Boosting for Fairness-Aware Classification

Ivan Safonov [1] **Nikita Morozov** [1] **Ekaterina Fadeeva** [1] **Artur Goldman** [1]

Abstract

The widespread adoption of machine learning-based decision making in socially impactful domains has raised concerns about potential discrimination based on sensitive attributes such as gender or race. Traditional fairness approaches have focused on improving fairness while maintaining classification accuracy, but accuracy is biased towards majority classes and is not a reliable performance indicator for imbalanced datasets. Recently, the AdaFair method was proposed to tackle the imbalance problem by reweighting training examples. In this work, we compare it with traditional methods and also introduce Attentive Gradient Boosting, a novel method based on application of AdaFair idea to gradient boosting. Our experimental results show that both AdaFair and Attentive Gradient Boosting outperform traditional fairness methods and achieve higher level of fairness.

Github repo: https://github.com/isaf27/fairness-aware-classification
Presentation file: https://github.com/isaf27/fairness-aware-classification/blob/main/presentation.pdf

1. Introduction

Fairness in machine learning has become a critically important topic in recent years, as machine learning models are increasingly being used to make decisions that have a significant impact on people's lives. From determining creditworthiness to predicting criminal behavior, these models are being used to make decisions that can have far-reaching consequences for individuals and entire communities. As a result, there is growing concern that these models may be perpetuating or even amplifying existing biases and discrimination in society. In Amazon's Prime case (Ingold & Soper), for example, in which the algorithm's task was to decide which areas of a city are eligible to advanced services, areas mostly inhabited by black people were ignored (racial-bias), even though the algorithm did not consider race as a feature.

A number of approaches have been proposed over the years to tackle the problem of fairness in machine learning. In this report, we compare AdaFair (Iosifidis & Ntoutsi, 2019) approach to a number of classical boosting approaches (Freund & Schapire, 1997; Chawla et al., 2003; Seiffert et al., 2009) in terms of predictive performance and fairness. In addition, we propose a novel method based on gradient boosting that we name Attentive Gradient Boosting. We show that it outperforms AdaFair in terms of both accuracy and fairness.

The main contributions of this report are as follows: 1) overview of the problem of fariness in ML and its mathematical formulation 2) novel approach to the problem of fairness: Attentive Gradient Boosting 3) experimental comparison of the proposed approach to AdaFair and a number of classical boosting methods on several classification datasets.

2. Preliminaries and definitions

In our work we consider that the classification task is binary, meaning $y \in \{-1, +1\}$ and that there exists single sensitive binary attribute S. That is, $S \in \{s, \overline{s}\}$ with $s, \overline{s}$ denoting protected and non-protected class. We use the notation $s_+(s_-), \overline{s}_+(\overline{s}_-)$ to denote the protected and nonprotected group for the positive (negative, respectively) class.

2.1. Fairness

There exist multiple fairness notions in classification (Romei & Ruggieri, 2014; Verma & Rubin, 2018). One of widely used metric is called *equalized odds*. This metric accounts for the difference of true classifed instances between protected and non-protected group in all classes. In case of binary classification it is defined as

$$Eq.Odds = |\delta FPR| + |\delta FNR| \tag{1}$$

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Ivan Safonov <Ivan.Safonov@skoltech.ru>, Nikita Morozov <Nikita.Morozov@skoltech.ru>, Ekaterina Fadeeva <Ekaterina.Fadeeva@skoltech.ru>, Artur Goldman <Artur.Goldman@skoltech.ru>.

Final Projects of the Machine Learning 2023 Course, Skoltech, Moscow, Russian Federation, 2023.

where

$$\delta FPR = P(y \neq \hat{y}|\overline{s}_-) - P(y \neq \hat{y}|s_-), \tag{2}$$

$$\delta FNR = P(y \neq \hat{y}|\overline{s}_+) - P(y \neq \hat{y}|s_+), \tag{3}$$

$\hat{y}$ - predicted label. The lower $Eq.Odds$ metric, the better. However, we also need to keep track of extreme cases of misclassifying a lot of elements in similar proportions in both protected and non-protected classes. For this reason we should also make sure that, for example, TNRs and TPRs are high for both protected and unprotected groups.

3. Related work

Before suggesting our method, we first get a sense of the problem by reimplementing and experimenting on already existing approaches. We will briefly describe main idea behind each of them.

3.1. SMOTE

It can be noted, that fairness problems arise when class-imbalance takes place. It can be seen in datasets we use, see table 1. For this reason the most straightforward idea might be to try and fix class imbalance. One would hope that fairness can be improved after such manipulation.

SMOTE (Chawla et al., 2002) is a classical oversampling technique that adds new samples to the dataset from segments connecting nearest neighbor points from minority class. We consider using it as a baseline with linear regression and random forest as classification models used on top of SMOTE.

3.2. SMOTEBoost & RUSBoost

Another way of tackling class-imbalance problems are suggested in SMOTEBoost (Chawla et al., 2003) and RUSBoost (Seiffert et al., 2009) algorithms. They combine classical AdaBoost (Freund & Schapire, 1997) algorithm with upsampling method SMOTE and undersampling method Random UnderSampling. These models are also used as baselines.

3.3. AdaFair

In order to construct more accurate model and use $Eq.Odds$ metric (1) we need a more sophisticated procedure. Authors of AdaFair (Iosifidis & Ntoutsi, 2019) come up with such an algorithm.

In the AdaFair paper, the authors adapt AdaBoost for fairness by adjusting the reweighting process. They directly consider fairness in the model's behavior during reweighting by introducing the concept of cumulative fairness, which evaluates the model's behavior with respect to Eq.Odds up to the current boosting round. They also utilize reliability metrics in the reweighting process to account for different instance weighting depending on the model's confidence in its class.

The cumulative boosting fairness of the model $H_{1:j}(x)$ on round j is defined in terms of $\delta FNR^{1:j}$ and $\delta FPR^{1:j}$, calculated for the current ensemble output of j models. Vanilla AdaBoost already boosts misclassified instances for the next round, so reweightning process in AdaFair aims to achieve fairness across the groups and the classes. The fairness-related cost u is estimated at each round using the following formula:

$$u_i = \begin{cases} |\delta FNR^{1:j}|, \text{ if } y_i \neq h_j(x_i), \delta FNR^{1:j} > \epsilon, x_i \in s_+ \\ |\delta FNR^{1:j}|, \text{ if } y_i \neq h_j(x_i), \delta FNR^{1:j} < -\epsilon, x_i \in \overline{s}_+ \\ |\delta FPR^{1:j}|, \text{ if } y_i \neq h_j(x_i), \delta FPR^{1:j} > \epsilon, x_i \in s_- \\ |\delta FPR^{1:j}|, \text{ if } y_i \neq h_j(x_i), \delta FPR^{1:j} < -\epsilon, x_i \in \overline{s}_- \\ 0, \text{ otherwise} \end{cases} \tag{4}$$

With this defined, AdaFair makes only modification of the reweightning process in AdaBoost:

$$w_i \leftarrow \frac{1}{Z_j} w_i \cdot e^{\alpha_j \cdot \hat{h}_j(x_i) \cdot \mathbb{I}(y_i \neq h_j(x_i))} \cdot (1 + u_i) \tag{5}$$

Where $\hat{h}_j(x)$ denotes the confidence of the j -th estimator on sample x. Note that for $\hat{h} = 1$ and $u_i = 0$ algorithm is the same as AdaBoost. Parameter ϵ reflects the tolerance to fairness and is typically set to zero or to a very small value. While vanilla AdaBoost reweightning process pays more attention to the misclassified samples, the described reweightning procedure in AdaFair also counts the fairness of the model.

Authors of (Iosifidis & Ntoutsi, 2019) also introduces a procedure to automatically adjust optimal number of weak learners j by using BER metrics. In our experiments we decided to optimize the number of learners manually.

4. Attentive Gradient Boosting

We came up with our novel method for Fairness Aware Classification based on boosting.

Let's consider GBDT algorithm. On each iteration we add the new tree into ensemble, such that the loss function after addition is minimized. On each iteration we will set weights for the objects in the training dataset in such way, that the resulting ensemble will be more fair. Weights for objects are supported in the GBDT algorithm, each term in the total loss function is multiplied to weight of the corresponding object in the dataset.

We used the following algorithm to calculate weights:

- Let's consider $\delta FPR = FPR_{prot} - FPR_{non.prot}$, $\delta FNR = FNR_{prot} - FNR_{non.prot}$. We want to make $|\delta FPR|$ and $|\delta FNR|$ smaller.
- Suppose that $\delta FPR > \varepsilon$. In this case we want to make FPR_{prot} smaller, $FPR_{non.prot}$ bigger. Metric FPR_{prot} is equal to the ratio of $(protected, negative)$ objects in dataset classified incorrectly, $FPR_{non.prot}$ is equal to the ratio of $(non.protected, negative)$ objects in dataset classified incorrectly.
- Let's set weights for $(protected, negative)$ objects equal to 1, weights for $(non.protected, negative)$ objects equal to C (some hyperparameter $C < 1$). In this case on the new iteration algorithm will be more attentive to $(protected, negative)$ objects and less attentive to $(non.protected, negative)$ objects. So the value FPR_{prot} will be increased, value $FPR_{non.prot}$ will be decreased. As a result we will make them closer and $|\delta FPR|$ smaller.
- If $\delta FPR < -\varepsilon$ we will set weights for $(protected, negative)$ objects equal to C, weights for $(non.protected, negative)$ objects equal to 1.
- If $|\delta FPR| \leq \varepsilon$ we will set weights for all $negative$ objects equal to 1.
- For $|\delta FNR|$ we will do the same algorithm to set weights for all $positive$ objects.

As a result on each iteration of GBDT weights of objects will help model to be more attentive to categories that are classified unfairly by model.

We set parameters $C = \frac{1}{2}$ and $\varepsilon = 10^{-2}$.

This method can be applied to any boosting algorithm supporting weights of objects. GBDT algorithms like XGBoost or CatBoost based on 2^{nd} order optimization methods are state-of-the art for many problems and our method can be applied for them. As a result all advantages of them will remain, but the model will be fair.

5. Experiments and Results

Experiments can be found in notebook in the project repository.

5.1. Datasets

We evaluate our approach on four real-world datasets whose characteristics are summarized in Table 1. As we can see, they vary w.r.t. cardinality, dimensionality and class imbalance and therefore provide an interesting benchmark for evaluation.

	Adult Census	Bank	Compass	KDD Census
#Instances	45,175	40,004	5,278	299,285
#Attributes	14	16	9	41
#Feat. (after prep.)	107	51	11	409
Sen.Attr.	Gender	Marit. Status	Gender	Gender
Class ratio (+:−)	1:3.03	1:7.57	1:1.12	1:15.11
Positive class	*>50K*	*subscription*	*recidivism*	*>50K*

Table 1. An overview of the datasets.

Adult census income (Bache & Lichman, 2013) dataset contains demographic data from the U.S. and the task is to predict whether the annual income of a person will exceed 50K dollars. The sensitive attribute is $S = Gender$ with $s = female$ being the protected group; the positive class is people receiving more than 50K. We remove duplicate instances and instances containing missing values. The positive to negative class ratio is 1:3 (exact ratio 24%:76%).

Bank dataset (Bache & Lichman, 2013) is related to direct marketing campaigns of a Portuguese banking institution. The task is to determine if a person subscribes to the product (bank term deposit). As positive class we consider people who subscribed to a term deposit. We consider as $S = maritial\ status$ with $s = married$ being the protected group. The dataset suffers from severe class imbalance, with a positive to negative ratio of 1:8 (exact ratio 11%:89%).

Compass dataset (Larson et al., 2016) contains information on prisoners in Broward County such as the number of juvenile felonies. The task is to determine if a person will be re-arrested within two years *(recidivism)*. We consider *recidivism* as the positive class and $S = Gender$ with $s = female$ as the protected group. For this dataset, we followed the pre-processing steps of (Zafar et al., 2017). The dataset is almost balanced, the exact positive to negative ratio is 46%:54%.

KDD census income (Bache & Lichman, 2013) has the same prediction task as the **adult census** dataset. However in KDD census "the class labels were drawn from the total person income field rather than the adjusted gross income" (Bache & Lichman, 2013). We consider $S = Gender$ with $s = female$ as the protected group, and as positive class people receiving more than 50K annually. This is the most skewed dataset in our benchmark with a positive to negative ratio of 1:15 (exact ratio 6%:94%).

5.2. Reported results

For each dataset and method we report metrics on test. Each method uses only train dataset for training.

We calculate accuracy, balanced accuracy, equalized odds,

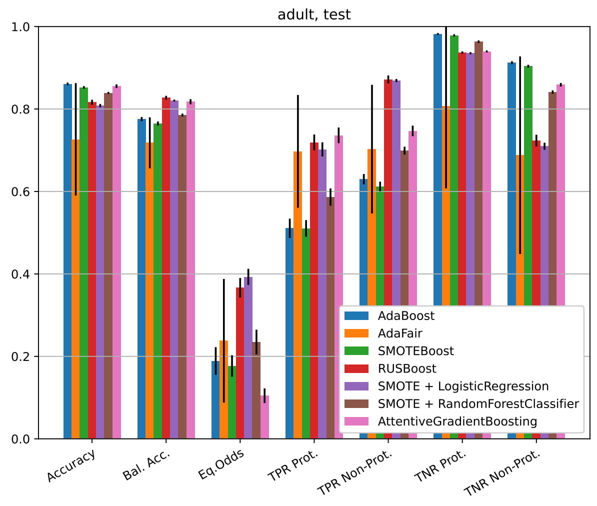

Figure 1. **adult** dataset, metrics on **test**

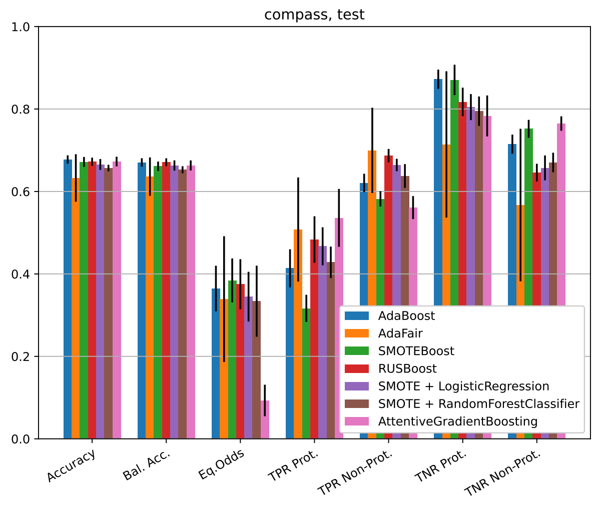

Figure 3. **compass** dataset, metrics on **test**

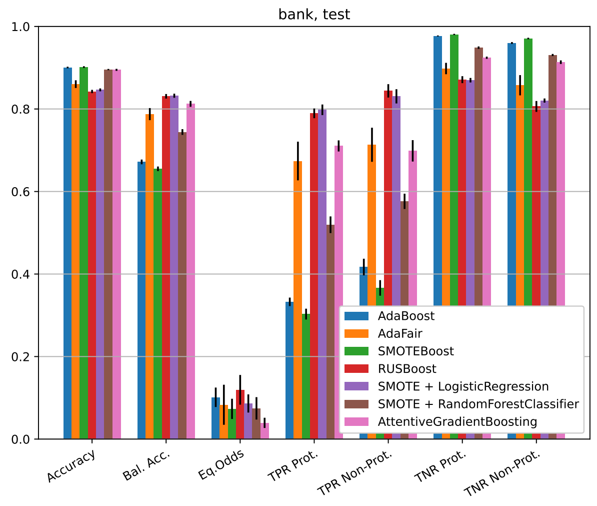

Figure 2. **bank** dataset, metrics on **test**

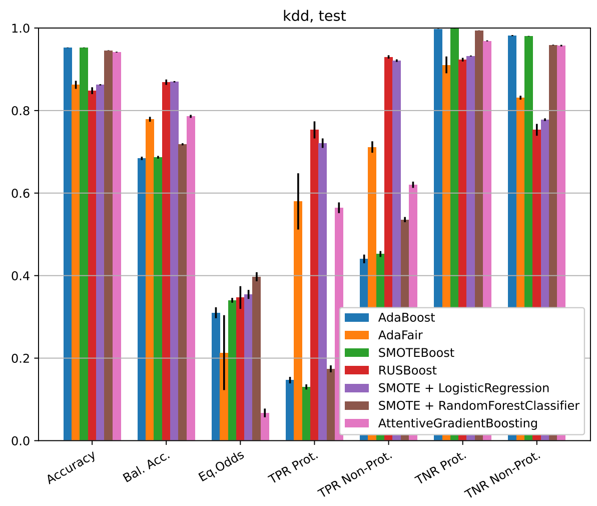

Figure 4. **kdd** dataset, metrics on **test**

TNR (for protected objects), TNR (for non protected objects), TPR (for protected objects), TPR (for non protected objects).

For all values except equalized odds bigger values are better, for equalized odds smaller values are better. The closer values of TNR or TPR for protected and non protected groups the more fair model is.

Test size is 25% of all data. We perform 10 runs with different *seed* values (for splitting data) and report mean and standart deviation for each metric.

For preprocessing we used one hot encoding of categorial features, numeric features were not changed.

5.3. Algorithms parameters

In methods AdaBoost and AdaFair we used Decision Tree from sklearn with default parameters (except maximum depth 1) as a base model. We used standard sklearn implementations of LogisticRegression, RandomForestClassifier with default parameters for our other methods.

For method AttentiveGradientBoosting we used CatBoost for training GBDT, each new model was built with one tree, new weights for objects are calculated using our algorithm.

5.4. Results description

First of all we see, that $Eq.Odds$ of our fair methods AdaFair and AttentiveGradientBoosting is smaller than for other methods, that does not take protected groups into account. For some datasets like `adult` or `compass` $Eq.Odds$ for AdaFair is not much smaller than for usual AdaBoost, but for others it is around two times smaller. AttentiveGradientBoosting have much smaller $Eq.Odds$ than other methods for all datasets. So, our fair methods provide better fairness than usual algorithms and AttentiveGradientBoosting gives much smaller values for $Eq.Odds$.

For all datasets we can compare values of TPR, TNR for protected and non-protected groups and see, that for AdaFair and AttentiveGradientBoosting these values are very close, but for other methods they can differ a lot. This is exactly what fairness means and what metric $Eq.Odds$ controls.

Let's compare how good our models are in terms of Balanced Accuracy. We see, that methods SMOTEBoost (with LogisticRegression) and RUSBoost, that are designed for imbalance classification give the best values of Balanced Accuracy. However AdaFair and AttentiveGradientBoosting give a bit smaller values of Balanced Accuracy, but much better than unfair methods, that does not take class imbalance into account.

In terms of usual Accuracy we see that AdaBoost, SMOTEBoost and AttentiveGradientBoosting give the best values. We see, that AttentiveGradientBoosting is fair, but it is the best method for Accuracy and one of the best methods for Balanced Accuracy. So using this algorithm we can make the model fair without loss of quality.

6. Conclusion

We evaluated some algorithms on four datasets and found, that methods AdaFair and AttentiveGradientBoosting provide fairness without significant loss of quality. Also our novel approach AttentiveGradientBoosting give much smaller $Eq.Odds$ than other algorithms and same accuracy and balanced accuracy as unfair methods. So we were able to find a way to make GBDT ensemble fair without loss of all advantages of it.

We think that it is very interesting result, because algorithm like AdaFair is based on AdaBoost, which is not practical algorithm nowadays. Currently GBDT is often used for tabular data and give a very good quality. We found a way to make it fair.

In the further work more different ways to improve AttentiveGradientBoosting should be carefully tested and possibly better algorithm can be found. We think that it can help to create more practical algorithm, that people will use in practice to make models fair.

References

Bache, K. and Lichman, M. Uci machine learning repository, 2013.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Chawla, N. V., Lazarevic, A., Hall, L. O., and Bowyer, K. W. Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003: 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003. Proceedings 7*, pp. 107–119. Springer, 2003.

Freund, Y. and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Ingold, D. and Soper, S. Amazon doesnt consider the race of its customers. should it?, april 2016. *URL https://www. bloomberg. com/graphics/2016-amazon-same-day*.

Iosifidis, V. and Ntoutsi, E. Adafair: Cumulative fairness adaptive boosting. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 781–790, 2019.

Larson, J., Mattu, S., Kirchner, L., and Angwin, J. How we analyzed the compas recidivism algorithm. *ProPublica (5 2016)*, 9, 2016.

Romei, A. and Ruggieri, S. A multidisciplinary survey on discrimination analysis. *The Knowledge Engineering Review*, 29(5):582–638, 2014.

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., and Napolitano, A. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(1): 185–197, 2009.

Verma, S. and Rubin, J. Fairness definitions explained. In *Proceedings of the international workshop on software fairness*, pp. 1–7, 2018.

Zafar, M. B., Valera, I., Gomez Rodriguez, M., and Gummadi, K. P. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th International Conference on World Wide Web*, pp. 1171–1180. WWW, 2017.

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Ivan Safonov (25% of work)

- Novel fairness method: Attentive Gradient Boosting
- Implementation of Attentive Gradient Boosting
- Conducting experiments

Nikita Morozov (25% of work)

- Work with the literature
- Processing and preparing datasets for experiments
- Implementation of baseline methods: SMOTE + LinearRegression / RandomForest
- Conducting experiments

Ekaterina Fadeeva (25% of work)

- Work with the literature
- Implementation of AdaFair
- Conducting experiments

Artur Goldman (25% of work)

- Work with the literature
- Implementation of SMOTEBoost and RUSBoost
- Conducting experiments

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   - [✓] Yes.
   - [ ] No.
   - [ ] Not applicable.

   General comment: If the answer is **yes**, students must explicitly clarify to which extent (e.g. which percentage of your code did you write on your own?) and which code was used.

   Students' comment: We used implementation of SMOTE algorithm and RUSBoost from imbalanced-learn; LinearRegression, RandomForest and AdaBoost from scikit-learn. AdaFair, SMOTEBoost and Attentive Gradient Boosting we implemented from scratch by ourselves. All code for data processing and experimental evaluation was also written from scratch.

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   - [✓] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   - [✓] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.

   - [✓] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   - [✓] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: Provided in the github repository

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   - [✓] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   - [✓] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   - [ ] Yes.
   - [✓] No.
   - [ ] Not applicable.

   Students' comment: Authors of AdaFair did not tune hyperparameters, so we used their hyperparameter values.

9. The exact number of evaluation runs is included.

   - [✓] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.

    - [✓] Yes.
    - [ ] No.
    - [ ] Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    - [✓] Yes.
    - [ ] No.
    - [ ] Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.

    - [✓ Yes.]
    - ☐ No.
    - ☐ Not applicable.

    Students' comment: None

13. A description of the computing infrastructure used is included in the report.

    - [✓ Yes.]
    - ☐ No.
    - ☐ Not applicable.

    Students' comment: We used Google Colab for experiments.