Domain Generalization, Invariance and the Time Robust Forest

Luis Gustavo Moneda dos Santos

Dissertation presented
to the
Mathematics and Statistics Institute
of the
University of São Paulo
to
obtain the title
of
Master of Science

Program: Graduate Program in Computer Science

Advisor: Prof. Dr. Denis Deratani Mauá

São Paulo, September, 2021

Domain Generalization, Invariance and the Time Robust Forest

This is the original version of the dissertation elaborated by the candidate Luis Gustavo Moneda dos Santos, as submitted to the Judging Committee.

Domain Generalization, Invariance and the Time Robust Forest

Judging Committee

- Prof. Dr. Denis Deratani Mauá - IME-USP
- Prof. Dr. Ricardo Bastos Cavalcante Prudêncio - CIN-UFPE
- Prof. Dr. Rodrigo Fernandes de Mello - ITAU

Acknowledgements

I am grateful to my parents, Venilton and Maria Aparecida, and my fiancee, Jéssica, for all the support during this time and for accepting my double journey of work and research; to my advisor, Denis, for all the discussions, patient reviews, and openness to let curiosity guide the direction of this work; to my fortune, which provided me the opportunity to investigate such an exciting subject motivated by a very particular set of people, classes, papers, books, and presentations; and to all the practitioners, students, researchers, developers, etc, who share their work and empower the whole community.

Abstract

Moneda, L. G. **Domain Generalization, Invariance and the Time Robust Forest**. 2021. 120 f. Master thesis - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2021.

As time passes by, the performance of real-world predictive models degrades due to distributional shifts. Typical countermeasures, such as retraining and online learning, can be costly and difficult to implement in production, especially when business constraints and culture are accounted for. Causality-based approaches aim at identifying invariant mechanisms from data, thus leading to more robust predictors at the possible expense of a decrease in short-term performance. However, most such approaches scale poorly to high dimensions or require extra knowledge such as segmentation of the data in representative environments. In this work, we review the literature on the limitations of Machine Learning in real settings, with a focus on approaches that use causality concepts to improve generalization. Motivated by the shortcomings discussed above, we develop Time Robust Forests (TRF), a new algorithm for inducing decision trees with an inductive bias towards learning time-invariant rules. The algorithm's main innovation is to replace the usual information-gain split criterion (or similar) with a new criterion that examines the imbalance among classes induced by the split through time. Experiments with real data show that our approach can improve long-term generalization, thus offering an interesting alternative for dynamical classification problems.

Keywords: Causal Invariance, Domain Generalization, Inductive Bias.

Resumo

Moneda, L. **Generalização de domínio, Invariância, e a Floresta Temporalmente Robusta**. 2021. 120 f. Dissertação de Mestrado - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2010.

Com o passar do tempo, o desempenho de modelos preditivos em dados reais degrada devido a mudanças na distribuição dos dados. Medidas típicas como o retreino e aprendizado em tempo-real podem ser custosas e difíceis de implementar em produção, especialmente quando restrições de negócio e cultura organizacional são levados em conta. Abordagens baseadas em causalidade buscam identificar mecanismos invariantes nos dados, resultando em preditores mais robustos às custas da diminuição de desempenho no curto prazo. Grande parte dessas abordagens, porém, não escala bem com alta dimensionalidade, ou requer conhecimento extra, tal como a segmentação do conjunto de dados em ambientes representativos. Neste trabalho, revisamos a literatura sobre as limitações do Aprendizado de Máquina em cenários reais com um foco em abordagens que usam conceitos de causalidade para melhorar a generalização. Motivados pelas deficiências discutidas acima, desenvolvemos a Floresta Temporalmente Robusta (TRF), um novo algoritmo para induzir árvores de decisão com um viés indutivo para o aprendizado de regras temporalmente invariantes. A inovação do algoritmo está em substituir o habitual critério para divisão baseado em ganho de informação por um novo critério que toma em consideração o desbalanceamento entre as classes a serem separadas em uma perspectiva temporal. Experimentos com dados vindos de aplicações reais mostram que nossa abordagem pode melhorar a generalização no longo prazo, oferecendo desta forma uma alternativa para problemas de classificação de caráter dinâmico.

Palavras-chave: Invariância causal, Generalização de domínio, Viés indutivo.

Contents

List of Abbreviations

OOD	Out Of Distribution
ICP	Invariant Causal Prediction
IRM	Invariant Risk Minimization
TRT	Time Robust Tree
TRF	Time Robust Forest
ERM	Empirical Risk Minimization
i.i.d	Independent and Identically Distributed
SCM	Structural Causal Model
DAG	Directed Acyclic Graph
PIMP	Permutation Importance
ICM	Independent Causal Mechanism
ANM	Additive Noise Model
CART	Classification and Regression Trees
MMD	Mean Discrepancy Distance
RKHS	Reproducing Kernel Hilbert Space
RCT	Randomized Controlled Trials
DT	Decision Tree
RF	Random Forest
GI	Gini Impurity
AUC	Area Under the Curve
GAM	Generalized Additive Model

List of Symbols

Y	Target variable
X	The set of input variables
X_i	A specific feature i from the input variables
PA_i	The set of parent variables of a certain variable i
T_{stamp}	Timestamp information
T_{period}	The set of segments of time
s^*	Optimal feature split
f^*	The feature from the optimal feature split
v^*	The value of the optimal feature split
F	The set of available features
X_{node}	Examples contained in a node
$\hat{y}$	Predicted target
X_{left}	Examples which lie in the left when feature split is performed
X_{right}	Examples which lie in the right when feature split is performed
ρ	Minimum number of examples by period to split
d	Maximum depth
M	Number of estimators
e	Environment variable

List of Figures

List of Tables

Chapter 1

Introduction

Machine Learning techniques are mainly evaluated by their ability to generalize, that is, to find useful patterns from a training data sample that satisfactorily apply to unseen instances [Bis06]. Typically, that process involves a time dimension: training data refers to the past, while unseen instances will be obtained in the future. This temporal characteristic is usually dismissed by a time-stationary assumption of the data generating distribution. In practice, sampling distributions are seldom stationary, which causes spurious correlations to be learned.[1] By blindly minimizing training error (or empirical risk), Machine Learning models absorb such relationships [ABGLP19] and fail to generalize when deployed in a real setting, even when a generalization promise from the validation stage is observed [DHM+20, RSG16]. This notion of generalization beyond the distribution observed in the training set is called Domain Generalization [GLP20], or Out of Distribution (OOD) Generalization [Arj21].

Spurious correlations can be defined as the non-causal statistical relationships between the target and non-target variables [Pea97, Pea09]. Causal analysis, however, is most commonly used on problems involving interventions or counterfactuals, as ensuring causal ordering of the relationships can be detrimental to predictive performance (hence to generalization). Recently, however, researchers have started advocating the benefits of ensuring properties of causal relationships even for purely predictive problems [GLP20].

One such property is invariance: causal relationships are invariant with respect to different environments, which are external settings of the covariates [PJS17, Car03]. By enforcing invariance in a learning algorithm, we regularize against spurious correlations [ABGLP19]. For example, Invariant Causal Prediction [PBM15] learns invariant predictors by finding a subset of the covariates such that the corresponding residual error when regressing on the target variable follows certain properties. Invariant Risk Minimization [ABGLP19] attempts to remedy generalization issues by optimizing an objective function that penalizes lack of invariance. The approach is more suited for parametric models, and a practical objective function has only been derived for linear models.

The previous approaches strongly rely on acquiring data from different environments. Such data can be collected from similar but different sub-populations [MPJ+16, SS18], or they can be fabricated by manipulation, such as by altering image backgrounds [ABGLP19]. Another prevalent source is time: as time passes by, we obtain different environments by a process known as distributional shift [QCSSL09]. Since data are often collected through extensive periods, obtaining environments by segmenting time periods is appealing. Environment diversity is then corroborated by observing that model performance decays over time [LLD+18, Rea18].

[1] There is often an inductive bias in learning algorithms towards estimating simpler accurate models, and for complex tasks, spurious correlations are often simpler than non-spurious ones [WRS+17, ABGLP19]. A typical example is learning to classify a wolf image by the presence of snow.

1.1 Contribution

In this work, we develop Time Robust Forests, a new recursive partitioning learning algorithm for generating classification or regression tree ensembles robust against distributional shifts. The algorithm takes time-stamped data segmented in groups composed of sequential examples to represent environments, which it uses to learn more stable predictors. By exploiting the prevalent available time ordering of data instances, we can balance short-term prediction performance and robustness to out-of-distribution data, with nearly no overhead to the user. Being a recursive partitioning learner, the proposed approach offers the flexibility of a non-parametric and nonlinear model while inheriting a rich toolset for model analysis and interpretation [BFSO84].

1.2 Outline

The rest of this monograph is organized as follows: In Chapter 2, the needed concepts related to predictive tasks are presented, like Empirical Risk Minimization, Validation, Dataset shift, and the algorithms for the Decision Tree and Random Forests. At the same time, we briefly introduce the challenges of generalization. In Chapter 3, we present the necessary concepts from the Causality field to motivate their exploration to overcome the current challenges regarding generalization: Spurious correlation, Causal Forests, Structural Causal Models, Independent Causal Mechanisms, Interventions, and Environments. In Chapter 4, we review a couple of the current approaches to achieve domain generalization (Invariant Causal Prediction (ICP), Invariant Risk Minimization (IRM)) and briefly introduce others. In Chapter 5, the Time Robust Trees (TRT) and Time Robust Forests are presented, followed by a motivational and a synthetic example to illustrate them, while in Chapter 6 we test these algorithms in a variety of real-world data cases. Discussions about the advantages, the limitations, and future work are found in Chapter 7.

Chapter 2

Statistical Learning Theory

2.1 The Empirical Risk Minimization inductive principle

The statistical learning theory seeks to answer which cases and how well one can learn a function that maps one or more random variables to a random variable of interest. This fundamental problem can be stated as the desire to learn the function f^* that maps a random variable X to a target variable Y, $X \xrightarrow{f^*} Y$. To learn it, we have at our disposal a training sample S, which means n pairs of instances in which we observe X and Y associated, $S = \{(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)\}$. We assume the function f^* we intend to learn belongs to a set of possible functions called hypothesis space, $f \in \mathbb{H}$. With all these elements, a learning algorithm is determined to learn a particular $\hat{f}$ in $\mathbb{H}$. A loss function L is defined to evaluate the quality of the learned function $\hat{f}$ in the task of associating the variables in X to Y, so it is possible to choose the best solution for this problem. We want to minimize the expected loss considering the unknown distribution of X and Y, $p(X, Y)$. The expected loss is called risk, represented for a certain function f as:

$$R(f) = \int L(Y, f(X)) dp(X, Y) \tag{2.1}$$

Since we do not observe $p(X, Y)$, the practical setting of the learning problem is to minimize the Equation 2.1 in respect to an independent and identically distributed (i.i.d.) sample from $p(X, Y)$ [Vap13], known as the Empirical Risk:

$$R_{emp}(f) = \frac{1}{n} \sum_{i}^{n} L(Y_i, f(X_i)) \tag{2.2}$$

The properties provided by this learning theory about the convergence of this empirical approach and how well it generalizes depend on the i.i.d. assumption. The learned function $\hat{f}$ is what we will call a model[1].

2.2 Generalization

Generalization is a widely used term in Machine Learning. The general meaning relates to the ability to understand new instances of a problem after extracting supposedly general rules from a number of specific examples. The f function presented in the ERM principle dictates the relationship between any example coming from the distribution $p(X, Y)$, so there is no distinction between how well f generalizes for any other random sample from this distribution. However, the statistical learning problem we have presented is about learning from the training sample S, which has a distribution $p_{train}(X, Y)$. If we expect it to do well in $p(X, Y)$, it means we expect it to do well in random samples from it. As a proxy for this random sample, a distribution $p_{test}(X, Y)$ obtained from a different data set S_{test} is often used. While, in theory, generalization means doing well in

[1]The term model is overloaded since it is used to describe the data generating process and its joint distribution.

$p(X, Y)$, in practice, it is assessed by how well the model does in $p_{test}(X, Y)$. The risk representing the generalization power of the learned $\hat{f}$ is represented by the risk $R_{test}(f)$ calculated replacing $p(X, Y)$ with $p_{test}(X, Y)$ in Equation 2.1.

2.3 Model selection and validation

Machine learning success is backed by its empirical nature that enables one to verify the prediction power of the models. There are two stages during model development where the model is assessed. The first one is the Model Selection stage [K+95] [Sto74], in which one wants to select the best procedure possible among all the alternatives since there is typically no single best option [WM+95], including learning algorithm, data pre-processing, feature engineering, hyper-parameters, and decision threshold. The second stage is called validation, and it estimates how the selected model from the previous stage will perform when used in a random sample of the true distribution. Performance metrics are estimated in both of them as a proxy of the generalization power of the models, so we can select the best one and estimate how good the solution using it will be when facing the real-world data. While it is common to find practitioners that name either stage as validation, each stage performs very different assessments. In the first stage, data is used to make modeling decisions. In the second stage, data is used only to estimate future performance (so no decisions are taken here).

More important than the usage of the performance metrics in both stages is the way they are designed and their implications on how well they represent the generalization power of the model [DHM+20]. In the following subsections, we review the most common design choices for model selection and validation.

2.3.1 Data split: train, validation and test sets

Inspired by the theoretical framework of ERM, in which we learn from a sample and performance boundaries are guaranteed for a random sample from the same distribution used to train, a straightforward validation design is to randomly select a proportion of the data available to train the model. In contrast, the remaining examples are used to assess its quality [Lar31]. Since a couple of modeling decisions need to be made and practitioners want to use performance as a compass, a third split is created, which is not used to train the model but to evaluate how it is impacted by different choices for input variables (feature selection), type of model, hyper-parameters, etc. A usual nomenclature is "Training set", "Validation set" and "Test set", or "Holdout set". However, practitioners will often use the last three terms interchangeably[2].

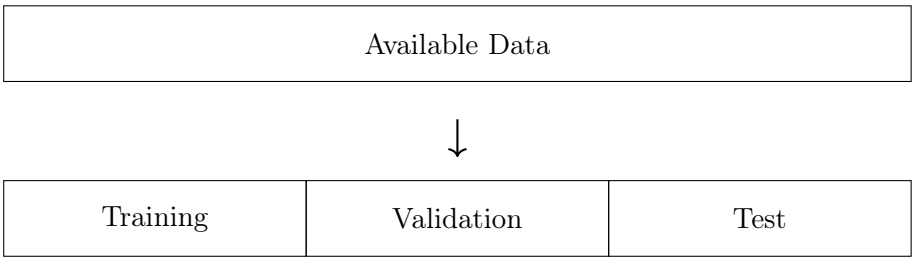


Figure 2.1: *A common practice to evaluate models is to pretend we have unseen random i.i.d data by splitting all the available data into three groups. The first is used to take modeling decisions, while the second provide the final assessment in terms of predictive performance*

Another common way of splitting the data to assess generalization is the K-fold cross-validation [MT68]. It consists in slicing the data into K groups and performing training K times using every

[2]Note the confusion: the validation stage uses the test set portion, while the validation dataset is used in the model selection stage to provide a less biased evaluation of different modeling decisions.

slice as the test data once. In the end, the K measurements can provide average and standard deviation about the performance metric.

All the mentioned methodological designs enable the user to perform bootstrapping, a resampling technique consisting of sampling with replacement to have a variability measurement of the estimated metrics, a common practice when the sample size is small or when a significant variance in the results is expected.

2.4 Why generalization is challenging

The ERM is a widely used principle in Machine Learning as many learning algorithms indeed rely on learning from an observed sample and have their properties guaranteed when they need to generalize to a new i.i.d. sample. It turns out this assumption does not hold most of the time, primarily due to what is called dataset shift.

Dataset shift can materialize as different categories of problems, and we focus on three of them. We can have a change in the inputs' distribution, $p(X)$, a covariate shift; or we may have a change in p(y), the output, a target shift, if $p(Y \mid X)$ is fixed, and a generalized target shift if $p(Y \mid X)$ also changes; finally, we may have a change in $p(Y \mid X)$, while $p(Y)$ is fixed, a conditional shift, also called concept drift or pure concept drift [ZSMW13] [QCSSL09]. The term concept drift or shift is sometimes used interchangeably with dataset shift.

When we have a covariate shift, it might happen that the new distribution does not share the same input features' range observed in the training examples. In a non-parametric model, the values might be truncated by what was observed. In contrast, a parametric model tries to generalize following what we have learned with the available data and the given constraints about the functional form. The generalization estimates will not reflect the actual performance, as distributionally different examples are feeding the model. In the case of a learning algorithm that learns the optimal average case, it will no longer be optimal since the average case has changed [Shi00] [SSN+08].

A target shift causes problems depending on the assumptions of the model [Sto09]. In Naive Bayes [HY01], for example, a model of the joint distribution $p(X, Y) = p(X \mid Y)p(Y)$, which is used to compute $p(X \mid Y)$. Thus, a significant change in $p(Y)$ between train and test time will cause the degradation of the predictions. If the change in test for $p(Y)$ is known, considering that $p(X \mid Y)$ is fixed, it is possible to learn from the train and succeed in the test, but it is not the usual case.

Concept shift causes problems because it means a change in the relationship between one or more input variables and the target, $p(Y \mid X)$. For example, if high average temperatures are positively associated with ice cream consumption, a concept drift could mean it becomes less associated, disassociated, or inversely associated. Since a model has learned in a training set the initial association and counts on it to do well in future predictions, any change in this association will degrade the model performance.

There are other kinds of dataset shifts described in the literature: domain shift, when changes in measurements make the relationship between variables change; sample selection bias, when the distribution changes due to different filtering processes applied in training data and the sets the model will be applied; source component shift, when the data is composed by different sources and their proportion change; and imbalanced data, a deliberate dataset shift to favor some modeling approaches [QCSSL09].

The behavior described in dataset shift brings a dynamic view of the data, while the described framework under ERM expects a static and well-behaved dataset to learn from and generalize for unseen instances. This dynamic nature of data is one of the things that makes generalization hard by minimizing training error.

2.5 Domain Classifier

A Domain Classifier evaluates how much dataset shift there is in a particular problem. The training data is also called the source domain, and the data on which we want to apply the model

to solve a task would be the target domain [RGL18]. As in validation, we can use part of the source domain as a target domain to emulate the unseen future situation. Framing the ERM assumption using these definitions, we can say that learning algorithms expect the source domain and target domain to be the same, to follow identical distributions and relationships. One way to verify how far from this assumption we are is to use a Domain Classifier [RGL18]. This technique uses a model trained to classify data points into source or target domains. For example, one can use the temporal data split from the previous section, label the train data and the out of time holdout differently, and build a binary classifier to identify them. A model with a strong performance in such a task can easily differentiate past data from future data, which means the data from the future is very different from what was observed in the past, while a bad classifier means they are indistinguishable. If the classifier cannot differentiate source and target domains, learning from the source will be helpful to generalize in the target, while the opposite makes it hard to extrapolate in the target domain.

2.6 Validation under dataset shift

Motivated by the challenges exposed in the Section 2.4 and the notion of source and target domain described in the Section 2.5, instead of using a random split of the available data, it became common to split it temporally, creating a past training set and assessing performance in future unseen data. This design does not solve the challenges described, but it enables practitioners to evaluate its effect and take action if needed. The more different types of dataset shift impact the future data, the harder it is for a model trained on past data to generalize to the future data. If we split the data in random proportions, we would lose the notion of dynamism and evaluate the model in a likely unrealistic setting. The temporal split means the model will face a data distribution that is more similar to the reality, which makes the validation stage closer to its final goal: assessing model generalization power when applied to new examples.

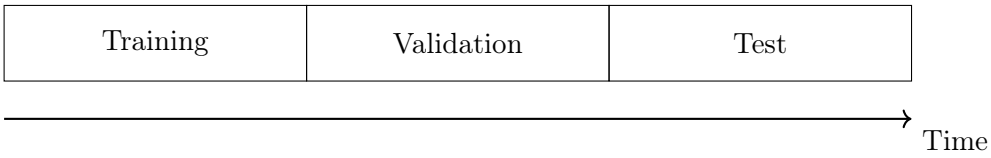


Figure 2.2: *In order to make the validation stage closer to the real scenario, instead of using a random sample, a time split is performed to offer the intuitive set of using past data to predict future outcomes. This practice shows how validation strategy changed to better address the problem of spurious correlations.*

2.7 Decision Tree

Classification and Regression Trees (CART) [BFSO84], also known as Decision Trees, is a partitioning representation that splits instances according to their characteristics until a decision or inference can be made. To learn a decision tree classifier or regressor, algorithms such as ID3 [Qui86], and C4.5 [Qui14] use a recursive approach that partitions the data using the available input information until it reaches different stop conditions to make a leaf, where it uses the containing examples to make a prediction.

The splitting criterion differs depending on whether it is a regression or a classification task or simply by different choices for the same task. In regression, one can use the variance reduction [BFSO84], which tries to reduce as much as possible the variance of the target variable in the two generated nodes after splitting in respect to the previous variance observed in the parent node, as seen in the Equation 2.3, where S_{left} and S_{right} are the sets of examples contained in the left and right splits. At the same time, y_i is the ith example of the target variable, and μ is the average of the target for the respective split. In the classification case, the Gini Impurity measures the mistakes of predicting the majority class, as given by Equation 2.4. In that equation, J is the number of

classes, p_i is the probability of being from the class i. Another option for the classification task is the Information Gain [Qui86] that uses the Shannon Entropy $H(X)$ [Sha48], where T represents the training set, $(\mathbf{X}, Y) = (X_1, X_2, X_3, ..., X_k, Y)$; and $S_a(v)$ represents the subset from T where the feature a has a certain value $v \in vals(a)$, that is, $S_a(v) = \{x \in T \mid x_a = v\}$. We see in Equation 2.5 that the IG compares the entropy difference with and without the split using the feature a.

$$I_V(N) = \sum_{i \in S_{left}} (y_i - \mu_{left})^2 + \sum_{i \in S_{right}} (y_i - \mu_{right})^2 \tag{2.3}$$

$$I_G(p) = 1 - \sum_{i=1}^{J} p_i^2 \tag{2.4}$$

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n} P(x_i) log P(x_i) \\ H(T \mid a) &= \sum_{v \in vals(a)} \frac{|S_a(v)|}{|T|} . H(S_a(v)) \\ IG(T, a) &= H(T) - H(T \mid a) \end{aligned} \tag{2.5}$$

There are different stopping conditions in Decision Trees that work as hyper-parameters of the model. The objective is to reduce overfitting since the limit case splits the data until the leaves contain a single example. The *maximum depth* constraints the number of times the algorithm splits the data by stopping it as soon as the current depth is equal to it. The root node is at depth 0, and after a split, we increment it by one; the *minimum samples to split* considers we have a minimum sample in a node in order to split it; the *minimum samples at leaf* constraints the number of examples needed to consider a node a leaf, which means the minimum samples to make inferences. Though there are other hyper-parameters the state-of-art decision trees implement, these are the most relevant for this work.

We describe a simple version of the ID3 in the Algorithm 1. We do the first call to LearnDecisionTree with all the training data in X, a minimum number of examples to keep splitting it, σ, and the maximum depth we want the tree to grow to, d. Notice σ and d constitute the stop conditions. If these conditions are not met, we split the data by calling CreateSplit, which finds the best split, apply it to the data creating two samples of it, X_{left} and X_{right}, to which we keep recursively splitting until the stop conditions are met. To find the best split, we use the FindBestSplit, which iterates in all possible variables and their values to evaluate them considering a specific criterion, like the Gini Impurity.

2.8 Random Forest

A Random Forest [Bre01] is an ensemble of Decision Trees, which means it combines many Decision Trees solving the same task to get a stronger model. If all the trees making the forest were equal, there would not be any benefit in combining them. In order to provide some variability to the trees, Random Forest has two main strategies: bootstrapping the training examples before building a new tree and taking into consideration only a subset of the features when searching for the optimal split. The number of trees in the forest is controlled by the *number of estimators* and the number of features considered when splitting by the *maximum features* hyper-parameter.

If a Random Forest contains M estimators, the final prediction $\hat{y}$ becomes $\frac{1}{M} \sum_{m=1}^{M} \hat{y}_m$, which means it takes the average result as the final output.

Algorithm 1 Decision Tree ID3 Algorithm

```
1: procedure LearnDecisionTree(X, σ, d)
2:     if d ≥ 1 and there are at least σ examples in X then
3:         CreateSplit(X, σ, d)
4:     end if
5: end procedure
6:
7: procedure CreateSplit(X, σ, d)
8:     X_left, X_right = FindBestSplit(X, σ)
9:     LearnDecisionTree(X_left, σ, d − 1)
10:    LearnDecisionTree(X_right, σ, d − 1)
11: end procedure
12:
13: procedure FindBestSplit(X, σ)
14:    score = -inf
15:    for Every variable f in X do
16:        for Every value v_f of f do
17:            X_left = examples where X_f ≤ v_f
18:            X_right = examples where X_f > v_f
19:            if Number of examples in X_left and X_right is greater than σ then
20:                current_score = GiniImpurity(X_left, X_right)
21:                if current_score < score then
22:                    score = current_score
23:                    f* = f
24:                    v*_f = v_f
25:                end if
26:            end if
27:        end for
28:    end for
29:    if score ≠ -inf then
30:        X_left = examples where X_f* ≤ v*_f
31:        X_right = examples where X_f* > v*_f
32:    else
33:        X_left = {}, X_right = {}
34:    end if
35:    return X_left, X_right
36: end procedure
```

Chapter 3

Causality concepts

In this Chapter, we present some concepts related to causality to connect generalization to the data generating process described by a Structural Causal Model (SCM), a series of relations seen as descriptions of causes by their effects. The SCM is needed to define interventions and environments, which are needed to describe the principle of the Independent Causal Mechanism. The principle motivates the idea of invariance through environments, the core idea behind the inductive bias on which Time Robust Trees are built. Although the contribution regards the usage of this new algorithm for predictive tasks and without using a causal representation, the motivation and the property behind it comes from the Causality field.

3.1 Structural Causal Models

To represent causal relationships, it is common to use Structural Causal Models (SCM) [Pea09], an extension of the Structure Equation Modeling [Wri18].

Definition 1. *A Structural Causal Model is composed of three sets:* (V, U, F)*, where* V *is the endogenous variables set, which are defined by other variables inside the considered system;* U *are the exogenous variables, assumed to be fully independent, , whose values are defined outside the system;* F *is the set of functions that maps the values of the variables in* V *to other variables in* V *and* U*, in the form* $V_i = f_i(PA_i, U_i)$*, where* PA *are the parents of* V_i*, that is, the set of variables in* V *that define* V_i*,*

We can use the information about which variables define each other to factorize the joint distribution.

$$p(V_1, ..., V_n) = \prod_{i=1}^{n} p(V_i \mid PA_i)$$

A Directed Acyclic Graph (DAG) is a graphical representation of the relations in an SCM made by connecting with directed arrows the variables in PA_i to V_i. For example, consider the following SCM:

$$\begin{aligned} X_1 &\leftarrow U_1 \\ X_2 &\leftarrow X_1 + U_2 \\ Y &\leftarrow 2X_2 + U_3 \end{aligned} \tag{3.1}$$

We can represent it as the following DAG, where we omit the U:

$$X_1 \longrightarrow X_2 \longrightarrow Y$$

The DAG offers a graphical representation of an SCM, which provides clarity on the assignment's assumptions and enables the creation of graphical criteria for causal concepts [Pea09].

One important theorem is the Causal Markov Condition, which states that every variable V_i is independent of all its non-descendants given its parents $PA(V_i)$ under any intervention, a concept we will define in a following section [VP91]. Using the example provided, it means that $X_1 \perp\!\!\!\perp Y \mid X_2$.

3.2 Spurious Correlation

Intuitively, spurious correlations are correlations that do not stand the test of time [ABGLP19].

To define correlation, Pearson [A+95] starts from a causation definition before he states the importance of this broader association type, which ranges from absolute independence to complete dependence. An event A would be considered a cause of the event B if A always precedes B, and without A, B would not take place [Pea97], a notion later criticized since the B can have multiple sufficient causes, which would make A not necessary. Having the notions of causation, Pearson states that "Causation is correlation, except when correlation is spurious, when correlation is not causation" [A+95]. Everything that presents association, when it is not via a causal relationship, is a spurious association. For Pearson, correlation is powerful because though it is not always causation, sometimes it is.

Yule (1926) highlights the kind of relationship in which even though two events might not present a causal relationship, they can be related to each other indirectly by a very indirect chain of causation. He considers this is the case when completely nonsense variables show association for a short period [Yul26]. A classical repository for this case is the "Spurious Correlation website"[1], where one can find, for example, a plot showing a high correlation between divorce rate and margarine consumption. Yule (1926) characterizes this case as a result of a small sample in a shorter period than needed to give us the true correlation. Though the discussion from Yule and the examples from the Spurious Correlation website are about this exact case, time series in a reasonably short time, it is not that rare that this kind of nonsense correlation happens in large tabular data. We take the parallel of something that happens by chance in a sample and that we would probably not keep observing if we have access to the population. Some features correlate with the target in supervised learning even if shuffled, a fact that enabled feature selection procedures like Permutation Importance (PIMP) [ATSL10].

Reichenbach (1956) [RR56] introduces a third variable to define causality on his common cause principle, according to Peters (2017) [HDPM17].

Definition 2. *If two random variables X and Y are statistically dependent ($Y \not\perp\!\!\!\perp X$) there exists a variable Z that causally influences both. (As a special case, Z may coincide with either X or Y). Furthermore, this variable Z screens X and Y from each other in a sense that given Z, they become independent, $X \perp\!\!\!\perp Y$.*

Thus this variable Z, also known as a confounder when not in the special case, becomes a source of a spurious correlation when not considered in the analysis or not observed.

A classic example of the common cause case, known as confounding, is the observed spurious correlation between ice cream consumption (X) and crime rate (Y), which is confounded by day temperature (Z). So in the summer, when it is hot, people consume more ice cream, but they also occupy more public spaces and are more prone to crimes. We represent it by the DAG in Figure 3.1. In this case, the spuriousness can be spotted in the causal structure that defines the phenomena involving X, Y, and Z, which does not match the nonsense view of spuriousness since we can come up with a story to explain it. We are not talking about something we observe by chance because Z would cause this confusion about X and Y under every circumstance. The confounding case does not cover other sources of non-causal association between X and Y, which is not related to a

[1] https://www.tylervigen.com/spurious-correlations

common cause. One example is the conditioning in a common effect, which creates selection bias. A typical example of this source of spuriousness regards the selection of a prestigious school that uses sporting ability and academic ability to select its students. In the general population, these traits are uncorrelated, while looking at the selected students in these schools will make us observe a spurious negative correlation between academic and sporting ability [PM18].

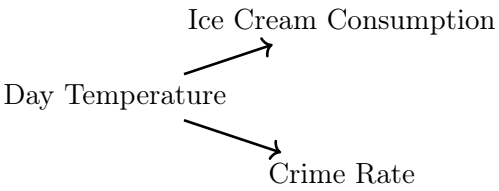


Figure 3.1: *The common cause setting illustrates the concept of confounding, when a variable (Day Temperature) impacts both the cause we are interested in isolate (Ice Cream Consumption) and outcome (Crime Rates), which brings confusion to the relationship between the cause and the effect of interest.*

As noted by Simon (1954) [Sim77], the spurious correlation problem needs a "precise and operationally meaningful definition of causality". Reinchenbach (1956) [RR56] took it further from Pearson's definition, then Pearl (2009) [Pea09] has offered theorems, concepts, and tools to enable manipulation and study of phenomena related to cause and effect.

From Pearl's framework, we present the definition of spurious association between two variables X and Y [Pea09].

Definition 3. *Two variables X and Y are spuriously associated if they are dependent in some context and there exists two other variables (Z_1 and Z_2) and two contexts (S_1 and S_2) such that:*

1. Z_1 and X are dependent given S_1 (i.e $Z_1 \not\perp\!\!\!\perp X \mid S_1$)
2. Z_1 and Y are independent given S_1 (i.e $Z_1 \perp\!\!\!\perp Y \mid S_1$)
3. Z_2 and Y are dependent given S_2 (i.e $Z_2 \not\perp\!\!\!\perp Y \mid S_2$)
4. Z_2 and X are independent given S_2 (i.e $Z_2 \perp\!\!\!\perp X \mid S_2$)

Notice this does not cover the common cause case since when we can observe all the variables involved under the common cause, it would not produce a correlation between X and Y under the causal Markov condition. The context variables S_1 and S_2 accept the empty case. Whenever we observe a dependence between X and Y, and with the help of the control variables Z_1 and Z_2, given the enumerated conditions, the only explanation for the dependence is a latent common cause, a spurious association.

After starting this section with intuition and going through a couple of definitions, we get back to intuition because the current causation concept is a construction to enable mathematical manipulation and not a strict notion. Since spuriousness is constantly defined as causation absence, it makes the latter contain a myriad of meanings. In general, we expect spurious relations not to keep up, whether when we go from a sample of the data to the population, from one period of the time to the next, or from one place to another. Even within this general view, though, we find gaps. We can illustrate these gaps adapting Leibniz's concepts of necessity and contingency [Ada82]. Necessary relations are true in every possible world, like gravity, while contingent ones are open to different outcomes due to free will, as the relationship between salary and education years. A contingent relationship could offer a certain degree of invariance, but we would hardly admit this relationship is not open to changes.

Since characterizing the nature of the phenomena that orbits causation and randomness is entirely out of scope, it is helpful to think about a spectrum from necessary causation to randomness. Everything that departs from causation to randomness loses its keeping up property and becomes

more spurious. We understand the notion of the community when talking about spurious correlations refers to any relationship more or less apart from necessary causation and closer to randomness.

In this work, we will consider as a spurious correlation any relationship between variables that might change if the SCM is modified, whether by interventions in exogenous variables, environment changes, dataset shift, etc. These concepts will be described in the following sections.

3.3 Interventions and environments

An intervention replaces one or more of the functions f_i defining the variables V_i in an SCM, which includes replacing it with a constant. It models the replacement or modification of the causal relationship between the variables of our problem. A valid intervention does not destroy too much the causal relationship involving Y [ABGLP19]. Suppose we have the SCM presented in the Equation 3.1.

An example of intervention would be replacing X_1 definition with $X_1 \leftarrow 10$, which consists of querying the model about what happens if one can force, determine or choose to set the variable X_1 to 10. Another example would be replacing the X_2 equation by $X_2 \leftarrow 2X_1 + N_2$.

Definition 4. *Consider an SCM whose endogenous random variables are $(X_1, ..., X_d, Y)$, and the goal of learning is a predictive model for Y using $X = (X_1, ..., X_d)$. The set of environments ϵ_{all} is defined by the distributions $P(Y^e, X^e)$ generated by a valid intervention e. An intervention $e \in \epsilon_{all}$ is valid if the causal graph derived from the SCM remains acyclic, the variable of interest Y expectation is the same, that is, $\mathbb{E}[Y^e \mid Pa(Y)] = \mathbb{E}[Y \mid Pa(Y)]$, and the variance $V[Y^e \mid Pa(Y)]$ remains finite.*

Notice that an intervention is not necessarily human-driven or wanted. It covers the case of exogenous variables changing due to the complex interaction of systems. Using this environment definition, we argue that time is a good proxy for identifying them since many data changes happen in sequences over time.

3.4 The principle of Independent Causal Mechanisms (ICM)

The principle of Independent Causal Mechanisms is defined by Scholkopf [SJP+12], as follows.

Definition 5. *The causal generative process of a system's variables is composed of autonomous modules that do not inform or influence each other. In the probabilistic case, this means that the conditional distribution of each variable given its causes (i.e., its mechanism) does not inform or influence the other mechanisms.*

The prominent example to illustrate this principle is the altitude and temperature relationship [PJS17]. Imagine a dataset with the altitude and the average annual temperature from different cities in a country, which provides us the joint probability $p(a, t)$. One can factorize this probability in two different ways.

$$\begin{aligned} p(a,t) &= p(a \mid t)p(t) \\ &= p(t \mid a)p(a) \end{aligned} \tag{3.2}$$

These two factorizations suggest different assumptions about the causal structure. The first suggests the causal graph $T \rightarrow A$, and the second suggests $A \rightarrow T$. How would we know what the right direction is?

Before analyzing the implications of these two possibilities, we could imagine interventions on this problem, even if they are not reasonable or feasible.

Intervention 1: we somehow elevate all the cities in our dataset. What is the effect on the average annual temperature? Considering what we know about physics, it is going to drop.

Intervention 2: we artificially change the cities' temperature by using a giant air conditioner. What is the effect on altitude? We would not expect any change.

This brief imagination exercise reveals that an intervention could help us disambiguate such a situation. We intervene in the hypothesized cause, and we watch for the effect. Though someone can estimate from data both relationships, since nothing holds us from regressing T on A or A on T, one of them is going to be a simple association, while the other reveals a causation mechanism in which one can convert changes in A into changes in T.

What if the intervention is not available and our intuition cannot help with the causal direction disambiguation? Instead of thinking about this data coming from a single country, imagine we have different datasets from two countries, which are seen as the environments following the definition from the previous section. We are going to identify them by a superscript.

$$\begin{aligned} p^{Brazil}(a,t) &= p^{Brazil}(t \mid a)p^{Brazil}(a) \\ p^{Germany}(a,t) &= p^{Germany}(t \mid a)p^{Germany}(a) \end{aligned} \tag{3.3}$$

Brazil and Germany have a different distribution of cities altitude, but should not the relationship between altitude and temperature be equal no matter where we measure it? It follows (in idealized settings), but only when we get the causal direction right. To make it clear, we can rewrite it.

$$\begin{aligned} p^{Brazil}(a,t) &= p(t \mid a)p^{Brazil}(a) \\ p^{Germany}(a,t) &= p(t \mid a)p^{Germany}(a) \end{aligned} \tag{3.4}$$

The invariance $p(t \mid a)$ under different countries would enable Transfer Learning. Since it is unnecessary to learn from Germany's data to predict how to translate its cities' altitude into temperature predictions, we can reuse the same module learned from other countries.

3.4.1 ICM, Causal Discovery and a real example

The problem of identifying the relationships and their directions is known as causal structure discovery, causal discovery, or structure identification, which is the problem needed to be solved when deciding between $T \to A$ and $A \to T$.

With the assumption that the function connecting A and T is an Additive Noise Model (ANM), there is a method based on the independence of residuals [MPJ+16]. The assumption is needed to explore the invariance property we get when the causal model specification is correct. A correct specification means the effects are defined by causes (causal direction) instead of the opposite (anti-causal direction). Assuming an ANM:

$$\begin{aligned} Y &= f(X) + N_Y \\ X &= g(Y) + N_X \end{aligned} \tag{3.5}$$

Where Y and the noise N are independent, and X is independent of N. The method to verify it follows three steps.

1. Fit a function f as a non-linear model of X on Y
2. Compute the residual $N = Y - f(X)$
3. Check whether N and X are statistically independent

The results from Mooji (2016) [MPJ+16], which provides evidence of this method for other real data cases, show a strong dependency in the anti-causal direction. We reproduce the result for the

altitude and temperature case using data from Brazil [Mon21]. In Figure 3.2, the horizontal axis of the two plots contains the X variable, which is the input, the cause. We can see that when the temperature is taken as the cause of altitude (second plot), we observe the dependency. By estimating the Pearson correlation between the residuals and the input, we have 0.077 (p-value, 0.06) for the anti-causal direction and 0.004 (p-value, 0.92) for the causal direction.

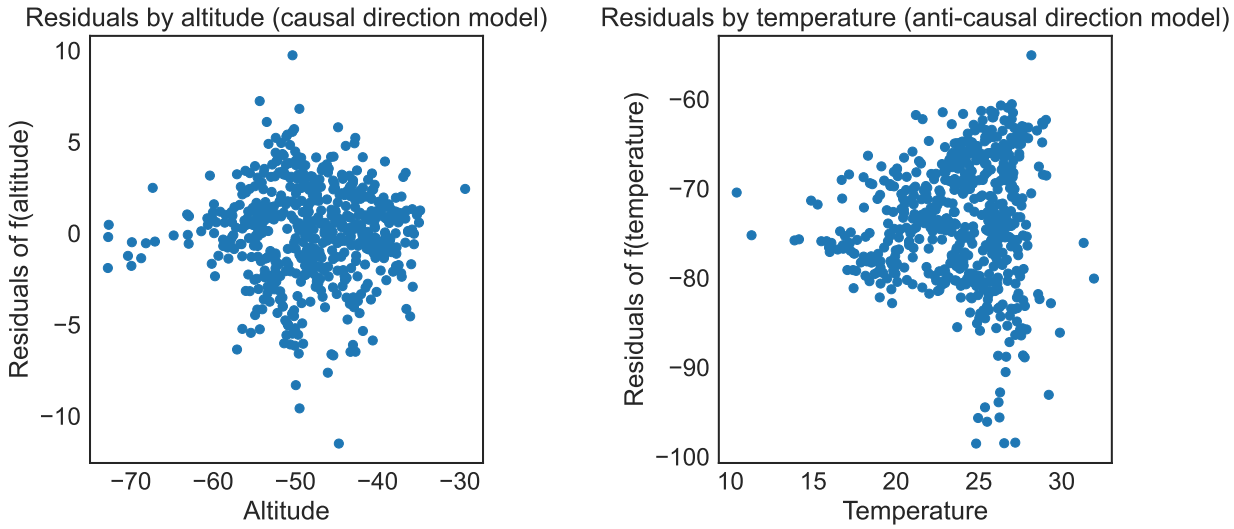


Figure 3.2: *The first plot shows no dependency between residuals and the cause, altitude. The second plot shows a dependency, which indicates that the assumed cause, temperature, is in fact the effect.*

3.5 Causal Forest

The Causal Forest [WA18], later extended to Generalized Random Forest, is a model to estimate heterogeneous treatment effect from observational data, which means the ability to predict how a treatment relates to an outcome. A treatment is any action we want to take to impact the outcome. This task belongs to the Causal Inference field, in which the most important thing is to get right how the outcome changes concerning changes in the treatment instead of making predictions with the lowest error possible.

In general, causal claims are uncovered using Randomized Controlled Trials (RCT) [LF50]. In the binary treatment case, it means we can observe two samples with the same input distribution, except for the treatment. Randomization relies on exchangeability, which means the result for the treatment does not depend on the particular group it is applied to, thus the result would be the same had the other group received the treatment [HR20].

However, the Causal Forest proposes to do the same with non-experimental data. The specific change in the Random Forest design algorithm intended to achieve it was an inspiration for the Time Robust Forest we introduce later. To learn the effect of the treatment, the Causal Forest forces that every leaf should contain at least k examples from the treated and the untreated group. The prediction then becomes the difference of the target variable in the two groups. The fact they are in the same leaf means their covariates are so similar that it gets us closer to an RCT setting.

We can formalize it by setting the observed data as the triple $(X_i Y_i, W_i)$, where X_i is the covariates, Y_i is the target variable, and W_i is the treatment class. We can learn a Causal Forest by the following steps:

1. Draw a random subsample of size s from $1, ..., n$ without replacement, and then divide it into two disjoint sets of size I and J, both of size $s/2$.

2. Grow a tree via recursive partitioning. The splits are chosen using the data from the J sample but without using any information from I. The criterion maximizes the variance between

classes of W while constrained by having at least k examples from every class W.

3. Estimate the leaf-wise responses using only the I sample observations' Y.

Using such an algorithm, we estimate the effect of the treatment on a leaf, in the binary treatment case, as:

$$\hat{\tau}(x) = \frac{1}{|i : W_i = 1, X_i \in Leaf} \sum_{i:W_i=1, X_i \in L(x)} Y_i - \frac{1}{|i : W_i = 0, X_i \in Leaf} \sum_{i:W_i=0, X_i \in L(x)} Y_i \quad (3.6)$$

Equation 3.6 states the estimated average effect $\hat{\tau}(x)$ is the difference between the treated and the untreated examples from the set I that lie in the same leaf as the example x.

3.6 Generalization challenges under a causal view

In Section 2.4, we provided a brief overview of the dataset shift problem, which is characterized by the distribution changing when learning from data. In this section, we discussed how the causal relationship defined in the data generating process is helpful to understand how spuriousness makes the dataset shift even worse. Further, we reviewed the concept of generalization and combined the presented concepts to motivate the following sections.

We can split the issues caused by a concept drift problem in two. First, if there is a causal relationship between Y and X, but $p(Y \mid X)$ changes, we need to learn it again, and it thus becomes a new piece of knowledge.That means that the Y in the data generating process described by the underlying unknown SCM has changed. Second, the concept learned can involve an anti-causal relationship, which is worse since anti-causal direction relationships are more prone to concept drift [KPS18]. Furthermore, in the anti-causal conditional drift case, covariate shifts will change the concept learned from the training data, making the $p(Y \mid X)$ change in every training iteration depending on how the X distribution changes.

In the case of causal concept drift, we end up in a trade-off between including more causal terms prone to real concept drift to have a better performance in the short term or rely on the hardly changing concepts to have a better performance in the long-term, in a scenario they would not be updated.

We have defined generalization in Section 2.2. Still, after presenting the concept of Dataset Shift and environments, it is clear it is insufficient to characterize the degree of generalization we expect from our models. This degree is present in ICMs, which can perform well under distributions from different environments. This notion of generalization is known as Out of Distribution (OOD) or Out of Domain generalization [Wal45] [ABGLP19]:

$$R^{OOD}(f) = max_{e \in \epsilon_{all}} R^e(f) \quad (3.7)$$

In this formulation, we do not expect to deal with a random sample from the same distribution we have learned the model. We accept that the data will come from different environments, and the learned model needs to generalize to all of them to some extent.

Notice that the Data Shift research literature, which characterizes the problem, is related to its solution field, which is called Domain Adaptation [ZSMW13]. In this area, the concern is how to adapt the learning problem when a dataset shift happens. In my experience with Machine Learning models in the financial industry, the nature of most of the problems is so dynamic that any Machine Learning solution should be prepared for Domain Generalization in the first place instead of considering adaptation as a particular case. A changing environment would be closer to being the rule rather than the exception.

Beyond the expectation for i.i.d data, a second hypothesis is commonly made along with ERM to guarantee the generalization bounds, which is about the capacity of the hypothesis class in which

we look for possible functions to learn and the number of examples in the training set. In models with a high capacity, like Neural Networks, the procedure used to train will favor the simplest solutions, and spurious relationships are often simpler to learn than non-spurious ones [ABGLP19] [WRS+17], like associating the snow to recognize a Husky instead of a complete profile of its shape. That means simpler models are not the correct answer if one seeks predictors that remain accurate in the long term.

We hypothesize that different kinds of relationships are included in a single problem: spurious, causal and stationary, causal and dynamic, and the ones that resist during the whole training period are more likely to respect the causal direction and survive in the future, which means a better OOD generalization in problems with environmental changes.

Chapter 4

The search for Domain Generalization

Previous works offer different ways of achieving robustness by exploiting the environment information. Invariant Causal Prediction (ICP) [PBM15] is a feature selection algorithm to find the subset of causal features by testing if the error in the residual on this subset follows a property only found on the target variable's parents under the needed assumptions, which results in selecting only the causal variables as inputs. Invariant Risk Minimization (IRM) [ABGLP19] modifies the objective function to iterate in training environments and penalize the lack of invariance. A penalization term is offered for the linear case. It learns at the same time the data representation for all the training environments and the optimal out of distribution model elicited by this common representation. In addition to these works exploring the environment information, we present Distributionally Robust Supervised Learning (DRSL) [Bag05], which uses adversarial learning to learn a model that minimizes the error in the worst mix possible of the training examples. In this chapter, we study how ICP, IRM, and DRSL work, while we briefly present other models that leverage environment information to achieve domain generalization.

4.1 Stress test, Credibility and Data Augmentation

As domain generalization became a known problem, the validation strategy changed. In order to avoid counting on the i.i.d assumption when validating models, practitioners developed other ways to prove their models can adapt to the real world data [DHM+20], calling them stress-tests and linking the results to how credible the model is. We briefly present a few ideas about identifying generalization issues before studying algorithms that include an inductive bias to avoid these same issues.

One issue is when the performance is different for the subgroups in the data. To identify it, **Stratified Performance Evaluations** are done using a specific feature to identify these groups (e.g., a gender feature) and evaluate the model for them, which comes from the Fairness literature where stratifying the data by different social groups is a valuable step to understand the different ways the model can impact them.

Model robustness is usually defined in terms of the ability to cope with the changes in the data described by the Dataset Shift field. To verify it, **Shift Performance Evaluations** are performed by changing the distribution of the input features. One might desire for a particular application that a specific transformation of one feature does not affect the model performance, which means invariance. This kind of evaluation connects with the notion of environments since one might test the model when capturing pictures using a different device or conditions [HD19], or changing the data acquisition process [BMA+19], which are examples of valid environments. In the end, this validation strategy connects with the search for invariance in different domains, environments, contexts, etc. Beyond validation, known desired invariances could improve the learning process itself via data augmentation, especially in computer vision by transformations such as rotation, translation, pixel flips, etc [SK19].

The **Constrictive Evaluations** are fine-grained versions of the shift evaluations focused on

decision flip and counterfactual. If changing a specific feature should be order-preserving regarding the model's output, one can measure how it disturbed this expectation for the specific feature and change.

4.2 Invariant Causal Prediction (ICP)

Peters [PBM15] explores the invariance in causal relationships through different environments as a way to make predictive models more robust. As explained previously, an environment might be interpreted as a different location, time, policy, etc. For example, in a healthcare problem, the data from different hospitals can be considered as coming from different environments.

Consider the environments as $e \in \varepsilon$ and the linear model case.

$$Y^e = \mu + X^e\gamma^* + U^e \tag{4.1}$$

The superscript e means we have a model for every environment. Where μ is the constant intercept, X^e are the input variables, Y^e is the output, the target variable, and U^e is the error term.

In a supervised problem, we usually have many inputs, which we can also call predictors. Be S^* the set of causal predictors, which are represented by the non-zero parameters of the vector γ^*, it is, $S^* := \{k; \gamma_k^* \neq 0\}$. So S^* contains only the causal predictors, which are followed by non-zero parameters. We are interested in this subset because of all the problems presented previously about using non-causal predictors.

Hypothesis 1. *(invariant prediction): there is a coefficient vector $\gamma^* = (\gamma_1^*, ..., \gamma_p^*)^t$ with supports $S^* := \{k : \gamma^* \neq 0\} \subseteq \{1, ..., p\}$ that satisfies*

$$\begin{aligned} &\textit{for all } e \in \varepsilon : X^e \textit{ there is an arbitrary distribution } e \\ &Y^e = \mu + X^e\gamma^* + U^e, U^e \sim F_U \textit{ and } U^e \perp X^e_{S^*}, \end{aligned} \tag{4.2}$$

where $\mu \in \mathbb{R}$ is an intercept, U^e is a random noise with zero mean, finite variance, and the same distribution F_U for all $e \in \varepsilon$.

Peters (2015) [PBM15] shows that the parent variables of Y in a Structural Equation Model satisfy the Hypothesis 1, which means that if we use only the parent variables in the model, the U^e distribution in every available environment e will have the same distribution and finite variance. At a high level, the ICP algorithm works as follows:

1. For every $S \subseteq \{1, ..., p\}$ and $e \in \varepsilon$:
   (a) Fit a linear regression with the pooled dataset, it is, from all the environments. In the case of linear regression, we get all the estimated parameters from the set S, $\hat{\gamma}^{pred}(S)$. Calculate the residuals $R = Y - X\hat{\gamma}^{pred}(S)$
   (b) Test the null hypothesis that the mean from the R is equal for every environment I_e, $e \in \varepsilon$, using a two-sample t-test for the residuals in I_e and the ones in I_{-e} (all the environments, except for e), and combine the tests to all the environments $e \in \varepsilon$ using the *Bonferroni* correction. In addition to that, test if the R variance is identical for I_e and I_{-e} using an F test, combining it again using via *Bonferroni* for every $e \in \varepsilon$. Finally, combine both p-values, the one from the same mean and the other about the same variance, by using twice the value of the minimum value. If the p-value for the set S is lower than α, reject S.
2. Then, we define the set $\hat{S}(\varepsilon)$:

$$\hat{S}(\varepsilon) := \bigcap_{S: H_{0,S(\varepsilon)} \text{not rejected}} S \tag{4.3}$$

3. If the set S is rejected, we have: $\hat{\Gamma}_S(\varepsilon) = \emptyset$. Otherwise, we define $\hat{\Gamma}_S(\varepsilon)$ as the conventional confidence interval for $(1-\alpha)$ for the parameters $\hat{\gamma}^{pred}(S)$.

The algorithm is independent of the kind of learning algorithm once the tests to define the set S are done using the residuals R, where $\hat{Y}$ can be generated by any other functional form derived from different algorithms, like a random forest, lightgbm, or a neural network.

The environment concept plays a central role. Here we are exploiting the property of causal invariance, so we need to identify the environments we expect to observe this invariance, and it is through an iteration that we can find stable sets of features by checking for the hypothesis.

This method has one assumption and one pitfall that requires attention. First, the assumption that the residuals' variance is the same under different environments might not be valid in a specific or even the general application. Second, there is a pitfall if there are highly correlated features because, in a group of significantly correlated features, any of them can replace each other. Hence they can alternate their presence in the S subsets, and then in step 2, we would not find an intersection.

There is a contribution from Heinze-Deml (2017) [HDPM17] to overcome the second problem by identifying these groups and consider the entire group as a single element for the intersection on step 2.

Furthermore, running the method in different possible features sets will become prohibitive quickly as the number of input features grows.

4.2.1 Synthetic example

If we use only the causal features to predict the target, considering no hidden confounding, we can estimate the parameter from the data generating process, which is interesting since it pretty much reveals to us "how things really relate". In this case, any changes on different concepts, input distribution, including valid interventions, will not affect it, and we end up with a robust model.

The ICP algorithm outputs to us this set of features $\hat{S}$. Then it is a matter of using this set as the input features of a learning algorithm to enjoy the benefits of correctly specifying the causal predictors of the target variable.

$$X_1 \longrightarrow Y \longrightarrow X_2$$

Figure 4.1: *In a very simple generating process, we have X_1 causing Y, which causes X_2. Consider Y the variable of interest and X_1 and X_2 the available input variables.*

Notice the environments play a central role. We need to generate it for at least two different cases. We will consider that the environments will change the variance of X_1. It is going to be generated following the graph in Figure 4.1 and the following SCM.

$$\begin{aligned} X_1 &\leftarrow \mathbb{N}(0, \sigma^2) \\ Y &\leftarrow X_1 + \mathbb{N}(0, 1) \\ X_2 &\leftarrow Y + \mathbb{N}(0, 1) \end{aligned} \tag{4.4}$$

When we run the ICP using the data generated by this SCM[1], it outputs X_1 as the causal variables set, which is correct.

Performance under confounding

We consider no hidden confounding in the previous case, which means no unmeasured variable that impacts both cause and effect. Unmeasured means this variable is unavailable for us. It will

[1]The code for this example and the one from the following section are available at https://github.com/lgmoneda/msc-thesis/tree/main/literature/icp_synthetic_examples.ipynb

not be present in the dataset and will not be part of the algorithm process, there is no expected invariance between X_1 and Y without controlling for the confounder, so it is not expected that the method will work.

We simulate the unmeasured confounder case and the case where we do measure the confounder. We include a confounder C_1 to the previous setting's graph and run the ICP algorithm.

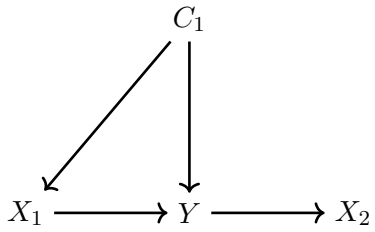


Figure 4.2: *Confounding happens when a variable (C_1) influences both the outcome (Y) and a cause of the outcome (X_1). This is a very common challenging set when learning causal relationships. The ICP works under observed confounders.*

In this case, the ICP outputs X_1, C_1. However, if we do not measure C_1, the algorithm outputs X_1, X_2, which is wrong.

ICP offers an interesting way to explore causal invariance in Machine Learning and helps us build robust models. At the same time, it suffers from the traditional problem of unmeasured confounders. It also has other weaknesses that might discourage its use, like the assumption of equal variance for the error in every environment, the pitfall of highly correlated features, and the cost of searching through every subset.

4.3 Invariant Risk Minimization (IRM)

In the Invariant Risk Minimization framework, the preference for invariance across the environments is expressed in the loss function by an additional term and an iteration on all the environments in the training set [ABGLP19]. Consider the data was collected under different environments $e \in \varepsilon$, it is $D_e := \{(X_i^e, Y_i^e)\}_i^{n_e}$. The training environments are a subset of all possible environments, $\varepsilon_{train} \subset \varepsilon_{all}$. We present again the Equation 3.6 to make the IRM objective clear in terms of minimizing the OOD Risk.

$$R^{OOD}(f) = \max_{e \in \varepsilon_{all}} R^e(f) \tag{4.5}$$

The subscript *all* means we want to minimize it for all possible environments and not only the ones in the training data.

The risk under a certain environment e is defined as $R^e(f) := \mathbb{E}_{X^e, Y^e}[L(f(X^e, Y^e))]$, where L is any loss function.

However, the function f has two stages. The first stage transforms the data from the different environments to find a representation that enables a single optimal classifier for all of them. The second one searches for this optimal classifier.

Definition 6. *A data representation $\Phi : \mathcal{X} \to \mathcal{H}$ elicits an invariant predictor $\omega \circ \Phi$ on all the environments ε if there is a model $\omega : \mathcal{H} \to \mathcal{Y}$ optimal for all environments at the same time, that is, $\omega \in arg\min_{\bar{\omega}:\mathcal{H}\to\mathcal{Y}} R^e(\bar{\omega} \circ \Phi)$ for all $e \in \varepsilon$*

$\mathcal{H}$ is the latent space in which the algorithm searches for a representation that enables the existence of the model ω. Therefore, the IRM paradigm can be described by the following optimization.

$$\min_{\substack{\Phi:\mathcal{X}\to\mathcal{H}\\ \omega:\mathcal{H}\to\mathcal{Y}}} \sum_{e\in\varepsilon_{train}} R^e(\omega\circ\Phi) \\ \text{constrained to } \omega \in arg\min_{\bar{\omega}\mathcal{H}\to\mathcal{Y}} R^e(\bar{\omega}\circ\Phi), \text{ for all } e\in\varepsilon \tag{4.6}$$

However, this optimization is hard to solve since it involves ω and Φ at the same time. To work around this problem, the authors propose the following approximation:

$$\min_{\Phi:\mathcal{X}\to\mathcal{H}} \sum_{e\in\varepsilon_{train}} R^e(\Phi) + \lambda\left\|\nabla_{\omega|\omega=1.0}R^e(\omega\circ\Phi)\right\|^2 \tag{4.7}$$

Where Φ becomes the complete invariant predictor without the need of composing it with ω, which is fixed as 1, turning into a dummy model. The gradient norm penalty in the loss function, the second term in the sum, is used to measure how optimal the model is in every environment e. $\lambda \in [0,\infty]$ is the regularizer term that balances between the predictive power, the one minimizing R following the ERM paradigm, and the invariance of the predictor $1.\Phi(x)$.

Thus the objective function can be expressed as:

$$L_{IRM}(\Phi,\omega) = \sum_{e\in\varepsilon_{tr}} R^e(\omega\circ\Phi) + \lambda\mathbb{D}(\omega,\Phi,e) \tag{4.8}$$

The $\mathbb{D}$ represents the penalization, and we need it to be differentiable with respect to Φ and ω.

In the case of learning an invariant predictor $\omega\circ\Phi$, where ω is a linear regression learned with least squares, we would have the penalty $\mathbb{D}_{lin}$.

$$\mathbb{D}_{lin}(\omega,\Phi,e) = \left\|\mathbb{E}_{X^e}[\Phi(X^e)\Phi(X^e)^T]\omega - \mathbb{E}_{X^e,Y^e}[\Phi(X^e)Y^e]\right\|^2 \tag{4.9}$$

It is a differentiable function that enables us to optimize the objective function using gradient descent.

The conditions about what can be considered an environment in ε_{all} follows the Definition 4. They need to be respected in the training environments to enable OOD, which implies optimal performance in all the possible valid environments.

As seen, an environment can be seen as an intervention, which is when we modify one or more equations from an SCM, forcing particular values or functional forms.

For example, consider the SCM $\mathcal{C} = (\mathcal{S},\mathcal{N})$. An intervention e in $\mathcal{C}$ is the replacement of one or more equations to obtain a new SCM, $\mathcal{C}^e = (\mathcal{S}^e,\mathcal{N}^e)$, represented by:

$$\mathcal{S}_i^e : X_i^e \leftarrow f_i^e(Pa^e(X_i^e), N_i^e) \tag{4.10}$$

We consider the variable X^e was intervened on if $\mathcal{S}_i \neq \mathcal{S}_i^e$, that is, the equations are different, or $N_i \neq N_i^e$, that is, the noise term is different. In the Example A.7, an intervention would be changing X_2^e by $X_2^e \to 0$.

Arjovisky et al. (2019) [ABGLP19] details how the low error on ε_{train} and the invariance on ε_{all} leads to low error on ε_{all}, and how learning these invariances from ε_{train} would be able to identify the same invariances we expect to see on ε_{all}.

In order to learn valuable invariances, it is needed that a degree of diversity exists in the available environments. Two random samples from the same dataset would not be able to offer this diversity. An illustrative application of IRM for the linear model case using synthetic data can be found in Appendix A.

4.4 Distributionally Robust Supervised Learning

Instead of defining environments to achieve OOD generalization, DRSL [Bag05] changes the optimization during the learning process to address the expectation of a change in the distribution between the training samples available and the future unseen data. We present this model because the min-max optimization resembles the Robust Time Tree design.

Considering the training data as $p_{train}(X, Y)$ and the future data as $p_{test}(X, Y)$, we define a family of possible distributions for $p_{test}(X, Y)$ denoted by Q using a constraint about how different it is from the training sample in terms of KL-divergence, $KL(p_{test} \mid p_{train}(X)) < \epsilon$, in Equation 4.11. The learned model is expressed as a set of parameters $\theta \in \Theta$, which represents the hypothesis space $\mathbb{H}$.

$$\mathcal{KL}(f \mid g) = \sum_X f(X) ln \frac{f(X)}{g(X)} \tag{4.11}$$

Considering the objective is to learn the best model θ in the worst p_{test} case, the optimization problem becomes the following min-max:

$$min_{\theta \in \Theta} max_{p_{test} \in Q} \mathbb{E}_{p_{test}}[g(X, \theta)] \tag{4.12}$$

Where $g(X, \theta)$ is the risk function:

$$g(X, \theta) = \mathbb{E}_{p(Y|X)}[L(Y, h_\theta(X))] \tag{4.13}$$

In order to learn it with the available information, it is, the p_{train}, an empirical risk function is defined to use the training sample. The weights applied in the training data to create the test set are represented by q, the β parameter is chosen to match the $\mathcal{KL}$ boundary ϵ. The final algorithm proposed [Bag05] can be seen in the Algorithm 2.

Algorithm 2 Robust Supervised Learning Algorithm

1: **procedure** RobustSupervisedLearning(X, Y, β)
2: Initialize weights $q = [\frac{1}{N}, ...]$
3: **while** θ has not converged **do**
4: $\theta \leftarrow$ Learner(X, Y, q)
5: $\hat{q} \leftarrow max_{q \in Q} \frac{exp(\beta g(x_i, y_i, \theta))}{\sum_i exp(\beta g(x_i, y_i, \theta))}$
6: $q \leftarrow \alpha q + (1 - \alpha)\hat{q}$
7: **end while**
8: **end procedure**

4.5 Other models exploring the ICM principle

There are other approaches designed to take advantage of the ICM principle. In the Recurrent Independent Mechanisms (RIM) network [GLH+19], attention [VSP+17] is used to activate different modules composed by RIMs. These modules learn different aspects of the problem, and it is expected that the invariant aspects will be useful when it needs to predict data that differ from the training distribution. Neural Causal Models [KBG+19] leverage known or unknown interventions in the observational data to learn. The weakly supervised disentanglement approach [LPR+20] learns how to disentangle components from high-dimensional feature spaces, like images, using the hypothesis that they are composed by a small number of relevant factors that can change, thus exploring the modularity of ICMs. The do-calculus in the presence of interventional data [BP14] is also used to tackle the problem of learning from available data from a few environments and generalizing to unseen environments. In Robust Supervised-Learning [Bag05], the environment concept is not present, but the learning algorithm includes weighting training data to prepare the model for the

worst mixture possible of it, hoping it is also the worst case for the future and unseen examples. Not related to ICM principle, but considering time, the environment we explore in this work, the Temporal Decision Trees [KH01, KH10] use time in order to build the model rules, but it is about sequential data, they are not independent of each other. In this case, the relationship between examples is that they are part of a chain of events instead of being independently sampled under the same generating process in a particular environment.

Chapter 5

Time Robust Trees: Using Invariance to Improve Generalization in Prediction Tasks

In this chapter, we present the Time Robust Tree (TRT), in which the ensemble creates the Time Robust Forest (TRF). We start with a simple example to motivate slicing the data into different environments made of periods of time. The algorithm is presented before showing a few synthetic examples that uncovers how to set its parameters and one of its limitations.

5.1 A recursive partitioning algorithm for learning time-invariant predictions

Ideally, we want to learn concepts or models that we can transport to the future with minimal effort. As discussed in the previous chapters, one way of achieving such a goal is to ensure invariance to different environments, which are originated by valid interventions in a causal model.

Our proposed algorithm considers the time order of data instances to segment them as valid environments. Typically, recursive partitioning schemes such as CART seek generalization ability by setting a lower bound on the number of training instances available to grow a tree further or by some post-processing technique. Instead of constraining a data split to have a minimum number of training examples, we constrain the minimum number of examples on each period, thus forcing the model to learn time-balanced splits. The algorithm does not assume any information regarding environments other than the instance's time period. Additionally, we can express the desire for time stability by taking the average score by period or the worst case.

In terms of dataset shift, it involves learning concepts that do not drift with the hypothesis that they are the most successful when transferred to other environments. In our case, where environments are periods, it means the future. It expresses a preference for non-spurious variables, which makes the TRT select its rules using causal predictors.

5.2 A Motivating Example

Before formally describing the proposed algorithm, we first motivate the necessity of time-robust learning methods and explain the limitations of current approaches with a toy example. Thus assume two finite-valued input variables X_1 and X_2, a binary target variable Y, and a time period T_{period}, used to segment the data in different diverse environments. We use three time periods to illustrate, thus $T_{period} = \{1, 2, 3\}$. Consider the data illustrated in Figure 5.1. Imagine the data from $t = 1, 2$ consists in the available training set, while the $t = 3$ comes from the future after deployment. We will call it the holdout set. According to the example, X_1 is mildly predictive and stable, while X_2

		DT	TRT		
Variable	Split value	GI	GI at t=1	GI at t=2	Max. GI value
X_1	3	0.49	0.50	0.40	0.50
X_1	4	0.44	0.44	0.44	**0.44**
X_1	5	0.49	0.50	0.40	0.50
X_2	1	**0.27**	0.00	0.50	0.50

Table 5.1: *Split evaluation process for the Decision Tree and the Time Robust Tree for the motivating example*

is a perfect predictor at $t = 1$, but its relationship with the target changes in the following periods, representing a spurious correlation, or a non-static causal relation that shifted.

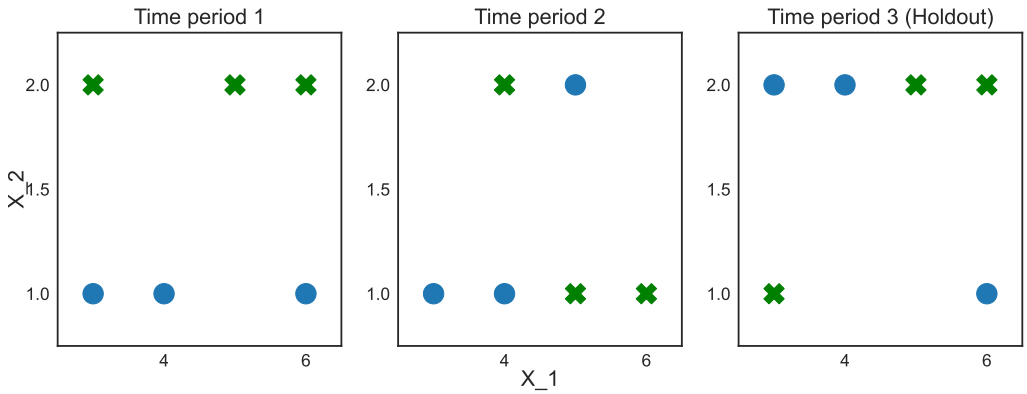


Figure 5.1: *In this simple motivating example, in the time period 1, there is an exact split on X_2 to separate classes, which does not stand for the time period 2 and the period 3, the holdout. A less exact split on X_1 is present in every time period, which the Time Robust Tree prefers.*

If the modeler uses all the available training data, a typical Decision Tree (DT) inducing algorithm will combine the data from periods 1 and 2 into a single training data set to evaluate the possible splits. In contrast, in the Time Robust Tree (TRT), as long as the modeler sets the period information as the environment, we consider the split performance separately when looking at every period. To illustrate it, we prune the example tree to have a single split in both cases. We use the Gini impurity (GI) minimization process in Table 5.1.

We use the Area Under the Curve (AUC) to evaluate predictions' quality. It goes from 0 to 1, and the higher, the better. By learning these splits, the Decision Tree would achieve a 0.83 AUC on training, but a poor result on holdout, 0.50 AUC, while the Time Robust Tree is worse in training, 0.67 AUC, but it keeps the same performance in the holdout.

In this motivating example, the time period-wise split choice illustrated resembles the Robust Supervised Learning presented in Section 4.4. The first difference we note is that by using the worst case from a training environment, we avoid being too pessimistic about the distributions we can encounter after the training stage, which happens when we let an adversary function look for the worst-case possible given a boundary by learning it as a mixture of the training examples [HNSS18]. However, this advantage depends on the quality of the environments in the terms discussed in Section 4.3. We make further considerations about the environment split in Section 7.3. The second important difference is that another design choice on TRT indirectly regularizes the split choice, which is the requirement of keeping a minimum number of examples in every period in every node. This second trait effect is illustrated in the following sections.

5.3 Time Robust Trees

A Time Robust Tree is obtained by modifying the standard Decision Tree induction algorithm, using the standard deviation reduction criteria for regression problems and impurity minimization following the Gini impurity for classification problems. These impurity functions are identified as L.

Consider a timestamp column T_{stamp} representing the data capture time with the same dimension of the random variables vectors $(X_1, ..., X_d, Y)$, where the X variables represent inputs and Y the variable of interest, the target. The time period T_{period} is an aggregation of sequential examples when ordered by T_{stamp} using a human-centered concept, like hourly, daily, weekly, monthly, yearly, or simply putting together a fixed number of examples and reducing T_{stamp} granularity.

Given n time periods $T_{period} = t_1, t_2, \ldots, t_n$ in the training set, we find the best split s^* to divide the examples in X_{node} into X_{left} and X_{right} using the rule $X_f \leq v_f$ where f is a feature from all available features F at a certain value v_f from all possible values for the feature f in the training set V_f by applying recursively to every node data X_{node} until the constraints are not satisfied, being the first node the root containing all the training set. Consider the sample size from every period as N_t, the chosen minimum number of examples by period to split as ρ, the maximum depth as d, the minimum impurity decrease as g, and the impurity decrease after the split as $ID(X_{node})$, which is defined in Equation 5.4.

$$\begin{aligned} s^* = & \min_{\forall f \in F, \forall v \in V_f} \max_{t \in T_{period}} L(X_{node}), \\ & \text{subject to } |X_{right,t}| \geq \rho, |X_{left,t}| \geq \rho \text{ and } ID(X_{node}) \geq g, \forall t \in T_{period} \end{aligned} \tag{5.1}$$

The ρ is a scalar representing the minimum number of examples in every time period to perform a split. The model also accepts the average loss criteria.

$$\begin{aligned} s^* = & \min_{\forall f \in F, \forall v \in V_f} \frac{1}{|T_{period}|} \sum_{t=1}^{T_{period}} L(X_t), \\ & \text{subject to } |X_{right,t}| > \rho \text{ and } |X_{left,t}| > \rho, \forall t \in T_{period} \end{aligned} \tag{5.2}$$

For the predictions $\hat{Y}$, the average from the leaf is taken without any consideration about the time period it belongs, $\hat{Y} = \frac{1}{|Y|} \sum y_i$.

It is worth isolating in the Equation 5.3 one of the differences from TRT, the period-wise score, which is taking into consideration how the model performs in the different periods defined by the user to decide the optimal split. Notice the other difference, the hyper-parameter ρ, interacts a lot with this part of the process, since higher ρ guarantees a higher sample in each period for their evaluation regarding the split.

$$\frac{1}{|T_{period}|} \sum_{t=1}^{T_{period}} L(X_t) \tag{5.3}$$

Similarly to the period-wise score, we have the impurity decrease calculation iterating over the periods. The $L_{befor\_split}(X_t)$ gives the impurity measurement in the node before splitting it with a rule, while $L(X_t)$ gives it after the split. Their contrast informs the decrease, which is weighted by the number of examples from a particular period in that node and the total available in the sample.

$$ID(X_{node}) = \frac{1}{|T_{period}|} \sum_{t=1}^{T_{period}} \frac{|X_t|}{N_t} * (L_{before\_split}(X_t) - L(X_t)) \tag{5.4}$$

The process is summarized in Algorithm 3. It starts with a call to LearnTimeRobustTree

f	v_f	X_{left}	X_{right}	score	current score	periods scores
X_1	3	$t_1 = \{(3,1,0),(3,2,1)\}$, $t_2 = \{(3,1,0)\}$	$t_1 = \{(4,1,0),(5,2,1),(6,1,0),(6,2,1)\}$, $t_2 = \{(4,1,0),(4,2,1),(5,1,1),(5,2,0),(6,1,1)\}$	0.50	0.50	[0.50, 0.40]
X_1	4	$t_1 = \{(3,1,0),(3,2,1),(4,1,0)\}$, $t_2 = \{(3,1,0),(4,1,0),(4,2,1)\}$	$t_1 = \{(5,2,1),(6,2,1)\}$, $t_2 = \{(5,1,1),(5,2,0),(6,1,1)\}$	0.44	**0.44**	[0.44, 0.44]
X_1	5	$t_1 = \{(3,1,0),(3,2,1),(4,1,0),(5,2,1)\}$, $t_2 = \{(3,1,0),(4,1,0),(4,2,1),(5,1,1),(5,2,0)\}$	$t_1 = \{(6,2,1)\}$, $t_2 = \{(6,1,1)\}$	0.50	0.44	[0.50, 0.40]
X_2	1	$t_1 = \{(3,1,0),(4,1,0),(6,1,0)\}$, $t_2 = \{(3,1,0),(4,1,0),(5,1,1),(6,1,1)\}$	$t_1 = \{(3,2,1),(5,2,1),(6,2,1)\}$, $t_2 = \{(4,2,1),(5,2,0)\}$	0.50	0.44	[0.0, 0.50]

Table 5.2: *The TRT simulation using the motivational example shows how the algorithm treat differently the time periods data to find splits that are good for all of them*

with all the features in X. Before learning a rule to split the data, there is a condition to stop learning on the maximum depth and the minimum number of examples by period. In CreateSplit, the algorithm learns a split that generates two subsets of the original data, X_{left} and X_{right}, for which we call the learning function again and keep splitting until the stop conditions are met. The search for the particular split happens on FindBestSplit, where we discard any split that does not keep the minimum number of examples in every period after applying it and evaluate the best as the one with the lowest score calculated by PeriodWiseScore, which represents the implementation of the Equation 5.2. In case we have opted to use the worst-case score, the PeriodWiseScore would store the score calculated in every period and return the worst case to represent the split quality. Similarly, we calculate the impurity decrease using the PeriodWiseImpurityDecrease, which can use the average value for every period or the worst case.

There is nothing particularly different in the step from Time Robust Tree to Time Robust Forest in comparison to the one from a Decision Tree to a Random Forest [Bre01]. Considering M trees, the final prediction $\hat{Y}$ becomes $\frac{1}{M}\sum_{m=1}^{M}\hat{Y}_m$, a random proportion of the input features F is considered when finding the best split for a node on Equation 5.1, and bootstrapping is performed in the training data before learning every tree.

5.3.1 Simulating the algorithm

In this section, we simulate the algorithm call in the motivating example. It starts with a call to LearnTimeRobustTree, with $\rho = 5$, $d = 1$ and $g = 0$. The conditional will test if d is equal or greater than 1, which it is, and if there is at least 1 example from periods 1 and 2. Then the CraeteSplit is called with the same parameters. The FindBestSplit is called inside it, which we show the simulation in Table 5.2. We add a column exposing the PeriodWiseScore values. The elements in X_{left} and X_{right} are triples with the values of (x_1, x_2, y) for a particular data point, we show them by period to make it easier to follow.

Since we have determined the maximum depth to be 1, after finding the first optimal split, we would call LearnTimeRobustTree with $d = 0$, which would not pass the conditional to call CreateSplit again. In Figure 5.2, we highlight the decision boundary resulting from this process. The green dotted line represents the TRT rule, while the red one represents the DT.

5.3.2 Going deeper in the motivational example

The idea of the motivational example is to come up with a simple situation we believe illustrates how data behave in reality and show how the algorithm would deal with it. Nevertheless, its simplicity raises multiple ways in which the naive approach can compromise the argument. In this subsection, we briefly raise some of these issues as a prelude to what will be discussed in the experimental section using real data.

Algorithm 3 Time Robust Tree Induction Algorithm

procedure LEARNTIMEROBUSTTREE(X, T_{period}, ρ, d, g)
  if $d \geq 1$ and there are ρ examples for every distinct period in T_{period} **then**
    CREATESPLIT(X, T_{period}, ρ, d)
  end if
end procedure

procedure CREATESPLIT(X, T_{period}, ρ, d)
  X_{left}, X_{right} = FINDBESTSPLIT(X, T_{period}, ρ)
  LEARNTIMEROBUSTTREE(X_{left}, T_{period}, ρ, $d-1$, g)
  LEARNTIMEROBUSTTREE(X_{right}, T_{period}, ρ, $d-1$, g)
end procedure

procedure FINDBESTSPLIT(X, T_{period}, ρ) (Equation 5.1)
  score = -inf
  for Every variable f in X **do**
    for Every value v_f of f **do**
      X_{left} = examples where $X_f \leq v_f$
      X_{right} = examples where $X_f > v_f$
      if Number of examples by time period for X_{left} and X_{right} is greater than ρ **then**
        current_score = PERIODWISESCORE($X_{left}, X_{right}, T_{period}$)
        impurity_decrease = PERIODWISEIMPURITYDECREASE($X_{left}, X_{right}, T_{period}$)
        if current_score < score and impurity_decrease > g **then**
          score = current_score
          $f^* = f$
          $v_f^* = v_f$
        end if
      end if
    end for
  end for
  if score $\neq$ -inf **then**
    X_{left} = examples where $X_{f^*} \leq v_f^*$
    X_{right} = examples where $X_{f^*} > v_f^*$
  else
    $X_{left} = \{\}$, $X_{right} = \{\}$
  end if
  return X_{left}, X_{right}
end procedure

procedure PERIODWISESCORE(X_{left}, X_{right}, T_{period})
  current_score = 0
  for Every distinct period t in T_{period} **do**
    current_score += SCORE($X_{left,t}$, $X_{right,t}$)
  end for
  return current_score / $|T_{period}|$
end procedure
procedure PERIODWISEIMPURITYDECREASE(X_{left}, X_{right}, T_{period})
  impurity_decrease = 0
  for Every distinct period t in T_{period} **do**
    score = SCORE($X_{left,t}$, $X_{right,t}$)
    total_count = $|X_{left,t}|$ + $|X_{right,t}|$
    previous_score = SCORE(APPEND($X_{left,t}$, $X_{right,t}$))
    impurity_decrease += (total_count / N_t) * (previous_score - score)
  end for
  return impurity_decrease / $|T_{period}|$
end procedure

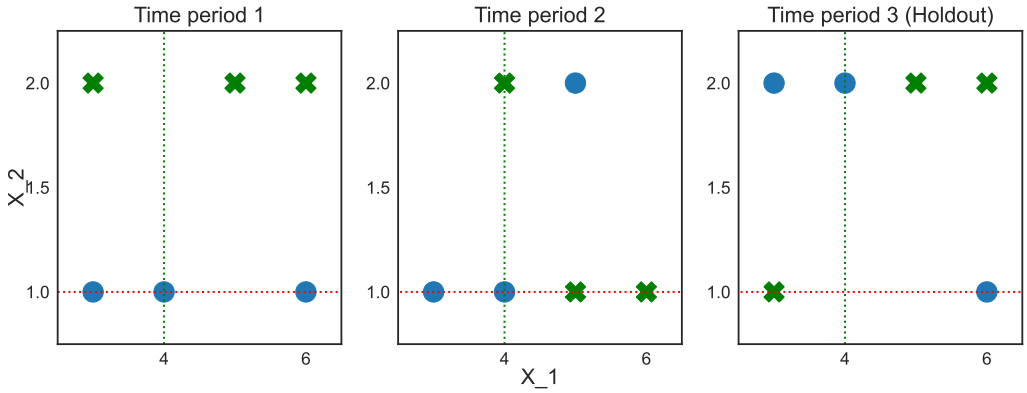


Figure 5.2: *After splitting the two classes using $X_1 \leq 4$ in the motivational example, the TRT decision boundary for a single split can be represented by the green dotted line, while the DT one is represented by the red dotted line.*

We considered a single split, which is illustrative but not realistic. We modify it to let both trees grow until the depth of 20 by setting it as their maximum depth. The number of examples was multiplied by a factor of 20 to enable more splits. The training was repeated setting different values of the minimum samples needed in the leaf for both models. To make it fair, since we have two periods, this value is divided by 2 to set the minimum examples in each period in the TRT. The Figure 5.3 can be interpreted as both models going from complex rules to simpler ones as the number of minimum examples needed increases. We can see the TRT also suffered at the beginning before having the training and holdout performance converging. The DT did the same, but later in the curve. This result creates the hypothesis that the time-slicing only forces the model to be simpler and that the benchmark would eventually perform as well as if the right hyper-parameters were set. We will investigate this hypothesis in the real examples in Chapter 6 when we optimize the benchmark to create a more fair comparison.

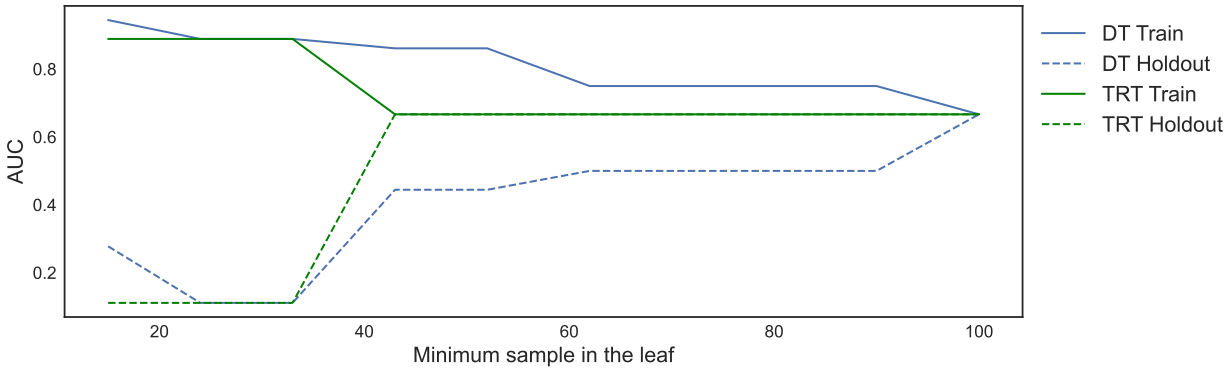


Figure 5.3: *Going beyond the first split in the motivational example for the DT and TRT does not compromise the illustration of the TRT algorithm since the DT cannot find a better rule even without a constraint to grow deeper.*

Further, we can consider that practitioners will usually discard old data, either by intuition or because they have built a reverse learning curve, a process in which we add examples to a model from the most recent to the oldest and see how useful the added data is to predict an out of time set. If we only use data from $t = 2$ to train the model, the DT and the TRT have the same holdout performance, a 0.67 AUC. Though it helps the DT, the hypothesis behind the TRT design is that old data can be helpful for two reasons. First, it might carry disappeared relationships, which we cannot use to predict future data, revealing spurious correlations. Second, it also reveals the ones that existed for a long time, which the persistent relationship suggests they are the most likely to keep existing and provide a better generalization in future data. The same old data that perturbs

DT should fuel TRT robustness. Later in this work, we verify using real cases if the benefit of discarding old data can overcome the benefit of leveraging it to find stable relationships. Notice that if we swap the data from $t = 1$ and $t = 2$, DT would again absorb the spurious relation and fail to generalize.

5.3.3 Synthetic example

After a simple and illustrative example, we move to a synthetic data case to show a more realistic setting, including the hyper-parameter definition. Once again, we include in the data generating process a spurious feature, which is, a variable that suffers a concept drift that makes it non-stable in the training set, X_2. The example is extreme, since X_2 mimics Y in $t = 1$, while it is random in $t = 2$, both of them available for training. The X_2 keeps random in the following periods, which consist the holdout set. It emulates the hypothesis that unstable properties are less likely to persist.

$$\begin{aligned} X_1 &\sim \mathbb{N}(0, 1) \\ Y &\sim X_1 + \mathbb{N}(0, 1) \\ X_2 &\sim f(e) \end{aligned} \tag{5.5}$$

where e is the time period variable, which is our environment. In the training, we have two training environments $\epsilon_{train} = \{1, 2\}$. The $f(e)$ defines X_2 following:

$$f(e) = \begin{cases} Y, \text{ if } e = 1 \\ \mathbb{N}(0, 1), \text{ if } e \neq 1 \end{cases} \tag{5.6}$$

We make it a binary classification task by converting y to a positive class when greater than 0.5, and to the negative one otherwise. The holdout is composed of the following periods, starting at $t = 3$.

At first, we apply the TRT and the DT using similar hyper-parameters: 30 as maximum depth, 0.01 as minimum impurity decrease, 10 as a minimum sample by period for the TRT, and 20 as a minimum sample to split for the DT since we have two periods. In this case, the TRT presents an AUC of 0.83 in train and 0.81 in the holdout, while the DT performs around 0.92 AUC in training and 0.64 in the holdout. It shows how the TRT avoids learning from the spurious variable X_2, which lowers its training performance but makes it succeed in the holdout, while the DT goes in the opposite direction. However, in a real-world case, we need to define the hyper-parameters following a process and objective criteria. In the following subsection, we show how to execute this step when using the TRT.

5.3.4 Hyper-parameter Optimization

In the Section 2.3, we presented the K-fold validation design. We will apply it to the example as the benchmark. However, during the hyper-parameter selection, this design pools together the data from the periods and then select the set of parameters in which the performance is the highest. This process does not favor the period-wise design from TRT. To overcome it, we use a K-fold that generates folds containing just one environment, used as test folds. We identify this approach as Environment K-Folds (Env K-Folds). Similar to what we use to learn the best split in the TRT, Besides taking the average performance in the folds to decide the hyper-parameters, we evaluate a second strategy when using the Env K-Folds. First, we average the performance in all folds consisting of the same environment and hyper-parameter set, then we group by only hyper-parameters sets and select the minimum performance, which is the worst environment case. Finally, among the worst cases, we take the set with the highest performance to determine the model using the best worst case. We identify this approach as Env K-folds Min-Max.

To evaluate these different designs, we bootstrap the data and repeat the process ten times. The results are the average of these ten best models following each approach. As seen in Figure 5.4,

the TRT performs significantly better than the DT in the holdout set when using the Env K-folds Min-Max, while in the other two strategies, they are very similar.

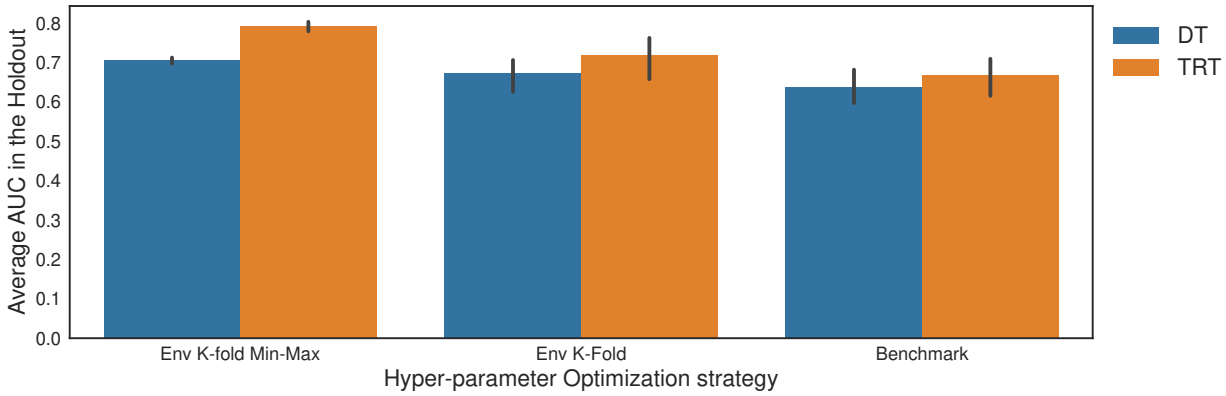


Figure 5.4: *A hyper-parameter optimization design that keeps the period-wise evaluation from the TRT algorithm is important to make the model keep its purpose of learning stable relationships.*

5.3.5 Performance under no shift

If we modify the Function 5.6 defining X_2 and keep the same function from $t = 1$ to all training and future periods, we exclude the concept drift from the problem. Under this case, the TRT and the DT perform similarly. Both get an 0.87 AUC score in train and holdout. If optimized, they achieve a performance of around 0.90 AUC. However, the TRT requires more steps due to its period-wise learning.

5.3.6 A limitation about the data generating process

In this second synthetic data example, we modify the first case to show the input empirical support limitation from Time Robust Forest. Consider the following SCM, which is based on the motivating example for IRM [ABGLP19]:

$$\begin{aligned} X_1 &\sim \mathbb{N}(0, g(e)) \\ Y &\sim X_1 + \mathbb{N}(0, g(e)) \\ X_2 &\sim f(e) \end{aligned} \tag{5.7}$$

as previously, e is the time period variable, $\epsilon_{train} = \{1, 2\}$. The g function defines the variance for every time period, $g(e) : [1, 2] \to [0.1, 1]$, while the $f(e)$ defines X_2 following:

$$f(e) = \begin{cases} Y + \mathbb{N}(0, e), \text{ if } e = 1 \\ \mathbb{N}(0, e), \text{ if } e = 2 \end{cases} \tag{5.8}$$

If we use Time Robust Tree on this data, the results are going to be inconsistent because the empirical supports for the two training environments do not overlap, as seen in Figure 5.5. We can overcome this problem by using a scaler on every input. This scaler uses the minimum and maximum value of the distribution to redistribute it between 0 and 1. However, this process is impossible to carry out in a real scenario since we would never know which scaler to apply on unseen future data once we expect future data to bring unseen environments on which it would not make sense to use the learned scales from the training environments.

The exercises with synthetic data suggest two investigation paths. First, the fact that we needed to perform the scaling in the training and holdout data could mean that the TRT success depends on how much the empirical support for the different variables overlap in the environments available

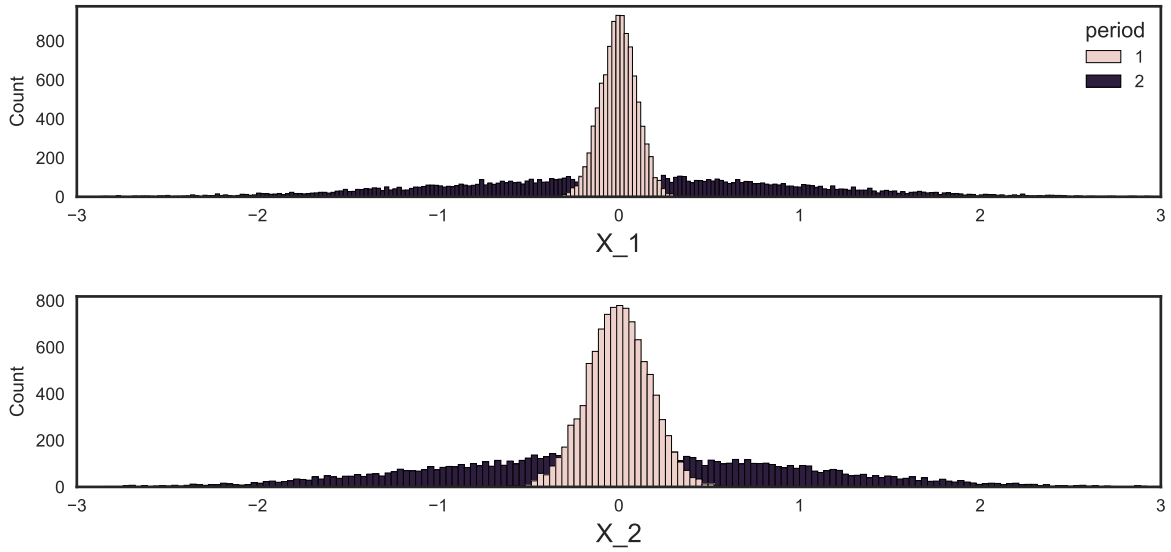


Figure 5.5: *Data from different environments can have different support for the input features. It causes problems for the TRT since it needs to find splits that are good for all the periods, which is impossible in the non-overlapping areas of the inputs' distributions.*

for training and future unseen data. Second, the similar behavior the DT has shown in terms of discarding the spurious correlation raises questions on if it will eventually perform as well as the TRT under this scenario, or if the better performance seen in TRT means it does not sacrifice as much complexity as the DT in order to prefer stable features. Third, if the TRT can focus on features whose relationship with the target is stable, it could be used as a feature selection algorithm. All these paths will be explored in Chapter 6.

5.4 Software package

The presented examples and the experiments we will present in the next chapter use the open-source package[1] developed after the algorithm was designed. It can be installed using *pip* the package installer for Python by running *pip install time-robust-forest*. The model follows the scikit-learn interface [PVG+11].

[1] https://github.com/lgmoneda/time-robust-forest

Chapter 6

Experiments

In the experiments chapter, we test a couple of real-world datasets to inspect the results towards evidence that the TRF has interesting properties when dealing with future data. In our context, using a different set of features, building different rules, resisting the test of time, or simply better performance in an out of time holdout would be considered exciting results.

First, we describe the datasets used and how we have turned them into a predictive task. Then we explain the experimental setting. The results are presented regarding holdout performance before we take a deeper look into the drop in performance between test and holdout, how the future data relates to the training sample, feature importance, impact of old data, and other specific analysis to understand the particular outcomes of the experiments.

6.1 Real-world data benchmark

Seven public datasets[1] with available timestamp information and a reasonable time range were selected to validate the approach. Information about the features, target, and data volume can be found in Table 6.1.

The GE News dataset [Mon20] contains six years of soccer news articles. The data contains articles about 21 different clubs. We make one of them the positive case and transform it into a binary classification. Some pre-processing is done to exclude mentions to the teams' names and different ways they are popularly called in order to not to trivialize the problem. A maximum vocabulary of 300 words is defined using the top words in terms of frequency from the entire corpus.

The 20 newsgroups [M+97] is a standard benchmark dataset. It contains 18000 newsgroups' posts about 20 topics. It is split into training and testing considering the time order, though it does not contain a timestamp column. Twenty different targets are used for all the topics to make it a binary classification to evaluate the Time Robust Tree. Since there is no timestamp column, we group the messages into nine periods in the training set containing 1300 messages each, except for the last, with 914 instances. We have five time periods in the test with 1300 instances, and the last one contains 1032. Since the dataset is small, the experiment was done multiple times using bootstrapping, and a confidence interval is included in Figure 6.1.

Kickstarter is a website for crowdfunding, which is when its future customers fund a project. In the Kickstarter dataset [Mou18], we have information about the country, currency, category, main category, goal, and campaign state. The campaign state is used to create a binary target, in which the positive class is a successful campaign, while the negative class holds the failed, canceled, suspended, and live states.

The Animal Shelter dataset [Dao21] contains data about animals intakes and outtakes in a shelter. We have built a target to predict if a specific animal will be adopted after the first 30 days in the shelter, considering their intake characteristics, like conditions, type, sex, breed, and color.

[1] The source code and datasets are available on GitHub at https://github.com/lgmoneda/msc-thesis

In the San Francisco Building Permits [Sha18] we have data about permit requests, and the task is to predict if it is going to be approved. The Olist e-commerce data [Sio19] contains online purchases and reviews, so we create a target to predict if a customer review is going to be five stars or not. The Chicago Crime dataset [oC21] lists many types of infractions committed in the city, and we use it for the task of predicting the binary target of arresting the criminal.

The GE News and the 20 newsgroups are slightly different from the other datasets since they are textual, and it is needed to convert words to features, which means having hundreds of features. Because of the difficulties of dealing with it, they are omitted from some of the further analyses done in this chapter.

Dataset	Categorical Features	Numerical Features	Target's positive class	Number of Examples
Kickstarter	country, currency, category, main_category	goal	Successful crowdfunding campaign	378k
Olist	seller_id, product_category_name	price, freight_value, product_name_length, product_description_length, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_width_cm, difference_estimated_delivered, difference_limit_delivered	A review score of 5 stars	114k
GE News	300 words	-	The article is about a certain club (Flamengo)	139k
20 News groups	100 words	-	The section the article belongs (one vs all)	18k
Animal shelter	Intake Type, Intake Condition, Animal Type, Sex upon Intake, Breed, Color	Age Upon Intake	Adopted animal after 30 days in the shelter	124k
Chicago crimes	Primary Type, Description, Location Description, FBI Code, Zone, Address, Domestic	Latitude, Longitude, Beat, District, Ward, Community Area	Arrested criminal	397k
Building permits	Street Suffix, Proposed Use, Unit Suffix, Fire Only Permit, Existing Use, Neighborhoods - Analysis Boundaries	Number of Existing Stories, Number of Proposed Stories, Supervisor District, Plansets, Estimated Cost, Existing Units, Proposed Units, Proposed Construction Type, Existing Construction	Permit conceded for a build modification	198k

Table 6.1: *The Table displays the available numeric and categorical features in every dataset tested in the chapter. The categorical features were transformed into unique numeric values, since tree-based models can deal with non-ordinal transformations.*

6.1.1 Data split and benchmarks

A dataset is split into three parts to perform the experiment: train, test, and holdout. First, a specific date from the timestamp column is chosen to split data between in time (past) and out of time (future), then the in time data is randomly split into train and test, while we call the out of time set the holdout. All plots contain test and holdout performance, so the former evaluates how good the model performs in the in time and out of train data, while the latter is about the ability to generalize to the out of time and out of train examples, which is an estimate of the out-of-distribution performance.

Two learning algorithms were chosen as benchmarks. First, the Random Forest (RF), since it is the most similar to the TRF, and it provides a fair comparison since we are not exploring boosting for the TRF. We use the RF implementation from the Scikit-learn library [PVG+11]. Second, for the GE News only, the LightGBM [KMF+17] which is state of the art for tree-based models. For both of them, we apply the pyCaret library [Ali20] to tune their hyper-parameters using a 5-fold in the train set. Notice this setting favors the benchmark since the implementations from these

packages have other strategies to improve generalization beyond the maximum depth, the number of examples to split, and the minimum impurity decrease to split, like the minimum child weight.

The TRF parameters are defined following the number of estimators and maximum depth from the optimized RF. For the minimum examples by period, we use a similar value to the minimum examples for a leaf from the RF. We do not optimize it following a grid search because the current implementation is too slow for multiple fits. However, we test it with different values of minimum examples by period using the same number of estimators and maximum depth to show how the performance would change. The metric used to compare them is the AUC since all the tasks were turned into classification ones.

6.1.2 Performance on unseen future examples

The performance in terms of AUC for the three sets can be seen in Table 6.2. The most interesting part is not the aggregated metric, but how it changes from the in time period (test) to the out of time (holdout), which can be seen in the column Δ *TRF-RF* by verifying how the difference between challenger and benchmark drops from test to holdout, and in Figure 6.1 on how the two performance curves get closer or invert after the vertical dashed line that marks the holdout period start.

When we look solely at the aggregated AUC, the TRF performs better in the GE News and Kickstarter tasks, while it has similar performance in the 20 Newsgroups and Chicago Crime, and it is inferior in the other three. However, for all cases, the difference in AUC between TRF and RF in the holdout is lower than the difference in the test. This result provides evidence that the TRF deals better with out-of-distribution examples.

In Table 6.2 it is possible to verify that the cases where TRF becomes an interesting challenger are the ones in which the benchmark has problems to perform in the holdout as well as it does in the test. A domain classifier was trained using the holdout as the target to provide further evidence about the TRF advantage under scenarios where the future data changes the most. The higher the AUC for the domain classifier, the more significant the difference between test and holdout. The results in Table 6.3 show the TRF performed better in the datasets with a more remarkable shift between training data and holdout data, the ones with a higher AUC for the domain classifier.

Additionally, we inspect if the difference between the RF and TRF performance in the in time test could guide if the TRF is a good choice for the task at hand. We regress the delta AUC in the out of time holdout, our quantity of interest, by the delta AUC in the in time test, which is a piece of information we could calculate at modeling time. In Figure 6.3, we see there is no relationship between them in the experimental datasets used. There is no evidence that the delta AUC in time could help in deciding about using the TRF.

6.1.3 The effect of older data

When discussing the motivational example from Chapter 5, we cited that practitioners can discard old data from the learning process if it is not benefiting the model to predict the future. It happens because the most recent data is usually the most similar to the future unseen cases. We raised the hypothesis that while older data could be harmful to learning algorithms in general, the TRF should take advantage of it. To discuss it, we have created reverse learning curves. We fixed the same holdout we have been using, then trained multiple models using different data slices. The first slice is the most recent data. We follow the time segmentation used to feed the TRF, which means both models start receiving the data from the most recent time segment before the out of time holdout, and in every step, they get an additional segment of the training set. In the first step, since there is only one segment, the TRF works like an RF, except the RF implementation has more hyper-parameters to optimize the learning. As we add segments, the TRF design should start having different behavior. The results can be seen in Figure 6.4.

We expected the models' performance to be closer in the first segment added, but we can observe they had a reasonable difference in four out of six cases, meaning the scikit-learn implementation differs reasonably from the TRF. Regarding how both models reacted to older data being added,

Dataset	Data split	Volume	Time range	RF	TRF	Δ TRF-RF
GE News	Train	21k	2015-2017	**.948**	.904	-.044
GE News	Test	5k	2015-2017	**.905**	.872	-.033
GE News	Holdout	58k	2018-2020	.821	**.834**	.013
20 News	Train	8k	-	**.822**	.780	-.042
20 News	Test	2k	-	**.774**	.746	-.028
20 News	Holdout	8k	-	**.708**	.703	-.005
Kickstarter	Train	98k	2010-2013	**.700**	.689	-.011
Kickstarter	Test	24k	2010-2013	**.688**	.681	-.007
Kickstarter	Holdout	254k	2014-2017	.620	**.648**	.028
Animal Shelter	Train	75k	2014-2017	**.794**	.790	.004
Animal Shelter	Test	19k	2014-2017	**.787**	.783	.004
Animal Shelter	Holdout	61k	2018-2021	**.789**	.787	.002
Olist	Train	41k	2017	**.720**	.660	-.060
Olist	Test	10k	2017	**.649**	.625	-.024
Olist	Holdout	62k	2018	**.636**	.627	-.009
Chicago Crime	Train	100k	2001-2010	**.902**	.900	-.002
Chicago Crime	Test	61k	2001-2010	**.900**	.898	-.002
Chicago Crime	Holdout	90k	2011-2017	**.901**	.899	-.002
Building Permits	Train	90k	2013-2015	**.974**	.967	-.007
Building Permits	Test	22k	2013-2015	**.970**	.963	-.007
Building Permits	Holdout	87k	2016-2018	**.959**	.955	-.004

Table 6.2: *When comparing the AUC in the holdout from the TRF to the RF, the benchmark gets better performance on three cases. However, the difference between challenger and benchmark in the holdout always drops compared to the same difference in the test.*

Dataset	Domain classifier AUC
Kickstarter	0.843
Olist	0.805
GE News	0.794
Chicago crimes	0.688
Animal shelter	0.684
Building permits	0.605

Table 6.3: *The higher the AUC in a domain classifier, the higher the difference between the training data and the data the model is applied to (holdout). TRF shows a better performance in the cases where this difference is high.*

the results are mixed. There was not a clear trend about always getting better or being harmed. But notice the TRF curve has a positive slope both for Kickstarter and GE News, the two cases where it performed better than the RF. Two interesting cases are the Olist and Chicago Crimes since the curves for both models started very close, but the TRF degraded as we approached the oldest data available. There is a chance it is not exactly that old data is impacting the TRF, but how the model uses each segment. Depending on the volume of the new segments, they will prevent the trees from growing until their maximum available depth, an aspect that deserves further investigation.

We can open this analysis to show not the aggregate performance in the holdout but its performance over time. In Figure 6.5 and 6.6, the darker curves include older segments. The blue curves represent the RF, while the green ones are for the TRF. It is not clear if the plots provide evidence that the older data can change the shape of the performance degradation. In the Animal Shelter dataset, we can see that adding older data made both models perform worse for future data from the holdout. At the same time, the opposite was true for GE News. In general, for both models,

the curves of different shades are very similar.

6.1.4 Feature importance and relationship with the score

In order to make feature importance comparable between RF and TRF, we use the permutation importance approach, which is model agnostic. The permutation importance shuffles a feature and checks how it impacts the model performance. It is used in the PIMP feature selection algorithm [ATSL10]. If the output is positive, the performance dropped after the permutation, so the feature is important for the model. The higher the output, the more important it is. If it is negative, the model could perform better using a shuffled version of a feature, which means the original feature behaves like a random noise in which the model could find some patterns during the training process.

The results for the tabular datasets can be seen in Table 6.4. The Kickstarter case is interesting since the TRF has not used two features (currency, country) that the feature importance proved not essential or close to random noise for the RF. It seems to support the hypothesis of the TRF as a possible feature selection algorithm. In general, there was no disagreement in the most important features, as we can see in Figure 6.8.

To make both algorithms' behavior more explicit, we inspected how changes in the features impacted their scores. Thus, we use a Generalized Additive Model (GAM) trained using the out of time holdout, which we split into train and test. The features from the original model are used as inputs, and its score is used as the output. The GAM is predicting how the RF or the TRF predict. The results shown are the predictions in the test set from the GAM split. A grid of input values is created to feed the GAM and reveal a functional form for both models use a particular feature.

A couple of illustrative cases can be found in Figure 6.9. We have only one numerical feature in the Kickstarter dataset, the crowdfunding campaign's goal, which is also the most important for the two models. The TRF seems to use "goal" in a simpler way compared to the RF since close to a goal of 100k, the effect in the score is pretty much the same. The main category feature also does not support much disagreement, there are only a few cases in which the two models see an opposite effect of a specific class, but it is more common to see higher impacts from the RF. The same thing goes in the Animal Shelter with "Age Upon Intake" and in the Olist dataset with the feature "difference_limit_delivered": they are similar, but the impact on the score of these features is less extreme for the TRF in general, which suggests the idea that its design could be just forcing models to be simpler.

Is there any pattern in features that changed their importance from RF to TRF?

An abrupt importance change from RF to the TRF could provide insight into TRF learning invariances. In Figure 6.8, we can see that in the Building Permits dataset, the least important feature in the RF was critical to the TRF. Investigating it, we observed that the "Existing Use" and "Proposed Use" have the same value 74% of the time, and the most important classes from them have a very similar behavior over time in respect to the average target value, as seen in Figure 6.11. It means the TRF has not found anything in particular. The two models are just using two different features to extract the same signal.

Using the Figure 6.8 and analyzing in Table 6.4 the absolute importance of the features that changed, we can see that changes in the feature importance from one model to the other mostly happened for unimportant features. In all datasets, there were dominant features that made it to the top of importance for both. We estimated the importance of the features while we added older data to the reverse learning curve to clarify it. In Figure 6.12, we see the result for the Olist dataset, in which for both models, the top features were stable as older data was added, while the other features changed quite frequently.

In the GE News dataset, when we look at the top 20 most important words in Table 6.5, we cannot see anything intuitively clear about the words selected by the TRF and the ones important for the RF.

	RF		TRF	
Dataset	Feature	Importance	Feature	Importance
Kickstarter	goal	.065	goal	.087
	category	.012	main_category	.010
	main_category	.010	category	.009
	currency	.003	country	-
	country	-.001	currency	-
Animal Shelter	Intake Type	.091	Intake Type	.090
	Age upon Intake	.070	Age upon Intake	.069
	Sex upon Intake	.030	Sex upon Intake	.020
	Intake Condition	.020	Breed	.016
	Animal Type	.008	Animal Type	.013
	Breed	.004	Intake Condition	.012
	Color	-.002	Color	.001
Olist	difference_limit_delivered	.046	difference_limit_delivered	.040
	difference_estimated_delivered	.022	difference_estimated_delivered	.024
	freight_value	.001	product_length_cm	.002
	product_length_cm	.000	freight_value	.001
	price	-.000	product_weight_g	.001
	product_category_name	-.001	product_description_length	.001
	seller_id	-.002	product_height_cm	-.000
	product_description_length	-.002	price	-.000
	product_photos_qty	-.002	product_category_name	-.000
	product_weight_g	-.002	product_name_length	-.001
	product_height_cm	-.002	product_width_cm	-.001
	product_width_cm	-.003	product_photos_qty	-.001
	product_name_length	-.004	seller_id	-.001
Chicago Crime	FBI Code	.234	FBI Code	.095
	Description	.052	Primary Type	.083
	Primary Type	.050	Description	.079
	Location Description	.016	Location Description	.013
	Domestic	.002	Domestic	.001
	Latitude	.002	Beat	.000
	Longitude	.001	Longitude	.000
	Beat	.001	Zone	-.000
	Address	.001	Community Area	-.000
	District	.000	Latitude	-.000
	Ward	.000	Ward	-.000
	Zone	.000	Address	-.000
	Community Area	.000	District	-.000
Building Permits	Estimated Cost	.174	Proposed Use	.134
	Existing Units	.020	Existing Use	.122
	Existing Construction Type	.018	Estimated Cost	.091
	Existing Use	.013	Plansets	.065
	Proposed Units	.009	Number of Proposed Stories	.056
	Supervisor District	.002	Proposed Units	.051
	Number of Existing Stories	.001	Number of Existing Stories	.016
	Plansets	.001	Existing Units	.009
	Number of Proposed Stories	.001	Existing Construction Type	.007
	Fire Only Permit	.000	Supervisor District	.006
	Proposed Construction Type	.000	Neighborhoods - Analysis Boundaries	.005
	Neighborhoods - Analysis Boundaries	-.000	Proposed Construction Type	.004
	Proposed Use	-.004	Fire Only Permit	.003
	Street Suffix	-.007	Street Suffix	-.000

Table 6.4: *The permutation feature importance shows the RF and TRF had very few cases of disagreement in the most important features. The most noticeable is in the Building Permits dataset, which we investigate in the section 6.1.4*

6.1.5 Time Robust Forest as a feature selection algorithm

In Chapter 5, we raised the question about the TRT design revealing a subset of the input features whose relationship with the target is stable. Such a subset would be helpful to feed any predictive model. To gather evidence to discuss it, we used the TRF's feature importance to select the most important ones or simply the ones used by the algorithm. For example, in the Kickstarter

RF		TRF	
Feature	Importance	Feature	Importance
atleta	0.043446	atleta	0.090731
cada	0.034933	sabe	0.036755
partidas	0.019554	cada	0.034831
primeiro	0.014367	partidas	0.021660
sabe	0.012230	lateral	0.016985
dia	0.010441	chegar	0.015238
cinco	0.009303	dia	0.012248
lateral	0.006325	cinco	0.011322
vídeo	0.006182	camisa	0.006628
rodrigo	0.005493	precisa	0.006192
defesa	0.004620	globoesporte	0.004892
globoesporte	0.004507	defesa	0.004150
precisa	0.004204	todos	0.003037
momento	0.003925	possível	0.002211
treinador	0.003699	lesão	0.001797
camisa	0.003505	vídeo	0.001595
hoje	0.003442	vitória	0.001177
difícil	0.003328	dentro	0.000943
possível	0.003327	neste	0.000874
todos	0.003132	nova	0.000581

Table 6.5: *The 20 most important words in the GE News task for the RF and TRF. Many of them were important in both models and there is no easy pattern differentiating which words the TRF prioritized.*

dataset, 2 out of 5 features were not used by the TRF: the currency and the country. We optimized an RF with this subset and compared it to the benchmark with all the features and the TRF. As seen in Figure 6.1, from the five datasets we could perform this exercise, only one of them has shown an improvement, the Chicago Crime. However, we can see the curve shifted vertically even before the out of time period, which does not offer evidence that the set of features is remarkably robust to future changes. Even in the Kickstarter case, where the TRF performed well, the feature selection did not work. It might suggest that the time-wise splits used by the TRF play an important role, and a feature can possess stable and unstable relationships with the target. The non-stable ones could be very predictive during a specific period and be preferred by the learning algorithm. As we have seen in the section 6.1.4, the "goal" was an important feature for the TRF, but its usage of this feature was slightly different when compared to the RF, which means that even if we deliver the set of features the TRF considers important to an RF, it might overfed.

6.1.6 The minimum examples by period

As mentioned, we have not performed a hyper-parameter optimization in the TRF, but we reused the parameters from the optimized RF. However, we changed the minimum parameters by period to understand its effect on the performance. One of the results from this experiment can be seen in Figure 6.13. The dashed lines represent results from the TRF, while the continuous are from the RF. We use the same colors for the same set from both models. Notice how the continuous green curve representing the holdout is always above the dashed one, except for the GE News dataset. It means that in most cases, there was no different value for this parameter that could make the TRF deliver a holdout performance higher than the RF. Nonetheless, the TRF has performed better when the RF could not pick a good value for this hyper-parameter.

We can also observe that the minimum examples by period assume low values in high-performing models even when the datasets are large. The maximum value for the holdout curve from the TRF

always happened at values lower than 25, which is expected, and we investigate the trade-off involved in this choice in the next section. It opens a future investigation about how this volume affects the model considering some periods have way fewer data than others. Bootstrapping all the periods to have the same volume or using a percentage requirement would be interesting experiments. It enforces the point on how different periods with different volumes of data impact the approach, as mentioned in the section 6.1.3.

6.1.7 A knob to control the trade-off between stability and short-term performance

The Figure 6.13 does not tell the entire history about hyper-parameter change and the influence on the performance. For the Time Robust Tree, the parameter that controls the minimum number of examples by period was hypothesized to be a knob to play with the trade-off between performance and stability, which could be more pronounced than the similar parameter that controls the necessary sample to split a node in the regular Decision Tree.

An experiment was done using a single DT and a TRT using the GE News dataset. As seen in Figure 6.14, in the DT, the shape of the curve does not change as much as in the TRT, in which it gets to the point of reverting the model performance degradation at the expense of a high-performance drop cost. However, when we extended this experiment to the RF and TRF in the other datasets, the TRF has not shown the same capability in general, as seen in Figure 6.15. It still happens in the GE News case, but not different than in the RF. Further, in the Building Permits, the RF shows this behavior while the TRF could not.

In Section 6.1.4, we have seen the numerical features seem to show a simpler behavior. When we combine it to the performance curves in Figure 6.15, it suggests the TRF could be mostly pushing to simpler model definitions instead of enabling it to learn rules the RF could not.

6.2 Evaluating the empirical support match in different environments

Considering the inductive bias in the TRF, the data must cover the same range of values in all the features for all the environments to enable the algorithm to find a common split under different contexts. To verify the empirical support match for numerical variables, we create quantiles using the entire training set using n bins; group the data by the environment, which is the time segment, e.g., year; count how many unique bins there are in every environment; divide it by the number of bins and take the average.

In the categorical case, it is similar. We go directly to count the unique categories in every environment since the quantile step is unnecessary. The result in Table 6.6 shows that most of the cases had a high empirical support match. The lowest value belongs to the Kickstarter case, which had a great benefit from the TRF, which suggests that different things come into play in terms of favorable properties of a problem to offer a good use case for TRF. It indicates a path for further investigation about how environment diversity and empirical support match relates.

Dataset	Empirical Support match
Building permits	0.886
Animal shelter	0.859
Chicago crimes	0.849
Olist	0.784
Kickstarter	0.611

Table 6.6: *The support match translates how the environments overlap in terms of feature coverage. The higher, the better to find common splits. The relationship of support match and diversity of environments needs further investigation.*

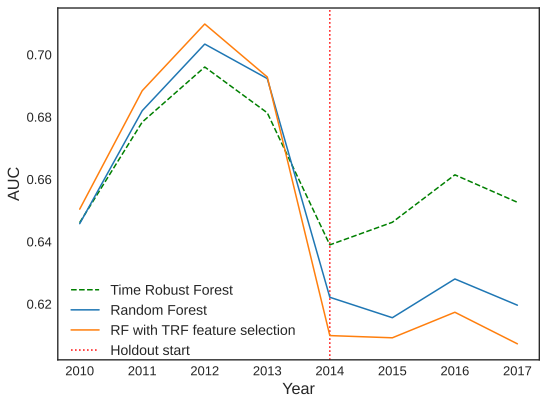
6.3 Would any practice be able to help the RF in the cases the TRF was better?

When we did the impossible exercise of changing the parameter related to the minimum number of examples in a leaf while looking to the future and unseen examples' performance, the RF presented a competitive performance in the Kickstarter dataset, as seen in Figure 6.13. It means it can learn the needed rules to achieve it. The following question is: would we be able to reach that set of hyper-parameters without looking into the future?

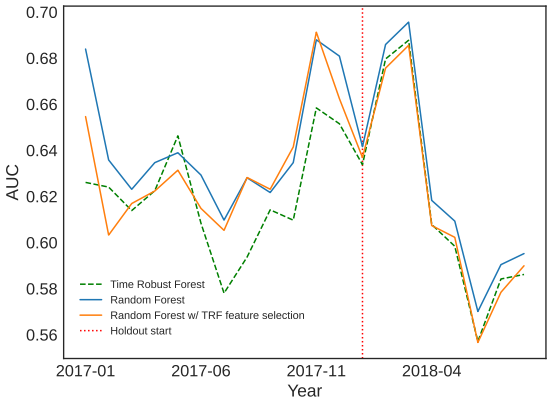
In the Kickstarter case, we can see the RF Holdout performance curve is superior to the TRF's one for most of the parameters, but the RF ended up picking a value in the region it is not better. Since the most recent data resembles the future unseen, we optimize the RF's hyper-parameters by doing a second time split in the in time set. We will use two designs to try to surpass the TRF performance: leaving the entire year of 2013 as a validation set for the optimization and leaving only half of it. After the optimization, the final model is trained using the whole period, including 2013.

As seen in Figure 6.18, these approaches could indeed improve the benchmark performance. Leaving 2013 as a validation set for the optimization process increased the performance in the out of time set. Even in this case, the TRF could perform better. It means that the TRF could perform better even under standard practices to improve generalization in an out of time set.

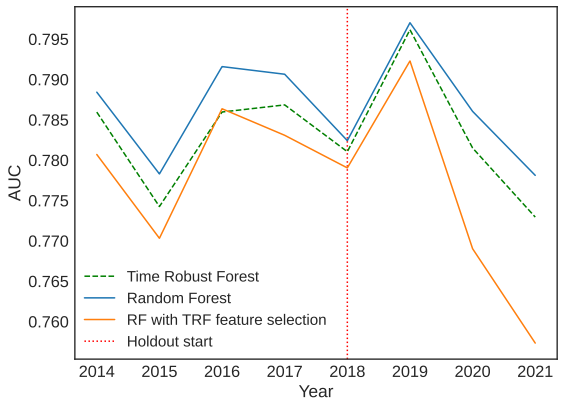
In the GE News case, the new optimization design could not improve the benchmark performance.



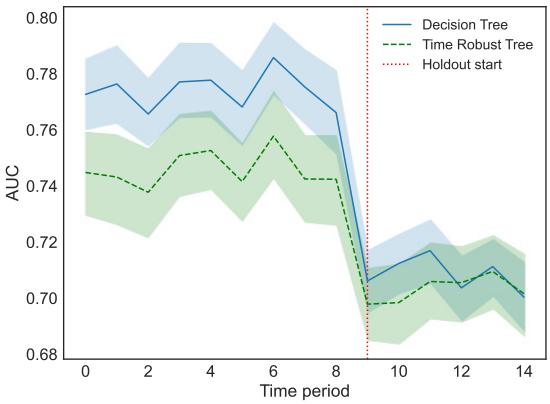
(a) *Kickstarter*



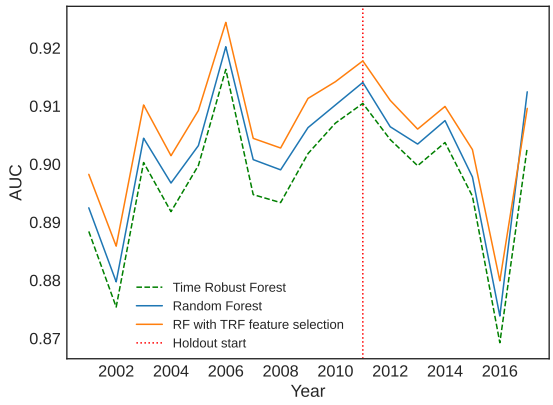
(b) *Olist*



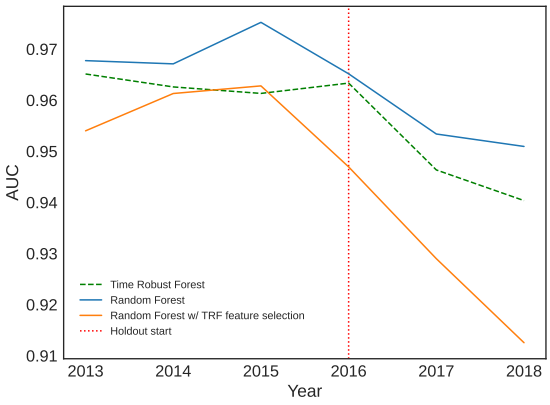
(c) *Animal Shelter*



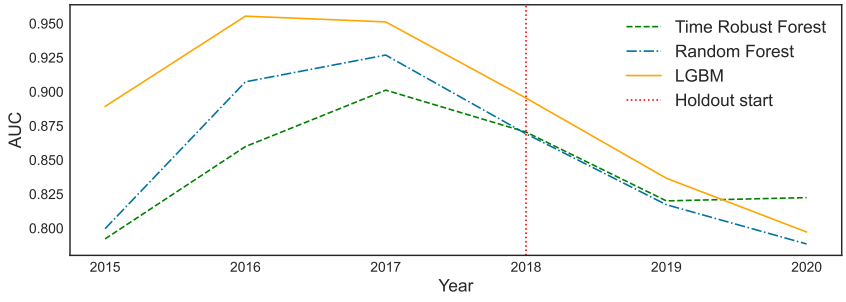
(d) *20newsgroup*



(e) *Chicago Crime*



(f) *Building Permits*



(g) *GE News*

Figure 6.1: *Comparing TRF to the RF in real-world public datasets. The benefits of using TRF vary depending on the dynamics of every dataset. In general, it shows better robustness when facing future unseen data.*

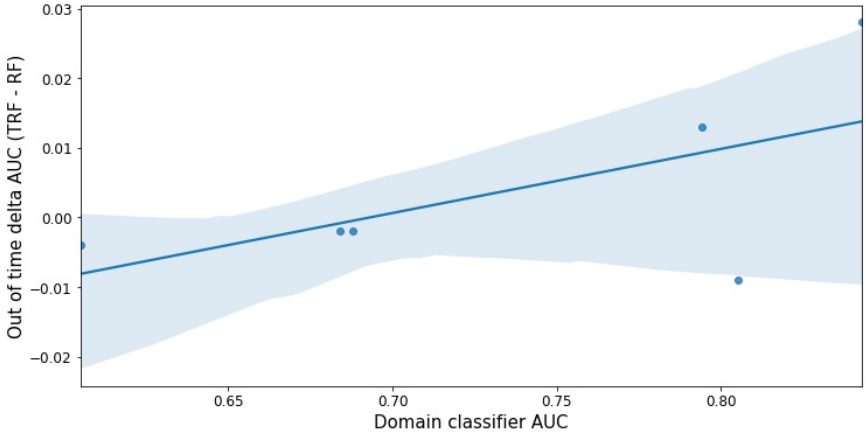


Figure 6.2: *The more significant the difference between the source data and the target data, which is translated by a high AUC for the domain classifier, the greater the benefit of learning invariant relationships in order to generalize to future unseen data. The shade shows the regression estimates confidence interval made with bootstrapping.*

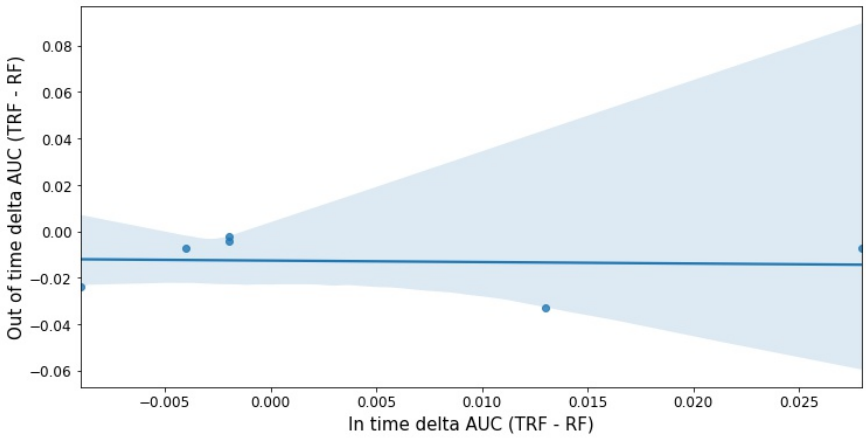
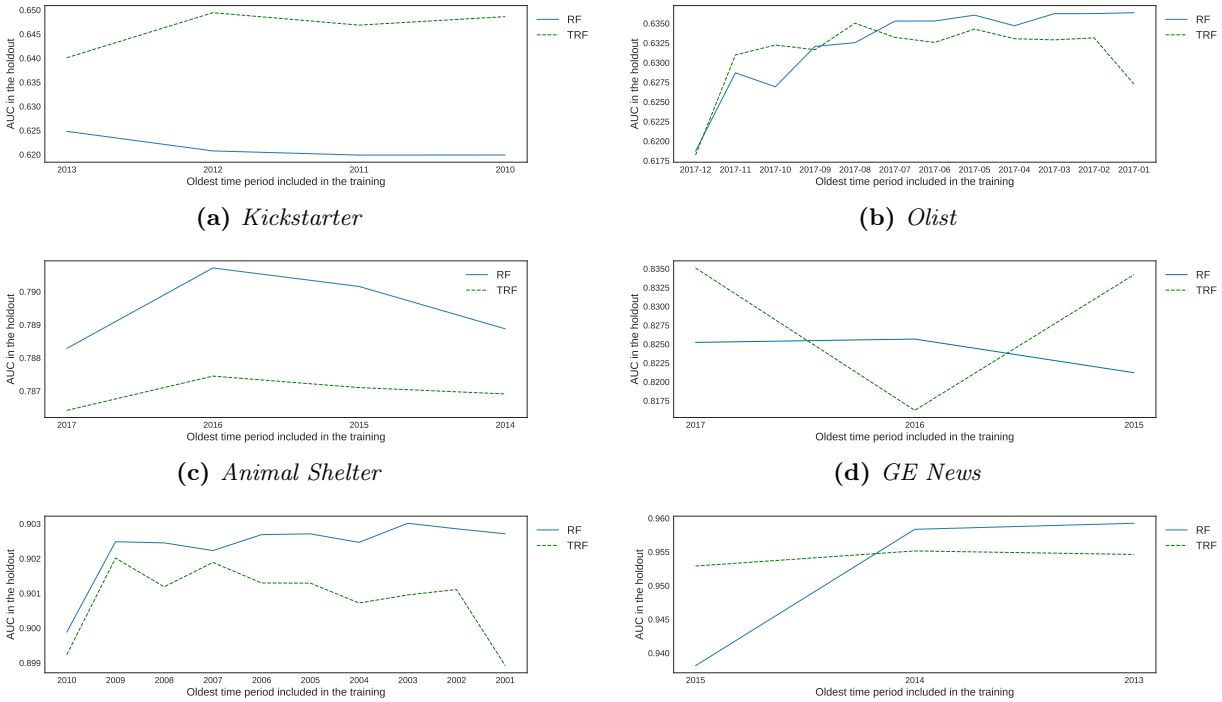


Figure 6.3: *The delta AUC in the in time test set cannot guide the TRF performance in the out of time holdout since there is no slope when we regress them.*



(a) *Kickstarter*

(b) *Olist*

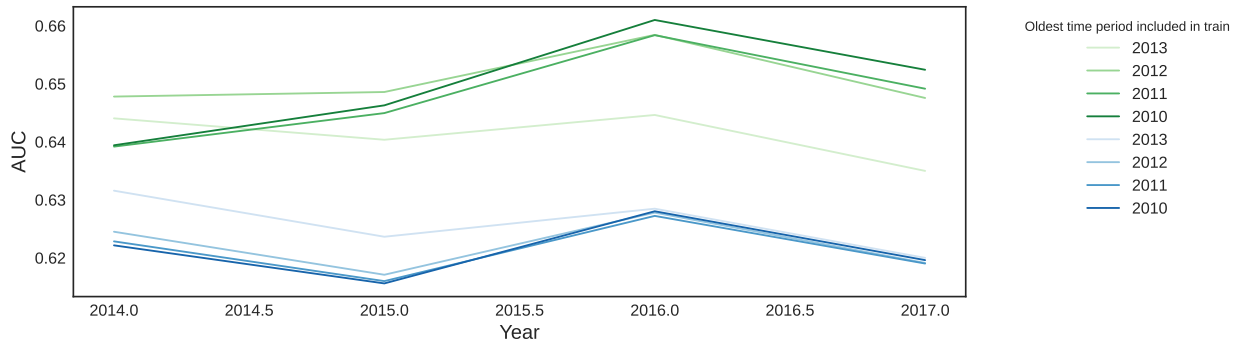
(c) *Animal Shelter*

(d) *GE News*

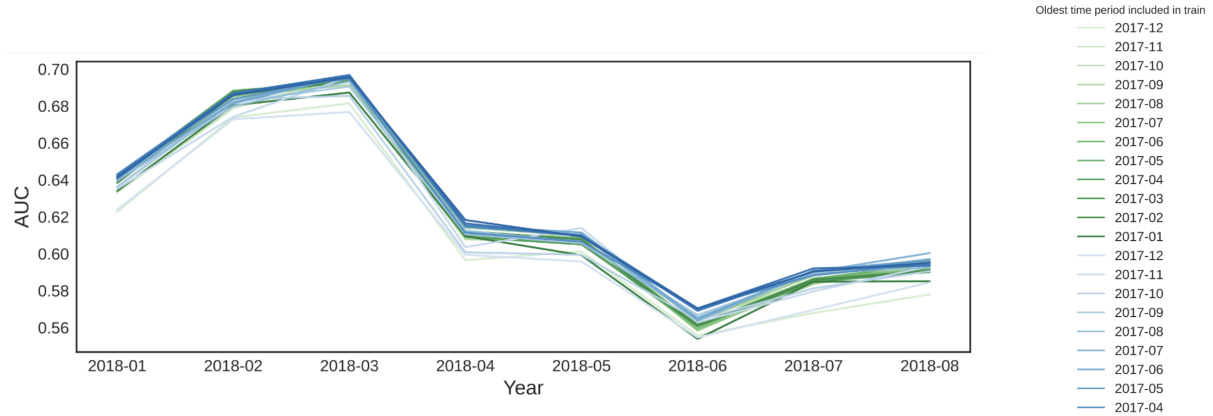
(e) *Chicago Crime*

(f) *Building Permits*

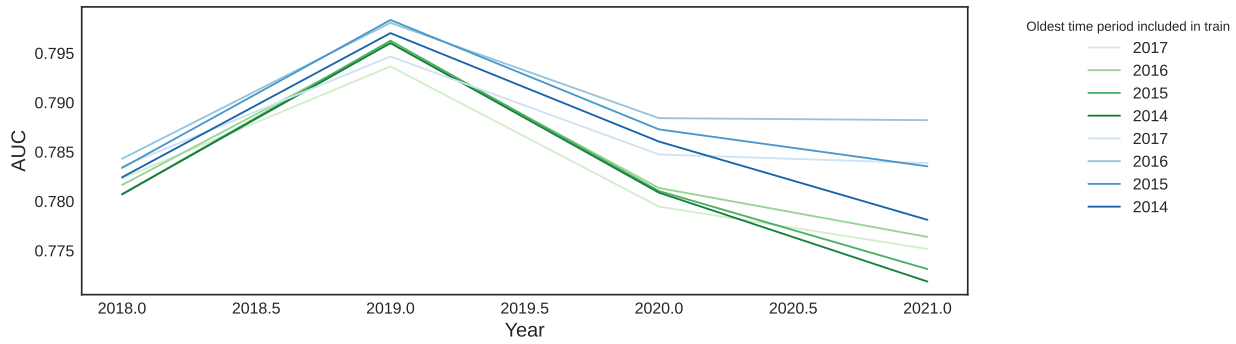
Figure 6.4: *In a reverse learning curve, older data is added to the training set, and a fixed out of time holdout is evaluated. We can see the TRF benefited most of older data exactly in the cases it performed better than the RF, which is in the Kickstarter and GE News.*



(a) *Kickstarter*

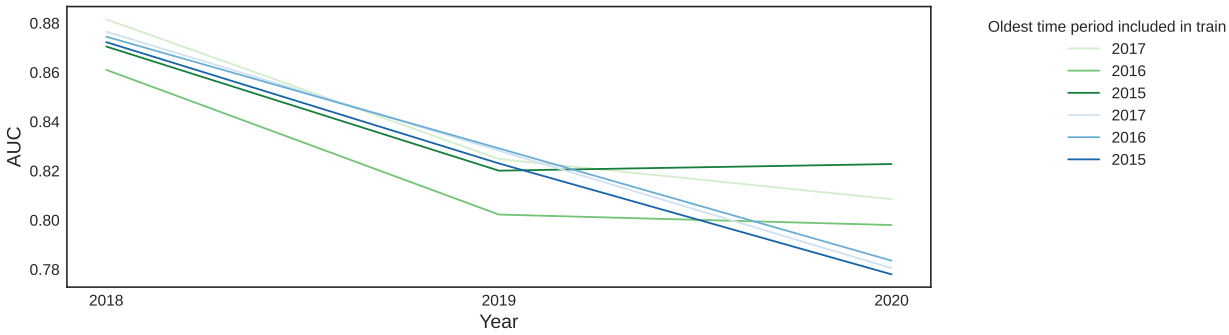


(b) *Olist*



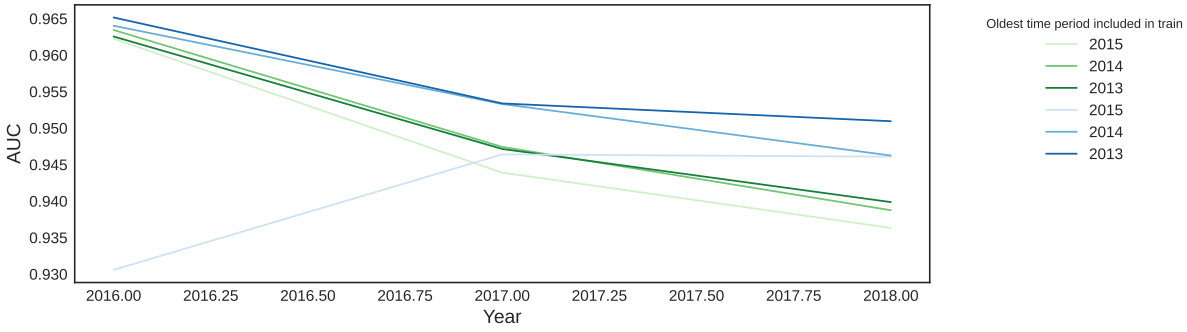
(c) *Animal Shelter*

Figure 6.5: *We added older segments to the training set and observed how the shape of the curve for the holdout performance overtime would change. In the three datasets in this figure, only the Animal Shelter shows older data degrading the performance in the last year included in the holdout, but for both models.*



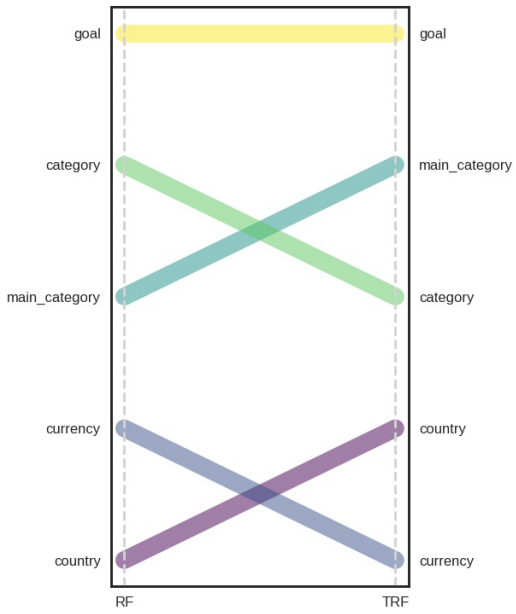
(a) *GE News*

(b) *Chicago Crime*

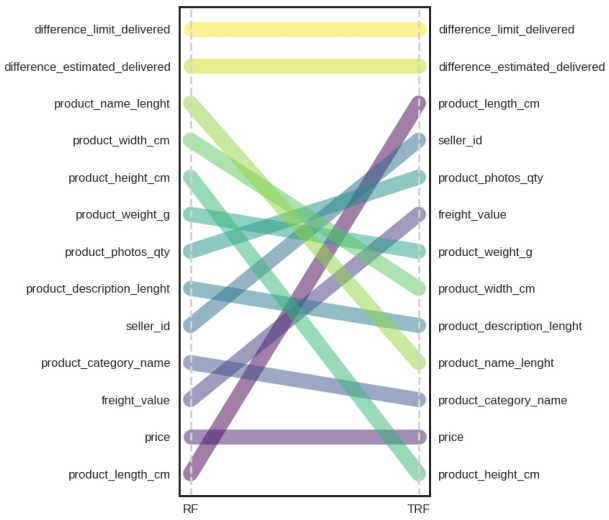


(c) *Building Permits*

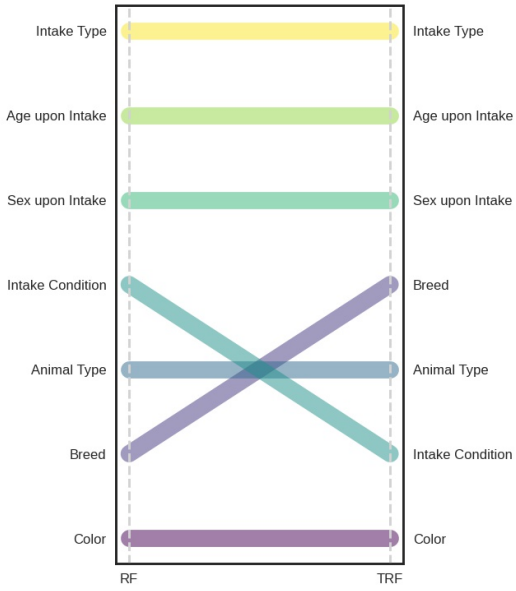
Figure 6.6: *We added older segments to the training set and observed how the shape of the curve for the holdout performance overtime would change. In the GE News, older data helped the TRF perform better in the future, while in the Building Permits, it helped the RF.*



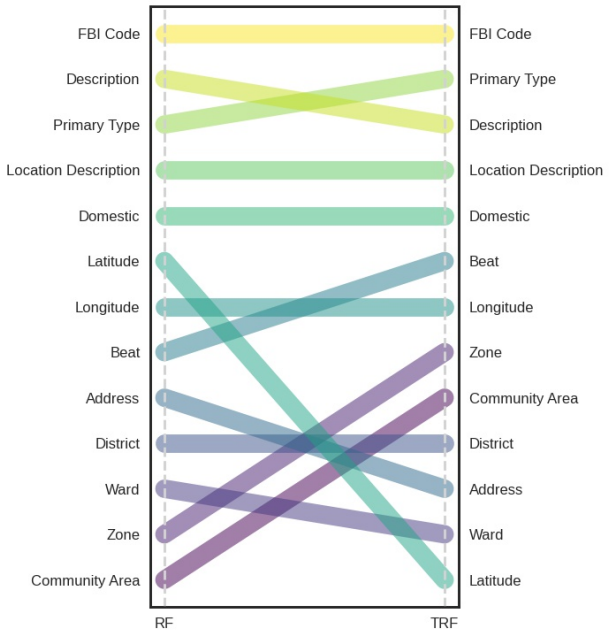
(a) *Kickstarter*



(b) *Olist*

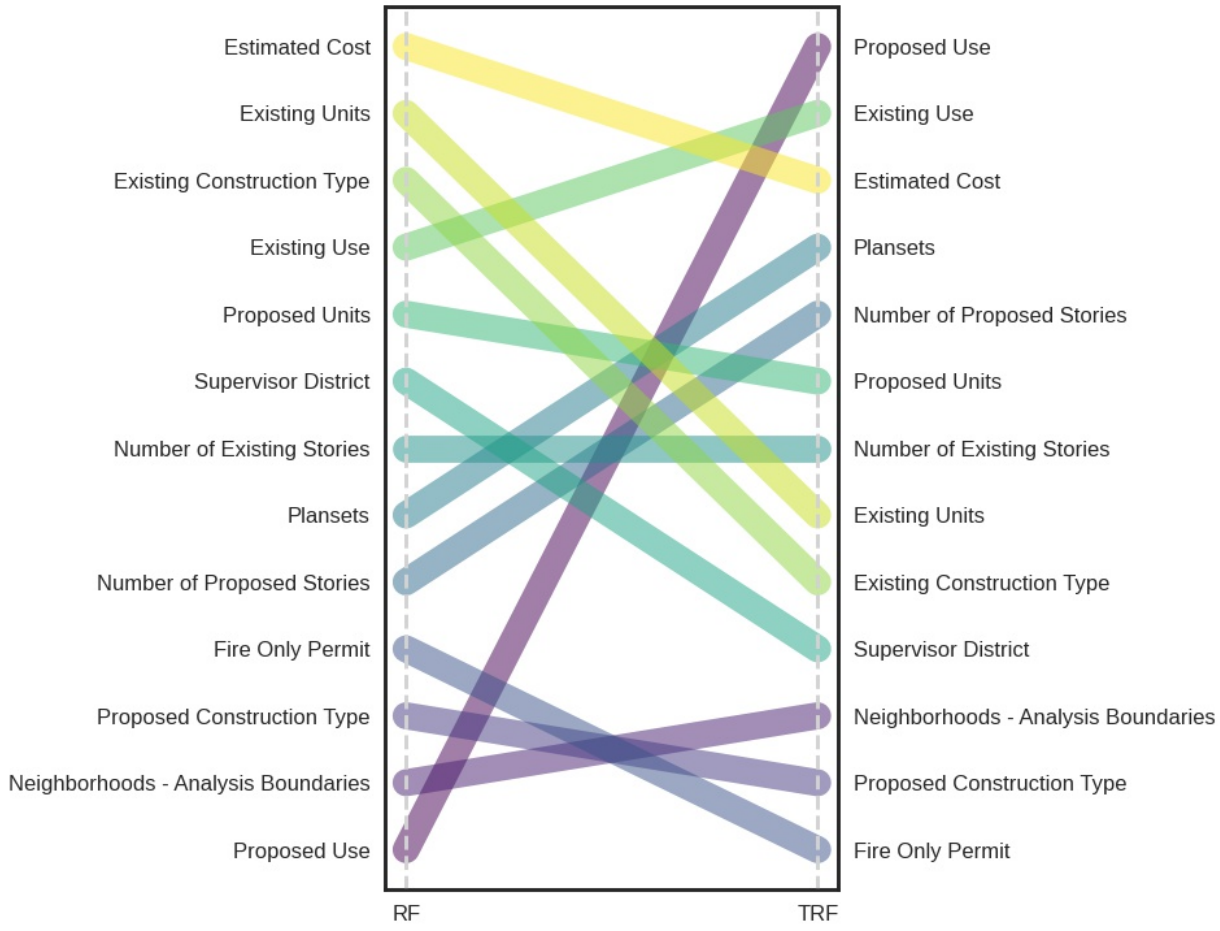


(c) *Animal Shelter*



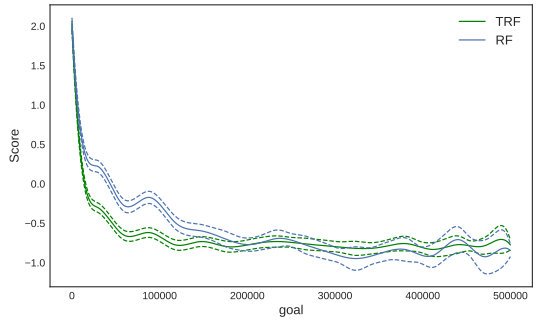
(d) *Chicago Crime*

Figure 6.7: *In general, the RF and the TRF shared the most important features.*

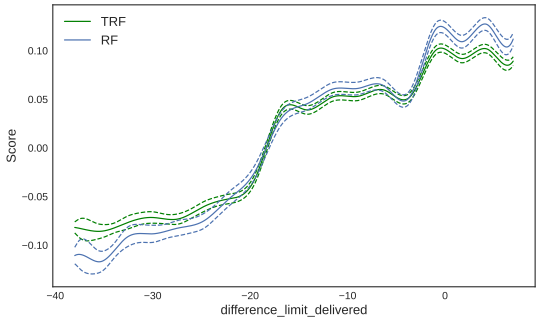


(a) *Building Permits*

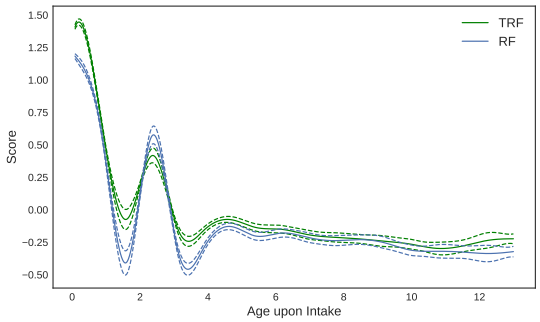
Figure 6.8: *For the Building Permits dataset, there was a case the least important feature for the RF migrated to the most important in the TRF. We investigate this case in the section 6.1.4.*



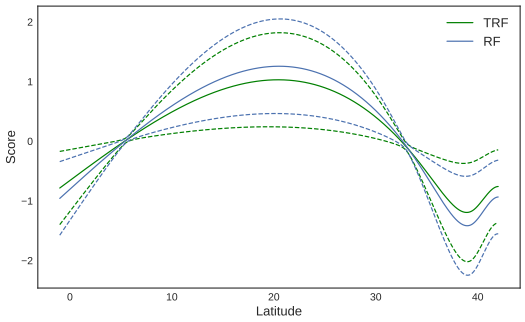
(a) *Kickstarter*



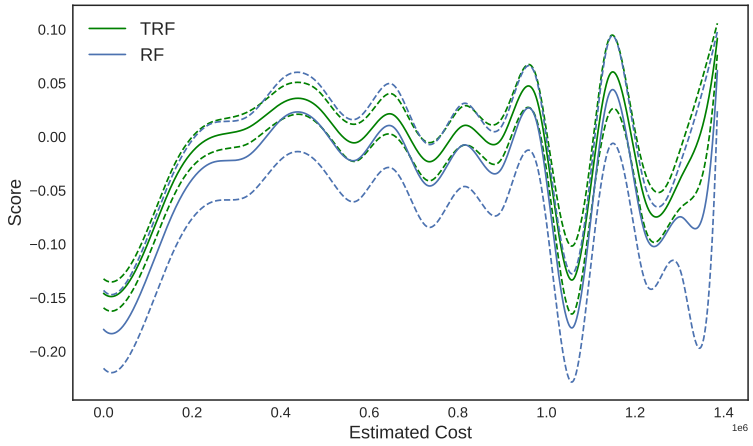
(b) *Olist*



(c) *Animal Shelter*

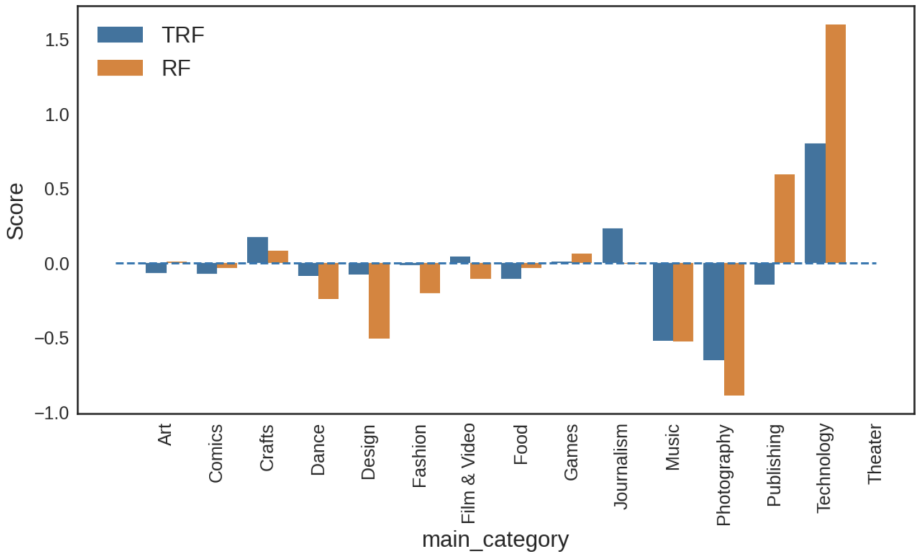


(d) *Chicago Crime*

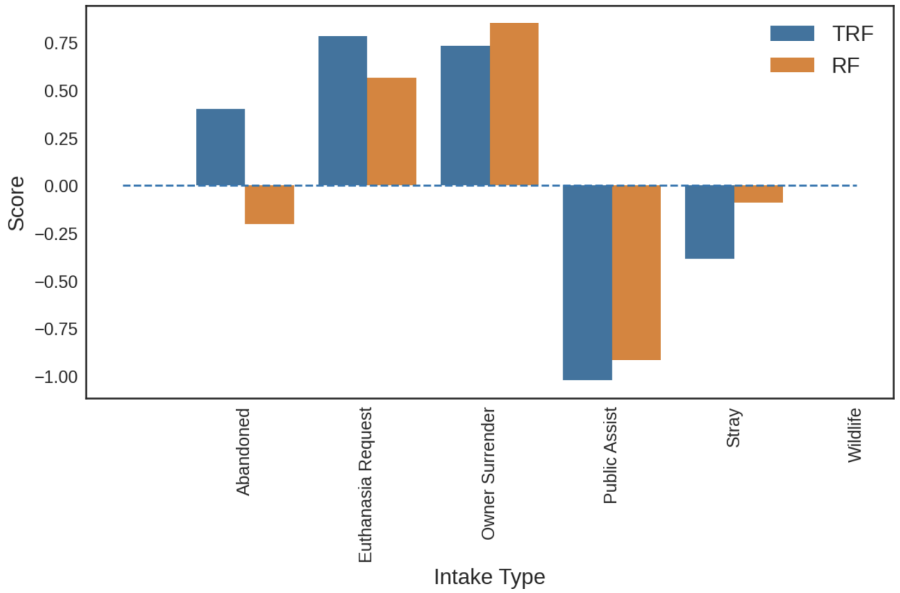


(e) *Building Permits*

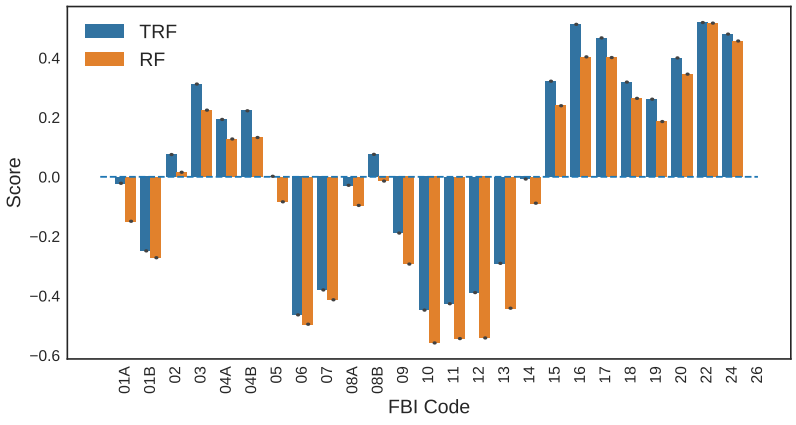
Figure 6.9: *The behavior of the most important numerical feature was very similar. At the most, we can say that in the TRF, it is slightly smoother. The dashed line represents a 95% confidence interval generated by the GAM.*



(a) *Kickstarter*



(b) *Animal Shelter*



(c) *Chicago Crime*

Figure 6.10: *The bars display how belonging to a specific class from a categorical feature impacted the score predicted by the RF and the TRF. We can see that the disagreement was very minimal.*

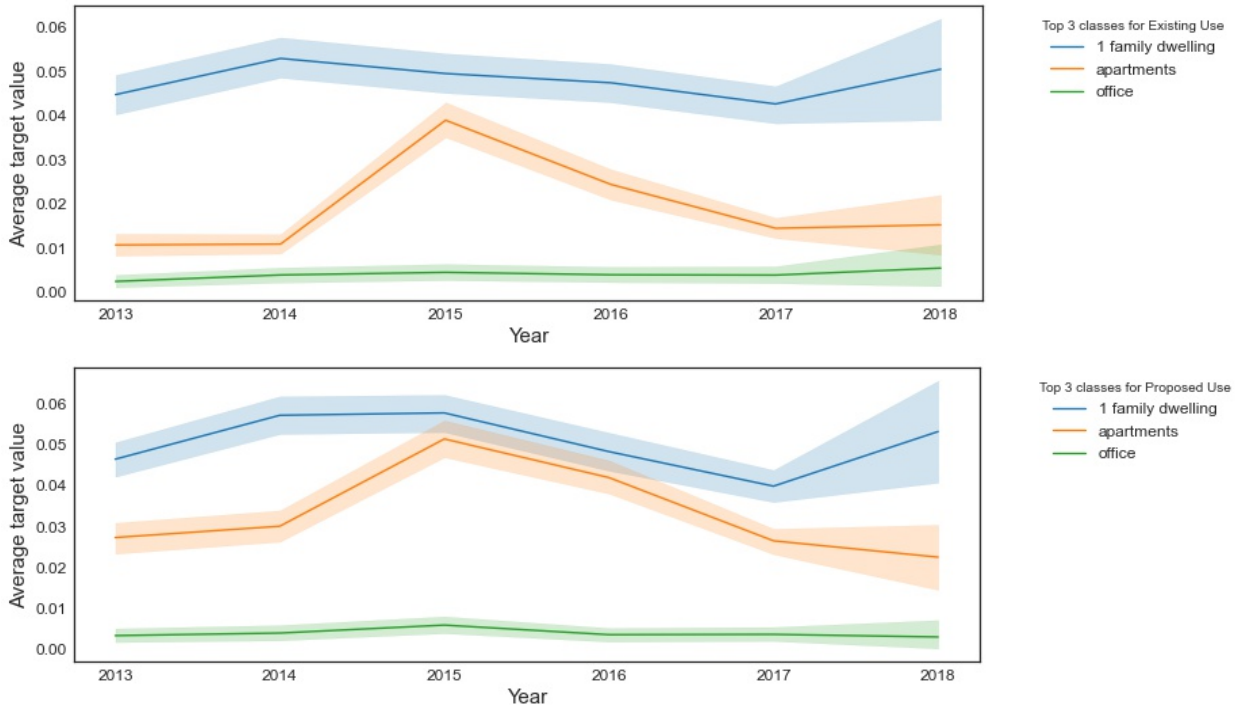


Figure 6.11: *The three most common classes in the Proposed Use and Existing Use features from the Building Permits datasets are equal and their relationship with the target over time is very similar. So the large difference in the Proposed Use importance in the RF and TRF does not provide any insight.*

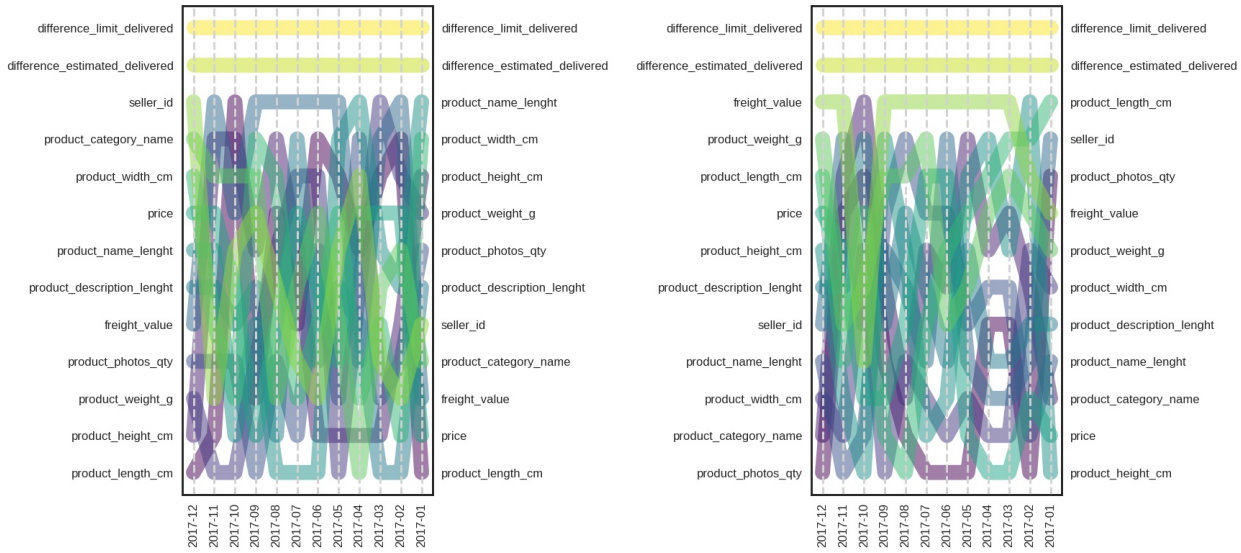
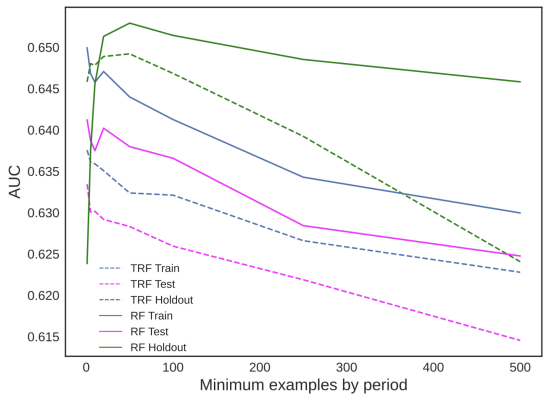
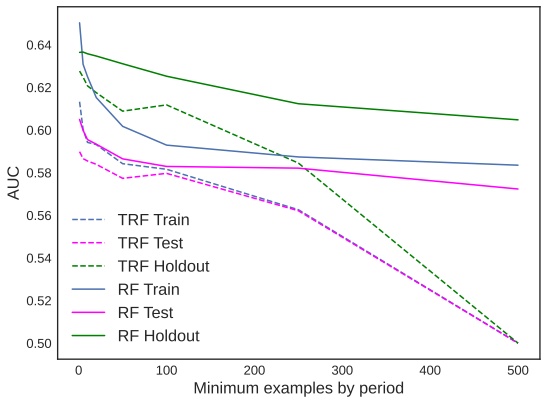


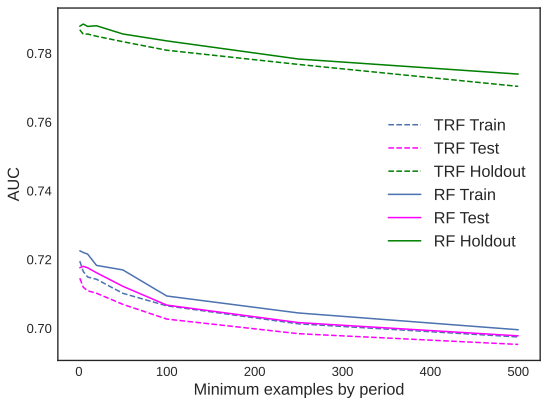
Figure 6.12: *In the horizontal axis, we have the oldest time segment, which was added to the training sample during the reverse learning curve. (a) RF, (b) TRF. As we added older data for the training, the importance of the features for the holdout prediction was stable for the top ones, while the unimportant changed in every round.*



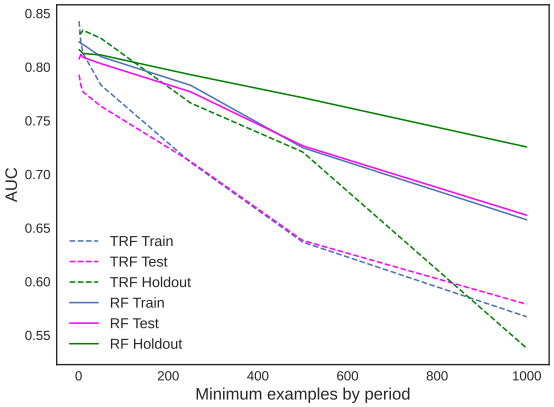
(a) *Kickstarter*



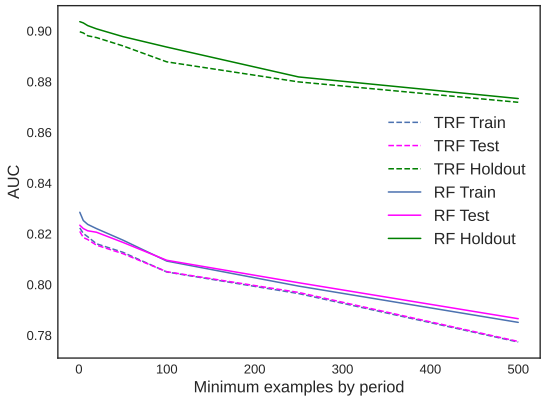
(b) *Olist*



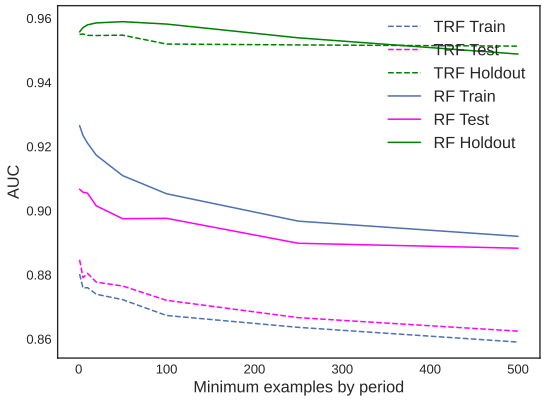
(c) *Animal Shelter*



(d) *GE News*



(e) *Chicago Crime*



(f) *Building Permits*

Figure 6.13: *As we changed the minimum example by period (TRF) or the minimum example for a leaf (RF) parameter, we could see how the performance in the different sets changed. The GE News dataset is the only one the TRF could achieve a performance the RF could not by only changing the minimum examples for a leaf node.*

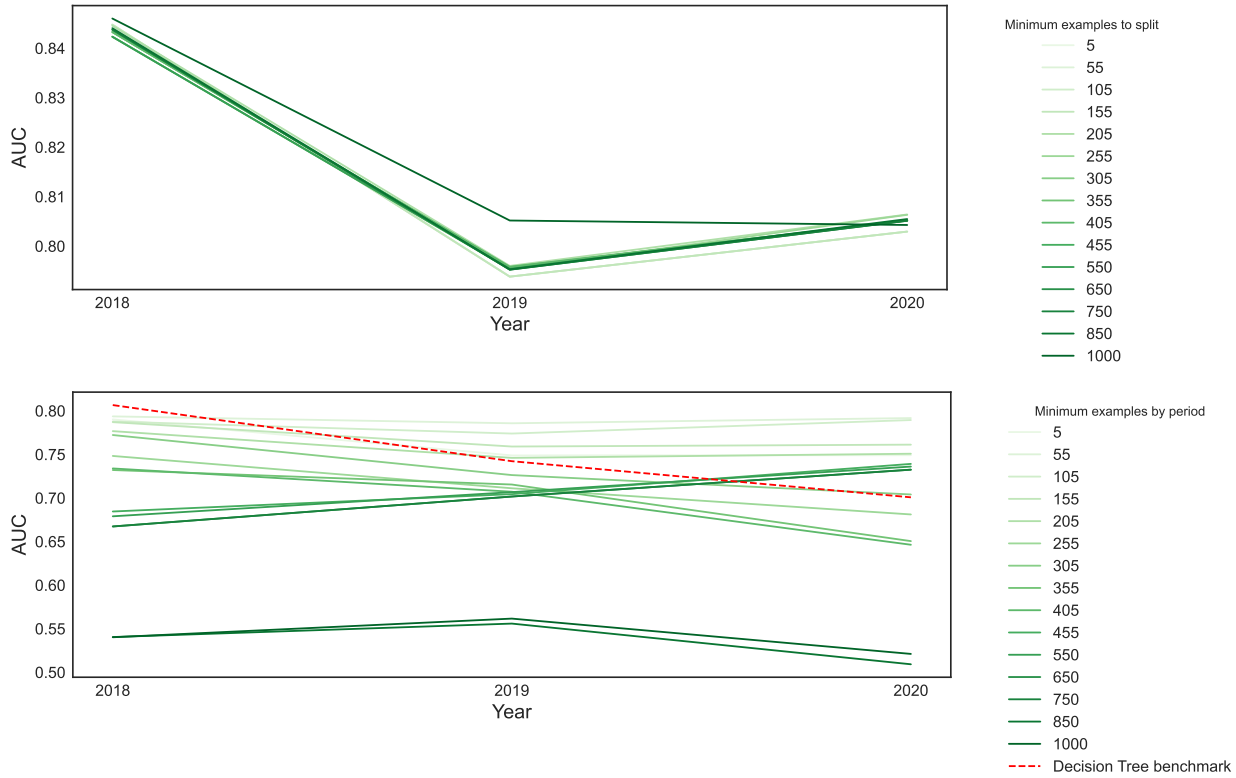
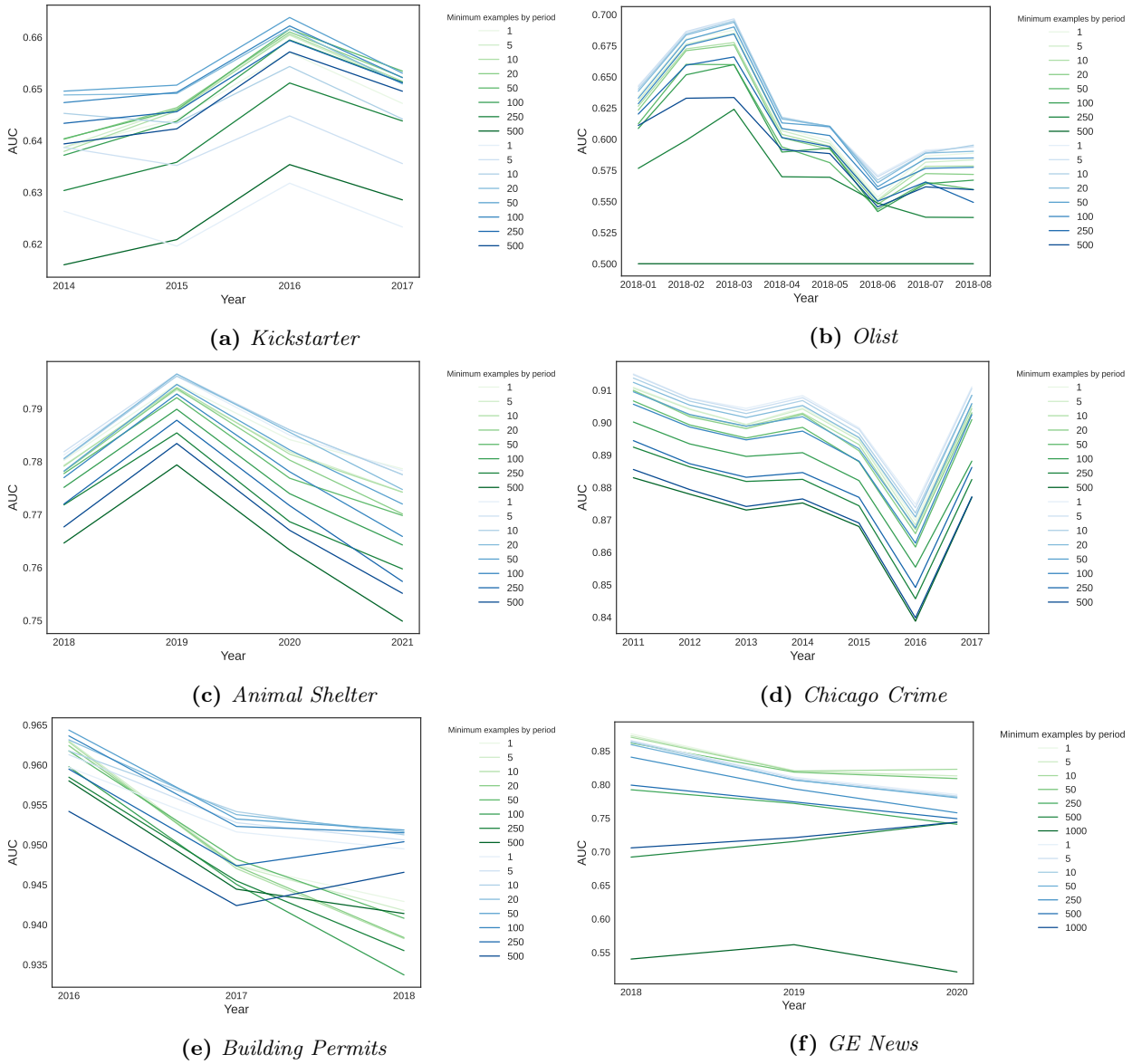


Figure 6.14: *The first image shows how the holdout performance change in time for a Decision Tree when the parameter for the minimum examples to split changes, while the one in the bottom shows the same thing for the Time Robust Tree's minimum examples by period to split. The data is from the GE News dataset. TRT shows an interesting impact on the performance curve shape over time.*



(a) *Kickstarter*

(b) *Olist*

(c) *Animal Shelter*

(d) *Chicago Crime*

(e) *Building Permits*

(f) *GE News*

Figure 6.15: *When changing the minimum examples needed in a leaf in the RF and by period in the TRF, we expected to see a higher impact in the curve shape overtime in the TRF case, but it was not observed. On (a) Kickstarter, (b) Olist e-commerce, (c) Animal Shelter, (d) 20 Newsgroups, (e) Chicago Crimes, (f) Chicago Building Permits, (g) GE News.*

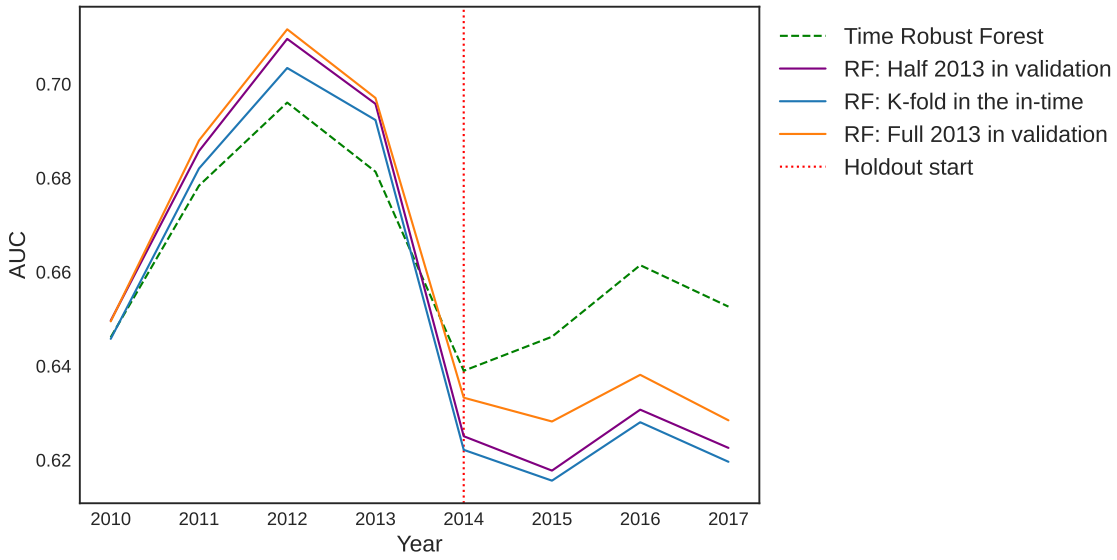


Figure 6.16: *Kickstarter*

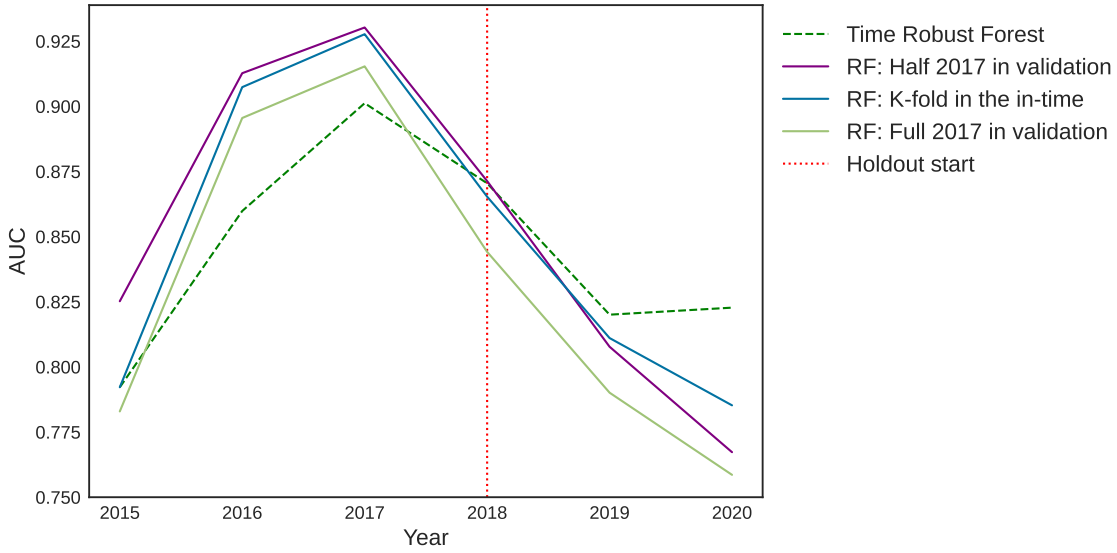


Figure 6.17: *GE News*

Figure 6.18: *Changing the hyper-parameter optimization process could help the benchmark in the Kickstarter case, but it was unable to achieve TRF's performance. In GE News, the process could not happen the initial benchmark. The base case is using the entire period with K-fold. The alternatives were leaving entirely or partially the most recent data from the training period (2013) as the validation set for the optimization process.*

Chapter 7

Conclusion

Time Robust Forest offers a new option to control the trade-off between short-term performance and stability, which is an advantage in a couple of settings where updating models is costly and the data distribution changes frequently. Beyond the practical implications, there is evidence that causal properties are an exciting dimension of predictive tasks. Though it has a severe design limitation, real-world data with a good time range should offer enough flexibility to enable a time period segmentation to overcome it.

7.1 When to use TRF

The Kickstarter and the GE News case look different. While the first has a few inputs and the RF could have performed better had the optimization process picked the right hyper-parameters, the GE News has 300 inputs and a highly dynamic task. The domain classifier exercise points to cases in which the data distribution changes more. However, we could see by the feature importance analysis that there is not much room to look for better features in most cases since the signal will concentrate on a few very important and reasonably stable features. It raises the hypothesis that the task needs to offer many different signals to enable a TRF to perform differently than an RF.

7.2 How the model limitations impact its application

When using the Time Robust Forest, it is necessary to have a timestamp column, a reasonable time range, and overlapping empirical supports for every period regarding the input features. The timestamp is commonly available information in real-world datasets since every data is generated in time, and it is common to store it. Long-range data is not uncommon since data time range follows its generator activity duration, and it is common to have laboratories and companies persisting for a couple of years. The empirical support of the input features in every period is the main limitation. It is possible to play with the time period definition until there is a decent overlap between them. As an example, imagine age as an input feature and an application for customer acquisition. If the time period definition is too short, like a day, it is very likely not to contain the age range present in other days. In the limit, one could define a time period short enough to have just one example on it, which would make the application of the algorithm impossible. If we grow this time period, there will be a size that will contain data about every age range for every period.

7.3 The environment segmentation as time periods

Though we segment the timestamp column in time periods to represent environments, it is an approximation prone to problems. Considering as different environments two periods of time while they were generated under the same environment does not harm our approach, except for requiring more data, consisting of a trade-off between making sure every period contains just one environment

and the data volume needed. Suppose we place two different environments in the same time period. In that case, we are losing an opportunity to offer the model two cases we want to keep relationships invariant and potentially enable the algorithm to create splits that are good only for one of the environments in the same time period. However, we still want invariance between this period with the two environments and all the other time periods in the training data. Thus, it is another side of the segmentation trade-off. While having fewer segments provides a higher volume of data in every period and enables learning more complex rules, it will also make it more likely that two different environments are sharing the same time period compromising the inductive bias for invariant rules.

7.4 Generalizing for different environments

Though we present here time as the environment we iterate over to find invariance, the model could have been presented as a Context Robust Forest. Context can take any environment to iterate, like geographical locations, including compounded ones like geographical-time. If the concern is generalizing under new data sources, like data from a new hospital in a healthcare application, one can use the hospital as the context or hospital-time.

7.5 Regularizing for any non-stationary knowledge is not all we need

By expressing our preference for stable and invariant relationships in the data, we can avoid absorbing spurious correlations, but we also prevent the model from learning genuine non-stationary relationships, which are causal but suffer constant concept drifts. Though invariant or slow-changing aspects of the world are an essential piece for intelligence, it is also noted that we need to learn the non-stationary parts of it [GB20]. The Time Robust Forest offers a way to find the slow-changing or stationary relationships we take longer to learn - the longer the time range in the data, the better to learn invariant properties. Intuitively, learning transient causal relationships helps us on intelligence tasks. Using the soccer example, the transient piece of knowledge of which are the two soccer clubs that classified to the championship finals can help in the task of identifying the club theme of a sports article when the term "championship finals" is present, even though it is far from being a stable property in respect to the two specific clubs.

7.6 Intersection with Domain Adaptation

Since disjoint input features for different environments is a general challenge in Machine Learning, there are approaches to perform a Domain Adaptation, such as Domain-adversarial Neural Networks (DANN) [GUA+16] and Maximum Mean Discrepancy Adversarial Auto Encoder (MMD-AAE) [LPWK18]. The general goal is to find a representation for the inputs that enables a better generalization for any domain.

In the DANN, the representation is an output of a feature extractor, which learns in order to solve at the same time the task of getting the target variable right and not getting the domain label right - which means the features do not carry environment specific anymore.

In the MMD-AEE, this representation is learned with an Auto Encoder. An Auto Encoder is a Neural Network architecture that learns a different representation of the input data by enabling it to be expressed by the hidden layers of a network that learns this representation using as the target output the input data itself [RHW85]. It learns how to represent the same data differently without destructing the original signal.

Adjustments of the loss function enable the modeler to force additional properties beyond the original information retention. One example of such loss is the Maximum Mean Discrepancy Distance (MMD) [GBR+12]. The MMD is a kernel-based approach that measures how different two distributions are. Given two distributions, we can define the MMD as the distance between their

embeddings in the Reproducing Kernel Hilbert Space (RKHS). It is called the Maximum Mean Discrepancy because it is equivalent to finding the function in the RKHS that maximizes the difference in the expectations between the two distributions. The most widely used metric for the distance between distributions is the Kullback-Leibler Divergence (KL-divergence). The usage of MMD in this work is motivated by the architecture proposed by Li (2018) [LPWK18] to solve the problem of the input distributions not sharing the same empirical support when coming from different environments.

7.7 Future work

Future lines of work for TRT itself or in Out of Distribution direction can explore automatic environment segmentation, other ways to learn a representation to match environments' empirical supports, how to combine boosting while respecting the invariance preference, the addition of parametric models to improve generalization out of seen data boundaries, and ensembles of differently regularized TRTs.

Appendix A

Deriving IRM for the linear regression case

In order to get a deeper understanding about the steps involved in the IRM paradigm, we derive the case for the linear regression. Consider a linear regression, where X is the set of input variables, Y is the target variable, β is the parameter vector, we want to estimate: $\hat{Y} = \hat{\beta}X$. In this case, the function Φ takes the form of βX.

To achieve it, we want to minimize the loss function $L(\beta)$. Using Gradient Descent, the search for the βs that minimize this function happens by the successive updates on them that take the following form:

$$\hat{\beta}_{t+1} = \hat{\beta}_t - \gamma \nabla_{\beta|\beta=\hat{\beta}_t} L(\beta) \tag{A.1}$$

Starting in the regular case, when we use the ERM, the minimized loss function L is the mean squared error. Thus, we learn $\hat{\beta}$ using:

$$\begin{aligned} L(\beta) = MSE &= \frac{1}{n}(\hat{Y} - Y)(\hat{Y} - Y)^T, \text{where } \hat{Y} = X\hat{\beta} \\ \nabla_{\beta|\beta=\hat{\beta}_t} L(\beta) &= -\frac{1}{N} X^T (X\hat{\beta} - Y) \end{aligned} \tag{A.2}$$

In the IRM, the loss function $L_{IRM}(\Phi, \omega)$ in Equation 4.8 is composed by two parts. The first is identical to the ERM case, however, now we iterate over the set of environments available in the training set ε_{train}. We distinguish the input matrix under a certain environment e as X^e and the target variable as Y^e. Notice that by using $\omega = 1.0$, we obtain $\omega \circ \Phi$ equal to β.

$$R^e(\omega \circ \Phi) = R^e(\hat{\beta}) = \frac{1}{n}(X^e\hat{\beta} - Y^e)(X^e\hat{\beta} - Y^e)^T \tag{A.3}$$

The second part of $L_{IRM}(\Phi, \omega)$ is composed by λ, a constant acting as a hyper-parameter, and the penalization function $\mathbb{D}$, in this linear case it is $\mathbb{D}_{lin}$ 4.9. We highlight that Φ becomes our predictor, which in this case means it is expressed by the parameters in β.

$$\mathbb{D}_{lin}(\omega, \Phi, e) = \left\| \mathbb{E}_{X^e}[(X^e\hat{\beta})^T (X^e\hat{\beta})]\omega - \mathbb{E}_{X^e,Y^e}[(X^e\hat{\beta})^T Y^e] \right\|^2 \tag{A.4}$$

Nonetheless, to update the parameters in $\hat{\beta}$ we need the gradient. Since L_{IRM} is composed by a sum, the gradient first term coincides with the derivation done for the ERM case, while the second term becomes $\nabla_{w|w=1.0}\mathbb{D}_{lin}(\omega, \Phi, e)$. Differentiating the Equation A.4 in respect to ω and then replacing it by 1.0:

$$\nabla_{w|w=1.0}\mathbb{D}_{lin}(\omega, \Phi, e) = 4\frac{1}{N}(((X^e\hat{\beta})^T X^e\hat{\beta}) - ((X^e\hat{\beta})^T Y)) \\ (X^e(X^e\hat{\beta})^T - ((X^e\hat{\beta})^T Y^e)) \tag{A.5}$$

Finally, the update of the weights under the IRM paradigm is:

$$\hat{\beta}_{t+1} = \hat{\beta}_t - \gamma \sum_{e\in\varepsilon_{tr}} \Big[(-\frac{1}{N}(X^e)^T(X^e\hat{\beta} - Y^e)) \\ +\lambda(4\frac{1}{N}(((X^e\hat{\beta})^T X^e\hat{\beta}) - ((X^e\hat{\beta})^T Y^e))(X^e(X^e\hat{\beta})^T \\ - ((X^e\hat{\beta})^T Y^e)))\Big] \tag{A.6}$$

A.1 Synthetic examples

To exemplify, we use the equations derived in the previous section to compare the IRM to the ERM paradigm[1]. Consider the following equations from the example provided by Arjovisky (2019) [ABGLP19]

$$\begin{aligned} X_1 &\sim \mathbb{N}(0, \text{environment}) \\ Y &\sim X_1 + \mathbb{N}(0, \text{environment}) \\ X_2 &\sim Y + \mathbb{N}(0, 1) \end{aligned} \tag{A.7}$$

They represent our training set. The *Environment*, however, can assume different values during the training and after it. We use ϵ_{unseen} to represent unseen environments considering the training examples available, they would be observed only when the model is applied to new data, which is different than random examples extracted from the training set in order to validate the model.

$$\begin{aligned} \epsilon_{train} &= (0.1, 1.0) \\ \epsilon_{unseen} &= (0.2, 0.05, 0.5, 1.2) \end{aligned} \tag{A.8}$$

As a consequence of the environment change, the relationship between X_2 and Y will also change on its functional form, since it is a non-causal relationship.

$$X_2 \sim \mathbb{N}(0, 1) \tag{A.9}$$

In this scenario, we want to build a model that can be trained on the data generated by the SCM A.7 that does not overfit to the non-causal relationships present on it, in a way that when it faces a new environment during the application stage, which follows the causal relationships present in the training, but presents different spurious correlations, it is able to present a good performance.

For this illustrative example, we will use three models:

- **ERM:** a linear regression using gradient descent without any kind of regularization. It is the benchmark model.

[1]The code is available at https://github.com/lgmoneda/msc-thesis/tree/main/literature/irm_logistic_regression_from_scratch.ipynb

- **Oracle:** It is a model built using the real parameters from the generating process for the target, and zero for any variable that does not define it. In this case, $\beta_1 = 1$ e $\beta_2 = 0$.
- **IRM:** It uses the loss function from IRM and a lambda value optimized using only training data. It is the challenger model.

As a performance metric, we use mean squared error (MSE) to compare the models. The error is always taken from out of train data: in the first case, the test data, they come from seen environments, but it is a random sample not used for training, while in the second case, the holdout data, they come from unseen environments.

As expected, the benchmark model explores the non-causal relationships to optimize the predictive performance showing the lowest error in the test data as seen in the Figure A.1. In this same test set, the Oracle model presents the highest error precisely by the fact it does not explore the spurious correlations from it. The challenger model is worse than the benchmark. Notice this situation is exactly the model validation framework extensively used to gather evidence about generalization power.

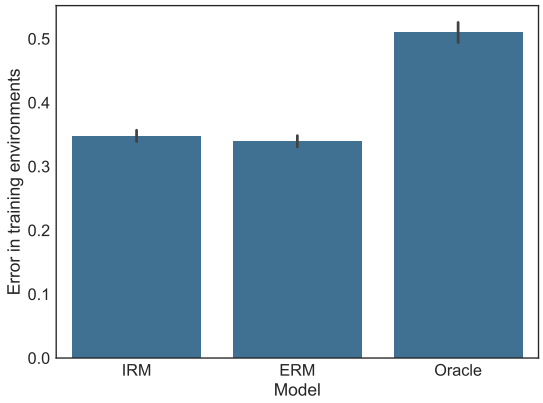


Figure A.1: *IRM has a worse or indifferent performance when we compare it to the ERM case for a unseen set that respects the i.i.d assumption, which means the data comes from environments that were observed in the training examples. At the same time, the Oracle model does not perform as well, since it is not exploring spurious correlations from the particular environment present in the train and random test data.*

However, the environment not seen in the train present different spurious correlations than the ones observed before, thus it is impossible to explore them. In this case, the performance ordering is reversed, as seen in the Figure A.2. Now the challenger model is better than the benchmark, while both are worse than the Oracle, which is expected.

Looking to the same results in a different perspective by grouping the data by environment, we can see in the Figure A.3 how the same ordering in the aggregated results is observed in all training environments and all unseen ones.

The hypothesis to argue that IRM is a better approach to general Machine Learning problems is that it is expected in most problems to face many interventions and environments after the learning stage.

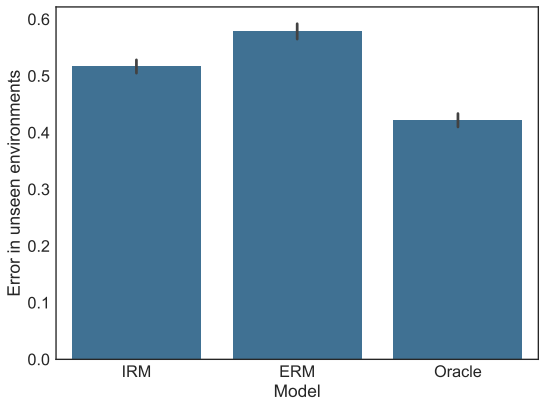


Figure A.2: *When we exposed IRM and ERM to a data generated by the same process than the training examples, but under a different environment, the ERM pays the price of exploiting spurious correlations and shows the highest error, while IRM has a better performance. The Oracle, as expected, performs well, since it follows the true causal relationships.*

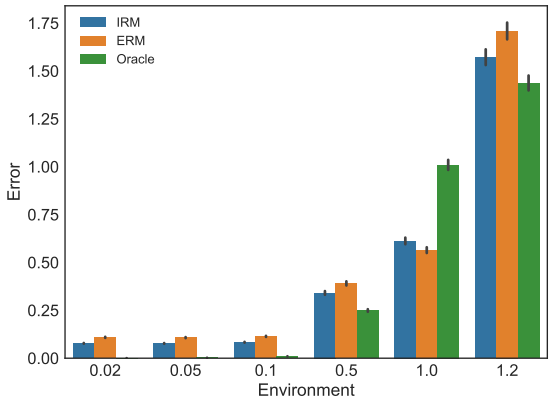


Figure A.3: *When we split the performance plot by environments, we can clearly see how IRM does better than ERM in unseen environments, which means domain generalization.*

Bibliography

[A+95] John Aldrich et al. Correlations genuine and spurious in pearson and yule. *Statistical science*, 10(4):364–376, 1995. 10

[ABGLP19] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2019. 1, 10, 12, 15, 16, 17, 20, 21, 32, 64

[Ada82] Robert M Adams. Theories of contingency. *Leibniz: Critical and Interpretive Essays, edited by Michael Hooker*, pages 243–283, 1982. 11

[Ali20] Moez Ali. *PyCaret: An open source, low-code machine learning library in Python*, July 2020. PyCaret version 2.3. 36

[Arj21] Martin Arjovsky. Out of distribution generalization in machine learning. *arXiv preprint arXiv:2103.02667*, 2021. 1

[ATSL10] André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 2010. 10, 39

[Bag05] J Andrew Bagnell. Robust supervised learning. In *AAAI*, pages 714–719, 2005. 17, 22

[BFSO84] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984. 2, 6

[Bis06] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. 1

[BMA+19] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32:9453–9463, 2019. 17

[BP14] Elias Bareinboim and Judea Pearl. Transportability from multiple environments with limited experiments: Completeness results. *Advances in neural information processing systems*, 27:280–288, 2014. 22

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 7, 28

[Car03] Nancy Cartwright. Two theorems on invariance and causality. *Philosophy of Science*, 70(1):203–224, 2003. 1

[Dao21] Jack Daoud. Animal shelter dataset. Version 1. Retrieved March 13, 2021 from https://www.kaggle.com/jackdaoud/animal-shelter-analytics, 2021. 35

[DHM+20] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghassen Jerfel, Alan Karthikesalingam, Mario Lucic, Yian Ma, Cory McLean, Diana Mincu, Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Natarajan, Christopher Nielson, Thomas F. Osborne,

Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Taedong Yun, Xiaohua Zhai, and D. Sculley. Underspecification presents challenges for credibility in modern machine learning. *CoRR*, 2020. 1, 4, 17

[GB20] Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *arXiv preprint arXiv:2011.15091*, 2020. 60

[GBR+12] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012. 60

[GLH+19] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019. 22

[GLP20] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020. 1

[GUA+16] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016. 60

[HD19] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019. 17

[HDPM17] Christina Heinze-Deml, Jonas Peters, and Nicolai Meinshausen. Invariant causal prediction for nonlinear models, 2017. 10, 19

[HNSS18] Weihua Hu, Gang Niu, Issei Sato, and Masashi Sugiyama. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning*, pages 2029–2037. PMLR, 2018. 26

[HR20] Miguel A Hernán and James M Robins. Causal inference: what if. *Boca Raton: Chapman & Hill/CRC*, 2020, 2020. 14

[HY01] David J Hand and Keming Yu. Idiot's bayes—not so stupid after all? *International statistical review*, 69(3):385–398, 2001. 5

[K+95] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995. 4

[KBG+19] Nan Rosemary Ke, Olexa Bilaniuk, Anirudh Goyal, Stefan Bauer, Hugo Larochelle, Bernhard Schölkopf, Michael C Mozer, Chris Pal, and Yoshua Bengio. Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*, 2019. 22

[KH01] Kamran Karimi and Howard John Hamilton. *Temporal Rules and Temporal Decision Trees: A C4. 5 Approach.* Department of Computer Science, University of Regina Regina, Saskatchewan . . . , 2001. 23

[KH10] Kamran Karimi and Howard J Hamilton. Generation and interpretation of temporal decision rules. *arXiv preprint arXiv:1004.3334*, 2010. 23

[KMF+17] 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. *Advances in neural information processing systems*, 30:3146–3154, 2017. 36

[KPS18] Niki Kilbertus, Giambattista Parascandolo, and Bernhard Schölkopf. Generalization in anti-causal learning. *arXiv preprint arXiv:1812.00524*, 2018. 15

[Lar31] Selmer C Larson. The shrinkage of the coefficient of multiple correlation. *Journal of Educational Psychology*, 22(1):45, 1931. 4

[LF50] Esmond R Long and Shirley H Ferebee. A controlled investigation of streptomycin treatment in pulmonary tuberculosis. *Public Health Reports (1896-1970)*, pages 1421–1451, 1950. 14

[LLD+18] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, page 1–1, 2018. 1

[LPR+20] Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-supervised disentanglement without compromises. In *International Conference on Machine Learning*, pages 6348–6359. PMLR, 2020. 22

[LPWK18] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018. 60, 61

[M+97] Tom M Mitchell et al. Machine learning. 1997. 35

[Mon20] Luis Moneda. Globo esporte news dataset. Version 11. Retrieved March 31, 2021 from https://www.kaggle.com/lgmoneda/ge-soccer-clubs-news, 2020. 35

[Mon21] Luis Moneda. Altitude and temperature in brazil 2020 dataset. Version 1. Retrieved February 19, 2021 from https://www.kaggle.com/lgmoneda/temperature-and-altitude, 2021. 14

[Mou18] Mickael Mouillé. Kickstarter projects dataset. Version 7. Retrieved March 13, 2021 from https://www.kaggle.com/kemical/kickstarter-projects?select=ks-projects-201612.csv, 2018. 35

[MPJ+16] Joris M Mooij, Jonas Peters, Dominik Janzing, Jakob Zscheischler, and Bernhard Schölkopf. Distinguishing cause from effect using observational data: methods and benchmarks. *The Journal of Machine Learning Research*, 17(1):1103–1204, 2016. 1, 13

[MT68] Frederick Mosteller and John W Tukey. Data analysis, including statistics. *Handbook of social psychology*, 2:80–203, 1968. 4

[oC21] City of Chicago. Chicago crime - bigquery dataset. Version 1. Retrieved March 13, 2021 from https://www.kaggle.com/chicago/chicago-crime, 2021. 36

[PBM15] Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference using invariant prediction: identification and confidence intervals. *arXiv preprint arXiv:1501.01332*, 2015. 1, 17, 18

[Pea97] Karl Pearson. On a form of spurious correlation which may arise when indices are useed in the measurement of organs. In *Royal Soc., London, Proc.*, volume 60, pages 489–502, 1897. 1, 10

[Pea09] Judea Pearl. *Causality*. Cambridge University Press, Cambridge, UK, 2 edition, 2009. 1, 9, 10, 11

[PJS17] Jonas Peters, Dominik Janzing, and Bernhard Schlkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017. 1, 12

[PM18] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018. 11

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. 33, 36

[QCSSL09] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009. 1, 5

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986. 6, 7

[Qui14] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014. 6

[Rea18] Jesse Read. Concept-drifting data streams are time series; the case for continuous adaptation. *arXiv preprint arXiv:1810.02266*, 2018. 1

[RGL18] Stephan Rabanser, Stephan Günnemann, and Zachary C. Lipton. Failing loudly: An empirical study of methods for detecting dataset shift, 2018. 6

[RHW85] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 60

[RR56] Hans Reichenbach and Maria Reichenbach. *The direction of time, ed.* 1956. 10, 11

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 1

[Sha48] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948. 7

[Sha18] Aparna Shastry. San francisco building permits dataset. Version 1. Retrieved March 13, 2021 from https://www.kaggle.com/aparnashastry/building-permit-applications-data, 2018. 36

[Shi00] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000. 5

[Sim77] Herbert A Simon. Causal ordering and identifiability. In *Models of Discovery*, pages 53–80. Springer, 1977. 11

[Sio19] Andre Sionek. Brazilian e-commerce public dataset by olist. Version 7. Retrieved March 13, 2021 from https://www.kaggle.com/olistbr/brazilian-ecommerce, 2019. 36

[SJP+12] Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. *arXiv preprint arXiv:1206.6471*, 2012. 12

[SK19] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019. 17

[SS18] Adarsh Subbaswamy and Suchi Saria. Counterfactual normalization: Proactively addressing dataset shift and improving reliability using causal mechanisms. *arXiv preprint arXiv:1808.03253*, 2018. 1

[SSN+08] Masashi Sugiyama, Taiji Suzuki, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008. 5

[Sto74] Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974. 4

[Sto09] Amos Storkey. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, pages 3–28, 2009. 5

[Vap13] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013. 3

[VP91] Thomas Verma and Judea Pearl. *Equivalence and synthesis of causal models*. UCLA, Computer Science Department, 1991. 10

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 22

[WA18] Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018. 14

[Wal45] Abraham Wald. Statistical decision functions which minimize the maximum risk. *Annals of Mathematics*, pages 265–280, 1945. 15

[WM+95] David H Wolpert, William G Macready, et al. No free lunch theorems for search. Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995. 4

[Wri18] Sewall Wright. On the nature of size factors. *Genetics*, 3(4):367–374, 1918. 9

[WRS+17] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292*, 2017. 1, 16

[Yul26] G Udny Yule. Why do we sometimes get nonsense-correlations between time-series?–a study in sampling and the nature of time-series. *Journal of the royal statistical society*, 89(1):1–63, 1926. 10

[ZSMW13] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, pages 819–827. PMLR, 2013. 5, 15