Proxima: A Proxy Model-Based Approach to Influence Analysis

Sunny Shree
sunny.shree@mavs.uta.edu
Dept. of Computer Science and Engineering, The University of Texas at Arlington
Arlington, Texas, USA

Yu Lei
ylei@cse.uta.edu
Dept. of Computer Science and Engineering, The University of Texas at Arlington
Arlington, Texas, USA

Raghu N. Kacker
raghu.kacker@nist.gov
Information Technology Laboratory, National Institute of Standards and Technology
Gaithersburg, Maryland, USA

D. Richard Kuhn
d.kuhn@nist.gov
Information Technology Laboratory, National Institute of Standards and Technology
Gaithersburg, Maryland, USA

ABSTRACT

Machine learning (ML)-based Artificial Intelligence (AI) systems rely on training data to function, but the internal workings of how ML models learn from these data are often opaque. Influence analysis provides valuable insights into the model's behavior by evaluating the effect of individual training instances on the model's predictions. However, calculating the influence of training data can be computationally expensive. In this paper, we propose a proxy model-based approach to influence analysis called Proxima. The main idea of our approach is to use a subset of training instances to create a proxy model that is simpler than the original model, and then use the proxy model and the subset of training instances to perform influence analysis. We evaluate Proxima on ML models trained using seven real-world datasets. We compare Proxima to two state-of-the-art influence analysis tools, i.e., FastIF and Scaling-Up. Our experimental results suggest that the proposed approach can successfully perform and speed up the influence analysis process, and, in most cases, perform better than FastIF and Scaling-Up.

KEYWORDS

Datasets, Machine Learning, Neural Networks, Influence Analysis, Influence Functions, Training Data Similarity

1 INTRODUCTION

Artificial Intelligence (AI) is revolutionizing various aspects of our lives by leveraging machine learning (ML) algorithms to analyze data, identify patterns, and make predictions This transformation is evident across industries like healthcare, transportation, finance, retail, and manufacturing, leading to improved efficiency and automation of tasks[33]. Many AI systems are built upon ML algorithms that perform tasks, such as classifications and predictions. ML relies on training data to function, but the way ML models learn from and use these data is often opaque [28, 33].

In ML, influence analysis determines how specific instances in training data impact an ML model's predictions [9, 19]. It helps understand the model's behavior, identify and correct errors, and enhance its robustness and reliability [13]. Influence analysis reveals the most significant instances for the model's decisions, providing valuable insights into its decision-making process [30]. It also helps identify and remove instances causing overfitting and add instances to improve generalization [13].

Given training dataset D, ML model M makes a decision Y for an instance X. The goal of influence analysis is to rank the training instances in D based on their impact or influence on the decisions made by M, for X. The key concept of influence analysis is to trace the relationship between an ML model's parameters and predictions back to its training data. During the training process, the model's parameters are adjusted based on prediction errors, which means the predictions are influenced by the training data instances used to adjust those parameters. By conducting an influence analysis, we can rank the training instances based on their impact or influence on the model's decisions. This analysis helps in identifying influential data points that have a significant effect on the model's performance [19].

Existing approaches can be divided into two categories:

(1) *Retraining Based:* These approaches [7, 9, 18, 32, 37] involve removing one data point from the dataset, retraining the model, and comparing the difference in predictions for that point to the original model's predictions. Retraining-based approaches can be easily implemented across various model types and perform accurate influence analysis. However, their primary drawback is the computational cost of repeated model retraining, especially for large datasets and/or complex models. (2) *Gradient Based:* These approaches [4, 5, 13, 14, 19, 20, 30, 32] in influence analysis compute the gradient of the model's output with respect to the influential instance to measure its impact. The gradient indicates the sensitivity of the output to input changes, with larger gradients indicating greater influence. These approaches quantify instance influence without repeated model retraining. However, their computational complexity is influenced by the dataset size and model parameters, leading to slower computations for large models. The high-dimensional derivatives and increased calculations required for larger datasets can significantly slow down the analysis process.

In this paper, we propose a proxy model-based approach called Proxima to influence analysis. Given an ML model M trained on the dataset D, which classifies an instance X into class Y, Proxima identifies and ranks the top-K most influential instances in D based on their impact on the model's prediction. We note that in practical applications, e.g., model debugging [36], influence analysis often focuses on a few most influential instances, due to resource limitations.

The main idea of our approach, Proxima, is constructing a simplified proxy model by using a subset of training instances, referred

to as the proxy subset, that preserves the most influential instances. To identify the proxy subset, we utilize a ratio of loss over distance. A training instance is identified to be in the proxy subset if its loss-over-distance ratio is smaller than a threshold. Our approach is inspired by the empirical evidence presented by Koh et al. in their influential function paper, which shows that the amount of influence of a training instance I on the model decision for X depends on the training loss of I and the distance between I and X [19].

In addition, the proxy model is trained on the instances in the proxy set that has been relabeled based on the predictions of the original model M, instead of the original label in the training set. This approach ensures that the proxy model preserves the training loss of the instances in the proxy subset while maintaining the unchanged distances between them. As a result, the proxy model retains the relative order of the instances' loss values, preserving the relative order of their influence. It is important to note that the proxy model does not retain the exact magnitudes of the loss values. This design choice aligns with the goal of Proxima, which is to preserve the relative order of influence rather than the precise values themselves.

At a high level, Proxima performs the following steps. During the preprocessing phase, Proxima applies metric learning on the dataset D to enable distance computation between any pair of instances. It also calculates the training loss for each training instance in D. Next, Proxima constructs a proxy model in a two-step process. In step 1, it iterates over D and computes a ratio of loss and distance for each training instance. If the ratio is below a threshold t, the instance is included in the proxy subset D'. Moving to step 2, given the proxy subset D' and the model M, Proxima utilizes M to classify each instance in D'. The resulting classifications serve as new labels for the instances in the proxy subset. Proxima then fits a proxy ML model, denoted as M', to the proxy subset D' using these updated labels. We refer to M' as the proxy model, for instance, X. Finally, Proxima uses existing influence analysis techniques such as Leave-One-Out [7] or Influence Function [19], by passing instance X, the proxy model M', and the proxy subset D' as inputs, to rank the instances in D' based on the amount of influence.

It is important to note that Proxima can work together with existing influence analysis methods, such as retraining-based and gradient-based approaches, to further accelerate the analysis. By substituting the entire dataset and original model with the proxy subset and proxy model as inputs, the computation cost of influence analysis is significantly reduced.

We evaluate our influence analysis approach using ML models trained on seven datasets: Adult-Income for income prediction [24], Lending Club for loan decisions [6], German-Credit for credit risk assessment [16], a dataset COMPAS for bail decision [1], MNIST [8], Fashion-MNIST [38], and CIFAR-10 [21]. We use pre-trained ML classifiers (models) with two ML algorithms, neural networks (NN) and random forest (RF) for tabular datasets, and VGG13 and Resnet-20 ML models for image datasets. For all models, experimental results on 500 randomly selected instances from each test dataset suggest that we can successfully perform influence analysis by ranking the top-10 training instances based on their influence on the model decision for all 100 % instances. Furthermore, Proxima had the highest average accuracy of around 95 %, which outperform the state-of-the-art FastIF [13] by 3 % and Scaling-Up [30] by 11 %. In terms of speed-up, Proxima is the fastest, with the shortest average time for all four datasets compared to Scaling Up and FastIF.

Contributions In summary, our contributions are as follows:

- We develop a new approach to speed up influence analysis. The novelty of Proxima is that we identify a subset of the training set that preserves the most influential instances and build a proxy model that can be used to perform influence analysis in place of the original dataset and model. To the best of our knowledge, our work is the first in this domain to use the loss-over-distance ratio to identify the subset. Proxima can be used together with existing influence analysis to speed up the influence analysis process.
- We present an experimental evaluation of Proxima. The evaluation shows that Proxima is effective in identifying the top-10 most influential instances and outperforms two state-of-the-art approaches.
- We implement Proxima and make the tool, experimental data, including design, datasets, and results, available to the public at [29].

2 BACKGROUND

This section presents concepts that are important for Proxima to perform Influence Analysis.

2.1 Metric Learning

Metric learning is a technique employed in ML applications to acquire a distance function that measures the similarity between samples. Unlike pre-defined metrics such as Euclidean or cosine distances, metric learning aims to directly learn a task-specific metric from the available training data. This approach often achieves better performance by capturing the unique distributions and characteristics of the data [26].

In our experiments, we utilize two state-of-the-art metric learning techniques:

(a.) Neighborhood Component Analysis - Goldberge et al. proposed Neighborhood Component Analysis (NCA), a distance metric learning technique, which aims to learn a transformation of the feature space that optimizes the distances between similar instances while preserving the distances between dissimilar instances [12]. NCA is very effective in learning similarity to generate nearest neighbors because it aims to find a distance metric that maximizes the likelihood of observing the nearest neighbors of each instance in the training data [35].

(b.) Siamese Deep Metric Learning - Yi et al. introduced a Deep Metric Learning (DML) method that employs a Siamese deep neural network to learn a similarity metric directly from image pixels [39]. Their approach combines discriminative feature learning and similarity measurement within a unified deep framework. The network architecture consists of symmetrical subnetworks connected by a cosine similarity layer, incorporating convolutional and fully connected layers. This method offers two key advantages: the direct learning of a similarity metric from image pixels and the simultaneous capture of color and texture information through multichannel filters in images [26].

It is important to highlight that the concept of similarity varies across different models, as each model incorporates its own understanding of similarity in its decision-making process. As a result, standard distance metrics might not accurately capture the similarity as perceived by a specific model. This underscores the significance of employing specialized techniques such as Neighborhood Component Analysis (NCA) and Siamese Deep Metric Learning to effectively model similarity. These advanced algorithms are considered state-of-the-art in the field of metric learning, offering enhanced performance and highlighting the necessity for customized approaches[26].

2.2 Influence Analysis

In ML, there are several ways of performing influence analysis. As a tool for analyzing the behavior of ML models, influence analysis has several advantages. They show how individual training instances affect the model's predictions, allowing for a more localized interpretation of the model's behavior. They can also be applied to a wide range of model architectures and loss functions, making them a versatile tool. As a drawback, influence analysis only provides information about the impact of individual instances on the model's predictions and may not accurately represent the model's overall behavior. In this subsection, we present the two most common approaches for performing influence analysis.

2.2.1 Leave-Out-One. Cook and Weisberg first described the Leave-Out-One (LOO) approach in their book "Residuals and Influence in Regression" [7]. It is used to measure the influence of an individual data instance on the outcome of a statistical model. The approach involves removing one data instance from the dataset at a time, re-fitting the model, and then computing the difference in the model's predictions with and without that data instance. This process is repeated for each data instance in the dataset, generating an estimate of how much each data instance influences the model's predictions.

The leave-one-out (LOO) method for influence analysis can be computationally intensive due to the requirement of refitting the model after an instance is removed. When the model is complex or the training set is large, or both, refitting can be very expensive [23].

2.2.2 Influence Function. To perform influence analysis, Koh et al. proposed using influence functions, a robust statistics approach, to measure the impact of a training instance on model predictions [19]. Unlike LOO, which involves removing instances, influence functions estimate model change by up-weighting the loss of a training instance.

On a high level, the influence function works by first calculating the optimal model parameters, then up-weighting a training instance by a small amount, and approximating the change in the model's parameters and loss value for a test instance. This approximation is then used to calculate the influence of the up-weighted training instance on the model.

While influence functions are versatile and flexible in their ability to analyze model behavior, they do have some weaknesses and limitations [5, 30]. Firstly, the calculation of influence functions can be computationally expensive, especially for complex models with many parameters. The cost of computing the influence function can be high, particularly when the dataset is large and the model has many parameters. Furthermore, their reliance on first-order Taylor series expansions may result in approximation errors, especially in models with complex interactions between parameters and inputs.

3 APPROACH

This section presents Proxima, a proxy model-based approach to influence analysis. Assume an ML model M is trained on a dataset D. Let X be an instance that is classified by M into a class Y. To perform the influence analysis, we want to identify and rank the top-K most influential instances in the dataset D based on the influence they had in classifying X into class Y. Figure 1 presents the overview of Proxima. It consists of three major phases - *(1) Pre-processing:* Metric Learning, where the metric learning model is trained on the dataset; and Training-Loss, where training loss of each training instance is calculated; *(2) Build a Proxy Model:* Identifying the Subset, where we identify a subset of D; Training a Proxy Model, where we train a simpler model on the subset of D; and *(3) Influence Analysis:* Computing the amount of influence, where we compute the amount of influence and then rank the top-K influential instances.

3.1 Pre-processing

In this phase, Proxima performs two essential tasks: metric learning and calculating the training loss.

Proxima utilizes metric learning algorithms to capture the underlying similarity between instances in the dataset D. Through an optimization process, these algorithms learn a robust metric that enables pairwise distance computations, facilitating meaningful similarity analysis and comparison.

In addition to metric learning, Proxima also calculates the training loss of each instance within the dataset, D, when evaluated against the original model, M. To calculate the training loss, Proxima employs the model, M, to predict the output for each training instance in the dataset. For each instance, the absolute difference between the predicted output and the actual output is computed. This absolute difference serves as a measure of the error, or loss, for each training instance, providing valuable information for calculating the ratio of $loss/distance$.

These steps only need to be performed once per dataset and can be reused for different test instances.

3.2 Build a Proxy Model

In this phase, Proxima identifies the subset and trains a proxy model. This phase has two major steps.

3.2.1 Step-1 : Identifying the Subset. Proxima identifies a proxy subset, denoted as D', from the dataset D by evaluating the ratio of training loss of the instance to the distance between X and the instance in D

This consists of the following 3 steps:

1. Compute the distance between X and every other instance in, D using the metric learned in 3.1.

$$Distance(n_i) = metricLearning(X, n_i) \quad (1)$$

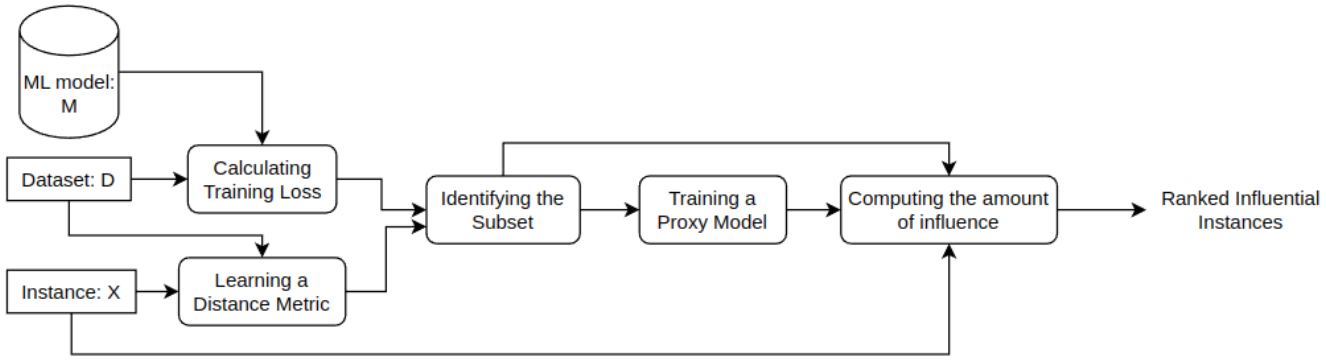

Figure 1: Work-flow of Proxima.

where X is the test instance and n_i is the i_{th} training instance in the dataset D.

2. Compute the ratio of loss over distance for each training instance, where *loss* is the training loss of each instance in D computed in 3.1.

$$Ratio(n_i) = \frac{Loss(n_i)}{\lambda \times Distance(n_i)} \tag{2}$$

Here, lambda (λ) is a scalar constant that is used to control the trade-off between the loss and the distance. A high value of λ will result in the loss having a greater impact, while a low value will result in the distance having a greater impact. In our experiments, we choose $\lambda = 1$. As part of future work, we intend to conduct an empirical study on various models and datasets to determine the optimal value of λ by holding one of the variables in *equation* (2) constant while varying the other, and vice versa, in order to fully understand the relationship between the variables and determine the optimal value of λ. We will examine the effect of this change on the amount of influence to determine whether there is a linear or non-linear relationship.

3. Create a proxy subset, $D' \subset D$ by selecting the instances that have a ratio less than a threshold t using the following equation:

$$D' = \{n_i \in \mathbb{D} \mid ratio(n_i) < t\} \tag{3}$$

where $|D|$ represents the size of the dataset D, $ratio(n_i)$ is the ratio of the $i-th$ element of D, and t is a threshold value.

The threshold t, used to create a proxy subset of the dataset around a test instance, is an important parameter to consider when performing sampling of dataset around X. There is no single best t that works well for all datasets; the optimal t is determined by the specific characteristics of the dataset, the chosen metric, and the problem's objective. An iterative approach was used to find the optimal value of t that produced the top-K most influential instances. The process began by incrementally increasing the value of t and comparing the generated instances until 80 % of the top-K most influential instances generated for the current value of t were also present in the set of most influential instances generated for the previous value of t.

The size of the proxy subset D' is an important consideration in the proxy model fitting process. In some cases, the threshold t may converge with as few as K instances in D'. To mitigate this, we set a minimum proxy subset requirement of $K \times 10$ instances. Here, K represents the desired number of influential instances to rank.

3.2.2 Step-2 : Training a Proxy Model. The goal of the proxy model is to mimic the original model M for the instances in D' by minimizing the difference between the predictions made for D and D'. If the prediction difference is minimal, the training loss will be retained, which we believe will maintain the relative order of the influential instances. Proxima begins this step by relabeling all instances in D'. This is done by passing each instance in D' as input to the original model M, which produces a prediction (classify) as output, using the following equation:

$$Y'_i = M(n_i), \forall n_i \in D' \tag{4}$$

where D' is the set of instances, n_i is an element of D', and $M()$ is the original ML model that takes n_i as input and makes a prediction (classify) as output.

After relabeling the instances in D' using equation (4), Proxima trains a proxy model M' on this proxy subset using the same model architecture as the original ML model M.

3.3 Influence Analysis

Given a test instance, a set of training data, and an ML model, there are several influence analysis tools that can be used to compute the amount of influence of each instance in D'. The two most commonly used influence analysis tools are LOO and Influence Function. Both of these tools take as inputs the instance X, proxy subset D', and a proxy model trained on D' and compute the influence of each instance in D' to rank them as output. In our experiments, we use the influence function to perform this step.

3.4 Choice of Ratio

In Proxima, the use of the ratio is employed to determine the selection of data instances in the proxy subset D'. Alternatively, a distance-based approach that considers only the proximity of instances is commonly used, assuming that nearby instances have a higher influence on each other due to their similarity. However, our observations indicate that instances with significant training loss or prediction errors can still exert substantial influence, even if they are not physically close to each other.

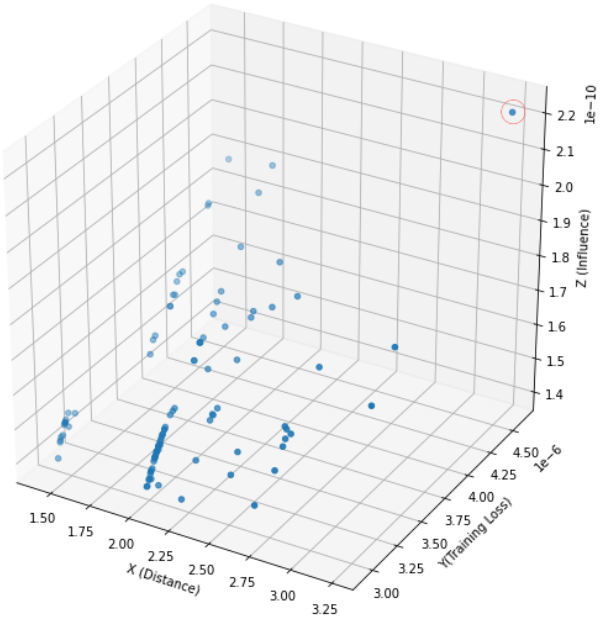

Figure 2: An example 3-D graph of influential instances, with distance on the X-axis, training loss on the Y-axis, and the influence value on the Z-axis. The blue dots represent the instances in the subset. The red circle denotes the most influential instance.

To investigate this further, we conducted experiments using a 3-dimensional visualization approach that incorporated distance, training loss, and influence values. Figure 2 presents an example graph illustrating this relationship. Notably, the most influential instance, represented by the red circle, is found at a distance of 3.23, while the remaining training instances are at a distance of 2.52 or less. A distance-based approach that primarily relies on the similarity of nearby instances would categorize the most influential instance at a distance of 3.23 as dissimilar or an outlier, potentially excluding it from the proxy subset. However, our experiments reveal that the ratio approach outperforms the distance-based approach in retaining the most influential instances. The ratio effectively captures the influence of instances with substantial training loss, ensuring their inclusion in the proxy subset and enhancing the overall effectiveness of the influence analysis process.

3.5 Limitations

While Proxima offers significant advancements in influence analysis, it is important to acknowledge its technical limitations.

The performance of Proxima can be influenced by the choice of hyperparameters, such as the threshold value for selecting the subset and the number of influential instances to consider. Finding the optimal values for these hyperparameters may require careful experimentation and tuning. Scalability can be a concern when dealing with high-dimensional datasets. As the dimensionality of the data increases, the computational complexity of Proxima may grow significantly, impacting the efficiency of the analysis. This scalability issue needs to be addressed to ensure the practical applicability of Proxima in real-world scenarios.

Proxima assumes that the dataset is static and does not account for changes or updates in the data over time. Analyzing dynamic datasets, where new instances are added, or existing instances are modified, poses a challenge for Proxima. Adapting Proxima to handle dynamic datasets and incorporating incremental learning techniques would be a valuable direction for future research. Another limitation of Proxima is its vulnerability to outliers. Outliers can disproportionately influence the analysis results and may lead to misleading conclusions. Developing robust techniques to identify and handle outliers within the influence analysis framework would enhance the reliability and accuracy of Proxima.

4 EXPERIMENT

In this section, we first present the experimental design, which includes the baselines for evaluation, the subject models and datasets, and the metrics used to assess the effectiveness of Proxima. Second, we present, discuss and contrast our findings with those of [30] and [13].

4.1 Research Question

- RQ1: How effective is Proxima in computing and ranking the top-10 most influential instances compared to the state-of-the-art influence analysis approaches?
- RQ2: How effective is Proxima in accelerating the analysis process of other influence analysis approaches?

4.2 Baselines

To evaluate the effectiveness of Proxima, we compare the results of Proxima with two state-of-the-art influence analysis tools.

(1) Scaling-Up: Schioppa et al. proposed a technique called scaling-up the influence function, which uses Arnoldi iteration to speed up the computation of the inverse Hessian matrix [30]. The Arnoldi iteration is a well-established method for estimating a matrix's dominant eigenvalues and eigenvectors. Schioppa et al. used this method to compute a low-rank approximation of the inverse Hessian matrix, which can then be used to efficiently compute the influence function. They present efficient calculation of influence functions for tracking predictions back to the training data.

(2) FastIF: Guo et al. proposed FastIF, which uses k-Nearest Neighbors (kNN) to narrow the search space to a subset of promising candidate data points, chooses optimal configurations that strike a balance between computation speed and quality when estimating the inverse Hessian-vector product, and introduces a fast parallel variant [13]. It provides a faster and more efficient alternative to the standard influence function method, increasing its applicability to large-scale ML problems.

It should be noted that the influence function [19] is used as the ground truth for evaluating *ResearchQuestion*, while the other two approaches (Scaling-Up and FastIF), which aim to speed up the computation of the influence function, serve as a baseline to compare against.

Table 1: Feature information and model accuracy.

Dataset	Samples	Categorical Features	Continuous Features	NN	RF
Adult-Income	48,842	7	6	0.83	80.80
Lending Club	10,000	4	4	0.82	79.67
COMPAS	7,214	4	1	0.67	0.64
German-Credit	1,000	15	5	0.77	0.72

4.3 Datasets and Models

We evaluate Proxima on the following four datasets:

(1) Adult Income: The dataset was created using data from the United States Census Bureau in 1994. The ML models trained on this dataset predict whether an individual's annual income is greater than or less than $50 000 [24].

(2) Lending Club: This dataset was created using loan data from Lending Club, a peer-to-peer lending company, over a five-year period (2007-2011) [6]. The ML models trained on this dataset predict an individual's ability to repay their loan.

(3) German-Credit: This dataset was created by analyzing loan data from individuals who obtained loans from a specific bank [16]. The ML models trained on this dataset predict the loan applicant's risk profile, determining whether the applicant is a good risk (likely to repay the loan) or a bad risk (unlikely to repay the loan).

(4) COMPAS: The Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) dataset was created using data collected for a study on recidivism decisions in the United States by ProPublica, a non-profit news organization [1]. The ML models trained on this dataset predict the likelihood of recidivism for bail applicants within the next two years.

(5) MNIST: The Modified National Institute of Standards and Technology (MNIST) dataset is a widely-recognized benchmark dataset in the fields of machine learning and computer vision [8]. It consists of a collection of 70 000 grayscale images of handwritten digits, ranging from 0 to 9. ML models trained on the MNIST dataset classify handwritten digits.

(6) Fashion MNIST: The Fashion MNIST dataset is a variation of the original MNIST dataset, tailored specifically for fashion and clothing recognition [38]. It consists of 70 000 grayscale images, depicting 10 different fashion items including T-shirts/tops, trousers, dresses, and more.

(6) CIFAR-10: The Canadian Institute For Advanced Research-10 (CIFAR-10) dataset is another popular benchmark dataset for image classification tasks [21]. It contains 60 000 color images and is divided into 10 different classes representing various objects such as airplanes, automobiles, birds, cats, and more.

Table 1 displays tabular dataset information such as the number of continuous and categorical features and the accuracy of the trained models. Each dataset is divided into 80 % training and 20 % test sets. For evaluation, we used a pre-trained neural network classifier for tabular data, a feed-forward neural network(NN) with one hidden layer of 12 ReLU units, and a Random Forest (RF) model [28].

For MNIST, Fashion MNIST, and CIFAR-10, we used the Visual Geometry Group 13 (VGG13) and ResNet-20 architecture proposed by Simonyan and Zisserman [34] and He et al. [15]. Following the architecture suggestion by Bae et al. [3], the base network underwent 200 epochs of training across all datasets, utilizing a batch size of 128. In the case of MNIST and Fashion-MNIST, a fixed learning rate was maintained throughout the training process. However, for CIFAR-10, the learning rate decayed by a factor of 5 at epochs 60, 120, and 160. To prevent overfitting, L2 regularization with a strength of 5×10^{-4} was employed, along with a damping factor of $\lambda = 0.001$.

4.4 Choice of Proxy Model

To train the proxy model, we utilized the same model architecture as the original model M for each dataset.

4.5 Metrics

In our experiment, we focus on generating the top-10 most influential instances, allowing for a more thorough evaluation of the system's critical components and a deeper understanding of the data's underlying patterns. The effectiveness of Proxima in generating quality counterfactuals is measured using the following two metrics:

(1) Accuracy: We benchmark the accuracy of influential instances generated by an approach by measuring the number of influential instances that match between the results of the approach and the results of the influence function proposed by Koh et al. [19]. A higher match % indicates greater accuracy.

(2) Computational Time: This metric measures the amount of time it takes to generate the top-K influential instances. We use Python's *time* module to compare runtime to measure the execution time required to generate influential instances [10]. Please note that the computation time metrics we report do not include preprocessing time, which is typically considered an amortizable cost, nor do they account for model construction time or influence analysis time.

All the experiments were carried out on Ubuntu 20.04.5 LTS system with Intel® Core™ i7-6700K CPU @ 4.00GHz × 8 processor, and NVIDIA Corporation TU106 [GeForce RTX 2060 SUPER] graphics card.

4.6 Choice of Ratio Threshold

In order to determine the optimal value of t that would result in the optimal proxy subset as described in Equation 3, we employed an iterative approach. The process involved setting an initial value of $t = 0.50$, creating the corresponding subset, training a proxy model, and identifying the top-10 most influential instances. Subsequently, we incrementally increased the value of t by 0.25, until we reached a point where 8 out of the 10 most influential instances generated for the current value of t were present in the set of most influential instances generated for the previous value ($t - 0.25$).

Table 2: Number of instances in the proxy subset for each dataset and model. Note: We compute the percentage of instances in the proxy subset for each model compared to the total number of samples in the dataset.

Dataset/Model	Total# samples	NN	RF	VGG13	ResNet-20
Adult-Income	48 842	9.72 %	9.43 %	NA	NA
Lending Club	10 000	10.30 %	10.72 %	NA	NA
COMPAS	7 214	9.85%	9.42 %	NA	NA
German-Credit	1 000	28.90 %	31.50 %	NA	NA
MNIST	70 000	NA	NA	26 %	23 %
Fashion-MNIST	870 000	NA	NA	32 %	24 %
CIFAR-10	60 000	NA	NA	29 %	25 %

4.7 Results and Discussion

We selected the entire test set of 200 samples of the German-Credit dataset and randomly selected 500 instances from the test set of each of the other datasets for the experiments. For each instance, we generated the top-10 most influential training instances. We only show a few examples of results in this section due to space limitations. The whole set of results can be found at [29].

(1) Influence Analysis

For the two models, NN and RF, Proxima successfully computed and ranked the top-10 influential instances for 200 test instances for the German-Credit dataset and for 500 test instances across the other datasets and models.

The table 2 provides information on the number of instances in the proxy subset for each dataset and model. The percentage of instances in the proxy subset is computed relative to the total number of samples in each dataset. For the tabular datasets (Adult-Income, Lending Club, COMPAS, and German-Credit), the proxy subset comprises a small percentage of instances for both the neural network (NN) and random forest (RF) models. The percentage ranges from approximately 9 % to 31.5 %, indicating that a subset of instances is considered influential for these models' predictions. The percentage of instances in the proxy subset varies significantly across different datasets. For example, the German-Credit dataset has a higher percentage of instances in the proxy subset compared to other datasets like Adult-Income, Lending Club, and COMPAS. This indicates that certain datasets may have a greater concentration of influential instances that play a crucial role in the model's predictions.

For image datasets (MNIST, Fashion-MNIST, and CIFAR-10), the table provides information for the VGG13 and ResNet-20 models. The percentage of instances in the proxy subset varies for these models, ranging from 23 % to 32 %. This suggests that a subset of instances is identified as influential for understanding the behavior of these image classification models. Comparing the image datasets, we observe that the VGG13 model has a higher percentage of instances in the proxy subset compared to the ResNet-20 model. This suggests that the VGG13 model may be more sensitive to specific instances or patterns in the dataset, leading to a larger proportion of influential instances. The variations suggest that different model architectures may capture distinct influential instances within the data.

The total number of samples in each dataset varies significantly, ranging from 1 000 to 870 000 instances. Interestingly, the percentage of instances in the proxy subset does not necessarily correlate with dataset size. For example, the German-Credit dataset with only 1,000 samples has a higher percentage of instances in the proxy subset compared to the larger-scale Fashion-MNIST dataset. This suggests that the German-Credit dataset may have a higher density of influential instances that strongly impact the model's predictions. This also indicates that dataset size alone does not determine the presence of influential instances; rather, it is the characteristics and patterns within the data that influence the proxy subset formation.

(2) Comparison

Table 3 displays the accuracy of different Influence Analysis approaches across various datasets. The time it takes to generate the top-10 most influential instances for different datasets is shown in the table 4.

Table 3: Accuracy of different Influence Analysis approaches.

Dataset/Approach	Scaling Up	FastIF	Proxima
Adult-Income	84 %	91 %	98 %
Lending Club	88 %	93 %	99 %
COMPAS	86 %	90 %	95 %
German-Credit	81 %	97 %	93 %
MNIST	78 %	86 %	93 %
Fashion-MNIST	83 %	92 %	94 %
CIFAR-10	91 %	89 %	87 %

Table 4: Average time (in seconds) taken to compute influential instances.

Dataset/Approach	Scaling Up	FastIF	Proxima
Adult-Income	9.2	6.4	4.1
Lending Club	6.0	6.7	1.3
COMPAS	4.3	5.6	0.9
German-Credit	3.4	4.9	0.25
MNIST	9.2	10.4	4.1
Fashion-MNIST	7.3	8.6	0.9
CIFAR-10	6.4	5.7	1.3

Proxima consistently performs well across various datasets. It achieves high accuracy rates of 98 % on Adult-Income, 99 % on Lending Club, and 95 % on COMPAS. Proxima also demonstrates strong performance on image datasets, with 93 % accuracy on MNIST, 94 % on Fashion-MNIST, and 87 % on CIFAR-10. However, the CIFAR-10 dataset shows lower accuracy compared to other datasets for all approaches. The Scaling Up approach achieves the highest accuracy (91 %) on CIFAR-10, while both FastIF and Proxima attain 89 % accuracy. It is worth noting that FastIF stands out on the German-Credit dataset, attaining the highest accuracy of 97 %, followed closely by Proxima with 93 % accuracy and Scaling Up with 81 % accuracy.

We observe that CIFAR-10 stands out as a more challenging dataset for influence analysis. All approaches achieve relatively lower accuracy on this dataset compared to others. Further investigation may be required to understand the reasons behind this lower accuracy. However, Proxima, FastIF, and Scaling Up show similar accuracy levels on MNIST and Fashion-MNIST datasets. This consistency indicates that these approaches are robust and reliable when analyzing the influence of instances in image datasets.

Additionally, Proxima consistently demonstrates the lowest computation times across various datasets. It outperforms both Scaling Up and FastIF, indicating its ability to efficiently compute influential instances. While FastIF generally performs well, there are instances where its computation time is higher than other approaches. Notably, in the Lending Club dataset, FastIF's computation time is higher compared to Scaling Up and Proxima. This suggests that FastIF may be more sensitive to certain dataset characteristics. The complexity of the dataset also appears to have some impact on computation time. In the case of image datasets (MNIST, Fashion-MNIST, CIFAR-10), FastIF and Proxima show slightly higher computation times compared to other datasets. This suggests that the presence of more features or greater complexity in image datasets may require additional computational resources.

Across all approaches, the German-Credit dataset with the lowest number of instances consistently demonstrates the lowest computation times. This indicates that the dataset's characteristics allow for faster computation of influential instances.

In conclusion, Proxima is a more efficient solution for generating influential instances, as it has higher accuracy and takes significantly less time than the other two approaches in all datasets.

5 RELATED WORK

This section discusses the existing work in influence analysis. Existing work [11, 17, 22, 25, 27, 31, 33] have been reported in Feature-level influence analysis that identifies the importance or influence of individual features on an ML model's predictions. It is often used to gain insights into how a model is making its decisions and to identify which features are most important for a specific prediction. The rest of the section discusses existing work on instance-level influence analysis that is closely related to our work.

5.1 Retraining Based Approaches

Retraining-based approaches involve retraining the model in order to understand the impact of individual instances on the model's predictions.

These approaches involve removing one data point from the dataset, refitting the model, and computing the difference in predictions for the test instance to the original model's predictions. The notion of Leave-One-Out (LOO)-based approach for influence analysis was first proposed by Cook et al. in 1982 [7]. Jia et al. proposed kNN Leave-One-Out (kNN LOO), which is a variant of LOO specifically designed for k-nearest neighbors (kNN) models [18]. It calculates the LOO difference in the model's predictions for each instance in the training dataset, and instances with higher LOO differences are considered to be more influential. Sharchilev et al. proposed LeafRefit, which is designed to be more efficient than traditional LOO approaches by refitting the decision tree ensemble only on the instances that are located in the same leaf node as the removed instance [32]. Wojnowicz et al. proposed influence sketching, which first randomly samples the data, and then uses a sketching technique to summarize the sampled data. The approach then applies linear regression to the sampled data and uses the regression coefficients to estimate the influence of each sample in the original dataset [37].

These approaches have the advantages of being highly accurate, simple to implement, and applicable to any type of model. Their main disadvantage is that they are computationally expensive and unsuitable for data with a large number of instances because the model must be retrained for each iteration of the LOO procedure, which can be time-consuming and computationally intensive. Their work is similar to Proxima in terms of the aim to quantify the impact of individual data points on a model's predictions. Proxima addresses the computational complexity issue inherent in LOO by implementing retraining of a proxy model; additionally, the new proxy model is significantly less complex and is based on a smaller dataset than the original model.

5.2 Gradient Based Approaches

Koh et al. proposed influence function (IF), which approximates the leave-one-out approach for influence analysis [19]. The approach estimates the change in the model's predictions by approximating the change in the model's parameters caused by removing an instance from the training data. These approximation values are used to rank the instances in the training data. Barshan et al. proposed an approach called the relative influence function estimator (RelatIF) to provide more semantically meaningful influence ranking [4]. Hammoudeh et al. propose renormalized influence, which replaces all gradient vectors with the corresponding unit vector in order to reduce the impact of outliers and improve the interpretability of the influence scores [14].

Sharchilev et al. proposed LeafInfluence, which is a specialized variation of IF that is specifically designed for gradient-boosted decision tree ensembles and is faster when applied to decision trees [32]. Guo et al. proposed FASTIF, which aims at speeding up IF by using a kNN approach to a sample training dataset and applying IF on the sampled data, rather than computing the IF over the entire training set [13]. Schioppa et al. proposed to speed up the IF by using Arnoldi iteration [2] to find the dominant eigenvalues of the Hessian of the loss function, which allows for faster inversion and caching of the diagonalized Hessian [30].

The advantage of using IF-based approaches is that these approaches are faster than LOO, especially for large datasets, because they only require the calculation of gradients rather than the model being re-trained for each observation. Additionally, these approaches can be more scalable to large datasets because they do not require repeated model training, which can be computationally expensive for large datasets.

However, the disadvantage of using the IF is that it is still computationally expensive, as it requires computing the second derivative of the model's output with respect to the parameters of the model. A complex model with many parameters can highly impact the computational complexity. Furthermore, in order to approximate the model's prediction, the IF should also up weight each instance in the training set, making computational speed sensitive to dataset size.

Their approach is similar to ours in terms of increasing the efficiency of influence analysis. However, Proxima speed-up the influence analysis in the following way. First, Proxima uses a proxy model, whereas other sampling-based approaches, such as FastIF, do not build a proxy model. Second, our sample is based on the loss-to-distance ratio, whereas FastIF is primarily based on distance. Additionally, Proxima is complementary to all the existing work, i.e., they could be used together. Our work is not dependent on any particular influence analysis techniques and can be used to speed up all the existing techniques of influence analysis.

6 CONCLUSION AND FUTURE WORK

In this paper, we present a proxy model-based approach for performing influence analysis. Given an ML model, M, trained on a dataset D, and a test instance X, Proxima identifies and ranks the top-10 most influential instances in D according to their influence on the model's prediction for X. The novelty of Proxima is that we identify a proxy subset of the training set based on a loss-over-distance ratio and build a proxy model that can be used in place of the original dataset and model to perform influence analysis. Proxima is independent of any specific influence analysis technique and can be used to speed up various types of influence analysis techniques. Overall, the evaluation shows that Proxima outperforms two state-of-the-art approaches in identifying the top-10 most influential instances.

Proxima's advancements in influence analysis pave the way for future research and development. Key areas to explore include scalability for large-scale datasets and complex models, improving the robustness of Proxima across different architectures and domains, enhancing result interpretability, handling online and dynamic scenarios, and integrating with other influence analysis techniques. By addressing these areas, we can further advance the field of influence analysis and empower users to gain a deeper understanding and control over machine learning models.

ACKNOWLEDGMENTS

This work is supported by research grant (70NANB21H092) from the Information Technology Laboratory of the National Institute of Standards and Technology (NIST).

Disclaimer: Certain software products are identified in this document. Such identification does not imply recommendation by the NIST, nor does it imply that the products identified are necessarily the best available for the purpose.

REFERENCES

[1] J. Angwin, J. Larson, S. Mattu, and L. Kirchner. 2016. Machine Bias: Risk Assessments in Criminal Sentencing. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing
[2] Walter Edwin Arnoldi. 1951. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of applied mathematics* 9, 1 (1951), 17–29.
[3] Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B Grosse. 2022. If Influence Functions are the Answer, Then What is the Question? *Advances in Neural Information Processing Systems* 35 (2022), 17953–17967.
[4] Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. 2020. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1899–1909.
[5] Samyadeep Basu, Phil Pope, and Soheil Feizi. 2020. Influence Functions in Deep Learning Are Fragile. In *International Conference on Learning Representations*.
[6] Lending Club. 2007-2011. Lending Club Loan Data. https://www.lendingclub.com/info/download-data.action
[7] R Dennis Cook and Sanford Weisberg. 1982. *Residuals and influence in regression*. New York: Chapman and Hall.
[8] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine* 29, 6 (2012), 141–142.
[9] Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems* 33 (2020), 2881–2891.
[10] Python Software Foundation. 2021. time – Time access and conversions. https://docs.python.org/3/library/time.html
[11] Amirata Ghorbani and James Zou. 2019. Data Shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*. PMLR, 2242–2251.
[12] Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. 2004. Neighbourhood components analysis. *Advances in neural information processing systems* 17 (2004).
[13] Han Guo, Nazneen Fatema Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. 2020. Fastif: Scalable influence functions for efficient model interpretation and debugging. *arXiv preprint arXiv:2012.15781* (2020).
[14] Zayd Hammoudeh and Daniel Lowd. 2022. Identifying a Training-Set Attack's Target Using Renormalized Influence Estimation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1367–1381.
[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[16] R. Herberich and A. K'stein. 1993. *Statlog (German Credit Data) dataset.* http://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)
[17] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas Spanos, and Dawn Song. 2019. Efficient task-specific data valuation for nearest neighbor algorithms. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1610–1623.
[18] Ruoxi Jia, Fan Wu, Xuehui Sun, Jiacen Xu, David Dao, Bhavya Kailkhura, Ce Zhang, Bo Li, and Dawn Song. 2021. Scalability vs. Utility: Do We Have to Sacrifice One for the Other in Data Importance Quantification?. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8239–8247.
[19] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*. PMLR, 1885–1894.
[20] Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. 2019. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems* 32 (2019).
[21] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
[22] Yongchan Kwon and James Zou. 2021. Beta Shapley: a unified and noise-reduced data valuation framework for machine learning. *arXiv preprint arXiv:2110.14049* (2021).
[23] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*. Springer, 9–48.
[24] M. Lichman. 2013. *UCI Machine Learning Repository adult data set.* https://archive.ics.uci.edu/ml/datasets/adult
[25] Jinkun Lin, Anqi Zhang, Mathias Lécuyer, Jinyang Li, Aurojit Panda, and Siddhartha Sen. 2022. Measuring the Effect of Training Data on Deep Learning Predictions via Randomized Experiments. In *International Conference on Machine*

Learning. PMLR, 13468–13504.

[26] Jiwen Lu, Junlin Hu, and Jie Zhou. 2017. Deep metric learning for visual understanding: An overview of recent advances. *IEEE Signal Processing Magazine* 34, 6 (2017), 76–84.

[27] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[28] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*. 607–617.

[29] Proxima. 2023. Proxima: A Proxy Model-Based Approach to Influence Analysis. https://github.com/DeltaExplainer/Proxima

[30] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. 2022. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8179–8186.

[31] Lloyd S Shapley. 1997. A value for n-person games. *Classics in game theory* 69 (1997).

[32] Boris Sharchilev, Yury Ustinovskiy, Pavel Serdyukov, and Maarten Rijke. 2018. Finding influential training samples for gradient boosted decision trees. In *International Conference on Machine Learning*. PMLR, 4577–4585.

[33] Sunny Shree, Jaganmohan Chandrasekaran, Yu Lei, Raghu N Kacker, and D Richard Kuhn. 2022. DeltaExplainer: A Software Debugging Approach to Generating Counterfactual Explanations. In *2022 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, 103–110.

[34] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[35] Juan Luis Suárez, Salvador García, and Francisco Herrera. 2021. A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing* 425 (2021), 300–322.

[36] Xiaofei Sun, Diyi Yang, Xiaoya Li, Tianwei Zhang, Yuxian Meng, Han Qiu, Guoyin Wang, Eduard Hovy, and Jiwei Li. 2021. Interpreting deep learning models in natural language processing: A review. *arXiv preprint arXiv:2110.10470* (2021).

[37] Mike Wojnowicz, Ben Cruz, Xuan Zhao, Brian Wallace, Matt Wolff, Jay Luan, and Caleb Crable. 2016. "Influence sketching": Finding influential samples in large-scale regressions. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 3601–3612.

[38] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[39] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. 2014. Deep metric learning for person re-identification. In *2014 22nd international conference on pattern recognition*. IEEE, 34–39.