Trustworthy Machine Learning

Master's Programme in Computer Science

Project Nr.3: Fairness in Machine Learning

Petteri Huvio, Luca Maahs

December 9, 2025

Faculty of Science

University of Helsinki

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki, Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author

Petteri Huvio, Luca Maahs

Työn nimi — Arbetets titel — Title

Project Nr.3: Fairness in Machine Learning

Ohjaajat — Handledare — Supervisors

I don't know yet.

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Trustworthy Machine Learning	December 9, 2025	8 pages

Tiivistelmä — Referat — Abstract

ACM Computing Classification System (CCS)
General and reference → Document types → Surveys and overviews
Networks → Network algorithms → Control path algorithms → Network design and planning algorithms

Avainsanat — Nyckelord — Keywords

Trustworthy Machine Learning, Fairness, Bias, Mitigation

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Course on Trustworthy Machine Learning

Contents

1 Project Idea

The idea of this project is to explore the concept of fairness in machine learning models. For that we chose a dataset of credit loan applications and their outcomes. The goal was to train normal machine learning models on this dataset and evaluate it regarding a chosen fairness metric. After that we looked into methods of mitigating unfairness and implemented one of those methods ourselves. This was then evaluated again regarding the fairness metric and compared to the initial results. Further we analyzed the trade-off between fairness and accuracy that comes with these methods and did a grid search on some hyperparameters to find the best possible solution for our project.

2 Methods

2.1 Data

Patel (2025) provides a dataset of loan applications from the US and Canada, which we use to evaluate fairness in machine learning models. The dataset includes features such as applicant income, credit score, and loan amount, along with a binary target variable indicating whether the loan was approved.

2.2 Base Model Training

We trained two base models being a Random Forest and a Neural Network.

2.2.1 Random Forest

How we trained it and what results.

2.2.2 Neural Network

How we trained it and what results.

2.3 Equal Opportunity

As a Fairness Metric, we chose Equal Opportunity because it seemed to be the best choice in our case. It shows if all applicants have the same opportunity to recieve a loan. It is defined by calculating True Positive Rates (TPR). TPR is the fraction of positive cases which were correctly predicted out of all the ground truths. It is usually referred to as sensitivity or recall, and it represents the probability of the positive subjects to be classified correctly as such. It is given by the formula: $TPR = P(prediction = 1|actual = 1) = \frac{TP}{TP+FN}$.

To now evaluate the fairness of trained models we are usually speaking of Equal Opportunity in the case of difference of two TPR. These two TPR are from a binary classification, which in our case is called a priviledged and unpriviledged group. Both of these groups together build the whole of one feature in the data. We are talking about two different versions of difference, the absolute and the relative difference of TPR. We refer to $\Delta = TPR_{priviledged} - TPR_{unpriviledged}$ for the absolute difference and to $\delta = (1 - \frac{TPR_{unpriviledged}}{TPR_{priviledged}}) * 100$ for the relative difference in percent. In both cases, a lower value means a fairer model, with 0 being perfectly fair.

2.3.1 Chosen Subsets

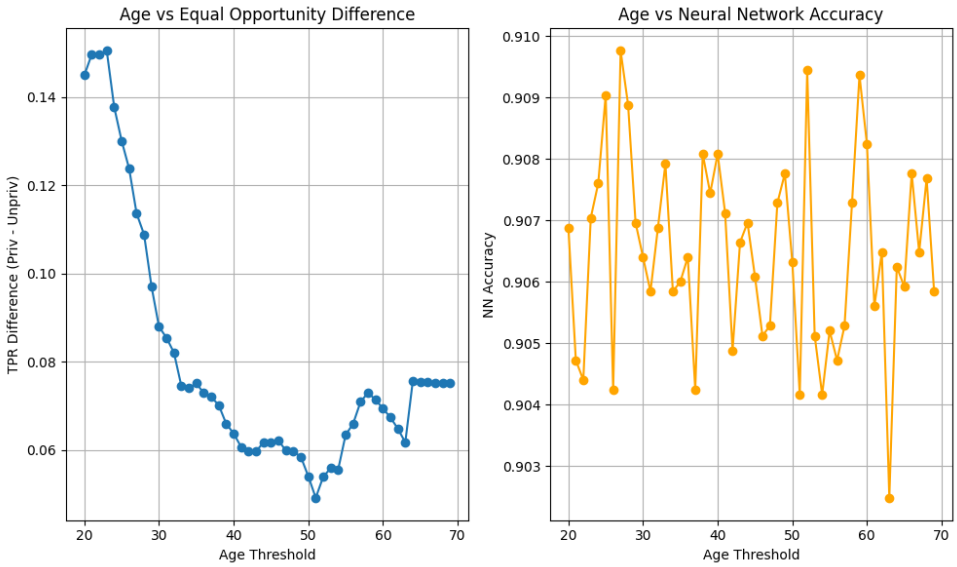

Figure 2.1: Fairness Results

2.4 Unfairness Mitigation

For mitigating the unfairness in our models, we could in principle choose between Preprocessing, Learning with Fairness Constraints, and Postprocessing methods. For that we have trained the random forest with scikit-learn, we decided to implement a Learning with Fairness Constraints method ourselves for the Neural Network. This means we modified the loss function of the Neural Network to include a penalty for unfairness according to the Equal Opportunity metric. The idea behind this was to directly optimize the model

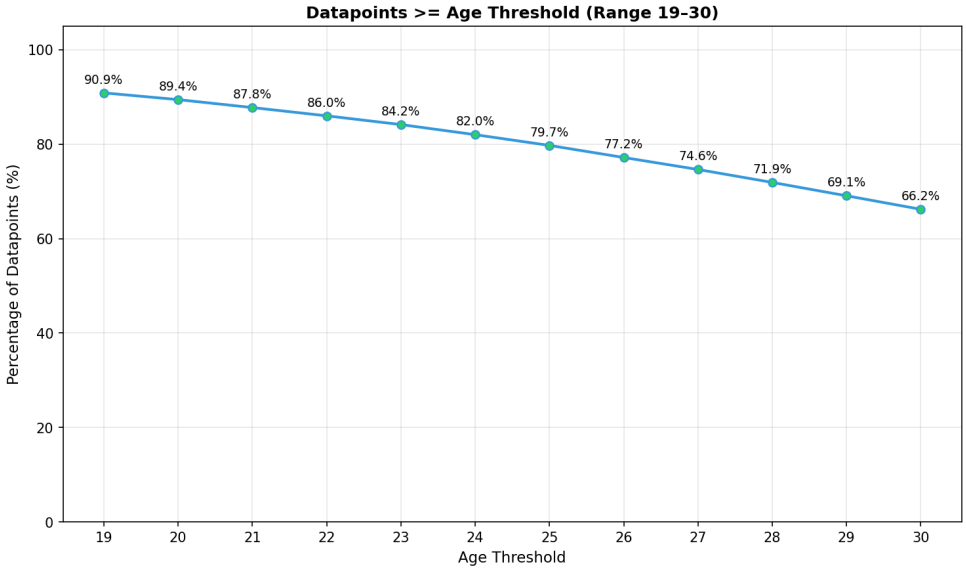

Figure 2.2: Age Threshold Distribution

for both accuracy and fairness during training and search for a local minimum of both objectives.

2.4.1 Equal Opportunity Loss Function

To implement our own loss function, which can be seen in Figure 2.3, we took the Binary Cross Entropy Loss as the actual loss, lable prediction probabilities, current sensitive attribute values as well as the labels.

We at first we created a mask for positive labels, with what we filtered the predicted probabilities and sensitive attribute values to only include positive samples. Then we created two masks for the priviledged and unpriviledged group within these positive samples. Using these we then calculated the TPR for both the groups. The penalty for the unfairness was then $|\Delta|$ between these two TPR, returning the actual loss plus the penalty multiplied with a λ coefficient to weight the importance of fairness. This loss was then used during training of the Neural Network to optimize the model.

```
 def EO_loss_fn(actual_loss, y_pred_probs, sensitive_attr, labels,
lambda_coef=0.1, epsilon=1e-7):
    pos_mask = (labels == 1).squeeze()

    y_pred_pos = y_pred_probs[pos_mask]
    sens_attr_pos = sensitive_attr[pos_mask]

    priv_mask = (sens_attr_pos == 1)
    tpr_priv = (y_pred_pos[priv_mask].sum()) / (priv_mask.sum() + epsilon)
    unpriv_mask = (sens_attr_pos == 0)
    tpr_unpriv = (y_pred_pos[unpriv_mask].sum()) / (unpriv_mask.sum() +
epsilon)
    eo_penalty = torch.abs(tpr_priv - tpr_unpriv)

    return actual_loss + (eo_penalty * lambda_coef)
```

Figure 2.3: Equal Opportunity Loss Function Implementation

3 Results

After having implemented our own Equal Opportunity loss function as described in Chapter 2, we started *Learning with Fairness Constraints*. We experimented with different hyperparameters and then decided that our λ coefficient and the number of training epochs e were the most interesting to analyze. For this we then set up two different Grid Searches, one for $\lambda \in \{0 \dots 1\}$ and one for $e \in \{10, 20, 30\}$.

The results from these these two experiments can be seen in Figure 3.1. As we can see, the with introducing a higher λ, and therefore fairness, we loose accuracy as expected. However, the fairness is increasing exponentially faster than the accuracy is decreasing, until a λ of about 0.5. After which the accuracy starts to drop faster than the fairness increases. This means that a λ of about 0.5 is a good trade-off between fairness and accuracy.

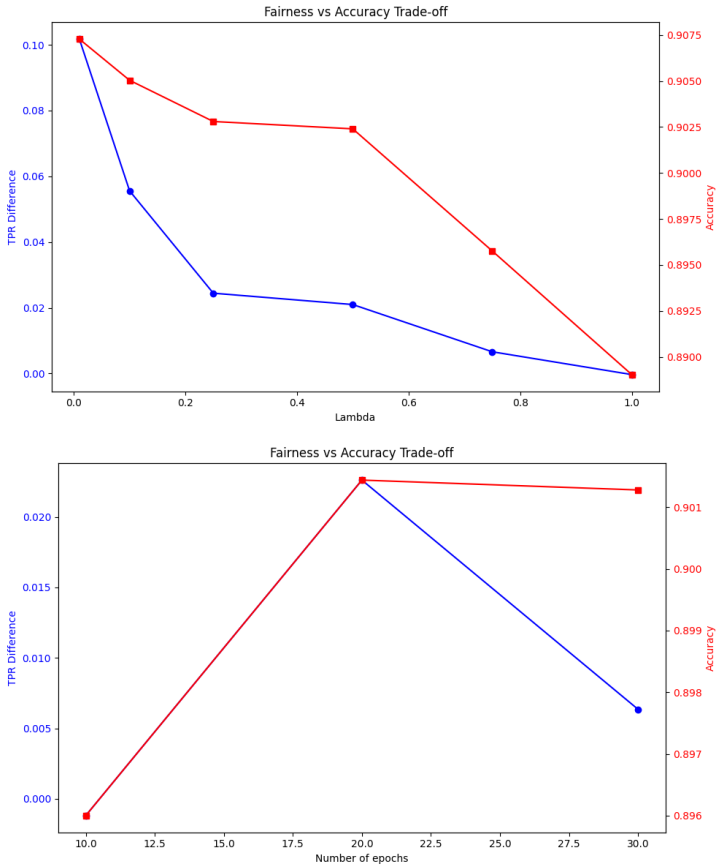

Figure 3.1: Fairness-Accuracy Tradeoff from Grid Searches

After running the experiment then also with the joined $\lambda = 0.5$ and $e = 30$ we had all our results to be summarized in Figure 3.2. Here we can see that the accuracy only drops slightly in each step of more fairness, while the relative fairness δ improves drastically from no fairness with $\delta = WRONG\%$ to 0.3% with $\lambda = 0.5$ and $e = 30$.

	Unfair	Fair	Best $\lambda = 0.5$	Increased Epochs
Accuracy	90.69%	90.42%	90.24%	90.13%
Fairness (δ)	-	11.2%	1.2%	0.3%

Figure 3.2: Neural Network Results Summary

Finally after concluding our experiments were a success, we plotted the ROC curves by sensitive attribute groups for the Fair Neural Network in Figure 3.3. That has been done to be sure not to only rely on the fairness metric and accuracy, but also see that the smaller class is not being ignored by the model to accieve this success. As we can see the two ROC curves are quite close to each other, which indicates that the model is treating both groups fairly equally. This backes up the conclusion of our model now being much fairer than in the beginning, while only loosing a small amount of accuracy.

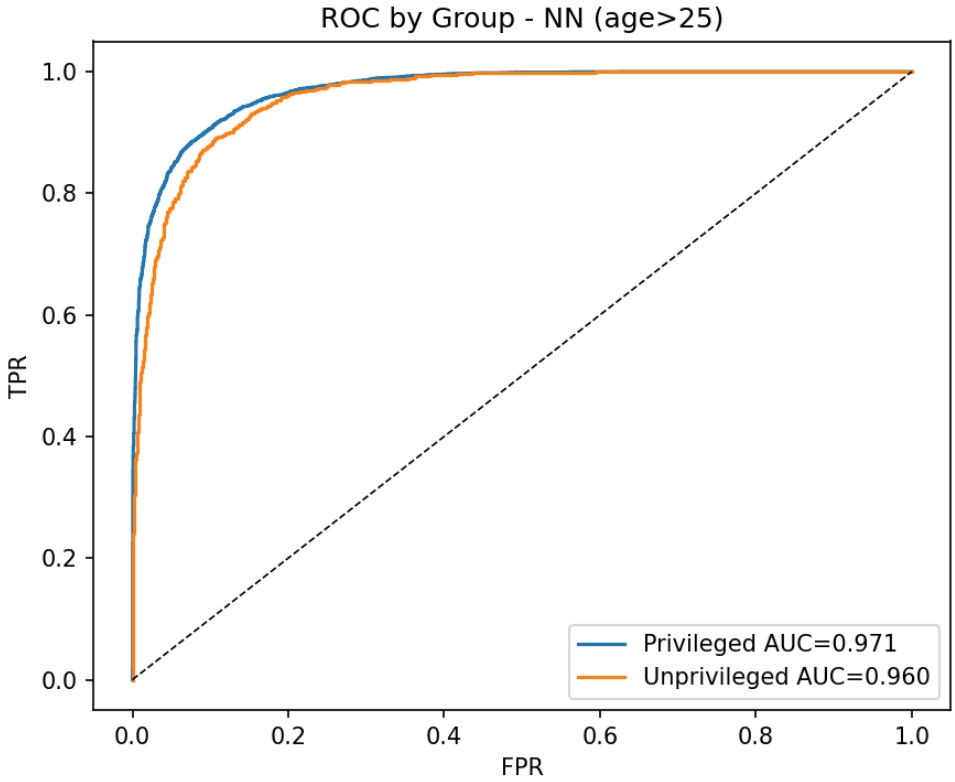

Figure 3.3: ROC by Group for Neural Network

Use of AI tools

We have been using AI tools to assist in brainstorming ideas in the initial phases of the project with Open AI's Chat GPT5 and Google's Gemini. Further we have used AI in order to help us with refining python code snippets in the form of Github's Copilot with Claude Sonnet 4.5.

Bibliography

Patel, P. (2025). *Realistic Loan Approval Dataset (US and Canada)*. https://www.kaggle.com/datasets/parthpatel2130/realistic-loan-approval-dataset-us-and-canada. Accessed: 2025-12-09.