Trustworthy Machine Learning

Master's Programme in Computer Science

Project Nr.3: Fairness in Machine Learning

Petteri Huvio, Luca Maahs

December 10, 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/

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 is synthetic but is built on real banking data criteria. The dataset incorporates realistic correlations and business logic that reflect actual lending decision processes. The target variable is loan approval status, making it suitable for binary classification tasks. There are 18 predictive features in total, including both numerical and categorical variables. Categorical features were one-hot encoded for model training. For numerical features, we applied standard scaling to normalize the data. There were no missing values in the dataset, so no imputation was necessary. The dataset includes features such as applicant income, credit score, and loan amount. Gender or race was not included in the dataset, so we selected age as a sensitive attribute for our fairness analysis. Age was a numerical feature. We wanted to keep things simple and therefore created a binary sensitive attribute by creating a threshold. Applicants older than the threshold were assigned to the privileged group, while applicants younger than the threshold were assigned to the unprivileged group.

2.2 Base Model Training

We then trained two base models using selected threshold. We decided to use a Random Forest classifier and a Neural Network. For Random Forest we used the implementation from scikit-learn and for the Neural Network we used PyTorch.

We were not sure which threshold to choose for age, so we decided to run a grid search over possible thresholds. For this, we trained a Neural Network for each threshold and evaluated the fairness of the model using Equal Opportunity metric, which will be explained in the next section. The results of this grid search can be seen in Figure 2.1. In the figure, on the left you can see the True Positive Rates difference for unpriviledged group over all thresholds and on the right accuracies of Neural Network with the same threshold. It is visible that there is a strong bias in the data for younger applicants. Now we had

several options for the threshold. We wanted to choose a threshold that shows a high unfairness while still having a decent amount of samples in both groups. So we checked the distribution of samples over thresholds from 19 to 30. This distribution can be seen in Figure 2.2. To keep at least 20 % of samples in the unpriviledged group, the highest threshold we could choose was 25 years. Therefore we chose 25 years as the threshold for our binary sensitive attribute.

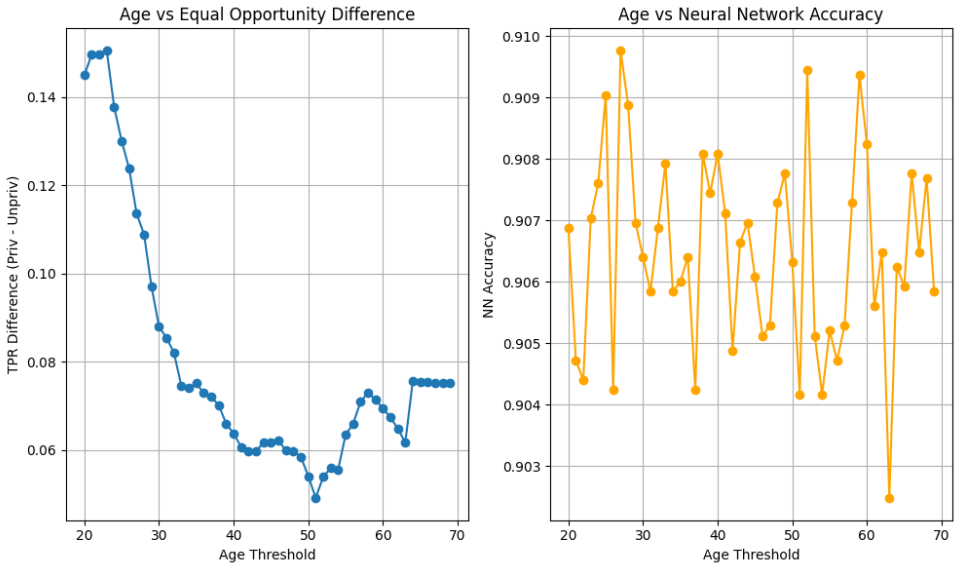

Figure 2.1: Fairness Results

2.2.1 Random Forest

For the Random Forest, we used mainly the default parameters of the scikit-learn implementation, only setting the n_estimators to 100 and the random state for reproducibility. The accuracy of the trained Random Forest on the test set was 91.26 %. True Positive Rate for the priviledged group was 94.17 % and 81.16 % for the unpriviledged group. So, the difference was 13.01 %. This shows that the model is quite unfair regarding the Equal Opportunity metric.

2.2.2 Neural Network

For the Neural Network, we created a simple feedforward architecture with 3 hidden layers, two first layers with 64 neurons and third with 32 neurons. We used ReLU as activation functions. The output layer used a sigmoid activation function for binary classification.

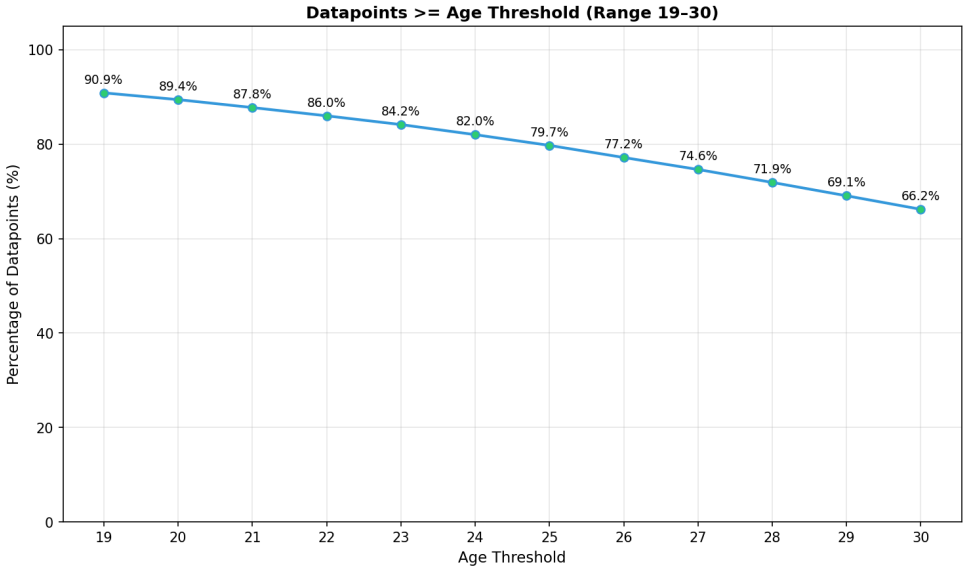

Figure 2.2: Age Threshold Distribution

We trained the model using the Adam optimizer with a learning rate of 0.001 and a batch size of 32 for 10 epochs. The accuracy of the trained Neural Network on the test set was 90.42 %. True Positive Rate for the priviledged group was 93.98 % and 82.52 % for the unpriviledged group. The difference was 11.46 %.

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.

2.4 Unfairness Mitigation

For mitigating the unfairness in our models, we could in principle choose between Preprocessing, Learning with Fairness Constraints, and Postprocessing methods. We decided to go with Learning with Fairness Constraints. It seemed to be the most interesting approach and also a good fit for our Neural Network model. 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 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. When we increased the number of epochs, we can see that at first both accuracy and unfairness increases. Then the accuracy starts to stagnate while the unfairness decreases.

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 in the NN only drops slightly in each step of more fairness, while the relative fairness δ improves drastically from no fairness with $\delta = 9.71\%$ to 0.3% with $\lambda = 0.5$ and $e = 30$.

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.

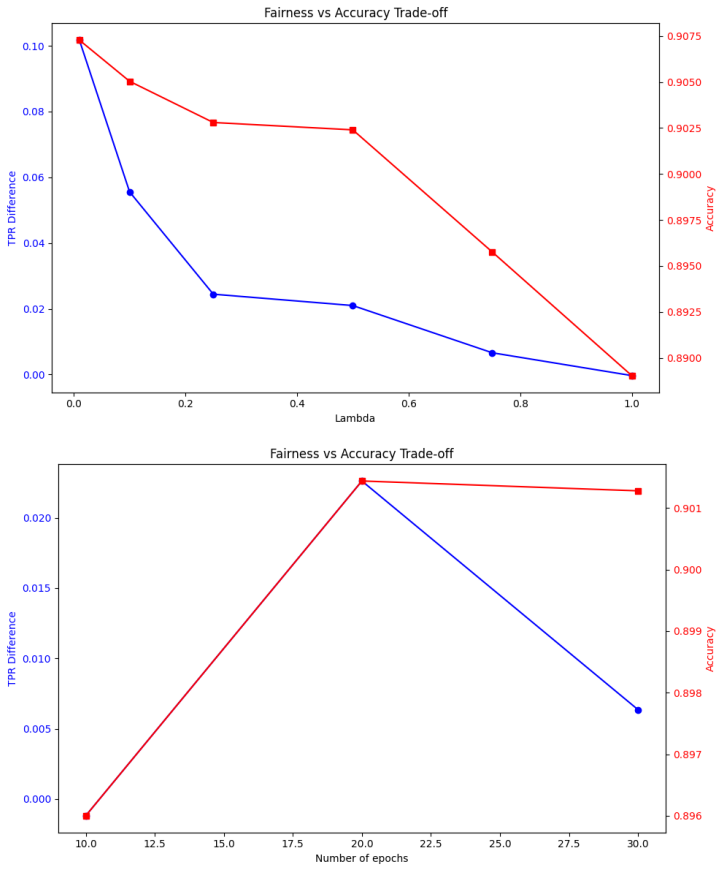

Figure 3.1: Fairness-Accuracy Tradeoff from Grid Searches

	Unfair		Fair	Best $\lambda = 0.5$	Increased Epochs
	RF	NN	NN	NN	NN
Accuracy	91.26%	90.69%	90.42%	90.24%	90.13%
Fairness (δ)	12.81%	9.71%	11.2%	1.2%	0.3%

Figure 3.2: Neural Network Results Summary

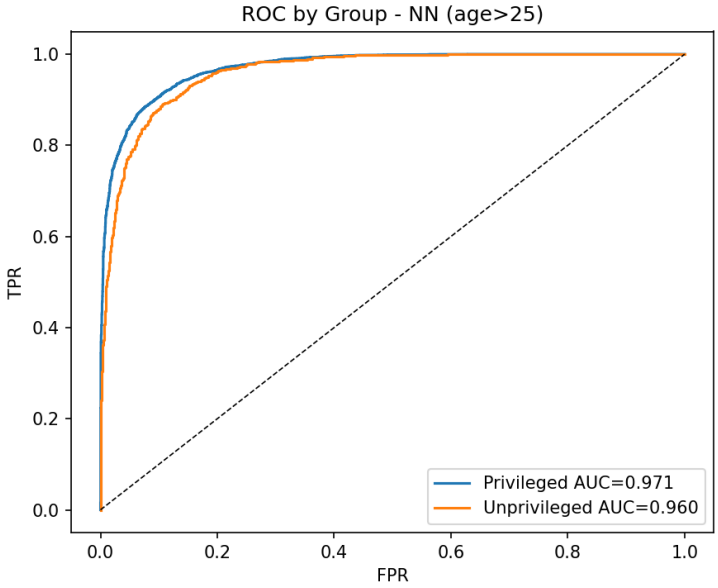

Figure 3.3: ROC by Group for Neural Network

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.