Title

Audio Scene Monitoring Using Redundant Ad Hoc Microphone Array Networks

Journal

IEEE Internet of Things Journal, 9(6)

ISSN

2372-2541

Authors

Gerstoft, Peter
Hu, Yihan
Bianco, Michael J
et al.

Publication Date

2022-01-15

DOI

10.1109/jiot.2021.3103523

Peer reviewed

Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

Two observations on the train/test methodology Draft, do not distribute.

Julaiti Arapat and Yoav Freund
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093, USA

Editor:

Abstract

We identify some problems with training classifiers for diverse datasets, and propose some remedies.

Keywords: Bathymetry, Active Learning, Bootstrap

1. Introduction

One of the basic assumptions in machine learning is that data is generated independently at random from a fixed but unknown distribution. We refer to this as the IID ("independently and identically distributed") assumption. The IID assumption justifies the common practice of splitting a dataset at random into a training set (RTrain) and a test set (RTest), we train the model on RTrain, and report its performance on RTest.

Reporting the test error on RTest is generally accepted as a reliable criteria for comparing the accuracy of different models. However, in some real world applications, performance on RTest is sometimes a poor predictor of the performance on a new test sample (NTest), which is collected in a related, but different environment. We identify two reason for the difference:

- **Domain shift:** In some applications of machine learning, RTest and NTest are drawn according to different distributions. For example, suppose we train a classifier to classify people from a photo of their faces. Suppose a large dataset is collected and split into RTrain and RTest, suppose further that the performance on RTest is good. The performance on a new dataset, taken several months after training of the classifier, might be significantly poorer. It may due to the changes of haircut, the changes of lighting, or capturing by a different camera. This problem is often referred to as "domain shift" and the solutions for it are called "transfer learning". Ng (2016) declared transfer learning as an important open problem.

- **Spurious correlations:** Datasets sometimes contain spurious correlations which appear in RTrain and RTest but do not carry over to NTest.

One of the conceptual foundations of machine learning is the randomized train-set/test-set methodology (RTTM). This methodology dictates a randomized method for creating

training testing sets. A large dataset is collected and then partitioned randomly into a training set and a test set. The classifier is trained on the training set and then tested on the test set. The main contribution of this paper is the identification of a fundamental problem with RTTM and the proposal of an approach for addressing this problem.

Most of the theory of machine learning is based on RTTM. Specifically, we assume that the training set and the test set are both drawn IID from the same distribution. Under this assumption the train error and the test error of a classifier are unbiased estimates of the same quantity: the generalization error.

However, in practice often one cannot justify this assumption. To make this concrete, consider a traffic sign classifier that is to be used as part of a self-driving car. Suppose we train our classifier on data that was collected in a hundred different locations. How should we collect a test set? If we collect the test set video from the same one hundred location, then our accuracy guarantees apply only to the same one hundred locations. If we want our guarantees to apply to other locations, we must use data collected from those locations. In this case the training set and the test set are likely to have different distribution. Making most theoretical results inapplicable. We call this the *data divesity problem*.

There is another problem with RTTM, which relates to the fact that videos consist of frames, and neighboring frames are usually very similar to each other. Suppose that we have a one second stretch of a video in which a stop sign appears in view. The thirty or so frames corresponding to that view are likely to be very similar to each other. Now suppose 15 of these frames appear in the training set and the other 15 in the test set. The images in the training set would be labeled as "containing stop sign". As the images in the test set are very similar, simple memorization based methods, such as k-nearesst-neighbors, are likely to perform well on this task. We call this problem the *data sequentiality problem*

In this paper we demonstrate the data sequentiality and the data diversity problem in a particular large scale study. We then describe a heuristic we used that can reduce the data diversity problem.

2. The standard framework of machine learning

The standard framework of machine learning is as follows.

Let $X \in \mathcal{X}$ be the input features, and $y \in \mathcal{Y}$ be the labels. The data instance (X, y) are independent and identically distributed (i.i.d.) random variables from a fixed but unknown joint distribution $\mathcal{D}$ over $X \times \mathcal{Y}$. Let $h : \mathcal{X} \to \mathcal{Y}$ be the candidate hypothesis from some hypothesis set $\mathcal{H}$. The objective of learning is to find the optimal hypothesis h^* that minimizes the *true* error:

$$\mathrm{err}_{\mathcal{D}}(h) \doteq P_{(X,y)\sim\mathcal{D}}\left[L(y, h(X))\right], \tag{1}$$

where $L(y, h(X))$ is the loss function used to measure the difference between the the true label y and the predicted label $h(X)$.

The learning algorithm is given two sets of i.i.d. data instances, a training set $\mathcal{T} = \{(X_1, y_1), \ldots, (X_n, y_n)\}$ and a test set $\mathcal{U} = \{(X_1, y_1), \ldots, (X_m, y_m)\}$. The objective of the learning algorithm is to find the hypothesis $\widehat{h}$ that minimizes the *training error*,

$$\widehat{\mathrm{err}}(h; \mathcal{T}) \doteq \frac{1}{n}\sum_{i=1}^{n} L(y_i, h(X_i)), \qquad (X_i, y_i) \in \mathcal{T} \tag{2}$$

If the training set size is small and/or the hypothesis set size is large, the algorithm may find a hypothesis that *overfits* the training set, in which case the true error of $\widehat{h}$ is likely to be significantly larger than the true error of h^*. To estimate the true error of $\widehat{h}$, the algorithm calculates the *generalization error* $\widehat{\mathrm{err}}(\widehat{h};\mathcal{U})$ on the test set (also referred to as the *test error*). Under the standard framework, the test error is an unbiased estimate of the true error and can be used to compare different learning algorithms.

In this paper, we discuss the workflow of a machine learning project in two phases: the training phase and the production phase. The training phase includes training the model on $\mathcal{T}$ and evaluating its performance on $\mathcal{U}$. The production phase starts after the model is deployed into production. The error rate in the production phase is evaluated on the newly collected data instances that are unseen in the training phase. In the light of the i.i.d. assumption, the error rate in production is expected to be close to the test error.

Randomized train/test split The *de facto* standard for generating the training set $\mathcal{T}$ and test set $\mathcal{U}$ is the randomized train/test split (RTTS) method. Starting with a sufficiently large dataset, one randomly divides the dataset into two to three parts: a training set $\mathcal{T}$, a test set $\mathcal{U}$, and sometimes a validation set $\mathcal{V}$. There is no agreed upon way for selecting the relative proportion of the three splits. Hastie et al. (2009) suggests reserving 50% of the dataset for $\mathcal{T}$, and 25% each for $\mathcal{V}$ and $\mathcal{U}$. Note that for the sake of simplicity, we will assume only a training set and a test set and no validation set in this paper.

The standard framework is at the heart of machine learning theory and practice. The assumption on how one collects the data instances (i.e. the i.i.d. assumption) is usually taken for granted. However, there are at least two ways in which practical data collection fails this assumption: the data diversity problem and the spurious correlation problem. In next two sections, we first explain these two problems with examples, then discuss them in the context of experimental design in machine learning.

3. Two problems of the standard framework

In this section we describe two ways in which common data collection procedures are inconsistent with the standard framework. The standard framework assumes that the examples are *independent and identically distributed* (i.i.d.). These inconsistencies, in turn, create biases in the error estimates used in RTTS. We present the inconsistencies in the two following subsections: in Section 3.1 we describe the problems in the *identically distributed* assumption; in Section 3.2 we describe the problems with the *independently distributed* assumption.

3.1 Data diversity problem

The purpose of the test set in RTTS is to provide an estimate of the error rate of the learned model when deployed "in the wild". For this estimate to be unbiased, the test set should be *representative* of the "in-the-wild" distribution. However, collecting a representative set of examples is often hard, as described below.

A data-set is representative if it has the same distribution as that of data collected "in the wild". When the data is low dimensional, it is easy to ensure that a dataset is representative. In practice, many applications are based on very high dimensional data,

Figure 1: The imaginary from the highway Camera mounted over California State Route 1, randomly sampled from a continuous 2-minute video clip.

such as images or video. The diversity of inputs is so large that collecting a *representative sample* is difficult.

Consider collecting a dataset for training an autonomous driver. The variables in the context of driving include: geographical locations[1], driving conditions[2], traffic conditions[3], different time of the day[4], seasonal and weather conditions[5]. Ideally, we assume the dataset should be both sufficiently large and sufficiently diverse so that it covers the various *combinations* of the variables above. The diversity of the data instances depends on what data sources are considered in the data collection step.

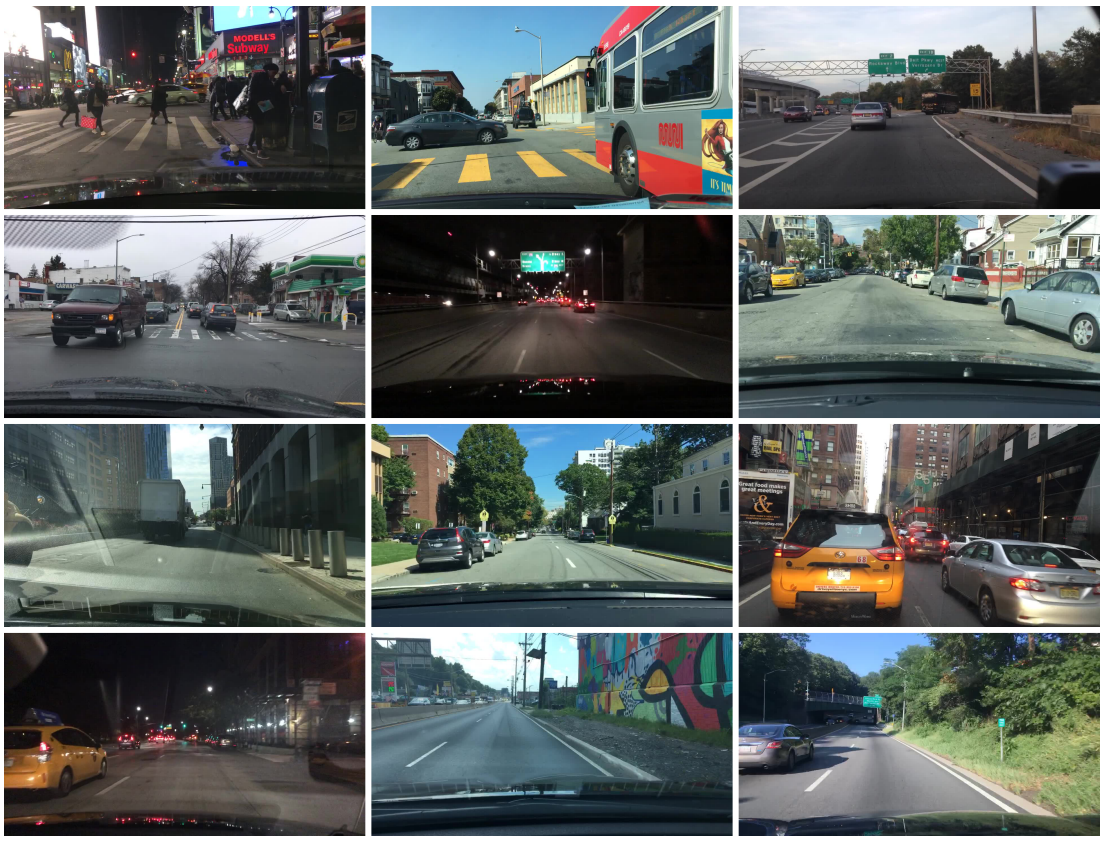
Contrast the image samples of the video streams from three cameras on the road. The stream A (Figure 1) is generated by a fixed camera monitoring a segment of California State Route 1. The stream B (Figure 2) is generated by the Google Street View project (**?**), from which the images are sampled as follows: select a geographic coordinate in United States uniformly at random, collect the imagery of Google Street View at that location if it exists, otherwise select another random coordinate. The stream C (Figure 3) is generated by the video cameras mounted on the cars from the San Francisco Bay Area and New York City (Yu et al. (2018)).

In the stream A, the diversity is limited, only a small part of the frame changes from moment to moment, such as the cars driving in the lanes. In the stream B, the diversity is higher, as we start to explore different *types* of roads located at different parts of the

1. For example, different states, cities, towns.
2. For example, highways, residential areas, primitive roads.
3. For example, peak hours and off-peak hours, with and without the presence of pedestrians.
4. For example, dawn, daytime, dusk, and nighttime.
5. For example, spring, winter, rainy days, snowy days.

Figure 2: The images sampled from Google Street View by repeatedly selecting a geographic coordinates in United States until there is an image taken at that location by Google Street View.

Figure 3: The samples from BDD100K selected uniformly at random.



country. We also observe the differences in weather conditions. However, these roads are mostly empty, likely located in the less populated areas. In the stream C, the diversity is very high. The driving conditions are complicated with the presence of other vehicles and pedestrians. The images also cover different time of the day.

Along with many other applications of machine learning, the driving datasets are drawn from a *long tail* distribution. It requires careful experimental design to collect a diverse dataset. For example, if the learning task involves detecting traffic signs, the data instances solely from the stream A or the stream B are not diverse enough because there are few images that contain the traffic signs mostly seen on the city streets. On the other hand, if the learning task involves detecting lanes or the edges of the roads, the data instances solely from the stream A or the stream C are not diverse enough because there are few images with the primitive roads and the roads located in rural areas.

Another challenge in these applications is that it is hard to acquire samples that represent the further end of the long tail. In the context of driving, it may be very *unusual* situations, such as the extreme weathers (e.g. heavy fog) and unexpected pedestrian behaviors (e.g. pedestrian walking on the highway).

As we see in these examples, if the training data is not diverse enough, then simply increasing the size of the training data will not fundamentally improve the model performance in the production phase. It is because the model has little prediction power over the data instances generated by the data sources that are drastically different from those seen in training.

In the standard framework, the data diversity problem is not a concern because the data instances are assumed to be drawn identically from a fixed distribution. In practice, the data instances are often drawn from multiple data sources each potentially with different distributions. The RTTS method addresses the presence of multiple data sources by shuffling the data instances before splitting them into the training set $\mathcal{T}$ and the test set $\mathcal{U}$, so that the distributions that generate $\mathcal{T}$ and $\mathcal{U}$ are identical. However, the data shuffling does not effectively improve the data diversity. Thus, the model trained on $\mathcal{T}$ cannot predict well for the data instances from previously unseen data sources. Consequently, the generalization error evaluated on $\mathcal{U}$ is no longer a reliable estimate for the model performance in the production phase, which is a more critical measurement for most practical machine learning projects. Instead, we will see a change in the distributions from the training set to the data instances collected in the production phase. This phenomena is referred as *distribution shift*. Many algorithmic solutions are proposed to mitigate distribution shift (see Kouw and Loog (2018)).

The data diversity problem is not unique to machine learning. It is generally observed in the scientific studies that use experiments and data to validate theories. **?** examines machine learning in the lens of experimental science. By definition, an experiment involves observing the behavior of a system while varying one or more of its components and keeping others constant. The methodology enables us to study the interactions between the variables in a system. Designing controlled experiments is straightforward with a few independent variables. However, in practical problems, controlling all variables are infeasible. As a result, the recommended method is to collect observations from each cell in experimental design, and justify the differences of the results between cells using the data. In the context

of machine learning, it means one should investigate a diverse assembly of data sources that are systematically selected.

The diversity in data sources is presence in many datasets in which the instances could be partitioned into subpopulations. For instance, consider the following two examples.

- **Medical image segmentation** Data diversity is a major challenge in deploying learning-based model in medical image segmentation to new environments (de Bruijne (2016); Van Opbroek et al. (2014)). The model performance is promising in a controlled, standardized environment. However, the accuracy decreases when the model is deployed in a new environment (e.g. a new hospital), due to the variance in imaging protocol, scanner model, or different patient population. In this example, we can see that the data distribution may change from one location to another.

- **Advertising clicks prediction** Predicting advertising clicks is central to most online advertising system. He et al. (2014a) has observed in practice that the prediction accuracy of the model decreases over time. The decreases is caused by the change in the behaviors of the advertisers and the website users over time. Such changes cannot be captured by a fixed model trained on historical data. In this example, the data instances are from a non-stationary distribution. The distribution changes from one period of time to the next.

3.2 Spurious correlation problem

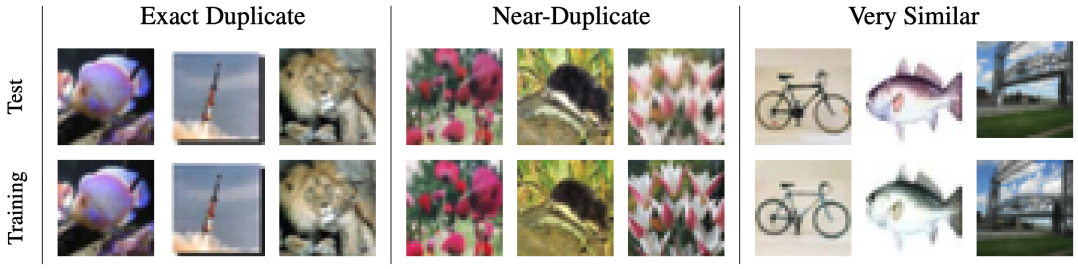
The power of machine learning methods is in their exceptional capability of discovering the statistical correlations between the data features and the data labels. In many cases, there is no cause-effect relationship between the two variables that are correlated. This type of correlations is referred as *spurious correlations* in statistics (**?**). The spurious correlations exist when there is a third random variable Z that is a common cause of the two random variables that are correlated. The common cause Z is also referred as the *confounding factor*.

An assumption in machine learning is that if the training set is sufficiently large, the correlations discovered in the training set are likely to be persist over the population. We argue that this conclusion is not guaranteed in practice because of the mutual dependencies between the training instances.

Under the standard framework, we assume the data instances in the training set are mutually independent. In practice, this assumption is difficult to satisfy for many reasons. One of them is that these data instances are collected by the same mechanism, meaning that they are from the same data sources, collected in some fixed order, or other reasons that could introduce a common causal factor between the data features and the data labels. If the common cause factor only associates with the data collection process but does not exist across the population in general, the correlations due to these causal factors are unlikely to exist beyond the training set.

Data duplications and near-duplications are among the most straightforward examples of spurious correlation in the dataset. For instance, **?** finds that 10% of the images from the CIFAR-100 test sets have duplicates in the training set, while the ratio is 3.3% for CIFAR-10

Figure 4: The example is from **?**. Samples of different types of duplicates discovered in CIFAR-100 test and train sets. The top row shows images from the test set. The bottom row shows the corresponding similar images in the training set. Exact duplicate images are identical almost pixel-by-pixel. Near-duplicates are same images that post-processed differently. Very similar images contain different contact but look highly similar.



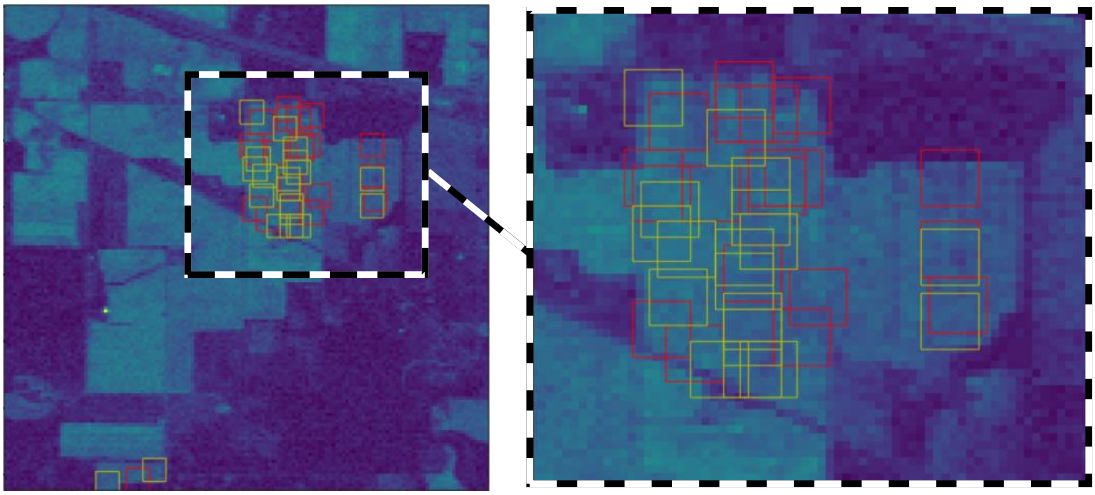
(see Figure 4). In ImageNet (**?**), 1.78% of the validation images have near-duplicate images in the training set (**?**), even though the benchmark dataset has gone through de-duplication.

These large-scale image datasets are collected by crawling images from popular image search engines, such as Google and Flickr, using a set list of keywords assembled from sources like WordNet (**?**). Unavoidably, some of the search query results contain duplicate images for reasons like same images being uploaded by more than one user or being used on more than one webpages. De-duplicating is not trivial in high dimension. Two data instances may not be exact, byte-by-byte match to each other, but still be very similar to each other before or after the feature extraction. As a result, some duplicate instances end up in these datasets.

Not all data dependencies are in the form of data duplication. Nalepa et al. (2019)) studies the spurious correlation between the training set and the test set for the image segmentation algorithms on the satellite data. As it shows in Figure 5, the training set and the test set are generated using the RTTS method over the random selections drawn from a fixed region on map. Some of these selections could have overlapping pixels, which essentially gives away the labeling of the overlapped region for the corresponding selections appeared in the test set. As we will see in Section 6, in some cases the selections does not even need to be overlaps, but merely to be close enough in distance, to introduce spurious correlations.

Another source of the spurious correlation is the common underlying properties that should not be considered in the prediction rules because they are inaccessible at the time of inference. For example, it is usually required to create both the temporal boundary and the entity-level boundary when working with the financial datasets. For example, when training a classifier for fraud detection using a dataset of credit card transactions, the data instances should be split into the training set and the test set in a way such that (1) all transactions in the training set happened before all transactions in the test set, and (2) the transactions belong to the same account holder should appear in either the training set or the test set but not both. The first requirement is to eliminate the mutual dependencies between the consecutive transactions. For example, a malicious party may attempt to put several purchases consecutively on the stolen credit cards before the accounts get frozen. The

Figure 5: The example is from Nalepa et al. (2019). The left figure shows the map from Indian Pines, while the right figure is zoom into the region of interest. Random selections are drawn from the region of interest, and then split into the training set and the test set, denoted by the yellow are red squares, respectively.



second requirement is to eliminate the mutual dependencies between the transactions due to the economic patterns of a specific account holder. For example, some credit card owners may have never experienced fraudulent charges. The model may learn specific patterns that could identify such owners, and predict all future transactions from their cards as normal.

In all the examples that have spurious correlation, we could use simple memorization-based method and perform well on the test instances that have duplicates in the training set. However, these prediction rules cannot be generalized to the population, thus have little utility in application.

In the context of predictive modeling, spurious correlation is a type of information leakage. Rosset et al. (2010) describes the information leakage as the "unintentional introduction" of the predictive information. Similar to other forms of information leakage, spurious correlation leads to an over-estimation of the model performance based on the performance on the test set. It is especially problematic when the learning model has a strong "memorization" capacity. For example, **?** finds that, after removing the duplicate and near-duplicate images, the error rate of the state-of-the-art model increases from 3.56% to 3.95% on CIFAR-10, and increases from 17.05% to 19.38% on CIFAR-100.

4. Experimental design in machine learning

The standard framework described in Section 2 deviates from the real-world machine learning projects on how the data instances are collected. In this section, we discuss an alternative framework that better fits the actual data collection process. We are not attempting to propose the resolutions for the data diversity problem and the spurious correlation problem, but to explore possible future work directions.

4.1 Collect data from environments

Under the standard framework, all data instances are drawn from a fixed distribution. However, as showed in the examples described in this paper, the data distributions are often not identical in training and in testing, because the data sources have changed.

Alternatively, we should treat each data source as a distinct *environment* from which we can draw data instances. Each environment e is governed by its own distributions $\mathcal{D}_e$ over (X, y). As a result, when we switch the environment from training to testing, we could see a distribution shift. In the driving example, the camera mounted over the vehicle could capture images from multiple environments, e.g. driving in the cities and driving in the rural areas.

Informally, we assume the data is generated by a two-step hierarchical model (should it be "stochastic process"?): first an environment e is drawn, then the data instances are drawn from its corresponding distribution $\mathcal{D}_e$. Julaiti comments:
what comments can be stated in the relationship between the different distributions?

In practice, we have no control over the data sources that would be used in the production time. In other words, the testing environment of a model is likely to be different from its training environment. This is very different from the assumption in the standard framework, where the training environment and the testing environment is identical statistically. In the design of the machine learning experiments, we should reflect this challenge.

4.2 Experimental design in machine learning

Machine learning is an experimental science (**?**). By definition, an experiment involves observing the behavior of a system while varying one or more of its components and keeping others constant. The methodology enables us to study the interactions between different variables.

Comparing to other fields of science, it is easier to design control experiments in machine learning because the practitioners have complete control over the data sources (while in training) and the learning algorithms.

A basic controlled experiment can be designed as follows (Table **??**). Given a set of k data sources, we have k separate environments $E = \{e_1, e_2, \ldots, e_k\}$ by treating the instances from each data sources as from separate environments. Collect separate pairs of the training set $\mathcal{T}_i$ and the test set $\mathcal{U}_i$ from each environment e_i. Then with each training set $\mathcal{T}_i$, we can train a model h_i. Finally for each trained model h_i, we can test its performance on all environments using all the tests set $\mathcal{U}_j, j = 1, \ldots, k$.

Generally, we expect to see that the model h_i performs best in the environment e_i and vice versa. More interesting results, however, are its performance on all other environments. The justification is to by using examples from separate environments, we are likely to better estimate the model performance in a non-identical environment, and also reduce or even eliminate the boost in model performance due to the spurious correlation embedded in the model.

4.3 Model ensemble and active learning

Under the standard framework, given k data sources, we collect examples from all of them and generate one training set and one test set using the RTTS method. If, instead, we follow the proposal in Section 4.2, we can create a model ensemble with k distinct models. We think the ensemble of k model is a better fit for the practical learning tasks because it enables us to measure the prediction confidence.

Evaluating the prediction confidence is more feasible with an ensemble than with a single model. For example, in a binary classification task with 0/1 label, we would be very confident in the model prediction if all k models predict the same, while we would be uncertain if $k/2$ of the models predict 0, and the other $k/2$ predict 1. When the model is uncertain about its prediction, it can abstain from prediction, which is a safer approach for many applications where the cost incorrect inference is high.

Delegate when uncertain In many high-stake applications of machine learning, such as autonomous driving and medical diagnoses, it is an over-reaching goal to replace the human experts entirely with computers. Instead, we should treat computers as an enhancement to human intelligent. Specifically, the models should make prediction only if it has a high confidence. For complicated examples on which the models are uncertain, it should print "I don't know" and delegate the prediction task to a human expert. The hybrid approach could reduce the risk of making high-cost mistakes, while still benefits from the automation.

Active learning The key idea of active learning is that the model can make improvement at lower cost by selectively choosing which data instances to use for training. It is best suited in the scenarios where it is cheap to acquire a large amount of data instances, but expensive to label them. It is suited in the context of ensemble learning for the following reasons. The learning models are expected to perform well on "easy examples". One way to define easiness of an example is by how likely the similar examples may be appear in the training set. The models are likely to perform well on the most commonly seen examples, but less confident on the rarely seen ones. Many learning tasks work with a long tail distribution, which means a vast pool of the data instances may not be collected in training but appear in testing. For example, it is rare but possible that pedestrians may be walking on the highway. In the data collection for model training, it is unlikely that such scenarios would be captured. But when the model is deployed at scale, the probability of encountering these rare cases is significant. These instances can be labeled by human experts, and later be added to the training set for improving the model.

5. Related work

5.1 Randomized train/test split

The Randomized train/test split (RTTS) method is used widely across the academic research and the real-world applications of machine learning. In the academic community, we rely on the performance reported on the test set generated by RTTS to decide which algorithms/architectures are the best. On the application side, RTTS is a default step in the data preprocessing, as seen in the popular machine learning packages such as scikit-learn (Pedregosa et al. (2011)) and TensorFlow (Abadi et al. (2016)).

The RTTS method is justified only if the data is drawn under the i.i.d. assumption. Darrell et al. (2015) identifies three reasons why the i.i.d. assumption could be violated in practical applications:

1. the distribution can be non-stationary and change over time;
2. consecutive data samples can be inter-dependent;
3. the data samples are generated by an adversary, instead of a distribution.

5.2 Learning from multiple environments

We use the term *environment* to recognize that the data instances may be drawn from separate, distinct data sources. This concept has been previously studied in Arjovsky et al. (2019), Peters et al. (2016), and Heinze-Deml and Meinshausen (2017), which aim to identify the invariant features or rules across different environments, and to show that the invariants are related to the causal explanation of the learning objectives. The end product of these work is a single model that generalizes well in all varieties of the environment. We propose a different learning objective in Section 4, which uses an ensemble approach, training multiple models that fit best in each environment.

5.3 Distribution shift

In practice, the training environment is often different from the testing environment. As a result, machine learning practitioners often observe the statistical difference at various scale between the environment that the model is trained, and the environment that the model is to be applied, i.e. a distribution shift.

Distribution shift, also been referred as data shift, is a general term to describe the discrepancy between the two distributions $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{T}}$: $\mathcal{D}_{\mathcal{S}}$ generates the examples for training the model, and $\mathcal{D}_{\mathcal{T}}$ generates the examples that we want to make predictions on. The discrepancy causes the model to make bias estimations on the examples from $\mathcal{D}_{\mathcal{T}}$. There are a number of approaches to address the distribution shift.

Transfer learning and domain adaptation In transfer learning and domain adaptation, the training data is sampled from the source domain, while the fresh examples are sampled from the target domain. The data distribution shifts from one domain to the other. Kouw and Loog (2018) describes the three types of data shift that happens, namely priority shift, covariate shift, and concept shift. Except for covariate shift, data labels are required to fit the model to the new domain.

Exploit invariance and causality Heinze-Deml and Meinshausen (2017); Arjovsky et al. (2019) proposed methods that prompt the invariant part (or core features) of the data in the model training process. The term *environment* is used to describe the distributions where the data can be sampled from. The idea is that by focusing on the data properties that persist across all environments, the model can fit the invariant casual associations between the variables, and generalize better to the fresh examples from an unseen distribution that shares these properties. For example, Arjovsky et al. (2019) introduce a regularizer to penalize the feature representations that cannot generalize well in different environments.

Periodic retraining Perhaps more popular in the applied side of machine learning, periodic retraining is a common routine to combat the distribution shift over time. He et al. (2014b) uses the term *data freshness* to describe the delay between the dates that the training and testing data sets are collected. They observe that the prediction accuracy of the model decreases as the delay between the training and testing increases. Thus to improve the prediction accuracy, they retrain the model on a daily basis so that the delay between the training and testing is as small as possible.

5.4 Spurious correlation and information leakage

In the context of predictive modeling, spurious correlation is a type of information leakage. Kaufman et al. (2012) argues that there are two causes of information leakage: leakage in features and leakage in training examples. Spurious correlation is a type of leakage in training examples.

To the best of our knowledge, there is no systematic method that can universally prevent the spurious correlation between the training and testing sets. In fact, even detecting the spurious correlation is a difficult task.

Rosset et al. (2010) proposes three general approaches for detecting spurious correlation.

Exploratory data analysis As a general data analysis approach, exploratory data analysis covers a wi de variety of statistical tools and techniques, and provides direct insights on the underlying structure of the data set. Many instances of information leakage, including those caused by the spurious correlation, can be identified by exploratory data analysis. We discuss a specific example of revealing the spurious correlation using a simple data visualization technique in Section 6.

Critical model evaluation In many real-world problems, the domain experts and the modelers often have a rough expectation for the model performance. When the model performance is too good to be true, it should raise concerns. In Section 6, we discuss how we saw stellar results on the prediction accuracy, which raised a red flag, and motivated us to discover the spurious correlation between the training and testing sets.

Exploration of usage scenarios Finally, one can break the fallacy of the overly optimistic testing performance by testing the models in the true application settings. In other words, the information leakage can be detected by using the fresh new data gathered after the model is trained.

Kaufman et al. (2012) proposes a sufficient condition for preventing information leakage in general. In short, the condition says that all training examples must be observable for the purpose of inferring all testing examples. More formally, they define a legitimate set $legitu$ of a random variable u. A second random variable v is said to be in $legit\{u\}$ if v is observable at the time of inferring u. Then, for all the feature and label pair $(\tilde{X}, \tilde{y})$ in the training set, and for all label y in the test set, the condition is defined as

$$\tilde{X} \in legit\{y\} \text{ and } \tilde{y} \in legit\{y\}.$$

However, this condition may be hard or expensive to validate on instance-by-instance basis for all training and testing examples pair. In addition, it requires a precise definition

for *observable* according to the application scenario. In some cases, *observable* is an ill-defined concept. For example, the data instances may be collected over time but the precise timestamp information is not available.

6. Case study: bathymetry data editing

Bathymetry is a study of the depths and shapes of underwater terrain. One of the key objectives in the bathymetry study is to produce a high-definition map of the global seafloor by aggregating the available data from multiple research institutions around the world. There is a small percentage of the bathymetry data that are corrupted, which should be eliminated from the final seafloor map. Therefore, it is critical to have a careful data editing step before merging the new data sources for eliminating the corrupted data. In this section, we describe how we use machine learning to assist the data editing process, and discuss the challenges we encountered as a case study for the data diversity problem and the spurious correlation.

6.1 Bathymetry data editing

The raw bathymetric data comes with three attributes: longitude, latitude, and the seafloor depth measurement at the specified location. In preprocessing step, we also have access to other meta data related to the location of the measurements as described in the next section.

The task of bathymetry data editing is modeled as a binary classification problem. For each measurement, we want to predict if the measurement is accurate. Prior to exploring the machine learning solution, a group of human data labelers are tasked with manually removing all incorrect measurements.

The bathymetry dataset has a clear hierarchical structure. The dataset is aggregated from multiple data sources (17 sources at the time of the writing). In addition, the data sources gather the measurements from multiple vessel cruises. Lastly, according to the measurement technology, we can further split the data instances into two classes: measured by a single-beam echo sounder or a multi-beam echo sounder. Alternatively, we can also stratify the data instances according to their geolocations, since the measurements cover all over the globe.

6.2 Data description

We use a bathymetry datasets from 7 different research institutions around the globe. The datasets cover different geographic areas. In addition, as listed in Table 1, they are different in data sizes, and have different ratios between the valid and corrupt measurements.

In addition to the measured seafloor depth and the location, i.e. the longitude and latitude coordinates of the measurement, we include 32 additional features to describe the region where the data is collected, for example, the medium seafloor depth of the region, the seafloor variance of the region, the year in which the data is collected. The data label is a binary bit, where 1 means the human labeler marked the measurement as corrupted or incorrect, and 1 means that the labeler marked the measurement as valid. The learned classifier should ensure a low false negative rate (so that as few valid measurements are

Table 1: Overview of the bathymetry editing dataset

Agency	Area	Edit out	num. of cruises	num. of measures	imbalance*
all	*	5.105	7881	287 M	0.93
AGSO	0.3527	1.961	127	15 M	0.48
JAMSTEC	1.5226	4.306	541	81 M	0.81
NGA	0.0609	19.340	1374	4 M	0.92
NGDC	4.4000	4.751	1033	116 M	0.69
NOAA geodas	1.04	11.063	4079	20 M	0.72
SIO	0.9577	13.137	247	21 M	0.76
US multi	*	5.301	480	30 M	0.73

Measured by the percentage of the number of measurements in the largest 20% of the cruises

removed as possible), while having close to zero false positive rate (so that the corrupt data would not be integrated into the final map).

6.3 Case study for data diversity: learning from multiple environment

Under the standard framework, we would merge the measurements from all data sources and generate a training set and a test set using the RTTS method. However, the trained model would perform poorly since the bathymetry data is not drawn i.i.d..

We can treat the data from different research institutions as they are from different data sources. As it shows in Table 1, there are significant differences statistically between the data sources. The distribution that generates the measurements on which we would like to apply the trained model may be very different from any distributions that we observed during training. As a result, we have no guarantee over the generalization performance on the trained model because of the data diversity.

To show the problem in action, we partitioned the bathymetry dataset by the data sources, and trained a model for each of the data sources separately. Then we reported the model performance on each of the data sources. The result is presented in Figure 6.

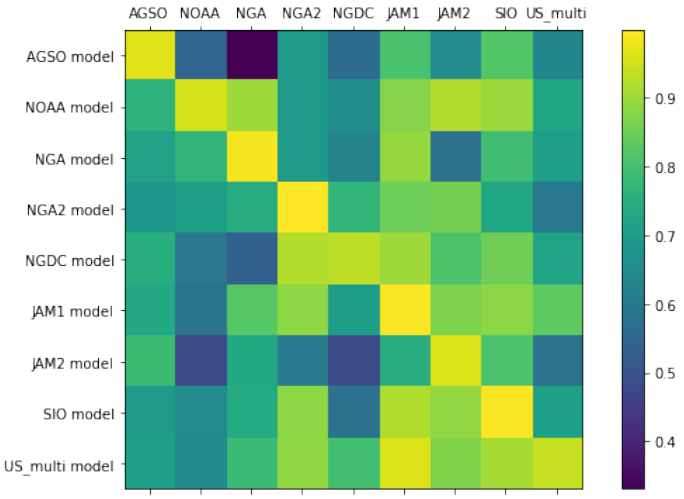
Identifying the coherent subsets of the training datasets can potentially improve the prediction accuracy in two ways. First, if we individually train a model for every data subset, we expect the model accuracy to improve since the data distribution is "easier" to learn (how?). Second, we can create an ensemble of the models trained on each coherent subsets, and get a better prediction accuracy than any single classifier (how?).

The general principal of identifying subsets within a dataset is to split data in the way that improves the model performance, and to stop further splitting when it causes overfitting.

The bathymetry datasets is aggregated from various research institutes around the globe. A natural assumption is that the examples collected from the same agencies are more likely to be coherent.

The data splitting is similar to the general cruise-based splitting, with the addition that we now create separate training and testing data sets for the data from each research institutes.

Figure 6: Model prediction accuracy cross different data sources



With every training set that contains examples from only one institution, we can train a model that is supposed to be performed best on the examples from the same coherent set. To further validate our assumption, we *cross test* the models using the examples collect by the other agencies, which under our assumption are sampled from a similar but different distribution.

6.4 Case study for spurious correlation: data partitioning

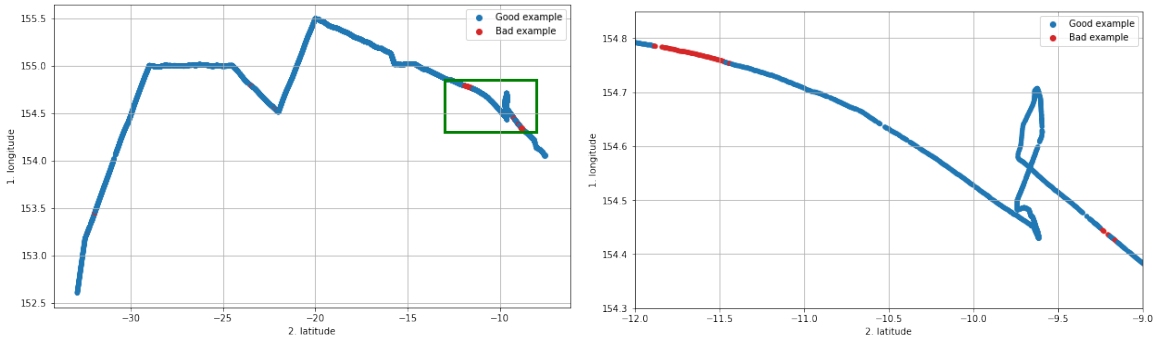
In the common practice, data sets are split randomly, which might not be adequate. To see that, we describe the problem it causes if we apply this type of data splitting in the bathymetry data editing task.

Individual-based splitting As in the common practice, we merge all measures from all the cruises, without noting to which research institution they belong. Then we decide the proportion of the measurements to be put into the training set, ρ. Finally, we iterate through all measurements, and assign them to the training set with the probability of ρ, and to the testing set with the probability of $1 - \rho$. An additional validation set can be generated similarly. We call this way of data splitting "individual-based splitting".

Individual-based splitting results in a classifier that delivers an exceptional accuracy rate. However, the high performance is due to a non-trivial case of overfitting caused by the data sequentiality. Upon further examination, we discover that the classifier is effectively memorizing the location of the measurements in the training sets and their corresponding labels.

As it shows in Figure 7, the label of one measurement is highly correlated with the label of the previous measure. In the data region marked out by the green box, we can see a clear split between the corrupted measurements (in red) and the valid measurements (in blue). The clear separation may have multiple explanation, for example, the echo sounder might have malfunctioned while traveling the ill-measured regions, but got fixed and worked properly after leaving the regions. If a machine learning model with the strong

Figure 7: Individual-based splitting can cause over-fitting



"memorizing" capability is applied in this case, it might exploit this phenomena and overfit in these regions. When it happens, the testing examples sampled from the same region will perform exceptionally, but we cannot trust this performance to be generalized to the regions not presented in our training data.

As a general principal, a reliable testing set should be acquired *independently* from the acquisition of the training set while preserving certain workflow involved in the data collection process. In the bathymetry data editing task, the measurements in the testing set should be collected by different cruises that are not observed in the training set. In certain task involving the video or other forms of sequential images, the testing data and the training data should not be sampled from the same clips. We argue that we cannot obtain a reliable generalization performance by simply guaranteeing that the training and testing examples do not overlap.

In Figure 7, we visualize the geolocation of the measurements, and use two different colors to denote their labels: the blue dots for the valid ones, and the red dots for the corrupted ones. We can see that there is a high correlation in the correctness of the measurements over the consecutive measurements. The reason is as follows. The cruises took the depth measurements sequentially over time. The measurements were predominantly accurate, until some abnormal activity, such as device malfunction, happened at some point t_1. Following the time t_1, all the measurements taken were all corrupted. Lastly, the cruises started to take correct measurements again at some point t_2 once the malfunction was fixed.

In the first iteration of our work, we created the training and testing sets using RTTM over the data instance. Specifically, we aggregate all measurements from all cruises, shuffle them, and randomly partition them into the training set and the testing set. We discovered that the accuracy of the trained model was almost perfect on the testing set (see Figure **??**). We then suspected that the model was able to exploit the sequentiality of the measurement correctness, and inferring the labels of the testing examples by comparing to the neighboring examples in the training set.

To test our hypothesis, we conducted the second iteration of the modeling. We added a restriction while using RTTM so that we assign the measurements from one cruise to either the training set or the testing set, but not both. As it shows in Figure **??**, We saw a significant decrease on the testing performance.

Cruise-based splitting We use cruise-based splitting method to address the overly-fined partitioning problem that we had in the individual-based splitting. To avoid the model to memorize and exploit the sequentiality in the measurements collected by the same cruise, we make the training/testing data split on the cruise level: all measurements from the same cruise will either be in the training set or in the testing set but not both.

Similar to the instance-based splitting, we first decide the proportion of *the number of cruises* to be put into the training set, ρ. Then we iterate through all cruise in random order, and assign them to the training set with the probability of ρ, and to the testing set with the probability of $1 - \rho$.

We expect to see a decrease in the model prediction accuracy.

6.5 Imbalance within the coherent subsets

Within one "coherent" subset, there can be an imbalance among the samples. For example, in the bathymetry data provided by one of the agencies (NGA), the top 5 cruises sorted by the number of measures (out of 1376 cruises) contain about the same number of measures that the rest of the cruises combined.

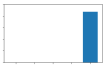
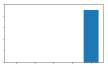
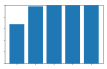
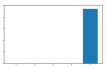
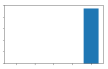
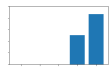
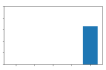
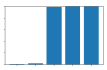
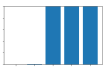
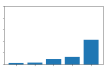
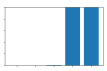
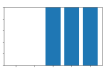
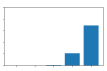
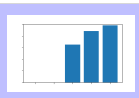
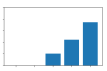
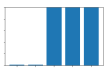
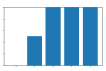
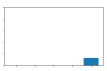
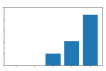
Table 2: Area Under ROC Curve (instance-based split)

Training data	AGSO	JAMSTEC	NGA	NGDC	NOAA geodas	SIO	US multi	all
AGSO	**99.99**	93.72	78.92	65.97	51.64	85.30	79.12	79.76
JAMSTEC	99.59	**99.96**	67.01	74.64	48.50	87.35	79.38	83.89
NGA	79.39	75.04	**99.64**	58.60	89.94	76.41	63.91	68.60
NGDC	70.83	84.90	51.35	**99.14**	46.12	76.24	75.38	88.22
NOAA geodas	53.72	45.54	88.55	55.04	**98.64**	68.20	47.13	59.20
SIO	91.67	76.47	76.19	67.61	73.23	**99.94**	79.70	78.82
US multi	96.45	88.20	76.88	74.07	53.85	88.54	**99.85**	86.33
all	99.34	98.41	98.69	97.56	96.21	99.18	97.87	**98.66**

Table 3: Area Under ROC curve (cruise-based split)

Model	AGSO	JAM	JAM2	NGA	NGA2	NGDC	NOAA	SIO	US multi
AGSO	**96.84**	80.84	65.36	33.02	69.01	56.75	54.61	82.00	63.76
JAMSTEC	73.46	**99.41**	86.96	82.45	88.56	70.48	58.90	88.20	83.26
JAMSTEC2	78.79	74.60	**96.20**	73.77	60.82	48.40	48.67	80.88	58.53
NGA	71.44	89.32	58.20	**99.07**	68.91	63.04	76.99	79.42	70.71
NGA2	68.76	85.14	86.12	74.77	**99.91**	77.40	71.01	73.11	59.75
NGDC	75.20	90.21	81.23	53.74	92.15	**93.20**	59.56	85.39	72.15
NOAA_geodas	76.29	87.71	92.26	90.16	69.11	65.97	**95.59**	89.86	72.41
SIO	69.65	91.97	89.16	74.21	88.63	57.99	65.19	**99.46**	71.34
US_multi	70.97	96.28	87.36	78.58	88.78	79.58	65.13	90.86	**94.12**

Table 4: Recall when the false discovery rates are 0.01%, 0.1%, 1%, 2%, and 5%

negative%	AGSO 1.68	JAMSTEC 0.53	NGA 36.14	NGDC 6.76	NOAA 19.73	SIO 28.02	US multi 11.74
AGSO model							
JAMSTEC model							
NGA model							
NGDC model							
NOAA model							
SIO model							
US multi model							
all model							

Acknowledgments

We would like to acknowledge support for this project from the National Institutes of Health (NIH grant) and the Scripps Institution of Oceanography, UC San Diego.

Appendix A.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, November 2016. USENIX Association. ISBN 978-1-931971-33-1. URL `https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi`.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Leo Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231, 2001.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

Trevor Darrell, Marius Kloft, Massimiliano Pontil, Gunnar Rätsch, and Erik Rodner. Machine learning with interdependent and non-identically distributed data (dagstuhl seminar 15152). In *Dagstuhl Reports*, volume 5. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

Marleen de Bruijne. Machine learning approaches in medical image analysis: From detection to diagnosis. *Medical Image Analysis*, 33:94 – 97, 2016. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2016.06.032. URL `http://www.sciencedirect.com/science/article/pii/S1361841516301098`.

Julian J Faraway. Data splitting strategies for reducing the effect of model selection on inference. *Comput Sci Stat*, 30:332–41, 1998.

Patrick J Grother. Nist special database 19 handprinted forms and characters database. *National Institute of Standards and Technology*, 1995.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009. URL `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`.

Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, page 1–9, New York, NY, USA, 2014a. Association for Computing Machinery. ISBN 9781450329996.

Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9, 2014b.

Christina Heinze-Deml and Nicolai Meinshausen. Conditional variance penalties and domain shift robustness. *arXiv preprint arXiv:1710.11469*, 2017.

Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4):1–21, 2012.

Wouter M Kouw and Marco Loog. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*, 2018.

Jakub Nalepa, Michal Myller, and Michal Kawulok. Validating hyperspectral image segmentation. *IEEE Geoscience and Remote Sensing Letters*, 16(8):1264–1268, 2019.

Andrew Ng. Nuts and bolts of building ai applications using deep learning. 2016.

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. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):947–1012, 2016.

Saharon Rosset, Claudia Perlich, Grzergorz Świrszcz, Prem Melville, and Yan Liu. Medical data mining: insights from winning two competitions. *Data Mining and Knowledge Discovery*, 20(3):439–468, 2010.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Annegreet Van Opbroek, M Arfan Ikram, Meike W Vernooij, and Marleen De Bruijne. Transfer learning improves supervised image segmentation across imaging protocols. *IEEE transactions on medical imaging*, 34(5):1018–1030, 2014.

Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2018.