Era Splitting: Invariant Learning for Decision Trees

Timothy DeLise*
July 24, 2024

Abstract

Real-life machine learning problems exhibit distributional shifts in the data from one time to another or from one place to another. This behavior is beyond the scope of the traditional empirical risk minimization paradigm, which assumes i.i.d. distribution of data over time and across locations. The emerging field of out-of-distribution (OOD) generalization addresses this reality with new theory and algorithms which incorporate environmental, or era-wise information into the algorithms. So far, most research has been focused on linear models and/or neural networks (Arjovsky et al., 2020, Parascandolo et al., 2020 [2, 24]). In this research we develop two new splitting criteria for decision trees, which allow us to apply ideas from OOD generalization research to decision tree models, namely, gradient boosting decision trees (GBDTs). The new splitting criteria use era-wise information associated with the data to grow treebased models that are optimal across all disjoint eras in the data, instead of optimal over the entire data set pooled together, which is the default setting. In this paper, two new splitting criteria are defined and analyzed theoretically. Effectiveness is tested on four experiments, ranging from simple, synthetic to complex, real-world applications. In particular we cast the OOD domain-adaptation problem in the context of financial markets, where the new models out-perform state-of-the-art GBDT models on the Numerai data set. The new criteria are incorporated into the Scikit-Learn code base and made freely available online.

1 Introduction

Gradient boosting decision trees (GBDTs) (Friedman, 2001a [11]) and descendent algorithms such as those implemented via the Light Gradient Boosting Machine (Ke et al., 2017b [17]), XGBoost (Chen and Guestrin, 2016a [7]),

^{*}Département de mathématiques et de statistique, Université de Montréal, Montreal, QC, Canada. timothy.delise@umontreal.ca

Acknowledgment: This research was funded, in part, by the Numerai during a 2023 research internship.

and Scikit-learn's HistGradientBoostingRegressor (Pedregosa et al., 2011 [25]), among others, are considered state of the art for many real-world supervised learning problems characterized by medium data size, tabular format, and/or low signal-to-noise ratio (Grinsztajn et al., 2022, Fieberg et al., 2023 [14, 10]). One such data science problem occurs with predicting future returns in the capital markets. When this problem is framed as a standard supervised learning problem, the target variable is assumed to be a continuous score, representing the expected return (alpha) for a particular stock. The input data is the available information about that stock at that particular time. Each stock's input data and associated target for each date are represented as one row in a data set. This problem is generally referred to as predicting the cross-sectional returns of the equity market.

Supervised learning models in their original form are not aware of the date information presented in the data. One row of input data is associated with one target. Samples are assumed to be drawn I.I.D. from the training and test sets, per the empirical risk minimization (ERM) principal (Vapnik, 1991 [27]). While in reality N samples are drawn simultaneously from the data set, where N is the number of stocks in the tradable universe. The predictions for all N stocks are created together at the same time. These N data points may not be completely I.I.D. either, as there are measurable positive correlations among stocks in the stock market over any particular time period. Indeed these ideas are the foundation of the capital asset pricing model (CAPM) (Fama and French, 2004 [9]) and modern portfolio theory (Markowitz, 1952 [21]), as well as the principles of risk factor models such as the Barra risk model (Blin et al., 2022 [5]). Thus, the cross-sectional stock prediction problem violates many of the assumptions of the ERM principle implicitly assumed in the typical supervised learning setting. Meanwhile, practitioners will often take a GBDT model off the shelf and implement it under these assumptions.

The discrepancy between the foundations of the ERM principle and the realities of ML application fits into the contemporary literature in the academic field of out-of-distribution generalization (OOD) research (Liu et al., 2023, Arjovsky et al., 2020, Parascandolo et al., 2020 [20, 2, 24]). This field of study recognizes that the ERM principle does not hold true for many, if not all, reallife applications. It does not assume that the data distribution of our training and test sets are identical. Instead, the framework assumes a certain distributional shift among disjoint data sets comprising the training data. Models are designed to account for these shifts. In the literature, this concept is often described in terms of environments or domains (Arjovsky et al., 2020, Peters et al., 2015, Koh et al., 2021 [2, 26, 18]). In the context of this research, an environment is a domain, is an era. Different environments define different experimental conditions that can arise when drawing data at different times or from different locations. Only data drawn from the same environment follows the same distribution, but across environments the distribution can change. Data sets in OOD generalization research are composed of data from several different environments.

There is assumed to be some distributional shift in the data-generating pro-

cess from one environment to another. The purpose of this is to recognize that there may be spurious signals that work in a single or even a group of environments, but fail to generalize to out-of-sample (OOS) data. It is shown in (Parascandolo et al., 2020 [24]) that these spurious signals can exist when many environments are pooled together. But these spurious signals change or disappear from one environment to another. These spurious signals usually will present themselves as simpler interpretations of the problem, so a naive model will quickly latch onto them. Think of the cow on grass and camel on sand image recognition problem (Nagarajan et al., 2020 [23]). During training an image classification task for predicting the label of the animal in an image, cows are always displayed on green grass backgrounds and camels always appear on sandy backgrounds. Naive ML models will simply learn to associate the color green with a cow and the color of sand with a camel. Of course, this is a severe over-simplification of the problem. During evaluation, a cow appearing on a sandy background is classified as a camel, and a camel on a grassy background is classified as a cow. A model which latches onto these spurious signals will perform well in-sample, but have severe flaws when evaluated OOS. Fortunately there may also exist *invariant* signals which work in all environments. These signals can often be much more complex than the spurious signals, so naive models are not likely to learn these invariant signals as readily. In this example, the invariant signal is the cow or camel that is actually present in the image. If an ML model truly understands what a camel or cow looks like, then it will have no problem predicting correctly, no matter the background.

Figure 1 shows a conceptual example of what eras represent in financial data. Each era spans a period of time. The boxes spanning each era contain descriptors for the macro-economic environment during each time period. The implication is that financial data coming from each time period (era) contains implicit biases due to the specific conditions of each time period. A question mark is used to show that it is impossible to describe the economic conditions in the future, since that time is still unknown. The traditional ML setting, assuming the ERM principle, implies that the data from the in sample period obey all the same laws as data from the OOS period. OOD research recognizes that this assumption is too idealized for real-life situations, where laws affecting outcomes change from one time to another, or from one place to another.

The purpose of the OOD field of research is to design ML procedures which allow models to ignore the spurious and learn the invariant signals in data, by way of a more accurate representation of the problem. In the work of (Arjovsky et al., 2020 [2]), which is considered foundational in the field, the authors define their problem as a constrained optimization problem and convert it into a loss function that can be used in gradient descent-style training procedures, such as deep learning with neural networks (Goodfellow et al., 2016 [13]). The loss function combines the traditional ERM risk term, which minimizes the average error over all environments, and an additional penalty on the magnitude of the gradients of the error in each environment. The gradient norm term is used to measure the optimality of the solution in each environment. Considerable theoretical and data analysis is performed around this idea, and the technique

Era-Wise OOD Data

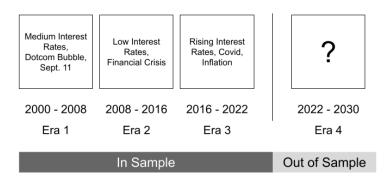


Figure 1: Financial data experience distribution shifts over time due to changing macro-economic variables and current events. OOD algorithms endeavor to uncover signals which are present in all eras (environments) instead of spurious signals only present in some.

is implemented via neural networks and shows that it works on synthetic data problems designed for this purpose, such as the Colored MNIST problem.

A similarly influential work (Parascandolo et al., 2020 [24]) also analyzes the phenomenon of spurious and invariant signals existing simultaneously in data, and how to disentangle the signals with linear models and neural networks. The authors consider the loss surface (the loss function of model parameters), and describe the co-existence of local minima on the surface where some minima correspond to spurious predictors and others correspond to the invariant predictors. The insight here is to realize that spurious minima occur is some environments and not others, but invariant minima occur in all environments in the same location. The authors extend this reasoning and consider the gradients of the loss function with respect to the model parameters. The gradient is constructed as a vector with as many elements as model parameters. The technique is to compute a gradient vector from data in each environment separately. The directions of each gradient element should agree across all environments toward invariant minima but the directions will disagree for spurious signals. In order to realize this understanding practically, the authors alter the traditional gradient descent algorithm by way of the **AND-mask** (Parascandolo et al., 2020 [24]). The AND-mask is a vector the same size as the gradient, containing only ones and zeros (true or false). Elements of the AND-mask corresponding to gradient elements that agree (have the same sign) across all environments are set to 1, and the other elements that disagree (have different signs) are set to zero. The AND-mask is then multiplied element-wise by the gradient of the loss, canceling out elements with disagreement in direction across environments. The masked gradient vector is then used in the downstream gradient descent step, propagating the model parameters toward invariant minima in the loss surface and not toward the spurious minima. The authors then show empirically how neural network models can effectively extract invariant predictors from data while ignoring spurious ones. It is important to note that the theoretical insights form the previous two references are limited to the linear settings, while the neural network models function in high dimensions and non-linear signals.

This current research article investigates the OOD generalization problem in the context of gradient boosted decision trees (GBDTs), and related decision tree algorithms. In particular we focus on regression trees, which take a continuous variable as target, as opposed to trees for classification which take a class label as target. This field of research is largely unexplored, with the only reference to the idea known at the time of writing in (Liao et al., 2024 [19]), which also looks like a promising direction. It is a related idea but distinct from this paper. The researchers define a criterion to identify invariant splits from the set of potential splits at each node. The predicted values at each node should be consistent over every era of data, to be invariant. The rule is adapted to a regularization term which is combined with the original splitting criterion. The idea is similar, but less explicit, to directional era splitting, which is explained more in detail later. In this research, we do not employ penalties, but instead replace the original split criterion with our own new criteria.

The rest of the introduction offers a review of decision trees for regression and the GBDT algorithm, highlighting the original splitting criterion used in most off-the-shelf GBDT libraries for regression (Chen and Guestrin, 2016b, Ke et al., 2017a [8, 16]). The subsequent sections then describe the novel contributions of this research. The *era splitting* and *directional era splitting* criteria are presented in sections 2 and 3, which are new contributions of this research. A theoretical breakdown follows in section 4 which offers key insights and motivations to the design choice of this research. Section 5 describes our experimental methods, specifying the OOD details of each of the 4 experiments. Finally, the paper is concluded with a discussion after presenting results. The results indicate that our new splitting criteria lead to better out-of-sample performance and a smaller generalization gap. In particular, directional era splitting stands out as the best. Links to the open source code base are given in the last section of the paper.

1.1 Regression Trees

The basis of the GBDT algorithm for regression are decision trees for regression, also known as regression trees. This section follows closely from the chapter, Regression Trees, of (Breiman et al., 1984 [6]). Regression trees are a practical tool that dates back to the 1960s (Morgan and Sonquist, 1963 [22]). It has been akin to linear regression in spirit and underlying theory, but the implementation details are much different.

The problem setup is common in machine learning and statistics, more generally known as *supervised learning*. The data consists of pairs of instances, $(x, y) \in (X, Y)$, where X is called the *measurement space* and Y is the *target space*. Usually the measurement space is the real numbers of some integer

dimension, d > 0, and the target space is usually just the real numbers, so $(x,y) \in (\mathbb{R}^d \times \mathbb{R})$. The x are called the independent (or predictor) variables and the y are the dependent (or response) variables. A prediction rule, or predictor, is a real-valued, parameterized function $f(x,\theta)$ on \mathbb{R}^d , where θ represents all the parameters of the predictor. The foundational assumption is made that

$$\mathbb{E}[Y|X=x] = f(x,\theta). \tag{1}$$

Regression analysis is the term describing how to construct $f(x, \theta)$ from a training sample \mathcal{L} consisting of N data points $(x_1, y_1), ..., (x_N, y_N)$. Before getting into the specifics of regression trees as a predictor f, let's first define terms to help us compute the error of a predictor.

Definition 1 (Mean Squared Error, (Breiman et al., 1984 [6])). The mean squared error $R^*(f)$ of the predictor f is defined as

$$R^*(f) = \mathbb{E}[(\mathbf{Y} - f(\mathbf{X}))^2],\tag{2}$$

where (X, Y) is an independent sample taken from \mathcal{L} .

Given that the training samples are also assumed to be independently sampled from \mathcal{L} , the *re-substitution estimate*, R(f), is the usual method for estimating $R^*(f)$, given by

$$R(f) = \frac{1}{N} \sum_{n=1}^{N} (y_n - f(x_n))^2.$$
 (3)

In ML parlance this would be called the *training* error or loss, since it is the error calculated with the same data that was used to train the model.

The task of regression is to find parameters of a the predictor function of independent variables that best explains a dependent variable in the least squared sense. If θ is a finite set of parameters, $\theta = (\theta_1, \theta_2, ...)$, then $\hat{\theta}$ is the parameter value that minimizes the re-substitution error, satisfying

$$R(f(x,\hat{\theta})) = \min_{\theta} R(f(x,\theta)). \tag{4}$$

In linear regression, f takes the functional form $f(x,\theta) = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2...$ where θ is a vector of real-valued numbers to be estimated. In tree regression, θ represents the tree structure, splitting rules, and parameters that go into defining the tree.

Tree predictors function by **splitting** the training data into subsets according to some defined splitting rule. The measurement space X is partitioned by a series of binary splits so that every value of $x \in X$ falls into a terminal node. Each terminal node is assigned a constant value that acts as the prediction for data instances that end up in that node. In Figure 2, an example tree is drawn, taken from (Breiman et al., 1984 [6]). Each node t_i for $i \in \{1, 2, ..., 9\}$ is either a parent node or a terminal node. Parent nodes split the data into two child

nodes according to the particular split rule for that node. Terminal nodes assign constant values $y(t_i)$ to the nodes. Any data point falls into one of the terminal nodes of the tree structure, which is assigned that node's predicted value.

The three necessary elements for building a regression tree from training data \mathcal{L} are:

- 1. A method of selecting a split at each node (a splitting criterion)
- 2. A method to determine if a node is terminal
- 3. A rule to assign the predicted value $y(t_i)$ at terminal nodes

An important point to understand is that nodes are said to contain data points. Indeed, trees are grown according to the training data \mathcal{L} . The root node, t_1 always contains all the training data. Split 1 is the rule which partitions all the data into two disjoint subsets, sending the first subset to t_2 and the second subset to t_3 . Each successive node is then split in a similar fashion, according to the data it contains, until a terminal node is reached. The rest of this section focuses on items 1 and 3 from the list above, defining the splitting criterion of each node and the predicted value of terminal nodes. The method for determining when a node is terminal is something a little less precise. In modern libraries, there are several parameters that control when to stop tree growth, such as strict limits on tree depth or total number of leaves. Growing a tree and then reducing the total number of leaves is also called *pruning*. A complete treatment of best practices to limit tree growth is beyond the scope of this text. Interested readers are invited to read the reference (Breiman et al., 1984 [6]) and review the hyper-parameters of XGBoost (Chen and Guestrin, 2016b [8]) and LightGBM (Ke et al., 2017a [16]).

Proposition 1 (Proposition 8.10, (Breiman et al., 1984 [6])). The value of y(t) that minimizes R(d) is the average of y_n for all cases (x_i, y_i) falling into t; that is, the minimizing y(t) is

$$\bar{y}(t) = \frac{1}{N(t)} \sum_{x_i \in t} y_i \tag{5}$$

where the sum is over all y_i such that $x_i \in t$ and N(t) is the total number of cases in t.

Now we take the predicted value in any node t to be $\bar{y}(t)$ and set

$$R(T) = \frac{1}{N} \sum_{t \in \tilde{T}} \sum_{x_i \in t} (y_i - \bar{y}(t))^2,$$
 (6)

where \tilde{T} is the set of all terminal nodes, and $\sum_{x_i \in t} (y_i - \bar{y}(t))^2$ is the within node sum of squares. Notice the relation between R(T) and R(f). Given a set S of potential splits in any node $t \in \tilde{T}$,

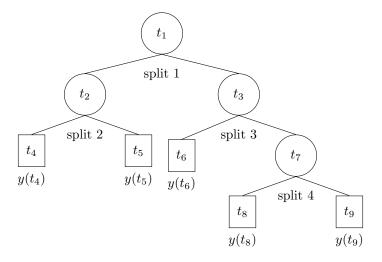


Figure 2: Figure 8.2 (Breiman et al., 1984 [6]), an example tree.

Definition 2 (Definition 8.13, (Breiman et al., 1984 [6])). The best split s^* of t is that split in S which most decreases R(T).

The format of this definition that is more common is to consider that any $s \in \mathcal{S}$ of t splits the data into t_L (left child node) and t_R (right child node), and let

$$\Delta R(s,t) = R(t) - R(t_L) - R(t_R), \tag{7}$$

then the best split s^* is the one which satisfies

$$\Delta R(s^*, t) = \max_{s \in \mathcal{S}} \Delta R(s, t). \tag{8}$$

The mean-squared error is analogous to the *impurity measure* for classification trees. The best split is the one which results in the greatest reduction in impurity of the child nodes. In practice, all split rules are defined by one of the independent variables, x_i , and some real value v, such that all data points with $x_i \le v$ are sent to t_L and the other data points go to t_R . Define such a split of node t as $s(x_i, v)$. Thus, the set S contains all possible splits $s(x_i, v) \, \forall \, i \in N(t)$ and $\forall v \in \mathbb{R}$ and each node. Since data sets are finite, the search for the best $v \in \mathbb{R}$ comes down to searching over the number of distinct values of x_i in \mathcal{L} .

This subsection described regression trees and how they fit into the field of regression analysis as a type of least squares approximation in function space. We've also reviewed how trees are grown, how tree nodes are split, and how predicted values are derived. In the next subsection we will put these regression trees into context with a review GBDTs for regression.

1.2 GBDTs for Regression

The previous section described how to grow a tree on a training data set, by splitting nodes at each step, successively reducing the error of the tree model predictor. GBDTs grow whole trees at each step, resulting in predictors which are a linear combination of many trees. Moreover, each tree is not regressed on the dependent variables y_i themselves, but on the gradients of the dependent variables with respect to each data point. This section provides a minimum foundation of GBDTs, following closely from (Friedman, 2001a [11]), concluding with the definition of the original split criterion (Chen and Guestrin, 2016b [8]) that is the standard way to score splits in GBDTs for regression.

This section continues with the notation developed in the previous subsection on regression trees. The general machine learning problem, also called the "predictive learning" problem, is described in (Friedman, 2001b [12]) as optimal function estimation. The goal is to find an estimate $\hat{F}(x)$ for the function $F^*(x)$, which maps from X to Y, that minimizes the expected value of some loss function over the joint distribution of all $(x, y) \in (X, \mathbb{R})$

$$F^* = \min_{F} \mathbb{E}_{x,y} R^*(F) = \min_{F} \mathbb{E}_x \left[\mathbb{E}_y \left[R^*(F) \right] | x \right]. \tag{9}$$

with $R^*(F)$ the mean-squared error loss function, as defined in definition 1. GBDTs belong to a class of functions called *additive expansions*, which are a parameterized class of functions $F(x,\theta)$, with θ a generic list of parameters, having the form

$$F(x, \{\beta_m, a_m\}_1^M) = \sum_{m=1}^M \beta_m h(x, a_m),$$
 (10)

with $\{\beta_m, a_m\}_1^M$ the set of parameters and $h(x, a_m)$ a generic function. In practice h will be a regression tree. This parametric formulation changes the function optimization problem to that of parameter optimization. It is described in (Friedman, 2001a [11]) how the optimal value of the parameters is often attained by starting with an initial guess, and taking successive *steps* or *boosts* based on the sequence of the previous steps toward the optimal value.

Steepest-descent, also known as gradient descent, is one such method. The gradient for element j of the gradient vector at the m^{th} step is computed as

$$\{g_{j,m}\} = \left\{ \left[\frac{\partial \mathbb{E}_{x,y} R(F(x,\theta))}{\partial \theta_j} \right]_{\theta = \theta_{m-1}} \right\},\tag{11}$$

where θ_{m-1} are the parameters resulting from the previous step. That parameter is computed as

$$\theta_{m-1} = \sum_{i=0}^{m-1} p_i,\tag{12}$$

where p_i are the results of the boosts at each step. These are defined by

$$p_i = -\rho g_m \tag{13}$$

where g_m is the vector containing all the $\{g_{j,m}\}$, and ρ is known as the *learning rate*. This is one technique for numerical optimization in parameter space. An interesting pivot is to consider optimization of a non-parameterized function $F(\mathbf{x})$ in the same way, where the data, \mathbf{x} , is taken to be the "parameter". The goal is to minimize

$$\min_{E} \mathbb{E}_{y} \left[L(y, F(\mathbf{x})) | \mathbf{x} \right] \tag{14}$$

where L is some loss function. Following the numerical optimization with additive expansion paradigm, take the solution to be of the form

$$F^*(\mathbf{x}) = \sum_{m=0}^{M} f_m(\mathbf{x}), \tag{15}$$

with $f_0(\mathbf{x})$ the initial guess and $f_i(\mathbf{x})$ the subsequent boosting rounds for $i \in \{1, 2, ..., M\}$. For steepest decent we then have

$$f_m(\mathbf{x}) = -\rho g_m(\mathbf{x}),\tag{16}$$

with

$$g_m(\mathbf{x}) = \left\{ \left[\frac{\partial \mathbb{E}_y \left[L(y, F(\mathbf{x})) | \mathbf{x} \right]}{\partial \mathbf{x}} \right]_{F(\mathbf{x}) = F(\mathbf{x})_{m-1}} \right\},\tag{17}$$

where

$$F_{m-1} = \sum_{i=0}^{m-1} f_i(\mathbf{x}). \tag{18}$$

This ideal approach is non-parametric and assumes continuous data. In practice data is finite. With sufficient regularity integration and differentiation can be exchanged, resulting the gradient estimate

$$g_m(\mathbf{x}) = \mathbb{E}_y \left[\frac{\partial \left[L(y, F(\mathbf{x})) | \mathbf{x} \right]}{\partial \mathbf{x}} \right]_{F(\mathbf{x}) = F(\mathbf{x})_{m-1}}.$$
 (19)

And so

$$F_m(\mathbf{x}) = F_{m-1} - \rho g_{m-1}(\mathbf{x}). \tag{20}$$

The final step here is a switch back to an assumption of a parameterized additive expansion, as in equation (10). This enforces a certain smoothness over the data, allowing to interpolate values for x lying in between the data points $\{x_i\}_1^N$. The function h from equation (10) is known as the base learner or weak learner. In regression, the base learner is a regression tree, which can define an output in \mathbb{R} for any input in the input space X. The optimization problem becomes a parameter estimation problem again

$$\{\beta_m, a_m\}_1^M = \min_{\hat{\beta}_m, \hat{a}_m} \sum_{1}^N L\left(y_i, \sum_{m=1}^M \hat{\beta}_m h(x_i, \hat{a}_m)\right). \tag{21}$$

The reference (Friedman, 2001b [12]) then introduces a greedy-stagewise approach to this minimization, for $m \in \{1, 2, ..., M\}$, with

$$(\beta_m, a_m) = \min_{\beta, a} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta h(x_i, a))$$
 (22)

where

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta h(\mathbf{x}, a).$$
 (23)

This is what is called *boosting* in the ML literature. Notice the similarity between equations (20) and (23). The function $\beta h(\mathbf{x}, a)$ is analogous to the steepest descent step toward $F^*(\mathbf{x})$ under the additive expansion assumption of equation (10). However, the the gradients $g_m(\mathbf{x})$ can be computed at the data points x_i , while $h(\mathbf{x}, a)$ is a parameterized function that we wish to give a best estimate of $g_m(\mathbf{x})$ for all \mathbf{x} . In order to do that, the optimal parameters a_m for the regression tree h are obtained by

$$a_m = \underset{a,\beta}{\arg\min} \sum_{i=0}^{N} [-g_m(x_i) - \beta h(x_i, a)]^2.$$
 (24)

The gradient boosting model is complete, by defining the update step as follows,

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}, a_m).$$
 (25)

In the preceding derivation from (Friedman, 2001a [11]) leaves out a step from the reference, which optimizes the learning rate(s) ρ for each update. This was left out because in modern ML libraries, the learning rate is usually chosen as a hyper-parameter of the model and kept constant throughout training. Most likely, this preference over time has had to do with the classic bias-variance trade off (Hastie et al., 2001 [15]), where fitting training data perfectly can be detrimental to model generalization. Usually, optimal parameters are tuned using a hold-out or test set of data or by cross-validation (Breiman et al., 1984 [6]).

Finally, the first algorithm presented is the gradient boosting algorithm, without tuning the learning rate, in its general form. L is any differentiable loss function, and ρ is the learning rate.

In the case of least squares regression, when the loss function is $L(y, F(x)) = (y - F(x))^2/2$, then the gradients at each step are just $g_i = y_i - F_{m-1}(x)$. This finishes our derivation of the gradient boosting algorithm for regression, in the following algorithm.

When the base learner h is a regression tree, then algorithm 2 defines the GBDT for regression algorithm. In this case, notice that line 4 of the algorithm is a least-squares fit of a regression tree via the re-substitution estimate, R(f), of equation (3).

Algorithm 1: Algorithm 1: Gradient Boost (Friedman, 2001b [12])

```
1 F_0(\mathbf{x}) = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, \rho)

2 for m = 1 to M do

3 for i = 1 to N do

4 g_i = \left[\frac{\partial [L(y_i, F(x_i))|x_i]}{\partial x_i}\right]_{F(\mathbf{x}) = F(\mathbf{x})_{m-1}}

5 a_m = \arg\min_{a,\beta} \sum_{i=1}^{N} [g_i - \beta h(x_i, a)]^2

6 F_m(\mathbf{x}) = F_{m-1} + \rho h(\mathbf{x}, a_m)
```

Algorithm 2: Algorithm 2: LS Boost (Friedman, 2001b [12])

```
1 F_0(\mathbf{x}) = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, \rho)

2 for m = 1 to M do

3 for i = 1 to N do

4 g<sub>i</sub> = y<sub>i</sub> - F<sub>m-1</sub>(x)

5 a_m = \arg\min_{a,\beta} \sum_{i=1}^{N} [g_i - \beta h(x_i, a)]^2

6 F<sub>m</sub>(x) = F<sub>m-1</sub> + \rho h(\mathbf{x}, a_m)
```

1.3 Splitting

Optimizing a regression tree f is equivalent to finding the optimal split points of the input data x_i at each node until a terminal node is reached. Each split point is defined by a single feature and a single value of that feature's data. All the data with that feature less than or equal to the split value gets partitioned into the left child node, and the data points with the feature greater than the split value go to the right child node. In successive steps the child nodes become parent nodes, and the data in those nodes is split in the same manner as the root node. Nodes are split like this until terminal nodes are reached. The function that determines the optimal split point at each node is called the splitting criterion, which measures the reduction in impurity. The split criterion was introduced in equation (7). The minimum value of $\Delta R(s,t)$ over all splits s in the set of all splits s is the split which most reduces most the average error of the tree prediction, R(T), as per definition 2 above. Minimizing $\Delta R(s,t)$ is equivalent to maximizing the original split criterion, given in the next definition.

Definition 3 (Original Split Criterion, (Chen and Guestrin, 2016b [8])). *Define the original split criterion as*

$$\mathcal{L}_{split} = \frac{1}{2} \left[\frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i\right)^2}{\sum_{i \in I} h_i + \lambda} \right], \tag{26}$$

where I, I_L, and I_R are data set identifiers corresponding to the data of the

parent node, the left child node, and the right child node respectively. The g_i are the gradients as defined in algorithm 2, above, h_i are the hessians (which are constant for regression), and λ is the L2-regularization term.

With the original split criterion defined, we have outlined the state-of-theart in GBDTs, which we will take as baseline. The original contributions of this paper will be 2 new alternative splitting criteria inspired by recent trends in the OOD generalization research. For this reason we need to first define eras more concisely.

1.4 Eras (Environments)

This research employs existing conventions to define what are called eras, also known as environments (Peters et al., 2015, Arjovsky et al., 2017 [26, 1]) and sometimes domains as in (Koh et al., 2021 [18]). This paragraph paraphrases from (Peters et al., 2015 [26]), which is the standard setup in related works (Arjovsky et al., 2020 [2]) and (Parascandolo et al., 2020 [24]). Firstly, assume there are different experimental conditions belonging to a the set of all experimental conditions, $e \in \mathcal{E}$, and we take an i.i.d. sample $(X^e, Y^e) \in (\mathbb{R}^d, \mathbb{R})$ from each environment, where X^e and Y^e are the independent and dependent variables respectively. In particular, the different distributions of X^e in the environments are unknown and not precisely controlled.

Definition 4 (Era (Environment) (Peters et al., 2015 [26])). An era (environment) refers to a particular experimental conditions e from the set of all possible conditions \mathcal{E} .

For this research, it is assumed that every data point originates from some environment, and our entire data set is the union of the data from all the environments. Practically speaking, this comes down to assigning an environmental identifier j to each data point. The identifier is an integer referring to the index of the environment the data comes from. In the following text it is usually assumed that we have data from M eras, and so the training data (X,Y) is the union of data from all M environments

$$(X,Y) = \bigcup_{j=1}^{M} (X^{j}, Y^{j}). \tag{27}$$

In the original setting, GBDTs pool all the training data together when computing the splitting criterion. If the training data inherently comes from separate environments (eras), as defined in the OOD literature (Peters et al., 2015, Arjovsky et al., 2020 [26, 2]), what is lost by pooling all the training data together to find the best split? Would it be better to employ a splitting criterion that treats data from each environment (era) separately? This research endeavors to show that the answer to this question is, "yes".

1.5 New Contributions and Outline

A new splitting criterion called *era splitting* is proposed in section 2. It combines the impurity reduction calculation from each era separately into one final new criterion, which is a smooth average of the impurity reduction of the era-wise splits. It captures the essence of invariance from the IRM paradigm (Arjovsky et al., 2020 [2]) applied to the original split criterion. Another new splitting criterion, called directional era splitting, is presented in section 3, which is designed after the AND-mask (Parascandolo et al., 2020 [24]), and looks for splits which maximize the agreement in the direction of the predictions implied by the split for each era (environment). In section 4, some theoretical concepts are discussed, highlighting the motivation for the new splitting criteria. Section 5 then describes the experimental design used to validate our new splitting criteria. The experiments include a one-dimensional toy model aimed at verifying the objectives in a simple setting, a high-dimensional synthetic dataset proposed for the same purpose in (Parascandolo et al., 2020 [24]), and two realworld empirical studies. The first is the Camelyon17 dataset (Bandi et al., 2018 [4]) breast cancer detection domain generalization problem, the second is the financial stock market cross-sectional return problem provided by the Numerai data science tournament. In section 6, the results from the experiments are presented. Finally section 7 provides interpretation of the results and discussion about future work.

2 Era Splitting

In the context of the original splitting criterion in equation (26), the data set definitions I, I_L , and I_R need to be refined when dealing with environments. With era splitting, each training data point comes from one of M distinct eras. Thus there exists data sets I^j , I_L^j , and I_R^j , referring to data coming from the j^{th} era, such that $I^j \in I$, $I_L^j \in I_L$, and $I_R^j \in I_R$ for $j \in \{1, 2, ..., M\}$. Using this convention, a new era-wise splitting criterion, which computes the information gain of a split for the j^{th} era of data is given by

$$\mathcal{L}_{\text{split}}^{j} = \frac{1}{2} \left[\frac{\left(\sum_{i \in I_{L}^{j}} g_{i}\right)^{2}}{\sum_{i \in I_{L}^{j}} h_{i} + \lambda} + \frac{\left(\sum_{i \in I_{R}^{j}} g_{i}\right)^{2}}{\sum_{i \in I_{R}^{j}} h_{i} + \lambda} - \frac{\left(\sum_{i \in I^{j}} g_{i}\right)^{2}}{\sum_{i \in I^{j}} h_{i} + \lambda} \right], \quad (28)$$

which is just the original split criterion applied to the data from one particular era. The era splitting criterion in its basic form is written as a mean average of the era-wise splitting criterion for each era of data.

$$\mathcal{L}_{\text{era split}} = \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}_{\text{split}}^{j}$$
 (29)

This criterion defined in equation (29) will have the highest score for the split which provides the greatest average decrease in impurity over all of the

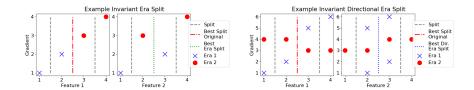


Figure 3: Degenerate split decisions induced by the traditional splitting criterion (Eq. 26) when viewed from an OOD setting. Each plot displays example data with 2 features from two eras (environments). The y-axis plots the gradients corresponding to each data point. On the left, the original splitting criterion chooses a split which doesn't improve impurity in any era. The era splitting criterion (Eq. 36) chooses a split that improves impurity in both eras. On the right the original criterion chooses a split which results in conflicting directions, while directional era splitting (Eq. 39) chooses a split resulting in consistent directions in each era.

eras of data. Let us notice that this new criterion recovers equation 26 in the case where there is only one era of data. Therefore the era splitting criterion can be viewed as a generalization of the original splitting criterion. This criterion will be less-likely to favor splits which work really well in some eras but not in others. Splits are not considered which are undefined in any era. Potential split points must have data to the left and right of them to be considered as optimal. The reason for this choice is illustrated in figure 3. The plot on the left shows a data configuration where the original splitting criterion will choose a split which doesn't induce an impurity decrease in any era of data by itself. On the other hand, the criterion in equation (29) chooses a split reducing impurity in both eras simultaneously.

A further generalization is made by way of a smooth max function, called the Boltzmann operator (Asadi and Littman, 2017 [3]). The Boltzmann operator operates on an array of real numbers, and has one parameter, called α . When $\alpha=0$, the Boltzmann operator recovers the arithmetic mean, when $\alpha=-\infty$ it recovers the min, and when $\alpha=\infty$ it recovers the max. It is known as a *smooth max (min)* function.

Definition 5 (The Boltzmann Operator). Let x_i for $i \in \{1, ..., n\}$ be n distinct real numbers, and alpha $\in [-\infty, \infty]$, then define the Boltzmann Operator \mathcal{B}_{α} as

$$\mathcal{B}_{\alpha}(x_1, ..., x_n) = \frac{\sum_{i=1}^{n} x_i \exp\left(\alpha x_i\right)}{\sum_{i=1}^{n} \exp\left(\alpha x_i\right)}.$$
 (30)

Proposition 2 (Limit of the Boltzmann Operator).

$$\lim_{\alpha \to \infty} \mathcal{B}_{\alpha}(x) = \max(x) \tag{31}$$

$$\hat{x} = \max(x)$$

and factor the \hat{x} term from both the numerator and denominator of $\mathcal{B}_{\alpha}(x)$

$$\mathcal{B}_{\alpha}(x) = \frac{\hat{x}e^{\alpha\hat{x}} + \sum_{x_i \neq \hat{x}} x_i e^{\alpha x_i}}{e^{\alpha\hat{x}} + \sum_{x_i \neq \hat{x}} e^{\alpha x_i}}.$$
(32)

Now, divide numerator and denominator by $e^{\alpha \hat{x}}$,

$$\mathcal{B}_{\alpha}(x) = \frac{\hat{x} + \sum_{x_i \neq \hat{x}} x_i e^{-\alpha(\hat{x} - x_i)}}{1 + \sum_{x_i \neq \hat{x}} e^{-\alpha(\hat{x} - x_i)}}.$$
(33)

The term $\hat{x} - x_i > 0$ for all $x_i : x_i \neq \hat{x}$, which means that

$$\lim_{\alpha \to \infty} e^{-\alpha(\hat{x} - x_i)} = 0, \forall x_i : x_i \neq \hat{x}.$$
(34)

Thus, the limit of $\mathcal{B}_{\alpha}(x)$ is found by taking the limit of numerator and denominator, noticing that the sums of exponential functions all go to zero, yielding

$$\lim_{\alpha \to \infty} \mathcal{B}_{\alpha}(x) = \frac{\hat{x}}{1} = \hat{x}.$$
 (35)

Proposition 2 for the limit and its proof can be given similarly to prove that $\lim_{\alpha\to-\infty}\mathcal{B}_{\alpha}(x)=\min(x)$. The final version of the era splitting criterion is defined next, which is understood as the smooth maximum (minimum) of the era-wise impurity reduction over all of the eras, and is a generalization of the original splitting criterion in Eq 26.

Definition 6 (Era Split Criterion). The era split criterion $\mathcal{L}_{era\ split}^{\alpha}$ is defined by

$$\mathcal{L}_{era\ split}^{\alpha} = \mathcal{B}_{\alpha} \left(\mathcal{L}_{split}^{1}, ..., \mathcal{L}_{split}^{M} \right). \tag{36}$$

The default value of α is set to zero, which recovers equation 29. Conceptually, varying α toward $-\infty$ will lead the criterion to prefer splits which increase the minimum information gain over all of the eras. This setting prefers splits that work in all eras. Conversely, varying α toward positive ∞ will cause the criterion to favor splits that improve the best single-era performance. In the prior case splits are preferred that have a more uniform affect on impurity decrease over all the eras. In the later case splits are preferred that improve our best performance in any era. In the context of learning an invariant predictor, negative values for α are preferred.

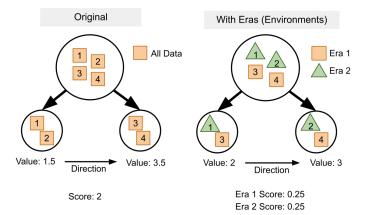


Figure 4: A schematic example of splitting data at each tree node. The target values are stored inside each data point. The original setting pools all the data together. Era splitting computes split scores on a per era basis. The value of the child nodes, the *directions* of the splits and the original and era split criteria scores are indicated. Notice era splitting does not choose the same split as the original, since it wouldn't improve impurity in any era.

3 Directional Era Splitting

There is a subtle ambiguity that arises with equation 36, which requires the introduction of the concept of the direction implied by a split. A split implies a direction via the predicted values of the child nodes. Remember from proposition 1 that the predicted value at any node is the mean of all the dependent variables y contained in the node. See figure 4. If the value of the left child node is larger than the value of the right child node, then the direction of this split is too the left. If the value of the right child node is larger, then the direction is to the right. Let the value of the left child node be v_l , and the value of the right child node be v_r , then without loss of generality, the direction of the split, $d_{\rm split}$, is defined as the sign of the difference between the left child node value and the right child node value

$$d_{\text{split}} = \operatorname{sign}(v_l - v_r). \tag{37}$$

In the original setting where all the training data is pooled together, there can be only one direction implied over the entire data set. The traditional splitting criterion (eq. 26) does not consider the direction of the split, since it is superfluous. It doesn't matter whether the direction implied by the split is to the left or to the right, since there is only one era, just as long as the impurity is reduced. In era splitting, the split criterion is computed for each era separately, and then summarized with the Boltzmann operator. A problem that arises here is that there could be splits that imply conflicting directions from one era to the

next. This is akin to single variable linear regression models having coefficients with opposite signs. We can describe the direction of a split in era j by adapting equation 37 with era-wise descriptors, as follows.

$$d_{\text{era split}}^j = \operatorname{sign}(v_l^j - v_r^j) \tag{38}$$

The directions implied by the split for each era of data results in an array of length M, $\{d_{\text{era split}}^1, d_{\text{era split}}^2, ..., d_{\text{era split}}^M\}$.

Definition 7. Directional era splitting, $\mathcal{D}_{era\ split}$, is the average agreement in the direction of a split over all M eras.

$$\mathcal{D}_{era\ split} = \frac{1}{M} \left| \sum_{j=1}^{M} d_{era\ split}^{j} \right|$$
 (39)

Each $d_{\text{era split}}^j$ will take a value of either 1 or -1, depending on the direction of the split for that era. The sum is then bounded on the interval [-M, M]. By taking the absolute value and dividing my M, $\mathcal{D}_{\text{era split}}$ is then bounded on the interval [0,1]. At the low end, with a value of $\mathcal{D}_{\text{era split}} = 0$, half of the directions go in one way while half go in the other way, the highest level of disagreement. At the high end, with a value of 1, all the directions go the same way, the most agreement. When choosing the best split during tree growth, higher values of $\mathcal{D}_{\text{era split}}$ are preferred.

4 Theoretical Breakdown

This section expounds three propositions which are illustrative of the motivations behind era splitting. Here the focus is on era splitting (eq. 36), and not directional era splitting (eq. 39). The former being more anchored in traditional theory, and connected with the original criterion. The later is new and experimental, with impurity tied to the direction of the predictions in each era. Nevertheless, the following proposition 5 applies to both. This section contains all new, original content.

One problem with traditional splitting methods in the OOD setting is that the "optimal" split can be *degenerate* in any or all of the eras (environments). The term, degenerate, indicates that the split doesn't result in improved impurity in the child nodes compared to the parent for any particular era of the data set.

Definition 8 (Degenerate Splits). A split is considered degenerate if $\mathcal{L}_{split}^{j} <= 0$, or is undefined, for any era $j \in M$ for M eras of training data. Where \mathcal{L}_{split}^{j} is defined in equation (28). The expression is undefined if the split results in one completely empty child node.

This is a split that never would have been if the tree was grown on that era of data independently. The left side of figure 3 shows a simple example of a degenerate split with a toy data set of 4 data points. The original split criterion favors the split (red dashed line) which effectively separates era 1 data from era 2 data. In each era, this split does not induce any reduction in impurity. The era splitting criterion, on the other hand, favors a split (green dotted line) which reduces impurity in both eras simultaneously. Traditional splitting methods can induce splits that don't split the data in some particular environment at all.

Proposition 3. The original splitting criterion defined in equation 26 can result in degenerate splits.

Proof. Consider a data set consisting of two eras of data, two features and two data points, as defined in table 1, and appearing on the left plot of figure 3.

The optimal split score for the original split criterion defined in equation 26 is equal to 2 and splits feature 1 between values 2 and 3. When computed independently over each era, this split results in an undefined score in both era 1 and era 2, since it does not split the data in either era. This split is degenerate.

This is one of the main conceptual motivations for the current research: it is possible that the original method misses, ignores, or misinterprets key environmental information. Era splitting addresses this problem in a very direct way: evaluate each split based on each environment separately. There is one configuration of era splitting which ensures the split points will **always** reduce impurity in every environment (era) of data simultaneously, never producing degenerate splits. This result is presented in proposition 4.

Proposition 4. The era splitting criterion, as defined in equation (36), with Boltzmann alpha parameter approaching $-\infty$ will never induce a degenerate split.

Proof. The Boltzmann operator \mathcal{B}_{α} approaches the minimum operator as the value of α approaches infinity.

$$\mathcal{B}_{\alpha} \to \min \text{ as } \alpha \to -\infty$$

Feature 1 | 1, 2, 3, 4 Feature 2 | 1, 3, 2, 4 Era Identifier | 0, 0, 1, 1 Target | -1, -2, -3, -4 Prediction | 0, 0, 0, 0 Gradient | 1, 2, 3, 4

Table 1: Data for proposition 3.

In the context of era splitting (equation 36), the input to the operator are the split criterion scores from equation 26 computed in each era independently. The minimum is the lowest score from any era. In order to induce a split, this score must be greater than zero, as per the rules of node splitting. The worst improvement will still be positive. Thus, the era splitting criterion with Boltzmann alpha parameter of $-\infty$ will only induce splits which reduce impurity in every era simultaneously.

A simple but important upshot of era splitting is that it always produces less impure splits, when viewed from the traditional setting. If we evaluate all splits using the original splitting criterion, other splits induced by other splitting criteria will, by definition, be worse (result in more impurity in the child nodes when all the data is pooled together as in the traditional setting). This means that trees grown with other splitting criteria will fit the training data to a lesser degree than the original. Impurity will not have been reduced as much. In-sample performance will be poorer. The hope is that this will result in better generalization to OOS performance. This is exactly the concept of regularization in ML. The experiments go on to confirm that this is exactly what is happening.

Proposition 5. (Era Splitting as a Regularizer)

At each node, the era splitting (eq. 36) and directional era splitting (eq. 39) criteria will always choose splits with a score **less than or equal to** (worse than) the original criterion (eq. 26), when measured with the original criterion.

Proof. Assume the original criterion (eq. 26) is used to find the optimal split point at a node. This split attains the highest score by definition. It is impossible for the era splitting (or any other) criterion to induce a split with a better score than the optimal split point, when measured using the original criterion. \Box

5 Experimental Methods

Four experiments have been designed to empirically validate the newly proposed splitting criteria. The first experiment is an original design, called the shifted sine wave, validating the era splitting models in a simple one-dimensional setting. The second and third experiments are binary classification problems from the OOD literature: the synthetic memorization data set presented in (Parascandolo et al., 2020 [24]) and the Camelyon17 (CAncer MEtastases in LYmph nOdes challeNge) data set from (Bandi et al., 2018 [4]). These two experiments confirm the new splitting criteria are effective in known OOD settings. The forth is the target application of this research, the Numerai data set.

The experiments follow a similar procedure for each. Random configurations are drawn from a grid of parameter values. The parameters include the number of boosting iterations, maximum number of leaves, learning rate, etc. All rel-

evant parameters in Scikit-Learn's HistGradientBoostingRegressor (Pedregosa et al., 2011 [25]) (the baseline model) are available. The number of random configurations (between 15 and 30) and the exact parameter ranges vary slightly from one data set to another, depending on training time and specifics of each data set. For the full list of parameters and example values, see table 2. See the code for details, referenced at the end of this subsection. Each of the three experimental models are paired with each random configuration: original, era splitting, and directional era splitting. In this way all three models are trained and evaluated over the same set of parameter values. In each case, the model is trained on the training set and evaluated on the OOS test set. Evaluation statistics are recorded for each configuration. Accuracy is the key statistic for classification problems. The regressor is converted to a binary classifier by rounding the predictions to the nearest integer (0 or 1). For regression problems the mean-squared error (MSE) and Pearson correlation (Corr) are observed.

5.1 The Shifted Sine Wave Data Set

The first experiment is a proof of concept for invariant learning in a simple regression setting. There is a 1-dimensional input feature which has an invariant aspect that is obscured by two types of randomness. One is a consistent Gaussian blur and the other is a random shift which is consistent inside each era but different across different eras of data. This part in particular simulates the distributional shift in the data generation process from one era to another.

The number of eras (environments) and the number of data points per era are pre-defined. The 1-dimensional input data is a random number on the interval $[0,2\pi]$. The target variable is a function of the the input. The invariant part of the target is a sine wave plus standard Gaussian noise. The environmental distribution shift is realized through a vertical shift of the target, where the magnitude of the shift is random, but consistent throughout each era. In this way, each era of data is a blurry sine wave that is shifted by a random amount. Pooling many eras of data together into one data set forms a cloud of data

```
Parameter
                          Example Range
Column Sample by Tree
                          0, 0.1, 0.3, 0.5, 0.7, 0.9, 1
      L2 Regularization
                          0, 0.2, 0.4, 0.6, 0.8, 1
         Learning Rate
                          0.01, 0.05, 0.1, 0.5, 1.0
  Max Number of Bins
                          3, 4, 5, 7, 9
            Max Depth
                          2, 3, 4, 5, 7, 9, 15
 Max Number of Leaves
                          5, 7, 10, 16, 32
     Min Child Samples
                          1, 3, 5, 10, 20
       Boosting Rounds
                          5, 10, 20, 50, 100, 150
             Split Type
                          Original, Era Split, Dir. Era Split
      Boltzmann Alpha
                          -2, -1, 0, 1, 2
```

Table 2: Caption

where the invariant mechanism, the sine wave, fades away and is not visually apparent. The experimental data set consists of 8 eras of data, each era having 64 data points.

Evaluation metrics are computed against the test data, which is randomly generated by an additional era.

5.2 Synthetic Memorization Data Set

The synthetic memorization data set was developed in (Parascandolo et al., 2020 [24]) specifically to investigate an ML model's ability to learn invariant predictors in high-dimensional data in the presence of confounding spurious signals. It is a supervised binary classification task with target labels taking the values {0, 1}. The data set is composed such that the first two input dimensions encode the invariant signal, which is an interlaced spiral shape, where one of the arms belongs to one class, and the other arm the other class. The remaining input dimensions are used to encode simple shortcuts, or spurious easy-to-learn signals. These are simply two clusters of data points in high dimensions, where one cluster belongs to one class and the other the other class. The clusters are linearly separable, being easy to learn. The spurious aspect is realized through a mechanism where each era (environment) of data has a shift in the location of the two clusters. The shift happens in such a way that, when all the environments of the training data set are pooled together, the shortcut is still present. Refer to figure 6 or figure 5 from (Parascandolo et al., 2020 [24]) for a visualization of the data in four dimensions. When environments 1 & 2 are pooled together, there is still an easily identifiable linear decision boundary present in the spurious signal feature dimensions.

During test time the spurious signal is replaced by random noise. Any model that bases its predictions solely on the spurious signal will be worthless when evaluated on the test data. If a model has actually learned the invariant, albeit harder-to-learn signal, it will still perform well on the test set. The authors

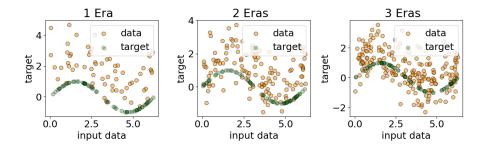


Figure 5: A visual description of the data generation process for the shifted sine wave data set. Each era of training data starts with a sine wave (green), adds a random vertical shift and a random blur (Gaussian noise).

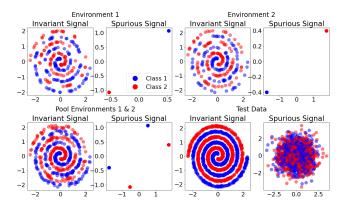


Figure 6: The synthetic memorization data set from (Parascandolo et al., 2020 [24]). Each environment contains a random sample of the invariant spiral signal along the first two dimensions. The third and forth dimension contain a simple linearly-separable *spurious* signal that changes with each environment. When pooling environments together, the pooled spurious signals still create a simple linear decision boundary. During OOS testing, the spurious signal is replaced by random noise.

of (Parascandolo et al., 2020 [24]) developed a technique called the AND-mask, which is an alteration to the naive gradient descent algorithm, incorporating the era-wise information. They were able to show that a naive MLP model simply learns the spurious signals in the data, and performs poorly on the test data. Meanwhile their AND-mask model was able to learn in such a way where it still performs well (98% accuracy) on the test data.

The training data contains 12,288 data points, each having 18 dimensions, coming from 16 eras. The test set is made of 2,000 data points. In order to use a regression model as a classifier, the model is trained on the classification labels $\{0,1\}$. Predictions are then rounded to the nearest integers, which is either a 0 or a 1. The accuracy of the predictions can then be computed in the usual way according to 2-class classification problems.

5.3 Camelyon17 Data Set

The Camelyon17 data set (Bandi et al., 2018 [4]) is an example of a real-world dataset exhibiting distributional shifts. This is one of the Wilds data sets that is distributed and supported by a team at Stanford (Koh et al., 2021 [18]). This data set aims at the automation of breast cancer detection in medical settings. The input data is a 96×96 pixel histopathological image and the target label is a binary indicator that is positive if the central 32×32 pixel region contains any tumor tissue. There are variations in data generation and processing technique from one hospital to another. The domain generalization problem is to learn to

detect tumors from images from a number of hospitals, and have this detection ability transfer to new hospitals not in the training set.

Accompanying each input data point, there is an integer identifier corresponding to the hospital. This identifier is used as the training era (environment). The raw size of the image has $96 \times 96 \times 3 = 27,648$ features, which was taxing the memory capabilities of the research computer. To reduce the memory overhead the length and width of the inputs are reduced by a factor of 1/3 using the Wilds API, reducing the number of features to a more manageable 3,072. There are 302,436 data points in the training set coming from 3 hospitals. The validation set contains 34,904 data points from one hospital and the test set contains 85,054 data points from one hospital. The Wilds data set provides established train, validation and test sets. Each model configuration is trained on the training set, and evaluated on the test set.

5.4 Numerai Data Set

Numerai is a San Francisco-based hedge fund which operates via a daily data science tournament in which the company distributes free, obfuscated data to participants, and those participants provide predictions back to Numerai. The predictions represent expected alpha, which is a directional up/down, bull/bear style prediction. On any day, a tournament participant will need to produce predictions for 5,000 to 10,000 unique stocks. The predictions are then scored against the true realizations that transpire in the markets, which then become the basis for future target labels.

Numerai distributes a large data set designed for supervised learning. Each row of data represents one stock at one time (era). The rows consist of many input feature columns, and also several target (output) columns. This research uses the main scoring target, called *Cyrus*. The Cyrus target values correspond to the optimal expected result over the subsequent 20 trading days, which is roughly equivalent to the stock's future returns in the market subject to company-level risk requirements.

The raw time periods (eras) are numbered sequentially, starting at 1, and occur at weekly intervals. Each era represents one week of time. Currently there are over 1,060 eras of historical input and target pairs in the data set, and new eras are added each week, as the data becomes available. Each era contains more than 5,000 rows of data, on average, so the entire data set consists of over 5 million rows. The time horizon of the targets spans 4 weeks. Currently the full data set provides over 1,600 input features. In order to reduce training resources, a compact set of 244 features made popular by the tournament participant ShatteredX is used as our base feature set. It was shown with a baseline model that these features lead to out performance of the model which uses all the features.

For the experimental data, the entire data set is partitioned into 5 folds over time. The first 4 folds are used for training, the the fold identifier is used as the era identifier, in place of the true era. The last fold is used for evaluation. This process is important and effectively reduces the number of eras in the data from over 1,000 to 5. The terminology can get confusing, so these larger eras are called *training eras*. Each training eras spans about 4 years of time. This is a parameter choice that can reduce the complexity of the model. Reducing the number of training eras reduces the number of iterations in the algorithm, but also can lead to different performance capabilities. The current experiment did not explore fully the whole range of training era sizes, due to time constraints.

A notable grid search parameter is the number of estimators (boosting rounds), which varies between 50 and 1,000, to conserve training time. One "benchmark" model well-known to the community is the LightGBM model with 2,000 trees, maximum depth of 5, 32 leaves, learning rate of 0.01 and column-sample-by-tree parameter of 0.1. This model is trained separately in order to make a comparison of our results to a well-known model.

The main evaluation metric is the *era-wise correlation*, which is called *corr*. The era-wise correlations are the mean Pearson correlations of the predictions with the targets computed over each era of data independently. A secondary important metric is called *corr Sharpe*, which is the mean era-wise correlation divided by the standard deviation of the era-wise scores.

6 Results

The results for all four experiments are summarized in table 3 and figure 7. Table 3 tabulates the best evaluation score over all the random configurations for each model split type. In each experiment the best model is either an era splitting model or a directional era splitting model. Figure 7 displays box plots of the evaluation metrics for both training and test sets over all the model configurations, separated by split type. A consistent trend emerges. The original split models are characterized by the best in-sample performance and the worst OOS performance, ie. the largest generalization gap. The era split and directional era split models have decreased in-sample and increased OOS performance, a smaller generalization gap, consistently in every experiment. This supports the claim of proposition 5, that these new splitting criteria act as effective regularizers.

Figures 8 and 9 visualize the decision surfaces as a function of input data values for the shifted sine wave and synthetic memorization data sets. The sine wave experiment is one dimensional and easy to plot. The original split

| Experiment | Original | Era Split | Dir. Era Split |
|-------------------|----------|-----------|----------------|
| Sine Wave (MSE) | 1.79 | 1.71 | 1.75 |
| Synth. Mem. (Acc) | 52 | 88 | 96 |
| Camelyon17 (Acc) | 60 | 65 | 7 8 |
| Numerai (Corr) | 0.0236 | 0.0203 | 0.0265 |

Table 3: Best Test Scores by Model.

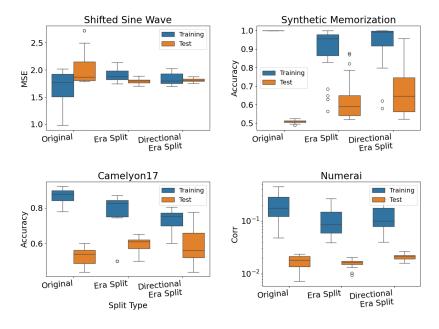


Figure 7: Box plots of the evaluation metrics for each experiment, for training and test sets.

criterion results in a much noisier decision boundary than both era splitting and directional era splitting for this configuration. The synthetic memorization data set is high-dimensional, with the invariant swirl signal encoded on the first two dimensions. Plotting the predicted values as a function of these first two dimensions reveals whether or not the model has learned to use this signal. The original split model has been unable to learn the swirl signal, as well the predicted values are very extreme, meaning the model is very confident in those predictions. The high in-sample and low OOS accuracy scores indicate that these models have only learned the spurious signals which disappear in the OOS test set. On the other hand, both the experimental models have clearly learned the swirl signal, with higher predicted values along one arm and lower values along the other.

Both new splitting criteria improve OOS results on the Camelyon17 data set, indicating that these new criteria can be useful in real-world health related applications outside of finance.

The Numerai experiment shows that directional era splitting tends to consistently surpass the original splitting criterion, while the era splitting criterion did no surpass that of the original. In the direct comparison with the 2,000 tree benchmark model, the directional era splitting attained higher OOS corr with the target, 0.0271 compared too 0.0261 of the original. Results can be more nuanced than the bottom-line evaluation metric can show. In such a large parameter space, 15 to 20 configurations does not really explore a large portion

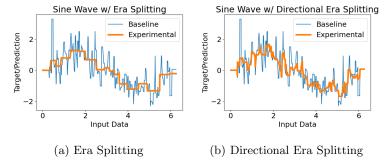


Figure 8: Visualization of the decision boundary of the baseline model vs. the experimental model. On the left era splitting, on the right, directional era splitting. Both baseline and experimental model are trained with identical base parameters: boosting rounds = 100, max depth = 10, learning rate=1, and all other parameters default.

of the space. To give more insight into the problem, there are a couple extra notebooks with experiments spanning more model parameters available in the online repository. In an additional experiment where the number of training eras is also varied from 3 to 8, several configurations of directional era splitting attained high correlation Sharpes, topping out at 1.26. That experiment can take 5 days or more to complete for 25 configurations.

7 Discussion

The new splitting criteria presented in this research improve OOS performance and reduce in-sample performance of GBDT models, reducing the generalization gap in 4 concrete experiments. This supports the claimed roll as an effective regularizer. These experiments contain known and unknown distributional shifts within the training data and also between training and tests sets. Trees grown according to our new splitting criteria are more likely to generalize in these settings than trees grown according to the original splitting criterion.

Both of the new criteria are especially effective in the synthetic memorization problem and the Camelyon17 data set. The synthetic memorization contains a non-linear spiral signal as the invariant signal, while the true invariant signals present in the Camelyon17 data set are unknown. The positive results here showcase that there is a lot of information to lose if one does not take into account the domain (environmental) information.

The Numerai data set proved to be the most difficult to improve over the baseline, but the directional era splitting consistently did. It should be noted here that the era splitting criterion also tended to out-perform in preliminary experiments where a small grid was used, with a lower number of total trees. As the number of trees increased, the era splitting criterion itself appeared to

become less effective. However there is still a lot left to explore here. While the era splitting correlation score ends up lower than the benchmark, other statistics can be much better, like correlation Sharpe and maximum draw down. The current Numerai experiment took 48 hours to train and evaluate 15 configurations. There are still many model configurations to experiment with. The number of training eras can also be varied and adapted, and this space is vast and relatively unexplored. An exploratory experiment showed some promise in this direction. In future experiments, the number of models will be expanded to several hundred, or even 1,000, which will take weeks or months to complete training. While out performance of individual models appears modest, the added variety from the the new split criteria can create the opportunity for more robust ensembles, which are averaged groups of models.

This research has taken a large step in the development of invariant learning for GBDTs, but is still in its infancy. Directional era splitting appears to work the best in real-world data sets, even though it strays farther from traditional ideas in decision trees than the era splitting criterion. Instead of variance-based impurity measures, this uses a *directional*-based measure, which is completely new to the field. This provides fresh ground for further theoretical interpretation and research.

Time Complexity

One of the biggest obstacles to the wide-spread adoption of this model is the added time complexity that is added by the era splitting routine. Indeed, one of the main issues with GBDTs in general is that, in order to grow a tree, each node must compute the split score (equation 26) for every potential split in the entire data set. The calculation must be made for every distinct value for every feature in the data. The histogram method (Chen and Guestrin, 2016a [7]) in modern software libraries has helped by reducing the number of unique values in the features down to a set parameter value. Setting this parameter to a low value, like 5, can greatly reduce training time. The colsample bytree method also reduces the total number of computations needed by randomly sampling a subset of the features to use to build each tree, similar to the way bagging can reduce the number of rows.

In Era Splitting, the split gain must be computed not once but M times per split, where M is the number of training eras in our training data. For the Numerai data, M can potentially take values up to 1,000. Run times for this implementation would take 1,000 times longer than the baseline model. This becomes prohibitive, as training times for larger models can easily take several days to complete training. In the experiment, the number of raw eras was reduced down to 5 training eras, which has a small time complexity and good performance. In other domains, such as the health domain characterizing the Camelyon17 data set, environments (such as hospitals) could be grouped together in logical ways, perhaps by geographic location. This, however, wasn't an issue for Camelyon17 data, which only came from a handful of hospitals. More progress could be made to understand the trade offs, limitations, and

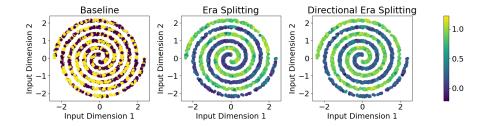


Figure 9: Comparison of the decision boundaries between the experimental models on the first two dimensions of the synthetic memorization data set. The color of the data point shows the predicted value based on that input. The original model was unable to learn the swirl signal while era splitting and directional era splitting models were able to.

performance affects pertaining to the number of training eras in the data.

8 Paper Code

All the code for reproducing the experiments presented in this paper is available in the following two links. The first contains the forked version of Scikit-Learn(Pedregosa et al., 2011 [25]) which implements the era splitting models, and the second contains all the notebooks to recreate the figures and experiments from this research.

- github.com/jefferythewind/scikit-learn-erasplit
- github.com/jefferythewind/era-splitting-notebook-examples

Acknowledgments

I would like to express my gratitude to Richard Craib, Michael Oliver, and the entire Numerai research team for their invaluable assistance and support throughout this project. This project was funded in part through an internship with Numerai from the summer of 2022 to 2023.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2020.

Kavosh Asadi and Michael L. Littman. An alternative softmax operator for reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings*

- of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 243–252. PMLR, 8 2017. URL https://proceedings.mlr.press/v70/asadi17a.html.
- Peter Bandi, Oscar Geessink, Quirine Manson, Marcory Van Dijk, Maschenka Balkenhol, Meyke Hermsen, Babak Ehteshami Bejnordi, Byungjae Lee, Kyunghyun Paeng, Aoxiao Zhong, et al. From detection of individual metastases to classification of lymph node status at the patient level: the camelyon17 challenge. *IEEE Transactions on Medical Imaging*, 2018.
- John Blin, John Guerard, and Andrew Mark. A History of Commercially Available Risk Models. In Cheng-Few Lee and Alice C. Lee, editors, *Encyclopedia of Finance*, Springer Books, chapter 97, pages 2275–2311. Springer, 6 2022. doi: 10.1007/978-3-030-91231-4.
- Leo Breiman, Jerome H. Freidman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth, Inc., 1st edition, 1984.
- Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, August 2016a. doi: 10.1145/2939672.2939785. URL http://dx.doi.org/10.1145/2939672.2939785.
- Tianqi Chen and Carlos Guestrin. XGBoost. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 8 2016b. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145%2F2939672.2939785.
- Eugene F. Fama and Kenneth R. French. The capital asset pricing model: Theory and evidence. *Journal of Economic Perspectives*, 18(3):25–46, 9 2004. doi: 10.1257/0895330042162430.
- Christian Fieberg, Daniel Metko, Thorsten Poddig, and Thomas Loy. Machine learning techniques for cross-sectional equity returns' prediction. OR Spectrum, 45(1):289–323, 2023. ISSN 1436-6304. doi: 10.1007/s00291-022-00693-w. URL https://doi.org/10.1007/s00291-022-00693-w.
- Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29(5), 2001a. doi: 10.1214/aos/1013203451. URL https://doi.org/10.1214/aos/1013203451.
- Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29(5):1189 1232, 2001b. doi: 10.1214/aos/1013203451. URL https://doi.org/10.1214/aos/1013203451.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

- Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data?, 2022.
- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.
- Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017a. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.
- Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017b.
- Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*, 2021.
- Yufan Liao, Qi Wu, and Xing Yan. Invariant random forest: Tree-based model solution for ood generalization, 2024.
- Jiashuo Liu, Zheyan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey, 2023.
- Harry Markowitz. Portfolio selection*. The Journal of Finance, 7(1):77–91, 1952.
- James N. Morgan and John A. Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58:415–434, 1963. URL https://api.semanticscholar.org/CorpusID:1825515.
- Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. Understanding the failure modes of out-of-distribution generalization. CoRR, abs/2010.15775, 2020. URL https://arxiv.org/abs/2010.15775.
- Giambattista Parascandolo, Alexander Neitz, Antonio Orvieto, Luigi Gresele, and Bernhard Schölkopf. Learning explanations that are hard to vary, 2020.

- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel,
 M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos,
 D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn:
 Machine learning in Python. Journal of Machine Learning Research, 12: 2825–2830, 2011.
- Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference using invariant prediction: identification and confidence intervals, 2015.
- V. Vapnik. Principles of risk minimization for learning theory. In J. Moody, S. Hanson, and R.P. Lippmann, editors, Advances in Neural Information Processing Systems, volume 4. Morgan-Kaufmann, 1991.