieve a desired level of performance is rooted in the test's design: *It is fundamentally a modeling problem.* The current generation of test statistic-based hypothesis tests can be viewed as a convenient approximation of unit roots, combining input parameters in a way that balances practical arithmetic with accuracy. It stands to reason that we can achieve improved performance by constructing a composite test that integrates parameters from multiple tests and an arbitrary number of relevant features derived from time-series of interest. In this section, we lay out the blueprint of such a ML-based composite test and illustrate the marked performance improvements made possible through this novel testing strategy.

4.1 Input Features

The input features are the drivers of predictive accuracy in any modeling problem and are dependent on how much unique information each input feature offers. As we have seen in previous sections, there are marked differences in accuracy test statistics, which can also be viewed as disagreement among these tests. From another perspective, the disagreement is an opportunity to reconcile seemingly uncorrelated information that can make a test more robust. The question is *how much unique information does each input feature provide a composite ML test?*

By drawing on our bank of simulated time-series, we can examine the quality of information through two-way comparisons between test statistics in Figure 4. Perhaps what is most striking about these bivariate comparisons is that none exhibit a continuous, linear relationship. For example, the ERS-d and the ADF exhibit non-linear and disjoint relationships that could not be captured through linear methods. The complexity of these patterns is unsurprising as the hypothesis space combines three different unit root processes, requiring one to diagnose each time-series for its "true" unit root process in order to identify the appropriate critical value. One could imagine that misidentifying the time-series process could lead to misleading inferences, adding to the lack of agreement between tests.

[Figure 4 about here.]

We can further investigate these relationships between test statistics by calculating their mutual information. While a Pearson's correlation could quantify the strength of their *linear* relationship, we can evaluate the uniqueness of each test's information through *mutual information*, which is based in information theory. By calculating how each test encodes information as *bits*, we can estimate mutual information, given as:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y) log \frac{P(x,y)}{P(x)P(y)}. \tag{6}$$

Mutual Information establishes how much information – linear or non-linear – is shared between two random variables, which in turn gives a sense of how unique signal is present in among our input features. Further, one can look at the *Information Quality Ratio* defined as,

$$\text{IQR}(X,Y) = \mathbb{E}\big[I(X;Y)\big] = \frac{I(X;Y)}{H(X,Y)}, \tag{7}$$

where $H(X,Y)$ is the joint entropy of the two random variables. The Information Quality Ratio measures the amount of information in X based on Y against complete uncertainty (Wijaya et al., 2017). In Table 5, we present the Information Quality Ratio for each pair of tests where a value of 1.0 indicates observing random variable X provides perfect information about random variable Y, or more plainly; there is no unique information in Y relative to X . A value of 0.0 indicates that observing X provides no information about Y, each random variable is unique in its information content. In each pairwise comparison, at least 70% of the information is unique.

[Table 5 about here.]

Given what we have exposed about the interface between near unit roots and unit roots, we can imagine that mapping the hypothesis space may require more information than is available in current unit root tests. Because ML techniques can evaluate and integrate an arbitrarily large number of input features, we believe that saturating the model with additional input features can more richly represent the contours of the unit root surface. While traditional statistical and econometric methods run the risk of unstable results when over-parameterized, ML techniques are engineered to mitigate the effects of overfitting and capture interactions amongst input features. As outlined in Table 6, we include a set of meta-characteristics calculated from each time-series as outlined by Wang et al. (2006) and Wang et al. (2009). Moreover, we include the series length, frequency, and variance ratio between the first difference and level data.[14]

[Table 6 about here.]

When the current generation tests are applied to real world problems, we may assume that any pair of tests can arrive at seemingly contradictory verdicts about the presence of a unit root test. By drawing upon a full spectrum of unit root statistics and features, as many ML-based approaches are effective in handling large feature sets and non-linearities, we believe that even a *standard* set of algorithms can effectively map the contours of the NUR-UR interface and reconcile seemingly divergent information about unit root cases.

4.2 Mapping Function: Gradient Boosting Machine

While the center of gravity of the machine learning field has shifted to deep learning, we believe that traditional machine learning can be employed as reliable, no frills mapping functions in the unit root testing context. We

[14]The variance ratio is a heuristic whereby evidence of non-stationarity in the level series is present if the ratio between $\text{var}(\Delta y)/\text{var}(y)$ is less than one half.

thus focus on gradient boosting (Friedman, 2001; Chen and Guestrin, 2016), which is a tree-based ensemble learner that is well-suited for structured data problems and can accommodate an arbitrarily large number of variable interactions to map non-linear and disjoint hypothesis spaces.

Gradient boosting machines iteratively grow decision trees to *boost* overall model accuracy. Each iteration is seen as an opportunity to target, and correct, for the residuals of the previous iteration. All trees are grown to the same pre-specified terminal depth, but can also be de-correlated from other trees via bootstrap and random feature sampling. In general, the algorithm continues to grow additional trees until either a pre-determined number have been formed or until no additional improvement is realized. However, the number of trees is a parameter that is balanced with the learning parameter, η, which is a shrinkage parameter that is designed to inhibit overfitting by reducing the contribution of each additional tree. The final prediction is a weighted combination of these iteratively built trees, weighted by the learning parameter.

One practical quality of gradient boosting algorithms is the number of transparent, well-vetted open source implementations. For our purposes, we rely on Extreme Gradient Boosting (or XGBoost) – an open source framework for scalable gradient boosting. In Algorithm 1 we have outlined the underlying logic of a gradient boosting machine (Friedman, 2001). [15]

Algorithm 1 Gradient Boosting Machine for Classification

1: Input: Data, $\mathcal{D} = (X, Y)$, and a differentiable loss function, $L(y - i, F(x))$.
2: Initialize model with a $F_0(x) = \underset{\gamma}{\text{argmin}} \sum_m^M L(y_m, \gamma)$
3: **while** $b \leq B$ **do**
4: Compute $r_{mb} = -\left[\frac{\delta L(y_i, F(x_i))}{\delta F(x_i)}\right]_{f(x)=F_{b-1}(x)}$ for $m = 1, \ldots, M$
5: Fit a tree to the r_{mb} values and create terminal regions R_{jb} for $j = 1, \ldots, J_b$.
6: For $j = 1, \ldots, J_b$, compute $\gamma_{jb} = \underset{\gamma}{\text{argmin}} \sum_{x_m \in R_{mj}} L(y_i, F_{b-1}(x_m) + \gamma)$
7: Update $F_b(x) = F_{b-1}(x) + \nu \sum_{j=1}^{J_b} \gamma_b I(x \in R_{jb})$.
8: **end while**
9: Output $F_B(x)$.

To calibrate the mapping function and identify the optimal set of hyperparameters we use a grid search over the parameters listed in Table 7. We conducted this grid search following a five-fold cross-validation design using the $M = 350,000$ series training set. The parameter space spans a total of forty combinations requiring a total of 240 model runs. The optimal hyperparameters have been bolded in Table 7. Upon identifying the optimal parameters we train a "final" version of each algorithm on the full training set, which can then be applied to any new time-series to obtain a predicted probability of containing a unit root.

[Table 7 about here.]

[15] The primary difference between the algorithm presented here and XGBoost specifically is adjustments made for speed and scalability. We encourage readers to see both Friedman (2001) and Chen and Guestrin (2016) for more information.

4.3 Performance of the *pseudo*-Composite Unit Root Test

One feature of this ML-based test is that it can be easily applied to any scenario without modification. That is, unlike existing test statistics there is no *a priori* choice of the data generating process. The mapping function has already learned how to differentiate the processes based on unique combinations of test statistics and time-series features provided during the training phase. Since our primary interest is in understanding the Type I and Type II error rates, and not the alternative error paths that can be opened by choosing the wrong DGP, we assume that for the traditional tests the true data generating process is known and correctly specified. This means that, in this simulated environment, we are computing the lower-bound performance improvement of the ML-based test. In practice the accuracy of the traditional test statistics would be lower due to the additional error path which does not exist for our proposed method.

Additionally, since we are often interested in choosing some fixed Type I error rate through α, we not only report the optimal accuracy classification but also alternatives based on a threshold which approximates the standard choices of α. These thresholds were calculated in the validation step by examining the Receiver Operating Characteristic Curve and choosing a specificity corresponding to the desired alpha. This choice provides the appropriate decision threshold. Alternatively, one could choose a threshold based on a weight ratio which outlines the cost of Type II relative to Type I errors. In some fields where the cost function around the error types is more explicit this may be preferable for determining the appropriate threshold.[16]

[Table 8 about here.]

In either scenario, the threshold is applied to the scored out-of-sample test set in order to evaluate its accuracy. As seen in Table 8, the out-of-sample performance gains from ML-based tests are approximately seventeen percentage points greater than the next highest alternative (ERS-p). By exploiting the variation between tests and ancillary information, the *pseudo*-composite test increases accuracy primarily through an increase in sensitivity (empirical power). The proposed method is thirty-six percentage points more powerful than the traditional alternatives with comparable specificity (empirical size).

In contrast, the average empirical size of the traditional Unit Root tests in this simulated environment is approximately three percent using the published 5% critical values. The most comparable result from the proposed ML-based composite test comes from using a threshold corresponding to $\alpha = 0.05$. In this case the empirical size of the ML-based test is approximately 2.4%, roughly comparable, but with power nearly thirty-three percentage points higher than traditional tests.

[Figure 5 about here.]

[16] For example, one could identify $c(\boldsymbol{e}) = c(e_2)/c(e_1)$ where $c(e_2)$ and $c(e_1)$ are the costs of Type II and Type I errors respectively. While these may not be known explicitly their ratio, $c(\boldsymbol{e})$, maybe known and thus used to determine the appropriate threshold.

Beyond the top-line accuracy, we find evidence that the ML-based approach sustains its relative performance gains across all three DGPs. As shown in Figure 5, we have constructed four sets of power curves.[17] In each figure we have plotted the accuracy optimizing threshold choice for the proposed method and provided a band indicating accuracy at a desired test size of $\alpha = 0.10$ (upper bound) and $\alpha = 0.01$ (lower bound). From these power curves and the information contained in Table 8 it is apparent that, in the case of the full sample, the proposed method outperforms the traditional testing methods by a rather large margin. However, in the case of DGP 3 (*i.e.* autoregression without drift or trend) the range of empirical size is much more variable. Recall however that, for existing tests, we have provided the correct data generating process to the test statistic and thus have provided the upper bound on accuracy. In practice the data generating process is unknown and thus true power would be lower. In contrast, the ML-based test is markedly more effective when applied to a more complex autoregressive processes, namely DGP 1 (*i.e.* autoregression with drift and trend) and DGP 2 (*i.e.* autoregression with drift).

To understand what drives these differences in predictive power we turn to feature importance metrics. In this case we rely upon three metrics of interest. The first, *gain*, is a representation of the fractional contribution of each feature to the model based on the total gain provided by splits using the feature. This is bounded between zero (no information provided to them model) and one (all information in the model comes from this feature). A higher value indicates a more important predictive feature. Second, we use *cover*, which is the sum of the second order gradient of training data classified to a leaf. Finally, *frequency* is the rate at which that feature is selected for use in the underlying trees. In Table 9 we have outlined the top five features for the full sample and each underlying data generating process.

[Table 9 about here.]

Examining the results of the feature importance provides some interesting insight into how the composite test is making predictions. Recall that one of the fundamental problems in testing for unit roots is that one must choose a data generating process to govern the null distribution (*e.g.,* drift and trend), and that this choice opens up an additional error path beyond the traditional Type I and Type II errors. The proposed composite test is making that choice for the researcher; when either a drift or a trend term is included in the training series the Lyapunov Exponent (in levels) quickly becomes an important indicator. This would imply that the composite test is using this feature as a way to "triage" series into the appropriate data generating process, then exploits the variation between the other unit root statistics to more accurately predict if a series contains a unit root. When the underlying data generating process does not have a trend or drift term the Lyapunov Exponent takes a back seat to the more traditional unit root statistics (*e.g.,* ADF Statistic).

[17] These are not true power curves in the traditional sense since the number of observations is allowed to vary across each of the time-series in the test bank. These curves respresent the average power over the sample sizes (60-300 observations) evaluated.

By simulating data which satisfies both the null and alternative distribution we are able to use modern ensemble methods via gradient boosting to form a *pseudo*-composite hypothesis test for the presence of a unit root. This composite test is more accurate, by any reasonable measure of classification accuracy, than any single test statistic. Moreover, since the critical values we use are derived from simulations such as this it stands to reason that a trained aggregator could be provided as part of a standard statistical package providing tests for unit roots.[18]

5 An Empirical Example: Nelson & Plosser Data

Following the seminal works of Dickey and Fuller (1979) and Dickey and Fuller (1981) a conversation began about the presence of unit roots in macroeconomic indicators. Nelson and Plosser (1982) examined fourteen such indicators for the presence of a unit root; these included four measures of Gross National Product (real, nominal, per capita, and the deflator), both national employment and unemployment, real and nominal wages, money stock and velocity, bond yields, stock prices, and indices of industrial production and consumer prices. These series were annual in nature and ranged from 62 to 111 years in length. Table 10 provides the summary statistics for the data set as put forth by the authors.

[Table 10 about here.]

Of the fourteen indicators included in their analysis, the authors came to the conclusion that thirteen of them contained a unit root. The fourteenth – unemployment rate - was deemed to be stationary. As research into unit roots continued over the subsequent decades this data was revisited a number of times including works such as Perron (1989); Stock (1991); Kwiatkowski et al. (1992); Andrews and Chen (1994); Zivot and Andrews (2002), and Charles and Darné (2012). Each of these re-examined the data from Nelson and Plosser (1982) using either new tests, assumptions, estimation methods, or a combination of all three, and found as few as three indicators containing a unit root (Perron, 1989) to a full affirmation of the original work (Stock, 1991; Kwiatkowski et al., 1992; Andrews and Chen, 1994).

Following other authors, we examined the log of these series for the presence of a unit root using our composite testing mechanism. For each series we calculated the full set of features used for training in our mapping function. Further, we assumed three long-run Type I error rates, $\alpha \in (0.01, 0.05, 0.10)$, indicating a decision threshold which is progressively more biased towards failing to reject the null. In Figure 6a we have provided the probability that a unit root is present for each series (filled diamonds) as well as the decision thresholds corresponding to the proposed values of α. If a 10% Type I error rate is desired then this corresponds to a probability threshold of 0.665 meaning the decoder would need to identify a higher bar of evidence to suggest a series is a unit root than the naive 0.50 threshold (recall that the output is the probability a series

[18] We are currently developing such a package which will provide a trained model and an appropriate function for calculating the features. This will be available via `https://github.com/DataScienceForPublicPolicy/unitrootML` when complete.

contains a unit root). In this scenario three series would be identified as containing a unit root; Common Stock Prices, Money Stock, and Industrial Production. When $\alpha = 0.01$ is chosen this indicates a low bar of evidence needed to identify a unit root. The threshold drops to 0.042 and nearly all of the indicators would be considered as unit roots.

In Figure 6b we compare our results to the aforementioned works. Filled squares indicate those series for which the corresponding paper indicated a unit root while the empty square indicates a stationary series. There is no single macroeconomic indicator for which there is universal agreement across the literature regarding its unit root status. If we consider a fixed Type I error rate of $\alpha = 0.10$ the method outlined here indicates that eleven of the fourteen series do not contain a unit root. As we move towards smaller values of α we are demanding either more evidence that the alternative (not unit root) is true, or conversely less evidence that the null (unit root) is true. By the time we get to $\alpha = 0.01$ we can see that the majority of the indicators, twelve out of fourteen, are considered to contain a unit root. Three series show a unit root at all levels of Type I errors: Common Stock Prices, Money Stock, and Industrial Production. The majority of those indicators which we find are stationary at $\alpha = 0.10$ only switch a unit root when the highest bar of evidence is applied, the lone exception is Velocity which we identify as a unit root at $\alpha = 0.05$. Note that we have omitted the "most accurate" threshold which corresponds to $\alpha \approx 0.074$ as the decisions are identical to the case where $\alpha = 0.10$.

[Figure 6 about here.]

6 Conclusion

Testing for a unit root is one of the foundational pieces in all of time-series econometrics. Its importance is made apparent by the thousands of hours of human capital which have been spent developing and refining test statistics under a variety of different contexts. In this paper we have proposed a *pseudo*-composite test for unit roots which leverages modern computational power and classification algorithms to exploit variation between existing tests and other pertinent information. The proposed testing method is more powerful at differentiating unit roots from near unit roots, an area in which current tests are known to struggle. Moreover, by leveraging the train-validation framework common to the Machine Learning literature we are able to retain control over long-run Type I error rates by allowing users to specify a desired value of α.

Using a simple simulated environment common to the unit root literature we show that the ensemble approach – using tree-based algorithms that learn from nine common test statistics and other features of the time-series bank – is approximately seventeen percentage points more accurate than the next best, single test alternative. The majority of this increase in accuracy comes from the ability to more accurately reject the null when the null of a unit root when the null is false. The method proposed herein shows a thirty-six percentage point increase in sensitivity (empirical power) relative to the next best, single test alternative without significant

size distortion. Traditional power curves, receiver operating characteristic curves, and classification measures (*e.g.* Matthew's Correlation Coefficient) all point to the ensemble method as strictly dominating the evaluated test set.[19]

Finally, we revisit the fourteen macroeconomic indicators first examined in Nelson and Plosser (1982) to place the results of our approach in context with related literature. Of these fourteen indicators the original research and test results indicated thirteen of the fourteen were a unit root process with the lone exception being the national unemployment rate. This specific data set has been revisited over time with as few as three being declared a unit root process (Perron, 1989). Our method indicates that, under the accuracy maximizing value of α, a total of three indicators should be considered a unit root: common stock prices, money stock, and the industrial production index. For the other eleven indicators the probability the series contains a unit root is less than 0.50.

As we have demonstrated with the unit roots case, the unit root hypothesis space's underlying multivariate distribution is not well-behaved in practice and the abilities of current hypothesis tests are quite variable. Thus, the ability for ML-based hypothesis tests to reconcile conflicting diagnostics and realize large performance gains has far-reaching implications – the quality of inferences can be vastly improved and the disagreement among tests are resolved. Much like replicating a sample of DNA for further study, ML-based testing can be extended to any phenomenon that is well-defined and can be simulated (*e.g.* equality of distributions, normality). An intriguing possibility is the detection of nuanced sub-conditions (*e.g.* stationary, near unit root, unit root, explosive process), which in turn allows for richer characterizations of random variables and lend greater confidence to the quality of inferences and predictions.

7 Appendix

[Table 11 about here.]

[Table 12 about here.]

[Table 13 about here.]

References

Andrews, D. W. and H.-Y. Chen (1994). Approximately median-unbiased estimation of autoregressive models. *Journal of Business & Economic Statistics 12*(2), 187–204.

Bai, J. and S. Ng (2004). A panic attack on unit roots and cointegration. *Econometrica 72*(4), 1127–1177.

[19] We have developed an R package which will be available for practitioners to use our simulated environment to test for the presence of a unit root with no additional computational or programming burden over the standard function syntax for most available test functions.

Breitung, J. (2002). Nonparametric tests for unit roots and cointegration. *Journal of Econometrics 108*(2), 343–363.

Breitung, J. and R. Taylor (2003). Corrigendum to" nonparametric tests for unit roots and cointegration"[j. econom. 108 (2002) 343-363]. *Journal of Econometrics 117*(2), 401–404.

Charles, A. and O. Darné (2012). Trends and random walks in macroeconomic time series: A reappraisal. *Journal of Macroeconomics 34*(1), 167–180.

Chen, T. and C. Guestrin (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.

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

Dickey, D. A. and W. A. Fuller (1981). Likelihood ratio statistics for autoregressive time series with a unit root. *Econometrica: Journal of the Econometric Society*, 1057–1072.

Elliot, B., T. Rothenberg, and J. Stock (1996). Efficient tests of the unit root hypothesis. *Econometrica 64*(8), 13–36.

Elliott, G., T. J. Rothenberg, and J. H. Stock (1992). Efficient tests for an autoregressive unit root.

Elliott, G. and A. Timmermann (2004). Optimal forecast combinations under general loss functions and forecast error distributions. *Journal of Econometrics 122*(1), 47–79.

Friedman, J., T. Hastie, and R. Tibshirani (2001). *The elements of statistical learning*, Volume 1. Springer series in statistics New York.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Granger, C. W. and P. Newbold (1974). Spurious regressions in econometrics. *Baltagi, Badi H. A Companion of Theoretical Econometrics*, 557–61.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

Jansson, M. and M. Ø. Nielsen (2012). Nearly efficient likelihood ratio tests of the unit root hypothesis. *Econometrica 80*(5), 2321–2332.

Kennedy, P. (2008). *A guide to econometrics*. John Wiley & Sons.

Kuhn, M., K. Johnson, et al. (2013). *Applied predictive modeling*, Volume 26. Springer.

Kwiatkowski, D., P. C. Phillips, P. Schmidt, Y. Shin, et al. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics 54*(1-3), 159–178.

MacKinnon, J. G. (1996). Numerical distribution functions for unit root and cointegration tests. *Journal of applied econometrics 11*(6), 601–618.

MacKinnon, J. G. (2010). Critical values for cointegration tests. Technical report, Queen's Economics Department Working Paper.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure 405*(2), 442–451.

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

Neyman, J. and E. S. Pearson (1933). The testing of statistical hypotheses in relation to probabilities a priori. In *Mathematical Proceedings of the Cambridge Philosophical Society*, Volume 29, pp. 492–510. Cambridge University Press.

Ng, S. and P. Perron (2001). Lag length selection and the construction of unit root tests with good size and power. *Econometrica 69*(6), 1519–1554.

Oliver, J. J. and D. Hand (1994). Averaging over decision stumps. In *European Conference on Machine Learning*, pp. 231–241. Springer.

Pantula, S. G., G. Gonzalez-Farias, and W. A. Fuller (1994). A comparison of unit-root test criteria. *Journal of Business & Economic Statistics 12*(4), 449–459.

Perron, P. (1989). The great crash, the oil price shock, and the unit root hypothesis. *Econometrica: journal of the Econometric Society*, 1361–1401.

Phillips, P. C. and P. Perron (1988). Testing for a unit root in time series regression. *Biometrika 75*(2), 335–346.

Schapire, R. E. and Y. Freund (2013). Boosting: Foundations and algorithms. *Kybernetes*.

Schmidt, P. and P. C. Phillips (1992). Lm tests for a unit root in the presence of deterministic trends. *Oxford Bulletin of Economics and Statistics 54*(3), 257–287.

Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics 6*(2), 461–464.

Stock, J. H. (1991). Confidence intervals for the largest autoregressive root in us macroeconomic time series. *Journal of monetary economics 28*(3), 435–459.

Tharwat, A. (2018). Classification assessment methods. *Applied Computing and Informatics*.

Timmermann, A. (2006). Forecast combinations. *Handbook of economic forecasting 1*, 135–196.

Tong, X., Y. Feng, and A. Zhao (2016). A survey on neyman-pearson classification and suggestions for future research. *Wiley Interdisciplinary Reviews: Computational Statistics 8*(2), 64–81.

Wang, X., K. Smith, and R. Hyndman (2006). Characteristic-based clustering for time series data. *Data mining and knowledge Discovery 13*(3), 335–364.

Wang, X., K. Smith-Miles, and R. Hyndman (2009). Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing 72*(10-12), 2581–2594.

Wijaya, D. R., R. Sarno, and E. Zulaika (2017). Information quality ratio as a novel metric for mother wavelet selection. *Chemometrics and Intelligent Laboratory Systems 160*, 59–71.

Zivot, E. and D. W. K. Andrews (2002). Further evidence on the great crash, the oil-price shock, and the unit-root hypothesis. *Journal of Business & Economic Statistics 20*(1), 25–44.

Table 1: A 2×2 Confusion Matrix

	True NUR	True UR
Predicted NUR	True Positive (TP)	False Positive (FP)
Predicted UR	False Negative (FN)	True Negative (TN)

Table 2: Calculating Measures of Efficacy

Measure	Calculation
Accuracy	$\frac{(TP+TN)}{(TP+TN+FP+FN)}$
Sensitivity	$\frac{TP}{(TP+FN)}$
Specificity	$\frac{TN}{(TN+FP)}$
Positive Predictive Value	$\frac{TP}{(TP+FP)}$
Negative Predictive Value	$\frac{TN}{(TN+FN)}$
F-Measure	$\frac{(1+\beta^2)(TP)}{((1+\beta^2)(TP)+\beta^2(FN)+FP)}$
MCC	$\frac{(TP\times TN)-(FP\times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

Table 3: Accuracy Evaluation of Current Unit Root Tests

	ACC	SEN	SPE	PPV	NPV	F^1	MCC
ADF	0.763	0.546	0.980	0.964	0.684	0.697	0.583
PP	0.744	0.512	0.975	0.953	0.667	0.666	0.549
KPSS	0.614	0.250	0.977	0.916	0.567	0.393	0.331
PGFF	0.745	0.499	0.989	0.978	0.665	0.661	0.560
BREIT	0.672	0.361	0.981	0.951	0.607	0.524	0.437
ERSd	0.762	0.545	0.979	0.963	0.683	0.696	0.582
ERSp	0.770	0.564	0.976	0.958	0.692	0.710	0.592
URZA	0.635	0.309	0.959	0.883	0.582	0.458	0.354
URSP	0.727	0.552	0.903	0.850	0.669	0.669	0.485

Note: For each of these tests we have used the correct originating data generating process to select the null distribution and critical value. This implies that – under the simulation parameters outlined – this is the "best" these tests can do. We have closed off any additional error path beyond the typical Type I and Type II errors.

Table 4: Drawing parallels between hypothesis tests and classification models

Qualities	Hypothesis Tests	Classification problem
Outcomes	Complementary hypotheses	Binary target
Mapping function	Test statistic and null distribution	Classification algorithm
Model Validation	Type I and Type II errors	Type I and Type II errors
Threshold	Rejection threshold	Decision threshold

Table 5: Mutual Information of Test Statistics

	ADF	KPSS	PP	PGFF	Breit	ERS-d	ERS-p	URZA	URSP
ADF	1.000								
KPSS	0.158	1.000							
PP	0.084	0.028	1.000						
PGFF	0.142	0.142	0.064	1.000					
Breit	0.218	0.214	0.052	0.131	1.000				
ERS-d	0.273	0.162	0.101	0.144	0.259	1.000			
ERS-p	0.085	0.068	0.084	0.027	0.037	0.064	1.000		
URZA	0.069	0.026	0.206	0.041	0.038	0.073	0.092	1.000	
URSP	0.061	0.024	0.257	0.056	0.044	0.081	0.043	0.096	1.000

Note: In this table we have provided the Information Quality Ratio (IQR) for each two-by-two test comparison. The IQR measures the amount of mutual information, $I(X;Y)$, relative to the joint entropy $H(X,Y)$ and is a representation of total correlation.

Table 6: Features for Classification

UR Tests	Level and First Difference	STL Decomposed Series	Miscellaneous
ADF	Skewness	TNN Test	Length
PP	Kurtosis	Skewness	Frequency
PGFF	Box Statistic	Kurtosis	$\text{var}(\Delta y)/\text{var}(y)$
KPSS	Lyapunov Exponent	Box Statistic	
ERS (d & p)	TNN Test		
URSP	Hurst Exponent		
URZA	Strength of Trend		
Breit	Strength of Seasonality		

Note: The statistics calculated on the STL decomposed series are done both on the level and the first difference. Going forward we use Δ to denote a statistic on the first difference and "Decomposed" to denote those on the adjusted series. TNN Test refers to the Teraesvirta Neural Network test. See Wang et al. (2006), Wang et al. (2009), and `https://robjhyndman.com/hyndsight/tscharacteristics/` for more information.

Table 7: Grid Search Parameter Space

Hyperparameters	Gradient Boosting
Eta (Shrinkage Parameter)	$\{0.01, 0.03, \mathbf{0.1}, 0.3, 0.5\}$
Column Sample by Tree (% of Columns)	$\{\mathbf{0.8}.1\}$
Subsampling of Training Instances	$\{\mathbf{0.8}, 1\}$
Max Tree Depth (Number of Levels)	$\{4, \mathbf{6}\}$

Note: As identified through five-fold cross validation, the winning value for each hyperparameter is bolded.

Table 8: Main Results

	ACC	SEN	SPE	PPV	NPV	F^1	MCC
GB$\|\alpha = 0.100$	0.937	0.951	0.923	0.925	0.950	0.938	0.874
GB$\|\alpha = 0.074^*$	0.941	0.934	0.948	0.947	0.935	0.941	0.882
GB$\|\alpha = 0.050$	0.938	0.900	0.976	0.973	0.908	0.935	0.878
GB$\|\alpha = 0.010$	0.892	0.785	0.998	0.998	0.823	0.879	0.802
ADF	0.763	0.546	0.980	0.964	0.684	0.697	0.583
PP	0.744	0.512	0.975	0.953	0.667	0.666	0.549
KPSS	0.614	0.250	0.977	0.916	0.567	0.393	0.331
PGFF	0.745	0.499	0.989	0.978	0.665	0.661	0.560
BREIT	0.672	0.361	0.981	0.951	0.607	0.524	0.437
ERSd	0.762	0.545	0.979	0.963	0.683	0.696	0.582
ERSp	0.770	0.564	0.976	0.958	0.692	0.710	0.592
URZA	0.635	0.309	0.959	0.883	0.582	0.458	0.354
URSP	0.727	0.552	0.903	0.850	0.669	0.669	0.485

Note: The $\cdot^*$ indicates the α which corresponds to the accuracy maximizing threshold according to the Gradient Boosted Machine decoder as outlined earlier. For each of the alternative tests (*e.g.,* ADF) we have provided the test with the correct originating data generating process. This means that our comparison group is the "best" these tests can do in the environment provided and contains no model specification error.

Table 9: Top Five Features

Full Sample				DGP 1			
Feature	Gain	Cover	Frequency	Feature	Gain	Cover	Frequency
Lyapunov Exponent	0.446	0.083	0.045	Lyapunov Exponent	0.520	0.157	0.040
ADF Statistic	0.137	0.145	0.106	Strength of Seasonality	0.206	0.034	0.003
PP Statistic	0.129	0.072	0.033	Box Statistic	0.139	0.080	0.034
PGFF Statistic	0.063	0.047	0.048	ERSd Statistic	0.032	0.114	0.043
ERSp Statistic	0.058	0.054	0.045	PGFF Statistic	0.029	0.137	0.051
DGP 2				DGP 3			
Lyapunov Exponent	0.902	0.380	0.063	ERSp Statistic	0.509	0.057	0.031
ERSp Statistic	0.031	0.079	0.034	ADF Statistic	0.255	0.079	0.045
ADF Statistic	0.018	0.044	0.042	PGFF Statistic	0.037	0.038	0.028
Variance Ratio	0.005	0.109	0.047	ERSd Statistic	0.015	0.049	0.031
Breit Statistic	0.005	0.043	0.035	Lyapunov Exponent (Δ)	0.010	0.033	0.029

Table 10: Summary Statistics for Nelson and Plosser (1982) Data

Variable	Start	End	T	Min	Max	SD
Real GNP	1909	1970	62	116.80	724.70	180.32
Nominal GNP	1909	1970	62	33,400	974,126	252,334.20
GNP Per Capita	1909	1970	62	1,126	3,577	726.59
Industrial Production	1860	1970	111	0.90	110.70	27.65
Employment	1890	1970	81	21,102	81,815	16,755.98
Unemployment Rate	1890	1970	81	1.20	24.90	5.56
GNP Deflator	1889	1970	82	22.10	135.30	31.38
CPI	1860	1970	111	25.00	116.30	23.41
Nominal Wages	1900	1970	71	487	8,150	2,134.63
Real Wages	1900	1970	71	19.48	70.81	16.46
Money Stock	1889	1970	82	3.60	401.30	102.67
Velocity of Money	1869	1970	102	1.16	5.61	1.14
Bond Yields	1871	1970	100	3.14	98.70	24.05
Stock Prices	1900	1970	71	2.43	7.60	0.96

Table 11: Results for DGP 1

	ACC	SEN	SPE	PPV	NPV	F^1	MCC
GB$\|\alpha = 0.100$	0.980	0.960	1.000	1.000	0.962	0.980	0.961
GB$\|\alpha = 0.074^*$	0.982	0.964	1.000	1.000	0.965	0.982	0.964
GB$\|\alpha = 0.050$	0.978	0.955	1.000	1.000	0.957	0.977	0.956
GB$\|\alpha = 0.010$	0.966	0.933	1.000	1.000	0.937	0.965	0.935
ADF	0.708	0.423	0.992	0.981	0.632	0.591	0.505
PP	0.722	0.453	0.991	0.980	0.644	0.620	0.526
KPSS	0.571	0.159	0.983	0.903	0.539	0.271	0.250
PGFF	0.692	0.385	0.998	0.994	0.619	0.556	0.485
BREIT	0.652	0.309	0.995	0.986	0.590	0.471	0.419
ERSd	0.703	0.410	0.997	0.992	0.628	0.580	0.502
ERSp	0.706	0.421	0.990	0.977	0.631	0.589	0.500
URZA	0.640	0.295	0.985	0.953	0.583	0.450	0.387
URSP	0.763	0.540	0.987	0.976	0.682	0.695	0.589

Note: The $\cdot^*$ indicates the α which corresponds to the accuracy maximizing threshold according to the Gradient Boosted Machine decoder as outlined earlier. For each of the alternative tests (*e.g.,* ADF) we have provided the test with the correct originating data generating process. This means that our comparison group is the "best" these tests can do in the environment provided and contains no model specification error.

Table 12: Results for DGP 2

	ACC	SEN	SPE	PPV	NPV	F^1	MCC
GB$\|\alpha = 0.100$	0.990	0.992	0.987	0.987	0.992	0.990	0.979
GB$\|\alpha = 0.074^*$	0.992	0.991	0.992	0.992	0.992	0.992	0.984
GB$\|\alpha = 0.050$	0.993	0.989	0.997	0.997	0.990	0.993	0.987
GB$\|\alpha = 0.010$	0.986	0.972	1.000	1.000	0.973	0.986	0.973
ADF	0.760	0.522	0.996	0.992	0.678	0.684	0.589
PP	0.772	0.550	0.992	0.985	0.690	0.706	0.605
KPSS	0.647	0.291	1.000	1.000	0.588	0.451	0.413
PGFF	0.773	0.543	1.000	1.000	0.689	0.704	0.612
BREIT	0.694	0.384	1.000	1.000	0.621	0.555	0.488
ERSd	0.763	0.524	1.000	1.000	0.680	0.688	0.597
ERSp	0.777	0.551	1.000	1.000	0.692	0.710	0.617
URZA	0.633	0.315	0.947	0.854	0.583	0.460	0.338
URSP	0.705	0.544	0.865	0.799	0.657	0.648	0.432

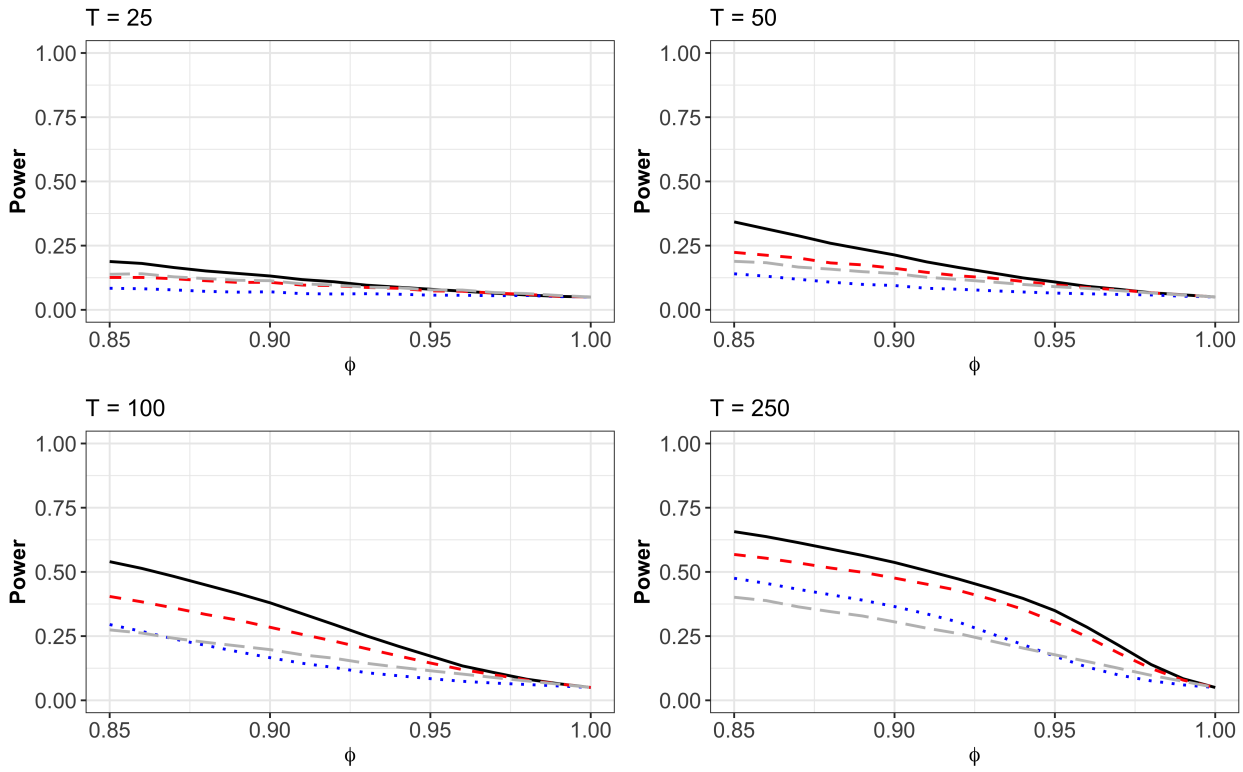
Note: The $\cdot^*$ indicates the α which corresponds to the accuracy maximizing threshold according to the Gradient Boosted Machine decoder as outlined earlier. For each of the alternative tests (*e.g.,* ADF) we have provided the test with the correct originating data generating process. This means that our comparison group is the "best" these tests can do in the environment provided and contains no model specification error.

Table 13: Results for DGP 3

	ACC	SEN	SPE	PPV	NPV	F^1	MCC
GB$\|\alpha = 0.100$	0.839	0.897	0.782	0.804	0.883	0.848	0.683
GB$\|\alpha = 0.074^*$	0.852	0.852	0.852	0.852	0.852	0.852	0.704
GB$\|\alpha = 0.050$	0.844	0.758	0.930	0.915	0.793	0.829	0.698
GB$\|\alpha = 0.010$	0.724	0.455	0.995	0.988	0.645	0.623	0.534
ADF	0.821	0.691	0.951	0.934	0.755	0.794	0.665
PP	0.737	0.532	0.943	0.903	0.668	0.669	0.520
KPSS	0.624	0.301	0.948	0.853	0.575	0.445	0.326
PGFF	0.768	0.568	0.969	0.949	0.691	0.711	0.586
BREIT	0.670	0.391	0.949	0.884	0.609	0.542	0.409
ERSd	0.820	0.700	0.940	0.921	0.758	0.796	0.659
ERSp	0.827	0.718	0.937	0.919	0.768	0.806	0.670
URZA	0.632	0.318	0.946	0.855	0.581	0.464	0.340
URSP	0.714	0.570	0.858	0.801	0.666	0.666	0.447

Note: The $\cdot^*$ indicates the α which corresponds to the accuracy maximizing threshold according to the Gradient Boosted Machine decoder as outlined earlier. For each of the alternative tests (*e.g.,* ADF) we have provided the test with the correct originating data generating process. This means that our comparison group is the "best" these tests can do in the environment provided and contains no model specification error.

Figure 1: Unit Root Tests Have Low Power



Note: This figure provides the traditional power curves conditional upon $T \in \{25, 50, 100, 250\}$ for the ADF (black solid line), PP (blue dotted), ERS (red dashed), and Breit (grey long-dash) tests. For simplicity, we focus only on the DGP in Equation 3 and have identified empirical critical values that allow the tests to be comparable at $\alpha = 0.05$. Note that in Table 2 we have an assortment of series lengths, $T \in (60, 600)$. Moreover, it is important to remember that some of these statistics, as mentioned previously, were designed to optimize power under slightly different conditions (*e.g.* heteroskedastic error terms).

Figure 2: Hypothesis Tests as Classification Problems

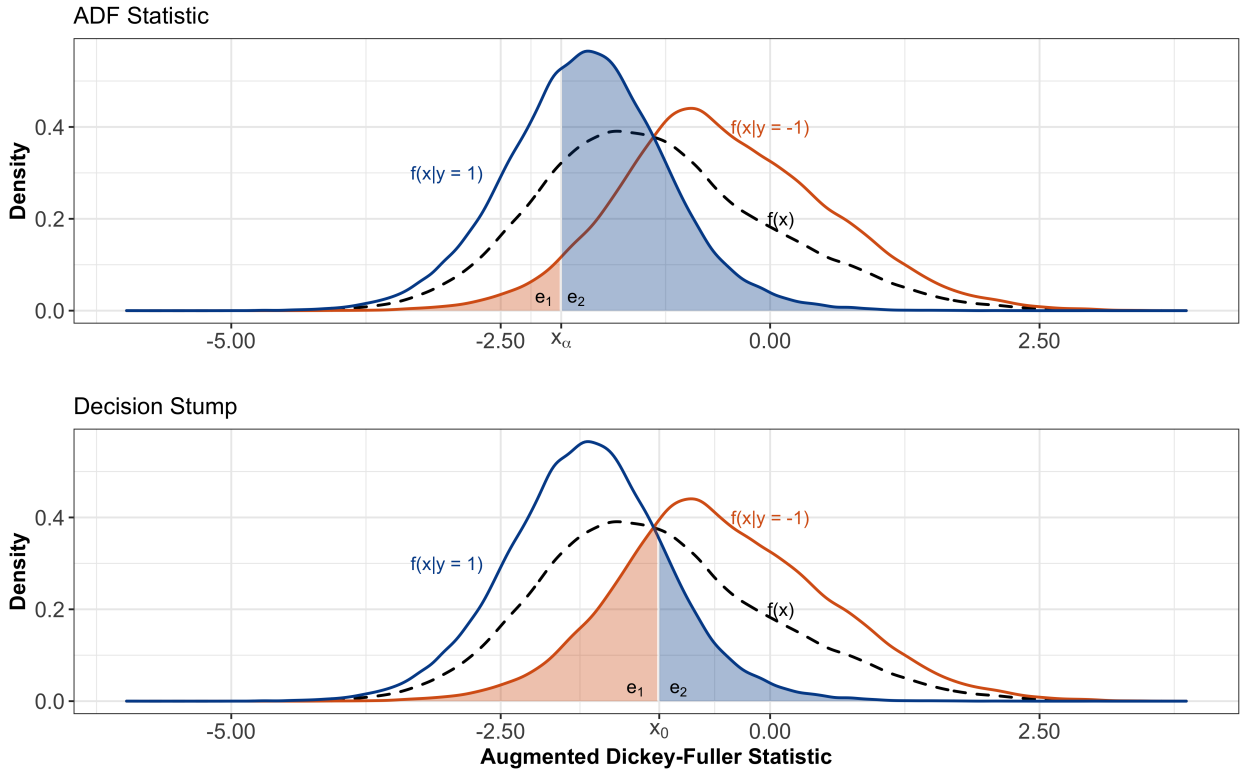
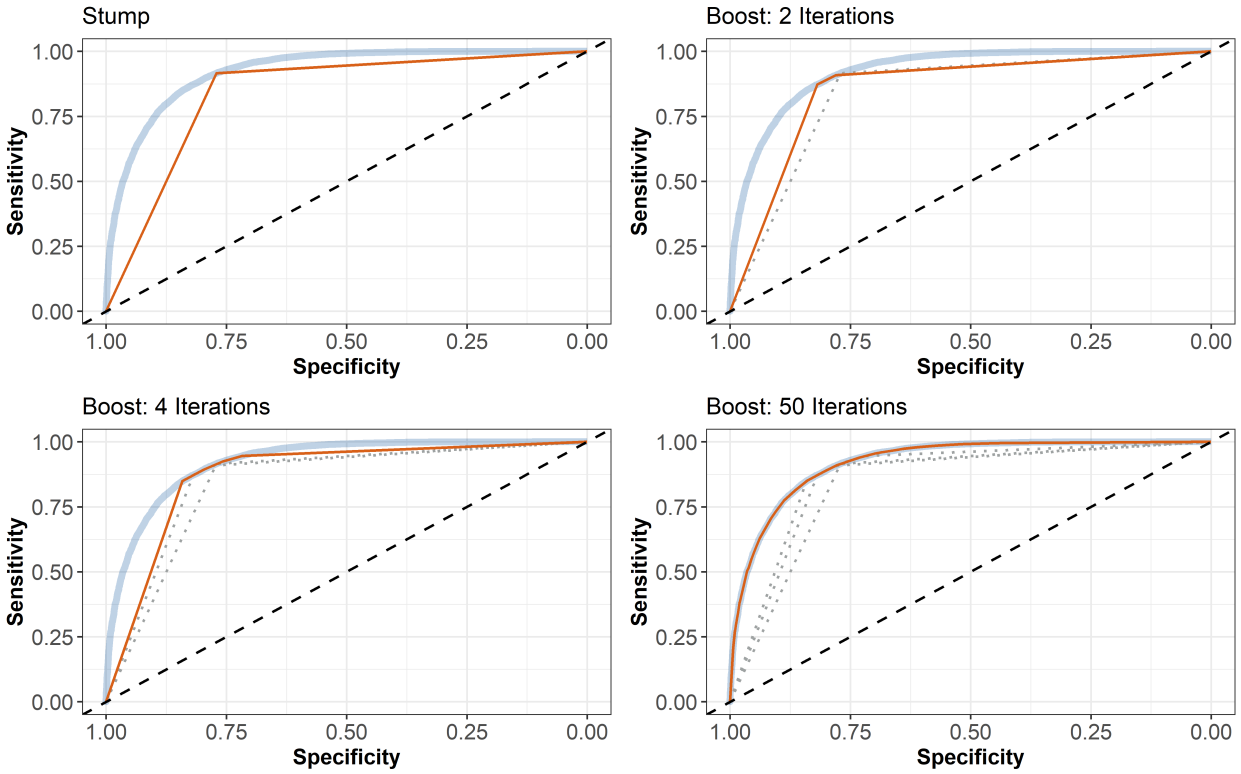
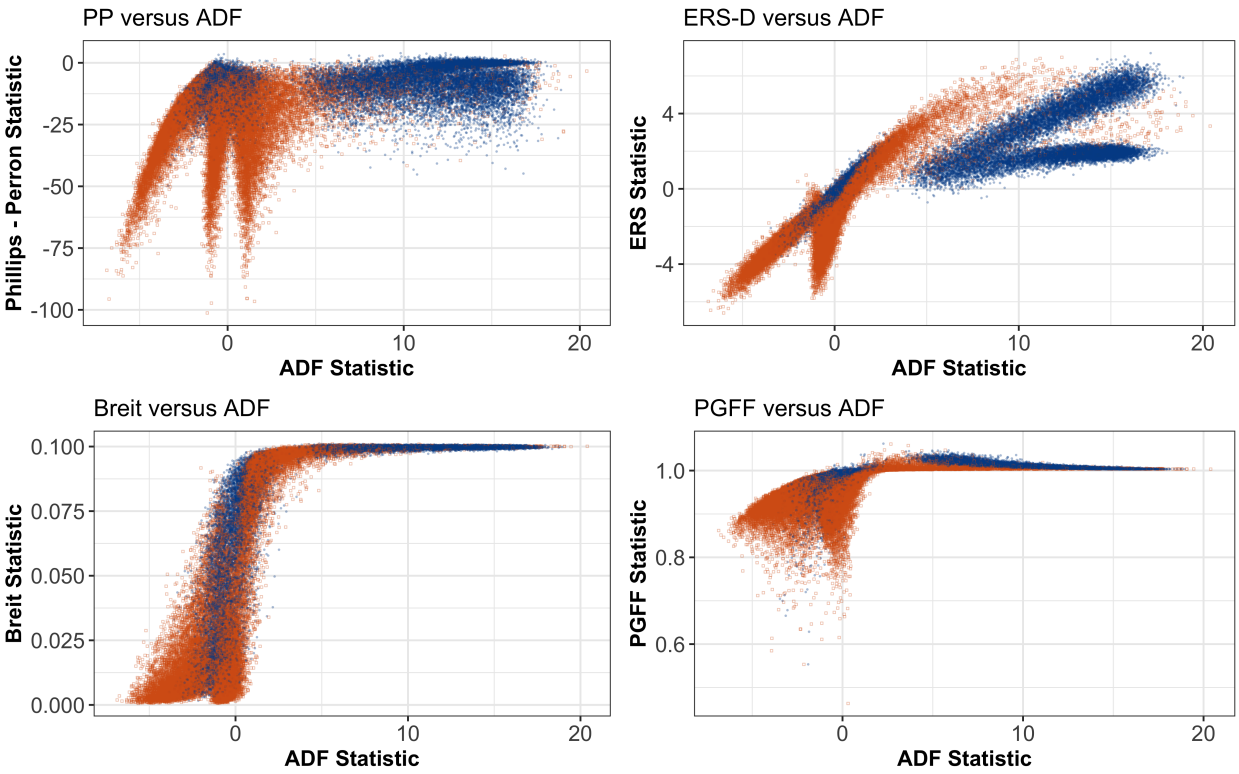


Figure 3: Receiver Operating Characteristic Curves: Convergence from Boosting



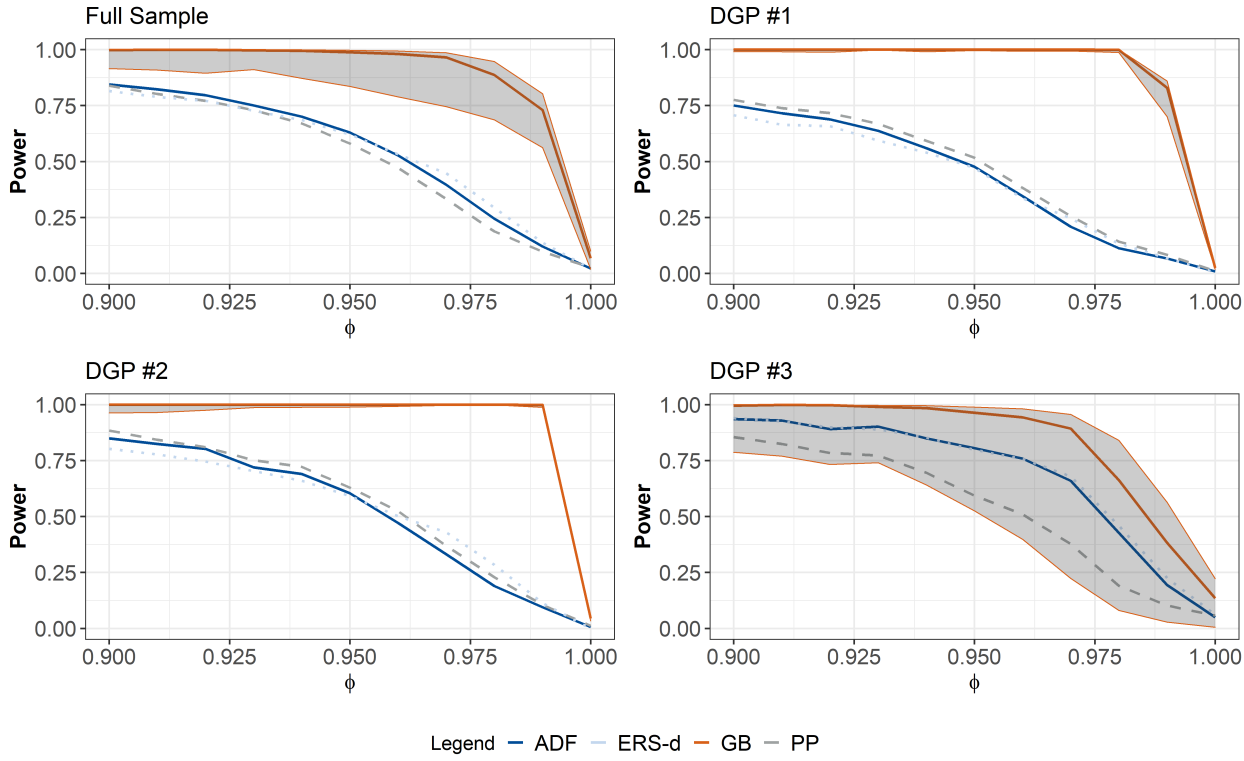
Note: Here we are plotting the Receiver Operating Characteristic (ROC) curve under the null (thick light-blue line) versus the boosted alternatives. To calculate the ROC curve under the null we calculated the corresponding probability for the ADF statistic under the cumulative probability distribution of the [simulated] null. This allows for a direct comparison to the predictions of the boosting algorithm.

Figure 4: Variation in Test Statistics.



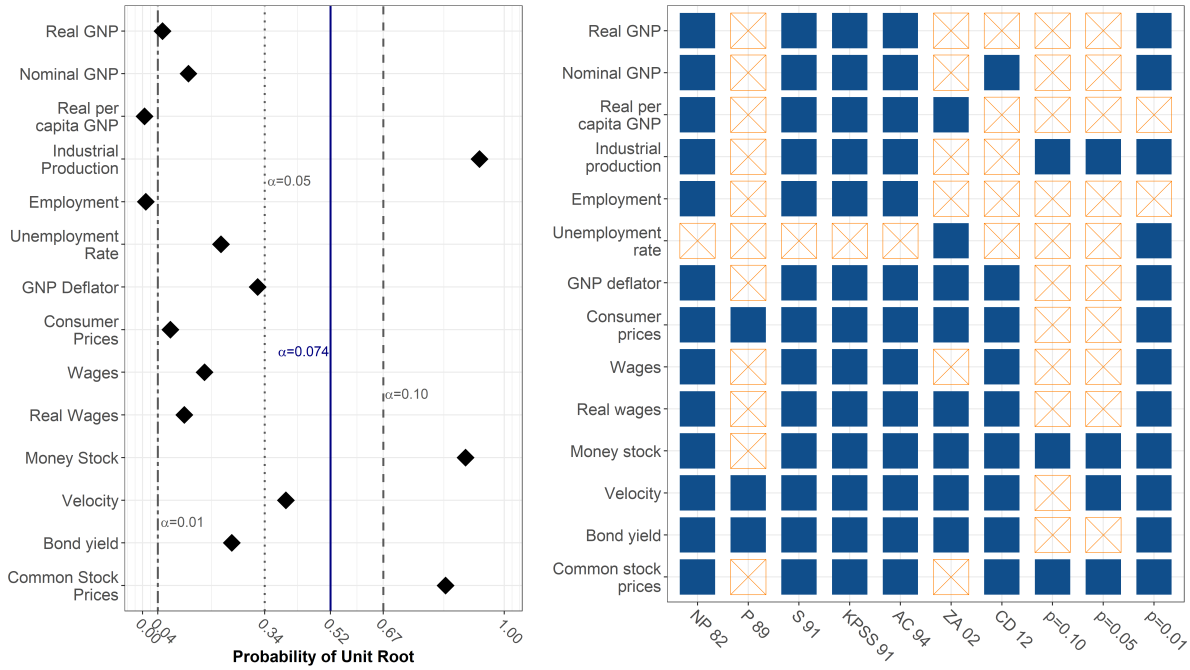
Note: Blue points are series with a unit root while orange points are stationary series.

Figure 5: Power Curves Comparing Test Statistics with ML-Based Test



Note: Based on a test sample of $M = 75,000$, the power curves are estimated using a critical value or threshold corresponding to $\alpha = 0.074$. The curves correspond to the XG (orange solid line), as well as the ADF (solid navy blue line), PP (dashed grey line), and ERS-d (dotted light blue line) tests. The upper bound of the XG curve represents performance conditional upon $\alpha = 0.10$, while the lower bound represents a desired α of 0.01.

Figure 6: Comparison of Findings on Data from Nelson and Plosser (1982)



Note: Here we have plotted, on the left, the probability each series is a unit root (filled diamonds) in concert with thresholds based on $\alpha \in (0.10, 0.05, 0.01)$. On the right we have contextualized our results in the literature. If the related paper indicates the series is a unit root it is represented here by a shaded blue square. A decision indicating stationarity is shown here by an orange square with an X inside.