Unstructured Data, Econometric Models, and Estimation Bias

Max Wei and Nikhil Malik*

Abstract

This article examines the powerful combination of machine learning and econometric models to study unstructured data. Researchers estimate an econometric model (e.g., discrete choice) that relates an outcome of interest (e.g., consumer purchases) to a focal feature in unstructured data (e.g., pet presence in product images), with the feature extracted by machine learning. We examine potential bias in the estimate of econometric model due to extraction errors by machine learning. Extraction errors are not white noises but functions of unstructured data. Consequently, the mechanisms and directions of bias are different from those in measurement errors. We derive general approaches to alleviate the bias as compared to an "oracle" who directly knows the focal feature. The approaches extend and improve the few pioneering works in this area, by: (i) covering general nonlinear econometric models, and (ii) removing the restrictive assumption that non-focal features of unstructured data have no effects on outcome of interest.

*Marshall School of Business, University of Southern California. yanhaowe@usc.edu and maliknik@usc.edu. We very much thank the comments by the participants of the VQMS 2022 (virtual) and AIML 2022 (Harvard).

1 Introduction

In recent years, more and more studies marry machine learning and econometric models. One popular theme is to use machine-extracted variables in econometric models. The focus is usually a feature of unstructured data (e.g., pet presence in product images). Researchers hope to estimate an econometric model for the relation between this feature and an outcome of interest (e.g., sales). One can manually label the feature but doing so for the entire sample can be too expensive. So, only a small subsample is labeled. The subsample is used to train a machine learning model, which then extracts the feature for the rest of the sample. In some cases, researchers directly use pre-trained machine learning model or APIs.[1]

This practice has undoubtedly led to insights which were traditionally not possible with unstructured data. However, it also brings new estimation issues, which this article aims to make clear and tackle. We focus on the following questions: should one expect bias in the estimate of econometric model after replacing a feature's true value with machine-extracted value? If yes, what are the causes of bias? Can we alleviate the bias (as compared to an "*oracle*" who knows the feature's true value for the entire sample)?

At first glance, some may think of the context here as a special case of measurement errors, with the "error" being the difference between machine-extracted value and feature's true value. However, measurement errors are typically studied as white noises, or more broadly, noises with no additional effects on the outcome of interest beyond the feature's true value.[2] In contrast, the error in our context comes from unstructured data and is a function of the unstructured data. Thus, it is not a white noise, and as we will discuss, it may contain "peripheral features" with additional effects on the outcome of interest. Therefore, the mechanisms and direction of bias in the econometric model can differ from those in measurement errors. For these reasons, we refer to the error as "extraction error" instead of "measurement error."

In this article, we clarify the different ways that extraction errors can introduce bias in the estimate of the econometric model. Our benchmark is an oracle who knows the feature's true value for the entire sample. It is assumed that human labeling can give the feature's true value.[3] There are a few pioneering works studying this setting, which we will discuss in a moment. We make two extensions. First, existing works require a quite restrictive assumption that non-focal features of unstructured data have no effect on the outcome of interest. We relax this assumption. Second, we cover general nonlinear econometric models.

Our first extension concerns an important property of machine learning. When extracting a focal feature, machine learning algorithms are designed to maximize prediction accuracy. To do so, they often seek "peripheral" (i.e., non-focal) features in unstructured data that are easy to recognize and correlate with the focal feature. As such, the machine extraction is inevitably contaminated with peripheral features in unstructured data. This property introduces a source of

[1] See Section 2 for some examples of papers applying this approach.

[2] See Chen et al. (2011) for a survey on classical and non-classical measurement errors. Also see Chen et al. (2005).

[3] We do not attempt to address possible human errors in labeling. It is an important question for future research.

bias in the econometric model. One convenient way to preclude this bias is to assume that all peripheral features have no effect on the outcome of interest (e.g., sales) in the econometric model, or equivalently, unstructured data affect the outcome of interest only via the focal feature. We refer to this assumption as "*no-effect-by-peripheral-features*." This assumption is indeed made in existing works to simplify analysis, but can be restrictive or unrealistic for applications.

To make the above more concrete, consider an example with the lodging market. Suppose we are interested in how property sales relate to pet presence in property photos. An oracle can regress sales directly on pet presence (plus other controls). But we must first train a machine to recognize pets on a labeled subsample, and then regress sales on machine-learned pet presence. Photos with pets may have colors, brightness, or backgrounds that on average are different from other photos. These peripheral features can be (rightfully) exploited by machines to help recognizing pets. To the extent that these peripheral features have effects on sales too, our regression result will be different from the oracle's (i.e., a bias).

To alleviate this source of bias, we train a second machine learning model that predicts the true value of the focal feature from its machine-extracted value. This allows us to construct estimation objective functions that are the same as the oracle's in population. Importantly, this second machine uses both the control variables and *the outcome of interest* in the econometric model as additional inputs. Here, including the outcome of interest (e.g., sales) as an input may be counter-intuitive. However, as we will show, it is exactly this inclusion that allows us to remove the no-effect-by-peripheral-features assumption.

Our second extension on the existing works is to cover general econometric models, including linear regression, instrumental variable, discrete choice, as well as structural econometric models. Specifically, we consider both MLE and GMM, and construct likelihood or moment functions that are the same as an oracle's in population.

An important note on bias alleviation is that our work focuses on bringing the estimate close to that of the oracle (who observes the focal feature's true value for the entire sample). However, the oracle's estimate can be biased too when the econometric model is mis-specified. For example, the focal feature is endogenous but the endogeneity is not accounted for. In this case, the oracle's estimate has a causal bias and is only correlational. We do not attempt to correct the oracle's estimate. Our approach always tries to recover oracle's estimate, whether the oracle's estimate is biased or not.

As mentioned, our extensions are made to the few pioneering papers that study the bias in econometric models caused by machine-extracted variables. Qiao and Huang (2021) focus on generalized linear regressions with distributional assumptions, under the restrictive no-effect-by-peripheral-features assumption. Fong and Tyler (2020) study linear regressions, again under the restrictive assumption.

In what follows, Section 2 reviews the current machine learning practice for feature extraction from unstructured data. Section 3 defines extraction errors, then uses examples to illustrate the different channels through which extraction errors cause bias in the econometric model. Section 4

and 5 develop the two-step learning (2SL) approach that alleviates bias. We develop the approach in two phases. We first extend the existing works to general nonlinear econometric models. The resulting approach is referred to as "2SL type 1" and still requires the no-effect-by-peripheral-features assumption. We then relax this assumption. The resulting approach is referred to as "2SL type 2." Section 6 conducts Monte Carlo studies. Section 7 tries out 2SL in an empirical application with Amazon reviews data.

2 Literature Review

We first review some examples in marketing and related fields that combine machine learning and econometric models to derive important insights. We highlight the diversity of machine learning techniques used in these studies. Then, using pre-training as a particular example, we discuss how machine learning algorithms rely on correlational data patterns and thus easily pick up peripheral features. Finally, we consider recent machine learning research that guides algorithms towards using true drivers instead of correlational patterns.

Researchers across business disciplines have always been interested in studying unstructured data such as text (Feldman et al. 2010), images (Biddle and Hamermesh 1998), video, and audio (Vickery et al. 2004). Marketing literature has studied impact of user review text features such as sentiment (Berger et al. 2012), valence, volume, variance (Chintagunta et al. 2010) and dispersion (Godes and Mayzlin 2004) on sales. Earlier analyses were limited by the technology then; researchers could only label simple concepts using traditional data mining methods. If they wanted more nuanced concepts like readability, novelty (Burtch et al. 2021) or truthfulness (Clarke et al. 2020), they could only get a few hundred labels using costly human experts.

Emergence of machine learning methods that can mimic human experts (Krizhevsky et al. 2012, Szegedy et al. 2015) has revolutionized the labeling of unstructured data at scale. Marketing research has picked up a variety of methods including random forests, XGBoost, neural nets, etc. For example, Puranam et al. (2021) are able to study the relationship between restaurant wages and nuanced user review text features such as staff courtesy, friendliness, wait time and cleanliness using nearly 100,000 reviews. Typically, marketing researchers split their analysis into two steps. In the first step, a machine learning model is used to extract features from unstructured data and in the second step the features are used as variables in econometric models to establish either correlational or causal effect of the features on outcomes such as clicks and sales. This approach has become very popular in the broad marketing literature and created great value to marketing research. Yet, machine learning models were not developed with a second-step econometric model in mind. Thus, it is important to examine whether the machine-extracted features should be used as-is in econometric models.

One important characteristic of the current research practice is the diversity of machine learning methods used to extract features from unstructured data. This diversity is a result of high customizability of machine learning methods. Take the neural net family for example. Image data

are generally analyzed using convolutional neural networks (CNN) (e.g., Liu et al. 2020, Hartmann et al. 2021). Text data is generally analyzed using recursive or long-short-term memory (LSTM) networks (e.g., Wu et al. 2021, Chakraborty et al. 2021). Increasingly researchers combine these architectures (Nian et al. 2021, Malik et al. 2020), refine on top of these architectures (Puranam et al. 2021), and even construct custom architectures (Sun et al. 2021, Gabel and Timoshenko 2021) to derive better performance. Even for a given model architecture, say a CNN, the researchers use a variety of heuristics to select model hyperparameters such as number of convolutional layers, number of weights in each layer, strength of regularization for these weights, etc. Zhang et al. (2021) settle on 13 convolutional layers for Airbnb photos while Liu et al. (2020) use 5 convolutional layers for Instagram pictures.

An increasingly important development for the customizability in machine learning is pre-training or transfer learning. Consider for example, a deep learning model to recognize whether an Instagram picture depicts any glamorous elements (Liu et al. 2020). Given the cost, researchers may only be able to collect labels for a few thousand images. But such a small, labeled sample is unlikely to sufficiently train a CNN which typically has millions of weights to learn. A popular solution is to re-use (i.e., "transfer") weights that have been learned from a related task where large, labeled training data are available, and then fine-tune some of the weights for the peculiar task at hand. In marketing, Liu et al. (2020) use CaffeNet model which is pre-trained using 1 million image samples albeit on a different task. Troncoso and Luo (2020) use the VGG16 model pretrained on the ImageNet dataset. Timoshenko and Hauser (2019) and Wu et al. (2021) use the word2vec model pretrained on Google News (Mikolov et al. 2013) or Wikipedia (Bojanowski et al. 2016).

A property of machine learning algorithms, which especially applies to transfer learning, is that they are very good at picking up correlational patterns in data that contribute to more accurate prediction of the focal feature. These patterns do not necessarily reflect the "true driver" of the feature or the "mechanisms" by which human recognizes the feature. For example, blue background occurs frequently in pictures of airplanes and thus is picked up by machine to recognize airplanes. This strategy may in fact be effective in improving the machine's accuracy, but does not capture proper mechanism to identify an airplane. Similar examples should be more common in transfer learning, where the machine is explicitly forced to build on simpler, first-order features learned during pre-training. For example, consider the task of classifying the tidiness of a room based on a machine learning model pre-trained to classify common objects (e.g., AlexNet, GoogLeNet). The fine-tuning stage typically has a much smaller training sample than that used in pre-training. Thus, it must heavily rely on correlations, say, between the object count in the picture and the tidiness of the room. While this approach can be effective to achieve good accuracy given the small training sample, it misses true mechanisms such as how the objects in the room are organized. All in all, this tendency to pick up correlational patterns means that the machine-extracted variable can introduce into the econometric model some "peripheral" features of the unstructured data that researchers are unaware of or do not intend to model.

In recent years, computer science research has started to scrutinize and rectify this property of

machine learning. Research on machine learning bias and fairness (Chouldechova and G'Sell 2017, Zhang and Neill 2016, Ascarza and Israeli 2022) identify and correct machine learning predictions that show systematic biases across social groups. Another stream of research designs training algorithms such that predictions are based on true drivers. They do so by distilling rules written by human experts into machine learning models (Ganchev et al. 2010, Hu et al. 2016). An example is directly telling the machine to look for wings, body, and tail composed in a certain way when recognizing airplanes. In a similar vein, a stream of research explores learning from a single example by exploiting prior knowledge and rules (Miller et al. 2003, Lake et al. 2013). These techniques are useful, but as of now they are not generally applicable to the breadth of machine learning models used in business research. In addition, they are not developed with a later econometric model in mind. Future research is needed to examine their value for econometric exercises.

Rectifying machine learning models for econometric exercises faces another challenge. More recently it has become common to extract features from unstructured data via paid APIs (application programming interfaces). Examples include Li & Xie (2019) who use Google Vision API for emotion and colorfulness of images, Shin et al. (2020) who use Kairos API for aesthetics, celebrity appearance, and adult content in images, and Gunarathne (2021) for racial recognition. Such ready-for-use, black-box packages altogether block scrutiny and correction of the underlying machine learning models.

Given this context, instead of rectifying feature extraction, this article focuses on developing the estimation procedure for the econometric model after feature extraction. In other words, we put little restriction on which machine learning model is used, how it is configured, or how it is trained to recognize the focal feature. This approach follows the very few pioneering works on the same topic (Fong and Tyler 2020, Qiao and Huang 2021). As discussed in the introduction, we make two extensions. First, we consider a much larger class of econometric models (any model estimable by MLE or GMM). Second, we remove the restrictive no-effect-by-peripheral-features assumption. This assumption sidesteps the property of machine learning discussed above that it tends to pick up peripheral features in unstructured data.

3 Examples

We use examples to illustrate the mechanisms by which machine-extracted feature introduces bias in the estimate of an econometric model. In this section, we focus on linear econometric models and a binary feature, where the intuitions are most clear. The next sections will cover general cases (nonlinear models and the feature may have more than two classes) and describe ways to alleviate the bias.

Let w_i be a feature of the high-dimensional object $\boldsymbol{z}_i$ (e.g., an image). We are interested in how w_i relates to an outcome of interest y_i (e.g., sales). For this section we focus on cases where $w_i \in \{0, 1\}$ is binary. There is a mapping F such that $w_i = F(\boldsymbol{z}_i)$. We can think of F as manual labeling. Let $\mathcal{N} = \{1, ..., n\}$ index the sample. We take a random subsample $\mathcal{S} \subset \mathcal{N}$ for manual

labeling. Thus, w_i is known in $\mathcal{S}$ but not in $\mathcal{N}\backslash\mathcal{S}$. In this paper, we use "oracle" to refer to someone who knows w_i for the entire sample $\mathcal{N}$.

Using the subsample $\mathcal{S}$, we train an machine learning model f that tries to extract w_i from $\boldsymbol{z}_i$. Examples of f include neural net, tree, and SVM. The output $f(\boldsymbol{z}_i)$ can be either binary or a probability. In this article we do not take a stand on what machine learning model should be used or how it should be trained. Instead, we treat f as a generic prediction model, and write

$$w_i = \widehat{w}_i + \widehat{v}_i, \quad \text{where } \widehat{w}_i = f(\boldsymbol{z}_i).$$

In above, $\widehat{v}_i$ is the extraction error. Here we follow the convention in prediction models that defines an error as the truth minus prediction (instead of the prediction minus truth).[4] The extraction error $\widehat{v}_i$ picks up variation in w_i unexplained by $\widehat{w}_i$. Thus typically w_i and $\widehat{v}_i$ are positively correlated: $\mathbf{cov}(w_i, \widehat{v}_i) > 0$.

It is useful to note the differences between extraction errors and measurement errors. In the measurement error literature, the measurement is typically expressed as follows, with $\widetilde{v}_i$ being the measurement error.

$$\widetilde{w}_i = w_i + \widetilde{v}_i,$$

Most notably, classical measurement error is uncorrelated with truth: $\mathbf{cov}(w_i, \widetilde{v}_i) = 0$ (see, e.g., Wooldridge 2002). In contrast, as pointed out above we typically have $\mathbf{cov}(w_i, \widehat{v}_i) > 0$. A more fundamental difference is that $\widetilde{v}_i$ is assumed to have no effects on y_i beyond w_i (Chen et al. 2011). In contrast, the extraction error $\widehat{v}_i$ usually contains traits of unstructured data ($\boldsymbol{z}_i$) that may have effects on y_i. Below, we illustrate how these differences introduce new sources of bias in econometric model.

3.1 Linear regression

Let $\boldsymbol{x}_i$ collect the explanatory variables other than w_i. We consider the following linear regression model.

$$y_i = \boldsymbol{\beta}'\boldsymbol{x}_i + \delta w_i + \varepsilon_i, \tag{1}$$

with the OLS conditions $\mathbf{cov}(\boldsymbol{x}_i, \varepsilon_i) = \mathbf{0}$ and $\mathbf{cov}(w_i, \varepsilon_i) = 0$. Given this econometric model, the oracle simply regresses y_i on $\boldsymbol{x}_i$ and w_i by OLS to obtain the estimates of $\boldsymbol{\beta}$ and δ.

As a hypothetical example, consider lodging rentals (e.g., Airbnb). Photos are displayed for each rental listing. We want to estimate how showing pets in photos affects demand. Let i index listing-month combinations. Outcome y_i is the number of days booked, $\boldsymbol{z}_i$ represents the photos, $w_i \in \{0, 1\}$ indicates whether $\boldsymbol{z}_i$ show pets, and $\boldsymbol{x}_i$ collects control variables such as price, square footage, furnishing. If $\boldsymbol{x}_i$ and w_i are exogenous, model (1) is causal. If there is endogeneity, model (1) is mis-specified and the OLS estimates are only correlational. In other words, even the oracle has a "causal" bias. As discussed, our focus is the potential bias of our estimate as compared to the

[4] An example of this convention is linear regressions, where the prediction error (also known as the regression residual) equals the observed outcome minus the predicted outcome.

oracle's, whether the oracle has a causal bias or not. We will not attempt to correct the oracle's bias.

Unlike the oracle, we do not observe w_i in $\mathcal{N}\backslash\mathcal{S}$ but we are interested in reaching the oracle's estimates as much as possible. The prevailing approach is to replace w_i with $\widehat{w}_i$ (i.e., direct substitution). Then, we have for $i \in \mathcal{N}\backslash\mathcal{S}$,

$$y_i = \boldsymbol{\beta}'\boldsymbol{x}_i + \delta\widehat{w}_i + (\delta\widehat{v}_i + \varepsilon_i). \tag{2}$$

The regression residual becomes $\delta\widehat{v}_i + \varepsilon_i$. If we run a regression as above, the coefficient estimates will be biased in general (as compared to the oracle's). There are three sources of bias, which we describe below.

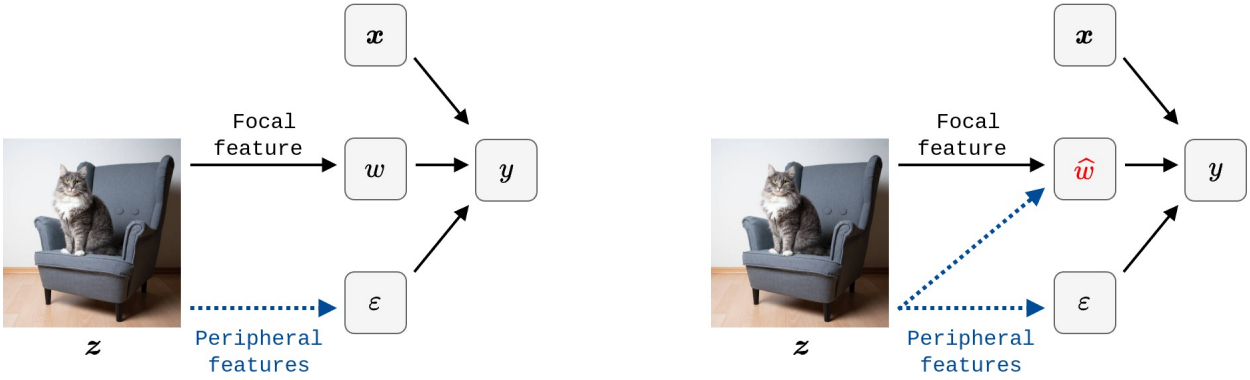
Three sources of bias The *first source of bias* is the correlation between $\widehat{w}_i$ and $\widehat{v}_i$. The sign of correlation depends on the machine learning model f. However, it is useful to note that for classification algorithms that output binary $\widehat{w}_i$ (e.g., SVM, classification forest), the correlation is always negative. To see this, note that because $w_i \in \{0, 1\}$, we have $\widehat{w}_i = 0 \;\Rightarrow\; \widehat{v}_i \geqslant 0$ and $\widehat{w}_i = 1 \;\Rightarrow\; \widehat{v}_i \leqslant 0$. As a result, $\mathbf{cov}(\widehat{w}_i, \widehat{v}_i) < 0$, which biases our estimate of δ towards zero as compared to the oracle.

For algorithms that give probabilistic $\widehat{w}_i$ (e.g., k-NN, neural net), $\mathbf{cov}(\widehat{w}_i, \widehat{v}_i)$ can be either negative or positive. Later we will show an example of positive correlation, leading to an overestimate of δ as compared to the oracle. This is the opposite of the attenuation bias mostly seen with classical measurement errors.

The *second source of bias* is the correlation between $\boldsymbol{x}_i$ and $\widehat{v}_i$. This correlation is likely nonzero in many applications because: (i) $\widehat{v}_i$ is a function of $\boldsymbol{z}_i$ and thus may pick up features in $\boldsymbol{z}_i$, and (ii) features in $\boldsymbol{z}_i$ may very well be correlated to the variables in $\boldsymbol{x}_i$. A non-zero correlation between $\boldsymbol{x}_i$ and $\widehat{v}_i$ causes estimation bias for $\boldsymbol{\beta}$, which indirectly affects the estimate for δ.

To see this source of bias in a concrete context, consider the lodging rentals example again. Let us take a step back and suppose $\mathbf{cov}(\widehat{w}_i, \widehat{v}_i) = 0$, i.e., the first source of bias is not present. Then, we necessarily have $\mathbf{cov}(w_i, \widehat{v}_i) > 0$ because $\mathbf{cov}(w_i, \widehat{v}_i) = \mathbf{cov}(\widehat{w}_i + \widehat{v}_i, \widehat{v}_i) = \sigma^2(\widehat{v}_i)$. That is, $\widehat{v}_i$ has to pick up some variation in w_i. Now suppose that $\boldsymbol{x}_i$ includes an indicator for whether the rental property is listed by a single person or a couple. To the extent that couples are more likely pet owners, $\boldsymbol{x}_i$ is correlated with w_i. Consequently, $\boldsymbol{x}_i$ is likely correlated with $\widehat{v}_i$ because $\widehat{v}_i$ picks up variation in w_i.

The *third source of bias* is the main focus of this article and it is the correlation between $\widehat{w}_i$ and ε_i. It is important to note that this correlation is not guaranteed to be zero by the OLS condition $\mathbf{cov}(w_i, \varepsilon_i) = 0$. In other words, $\mathbf{cov}(w_i, \varepsilon_i) = 0 \not\Rightarrow \mathbf{cov}(\widehat{w}_i, \varepsilon_i) = 0$. If indeed $\mathbf{cov}(\widehat{w}_i, \varepsilon_i) \neq 0$, then it will bias our estimate of δ as compared to the oracle's. Specifically, the case $\mathbf{cov}(\widehat{w}_i, \varepsilon_i) \neq 0$ arises when two events happen: (i) $\widehat{w}_i$ picks up peripheral features in $\boldsymbol{z}_i$ and (ii) these peripheral features enter ε_i. Event (i) is possible because the machine learning model f that outputs $\widehat{w}_i$ can exploit any information in $\boldsymbol{z}_i$. Event (ii) is possible because the peripheral features can have effects



Notes: Illustration of the third source of bias in linear regression. The left side depicts the oracle's econometric exercise, where y_i is the outcome of interest, w_i is the focal feature, $\boldsymbol{x}_i$ collects controls, and ε_i is the regression residual. The right side depicts our econometric exercise that replaces w_i with $\widehat{w}_i$. Peripheral features can enter both $\widehat{w}_i$ and ε_i (blue dashed arrows). As a result, the effect of $\widehat{w}_i$ on y_i does not only include the effect of w_i on y_i but also picks up some effect of peripheral features on y_i.

Figure 1: The Third Source of Bias

on y_i just like the focal feature.

To see this third source of bias in a concrete context, consider the lodging rentals example again. Suppose it is relatively easy for algorithms to spot "fluffy objects" in photos. Then, our machine learning model f may exploit fluffy objects to help predict pet presence. Consequently, $\widehat{w}_i$ picks up all types of fluffy objects other than pets: sofa, rug, etc. Now suppose that photos featuring sofa increase sales y_i, which means the presence of sofa enters ε_i in equation (1). Because sofa enters both $\widehat{w}_i$ and ε_i, we have $\mathbf{cov}(\widehat{w}_i, \varepsilon_i) \neq 0$, i.e., the third source of bias. In general, $\widehat{w}_i$ always has extraction error. The key observation here is that the extraction error is not a white noise but composed of peripheral features (sofa) relevant to the outcome y_i.

Figure 1 visually illustrates the third source of bias. The left side depicts the oracle's econometric exercise. The right side depicts our econometric exercise, which replaces w_i with $\widehat{w}_i$. As peripheral features can enter $\widehat{w}_i$ (blue dashed arrows), the effect of $\widehat{w}_i$ on y_i does not only include the effect of w_i on y_i but also picks up some effect of peripheral features on y_i.

One way to ensure $\mathbf{cov}(\widehat{w}_i, \varepsilon_i) = 0$ is to replace the OLS condition with a stronger assumption: $\mathbf{E}(\varepsilon_i|\boldsymbol{z}_i) = 0$. We have $\mathbf{E}(\varepsilon_i|\boldsymbol{z}_i) = 0 \Rightarrow \mathbf{cov}(\widehat{w}_i, \varepsilon_i) = 0$ because $\widehat{w}_i$ is a function of $\boldsymbol{z}_i$. This stronger assumption says that ε_i is "clean" of $\boldsymbol{z}_i$. Together with equation (1), the assumption implies that the only channel for $\boldsymbol{z}_i$ to affect y_i is the focal feature w_i, or that no peripheral features of $\boldsymbol{z}_i$ affect y_i. We thus refer to it as *no-effect-by-peripheral-features* assumption. This assumption has simplified analysis in previous studies on the econometric bias due to machine-extracted variables (Qiao and Huang 2021, Fong and Tyler 2020). However, the discussion so far should have shown that this stronger assumption often does not hold in applications.

We have identified three sources of bias. At a high level, these sources of bias arise because of the complexity of extraction errors – they are not just white noises but functions of the unstructured data.

Table 1: Monte Carlo Example - Linear Regression Model

Estimates of δ	(1)		(2)		(3)	
(True value = 1)	Mean	RMSE	Mean	RMSE	Mean	RMSE
Oracle	1.003	0.035	0.995	0.061	1.003	0.032
Direct Substitution	1.564	0.564	0.801	0.23	1.98	0.985
Projected Substitution	1.011	0.065	1.019	0.168	1.281	0.289
2-Step Learning	1.013	0.079	0.988	0.107	0.979	0.055

Notes: Based on estimates of δ from 100 Monte Carlo samples. True value of $\delta = 1$. Each sample has a size of 5000, and the labeled subsample has a size of 1000. Red color indicates bias of magnitude $\geqslant 10\%$.

Monte Carlo example Table 1 displays some Monte Carlo results that demonstrate the three sources of bias. Reported are mean and RMSE of the estimate for δ across 100 samples. The true value of $\delta = 1$. Standard errors are not reported to avoid clutter, but an upper bound for standard error can be quickly calculated as RMSE/$\sqrt{100}$ (because RMSE $\leqslant$ standard deviation). The details of the Monte Carlo setup follows the later Section 6 (except that the econometric model here is linear while in Section 6 is logistic). So we omit these details here.

We first focus on the row "direct substitution," which is an OLS regression of y_i on $\boldsymbol{x}_i$ and $\widehat{w}_i$. Column 1 shows a setup where both $\boldsymbol{x}_i$ and ε_i are independent of $\boldsymbol{z}_i$. Note this setup precludes the second and third sources of bias. We see a >50% over-estimate of δ compared to the oracle. The cause of bias is a positive correlation between $\widehat{w}_i$ and $\widehat{v}_i$ (the first source of bias). Column 2 follows the setup in Column 1 except that it specifies $\boldsymbol{x}_i$ to be correlated with w_i, which leads to $\mathbf{corr}(\boldsymbol{x}_i, \widehat{v}_i) \neq \mathbf{0}$. We see the bias is reversed from Column 1 to an under-estimate of δ. This result demonstrates the second source of bias. Column 3 follows the setup in Column 1 except that a peripheral feature of $\boldsymbol{z}_i$ now enters ε_i. As a consequence, $\mathbf{corr}(\widehat{w}_i, \varepsilon_i) \neq 0$. We see a larger bias than that in Column 1. This result shows the third source of bias.

The last two rows of Table 1 show "projected substitution" and "2-step learning." Projected substitution refers to a bias-correction method proposed by Fong and Taylor (2021). It assumes linear regression as the econometric model and requires the no-effect-by-peripheral-features assumption. Thus, as expected, we see it showing a bias in Column 3. Two-step learning is a method to be developed in this article. It works for general nonlinear econometric models and does not require the no-effect-by-peripheral-features assumption. It shows virtually no bias in all three columns.

3.2 Instrument variable

We consider the setting of instrument variable (IV) where w_i is endogenous. This setting is of particular interests because IV is a well-known method to correct the attenuation bias due to classical measurement error. Thus, when a researcher has an instrument for w_i, it is tempting to think that the IV regression will correct the bias due to the extraction error in $\widehat{w}_i$. In what follows, we show this is not the case. Actually, using IV can make bias even worse. In this regard, this example further illustrates the difference between extraction errors and classical measurement errors.

Table 2: Monte Carlo Example - Instrumental Variable

Estimates of δ (true value = 1)	Mean	RMSE
Oracle (OLS)	1.425	0.426
Oracle (IV)	1.002	0.089
Direct Substitution (OLS)	2.21	1.215
Direct Substitution (IV)	3.587	2.613
2-Step Learning (IV)	1.032	0.115

Notes: Based on estimates of δ from 100 Monte Carlo samples. True value of $\delta = 1$. Setup follows Column 1 of Table 1 except that w_i is endogenous. Red color indicates bias of magnitude $\geqslant 10\%$.

In terms of notation, let $\boldsymbol{x}_i = (\boldsymbol{s}_i', t_i)'$, where t_i denotes the instrument and $\boldsymbol{s}_i$ collects the exogenous regressors (i.e., controls). The econometric model is

$$y_i = \boldsymbol{\beta}'\boldsymbol{s}_i + \delta w_i + \varepsilon_i,$$

with $\mathbf{cov}(w_i, \varepsilon_i) \neq 0$, $\mathbf{cov}(t_i, w_i) > 0$, and $\mathbf{cov}(t_i, \varepsilon_i) = 0$. The oracle can estimate $\boldsymbol{\beta}$ and δ via standard IV regression. IV regression is a special case of GMM, with the moment condition being $\mathbb{E}\left[(y_i - \boldsymbol{\beta}'\boldsymbol{s}_i - \delta w_i) \cdot \boldsymbol{x}_i\right] = \mathbf{0}$.

Aside from addressing the endogeneity of w_i, an instrument can also address classical measurement errors. To see this, suppose we do not observe w_i but only a measurement of it: $\widetilde{w}_i = w_i + \widetilde{v}_i$ where $\widetilde{v}_i$ is a white noise. Replacing w_i with $\widetilde{w}_i$ in the econometric model, we have

$$y_i = \boldsymbol{\beta}'\boldsymbol{s}_i + \delta\widetilde{w}_i + (-\delta\widetilde{v}_i + \varepsilon_i).$$

The regression residual is now $-\delta\widetilde{v}_i + \varepsilon_i$. The key observation here is that t_i is a valid instrument for $\widetilde{w}_i$. To this see, note $\mathbf{cov}(t_i, -\delta\widetilde{v}_i + \varepsilon_i) = -\delta\mathbf{cov}(t_i, \widetilde{v}_i) + \mathbf{cov}(t_i, \varepsilon_i) = 0$. The second equality has used the fact that $\widetilde{v}_i$ is a white noise and thus uncorrelated with t_i. As a result, unbiased estimates of $\boldsymbol{\beta}$ and δ can be obtained via an IV regression of y_i on $\boldsymbol{s}_i$ and $\widetilde{w}_i$ using t_i as an instrument for $\widetilde{w}_i$.

The same IV approach, however, will most likely fail in the case of extraction error. Let $\widehat{w}_i$ be the machine-extracted value of w_i. Continuing with the notation so far, we write $w_i = \widehat{w}_i + \widehat{v}_i$. Replacing w_i with $\widehat{w}_i$ in the econometric model, we have

$$y_i = \boldsymbol{\beta}'\boldsymbol{s}_i + \delta\widehat{w}_i + (\delta\widehat{v}_i + \varepsilon_i).$$

The regression residual is $\delta\widehat{v}_i + \varepsilon_i$. As discussed in the beginning of this section, $\widehat{v}_i$ is not a white noise and typically picks up variation in w_i. This fact, coupled with the setup $\mathbf{cov}(t_i, w_i) > 0$, implies that $\mathbf{cov}(t_i, \widehat{v}_i) > 0$ is likely true. If so, then $\mathbf{cov}(t_i, \delta\widehat{v}_i + \varepsilon_i) = \delta\mathbf{cov}(t_i, \widehat{v}_i) \neq 0$, which says t_i is not a valid instrument for $\widehat{w}_i$. Therefore, IV cannot address extraction errors.

In fact, using IV in the case of extraction error can lead to an even larger bias than OLS. Table 2 shows a Monte Carlo example. The setup follows that in Column 1 of Table 1, with the exception that w_i is made endogenous (positively correlated with ε_i). A valid instrument is constructed for

w_i (see Appendix A for replication details). The true value of δ is 1. The first two rows show the oracle's case. We see that OLS has a significant over-estimate whereas IV has virtually no bias, which are expected. The next two rows show the estimates by direct substitution. OLS has a large bias but IV has an even larger bias.

Table 2 also reports the result from 2-step learning, the method to be developed in this article. The method is general enough to cover IV regression (as a special case of GMM). We see that it removes the bias, and the RMSE is not far from that of the oracle.

4 Setup

We now turn to general econometric models. Suppose that our interest is to estimate an econometric model that specifies some outcome y_i as a (linear or nonlinear) function of $w_i, \boldsymbol{x}_i, \boldsymbol{\varepsilon}_i$, and parameter $\boldsymbol{\theta}$. Here, w_i is a feature of high-dimensional object $\boldsymbol{z}_i$ (e.g., an image), $\boldsymbol{x}_i$ collects other explanatory variables and is much lower-dimensional than $\boldsymbol{z}_i$, and $\boldsymbol{\varepsilon}_i$ collects the residual/error terms. There is a mapping $F : w_i = F(\boldsymbol{z}_i)$. One way to think of F is manual labeling – a human can correctly label w_i from $\boldsymbol{z}_i$. In this article we study the case where w_i is a discrete variable with a finite number of possible values.

A hypothetical example of this setting was given in Section 3. We considered the context of lodging rentals (e.g., Airbnb) and particularly pet presence in rental listing photos. We wish to estimate the relation between pet presence and rental sales. In this example, i indexes listing-month combinations. Outcome y_i is the number of bookings, $\boldsymbol{z}_i$ collects the property photos, w_i indicates the presence of pets in photos, and $\boldsymbol{x}_i$ collects control variables such price, review rating, square footage, etc.

We use $\mathcal{N} = \{1, ..., n\}$ to denote the entire sample. An oracle refers to someone who directly observes w_i for all $i \in \mathcal{N}$. Below, we first recall common approaches to estimate $\boldsymbol{\theta}$ from the oracle's perspective. The purpose is to set up boundaries for the set of econometric models that we will consider in the next section, where we move to the non-oracle case in which w_i is only known for a subset of $\mathcal{N}$.

Common approaches to estimate $\boldsymbol{\theta}$ include MLE and GMM, which apply to a wide range of econometric models, including OLS, IV regression, discrete choices, and most structural econometric models. We first consider GMM, which sometimes starts with a conditional mean specification:

$$\mathbb{E}(y_i|\boldsymbol{x}_i, w_i) = m(\boldsymbol{x}_i, w_i, \boldsymbol{\theta}). \tag{3}$$

For example, a linear regression specifies $m(\cdot) = \boldsymbol{\beta}'\boldsymbol{x}_i + \delta w_i$, with $\boldsymbol{\theta} = (\boldsymbol{\beta}', \delta)'$. In structural econometric models, $m(\cdot)$ often has no closed form and is evaluated via simulations.

More generally, GMM does not have to rely on conditional mean specifications. It only requires a moment function $\boldsymbol{g}$ such that

$$\mathbb{E}\left[\boldsymbol{g}(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta})\right] = \mathbf{0}. \tag{4}$$

For example, under equation (3), we have $\mathbb{E}\{[y_i - m(\boldsymbol{x}_i, w_i, \boldsymbol{\theta})] \cdot (\boldsymbol{x}_i', w_i)'\} = \mathbf{0}$. So conditional specification such as (3) is covered by the more general moment condition (4). The oracle estimates $\boldsymbol{\theta}$ by trying to solve $\sum_{i \in \mathcal{N}} \boldsymbol{g}(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta}) = \mathbf{0}$ in case of exact-identification or using the two-step GMM procedure in case of over-identification.

Next, we turn to MLE, which relies on a likelihood function as implied by the econometric model specified by the researcher:

$$\Pr(y_i | \boldsymbol{x}_i, w_i) = \ell(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta}). \tag{5}$$

Take probit as an example. Let Φ be the standard normal cdf and $p_i \equiv \Phi(\boldsymbol{\beta}' \boldsymbol{x}_i + \delta w_i)$. We have $\ell(\cdot) = y_i p_i + (1 - y_i)(1 - p_i)$ with $\boldsymbol{\theta} = (\boldsymbol{\beta}', \delta)'$. The oracle estimates $\boldsymbol{\theta}$ by maximizing the sample objective function $\sum_{i \in \mathcal{N}} \log \ell(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta})$.

So far, we have been taking the oracle's perspective. When w_i is only known for a subset of $\mathcal{N}$, a common practice is to replace w_i with a machine-extracted value $\widehat{w}_i$ and then carry out GMM or MLE as usual (i.e., direct substitution). We know from the discussion in Section 3 that direct substitution leads to biased estimate of $\boldsymbol{\theta}$ as compared to the oracle's. The next section discusses how to reduce this bias.

5 Two-step learning

We now move to the non-oracle scenario, where w_i is known (e.g., via manual labeling) only in a random subsample $\mathcal{S} \subset \mathcal{N}$. Of course, one can choose to use only $\mathcal{S}$ for estimation. But in practice $\mathcal{S}$ is often a small portion of $\mathcal{N}$, especially with large data becoming common nowadays. So the interesting question is how to make use of $\mathcal{N} \backslash \mathcal{S}$. We propose a two-step learning (2SL) approach to construct correct likelihood/moment function in $\mathcal{N} \backslash \mathcal{S}$. The approach has two different types. Below we first describe type 1, which is more intuitive and motivates type 2. However, type 1 requires the no-effect-by-peripheral-features assumption. Type 2 removes the assumption. Both types are applicable to general nonlinear econometric models as outlined in Section 4.

5.1 Type 1

We start with the GMM case and then turn to the MLE case.

2SL type 1 for GMM starts with an econometric model that specifies a conditional expectation as in equation (3): $\mathbb{E}(y_i | \boldsymbol{x}_i, w_i) = m(\boldsymbol{x}_i, w_i, \boldsymbol{\theta})$. We do not know w_i for $i \in \mathcal{N} \backslash \mathcal{S}$ and thus cannot evaluate $m(\cdot)$ there. As noted, 2SL type 1 requires the no-effect-by-peripheral-features assumption. In GMM, the assumption is formalized as

$$\mathbb{E}(y_i | \boldsymbol{x}_i, \boldsymbol{z}_i) = m(\boldsymbol{x}_i, w_i, \boldsymbol{\theta}). \tag{6}$$

The left side of equation (6) conditions on $\boldsymbol{z}_i$, and thus it is stronger than and implies (3). It essentially says that $\boldsymbol{z}_i$ affects y_i only through the focal feature w_i, or that no peripheral features of $\boldsymbol{z}_i$ affect y_i. This is a rather restrictive assumption but it simplifies estimation (see Section 3.1 that illustrates the role of this assumption in linear regression).

To estimate $\boldsymbol{\theta}$ under (6), we proceed as follows. First, split the labeled subsample into two parts: $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$. One may consider equal split for simplicity. (We discuss why it is important to split $\mathcal{S}$ later in Section 5.3). We use $\mathcal{S}_1$ to train a machine learning model $\boldsymbol{f}$ that predicts w_i from $\boldsymbol{z}_i$. In applications the choice of $\boldsymbol{f}$ (e.g., neural net, boosting, k-NN) is up to the researcher. The output $\boldsymbol{f}(\boldsymbol{z}_i)$ is a vector collecting the predicted probability for each possible value of w_i. In the case that $\boldsymbol{f}$ makes deterministic predictions, the vector $\boldsymbol{f}(\boldsymbol{z}_i)$ has 1 in one entry and 0 in other entries.

Then, apply $\boldsymbol{f}$ out-of-sample for each $i \in \mathcal{N} \backslash \mathcal{S}_1$ and collect the predictions in a vector $\widehat{\boldsymbol{w}}_i = \boldsymbol{f}(\boldsymbol{z}_i)$. In the special case where w_i is binary, $\widehat{\boldsymbol{w}}_i$ has two elements that always sum up to one (thus we can use a scalar $\widehat{w}_i$, as we did in Section 3).

Next, we use $\mathcal{S}_2$ to train a second machine learning model $\boldsymbol{p}$ that makes probabilistic prediction of w_i using $\{\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i\}$. Notation-wise, let $\mathcal{K}$ collect the possible values of w_i. For each $k \in \mathcal{K}$, let $p_k(\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)$ denote the predicted probability for $w_i = k$. Note $\{\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i\}$ is low-dimensional compared to $\boldsymbol{z}_i$. Thus, it is possible for $p_k(\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)$ to achieve a good approximation of $\Pr(w_i = k | \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)$ even when the size of $\mathcal{S}_2$ is modest. We discuss the choice of $\boldsymbol{p}$ later in Section 5.3.

Finally, we consider the following conditional expectation for $i \in \mathcal{N} \backslash \mathcal{S}$.

$$\begin{aligned}
\mathbb{E}(y_i | \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) &= \sum_{k \in \mathcal{K}} \Pr(w_i = k | \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) \cdot \mathbb{E}(y_i | \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i, w_i = k) \\
&= \sum_{k \in \mathcal{K}} \Pr(w_i = k | \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) \cdot m(\boldsymbol{x}_i, w_i = k, \boldsymbol{\theta}) \\
&\simeq \sum_{k \in \mathcal{K}} p_k(\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) \cdot m(\boldsymbol{x}_i, w_i = k, \boldsymbol{\theta}).
\end{aligned}$$

The second equality makes use of assumption (6) and the fact that $\widehat{\boldsymbol{w}}_i$ is a function of $\boldsymbol{z}_i$. Also note the equality would not hold under only (3). Now, one can use the above expression of $\mathbb{E}(y_i | \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)$ to derive moment conditions. In particular, $\mathbb{E}\left\{[y_i - \mathbb{E}(y_i | \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)] \cdot \boldsymbol{\phi}(\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)\right\} = \mathbf{0}$ for any function $\boldsymbol{\phi}$. In case of exact-identification, we estimate $\boldsymbol{\theta}$ by trying to solve

$$\sum_{i \in \mathcal{N} \backslash \mathcal{S}} [y_i - \mathbb{E}(y_i | \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)] \cdot \boldsymbol{\phi}(\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) = \mathbf{0}. \tag{7}$$

In case of over-identification, one follows the usual two-stage GMM procedure.

It is useful to note that we can carry out 2SL twice by flipping the roles of $\mathcal{S}_1$ and $\mathcal{S}_2$. This will give us two estimates of $\boldsymbol{\theta}$, which we can average. At least in the case of equal split, the average estimate will have a smaller variance.

Example. (2SL type 1 for linear regression) It is instructive to see how 2SL type 1 applies (and

simplifies) in a linear regression with a binary focal feature $w_i \in \{0,1\}$. The econometric model is $y_i = \boldsymbol{\beta}'\boldsymbol{x}_i + \delta w_i + \varepsilon_i$ with $\mathbb{E}(\varepsilon_i|\boldsymbol{x}_i, w_i) = 0$, which implies $\mathbb{E}(y_i|\boldsymbol{x}_i, w_i) = \boldsymbol{\beta}'\boldsymbol{x}_i + \delta w_i$. Adding the no-effect-by-peripheral-features assumption, we have $\mathbb{E}(y_i|\boldsymbol{x}_i, \boldsymbol{z}_i) = \boldsymbol{\beta}'\boldsymbol{x}_i + \delta w_i$. Because w_i is binary, we can use a scalar $\widehat{w}_i$ to denote the output of $\boldsymbol{f}$. We may use another scalar $\widehat{\widehat{w}}_i$ to denote the output of the 2nd machine $\boldsymbol{p}$, that is, $\widehat{\widehat{w}}_i = p_1(\boldsymbol{x}_i, \widehat{w}_i) = 1 - p_0(\boldsymbol{x}_i, \widehat{w}_i)$. Applying the idea of 2SL type 1, we have for $i \in \mathcal{N}\backslash\mathcal{S}$,

$$\begin{aligned}
\mathbb{E}(y_i|\boldsymbol{x}_i, \widehat{w}_i) &= \Pr(w_i = 1|\boldsymbol{x}_i, \widehat{w}_i) \cdot \mathbb{E}(y_i|\boldsymbol{x}_i, \widehat{w}_i, w_i = 1) + \Pr(w_i = 0|\boldsymbol{x}_i, \widehat{w}_i) \cdot \mathbb{E}(y_i|\boldsymbol{x}_i, \widehat{w}_i, w_i = 0) \\
&= \Pr(w_i = 1|\boldsymbol{x}_i, \widehat{w}_i) \cdot (\boldsymbol{\beta}'\boldsymbol{x}_i + \delta) + \Pr(w_i = 0|\boldsymbol{x}_i, \widehat{w}_i) \cdot \boldsymbol{\beta}'\boldsymbol{x}_i \\
&\simeq \widehat{\widehat{w}}_i \cdot (\boldsymbol{\beta}'\boldsymbol{x}_i + \delta) + (1 - \widehat{\widehat{w}}_i) \cdot \boldsymbol{\beta}'\boldsymbol{x}_i \\
&= \boldsymbol{\beta}'\boldsymbol{x}_i + \delta\widehat{\widehat{w}}_i.
\end{aligned}$$

The last line suggests that we may regress y_i on $\boldsymbol{x}_i$ and $\widehat{\widehat{w}}_i$ to obtain the estimates for $\boldsymbol{\beta}$ and δ. So in the case of linear regression, 2SL type 1 essentially tells us to substitute w_i with $\widehat{\widehat{w}}_i$ (instead of $\widehat{w}_i$). Further, one can show that in this context, the 2nd machine learning model need not be flexible – a simple linear model will be sufficient. This special case is actually the estimator proposed by Fong and Taylor (2021). ■

2SL type 1 for MLE starts with the likelihood in equation (5): $P(y_i|\boldsymbol{x}_i, w_i) = \ell(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta})$. As noted, 2SL type 1 requires the no-effect-by-peripheral-features assumption. In MLE, the assumption is formalized as

$$P(y_i|\boldsymbol{x}_i, \boldsymbol{z}_i) = \ell(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta}). \tag{8}$$

The left side of equation (8) conditions on $\boldsymbol{z}_i$, and thus it is stronger than and implies (5). Like in GMM, the assumption essentially says that no peripheral features of $\boldsymbol{z}_i$ affect y_i.

The procedure of 2SL type 1 for MLE is the same as in the GMM case except for the last part. That is, we split $\mathcal{S}$ and train $\boldsymbol{f}$ on $\mathcal{S}_1$, let $\widehat{\boldsymbol{w}}_i = \boldsymbol{f}(\boldsymbol{z}_i)$, and then train $\boldsymbol{p}$ on $\mathcal{S}_2$. We omit the details as they follow exactly those in the GMM case. The procedure differs in the last part, where we consider the following conditional probability for $i \in \mathcal{N}\backslash\mathcal{S}$.

$$\begin{aligned}
\Pr(y_i|\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) &= \sum_{k\in\mathcal{K}} \Pr(w_i = k|\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) \cdot \Pr(y_i|\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i, w_i = k) \\
&= \sum_{k\in\mathcal{K}} \Pr(w_i = k|\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) \cdot \ell(y_i, \boldsymbol{x}_i, w_i = k, \boldsymbol{\theta}) \\
&\simeq \sum_{k\in\mathcal{K}} p_k(\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) \cdot \ell(y_i, \boldsymbol{x}_i, w_i = k, \boldsymbol{\theta}).
\end{aligned}$$

The second equality above follows from assumption (8) and the fact that $\widehat{\boldsymbol{w}}_i$ is a function of $\boldsymbol{z}_i$. Also note that the equality would not hold under only (5). With the above expression for $\Pr(y_i|\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)$,

we may estimate $\boldsymbol{\theta}$ by maximizing the following log likelihood

$$\sum_{i \in \mathcal{N} \setminus \mathcal{S}} \log \Pr(y_i | \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i). \tag{9}$$

As noted, we can always carry out 2SL twice by flipping the roles of $\mathcal{S}_1$ and $\mathcal{S}_2$. This will give us two estimates of $\boldsymbol{\theta}$, which we can average to reduce the estimation variance.

5.2 Type 2

Again, we start with the GMM case and then turn to the MLE case.

2SL type 2 for GMM starts with the moment condition as in equation (4): $\mathbb{E}\left[\boldsymbol{g}(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta})\right] = \mathbf{0}$, which covers the conditional expectation specification (3) as a special case. We do not need to make the additional no-effect-by-peripheral-features assumption.

The first step of 2SL type 2 is the same as in type 1, which we reproduce here. Split $\mathcal{S}$ into two parts: $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$. One may consider equal split for simplicity. (We discuss why it is important to split $\mathcal{S}$ later in Section 5.3). We use $\mathcal{S}_1$ to train a machine learning model $\boldsymbol{f}$ that predicts w_i from $\boldsymbol{z}_i$. In applications the choice of $\boldsymbol{f}$ is up to the researcher. The output $\boldsymbol{f}(\boldsymbol{z}_i)$ is a vector collecting the predicted probability for each possible value of w_i. In the case that $\boldsymbol{f}$ makes deterministic predictions, the vector has 1 in one entry and 0 in other entries. Let $\widehat{\boldsymbol{w}}_i = \boldsymbol{f}(\boldsymbol{z}_i)$.

The second step differs from 2SL type 1. We use $\mathcal{S}_2$ to train a second machine learning model $\boldsymbol{q}$ that makes probabilistic prediction of w_i using $\{y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i\}$. Note that y_i is used an input of $\boldsymbol{q}$. The reason for doing so will be clear in a moment. Let $\mathcal{K}$ collect the possible values of w_i. For each $k \in \mathcal{K}$, let $q_k(y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)$ denote the predicted probability for $w_i = k$. Again, because $\{y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i\}$ is low-dimensional, $q_k(y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)$ should achieve a good approximation of $\Pr(w_i = k | y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)$ even when the size of $\mathcal{S}_2$ is modest. We discuss the choice of $\boldsymbol{q}$ later in Section 5.3.

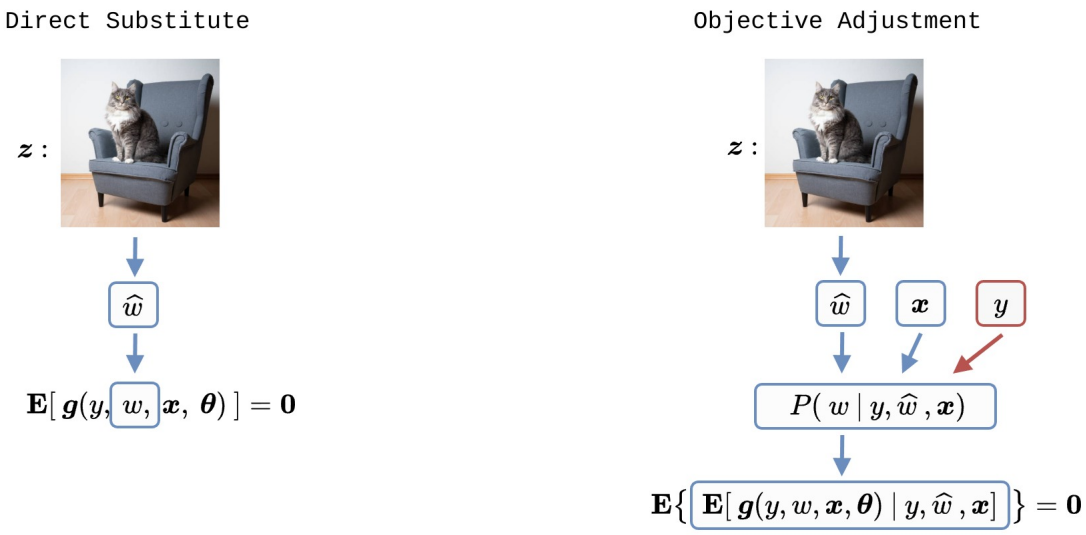
The key component of 2SL type 2 is to condition the moments on $\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i$, and y_i. Specifically, for $i \in \mathcal{N} \setminus \mathcal{S}$,

$$\begin{aligned} \mathbb{E}\left[\boldsymbol{g}(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta}) | y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i\right] &= \sum_{k \in \mathcal{K}} \Pr(w_i = k | y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) \cdot \boldsymbol{g}(y_i, \boldsymbol{x}_i, w_i = k, \boldsymbol{\theta}) \\ &\simeq \sum_{k \in \mathcal{K}} q_k(y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) \cdot \boldsymbol{g}(y_i, \boldsymbol{x}_i, w_i = k, \boldsymbol{\theta}). \end{aligned} \tag{10}$$

The second line uses the second machine learning model $\boldsymbol{q}$. By the law of iterated expectation, we have for $i \in \mathcal{N} \setminus \mathcal{S}$,

$$\mathbb{E}\Big\{ \mathbb{E}\left[\boldsymbol{g}(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta}) | y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i\right] \Big\} = \mathbb{E}\left[\boldsymbol{g}(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta})\right].$$

The inner conditional expectation on the left hand side can be evaluated with equation (10). The right hand side is the moment condition used by the oracle. In other words, equation (10) allows us to approximate a moment condition that equals the oracle's.



Notes: Illustration of the difference between direct substitution and 2SL type 2. Direct substitution is shown on the left, where the extracted feature $\widehat{w}_i$ directly replaces w_i in the moment function. 2SL type 2 is shown on the right. The $\widehat{w}_i$ is joined with $\boldsymbol{x}_i$ and y_i to predict w_i in the 2nd machine learning model, which then allows us to calculate the moment condition by law of iterated expectation.

Figure 2: Two-Step Learning (2SL) Type 2

From this point on, we follow standard GMM procedure. In the case of exact-identification, we estimate $\boldsymbol{\theta}$ by trying to solve

$$\sum_{i \in \mathcal{N} \setminus \mathcal{S}} \mathbb{E}\left[\boldsymbol{g}(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta}) | y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i\right] = \mathbf{0}. \tag{11}$$

The conditional expectation is again evaluated by equation (10). In the case of over-identification, we use the two-stage GMM. A graphical illustration of 2SL type 2 in comparison to direct substitution is given in Figure 2. Again, we note that y_i is included as input to the 2nd machine learning model.

As in type 1, we can carry out 2SL type 2 twice by flipping the roles of $\mathcal{S}_1$ and $\mathcal{S}_2$. This will give us two estimates of $\boldsymbol{\theta}$, which we can average. At least in the case of equal split, the average estimate will have a smaller variance.

Example. (2SL type 2 for linear regression) It is instructive to see how 2SL type 2 applies in a linear regression with a binary focal feature $w_i \in \{0, 1\}$. The econometric model is $y_i = \boldsymbol{\beta}'\boldsymbol{x}_i + \delta w_i + \varepsilon_i$ with $\mathbb{E}(\varepsilon_i | \boldsymbol{x}_i, w_i) = 0$, which implies $\mathbb{E}(y_i | \boldsymbol{x}_i, w_i) = \boldsymbol{\beta}'\boldsymbol{x}_i + \delta w_i$. The moment function is $\boldsymbol{g}(\cdot) = (y_i - \boldsymbol{\beta}'\boldsymbol{x}_i - \delta w_i) \cdot (\boldsymbol{x}_i', w_i)'$. Because w_i is binary, we can use a scalar $\widehat{w}_i$ to denote the output of $\boldsymbol{f}$. We may use another scalar $\widehat{\widehat{w}}_i$ to denote the output by the 2nd machine $\boldsymbol{q}$, that is, $\widehat{\widehat{w}}_i = q_1(y_i, \boldsymbol{x}_i, \widehat{w}_i) = 1 - q_0(y_i, \boldsymbol{x}_i, \widehat{w}_i)$. So the right side of equation (10) becomes

$$\widehat{\widehat{w}}_i \cdot (y_i - \boldsymbol{\beta}'\boldsymbol{x}_i - \delta) \begin{bmatrix} \boldsymbol{x}_i \\ 1 \end{bmatrix} + (1 - \widehat{\widehat{w}}_i) \cdot (y_i - \boldsymbol{\beta}'\boldsymbol{x}_i) \begin{bmatrix} \boldsymbol{x}_i \\ 0 \end{bmatrix} = \begin{bmatrix} (y_i - \boldsymbol{\beta}'\boldsymbol{x}_i - \delta\widehat{\widehat{w}}_i)\boldsymbol{x}_i \\ (y_i - \boldsymbol{\beta}'\boldsymbol{x}_i - \delta)\widehat{\widehat{w}}_i \end{bmatrix}.$$

Thus, the estimates for $\boldsymbol{\beta}$ and δ are obtained as the solution to

$$\sum_{i \in \mathcal{N} \setminus \mathcal{S}} \begin{bmatrix} (y_i - \boldsymbol{\beta}' \boldsymbol{x}_i - \delta \widehat{\widehat{w}}_i) \boldsymbol{x}_i \\ (y_i - \boldsymbol{\beta}' \boldsymbol{x}_i - \delta) \widehat{\widehat{w}}_i \end{bmatrix} = \mathbf{0}.$$

Note that the last moment has δ instead of $\delta \widehat{\widehat{w}}_i$ inside the parentheses. So the estimates are not the same as those from an OLS regression of y_i on $\boldsymbol{x}_i$ and $\widehat{\widehat{w}}_i$. ■

2SL type 2 for MLE starts with the likelihood specification in equation (5): $P(y_i|\boldsymbol{x}_i, w_i) = \ell(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta})$. Unlike in type 1, we do not need to make the additional no-effect-by-peripheral-features assumption.

The procedure of 2SL type 2 for MLE is the same as in the GMM case except for the last part. That is, we split $\mathcal{S}$ and train $\boldsymbol{f}$ on $\mathcal{S}_1$, let $\widehat{\boldsymbol{w}}_i = \boldsymbol{f}(\boldsymbol{z}_i)$, and then train $\boldsymbol{q}$ on $\mathcal{S}_2$. We shall omit the details as they follow exactly those in the GMM case. The procedure differs in the last part, where we consider the expectation of log likelihood conditional on $\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i$, and y_i:

$$\begin{aligned} \mathbb{E}\left[\log \ell(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta}) | y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i\right] &= \sum_{k \in \mathcal{K}} \Pr(w_i = k | y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) \cdot \log \ell(y_i, \boldsymbol{x}_i, w_i = k, \boldsymbol{\theta}) \\ &\simeq \sum_{k \in \mathcal{K}} q_k(y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) \cdot \log \ell(y_i, \boldsymbol{x}_i, w_i = k, \boldsymbol{\theta}). \end{aligned} \tag{12}$$

By the law of iterated expectation, we have for $i \in \mathcal{N} \setminus \mathcal{S}$,

$$\mathbb{E}\Big\{ \mathbb{E}\left[\log \ell(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta}) | y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i\right] \Big\} = \mathbb{E}\left[\log \ell(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta})\right].$$

The inner conditional expectation on the left side can be evaluated with equation (12). The right hand side is the population log likelihood used by the oracle. So equation (12) allows us to approximate a likelihood objective that equals the oracle's in population – the idea here is similar to that in the GMM case.

From this point on, we follow standard MLE procedure. Specifically, using sample analogue, we estimate $\boldsymbol{\theta}$ by maximizing

$$\sum_{i \in \mathcal{N} \setminus \mathcal{S}} \mathbb{E}\left[\log \ell(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta}) | y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i\right], \tag{13}$$

with the conditional expectation again evaluated by equation (12).

As noted, we can always carry out 2SL twice by flipping the roles of $\mathcal{S}_1$ and $\mathcal{S}_2$. This will give us two estimates of $\boldsymbol{\theta}$, which we can average to reduce the estimation variance.

Finally, it is instructive to note that the 2SL type 2 for MLE can be alternatively derived from 2SL type 2 for GMM. This is done by treating the first-order conditions of the likelihood as moment conditions (Newey and McFadden 1994), that is, $\mathbb{E}\left[\partial \log \ell(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta}) / \partial \boldsymbol{\theta}\right] = \mathbf{0}$. We give details in

the footnote.[5]

5.3 Discussion

We discuss several implementation details that are important to both types of 2SL. These include the importance to split subsample $\mathcal{S}$, choice of the 2nd-step machine learning model, and how to incorporate likelihood/moments from $\mathcal{S}$.

Split of subsample $\mathcal{S}$ Recall that we split $\mathcal{S}$ for separate trainings of the two machine learning models. The 1st machine ($\boldsymbol{f}$) is trained on $\mathcal{S}_1$ and the 2nd machine ($\boldsymbol{p}$ or $\boldsymbol{q}$) is trained on $\mathcal{S}_2$. (Note this split is different from the usual training-validation split. The training-validation split is used within the training of each machine learning model. For example, when training $\boldsymbol{f}$ we may split $\mathcal{S}_1$ into five folds for cross-validation. When training $\boldsymbol{p}$ or $\boldsymbol{q}$, we may split $\mathcal{S}_2$ into five folds for cross-validation).

The reason for the splitting $\mathcal{S}$ is the following. Take 2SL type 1 for example. Recall that we first train $\boldsymbol{f}$ to output $\widehat{\boldsymbol{w}}_i$. Then, when constructing the likelihood or moments in $\mathcal{N}\backslash\mathcal{S}$, we need the probability distribution $\Pr(w_i|\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)$ and we use $\boldsymbol{p}$ to learn it. Note in $\mathcal{N}\backslash\mathcal{S}$, $\widehat{\boldsymbol{w}}_i$ is an out-of-sample prediction. Thus, we need $\boldsymbol{p}$ to learn $\Pr(w_i|\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)$ where $\widehat{\boldsymbol{w}}_i$ is out-of-sample too. This is achieved by the splitting. If we forgo the splitting and train both $\boldsymbol{f}$ and $\boldsymbol{p}$ on the entirety of $\mathcal{S}$, then $\boldsymbol{p}$ learns $\Pr(w_i|\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)$ where $\widehat{\boldsymbol{w}}_i$ is within-sample.

The distinction between within-sample and out-of-sample is important here because machine learning models can be very flexible. As a consequence, the relation between w_i and in-sample $\widehat{\boldsymbol{w}}_i$ can be quite different from the relation between w_i and out-of-sample $\widehat{\boldsymbol{w}}_i$. An extreme example is $\boldsymbol{f}$ being k-nearest neighbors with $k = 1$. In this case, in-sample $\widehat{\boldsymbol{w}}_i$ always coincides with w_i but out-of-sample $\widehat{\boldsymbol{w}}_i$ does not. In general, the exact extent of problem caused by not splitting $\mathcal{S}$ depends on the model used for $\boldsymbol{f}$. The problem is likely more severe with "local" models, such as k-nearest neighbors, kernel methods, and random forest, as opposed to "global" models. We will verify this point in Monte Carlo studies (Section 6.4).

Splitting has an obvious downside that it reduces the size of training data for either machine learning model. However, as pointed out, this downside can be mitigated by flipping $\mathcal{S}_1$ and $\mathcal{S}_2$.

2nd-step machine learning model In 2SL, the 2nd machine learning model is denoted as $\boldsymbol{p}$ in type 1 and $\boldsymbol{q}$ in type 2. Model $\boldsymbol{p}$ estimates $\Pr(w_i|\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i)$ and $\boldsymbol{q}$ estimates $\Pr(w_i|\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i, y_i)$. Because the functional forms of these conditional distributions can be more complex than simple models (e.g., logit, multinomial logit), it is important to use a relatively flexible 2nd machine learning model. The extent to which using an inflexible 2nd machine learning model is problematic depends on the

[5]The moment function is $\boldsymbol{g}(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta}) = \partial \log \ell(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta})/\partial\boldsymbol{\theta}$. Note the number of moments equals the dimension of $\boldsymbol{\theta}$. Equation (10) becomes $\mathbb{E}\left[\boldsymbol{g}(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta})|y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i\right] \simeq \sum_{k\in\mathcal{K}} q_k(y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) \cdot \partial \log \ell(y_i, \boldsymbol{x}_i, w_i = k, \boldsymbol{\theta})/\partial\boldsymbol{\theta}$. As a result, the system of equations as given in (11) becomes $\sum_{i\in\mathcal{N}\backslash\mathcal{S}} \left[\sum_{k\in\mathcal{K}} q_k(y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i) \cdot \partial \log \ell(y_i, \boldsymbol{x}_i, w_i = k, \boldsymbol{\theta})/\partial\boldsymbol{\theta}\right] = \mathbf{0}$, which is the first-order conditions for the maximization problem in (13).

application. In Section 6.5 we illustrate a Monte Carlo study where noticeable biases still remain if logit is used for $\boldsymbol{p}$ and $\boldsymbol{q}$.

A candidate that we have found to work well for the 2nd machine learning model is neural nets. Neural nets, even the shallow ones, are universal approximators. In addition, statistical properties of shallow neural nets are well developed (e.g., White 1990, Chen 2007). A main result there is the generic consistency of neural nets when the number of training examples is large relative to the input dimension. In 2SL, the input to the 2nd machine is either $\{\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i\}$ or $\{\boldsymbol{x}_i, \widehat{\boldsymbol{w}}_i, y_i\}$, which are typically low-dimensional (especially compared to $\boldsymbol{z}_i$). Thus, a neural net should work well even with a modest sized $\mathcal{S}_2$.

It is important to carefully select the hyperparameters of the 2nd machine in 2SL, because the goal here is to estimate conditional probability distributions rather than just make a working predictor. The hyperparameters of a shallow neural net, including the number of hidden nodes and regularization parameter, can be chosen via cross-validation (on the subsample $\mathcal{S}_2$).

With the above said, hyperparameter selection can be time-consuming. So it can be a good idea to start 2SL with a simple 2nd machine learning model, such as logit. The resulting estimate of $\boldsymbol{\theta}$ typically has removed at least some bias from direct substitution. Thus, it serves as a preliminary estimate of $\boldsymbol{\theta}$ or a reference point when we move to "full-fledged" 2SL with a flexible 2nd machine learning model.

Incorporating likelihood/moments in $\mathcal{S}$ So far we have focused on constructing the correct likelihood or moments in $\mathcal{N}\backslash\mathcal{S}$. A question is how to incorporate the likelihood or moments from the observations in $\mathcal{S}$. A natural approach is to simply sum the likelihoods or moments across all observations. For example, in 2SL type 2 for GMM, the approach amounts to adding $\sum_{i\in\mathcal{S}} \boldsymbol{g}(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta})$ to the left side of equation (11). In 2SL type 2 for MLE, the approach amounts to adding $\sum_{i\in\mathcal{S}} \log \ell(y_i, \boldsymbol{x}_i, w_i, \boldsymbol{\theta})$ to the maximand (13). Intuitively, the resulting estimate of $\boldsymbol{\theta}$ will be some average between the direct estimate from $\mathcal{S}$ and the 2SL estimate from $\mathcal{N}\backslash\mathcal{S}$.

While the above approach is natural and intuitive, we note that it is not necessarily the optimal way to combine the likelihoods or moments from $\mathcal{N}\backslash\mathcal{S}$ and from $\mathcal{S}$. Finding the optimal estimator is beyond the scope of this article. It will be a valuable exercise for future research.

6 Monte Carlo

We consider a logistic regression with an explanatory variable extracted from images. Logit regression is one of the most widely used nonlinear econometric models. Our images are taken from CIFAR-10, a standard test ground for image recognition. Each image in CIFAR-10 has a resolution of 32x32 and shows one of the 10 objects including airplane, car, bird, cat, etc. We focus on a binary feature of these images: whether it shows a pet (cat, dog, or bird).

6.1 Econometric setup

We consider a binary outcome y_i that follows the logit model specified below:

$$y_i = \begin{cases} 1, & \text{if } \boldsymbol{\beta}'\boldsymbol{x}_i + \delta w_i + \varepsilon_i > 0; \\ 0, & \text{otherwise,} \end{cases}$$

In the above, $w_i = 1$ if image i shows a pet and $w_i = 0$ otherwise. Continuing with notations so far, we use $\boldsymbol{z}_i$ to denote the image. The error term ε_i follows logistic distribution and is independent of $\boldsymbol{x}_i$ and w_i. The likelihood implied by the logit model is

$$\Pr(y_i = 1|\boldsymbol{x}_i, w_i) = \frac{1}{1 + e^{-\boldsymbol{\beta}'\boldsymbol{x}_i - \delta w_i}}. \tag{14}$$

The no-effect-by-peripheral-features assumption is satisfied if ε_i is independent of the image $\boldsymbol{z}_i$, which implies $\Pr(y_i = 1|\boldsymbol{x}_i, \boldsymbol{z}_i) = 1/(1 + e^{-\boldsymbol{\beta}'\boldsymbol{x}_i - \delta w_i})$. As discussed, this assumption effectively says that the only channel for $\boldsymbol{z}_i$ to potentially affect y_i is through the focal feature w_i. We will examine scenarios where this assumption holds as well as scenarios where it does not hold.

CIFAR-10 has 60,000 images. We extend it to 120,000 by horizontally flipping every image. In a Monte Carlo experiment, we draw n images from the extended CIFAR-10 to form a sample. So by the notations so far, $\mathcal{N} \equiv \{1, ..., n\}$ and $\mathcal{S} = \{1, ..., s\}$ with $s < n$. We observe w_i for $i \in \mathcal{S}$ and pretend not observing w_i for $i \in \mathcal{N}\backslash\mathcal{S}$. As a benchmark, we also include the oracle that knows w_i in $\mathcal{N}\backslash\mathcal{S}$. We examine $n \in \{2500, 5000, 10000\}$ while fixing $s = n/5$.

We specify $\boldsymbol{x}_i$ to be 10 by 1. We explore several different distributions of $\boldsymbol{x}_i$, which we will describe later when discussing the results. We set $\beta_0 = 1$, $\beta_1 = \beta_2 = 0.5$, β_3, β_4, ..., $\beta_9 = 0$, and $\delta = 1$. The outcome y_i is simulated under these "true" values of coefficients. Note that although seven coefficients are zero, they not known to be zero a priori and must be estimated.

6.2 Feature extraction

For extracting w_i from $\boldsymbol{z}_i$, we make use of GoogLeNet, a pre-trained CNN that classifies images into 1000 objects. Examples of these objects are keyboard, coffee mug, pencil, wood rabbit, hare, bloodhound, beagle, golden retriever, airship, space shuttle, apron, cloak, etc. For each $\boldsymbol{z}_i$ (a 32x32 image), GoogLeNet outputs an 1000 by 1 vector $\boldsymbol{\pi}_i$, where entry π_{ij} gives the probability for object j. So $\sum_{j=1}^{1000} \pi_{ij} = 1$.

To project $\boldsymbol{\pi}_i$ into w_i, we consider two algorithms. The first algorithm tries to calculate the association between w_i and each of the 1000 objects that GoogLeNet includes. We refer to the algorithm as "scorecard." Specifically, we compute a score for each object j as $c_j = \sum_i \pi_{ij} w_i / \sum_i \pi_{ij}$, where the summations are over all training images. Intuitively, c_j measures the association of object j with $w_i = 1$. We predict w_i with $\widehat{w}_i = \sum_{j=1}^{1000} c_j \pi_{ij}$. The second algorithm is k-nearest neighbors (k-NN). Distances are calculated based on $\boldsymbol{\pi}_i$ and k is chosen via cross-validation. We only briefly examine k-NN and will specifically point out so when we do.

To get a feeling of these algorithms, we note that when using 1000 training images, the out-of-sample accuracy of scorecard is around 83% and of the k-nearest neighbors is around 81% (the baseline is 70%). These levels of accuracy are within the range typically found in marketing/IS literature.[6]

6.3 Main results

Table 3 shows our main Monte Carlo results. It considers four scenarios separated into four panels, which we will discuss in order below. Reported are means and root mean squared errors (RMSEs) for the estimate of δ, based on 100 Monte Carlo samples. The true value of δ is 1. Red color marks bias of magnitude $\geqslant$ 10%. Standard errors of means are not reported to avoid clutter, but an upper bound of standard error can be easily calculated as RMSE/$\sqrt{100}$ (because RMSE $\geqslant$ standard deviation).

Scenario (1) specifies both $\boldsymbol{x}_i$ and ε_i as independent of $\boldsymbol{z}_i$ (i.e., $\boldsymbol{x}_i \perp\!\!\!\perp \boldsymbol{z}_i$, $\varepsilon_i \perp\!\!\!\perp \boldsymbol{z}_i$). Each element of $\boldsymbol{x}_i \sim N(0,1)$. The first row shows the oracle that runs a logit regression of y_i on $\boldsymbol{x}_i$ and w_i in $\mathcal{N}\backslash\mathcal{S}$. The second row shows the direct substitution that runs a logit regression of y_i on $\boldsymbol{x}_i$ and $\widehat{w}_i$ in $\mathcal{N}\backslash\mathcal{S}$, where the algorithm that predicts $\widehat{w}_i$ is trained using the entirety of $\mathcal{S}$. We see a large over-estimate of δ, by near 50% (the true value of δ is 1). Importantly, this over-estimate persists as n becomes larger and thus is not just a finite-sample bias.

The third and fourth rows show 2SL. Here we use a shallow neural net in the 2nd step (with more details on implementation given later in Section 6.5). Split flipping and averaging are used. We see that both types of 2SL show little biases in estimates of δ, for all values of n. Also note that the RMSEs decline at roughly the rate of $\sqrt{n}$ when n increases.

Scenario (2) follows scenario (1) except that $x_{i,1}$ and $x_{i,2}$ correlate with w_i (so $\boldsymbol{x}_i \not\perp\!\!\!\perp w_i$). Specifically, we let $x_{i,1} = 1.5w_i + 0.75\xi_{i,1}$ and $x_{i,2} = -1.5w_i + 0.75\xi_{i,2}$, where $\xi_{i,1}, \xi_{i,2} \sim N(0,1)$. This results in $\mathbf{corr}(x_{i,1},w_i) \simeq 0.7$ and $\mathbf{corr}(x_{i,2},w_i) \simeq -0.7$. We see that direct substitution now under-estimates δ, so the correlations have reversed the bias. Both types of 2SL show modest biases at $n = 2500$ ($s = 500$) and little biases at larger n. In general, some bias with 2SL at small s should not be surprising because the 2nd machine learning model requires training data to perfect.

Scenario (3) illustrates a key result of this article. It follows scenario (1) except that a correlation of about 0.5 is added between ε_i and a peripheral feature of the image $\boldsymbol{z}_i$ (i.e., $\varepsilon_i \not\perp\!\!\!\perp \boldsymbol{z}_i$). We give the technical detail of this feature in Appendix B. The feature is independent of w_i so the econometric model (14) is still correctly specified. But the no-effect-by-peripheral-features assumption no longer holds. We see that the bias by direct substitution exacerbates greatly (about 140% over estimate). Also, 2SL type 1 exhibits a substantial bias too (about 50% over-estimate). Importantly, 2SL type 2 shows little bias.

Finally, we use scenario (4) to illustrate that it is possible for the estimate by direct substitution to have the wrong sign (which further contrasts the attenuation bias in classical measurement

[6]Some accuracy rates in the literature are 70% (Malik et al. 2019), 72% to 81% (Troncoso and Luo 2021), 92% (Liu et al. 2020), 95% (Zhang et al. 2021).

Table 3: Estimates of δ in Monte Carlo - Main Results

(1): $\boldsymbol{x}_i \perp\!\!\!\perp \boldsymbol{z}_i,\ \varepsilon_i \perp\!\!\!\perp \boldsymbol{z}_i$

	Mean			**RMSE**		
Sample size (n):	**2500**	**5000**	**10000**	**2500**	**5000**	**10000**
Oracle	1.025	1.011	1.018	0.135	0.097	0.073
Direct Substitution	1.45	1.462	1.486	0.544	0.511	0.507
2SL - type 1	1.016	1.031	1.025	0.252	0.169	0.106
2SL - type 2	0.985	0.981	0.979	0.22	0.16	0.112

(2): $\boldsymbol{x}_i \not\perp\!\!\!\perp w_i,\ \varepsilon_i \perp\!\!\!\perp \boldsymbol{z}_i$

Oracle	0.991	0.995	1.011	0.186	0.144	0.113
Direct Substitution	0.687	0.803	0.808	0.433	0.328	0.252
2SL - type 1	0.951	0.994	1.017	0.258	0.199	0.144
2SL - type 2	0.96	0.992	0.985	0.336	0.184	0.151

(3): $\boldsymbol{x}_i \perp\!\!\!\perp \boldsymbol{z}_i,\ \varepsilon_i \not\perp\!\!\!\perp \boldsymbol{z}_i$

Oracle	1.015	1.006	1.011	0.124	0.084	0.076
Direct Substitution	2.418	2.456	2.47	1.456	1.474	1.482
2SL - type 1	1.519	1.529	1.511	0.586	0.561	0.528
2SL - type 2	0.998	1.006	1.00	0.21	0.146	0.116

(4): $\boldsymbol{x}_i \not\perp\!\!\!\perp w_i,\ \varepsilon_i \not\perp\!\!\!\perp \boldsymbol{z}_i$

Oracle	0.981	0.986	1.005	0.196	0.173	0.11
Direct Substitution	-0.35	-0.35	-0.34	1.388	1.368	1.349
2SL - type 1	0.74	0.815	0.827	0.375	0.296	0.216
2SL - type 2	0.918	0.975	0.99	0.314	0.189	0.156

Notes: Based on estimates of δ from 100 Monte Carlo samples. True value of $\delta = 1$. Red color indicates bias of magnitude $\geqslant 10\%$. Standard errors of means are not reported but an upper bound can be easily calculated as RMSE/$\sqrt{100}$.

Table 4: Estimates of δ in Monte Carlo - Whether to Split Labeled Subsample

(1): GoogLeNet with scorecard for feature extraction

	Mean			**RMSE**		
Sample size (n):	**2500**	**5000**	**10000**	**2500**	**5000**	**10000**
2SL - type 1	1.016	1.031	1.025	0.252	0.169	0.106
2SL - type 1 - no split	0.904	0.948	0.977	0.234	0.164	0.096
2SL - type 2	0.985	0.981	0.979	0.22	0.16	0.112
2SL - type 2 - no split	0.932	0.956	0.977	0.203	0.151	0.115

(2): GoogLeNet with k-NN for feature extraction

2SL - type 1	1.027	1.022	1.031	0.352	0.223	0.12
2SL - type 1 - no split	0.841	0.856	0.897	0.303	0.216	0.148
2SL - type 2	0.985	0.973	0.976	0.239	0.167	0.111
2SL - type 2 - no split	0.900	0.912	0.933	0.228	0.174	0.132

Notes: Based on estimates of δ from 100 Monte Carlo samples. True value of $\delta = 1$. Red color indicates bias of magnitude $\geqslant 10\%$. Standard errors of means are not reported but an upper bound can be easily calculated as RMSE/$\sqrt{100}$.

errors). The setting follows scenario (2) except that it adds a correlation of about -0.5 between ε_i and the peripheral feature introduced in scenario (3). We see direct substitution now gives a negative estimate of δ. 2SL type 1 has the correct sign but shows sizable biases. 2SL type 2 shows a moderate bias at $n = 2500$ and little biases at larger n.

To summarize, Monte Carlo results show that direct substitution can result in large biases in either direction. Both types of 2SL perform well when the no-effect-by-peripheral-features assumption holds. When the assumption does not hold, 2SL type 1 shows sizable biases while 2SL type 2 performs well.

6.4 Results on splitting $\mathcal{S}$

An important detail of 2SL is splitting the labeled subsample $\mathcal{S}$. Table 4 examines what happens if we omit the split, that is, we use the entirety of $\mathcal{S}$ to train both machine learning models in 2SL. The upper panel of the table assumes scenario (1) in Table 3. We see that the impact of splitting is most noticeable at small s. At $n = 2500$ ($s = 500$), splitting leads to noticeably smaller biases but slightly larger RMSEs. So, in this case, splitting trades off some variance for lower bias.

The degree of bias caused by not splitting $\mathcal{S}$ depends on the 1st-step machine learning model $\boldsymbol{f}$. So far, the image recognition has used scorecard algorithm (see Section 6.2), which is a relatively global method (i.e., prediction for i pools information across all training data points instead of just the points close to i). The lower panel of Table 4 switches from scorecard to k-NN, with everything else following the upper panel. We see that omitting the splitting leads to larger biases. The result here is consistent with our discussion in Section 5.3. That is, splitting is more important when the 1st-step machine learning model uses a more local than global method.

Table 5: Estimates of δ in Monte Carlo - Choice of 2nd-Step Machine Learning Model

(1): $\boldsymbol{x}_i \perp\!\!\!\perp \boldsymbol{z}_i$, $\varepsilon_i \perp\!\!\!\perp \boldsymbol{z}_i$

	Mean			**RMSE**		
Sample size (n):	**2500**	**5000**	**10000**	**2500**	**5000**	**10000**
2SL - type 1 - neural net	1.016	1.031	1.025	0.252	0.169	0.106
2SL - type 1 - logit	0.956	0.982	0.989	0.225	0.154	0.097
2SL - type 2 - neural net	0.985	0.981	0.979	0.22	0.16	0.112
2SL - type 2 - logit	1.033	1.013	1.006	0.236	0.163	0.12
(2): $\boldsymbol{x}_i \perp w_i$, $\boldsymbol{x}_i \not\perp\!\!\!\perp w_i$, $\varepsilon_i \perp\!\!\!\perp \boldsymbol{z}_i$						
2SL - type 1 - neural net	0.919	0.998	1.031	0.241	0.152	0.107
2SL - type 1 - logit	0.817	0.866	0.868	0.279	0.192	0.165
2SL - type 2 - neural net	0.917	1.016	0.988	0.259	0.174	0.1
2SL - type 2 - logit	0.812	0.848	0.818	0.311	0.223	0.206
(3): $\boldsymbol{x}_i \perp w_i$, $\boldsymbol{x}_i \not\perp\!\!\!\perp w_i$, $\varepsilon_i \not\perp \boldsymbol{z}_i$						
2SL - type 1 - neural net	1.274	1.237	1.183	0.369	0.277	0.214
2SL - type 1 - logit	1.295	1.37	1.341	0.373	0.401	0.362
2SL - type 2 - neural net	0.974	0.998	0.991	0.216	0.156	0.112
2SL - type 2 - logit	0.841	0.84	0.821	0.276	0.224	0.214

Notes: Based on estimates of δ from 100 Monte Carlo samples. True value of $\delta = 1$. Red color indicates bias of magnitude $\geqslant 10\%$. Standard errors of means are not reported but an upper bound can be easily calculated as RMSE/$\sqrt{100}$.

6.5 Results on 2nd-step machine learning model

Another important detail of 2SL is the choice of 2nd-step machine learning model. Table 5 compares two choices: a shallow neural net and logit regression (note this logit regression is not the econometric model in equation (14) but the 2nd step in 2SL). The shallow neural net has been used for all the Monte Carlo results so far. It has 11 or 12 input nodes depending on the type of 2SL. There is one hidden layer with tanh activation. The number of hidden nodes $\in \{0, 4, 16\}$. The regularization factor $\in \{0, 0.001, 0.002, 0.004, 0.008, 0.016\}$. Both hyperparameters are chosen via 5-fold cross validation on $\mathcal{S}_1$. Given a choice of hyperparameters, training minimizes cross-entropy loss until the decrease in loss is sufficiently small.[7]

The upper panel of Table 5 assumes scenario (1) in Table 3, where $\boldsymbol{x}_i \perp\!\!\!\perp \boldsymbol{z}_i$ and $\varepsilon_i \perp\!\!\!\perp \boldsymbol{z}_i$. In this simplest scenario, we see that the neural net and logit perform in the same ballpark. The middle panel of Table 5 examines a more complex scenario. It follows the upper panel except for the specification of $x_{i,2}$. We let $x_{i,2} = x_{i,1} \cdot e^{-0.5+w_i+0.25\xi_i}$ with $\xi_i \sim N(0,1)$. In this case, $\mathbf{corr}(\boldsymbol{x}_i, w_i) = \mathbf{0}$ but $\mathbf{corr}(x_{i,2}/x_{i,1}, w_i) \simeq 0.88$. We see that the logit now results in sizable biases in both types of 2SL. Intuitively, this is because the logit model lacks the flexibility to capture the ratio $x_{i,2}/x_{i,1}$.

[7] We use the `trainNetwork` function in Matlab. Inputs are standardized.

So far, Table 5 has maintained the no-effect-by-peripheral-features assumption because $\varepsilon_i \perp\!\!\!\perp \boldsymbol{z}_i$. The bottom panel of Table 5 drops this assumption. It follows the middle panel except that it adds a correlation between ε_i and a peripheral feature of $\boldsymbol{z}_i$ (as we have done in scenario (3) of Table 3). We see that the 2SL type 1 shows sizable biases regardless of which 2nd-step machine learning model is used, which is expected. For 2SL type 2, the logit shows biases while neural net works well. Overall, the results here are consistent with our discussion in Section 5.3.

7 Empirical Application

We collect a popular Amazon review dataset and set up an empirical exercise that is simple yet representative in research. The main result is that while both direct substitution and 2SL type 1 show sizable biases compared to the oracle, 2SL type 2 removes most of the biases. The result highlights the third source of bias: peripheral features in unstructured data create a correlation between the extracted focal feature and the residual in the econometric model (see Section 3.1, particularly Figure 1).

7.1 Data and setup

We use a publicly available dataset of Amazon reviews in the fine food category (McAuley and Leskovec 2013).[8] We pick this dataset because textual analysis of product reviews is a popular exercise in research.[9] Further, two prior papers on the same topic as ours also use product reviews in their empirical exercises. Qiao and Huang (2021) use Amazon reviews while Yang et al. (2018) use TripAdvisor reviews. A final reason is that the dataset allows us to include an oracle as the benchmark, as we will make clear in a moment.

The dataset includes 568,454 reviews for fine food products on Amazon. For each review, it includes the review text, the rating (1 to 5 stars), and numbers of up votes and down votes by other users. An upvote means an user finds the review helpful. We focus on reviews that received at least one vote and are neither too short (bottom 5%) nor too long (top 5%). There are 281,767 reviews left, which form our Amazon review collection.

Although the exact data differ, our econometric setup roughly follows that of Qiao and Huang (2021).[10] We consider a researcher who wishes to study how the helpfulness of a review relates to the review sentiment. The helpfulness y_i is defined as 1 if all votes received by review i are upvotes and 0 otherwise. The sentiment w_i is defined as 1 if the review rating has at least four stars and 0 otherwise. Thus, the sentiment is known for all reviews. But we will pretend not so and try

[8] https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews. Last accessed in May 2023.

[9] See, e.g., Liu et al. 2019, Timoshenko and Hauser 2019, Shi et al. 2021, Puranam et al. 2021, Chakraborty et al. 2022.

[10] Both their and our papers study Amazon review data, but the product category and sample period most likely differ. In terms of econometric setup, there is one difference that we define $y_i = 1$ if all votes are upvotes while they require only 1 upvote. We do so because there are more upvotes than downvotes in our data. The original definition would lead to a substantial skew in our data, with 89% of y_i being 1. However, our main results still hold if the original definition is used.

Table 6: Descriptive Statistics of Amazon Review Data

		mean	s.d.	min	Q1	Q2	Q3	max
y_i	Helpfulness	0.620	0.485	0.0	0.0	1.0	1.0	1.0
w_i	Sentiment	0.717	0.450	0.0	0.0	1.0	1.0	1.0
$x_{i,1}$	Log number of votes	0.865	0.896	0.0	0.0	0.693	1.386	6.828
$x_{i,2}$	Log number of words	4.228	0.682	3.044	3.689	4.190	4.718	5.979

extracting sentiment from review text, which we will give the details for in a moment. We consider the logit model:

$$y_i = \begin{cases} 1, & \text{if } \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \delta w_i + \varepsilon_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

The explanatory variable $x_{i,1}$ is the log number of votes received by review i and $x_{i,2}$ is the log number of words in review i. Coefficient δ is our main interest.

We consider a researcher who has a sample of n reviews drawn from our Amazon review collection (281,767 reviews). The sample size n will be far smaller than the collection size, so that we may repeat the sampling many times to examine the distribution of δ's estimate. The researcher observes w_i for a subsample of s reviews and do not observe w_i for the rest $n-s$ reviews. We compare how different approaches utilize the $n-s$ reviews to estimate δ. We examine $n \in \{2500, 5000, 10000, 20000\}$ while fixing $s = n/5$. The fact that we actually know w_i for all reviews allows us to include an oracle for comparison.

Table 6 provides summary statistics for the 281,767 reviews.

7.2 Feature extraction

We describe the machine learning model for sentiment extraction from review text (i.e., predicting w_i). There are two common ways of text representation in machine learning. The first way is one-hot encoding, where a dummy variable is created for each unique word. This representation is practical only for very large training data. The other way is to use pre-trained embeddings. We use Google's pre-trained word2vec embeddings. We average the embeddings of all words in a review. As Google's word2vec embeddings are 300-dimensional vectors, the average embedding of a review is a 300 by 1 vector.

To project the average embedding into sentiment, we study two common machine learning models: k-nearest neighbors (k-NN) and logistic regression. In k-NN the hyperparameter is tuned using 5-fold cross validation. The logistic regression has no hyperparameter. For both models, the extracted sentiment $\widehat{w}_i$ is a probability between 0 and 1. It is important to note that the logistic regression here is different from the logistic model in equation (15). The former is a part of the sentiment extraction algorithm. The latter is the econometric model to describe the relation between review helpfulness and review sentiment.

Table 7 shows the out-of-sample accuracy. The k-NN performs better with smaller training

Table 7: Out-of-sample Accuracy for Sentiment Extraction

	Accuracy (%)			
Training sample size (s)	500	1000	2000	4000
k-NN	72.2	72.8	73.4	74.0
Logistic regression	71.7	72.6	74.8	77.1

data, whereas the logistic regression performs better with larger training data. Overall, the two models are on par of each other.

7.3 Estimates of δ

The main results from our empirical application are given in Table 8. As described before, we consider a researcher who has a sample of n reviews drawn from our Amazon review collection, where $s = n/5$ reviews have labeled sentiment. Columns of the table correspond to different sample sizes. The upper panel uses logit with word2vec to extract sentiment, while the lower panel uses k-NN with word2vec (see Section 7.2). Reported are estimates of δ averaged across 100 samples.

The rows of Table 8 correspond to the oracle, direct substitution, and two types of 2SL. The oracle directly runs a logit regression of y_i on $(\boldsymbol{x}_i, w_i)$ across the $n - s$ reviews. Direct substitution replaces w_i with $\widehat{w}_i$ in the logit regression. 2SL follows Section 5, and the second machine learning model is the same shallow neural net used in Section 6. We use red color to mark bias of magnitude $\geqslant 10\%$ compared to the oracle.

Compared to the oracle, both direct substitution and 2SL type 1 have sizable upward biases. In other word, both approaches exaggerate the association between sentiment and helpfulness. For direct substitution, the extent of bias ranges from 40% ($= 2.444/1.745-1$) to 228% ($= 5.732/1.745-1$). For 2SL type 1, the extent of bias tends to be smaller but still substantial, ranging from 19% ($= 2.076/1.740-1$) to 41% ($= 2.457/1.745-1$). In contrast, 2SL type 2 shows much smaller biases ($\leqslant 3\%$).

For a graphical illustration, Figure 3 plots the histograms of the estimates of δ at $n = 20000$. For reference, we use a red vertical line to represent the oracle's mean estimate. We see 2SL type 2 has virtually no bias, while direct substitution and 2SL type 1 show clear biases. We also see that 2SL type 2 has a smaller variance than the other two estimators.

7.4 Sources of Bias

Results so far indicate sizable biases for direct substitution. Below we investigate sources of bias. Following the discussion in Section 3, here we focus on the out-of-sample properties of the machine-extracted sentiment $\widehat{w}_i$ to find clues why the biases arise. Recall that in direct substitution, the machine learning model for sentiment extraction is trained using the s reviews. So we examine properties of $\widehat{w}_i$ among the $n - s$ reviews (thus out-of-sample).

Figure 4 shows the results. From top to bottom, the three rows examine three statistics of $\widehat{w}_i$: (i) prediction RMSE, (ii) correlation with extraction error $\widehat{v}_i \equiv w_i - \widehat{w}_i$, and (iii) correlation with

Table 8: Estimates of δ in Amazon Review Application

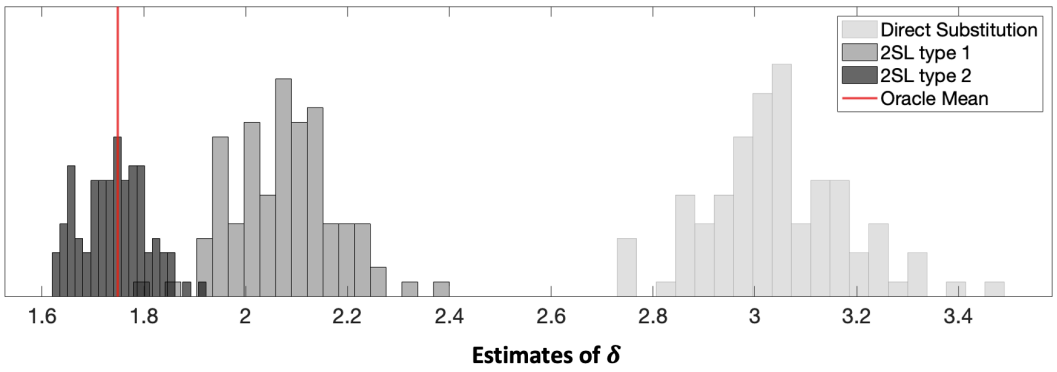
(1) logit with word2vec to extract sentiment

	Mean Estimate of δ			
Sample size (n)	**2500**	**5000**	**10,000**	**20,000**
Oracle	1.745 (.01)	1.741 (.01)	1.739 (.00)	1.740 (.00)
Direct Substitution	5.732 (.10)	4.487 (.05)	3.630 (.03)	3.045 (.02)
2SL type 1	2.192 (.04)	2.103 (.03)	2.105 (.02)	2.076 (.01)
2SL type 2	1.768 (.02)	1.747 (.02)	1.747 (.01)	1.742 (.01)

(2) k-NN with word2vec to extract sentiment

Oracle	1.745 (.01)	1.741 (.01)	1.739 (.00)	1.740 (.00)
Direct Substitution	2.444 (.08)	2.447 (.05)	2.585 (.05)	2.611 (.04)
2SL type 1	2.457 (.14)	2.222 (.05)	2.194 (.03)	2.192 (.02)
2SL type 2	1.790 (.02)	1.760 (.02)	1.750 (.01)	1.751 (.01)

Notes: Reported are the mean estimates of δ across 100 samples from Amazon review collection. Numbers in parentheses are standard errors. Red color indicates bias of magnitude $\geqslant 10\%$ (compared to oracle).



Notes: The setting is same as in Table 8, upper panel, last column.

Figure 3: Distribution of Estimates for δ in Amazon Review Application

the residual ε_i in the econometric model. The two plots in each row differ in how they extract sentiment: the left plot uses logit and word2vec while the right plot uses k-NN and word2vec. In each plot, the curve shows how the statistic changes with sample size n. Each reported value is averaged across 100 samples (drawn from our Amazon review collection). Vertical bars represent standard deviations across the 100 samples (so standard errors are 1/10 the bar lengths).

We start with the top row of Figure 4 that shows the prediction performance of $\widehat{w}_i$. We see that the RMSE decreases as the number training examples increases. This result is expected. It shows that machine learning models strive for prediction accuracy and do better so with more training data.

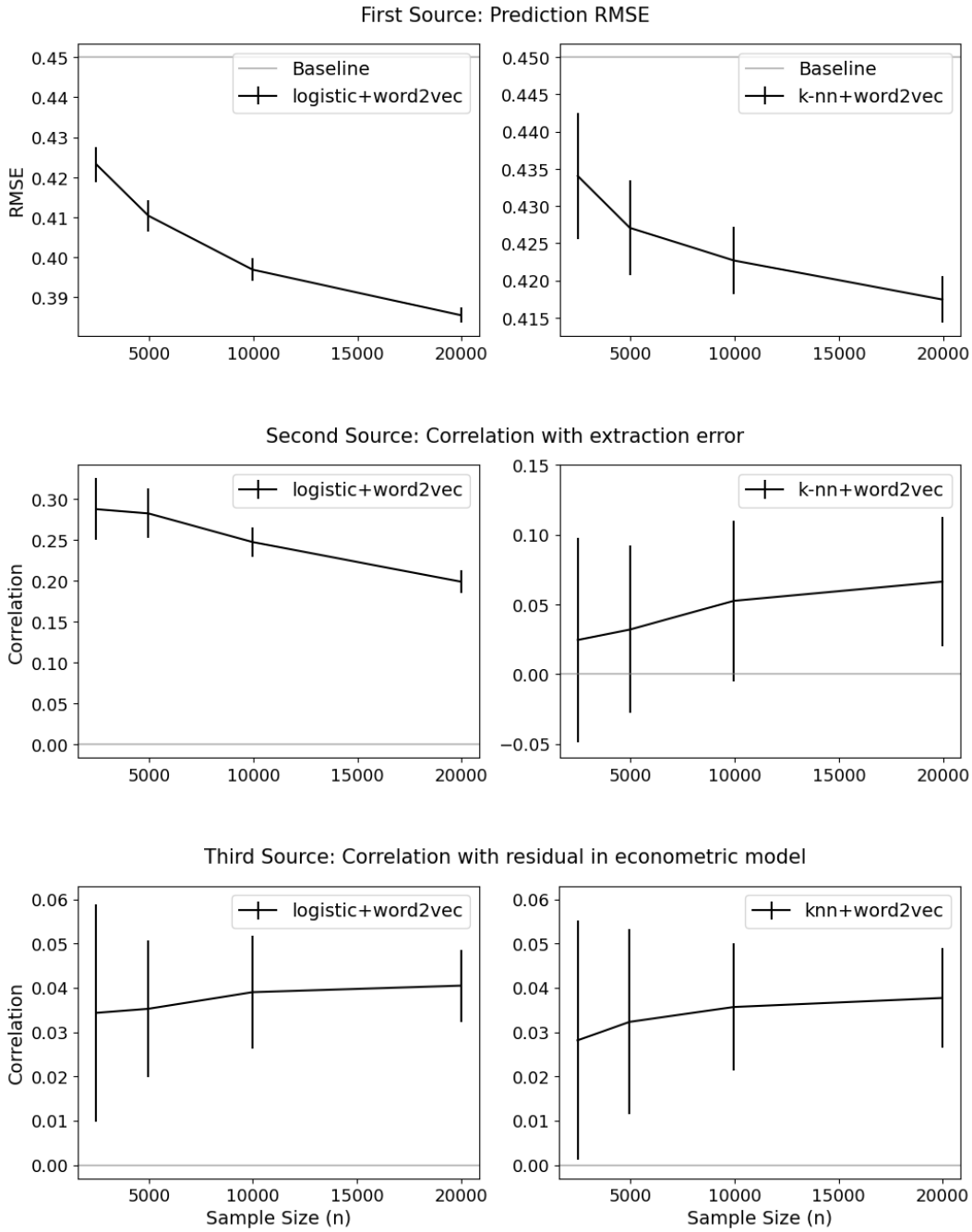
The middle row of Figure 4 shows the correlation between $\widehat{w}_i$ and its extraction error $\widehat{v}_i$. From discussion in Section 3.1, we know that in general this correlation is a source of bias. For the Amazon application here, we see $\mathbf{corr}(\widehat{w}_i, \widehat{v}_i) > 0$ in most of the samples we have drawn. In other words, the correlation is systematically positive, which explains the positive bias in δ. We note that the magnitude of correlations in the right plot appears small (around 0.05), but the mapping from correlation to bias is not simple and such small correlations can still manifest as sizable biases in the estimate of δ.[11]

The bottom row of Figure 4 examines the correlation between $\widehat{w}_i$ and residual ε_i in the econometric model. From discussion in Section 3.1, we know that in general this correlation is an important source of bias. A technical issue here is that we do not know ε_i. Unlike in linear regression, ε_i in a logit regression cannot be exactly backed out. As a proxy, we use its expected value $\widehat{\varepsilon}_i \equiv \mathbb{E}(\varepsilon_i|\boldsymbol{x}_i, y_i, \boldsymbol{\beta}, \delta)$.[12] We see from Figure 4 that $\mathbf{corr}(\widehat{w}_i, \widehat{\varepsilon}_i) > 0$ in most of the samples we have drawn. In other words, the correlation is systematically positive, which explains the positive bias in δ. We note the magnitude of correlations is small (around 0.03 - 0.04). However, as explained above, such small correlations can still manifest as sizable biases in the estimate of δ.

Overall, Figure 4 shows that while machine learning models strive for prediction accuracy, they are not designed to provide desired properties for econometric estimation. The purpose of 2SL is to account for possible undesired properties. Both types of 2SL account for $\mathbf{corr}(\widehat{w}_i, \widehat{v}_i) \neq 0$. But 2SL type 1 does not account for $\mathbf{corr}(\widehat{w}_i, \varepsilon_i) \neq 0$. Given that both correlations are present in this Amazon review application (as shown in Figure 4), 2SL type 1 can only partially correct the bias in δ. This is consistent with what we have seen in Table 8.

[11] The relation between the correlation and bias has a closed-form expression if we were to consider a simple linear regression: $y_i = \beta + \delta w_i + \varepsilon_i$. With direct substitution, we have $y_i = \beta + \delta\widehat{w}_i + \delta\widehat{v}_i + \varepsilon_i$. The bias in δ is $\mathbf{cov}(\widehat{w}_i, \delta\widehat{v}_i + \varepsilon_i)/\sigma^2(\widehat{w}_i)$. So, the impact of $\mathbf{corr}(\widehat{w}_i, \widehat{v}_i)$ on the bias scales with $\sigma(\widehat{v}_i)/\sigma(\widehat{w}_i)$. In our application, $\sigma(\widehat{w}_i)$ is much smaller than 1.

[12] The conditional expectation has a closed-form expression. Let $t_i \equiv \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \delta w_i$. Then $y_i = 1$ iff $t_i + \varepsilon_i > 0$. One can show $\mathbb{E}(\varepsilon_i|\boldsymbol{x}_i, y_i = 1, \boldsymbol{\beta}, \delta) = (1 + e^{-t_i}) \cdot \log(1 + e^{t_i}) - t_i$ and $\mathbb{E}(\varepsilon_i|\boldsymbol{x}_i, y_i = 0, \boldsymbol{\beta}, \delta) = -(1 + e^{t_i}) \cdot \log(1 + e^{-t_i}) - t_i$. Here, we take the values of $\boldsymbol{\beta}$ and δ from a logit regression of equation (15) using all the 281,767 reviews in our Amazon review collection.



Notes: The plots examine properties of the extracted sentiment to find the sources of bias in the estimate of δ. Each reported value is averaged across 100 samples. Vertical bars show standard deviations across the 100 samples (so standard errors are 1/10 of the bar lengths).

Figure 4: Sources of Bias in Amazon Review Application

8 Conclusion

This article examines an emerging theme in research that combines machine learning and econometric models to study features of unstructured data. We unpack potential econometric bias that arises from the extraction errors by machine learning algorithms. A key emphasis of our exploration is the distinction between extraction errors and measurement errors. We pay special attention to "peripheral" features of unstructured data that fall outside the focus of the econometric model, yet can be easily picked up by the machine learning model. The exploration highlights the fact machine learning methods were developed without econometric exercises in mind, and therefore, simply "stacking" machine learning and econometric models together can lead to considerably biased estimates. One should carefully adapt the two to each other in order to draw correct insights.

We propose an approach to mitigate the bias. The approach does not modify or restrict the machine learning algorithm used to extract features, which gives researchers flexibility in choosing the algorithm suitable for their unstructured data (and accommodates cases where researchers have little control over the algorithm, such as when using pre-trained models). Given this premise, our approach focuses on accounting for the extraction errors when estimating econometric models. It extends previous works in this area (Qiao and Huang 2021, Fong and Tyler 2020, Yang et al. 2018) by being applicable to general econometric models as well as removing the restrictive assumption that excludes potential effects of peripheral features.

Although the proposed approach has made important extensions, it is still important to be aware of its boundaries. First, the studies so far, including ours, aim to bring our estimate close to that of an oracle who directly observes the feature of interest in unstructured data. Put differently, the benchmark is the oracle's estimate. However, the oracle's estimate may have its own bias if the econometric model is mis-specified (e.g., due to unaccounted endogeneity). We do not attempt to correct this bias. Second, the studies so far have assumed that manual labeling can correctly classify the feature of interest from unstructured data. This assumption is reasonable in many applications. However, in some applications there can be sizable errors in manual labels. One way to mitigate the labeling errors is to use multiple labelers, but it also increases costs. Accounting for possible errors in manual labeling is a very important topic for future research.

Appendix

A: Details for the IV Example

We provide additional details for readers who wish to replicate the Monte Carlo example in Section 3.2. Recall that the IV setup requires: $\mathbf{corr}(w_i, \varepsilon_i) \neq 0$, $\mathbf{corr}(w_i, t_i) \neq 0$, and $\mathbf{corr}(t_i, \varepsilon_i) = 0$. A standard method to achieve these conditions is to generate w_i as a function of t_i and ε_i. However, this method is precluded by our setup where w_i is taken from CIFAR-10. So, we use a different method where we use re-ordering to have both t_i and ε_i correlated with w_i, and yet keep t_i and ε_i uncorrelated with each other.

Specifically, we first define a rank function. For any given $\boldsymbol{\xi} \equiv (\xi_1, ..., \xi_n)'$, let $\text{rank}(i, \boldsymbol{\xi})$ denote the ranking of ξ_i in $\boldsymbol{\xi}$ under an ascending sort, with ties broken randomly. For a toy example, let $\boldsymbol{\xi} = (6, 4, 1, 1, 5)'$, then $\text{rank}(1, \boldsymbol{\xi}) = 5$ and $\text{rank}(2, \boldsymbol{\xi}) = 3$. Next, we generate t_i^* from uniform Bernoulli distribution, generate $\varepsilon_i^* \sim N(0, 1)$, and also generate $\iota_i \sim N(0, 1)$. Let $c_i = \rho_1 \cdot t_i^* + \rho_2 \cdot \varepsilon_i^* + \iota_i$ for some ρ_1, $\rho_2 > 0$. Finally, for each i, let $t_i = t_j^*$ and $\varepsilon_i = \varepsilon_j^*$ where j satisfies $\text{rank}(j, \boldsymbol{c}) = \text{rank}(i, \boldsymbol{w})$. Note $\mathbf{corr}(t_i, \varepsilon_i) = \mathbf{corr}(t_i^*, \varepsilon_i^*) = 0$. We set $\rho_1 = 1$ and $\rho_2 = 0.3$, which results in $\mathbf{corr}(w_i, t_i) \simeq 0.35$ and $\mathbf{corr}(w_i, \varepsilon_i) \simeq 0.2$.

B: Peripheral feature in Monte Carlo studies

We provide technical details on the peripheral feature in Section 6, for readers who wish to replicate our Monte Carlo results. Below we denote this peripheral feature as u_i. The feature u_i was first introduced in Table 3: scenario (3), where u_i enters the logistic residual ε_i. The Monte Carlo setup requires $u_i \perp\!\!\!\perp w_i$ so that $\varepsilon_i \perp\!\!\!\perp w_i$ and the logistic equation (14) remains correctly specified for the oracle. (If $u_i \not\perp\!\!\!\perp w_i$, 2SL will still reduce the bias of our estimate as compared to the oracle's, but the oracle's estimate will have a bias itself.)

We construct u_i from the prominence of yellow color in an image. Specifically, under the RGB color coding, an image is characterized by three channels: red, green, and blue. In CIFAR-10, each image has 32×32 pixels. So each channel is represented by a 32×32 matrix. For each image i, we calculate the yellow prominence as the sum of two non-blue channels divided by the sum of all three channels. Next, we normalize the yellow prominence so that it follows $N(0, 1)$ conditional on $w_i \in \{0, 1\}$. Specifically, for each image i we let r_i be the percentile rank of i's yellow prominence among $\{j : w_j = w_i\}$. Then, let $u_i = \Phi^{-1}(r_i)$, where Φ is the standard normal cdf. Note that $u_i | w_i \sim N(0, 1)$ and thus $u_i \perp\!\!\!\perp w_i$. Finally, we let u_i enter the logistic residual ε_i as follows. Let L be the logistic cdf and $\iota_i \sim N(0, 1)$. We let $\varepsilon_i = L^{-1} \circ \Phi(\rho \cdot u_i + \sqrt{1 - \rho^2} \cdot \iota_i)$. Here, parameter $\rho \in (0, 1)$ calibrates the correlation between u_i and ε_i.

References

[1] Ascarza, E. and Israeli, A., 2022. Eliminating unintended bias in personalized policies using Bias Eliminating Adapted Trees (BEAT). PNAS, 119(11).

[2] Athey, S. and Imbens, G., 2019. Machine Learning Methods That Economists Should Know About. Annual Review of Economics, 11:685–725.

[3] Berger, J. and Milkman, K.L., 2012. What makes online content viral?. Journal of marketing research, 49(2), pp.192-205.

[4] Biddle, J.E. and Hamermesh, D.S., 1998. Beauty, productivity, and discrimination: Lawyers' looks and lucre. Journal of labor Economics, 16(1), pp.172-201.

[5] Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., 2017. Enriching word vectors with subword information. Transactions of the association for computational linguistics, 5, pp.135-146.

[6] Burtch, G., He, Q., Hong, Y. and Lee, D., 2021. How do peer awards motivate creative content? Experimental evidence from Reddit. Management Science

[7] Chakraborty, I., Kim, M. and Sudhir, K., 2021. EXPRESS: Attribute Sentiment Scoring with Online Text Reviews: Accounting for Language Structure and Missing Attributes. Journal of Marketing Research, p.00222437211052500.

[8] Chen, Xiaohong, 2007. Large Sample Sieve Estimation of Semi-Nonparametric Models. Handbook of Econometrics, 6B.

[9] Chen, Xiaohong, 2010. Penalized Sieve Estimation and Inference of Semi-nonparametric Dynamic Models: A Selective Review. Advances in Economics and Econometrics, Cambridge University Press.

[10] Chen, Xiaohong, Hong, H., and Tamer, E., 2005, Measurement Error Models with Auxiliary Data. Review of Economic Studies, 72(2).

[11] Chen, Xiaohong, Hong, H., and Nekipelov, D., 2011, Nonlinear Models of Measurement Errors. Journal of Economic Literature, 49(4).

[12] Chintagunta, P.K., Gopinath, S. and Venkataraman, S., 2010. The effects of online user reviews on movie box office performance: Accounting for sequential rollout and aggregation across local markets. Marketing science, 29(5), pp.944-957

[13] Chouldechova, A. and G'Sell, M., 2017. Fairer and more accurate, but for whom?. arXiv preprint arXiv:1707.00046.

[14] Clarke, J., Chen, H., Du, D. and Hu, Y.J., 2020. Fake news, investor attention, and market reaction. Information Systems Research, 32(1), pp.35-52.

[15] Feldman, R., Govindaraj, S., Livnat, J. and Segal, B., 2010. Management's tone change, post earnings announcement drift and accruals. Review of Accounting Studies, 15(4), pp.915-953.

[16] Fong, C. and Tyler, M., 2020. Machine Learning Predictions as Regression Covariates. Political Analysis, 29(4).

[17] Gabel, S. and Timoshenko, A., 2022. Product choice with large assortments: A scalable deep-learning model. Management Science, 68(3), pp.1808-1827.

[18] Ganchev, K., Graca, J., Gillenwater, J. and Taskar, B., 2010. Posterior regularization for structured latent variable models. The Journal of Machine Learning Research, 11, pp.2001-2049.

[19] Godes, D. and Mayzlin, D., 2004. Using online conversations to study word-of-mouth communication. Marketing science, 23(4), pp.545-560.

[20] Gunarathne, P., Rui, H. and Seidmann, A., 2022. Racial Bias in Customer Service: Evidence from Twitter. Information Systems Research, 33(1), pp.43-54.

[21] Hartmann, J., Heitmann, M., Schamp, C. and Netzer, O., 2021. The power of brand selfies. Journal of Marketing Research, 58(6), pp.1159-1177.

[22] Hu, Z., Ma, X., Liu, Z., Hovy, E. and Xing, E., 2016. Harnessing deep neural networks with logic rules. arXiv preprint arXiv:1603.06318.

[23] Koch, G., Zemel, R. and Salakhutdinov, R., 2015. Siamese Neural Networks for One-Shot Image Recognition. ICML deep learning workshop vol.2.

[24] Komer, Brent, James Bergstra, and Chris Eliasmith. Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. ICML workshop on AutoML. Vol. 9. Austin, TX: Citeseer, 2014.

[25] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

[26] Lake, B.M., Salakhutdinov, R.R. and Tenenbaum, J., 2013. One-shot learning by inverting a compositional causal process. Advances in neural information processing systems, 26.

[27] Li, Y. and Xie, Y., 2020. Is a picture worth a thousand words? An empirical study of image content and social media engagement. Journal of Marketing Research, 57(1), pp.1-19

[28] Liu, L., Dzyabura, D. and Mizik, N., 2020. Visual listening in: Extracting brand image portrayed on social media. Marketing Science, 39(4), pp.669-686.

[29] Malik, N., Singh, P.V. and Srinivasan, K., 2019. A Dynamic Analysis of Beauty Premium. Available at SSRN 3208162.

[30] Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[31] Newey, W. and McFadden, D., 1994. Large Sample Estimation and Hypothesis Testing. Handbook of Econometrics, vol.4.

[32] Nian, T., Hu, Y. and Chen, C., 2021. Examining the Impact of Television-Program-Induced Emotions on Online Word-of-Mouth Toward Television Advertising. Information Systems Research, 32(2), pp.605-632.

[33] Puranam, D., Kadiyali, V. and Narayan, V., 2021. The Impact of Increase in Minimum Wages on Consumer Perceptions of Service: A Transformer Model of Online Restaurant Reviews. Marketing Science, 40(5), pp.985-1004.

[34] Qiao, M. and Huang, K.W., 2021. Correcting misclassification bias in regression models with variables generated via data mining. Information Systems Research, 32(2), pp.462-480.

[35] Shin, D., He, S., Lee, G.M., Whinston, A.B., Cetintas, S. and Lee, K.C., 2020. Enhancing social media analysis with visual data analytics: A deep learning approach. MIS Quarterly, 44(4), pp.1459-1492.

[36] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), pp.1929-1958.

[37] Sun, C., Adamopoulos, P., Ghose, A. and Luo, X., 2021. Predicting Stages in Omnichannel Path to Purchase: A Deep Learning Model. Information Systems Research.

[38] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[39] Timoshenko, A. and Hauser, J.R., 2019. Identifying customer needs from user-generated content. Marketing Science, 38(1), pp.1-20.

[40] Troncoso, I. and Luo, L., 2020. Look the Part? The Role of Profile Pictures in Online Labor Markets. The Role of Profile Pictures in Online Labor Markets (October 12, 2020).

[41] Vickery, S.K., Droge, C., Stank, T.P., Goldsby, T.J. and Markland, R.E., 2004. The performance implications of media richness in a business-to-business service environment: Direct versus indirect effects. Management Science, 50(8), pp.1106-1119.

[42] White, Harlbert, 1990. Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings. *Neural Networks*, 3.

[43] Wooldridge, J., 2002. Econometric Analysis of Cross Section and Panel Data. MIT Press.

[44] Wu, J., Zheng, Z. and Zhao, J.L., 2021. FairPlay: Detecting and Deterring Online Customer Misbehavior. Information Systems Research, 32(4), pp.1323-1346.

[45] Zhang, S., Lee, D., Singh, P.V. and Srinivasan, K., 2021. What makes a good image? Airbnb demand analytics leveraging interpretable image features. Management Science.

[46] Zhang, Z. and Neill, D.B., 2016. Identifying significant predictive bias in classifiers. arXiv preprint arXiv:1611.08292.