

Learning from Disagreement for Event Detection

Anonymous Authors

Abstract—Using a newly developed model to upgrade a legacy model is a common practice in machine learning applications. After the upgrade, it is expected that the new model should outperform the legacy model in the regions of interest. However, it is observed that the new model often makes incorrect decisions on some instances where the legacy model still performs well. For a binary classification model (e.g., click-through-rate/CTR prediction model), such undesirable behavior could even occur in the low false positive region of the receiver operating characteristic (ROC) curve. Finding the reasons behind this phenomenon can help business partners in an organization gain confidence in adopting the new model and help modelers to improve the new model in future releases. In this paper, we present the “Learning from Disagreement” framework to understand and improve the performance of a predictive model. Under the setting of a binary classification task, this proposed approach focuses on instances that lead to contradictory decisions between a pair of models at a given operating point. We perform feature importance analysis exclusively on these instances, gain insights into the pair of models without even knowing their inner operations, and offer actionable feedback for model improvement. We demonstrate the usefulness of this framework on two real-world event detection datasets.

Index Terms—Learning from disagreement, performance metric, feature importance analysis, event detection

I. INTRODUCTION

Machine learning applications often need to compare the performance between a pair of models. Considering the following practical example scenarios:

Model refresh and upgrade. A model deployed in production needs to get retrained periodically with new data. Before the retrained model replaces the production model, we must ensure that the retrained model performs better than the production model. The need also arises when we upgrade a legacy model to an advanced (second-generation) model. For instance, upgrading a logistic regression model to a deep neural network model. In this scenario, the advanced model will go through an even more stringent scrutiny process to ensure the performance improvement justifies the cost associated with the upgrade.

Third-party score. An organization brings in a third-party predictive score to enhance or replace an in-house predictive score. In this case, for proprietary reasons, the details of the third-party score such as predictive features and algorithms may never be disclosed. To gain confidence in adopting the third-party score, the in-house team needs to figure out where the third party score performs better and where it does not. They also need to find the reasons behind that, even without the full knowledge of the third-party’s model.

Model ensemble. We want to leverage scores from two models to make a final decision. For instance, we can ensemble risk scores from the legacy model and the advanced model to determine whether a credit card transaction is legitimate. It is

a well-known fact that ensemble helps improve model performance, but ensemble can be costly to deploy [1] – the system needs to maintain two models with often different features and model architectures, not to mention other challenges such as score latency and memory footprint. In this scenario, we need to present a very strong case to convince our business partners or clients when the ensemble will work better and why.

In these example scenarios, it is observed that the “new” model often makes incorrect decisions on some instances where the “old” model still performs well. For a binary classification model (e.g., CTR prediction), such undesirable behavior could even occur in the far-left corner of the ROC curve, where the “new” model is supposed to prevail. Finding the reasons behind this phenomenon can not only help business partners in an organization gain confidence in adopting the “new” model, but also help modelers to improve the model in future releases.

However, it is a nontrivial task to reveal why one model makes incorrect decisions on some instances where the other one does not. A number of approaches have been proposed to understand the reasons behind a model’s prediction, among which the representative methods are LIME [2] and SHAP [3]. However, both LIME and SHAP are designed for a single model and do not help disclose prediction discrepancies from two models. Several recent works have explicitly tackled model discrepancies in a research setting. For instance, Marx et al. [4] formalize the problem as integer programming by constraining the models to be linear. Renard et al. [5] learn a graph connecting the training instances to discover discrepancies between models. Although mathematically elegant and performing well for small datasets, they are hard to scale. Also, while these methods raise awareness of the problem, they do not provide actionable insights into how discrepancies can be leveraged to improve model performance. Further, these methods only address certain aspects of the problem (e.g., how to measure discrepancies, how to locate discrepancies, etc.); they do not provide an end-to-end operational solution for practitioners.

Our contributions: Motivated by the aforementioned business needs and challenges, we present the “Learning from Disagreement” (LFD) framework (refer to Fig. 1), to assess, understand, and improve the performance of a predictive model. The core of LFD is to work on a pair of models, with one model serving as the *mirror* of the other model. LFD treats the pair of models as black boxes without the need of knowing their inner details. It aligns two predictive scores, ensuring both are at the same scale and thus comparable (Sec. III-B). It is focused on instances that lead to contradictory decisions between the pair of models at a given operating

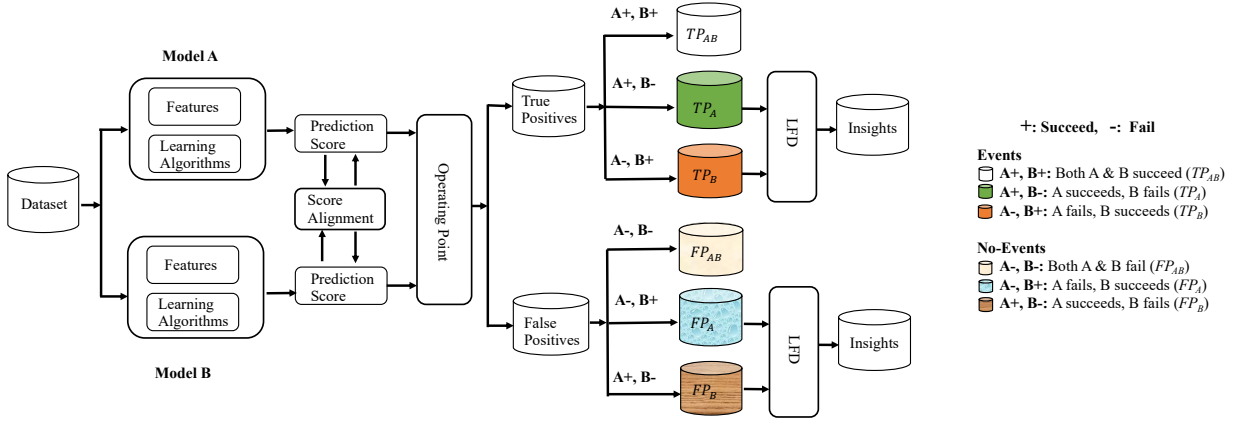


Fig. 1. An Overview of the LFD framework. The framework’s details are explained in Sec. III.

point, narrowing down the problem to a much smaller scope (Sec. III-C). It uses a new performance metric tailored for the pair of models, capturing their relative predictive strengths and weaknesses at the operating point (Sec. III-D). Equipped with a simple but effective feature importance analysis algorithm (Sec. III-E), LFD leverages the most precious knowledge from domain experts, in the form of oracle features, to help business partners and modelers gain insights into the predictive model and thereafter improve its performance. We demonstrate the usefulness of this framework on a public CTR prediction dataset and an in-house transaction anomaly detection dataset (Sec. IV). This framework should be particularly useful for practitioners in large scale predictive modeling problems.

II. DESIGN CONSIDERATIONS AND SOLUTION PATHS

The goal of LFD is to provide an operational framework that is easy to use and able to provide users with interpretable and actionable insights. With this in mind, the design of LFD prioritizes practical utilization over mathematical complexity, reflected in the following four design considerations and proposed solutions (refer to Fig. 1).

Aligning two predictive scores. When working on highly unbalanced datasets such as CTR prediction or transaction anomaly detection, it is a common practice to down-sample non-events (negative samples). After the model is trained, prediction scores are aligned or calibrated to reflect the true population probability. This can be done by adjusting the prediction score in the down-sampling space using a Bayesian formula [6] or isotonic regression [7], which requires obtaining the precise sampling rate. We could use these methods for each of the models in the pair to arrive at two aligned scores respectively. However, this pair of models are often created by two different teams and each team may apply quite different sampling strategies. Also, one model could come from a third party. In these scenarios, it is hard to get precise sampling rates to do the alignment. To overcome this burden, we propose a new score alignment algorithm in Sec. III-B to align the two scores without resorting to the sampling rate. The key idea is that instead of making the two scores match the population

probability to make them comparable, we do score alignment using one model’s score against another (i.e., one model serves as the reference). Although simple, this algorithm provides a flexible mechanism of aligning two predictive scores across a range of commonly used metrics in the industry such as percentage of instances to work on, catch rate, and false positive ratio.

Working on instances at a given operating point only.

In a real production system, there is always an *operating point* or score cutoff involved. For instance, in a transaction anomaly detection system the performance assessment of the model is performed at a specific operating point greater than 0. At the operating point 0, all anomalies would be correctly identified, but this model would be useless since this comes at the expense of a huge number of wrongly classified non-anomalies (i.e., false positives). Therefore, we are not really interested in whether a model can make correct predictions on all instances; rather, what we care most about is whether a model can make correct predictions at a given operating point.¹ In an ad-targeting system, the operating point is often at the top 5% of the population [9]. For a transaction anomaly detection system, the operating point is most likely located at the far-left corner of the ROC curve. Furthermore, because a production model is typically used by different clients (e.g., a transaction anomaly detection model may be used by multiple credit card issuers), the operating point can vary, depending on how a client will use the score. The design of LFD should offer the flexibility of analyzing instances at all possible operating points. Finally, working only on instances at a given operating point offers another benefit: the dataset now is more balanced because events (clicks or anomalies) are no longer the minority in the high score region [10]. We provide more details in Sec. III-C on how to work on instances at a given operating point.

Developing a new performance metric. LFD involves a pair of models so the success or failure of a model is always

¹Working only on instances above a given score cutoff is, in spirit, similar to the idea of “Sliced Analysis” in [8] where the authors point out that working on full instances is unnecessary and may even mask important effects, such as quality improving in one area but degrading in another.

compared with another model in the pair. The pair of models are bundled; a model when paired with a different model could lead to a different success or failure rate. Conceptually, if the first model in the pair is weak, but the second model is even weaker, then the first model would still have a higher success rate than the second model when making predictions. To that end, we propose a new performance metric in Sec. III-D to assess the performance of a pair of models *simultaneously*, at any given operating point. This new metric is a *relative* metric, tailored for a pair of models. Besides its uniqueness in coping with a pair of models, this new metric also addresses the issue of asymmetric importance in classification errors. For instance, in transaction anomaly detection the consequence of misclassifying an anomalous transaction as a normal one is more serious than the other way round.

Gaining insights and offering actionable feedback. The main motivation of our work is to gain insights, echoing “the goal of science is not wins, but knowledge” [8]. On the other hand, the ultimate purpose of gaining insights is to transfer them as feedback to business partners and modelers. For business partners, these insights help them better understand how a model works; and for modelers, these insights help them find ways of improving the model in future releases. Insights can be learned and presented using various forms, for example, revealing problems and suggesting feasible solutions [11], [12]. In our work, however, we approach this problem from a feature importance analysis perspective [13]–[15] based on two reasons. First, features are one of the most important factors that contribute to the success of a machine learning model [16]. Second, features are interpretable and actionable, that is, a business partner can easily grasp the findings and a modeler can quickly retrain another model with new features. To facilitate feature importance analysis, we make a nontrivial improvement to a widely used feature ranking method known as *information value* (IV) [17] in Sec. III-E. This improvement is made after experiencing its flaws while applying the original IV formula to features having high cardinality. Such features are quite common in CTR prediction and transaction anomaly detection.

By following the design principle of utilization over complexity and accommodating the four key considerations, LFD provides a practical approach to assessing, understanding, and improving the performance of a predictive model, which is currently being used across multiple internal teams.

III. THE FRAMEWORK

Fig. 1 presents the LFD framework and we elaborate its details by answering the following questions: how it aligns two predictive scores (Sec. III-B); what are the focused instances of LFD (Sec. III-C); how to evaluate the relative performance (Sec. III-D); and what insights LFD can offer for model improvements (Sec. III-E).

A. Dataset

The dataset used by the pair of models (named Model A and B in Fig. 1) is in the common tabular format (rows: instances,

columns: features). Different organizations may enhance it by adding their own proprietary information, but some key raw features should be retained. For CTR, some example raw features may include user ID, publisher ID, and advertiser ID [18]. For transaction anomaly detection, some raw features could be payment card number, merchant ID, and dollar amount, extracted from ISO 8583 Message [19]. The dataset should also include label information, for example, whether an impression has led to a click, or whether a credit card transaction has been confirmed as fraudulent.

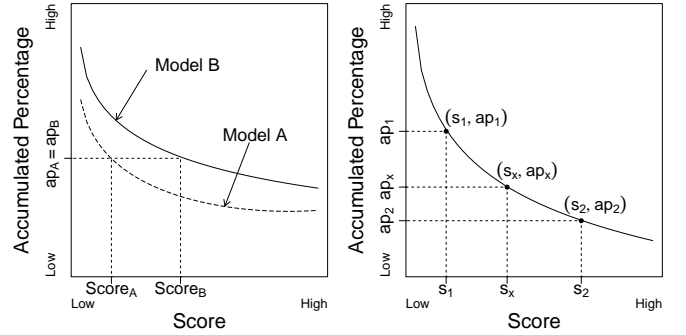


Fig. 2. Score alignment algorithm.

B. Score Alignment Algorithm

Instead of trying to make two scores match the population probability so that they are comparable [6], [7], we perform score alignment using one model’s score against another, without the need of acquiring the sampling rate whose precise value is often hard to obtain. Fig. 2 describes the score alignment algorithm. The left panel shows that $Score_A$ in model A is aligned to $Score_B$ in model B through ap_A and ap_B , the accumulated percentages of instances² from Model A and Model B, respectively. Here ap_A and ap_B have served as a bridge that connects two scores together. The right panel shows that, when ap_x is present from Model A but absent from Model B, the corresponding score s_x in Model B needs to be calculated via interpolation based on its neighbors (s_1, ap_1) and (s_2, ap_2) using the following formula:

$$s_x = s_2 + \frac{(s_2 - s_1)(ap_2 - ap_x)}{ap_1 - ap_2} \quad (1)$$

Fig. 3 shows a practical score alignment example from CTR prediction on the Avazu dataset [20], where Model A is aligned against Model B. These two models are detailed in Sec. IV-A. The alignment adjusts the blue curve to the red one, making the score distribution of Model A similar to that of Model B (i.e., the red and green curves are almost completely overlapped).

²This algorithm is flexible and can be easily adjusted based on clients’ requirements. For instance, we could replace percentages of instances with other metrics such as false positive ratio.

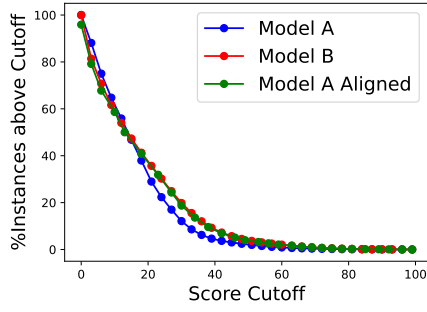


Fig. 3. An example of score alignment.

C. Locating Instances Receiving Conflicting Predictions

LFD focuses on instances at a given operating point, that is, instances scored above a given score cutoff. These instances are the most critical cases because they are classified as “events” by a predictive model, either correctly or incorrectly. In CTR prediction, when an instance is classified as an “event”, an impression will be presented to the visitor; in transaction anomaly detection, when an instance is classified as an “event”, a transaction may be declined automatically or a case may be created for further investigation.

This is the reason that we have only positives, either true positives or false positives in Fig. 1, dispersed in six boxes. The upper three boxes are true positives, and the lower three are false positives. Without confusion and for convenience, we denote instances in the upper three boxes as TP_{AB} , TP_A , TP_B , and instances in the lower three as FP_{AB} , FP_A , FP_B . Here we use “+” and “-” to indicate a model made correct and incorrect predictions respectively. In the upper three boxes, the pairs $(A+, B+)$, $(A+, B-)$, and $(A-, B+)$ represent true positive instances (events) correctly classified by both A and B , by A only, and by B only, respectively. In the lower three boxes, the pairs $(A-, B-)$, $(A-, B+)$, and $(A+, B-)$ represent false positive instances (non-events) incorrectly classified by both A and B , by A only, and by B only, respectively.

In LFD, we are only interested in instances that receive conflicting predictions from the pair of models, that is, TP_A and TP_B , which represent disagreements on events (true positives), and FP_A and FP_B , which represent disagreements on non-events (false positives). TP_{AB} and FP_{AB} are of less interest because they are given the same prediction by both models, either correctly or incorrectly. We investigate instances from these two types of disagreements separately. For the two boxes TP_A and TP_B , we assign a label 0 to the instances in TP_A , and a label 1 to the instances in TP_B . For the two boxes FP_A and FP_B , we assign a label 0 to the instances in FP_A , and a label 1 to the instances in FP_B . Notice here the assigned labels have nothing to do with the real class labels; they are just for the purpose of analysis.

D. Performance Metric for a Pair of Models

The proposed performance metric, dubbed as *relative success rate* (RSR), is specifically designed for assessing the performance of a pair of models simultaneously. It is defined

in Equation 2 and Equation 3 for Model A and Model B , respectively:

$$RSR_A = \frac{TP_{AB} + TP_A + \lambda \times FP_B}{TP_{AB} + TP_A + TP_B + \lambda \times (FP_{AB} + FP_A + FP_B)} \quad (2)$$

$$RSR_B = \frac{TP_{AB} + TP_B + \lambda \times FP_A}{TP_{AB} + TP_A + TP_B + \lambda \times (FP_{AB} + FP_A + FP_B)} \quad (3)$$

where λ is a discount parameter reflecting a cost for non-events.³

Because these two equations carry similar information, we are focused on Equation 2 only. Let’s start with the numerator. Recall that TP_{AB} are events correctly classified by both Model A and Model B , so TP_{AB} will be counted as a success for Model A . TP_A are events correctly classified exclusively by Model A , so it will also be counted as a success for Model A . FP_B are non-events incorrectly classified as events by Model B exclusively, it will be counted as a success for Model A as well. To put this another way, for the same FP_B non-events, Model B makes mistakes by scoring them high, but Model A does not, hence Model A should take credit for not making mistakes on these non-events; and meanwhile, because misclassifying a non-event is less serious than misclassifying an event, this credit should come with a discount, leading to $\lambda \times FP_B$. The denominator is just the sum of all events correctly classified by both models and all non-events incorrectly classified by both models (with a discount parameter λ). An alternative representation for the new metric is the *relative failure rate* (RFR), as defined by $RFR_A = 1 - RSR_A$ and $RFR_B = 1 - RSR_B$ for the pair of models, respectively.

We want to emphasize two key points on the new performance metric. First, the metric is *not* designed to assess the performance of a single model, although it reduces to the *precision* metric (also known as the hit rate) for a single model (that is, $TP_A / (TP_A + \lambda \times FP_A)$ for model A , and $TP_B / (TP_B + \lambda \times FP_B)$ for model B , where $\lambda=1$). It is a relative performance metric. The success or failure of a model is always measured against the other in the pair. Second, it takes costs into consideration, addressing the issue of asymmetric importance in classification errors. This is done through introducing the discount parameter λ .

E. Gaining Insights through Feature Importance Analysis

The key to discovering strong discriminative features lies in understanding properties of the data that distinguish one class from another [15]. There exist automatic tools [22] to facilitate this process, but ultimately it is the modeler’s job to come up with the best features that are predictive, interpretable, and deployable. Arguably, this is the most important step in building a predictive model [16]. Deep learning is capable of discovering powerful features automatically and has shown remarkable successes on image, speech and text data. However, in domains like CTR prediction and transaction anomaly detection, which are at their core an empirical discipline

³For CTR prediction, we assume $\lambda=0.0025$, that is, a \$2.5 cost per 1000 impressions [21]. For transaction anomaly detection, we assume $\lambda=0.02$, that is, a \$2 cost per \$100 purchase [21].

and involve human factors, feature engineering is hard to be replaced. Indeed, expert-crafted features have played a crucial role in real industrial production systems [6], [18], [23].

We first create a large feature pool ("oracle" features. Refer to Appendix B for some example features) based on our understanding of the data and domain knowledge, and then investigate which features contribute to the disagreements between two models at a given operating point using information value (IV) [17]. We focus exclusively on those instances that receive conflicting predictions from the pair of models. The hypothesis is that, if a feature (or a set of features) has the ability to discriminate those instances, then this feature must carry some sort of information that is overlooked in the features used in one of the current two models. In other words, the available features in the model cannot support reliable differentiation between classes, and thus cause the disagreements. Incorporating this new feature into the model should provide new discriminative power to the model, and thus help alleviate the disagreements.

We calculate each feature's IV at a given operating point on two sets of instances that represent two types of disagreements. The first set of instances are those in TP_A and TP_B , representing disagreements on events (true positives), while the second set of instances are those in FP_A and FP_B , representing disagreements on non-events (false positives). Mathematically, IV is calculated by $IV = \sum_{i=1}^C \left[\left(\frac{E_i}{E} - \frac{NE_i}{NE} \right) \times WOE_i \right]$, where C is the number of categories in a feature, E_i is the number of events in category i , NE_i is the number of non-events in category i , E is the total number of events, NE is the total number of non-events, and WOE_i , which stands for *weight-of-evidence*, is defined by $WOE_i = \log\left(\frac{E_i}{E}\right) - \log\left(\frac{NE_i}{NE}\right)$. Notice that IV is non-negative because the signs of $\frac{E_i}{E} - \frac{NE_i}{NE}$ and WOE_i are the same.

WOE and IV provide a simple but powerful way of making sense of a feature [17], [24]. However, we experience two flaws when applying them in analyzing the oracle features we created.⁴ The first flaw is that they treat each category in a feature equally, ignoring the fact that small counts in a category can lead to unreliable statistics; and the second flaw is that IV has a bias toward giving a higher value for features including more categories (that is, features with high cardinality). The first flaw is easy to spot and has been addressed in several previous studies [25], [26]. The second one is tied to IV and subtle to identify when the cardinality of a feature is low, as is often the case in features used in previous studies [17], [25], [26]. The problem is that, because each summand on the right side of the IV formula can never be a negative value, adding a large number of elements each having a small value can potentially lead to a large sum. We overcome these two flaws by introducing the following new

formulas, inspired by the m -estimate method for probability estimates [25]:

$$RWOE_i = \log\left(\frac{E_i + m \times \frac{E}{E+NE}}{NE_i + m \times \frac{NE}{E+NE}}\right) - \log\left(\frac{E}{NE}\right) \quad (4)$$

$$RIV = \sum_{i=1}^C \left[\left(\frac{E_i + m \times \frac{E}{E+NE}}{E} - \frac{NE_i + m \times \frac{NE}{E+NE}}{NE} \right) \times RWOE_i \right] \quad (5)$$

where m is a smoothing parameter. We name these two formulas as *robust weight-of-evidence* (RWOE) and *robust information value* (RIV), respectively. The idea is that, in each category of a feature, we add $m \times \frac{E}{E+NE}$ "artificial" events and $m \times \frac{NE}{E+NE}$ "artificial" non-events, given the fact that $\frac{E}{E+NE}$ and $\frac{NE}{E+NE}$ actually represent event rate and non-event rate, respectively. The parameter m controls how much $RWOE_i$ is shifted toward the population average. A large m will make the first part on the right side of Equation 4 close to $\log\left(\frac{E}{NE}\right)$, resulting in a zero $RWOE_i$ value (that is, population average). As the result, this category will not contribute anything to the RIV calculation of this feature, which effectively mitigates the bias from the traditional IV formula.⁵

IV. APPLICATIONS

In this section, we present two real world event detection applications to demonstrate the usefulness of the LFD framework. The first application is CTR prediction for online advertising, while the second is anomaly detection for credit card transactions.

A. CTR Prediction

The goal of CTR prediction is to predict the probability that a user makes a click on an ad. The dataset used in this application is the Avazu dataset [20], which contains more than 40 million instances across 10 days. Each instance is an impression and includes 21 anonymized raw categorical features.

LFD always works on a pair of models. In this application, the pair of models consist of a logistic regression (LR) model (Model A) and a graph neural network model known as FiGNN [27](Model B). Both models use the 21 anonymized raw categorical features as input, where the LR model encodes these features using "hash trick" [18], while FiGNN uses a novel architecture aiming at capturing interactions from these raw features automatically. The LR model is trained using an online learning algorithm [28], and its score is aligned against the FiGNN score using the score alignment algorithm introduced in Sec. III-B. When preparing data for model training and testing, we follow a production-like approach: we reserve a dataset for model testing; this dataset cannot be touched during model training. More details on data splitting are discussed in Appendix A. We view LR in this pair as a simpler model because it is a linear model including only 21 raw features, and view FiGNN as a more advanced model because it has a more complex structure designed for capturing feature interactions automatically.

⁵We empirically chose $m=1000$ for both CTR prediction and transaction anomaly detection in our experiments.

⁴A majority of our oracle features are designed to capture interactions among raw features, and thus tend to have high cardinality. For instance, the concatenation of User-ID, Site-ID, and Advertiser-ID results in a three-dimensional feature with high cardinality, where the number of instances in some categories of the feature is small.

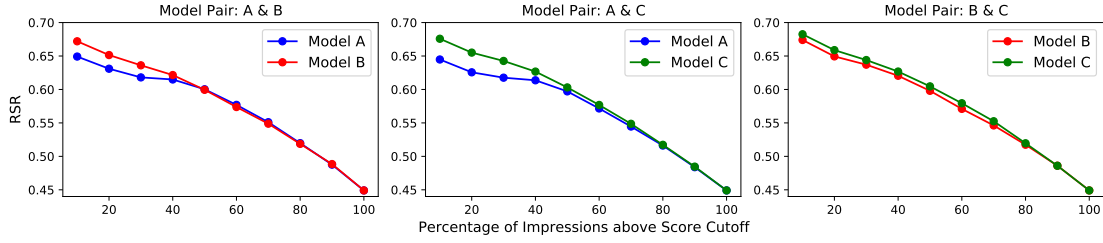


Fig. 4. Relative success rates between model pairs (CTR prediction) under different score cutoffs (the horizontal axis).

We create about 200 oracle features based on our understanding on the data and domain knowledge, paying close attention to feature interactions, especially interactions between users and websites (publishers). We perform feature importance analysis using RWOE and RIV on these 200 features on two sets of instances receiving conflicting predictions from this pair of models. The first set of instances are those in TP_A (label 0) from the LR model and those in TP_B (label 1) from the FiGNN model, and the second set of instances are those in FP_A (label 0) from the LR model and those in FP_B (label 1) from the FiGNN model. Appendix B lists some top-ranked features by RIV based on these two sets of instances. We select 70 features to train another LR model (Model C), using the same setting as the simpler LR model (Model A). The 70 features consist of 60 top-ranked oracle features by RIV, and 10 raw features from the 21 anonymized raw categorical features for diversity consideration.

Fig. 4 presents the relative success rates among the three models. The left panel shows that the FiGNN model (Model B) outperforms the simple LR model (Model A) in the high score region. This is not surprising because FiGNN is a more complex model than the simple LR model. The middle panel shows that the new LR model (Model C) outperforms the simple LR model (Model A) in the high score region as well. This is not surprising either because the new model includes more features. However, this result indicates that these oracle features indeed carry critical information missed by the features in the simple LR model. This finding is important because these oracle features have been selected by pretending *not* to know any inner details of the simple LR model, except its predictive scores. Furthermore, the right panel shows that the new LR model also outperforms the FiGNN model across all the operating points, including those in the high score region, although the margin is not as great as that shown in the middle panel. This indicates that these oracle features capture some important feature interactions overlooked by FiGNN, although FiGNN, by design, should have captured these feature interactions automatically. This experiment confirms our hypothesis in Sec. III-E.

The new LR model also outperforms FiGNN when evaluated by AUC and LogLoss, as shown in Table I. Also included in Table I is a LR model trained using the full 200 oracle features (Model D). The model shows a worse performance than Model C, suggesting the model has been overfitted while the model with LFD selected features (Model C) helps reduce

model overfitting. Note that, the AUC and Logloss differ from the results reported in [27] because of different ways of splitting the data for model training and testing, as discussed in Appendix A.

TABLE I
AUC AND LOGLOSS FROM FOUR MODELS.

Model	AUC	LogLoss
Model A	0.7431	0.3991
Model B	0.7470	0.3989
Model C	0.7530	0.3953
Model D	0.7483	0.3975

Another interesting observation from Fig. 4 is that, as the penetration goes deeper, the advantage of FiGNN and the new LR model over the simple LR model vanishes (refer to left and middle panels). We observe similar phenomenon when pairing the simple LR model with several other state-of-the-art CTR prediction models [29]–[31]. This indicates that the benefit of using a more complex model most likely comes from working on instances in the high score region, as is often the case for CTR prediction and transaction anomaly detection. We conjecture this observation may hold for any classification problems involving highly unbalanced datasets. We will test this hypothesis in future work.

Fig. 5 shows two types of disagreements among the three models: disagreements on true positives (clicks, left panel) defined as $(TP_A + TP_B) / (TP_{AB} + TP_A + TP_B)$, and disagreements on false positives (non-clicks, right panel) defined as $(FP_A + FP_B) / (FP_{AB} + FP_A + FP_B)$. There are two interesting observations from Fig. 5. The first observation is that, as the penetration goes deeper, disagreements between model pairs become smaller. This is particularly obvious for the true positives (clicks). The reason is that when we include more instances by setting a lower score cutoff, a majority of clicks are captured by all three models. The second observation is that disagreements on false positives are much more serious than disagreements on true positives. This observation motivates us to ponder the following question: in CTR prediction, or event prediction in general, all current efforts are focused on identifying events. Should we also put efforts to identify non-events, that is, try to reduce false positives? We will revisit this point in Sec. V-C.

B. Payment Transaction Anomaly Detection

The goal of transaction anomaly detection is to predict the probability that a payment card transaction is abnormal.



Fig. 5. Disagreements between CTR prediction models.

The pair of models used in this application is a gradient boosting machine (GBM) [32] and a recurrent neural network (RNN) model [33], [34], trained using one-year historical data involving billions of payment card transactions. We conduct two experiments in this application. In the first experiment, we want to see whether the LFD framework can help enhance the performance of the RNN model. Following the same procedure for CTR prediction, we first create about 300 oracle features based on domain knowledge, paying close attention to feature interactions, especially interactions between cardholders and merchants. We then perform feature importance analysis using RWOE and RIV on these 300 features on two sets of instances receiving conflicting predictions from the pair of models (Note the RNN score has been aligned against the GBM score using the score alignment algorithm introduced in Sec.III-B.) We next combine a set of selected features from the above analysis with the existing features (mainly feature embeddings created using a representation learning framework [35]) in the RNN model and train a new RNN model. Fig. 6 shows the performance lifts expressed in terms of anomaly catch rate at different operating points by the new RNN model over the initial RNN model discussed in [34]. We witness notable performance improvement from the new model. For instance, at the operating point where 1% of overall transactions are above the given score cutoff, we see a lift of 5.69%, and at the operating point where 5% of overall transactions are above the given score cutoff, we see a lift of 3.14%. The improvement is substantial considering the sheer volume of transactions processed by the payment network.

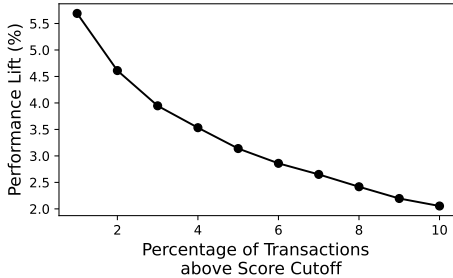


Fig. 6. Performance lift of the new RNN model over the initial RNN model.

In the second experiment, we investigate whether the LFD

framework can help explain the reason that ensembles, in this case GBM and RNN ensemble, can outperform each individual model, and see whether disagreements are reduced after the ensemble, which is one of the requirements from our business partners. We set the ensemble weights as 0.5 for both models and want to see how the ensemble will work if we treat two model scores equally.⁶

Fig. 7 shows relative success rates among three models, the GBM model (Model A), the RNN model (Model B), and the ensemble (Model C). The RNN model and the ensemble outperform the GBM model with substantial margins (the left panel and the middle panel), but this is not the purpose of this experiment. The interesting part is the fact revealed by the curves in the right panel – the GBM model is a “weaker” model, but combining it with the “stronger” RNN model using equal weights can still beat the RNN model at almost all the operating points in the high score region (notice that the scale of the x -axis is only up to 10% transactions), measured by the relative success rates.

Fig. 8 shows the disagreements on true positives (anomalies, left panel) and false positives (non-anomalies, right panel) among the three models. We can see that ensemble significantly reduces disagreements on both true positives and false positives. For instance, before the ensemble, at 1% of overall transactions, the disagreement between GBM and RNN on true positives is 56.44%. After the ensemble, the disagreement between GBM and the ensemble becomes 43.94%, a reduction of 22.14%; and the disagreement between RNN and the ensemble becomes 17.49%, a reduction of 69.01%. The reduction is also notable on false positives: before the ensemble, the disagreement between GBM and RNN is 82.22%. After the ensemble, the disagreement between GBM and the ensemble becomes 51.03%, a reduction of 37.94%; and the disagreement between RNN and the ensemble becomes 53.48%, a reduction of 34.96%. This experiment helps our business partners better understand what role each model in the pair has played in the ensemble.

V. DISCUSSION

In this section, we discuss some limitations of the LFD framework, and share our thoughts on future work.

A. Identifying the Root Cause of Disagreements Is Hard

LFD provides a principled and practical framework for identifying and leveraging disagreements between two models, without resorting to sophisticated algorithms as those proposed in [4], [5]. We approach the problem from a feature importance analysis perspective, but we do realize that disagreement can occur due to a slew of factors [36]–[38], and feature is just one of them. These factors can affect a model’s behavior either individually or in combination, which makes the process of identifying the root cause of disagreements a real challenge. However, based on our own experiences in developing and

⁶In practical deployment, we assign a larger weight to the RNN model. Deriving the best ensembling weights based on the models’ disagreement has been published in our another work but out of the focus of this paper.

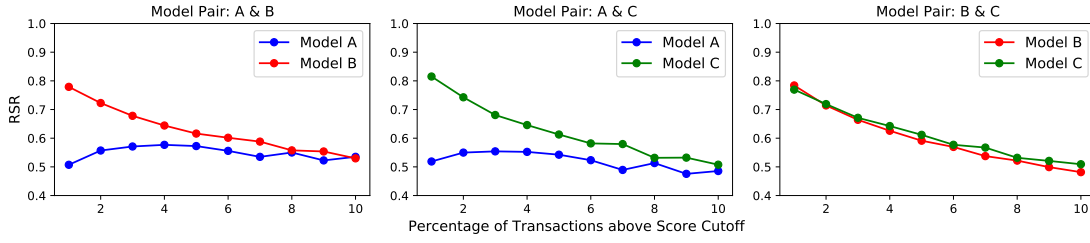


Fig. 7. Relative success rates between model pairs (anomaly detection).

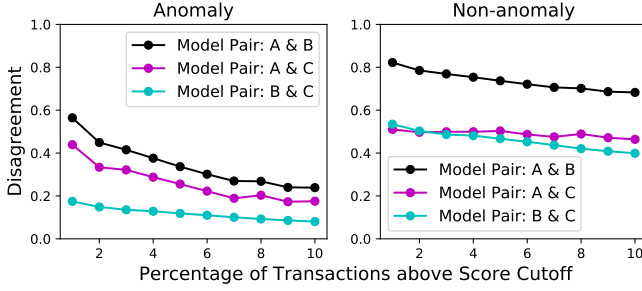


Fig. 8. Disagreements between anomaly detection models

deploying real production systems, when a problem occurs, the root cause, in a majority of cases, lies in the features used in the model – either important features indicative of the event are not included, or the predictive power of some existing features becomes illusive or degrades significantly due to changes in users’ behaviors. This agrees very much with the observation in [16]: “At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.”

B. Feature Importance Analysis by Groups

RIV provides a simple method of measuring the discriminative power of a predictive feature, without the need of model training, but it has one major limitation - features having high RIV often come from the same group, as seen in the top CTR features from Fig. 9 in Appendix B, which makes features into the model less diversified. A remedy to this problem is to first divide oracle features into groups (for instance, put raw features into the first group, two-dimensional features into the second group, and so on) and then select representative features from each group based on their RIV. This is the approach used in the current work and it works well. Further, we may combine RIV with a more sophisticated feature ranking method involving model training (see, for instance, [2], [3]), where RIV first selects a set of features from the oracle feature pool (can be redundant, but the size is smaller than the full feature set), and the sophisticated method then makes the final selection.

C. Focusing on Reducing False Positives

One finding from our experiments is that disagreement from non-events is much more serious than that from events.

At first glance we may think this is a known fact because non-events are abundant while events are rare. A moment of reflection may suggest that, for event detection, the hardest part might not be in how to identify events, but in how to identify non-events (more precisely, to reduce false positives). When an event happens, for instance, a person clicks an ad or commits an illicit transaction, there are usually some signals indicative of the event – this is the fundamental of any event detection. On the contrary, for non-events, the signals surrounding them tend to be weak. Therefore, it is actually the behavior of users from non-events that makes event detection hard. Current efforts in event detection are mainly focused on making correct predictions on events (for instance, identifying clicks and capturing abnormal transactions), but largely overlook the benefit of making fewer mistakes on non-events. This is especially problematic for transaction anomaly detection, because wrongly declining a legitimate transaction (false positive) would not only incur revenue loss, but also annoy the cardholder and damage the reputation of the organization. Creating more oracle features aiming at identifying non-events, besides those for events, and incorporating them into the LFD framework, will be performed in future work.

VI. RELATED WORK

Our work is inspired by the co-training algorithm [39]. The idea of using a pair of classifiers each focusing on a separate set of features, and using one view to help another, has greatly influenced our thoughts on LFD. Using a pair of models is also the key ingredient of several novel deep learning architectures [40]–[42], where one model guides another model or helps each other during training. LFD also works on a pair of models, but with a very different motivation: rather than tracking model behavior during training by leveraging training dynamics [38], LFD intends to identify weakness in established models, for instance, a model deployed in production or a new model ready for upgrading the deployed model. Cascade modeling [10], [43], active learning [44], [45], and dataset cartography [38], capture a similar intuition to that of LFD – instead of working on full instances in the whole region, focusing on instances in regions where a model is most likely to prevail. Apart from improving each individual model from a model pair, the identified disagreement can also be used to compare the two models and better ensemble them based on their behavior discrepancies, as discussed in [46].

LFD is also reminiscent of some recent works on “predictive multiplicity” [4], [5], [36], [47], [48], particularly the methods proposed by Marx et al. [4] and Renard et al. [5]. Marx et al. introduce formal measures to evaluate the severity of predictive multiplicity. Renard et al. propose a model-agnostic algorithm to capture and explain discrepancies locally, extending the idea of LIME [2]. However, there exist notable differences between LFD and the above methods in how to locate those conflicting predictions. In [4], discrepancies among a set of classifiers are computed using integer programming by optimizing prediction accuracy on the training data set, where the classifiers are constrained to be linear. In [5], discrepancies are located by producing a set of segments using a computationally intensive procedure searching through the full training dataset. In contrast, the process of identifying discrepancies in LFD is much simpler, and there is no need to constrain classifier types (linear or nonlinear) or to involve sophisticated algorithms, which makes LFD more practical for large-scale industrial applications.

VII. CONCLUSION

The need for discovering the reasons behind model discrepancies arises in many machine learning applications. This is a non-trivial task because discrepancies can occur due to a slew of factors. This is especially true for human-facing machine learning applications such as CTR prediction and transaction anomaly detection where users’ behaviors are constantly changing. In this paper, we present the “Learning from Disagreement” framework to understand discrepancies between models from a feature importance analysis perspective. The core of LFD is to work on a pair of models, with one model serving as the mirror of the other. It focuses on instances that lead to contradictory decisions between this pair of models at a given operating point, gaining insights into the pair of models without knowing their inner operations, and offering actionable feedback for improving the model performance. We demonstrate the usefulness of this framework through two real-world event detection applications.

REFERENCES

- [1] “Netflix never uses its 1 million algorithm due to engineering costs,” 2012. [Online]. Available: <https://bit.ly/3n8NMWi>
- [2] M. T. Ribeiro, S. Singh, and C. Guestrin, “‘why should i trust you?’ explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.
- [3] S. M. Lundberg, G. G. Erion, and S.-I. Lee, “Consistent individualized feature attribution for tree ensembles,” *arXiv preprint arXiv:1802.03888*.
- [4] C. Marx, F. Calmon, and B. Ustun, “Predictive multiplicity in classification,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 6765–6774.
- [5] X. Renard, T. Laugel, and M. Detryniecki, “Understanding prediction discrepancies in machine learning classifiers,” *arXiv preprint arXiv:2104.05467*, 2021.
- [6] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers et al., “Practical lessons from predicting clicks on ads at facebook,” in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, 2014, pp. 1–9.
- [7] B. Zadrozny and C. Elkan, “Transforming classifier scores into accurate multiclass probability estimates,” in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 694–699.

- [8] D. Sculley, J. Snoek, A. Wiltschko, and A. Rahimi, “Winner’s curse? on pace, progress, and empirical rigor,” 2018.
- [9] T. Raeder, O. Stitelman, B. Dalessandro, C. Perlich, and F. Provost, “Design principles of massive, robust prediction systems,” in *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining*, 2012, pp. 1357–1365.
- [10] W.-T. Yih, J. Goodman, and G. Hulten, “Learning at low false positive rates,” in *CEAS*, 2006.
- [11] M. F. Dacrema, P. Cremonesi, and D. Jannach, “Are we really making much progress? a worrying analysis of recent neural recommendation approaches,” in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 101–109.
- [12] Z. C. Lipton and J. Steinhardt, “Troubling trends in machine learning scholarship,” *arXiv preprint arXiv:1807.03341*, 2018.
- [13] S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim, “A benchmark for interpretability methods in deep neural networks,” *arXiv preprint arXiv:1806.10758*, 2018.
- [14] A. Fisher, C. Rudin, and F. Dominici, “All models are wrong, but many are useful: Learning a variable’s importance by studying an entire class of prediction models simultaneously,” *J. Mach. Learn. Res.*, vol. 20, no. 177, pp. 1–81, 2019.
- [15] K. Patel, S. M. Drucker, J. Fogarty, A. Kapoor, and D. S. Tan, “Using multiple models to understand data,” in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.
- [16] P. Domingos, “A few useful things to know about machine learning,” *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.
- [17] N. Siddiqi, *Credit risk scorecards: developing and implementing intelligent credit scoring*. John Wiley & Sons, 2012, vol. 3.
- [18] O. Chapelle, E. Manavoglu, and R. Rosales, “Simple and scalable response prediction for display advertising,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 4, pp. 1–34, 2014.
- [19] “Iso 8583,” 2021, accessed: 2021-09-01. [Online]. Available: <https://bit.ly/3pfI0K0>
- [20] AnyAI, “OpenCTR,” <https://zenodo.org/record/2002072>, 2018.
- [21] “How to get the lowest price for impression-based advertising,” 2021, accessed: 2021-09-01. [Online]. Available: <https://bit.ly/2Zb37O6>
- [22] J. M. Kanter and K. Veeramachaneni, “Deep feature synthesis: Towards automating data science endeavors,” in *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, 2015.
- [23] P. Covington, J. Adams, and E. Sargin, “Deep neural networks for youtube recommendations,” in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 191–198.
- [24] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, “Visualizing deep neural network decisions: Prediction difference analysis,” *arXiv preprint arXiv:1702.04595*, 2017.
- [25] S. Džeroski, B. Cestnik, and I. Petrovski, “Using the m-estimate in rule induction,” *Journal of computing and information technology*, vol. 1, no. 1, pp. 37–46, 1993.
- [26] B. Zadrozny and C. Elkan, “Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers,” in *ICML*, vol. 1. Citeseer, 2001, pp. 609–616.
- [27] Z. Li, Z. Cui, S. Wu, X. Zhang, and L. Wang, “Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 539–548.
- [28] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin et al., “Ad click prediction: a view from the trenches,” in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 1222–1230.
- [29] W. Chen, L. Zhan, Y. Ci, M. Yang, C. Lin, and D. Liu, “Flen: leveraging field for scalable ctr prediction,” *arXiv preprint arXiv:1911.04690*, 2019.
- [30] W. Deng, J. Pan, T. Zhou, D. Kong, A. Flores, and G. Lin, “DeepLight: Deep lightweight feature interactions for accelerating ctr predictions in ad serving,” in *Proceedings of the 14th ACM international conference on Web search and data mining*, 2021, pp. 922–930.
- [31] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang, “Autoint: Automatic feature interaction learning via self-attentive neural networks,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1161–1170.
- [32] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of statistics*, pp. 1189–1232, 2001.
- [33] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

TP _A vs TP _B		FP _A vs FP _B		TP _{AB} vs FP _{AB}	
siteid_appid_c17	4.194388	siteid_appid_c17	4.671380	siteid_dvmodel_appid_c17	0.289480
siteid_appid_c14	3.973263	siteid_appid_c14	4.505817	siteid_dvmodel_appid_c14	0.282160
siteid_c17	3.646253	siteid_dvmodel_appid_c17	4.335556	siteid_dvmodel_appid_c21	0.275388
siteid_appid_c21	3.630703	siteid_dvmodel_appid_c21	4.316343	siteid_dvmodel_appid_c19	0.267853
sitedomain_c17	3.592018	siteid_dvmodel_appid_c18	4.188084	siteid_dvmodel_appid_c18	0.261674
siteid_c14	3.583191	siteid_dvmodel_appid_c19	4.126306	siteid_devicemodel_appid	0.242305
siteid_dvmodel_appid_c21	3.563603	siteid_devicemodel_appid	4.040462	userid_siteid	0.240799
sitedomain_c14	3.529354	siteid_c17	4.016011	userid_appid	0.240561
siteid_dvmodel_appid_c17	3.520299	siteid_c14	4.009016	userid_siteid_appid	0.240351
siteid_dvmodel_appid_c18	3.509787	sitedomain_devicemodel_appid	3.999635	sitedomain_devicemodel_appid	0.239667

Fig. 9. Top 10 features by RIV for different populations.

- [34] W. Zhang, L. Wang, R. Christensen, Y. Zheng, L. Gou, and H. Yang, “Transaction sequence processing with embedded real-time decision feedback,” Oct. 19 2021, uS Patent 11,153,314.
- [35] C.-C. M. Yeh, D. Gelda, Z. Zhuang, Y. Zheng, L. Gou, and W. Zhang, “Towards a flexible embedding learning framework,” in *2020 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2020.
- [36] A. D’Amour, K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, C. Chen, J. Deaton, J. Eisenstein, M. D. Hoffman *et al.*, “Underspecification presents challenges for credibility in modern machine learning,” *arXiv preprint arXiv:2011.03395*, 2020.
- [37] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, “Hidden technical debt in machine learning systems,” *Advances in neural information processing systems*, vol. 28, pp. 2503–2511, 2015.
- [38] S. Swayamdipta, R. Schwartz, N. Lourie, Y. Wang, H. Hajishirzi, N. A. Smith, and Y. Choi, “Dataset cartography: Mapping and diagnosing datasets with training dynamics,” *arXiv preprint arXiv:2009.10795*.
- [39] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training,” in *Proceedings of the eleventh annual conference on Computational learning theory*, 1998, pp. 92–100.
- [40] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, “Co-teaching: Robust training of deep neural networks with extremely noisy labels,” *arXiv preprint arXiv:1804.06872*, 2018.
- [41] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, “Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 2304–2313.
- [42] E. Malach and S. Shalev-Shwartz, “Decoupling” when to update” from” how to update,” *arXiv preprint arXiv:1706.02613*, 2017.
- [43] D. Sculley, M. E. Otey, M. Pohl, B. Spitznagel, J. Hainsworth, and Y. Zhou, “Detecting adversarial advertisements in the wild,” in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 274–282.
- [44] Z. Lu, X. Wu, and J. C. Bongard, “Active learning through adaptive heterogeneous ensembling,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 2, pp. 368–381, 2014.
- [45] I. Muslea, S. Minton, and C. A. Knoblock, “Active learning with multiple views,” *Journal of Artificial Intelligence Research*, vol. 27, pp. 203–233, 2006.
- [46] J. Wang, L. Wang, Y. Zheng, C.-C. M. Yeh, S. Jain, and W. Zhang, “Learning-from-disagreement: A model comparison and visual analytics framework,” *IEEE Transactions on Visualization and Computer Graphics (arXiv preprint arXiv:2201.07849)*, 2022.
- [47] M. Pawelczyk, K. Broelemann, and G. Kasneci, “On counterfactual explanations under predictive multiplicity,” in *Conference on Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 809–818.
- [48] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.
- [49] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman, “Leakage in data mining: Formulation, detection, and avoidance,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 4, pp. 1–21, 2012.

APPENDIX

A. Data Splitting

Avazu dataset is collected during a time span of 10 days. We use the first nine days’ data for model training, and last day’s data for model testing. Notice here the way of splitting data is different from that in FiGNN [27], where the whole 10 days of data are mixed together for both training and testing. Our splitting is more realistic and closer to production settings. Table II should make this point clear where we perform an overlap analysis between entities in the first nine days and in the last day. Take “Device ID” as an example. There are 2484613 Device IDs in Day 1 through Day 9, and 285944 Device IDs in Day 10. Among the 285944 Device IDs, 201795 (70.57%) cannot be found in the first nine days. Apparently, including the last day’s data for model training will lead to information leakage [49].

TABLE II
OVERLAP BETWEEN ENTITIES IN DAY 1-9 AND 10.

Entity	Day1-9	Day 10	Day 10 Only	Percentage
Device ID	2484613	285944	201795	70.57
Device IP	6134351	1023643	595135	58.14
C14	2470	1191	156	13.10

B. Example Top-Ranked Features by RIV

Fig. 9 presents top 10 features by RIV based on two types of disagreements from Model A and Model B: disagreements on true positives (refer to left panel), and disagreements on false positives (refer to middle panel). It is interesting to notice that these two sets of features are pretty much in agreement. Also included in the table are the top 10 features from the agreed instances (refer to the right panel). These are instances either correctly classified by both models (TP_{AB}) or incorrectly classified by both models (FP_{AB}). Intuitively, it is hard to differentiate instances in TP_{AB} from instances in FP_{AB} . This is indeed the case: the RIV values in the right panel all have a smaller value, indicting the signals used to separate these two populations are very weak.