Label Denoising and Counterfactual Explanation with A Plug and Play Framework

Wenting Qi
Department of Computer Science
University at Albany, SUNY
Albany, New York, USA
wqi@albany.edu

Charalampos Chelmis*†
Department of Computer Science
University at Albany, SUNY
Albany, New York, USA
cchelmis@albany.edu

***Abstract*—Most supervised classification methods assume perfect training data, although this is not usually the case in the real–world. Meanwhile, counterfactual data generation approaches have emerged as a way to provide post–hoc explanation of decisions made by classification models. However, such approaches highly rely on the classification model output since different outputs lead to alternative, or even contradicting explanations. This work proposes a plug–and–play framework to learn a robust classification model in the presence of noisy labeled data and provide actionable suggestions for undesirable decisions (e.g., loan application rejection) made by a given classification model. The framework's generalizability is demonstrated by considering alternative noisy label detection and counterfactual explanation methods, as well as diverse supervised classification models. The framework's superiority against several baselines is demonstrated using three benchmark datasets.**

***Index Terms*—Counterfactual explanations, data quality, socially important data science, supervised learning**

I. Introduction

Learning accurate classifiers becomes increasingly crucial as supervised classification models are widely being applied to human–related domains, including but not limited to, disease diagnosis [1] and loan application assessment [2]. One of the critical factors in building a precise classification model is training data quality [3]. However, most classification algorithms implicitly assume perfect training data [4]. Unfortunately, in the real–world, training data quality cannot be taken for granted [5]. For example, in homelessness service provision (e.g., [6]), individuals are often assigned to shelters not necessarily based on their need, but based on availability and capacity constraints. Supervised classification models trained on unreliable data can lead to biased classifications [7], with potentially detrimental consequences, including misdiagnosing a disease and delaying treatment [8].

When training data is suspected to be noisy, machine learning models are instructed to either ignore noisy training data instances or be robust to noise. Recent works on detecting and filtering label noise from training data [9] result in potentially unnecessary data loss (i.e., dropping suspected noisy samples) which can lead to either an overfitted model or a totally unusable training dataset when the noise ratio is high. Note that noisy data can refer to either noisy labels or noisy feature values. Label noise is considered to be more harmful [10]. Therefore, we specifically focus on *supervised learning in the presence of noisy labeled data instances*.

Regardless of training data quality, making accurate classifications is often not enough; "explaining" the output of a classification model [11], or offering actionable suggestions for improvement [12] are crucial in algorithmic decision–making involving humans. Furthermore, in the explanation generation process, constraints should be considered in feature perturbation process and feature value distribution, including immutable features and feature directionality (e.g., a high education degree cannot be usually revoked), for realistic suggestions to be made.

Learning a classification model with noisy training data hurts the ability to offer correct or unbiased explanations of decisions made by the classification model. This is because noisy training data can affect the training process by introducing noise to the data distribution, shifting the model classification boundary, which in turn influences and even reverses classification output. However, provided "explanations" are directly dependent on the classification model output. When wrong classification results are used to generate corresponding "explanations", inaccurate suggestions can be made. Thus, *identifying noisy data* is key to offering accurate explanations.

This work focuses on *improving algorithmic decision–making in the presence of noisy label training data, while at the same time offering actionable suggestions for undesirable decisions*. In summary, its main contributions are:

- Generalizing our previously presented approach for classification with counterfactual data generation [13] into a plug–and–play framework designed to improve supervised classification model reliability in the presence of noisy labeled training data, while offering counterfactual explanations for undesirable decisions.
- Demonstrating the generalizability of the proposed framework by considering numerous state–of–the–art methods

This material is based upon work supported by the National Science Foundation under Grant No. ECCS–1737443 and IIS–1850097.

*Corresponding author.
†Both authors contributed equally.

978-1-6654-8045-1/22$31.00

for noisy label detection, classification, and counterfactual explanation generation.
- Evaluating the proposed framework through extensive experimentation and performance analysis under increasing levels of noise on three real–world datasets.

II. Related Work

Numerous methods have been proposed to detect noisy labels. Ensemble–based methods [14], [15] leverage multiple classification models to decide a label. If prediction results are inconsistent with the observed label, the corresponding data instances are dropped. However, such methods train each classifier without excluding the noisy training data. Local learning methods assume that noisy labeled data tend to differ from their neighbors' labels [16]. Our own cluster centroid–based method leverages both ensemble and local learning for noisy labeled data detection [13].

The explainability of classification models can be broadly divided into model explanation, and outcome explanation (i.e., post–hoc explanation) [17]. Model explanation aims to shed light into the internal logic of a model and strives for transparent and global explanation in terms of the original model [18]. However, model explainability is tailored to specific learning model structures. Outcome explanations focus on explaining a specific prediction given by a classification model [19]. One of the widely used methods in this category, generates explanations through feature importance [20]. Recently, [21] proposed example–based explanations by feature perturbation. Feature perturbation, as a means to generate counterfactual data, may lead to different prediction results given a learning model. A comprehensive survey of counterfactual data generation methods for explainable classification is provided by [22]. Different from existing counterfactual data generation methods [22], the ACDG method discussed here incorporates realistic constraints (e.g., directional perturbation for features) into the counterfactual data generation. Besides, all methods in [22] assume the input training data to be free of noise. We believe our work to be the first in bridging the gap between post–hoc explainability of classification models, and training robust classifiers in the presence of noisy labeled training data.

III. Preliminaries and Problem Statement

Let $D_T = (X, Y)$ denote the noisy training dataset, and N be the total number of data instances (i.e., $X = \{x_i\}_{i=1}^{N}$). Each data instance $x \in X$ is associated with a feature d in a $K-$dimensional feature vector, and a corresponding noisy binary label $y \in \{0, 1\}$, one of which (e.g., $y = 0$) may be undesirable (e.g., loan application is rejected). Given D_T and a noisy label detection method (NLD), the task is to learn a highly accurate supervised classification model, C, to predict the label (i.e., class) of previously unseen data.

Let $\bar{y}$ denote the predicted outcome of C for instance x. By comparing y with $\bar{y}$, the training data can be divided into four subsets, namely, the true positive set X_{11}, false positive set X_{01}, true negative set X_{00}, and false negative set X_{10}. In addition to ensuring a highly accurate model C is learned despite the presence of noise in the training data, we wish to obtain realistic counterfactual data x_{cf} using a counterfactual explanation generation model (CEG), $\forall x \in X_{00}$ that can lead to $C(x_{cf}) = 1$. Let the generated counterfactual data pair be denoted as $(x_{cf}, y_{y=1})$.

The key difference between this problem setting and that of standard supervised classification is that we inherently address the problem of noisy labeled data instances in both the training dataset and counterfactual data generated during testing. Therefore, original data (x, y) and counterfactual data $(x_{cf}, y_{y=1})$ belong to one of two sets: correctly labeled set (X_r, Y_r), and noisy labeled set (X_w, Y_w).

IV. Classification with Counterfactual Data Generation in the Presence of Noisy Labels

We propose $CGEP$, a plug–and–play framework for improved Classification with counterfactual data GENeration in the Presence of noisy labels. CGEP comprises three components: (i) noisy label detection, (ii) supervised classification model, and (iii) counterfactual data generation. The noisy label detection module (Section V) is used for detecting noisy labeled data instances. The counterfactual data generation module (Section VI) is used to generate counterfactual data for those data instances with undesirable label assigned by the classification model.

Initially, the input dataset is passed to the noisy label detection component, where noisy labeled data instances are identified. The result of this process is a filtered dataset, which is then used to train a classification model. The predictions of the classification model are inspected, and data instances with undesirable predictions are passed into the counterfactual generation module in order to obtain corresponding counterfactual data, which would achieve desirable predictions. The generated counterfactual data instances are inspected by the noisy label detection module to ensure that generated data indeed follows the original data distribution.

V. Noisy Label Detection Module

We demonstrate CGEP's ability to incorporate different noisy label detection methods by considering a number of such methods as described below. These methods are selected because they are (i) suited for numerical datasets[1], (ii) decoupled from the classification model[2], and can therefore be included in our plug–and–play framework, and (iii) represent broad noise detection strategies (e.g., ensembles; local learning methods; combination of ensembles and local learning).

*1) **Cluster Centroids–based Noisy Label Detection (CED) [13]:*** Introduced in our own work [13], CED is designed to identify data instances with desirable (or undesirable) labels as noisy if their distance to the centroid of the desirable class is larger (or smaller) than the distances to the centroid of the undesirable class.

[1]Methods specifically tailored to image datasets (e.g., [23], [24]) are not included in our analysis).

[2]Approaches, such as [25], that perform joint noisy data detection and classification model learning, are incompatible with our framework, but are considered as baselines for evaluation purposes.

*2) **Ensemble Filter (EF) [15]**:* EF divides the input dataset into n folds, and for each fold n, trains multiple classification models on the remaining $n-1$ folds. Using the trained models, it decides whether the data instances in the excluded fold are noisy labeled or not. Each classification model is constructed of several base detectors (i.e., classifiers). If the majority of base detectors cannot classify the data instances correctly, the data instances are flagged as noisy labeled.

*3) **Iterative–partitioning Filter (IPF) [26]**:* IPF divides the training dataset into subsets, and learns classification rules for each subset separately. Data instances are evaluated by all classification rules, and flagged as noisy if more than half rules predict a label that differs from the original.

*4) **Automatically Pre–select noisy labeled Data Instances (IMICD) [27]**:* IMICD begins by learning a classification model using all training data instances. The trained model is then used to obtain predicted labels for the training data instances, at which point the inner product score of the two labels is computed. Instances are then sorted by their inner product score in ascending order, and the top data instances within an a prior specified error ratio are labeled as noisy.

*5) **Reprocessing Instances that should be Misclassified (PRISM) [28]**:* PRISM is a filtering method that removes instances for which labels predicted by a classification model do not match their original label from the input dataset before passing the "clean" dataset to the learning model. PRISM identifies data instances as mislabeled if none out of classification models can correctly classify it.

*6) **Complete Random Forest–based Class Noise Filter (CNF) [29]**:* CNF detects noisy labeled data instances based on the idea that such data instances are more likely to be surrounded by data instances whose labels don't match that of the instance under question.

VI. Counterfactual Data Generation

We consider multiple state–of–the–art counterfactual data generation methods for CGEP's counterfactual data generation module. All methods, excluding the actionable counterfactual data generator (ACDG) that was introduced in our own prior work [13], have been experimentally evaluated as part of a comprehensive survey [11]. Instead of listing all works included in [11], we specifically focus on those methods that treat each feature independently, similarly to ACDG.

*1) **Actionable Counterfactual Data Generator (ACDG) [13]**:* ACDG formulates counterfactual data generation as an optimization problem incorporating validity [21], proximity [12] and actionability [13] constraints.

*2) **Contrastive Explanations Model (CEM) [30]**:* Given a feed forward neural network, CEM identifies both pertinent positive (PP) features as well as pertinent negative (PN) features to achieve the desired class. To find both PP and PN features, CEM optimizes separate functions using the projected fast iterative shrinkage–thresholding algorithm [31]. For a fair comparison, we focus on PP counterfactuals generated by CEM, since our goal involves counterfactual data generation only for the desirable class.

*3) **Diverse Counterfactual Explanations (DICE) [12]**:* DICE generates multiple explanations for each input data instance by incorporating proximity, diversity and validity constraints, in similar spirit to our ACDG approach, into the counterfactual generation objective function.

*4) **Actionable Recourse (AR) [32]**:* AR evaluates (i) the ability of a linear learning model to generate counterfactuals subject to a target class for all data instances; (ii) the difficulty to generate valid counterfactuals across all data instances; and (iii) the disparity of generated counterfactual for similar data instances towards a target label.

*5) **Wachter [21]**:* This approach generates counterfactual data by minimizing an objective function comprising the same validity and proximity constraints as in ACDG. Different from Wachter, ACDG incorporates multiple real–world constraints to generate realistic counterfactual data instances.

VII. Experiments

We conduct comprehensive experiments on three real–world datasets to explore the effectiveness of the proposed plug–and–play framework with the noisy label and counterfactual data generation methods described in Sections V and VI. The code for all baselines is publicly available. The majority of the baselines are available in Python 3.8 with PyTorch 1.9.0, whereas some baselines (i.e., EF, IPF, and PRISM) are available in R. The proposed framework has been implemented in Python 3.8 with PyTorch 1.9.0. For a fair comparison, the same training and testing datasets are used by all methods. All experiments were conducted on an iMac running macOS Big Sur with 3.8 GHz 8–core intel Core i7 processor and 16 GB 2667 MHz DDR4 memory.

A. Datasets

We use the real–world, publicly available datasets, that are widely used in counterfactual data generation evaluation for binary classification [17] as follows. **Adult–Income** [33] records individual–level information to predict individual annual income. **Bank–Marketing** [34] comprises detailed bank–marketing records related to campaigns of a Portuguese banking institution. **German–Credit** [35] records the background information of clients who have applied for a loan.

B. Baselines

For comparison, we consider baselines as follows. **XGBoost (Xgboost)** [36] has been shown to be the most robust to noisy labeled data [37]. **Naive Bayes (NB)** [38] has been shown to suffer the most from noisy labeled data compared with other classification models, including Decision Tree, Logistic Regression, Random Forest, KNN, and Adaboost [37]. **Peer Loss (PL)** [25] facilitates training of a classification model with noisy label data without knowing the noise rate. We consider two versions of PL: (i) **PL-Default** (i.e., peer loss with the default setting of neural network used in [25]); and (ii) **PL–NN** (i.e., peer loss with network architecture and settings described in Section VII-C2).

C. Experimental Setup

1) Noisy Label Detection: In noisy label detection experiments, we pre–split each dataset into training and testing sets with a ratio of 3:1. In the training set, we add label noise by randomly flipping the label of each data instance with noise rate $\tau = \frac{|\tilde{X}|}{|X|}, \tau \in \{0.1, 0.2, 0.4, 0.6, 0.8\}$. EF, IPF, and PRISM are available in R as part of the "NoiseFiltersR" package, while CNF and IMICD are implemented in Python. K Nearest Neighbor (KNN) is used to test the effectiveness of these methods in "filtering" the original, noisy dataset. The choice of KNN is based on experimental results in [37], which demonstrated that KNN is only moderately robust to label noise, as compared with other learning models, such as XGBoost (best), and Naive Bayes (worst).

2) Counterfactual Data Generation: In counterfactual data generation experiments, the training set remains free of noise. The CARLA [22] implementations of CEM, Wachter, AR, and DICE methods in Python are used. Wachter, CGEP, AR, and DICE are based on PyTorch, while CEM is based on TensorFlow. For ACDG, we use the AdamW optimizer [39], which is implemented in PyTorch [40], with a learning rate of 0.01, to minimize the loss function. We set $\varepsilon_0 = 0.1$ and $T_C = 30$. Finally, a feed–forward neural network with three hidden layers is used for classification, with the number of neurons in each layer being 64, 32, and 2 accordingly.

3) Overall Framework Evaluation: To evaluate the overall framework, we pre–split each dataset into training and testing sets with a ratio of 3:1. Label noise is introduced in the training set by randomly flipping the label of each data instance with noise rate $\tau \in \{0, 0.1, 0.3, 0.5, 0.7, 0.9\}$. Note that since counterfactual explanations are dependent on the classification output, we evaluate the impact of the classification model on the counterfactual data generation module separately.

D. Evaluation Metrics

1) Noisy Label Detection: We divide noisy label detection evaluation into two parts, namely detection performance analysis, and evaluation of classifiers trained on filtered data. We need to consider two questions: (i) how many of all detected noisy labeled data instances are truly noisy, and (ii) with respect to all noisy labeled data instances in the training set, how many of them are detected from noise detection methods. Therefore, inspired by [27], we define $NS - Precision = \frac{\#TrueNSdetection}{\#AllNSdetection}$ and $NS - Recall = \frac{\#TrueNSdetection}{\#AllNSdatainstances}$. Different from [27], the estimated noise rate is not included in these evaluation metrics, providing a stricter evaluation environment since the overall noisy label ratio is hidden from detection methods. We evaluate the classifiers' train and test accuracy separately as $\frac{\sum_{i=1}^{N}((y_i=\bar{y}_i=1)+(y_i=\bar{y}_i=0))}{N}$ in corresponding sets. To account for imbalanced datasets, we additionally report F1–score$= 2 * \frac{precision*recall}{precision+recall}$.

2) Counterfactual Data Generation: We evaluate counterfactual data generation methods' accuracy, true positive rate and proximity. Specifically, classification accuracy is computed using predictions based on generated counterfactual data. True positive rate (TPR) is defined as $TPR = \frac{\sum_{i=1}^{N} y_i=\bar{y}_i=1}{N}$, and is used to measure the overall validity rate for generated counterfactual data instances based on the classification model. The higher the true positive rate, the better the performance. When $Accuracy = TPR$, all data instances with undesirable class in the original dataset have been substituted with "valid" counterfactual data with desirable label. Proximity evaluates the distance between counterfactual data and the original data instance vector. In general, counterfactual data with low proximity is preferred as they are considered to be more actionable.

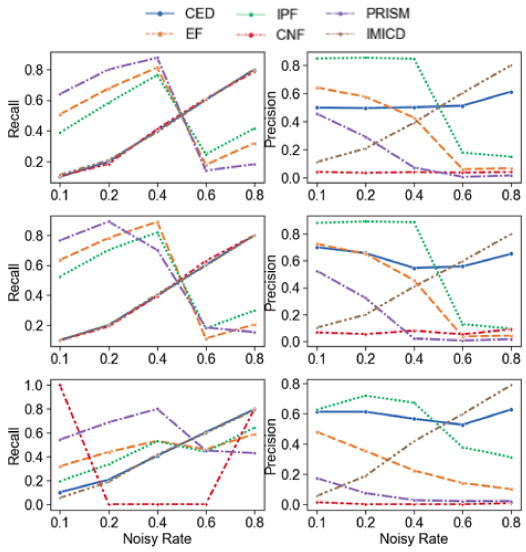

Fig. 1. NS–Precison and NS–Recall with respect to Adult–Income (top), Bank–Marketing (middle), and German–Credit (bottom) datasets, accordingly.

3) Overall CGEP Framework: Given fixed noisy label detection and classification modules, we evaluate CGEP using accuracy on clean testing data.

E. Evaluation Results

1) Noisy Label Detection: Fig. 1 shows the noisy label detection evaluation results. PRISM, EF, and IPF follow a similar trend with respect to $NS - Recall$, exhibiting higher recall when $\tau < 0.5$. Conversely, when $\tau > 0.5$, i.e., when noise becomes severe, recall drops sharply. Majority voting does not help in such regime since most base classifiers' accuracy degrades when trained with data containing high levels of noise. With respect to $NS - Precision$, IPF outperforms PRISM and EF, as well as the rest of the methods, when noise is moderate or low. Conversely, when $\tau > 0.5$, CED and IMICD outperform IPF. Overall, CED seems to be the best performing method when accounting for both $NS - Recall$ and $NS - Precision$ across noise rates. Nevertheless, according to $NS - Precision$ and $NS - Recall$, CED is a good choice in severe label noise (i.e., when $\tau > 0.5$) considered poor performance of PRISM when $\tau < 0.5$. In summary, IPF is the best choice when noise is lower than 50%.

2) Counterfactual Data Generation: Fig. 2(a) shows how the proposed ACDG, as well as the baselines fair with respect to true positive rate in the training set. CEM and DICE

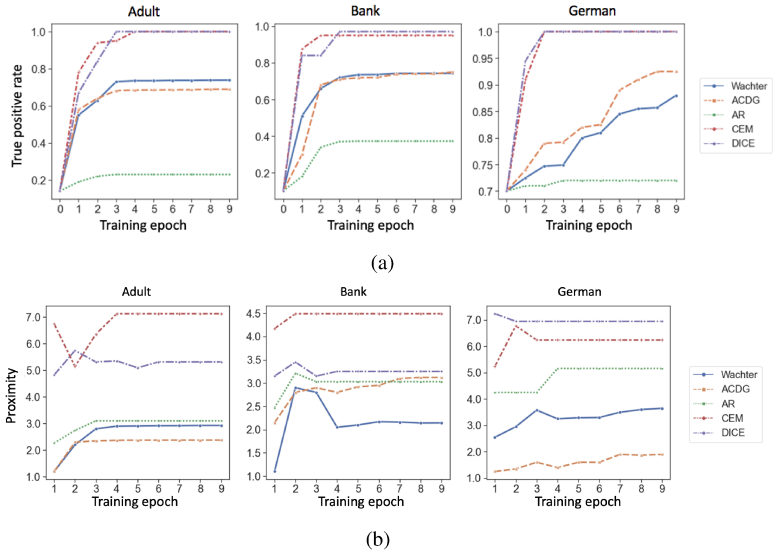

Fig. 2. Counterfactual data generation methods comparison with respect to (a) true positive rate and (b) proximity. In all cases, $x-$axis corresponds to training epoch. Results are shown for Adult–Income (left), Bank–Marketing (middle), and German–Credit (right) datasets.

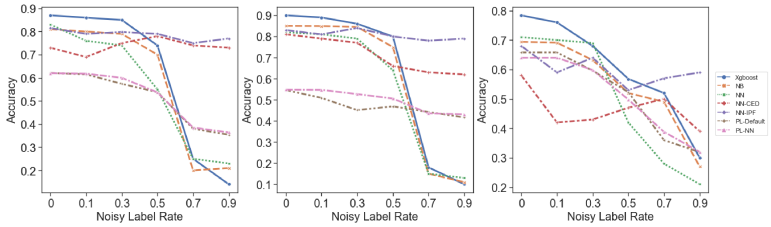

Fig. 3. Accuracy ($y-$axis) per different noisy label rate ($x-$axis) with respect to Adult–Income, Bank–Marketing, and German–Credit datasets, accordingly.

converge more quickly (i.e., within less training epochs) than other methods, indicating that the validity rate of generated counterfactuals is increasing with each epoch. DICE slightly outperforms CEM in the Adult–Income and Bank–Marketing datasets. We attribute this to DICE generating multiple counterfactuals for each data instance, leading to higher probability of a valid counterfactual candidate.

Nevertheless, true positive rate alone is not enough to justify which counterfactual generation method is supreme. Proximity (see Fig. 2(b)) help demonstrate the cost of a high validity rate, which translates into a larger number of perturbed features and greater distance between the counterfactual vectors and original data instances accordingly. CEM and DICE's high proximity can also be attributed to the one–hot encoding of categorical features, which leads to independent perturbation of such features. Intuitively, when proximity is smaller, counterfactual explanations can be considered to be more "actionable", since less things need to be changed to achieve to the desirable class. In this sense, ACDG appears to be a better choice among the candidate methods.

3) Overall CGEP Framework: To quantify the effect of noisy labels on our plug–and–play framework when using either CED or IPF, we compare its performance with other classification models (i.e., Xgboost, NB and NN without noisy label detection). We used feed forward neural network (FNN) for classification model in CGEP, as we have shown it to be a good choice in our own prior work [13], and [22] used FNN to drive the counterfactual generation process. We also compare CGPE to the peer loss baseline (see Section VII-B), which is specifically designed for learning in a noisy environment.

Fig. 3 shows that naively training a classification model (i.e., Xgboost, NB, NN) with noisy labeled data leads to a dramatic accuracy degradation with increasing noise rate. Generating counterfactual explanations would be futile when inaccurate classification models are used, since explanations would be meaningless in this case. In contrast, the classification model is more robust under the CGEP framework, as indicated in Fig. 3. Instead of struggling to learn class specific features with highly corrupted training data (which may be impossible, as label quality in this regime is close to random), CGED (both NN–CED and NN–IPF) correctly identifies and restores noisy labels before proceeding to train a highly accurate NN classification model. In summary, CGEP is superior to other standalone classification models as well as state–of–the–art methods designed to address noise in the learning process, particularly so in highly noisy environments.

VIII. Conclusion

As automated classification and algorithmic decision–making models become part of everyday life, the quality of

data used to train such models becomes critical. This work proposed a plug–and–play framework to learn a robust supervised classification model in the presence of noisy labeled data, while at the same time providing actionable suggestions for undesirable decisions made by the classification model. Extensive experimental evaluation results shed light into the framework's ability to incorporate alternative noisy label detection methods and counterfactual explanation approaches. The framework's superiority against several baselines. We hope that by proposing this framework, more effective methods will be developed to address the challenges associated with quality issues in the datasets used to train explainable machine learning models.

REFERENCES

[1] M. M. Ahamad, S. Aktar, M. Rashed-Al-Mahfuz, S. Uddin, P. Liò, H. Xu, M. A. Summers, J. M. Quinn, and M. A. Moni, "A machine learning model to identify early stage symptoms of sars-cov-2 infected patients," *Expert systems with applications*, vol. 160, p. 113661, 2020.

[2] A. K. Tiwari, "Machine learning application in loan default prediction," *Machine Learning*, vol. 4, no. 5, 2018.

[3] N. Gupta, S. Mujumdar, H. Patel, S. Masuda, N. Panwar, S. Bandyopadhyay, S. Mehta, S. Guttula, S. Afzal, R. Sharma Mittal *et al.*, "Data quality for machine learning tasks," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 4040–4041.

[4] N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. Paritosh, and L. M. Aroyo, ""everyone wants to do the model work, not the data work": Data cascades in high-stakes ai," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–15.

[5] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.

[6] C. Chelmis, W. Qi, and W. Lee, "Challenges and opportunities in using data science for homelessness service provision," in *Companion Proceedings of the Web Conference 2021*, 2021, pp. 128–135.

[7] H. Valizadegan and P.-N. Tan, "Kernel based detection of mislabeled training examples," in *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 2007, pp. 309–319.

[8] K. Huang, H.-G. Stratigopoulos, and S. Mir, "Fault diagnosis of analog circuits based on machine learning," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*. IEEE, 2010, pp. 1761–1766.

[9] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song, and S.-T. Xia, "Iterative learning with open-set noisy labels," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8688–8696.

[10] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study," *Artificial intelligence review*, vol. 22, no. 3, pp. 177–210, 2004.

[11] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning," *Journal of Artificial Intelligence Research*, vol. 70, pp. 245–317, 2021.

[12] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020, pp. 607–617.

[13] W. Qi and C. Chelmis, "Improving algorithmic decision–making in the presence of untrustworthy training data," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 1102–1108.

[14] C. E. Brodley and M. A. Friedl, "Improving automated land cover mapping by identifying and eliminating mislabeled observations from training data," in *IGARSS'96. 1996 International Geoscience and Remote Sensing Symposium*, vol. 2. IEEE, 1996, pp. 1379–1381.

[15] ——, "Identifying mislabeled training data," *Journal of artificial intelligence research*, vol. 11, pp. 131–167, 1999.

[16] J. S. Sánchez, R. Barandela, A. I. Marqués, R. Alejo, and J. Badenas, "Analysis of new techniques to obtain quality training sets," *Pattern Recognition Letters*, vol. 24, no. 7, pp. 1015–1022, 2003.

[17] S. Verma, J. Dickerson, and K. Hines, "Counterfactual explanations for machine learning: A review," *arXiv preprint arXiv:2010.10596*, 2020.

[18] R. Krishnan, G. Sivakumar, and P. Bhattacharya, "Extracting decision trees from trained neural networks," *Pattern recognition*, vol. 32, no. 12, 1999.

[19] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[20] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, 2017, pp. 4768–4777.

[21] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harv. JL & Tech.*, vol. 31, p. 841, 2017.

[22] M. Pawelczyk, S. Bielawski, J. v. d. Heuvel, T. Richter, and G. Kasneci, "Carla: a python library to benchmark algorithmic recourse and counterfactual explanation algorithms," *arXiv preprint arXiv:2108.00783*, 2021.

[23] G. Pleiss, T. Zhang, E. Elenberg, and K. Q. Weinberger, "Identifying mislabeled data using the area under the margin ranking," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 044–17 056, 2020.

[24] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2691–2699.

[25] Y. Liu and H. Guo, "Peer loss functions: Learning from noisy labels without knowing noise rates," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6226–6236.

[26] X. Zhu, X. Wu, and Q. Chen, "Eliminating class noise in large datasets," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 920–927.

[27] N. M. Müller and K. Markert, "Identifying mislabeled instances in classification datasets," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[28] M. R. Smith and T. Martinez, "Improving classification accuracy by identifying and removing instances that should be misclassified," in *The 2011 international joint conference on neural networks*. IEEE, 2011, pp. 2690–2697.

[29] S. Xia, G. Wang, Z. Chen, Y. Duan *et al.*, "Complete random forest based class noise filtering learning for improving the generalizability of classifiers," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 11, pp. 2063–2078, 2018.

[30] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das, "Explanations based on the missing: Towards contrastive explanations with pertinent negatives," *Advances in neural information processing systems*, vol. 31, 2018.

[31] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[32] B. Ustun, A. Spangher, and Y. Liu, "Actionable recourse in linear classification," in *Proceedings of the conference on fairness, accountability, and transparency*, 2019, pp. 10–19.

[33] R. Kohavi and B. Becker, 1996. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/adult

[34] P. C. S. Moro and P. Rita, 2014. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/bank+marketingwachter

[35] D. H. Hofmann, 2019. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

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

[37] P. Li, X. Rao, J. Blase, Y. Zhang, X. Chu, and C. Zhang, "Cleanml: A benchmark for joint data cleaning and machine learning [experiments and analysis]," *arXiv preprint arXiv:1904.09483*, p. 75, 2019.

[38] K. P. Murphy *et al.*, "Naive bayes classifiers," *University of British Columbia*, vol. 18, no. 60, pp. 1–8, 2006.

[39] S. Gugger and J. Howard, "Adamw and super-convergence is now the fastest way to train neural nets," *last accessed*, vol. 19, 2018.

[40] N. Ketkar, "Introduction to pytorch," in *Deep learning with python*. Springer, 2017, pp. 195–208.