# FairPut: A Light Framework for Machine Learning Fairness

Derek Snow[1,✉]

[1]Alan Turing Institute, British Library, 96 Euston Rd, London NW1 2DB, United Kingdom

**This is a holistic framework to approach fair outputs at the individual and group level. Some of the methods developed includes quantitative monotonic measures, residual explanations, benchmark competition, adverserial attacks, disparate error analysis, model agnostic pre-and post-processing, reasoning codes, counterfactuals, contrastive explanations, and prototypical examples. A number novel techniques are proposed in this framework, each of which could benefit from future examination.**

**Fairness | Explainability | Machine Learning | Robustness | Python | FairPut**
**Correspondence:** ***dsnow@turing.ac.uk***

## Introduction

FairPut[1]. is a light open framework that describes a preferred process at the end of the machine learning pipeline to enchance model fairness. The aim is to simultaneously enhance model interpretability, robustness, and fairness while maintaining a reasonable level of accuracy. FairPut unifies various recent machine learning constructs in a practical manner. This method is model agnostic, but the particular development instance uses LightGBM.

## Framework

FairPut combines three abstractions necessary to mitigate machine learning bias. (1) Explainability, (2) Robustness, and (3) Fairness.

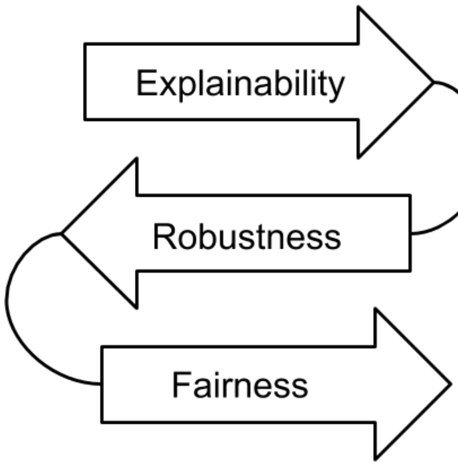


**Fig. 1.** Unifying augmentation abstractions

This report highlights the FairPut framework by walking through a practical example in the next section.

[1]https://github.com/firmai/ml-fairness-framework

The main subsections of this light framework is highlighted below. This nested list has been simplified and could be further built out to include further techniques, e.g., 3(a)(ii) model agnostic preprocessing can be further divided into reweighing preprocessing, disparate impact preprocessing, and calibrate equalized odds techniques. These additional subsections can be found in the accompanying notebook[2].

1. Explainability
   (a) Model Respesification
   (b) Quantitative Validation
2. Robustness
   (a) Residual Deviation
   (b) Residual Explanations
   (c) Benchmark Competition
   (d) Adversarial Attack
3. Fairness
   (a) Group
      i. Disparate Error Analysis
      ii. Model Agnostic Processing
      iii. Feature Decomposition
   (b) Individual
      i. Reasoning
      ii. Example Based

## Application

**Data.** The particular example is a US mortgage case study. The Home Mortgage Disclosure Act in the US was enacted in congress in 1975. As per the statute, lending institutions are required to report public lending data. In this example, we will look at a sample of consumer-anonymized loans from the HDMA database. These loans are a subset of all originated mortgage loans in the 2018 HMDA data that were chosen to represent a relatively comparable group of consumer mortgages.

[2]https://colab.research.google.com/drive/1uxSP5_CuhxjjhcT_iIeL6lsmx1gwO5B6

**Features.** A range of features were selected from this database to predict whether an applicant will default on their loan. The price of the loan is used as a proxy of default. The training set contains around 160k loans and the test set just below 40k loans. The data has been stanadardised to make it comparable accross models. The entire economic value comes down to around $50 billion.

**Protected Class.** The data includes protected class characteristics and has information on each borrower and co-borrower's race, ethnicity, gender, and age. Due to the nature of this information, certain measures that can be indicative of discrimination in lending can be calculated directly using the HDMA data. The HMDA data represent one of the most comprehensive source of data on highly-regulated information in 2020 that is publicly available, which makes it an ideal dataset to use for the types of methodological development.

## Explainability

**Model Respecification.** The FairPut framework is based on a process of selecting the best performing model for your prediction tasks and then finding model-agnostic solutions. However, often times the best performing models can be slightly respecified by adjusting the model's hyperparameters leading to better explainability, robustness, and fairness. This is especially true for gradient boosting models. To identify the reasons as to why certain groups or people are disadvantaged over others, we will create a monotononically constrained model based on target correlations. Although many researchers blindly constraint models to target correlations, this can be dangerous.

It is better to have a theoretical model to explain why correlations exist, for that reason, we will investigate all absolute correlations of 5% and up with the target variable and investigate any theoretical inconsistencies. The *monotonic constraint* is necessary when advising users on the reasons for the decision operation. On top of monotonic constraints another constraint called *feature sampling* is commended. Feature sampling helps to discourage collinear patterns from overemphasising one correlated feature over another. This value will be constrained at 50%.

These hyperparameters would be fixed, but other parameters would be allowed to float freely, and would be selected on using hyperparameter tuning methods. Blanket monotonic constraint would inhibit the model from achieving its full performance, however, each variable appearing in your reason codes should be monotonically constrained. The existence of model monotonicity can be confirmed with experiment. The empirical result of the monotonic constraints and 50% feature sampling shows that there is a 2 percentage point drop in the ROCAUC. Pure monotonicity lead to a 3 percentage point decrease. The above model would be the one we use, below is just a comparison between alternative models that can be selected, but would not be theoretically sound.

**Level Two Monotonicity.** We want to ensure that the values that would appear in our reason codes (all model constrained

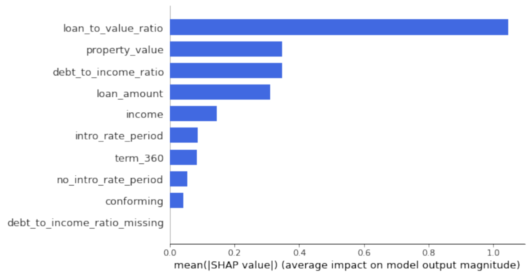


**Fig. 2.** Interpretable Global Explanations

variables) are monotonic. One might first wonder why they won't be given that they have been constrained. The reasons is that they are still allowed to interact with other variables that are not monotonically constrained. So here we want to ensure that a reasonable level of monotonicity is maintained. We will first do it visually and then quantitatively. The SHAP value on the x-axis is the change in log-odds, this value is used because of its additive properties, and because it is able to have negative values.

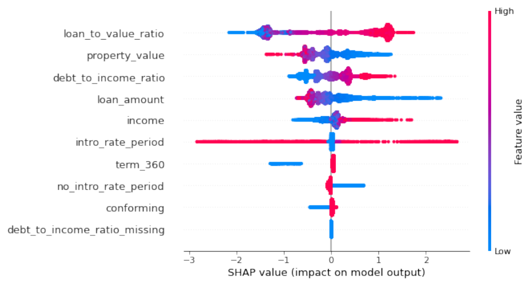


**Fig. 3.** Feature Impact on Model Output

Visually, the values of import, debt_to_income_ratio, loan_to_value_ratio, conforming, term_360, loan_amount, and property_value seems to be reasonably monotonic. The montonicity is gauged by looking at the gradual warming or cooling of colouring, without too many dots of anomalous discoloration from left to right or right to left. The only value showing some questionable behaviour is the loan_to_value_ratio. Notwithstanding the above, I am still impressed with the model's mostly gradual movement from high to low values.

**Quantitative Validation.** Here we will be studying two novel measures of monotonicity, the first is called sortedness, and the second proportionality. We will purely work with the reason code (5%+ correlation with target). Level one monotonicity promises that when the value of a feature increases all else kept equal, the output of the model increases in one direction. Level two monotonicity promises that when the value of a value increases with all other variables allowed to vary, the output of the model increases in one direction. Level two monotonicity is close to impossible, for that reason, we measure the extent of level two monotonicity. Two level two monotonicity measures developed here are sortedness and proportionality.

We have identified three different model constraints, to test the different quantitative measures. half-mono, full-mono, and no-mono. **Sortedness** essentially looks at the size of monotonic conflicts and can be seen as the effect size of monotonic deviation. The algorithm can be found on the notebook.

| | half-mono | full-mono | no-mono |
|---|---|---|---|
| **term_360** | 0.000000 | 0.000000 | 0.000000 |
| **loan_amount** | 0.263428 | 0.307617 | 0.401123 |
| **loan_to_value_ratio** | 0.229980 | 0.277588 | 0.383789 |
| **property_value** | 0.295410 | 0.547363 | 0.319336 |
| **debt_to_income_ratio** | 0.213867 | 0.264160 | 0.287109 |

**Fig. 4.** Monotonic Sortedness

**Proportionality** measures the proportion of monotonic conflicts, answering what proportion of the consecutive values are higher than the previous value.

| | half-mono | full-mono | no-mono |
|---|---|---|---|
| **term_360** | 0.000000 | 0.000000 | 0.000000 |
| **loan_amount** | 0.054901 | 0.104919 | 0.100037 |
| **loan_to_value_ratio** | 0.068359 | 0.071960 | 0.132202 |
| **property_value** | 0.107422 | 0.326660 | 0.092773 |
| **debt_to_income_ratio** | 0.031860 | 0.063965 | 0.073486 |

**Fig. 5.** Monotonic Proportionality

It seems that full monotonicity constraints impacts property value and debt-to-income ratios negatively. Because full-mono constraints all other non-reason codes, it seems to be affecting the core feature values. This gives us further evidence that half_mono would be the preferred method. All methods treat term-360 the same, this can be confirmed by looking at the gradual change in the colouring. The metrics obtained here corresponds to the coloured visualisation. For that reason, it seems that half-mono performs the best.
Lets us have a look at the effect size times the proportion per person affected for the entire sample observed. This answers the question, what features could lead to the the most possible monotonic injustices measured in quantity and size. You can think of this as the percentage of people that might face a **monotonic injustice** as a result of this feature. In this figure we will look accross all the features under investigation.

| | half-mono | full-mono | no-mono |
|---|---|---|---|
| **term_360** | 0.000000 | 0.000000 | 0.000000 |
| **conforming** | 0.000000 | 0.000000 | 0.000000 |
| **debt_to_income_ratio_missing** | 0.000000 | 0.000000 | 0.000000 |
| **loan_amount** | 0.014297 | 0.040683 | 0.035013 |
| **loan_to_value_ratio** | 0.023019 | 0.024239 | 0.064724 |
| **no_intro_rate_period** | 0.000000 | 0.000000 | 0.000000 |
| **intro_rate_period** | 0.985008 | 0.691334 | 1.176049 |
| **property_value** | 0.027686 | 0.103156 | 0.031240 |
| **income** | 0.026021 | 0.001365 | 0.024470 |
| **debt_to_income_ratio** | 0.010168 | 0.023027 | 0.027362 |

**Fig. 6.** Monotonic Injustic

**Partial Dependence Monotonicity.** This a less strict level of monotonicity. We can also call it level one monotonicity. Please see the notebook for additional information as this section is more involved. The following three figures are indicative of some partial dependence relationships.

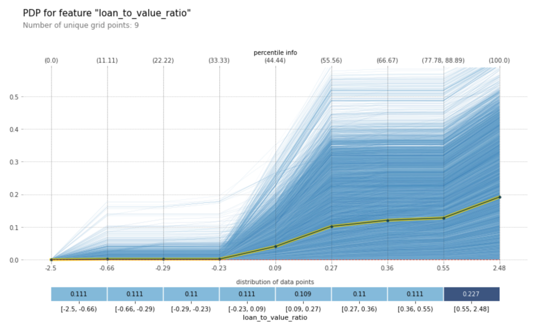


**Fig. 7.** Histogram Partial Dependence Income

The partial dependence plot for the average effect of a feature is a global method because it does not focus on specific instances, but on an overall average. The equivalent to a PDP for individual data instances is called individual conditional expectation (ICE) plot. A PDP is the average of the lines of an ICE plot. The ice plot comes closer to the concept of level two monotonicty, why do some instances react stronger to the same feature value change than others? Is this reaction justified. ICE plots provide more insight into interactions, like the shapley value method. Although visually appealing, the individual ICE method doesn't allow for a nice empirical study on level monotonicity.

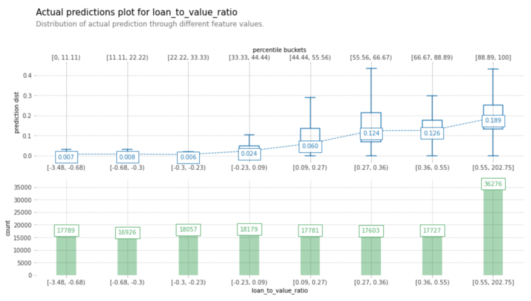


**Fig. 8.** Range Partial Dependence Loan-to-Value Ratio

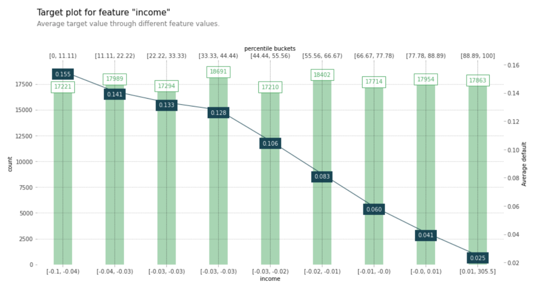


**Fig. 9.** ICE Plot of Dependence Loan-to-Value Ratio

**Feature Interaction.** Here we build a Shapely tree explainer using a small sample of observations. We construct the variable inter that primarily looks at feature interactions. This continues with our half-mono model. Here I am explaining the model's interactions on the training set. You can also look at the test set, or both sets combined, it should give you very similar results. Here we build a Shapely tree explainer using a small sample of observations. We construct the variable inter that primarily looks at feature interactions.

We take the mean absolute value and list the strongest interacting pairs, the goal which is to explain above non-linear relationships (features for which the colour patterns change, are scattered, or are not gradual). The hope is that we would come to understand why the features behave as they do as an additional robustness step.

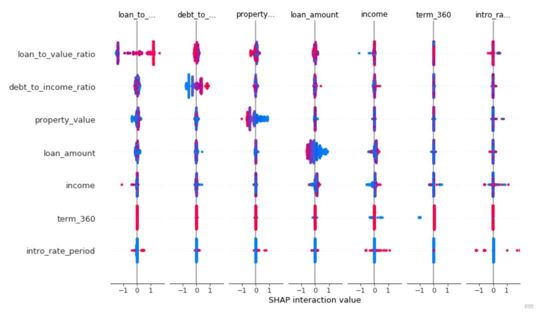


**Fig. 10.** Feature Interactions

The relationships that are the strongest is loan_amount and income (0.039) and next loan_to_value_ratio and no_intro_rate_period (0.007). The first interaction is by far the strongest.

At this point we can select the best model considering the previous discussions and select the preferred decision threshold.

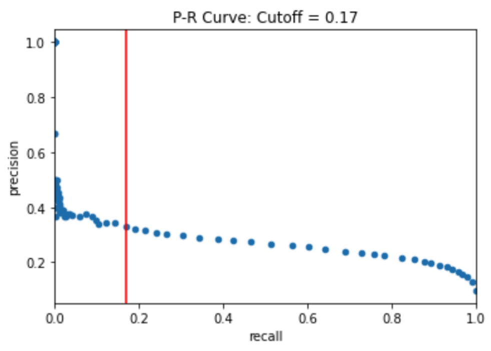
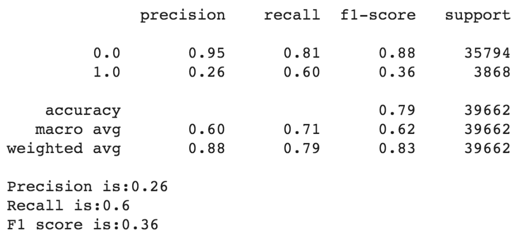


**Fig. 11.** Precision Recall Curve Optimal Decision Threshold



```
              precision    recall  f1-score   support

         0.0       0.95      0.81      0.88     35794
         1.0       0.26      0.60      0.36      3868

    accuracy                           0.79     39662
   macro avg       0.60      0.71      0.62     39662
weighted avg       0.88      0.79      0.83     39662

Precision is:0.26
Recall is:0.6
F1 score is:0.36
```

**Fig. 12.** Model Performance at Optimal Decision Threshold

## Robustness

**Residual Deviation.** Log-loss and residual deviation measures are plotted. Whether the deviation is the largest, the model has predicted at its worse. The individuals affected should be identified to ensure that their is no risk for systematic biases.

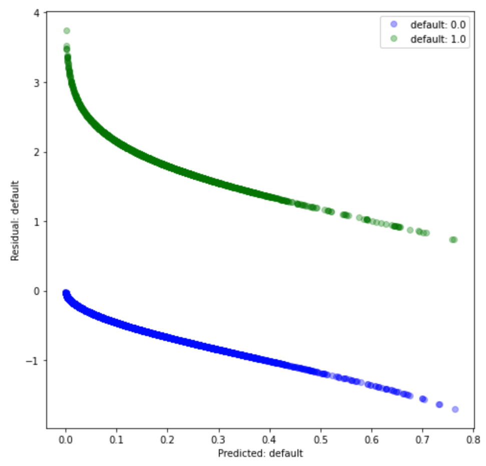


**Fig. 13.** Log-loss Residuals

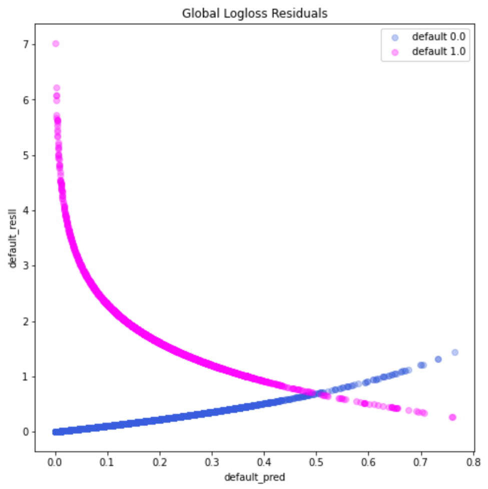
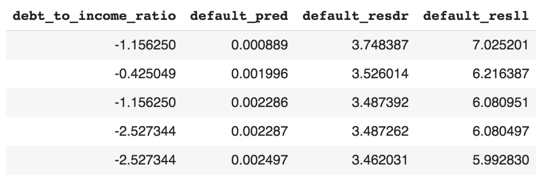


**Fig. 14.** Residual Deviation

An dataframe snippet of the most affected individuals.



| debt_to_income_ratio | default_pred | default_resdr | default_resll |
|---|---|---|---|
| -1.156250 | 0.000889 | 3.748387 | 7.025201 |
| -0.425049 | 0.001996 | 3.526014 | 6.216387 |
| -1.156250 | 0.002286 | 3.487392 | 6.080951 |
| -2.527344 | 0.002287 | 3.487262 | 6.080497 |
| -2.527344 | 0.002497 | 3.462031 | 5.992830 |

**Fig. 15.** Dataframe Sorted by Most Affected Individuals

High positive residuals seems to be caused by the model believing that the mortgage won't be defaulted on, but it in fact is. Three values that seem to be particularly low for these individuals is the loan_to_value_ratio, debt_to_income_ratio and term_360 value. There combined effect seems to confuse the prediction model. At some point we would have to see if we can obtain similar individuals that have indeed remain solvent.

**Residual Explanations.** In this analysis we also look for outliers by looking at the features. Application 5111213 was predicted to have a high default. But in reality it was low. Lets try and understand this outlier by looking at the shapley values individually for that amount in the next code block.

| Predicted Feature | Overprediction Index | Overpredict Percentage | Predicted (O) | Actual (O) |
|---|---|---|---|---|
| default | 4325410 | 92 | 0.923550 | 0.0 |
| default | 5681158 | 92 | 0.920335 | 0.0 |
| default | 157230 | 86 | 0.866779 | 0.0 |
| default | 1718320 | 86 | 0.866123 | 0.0 |
| default | 1338283 | 83 | 0.832266 | 0.0 |

**Fig. 16.** Outliers - Overpredicted

| Predicted (O) | Actual (O) | Underprediction Index | Underpredict Percentage | Predicted (U) | Actual (U) |
|---|---|---|---|---|---|
| 0.923550 | 0.0 | 2143088 | -84 | 0.151495 | 1.0 |
| 0.920335 | 0.0 | 4327992 | -84 | 0.156476 | 1.0 |
| 0.866779 | 0.0 | 5193624 | -83 | 0.162606 | 1.0 |
| 0.866123 | 0.0 | 5227931 | -83 | 0.163625 | 1.0 |
| 0.832266 | 0.0 | 1861080 | -83 | 0.166587 | 1.0 |

**Fig. 17.** Outliers - Underpredicted

Now lets take a deeper dive into the most over-predicted value.

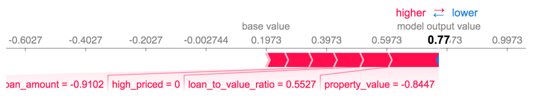


**Fig. 18.** Overprediction Explanation Application 5111213

And lets look at the feature attribution for a group of over-predicted samples. Looking at this figure we can see the driving factors, this can help us to identify whether we are dealing with unfair biases. We can also do the same for under-prediction, but that is skipped in this report; please see the notebook for additional information.

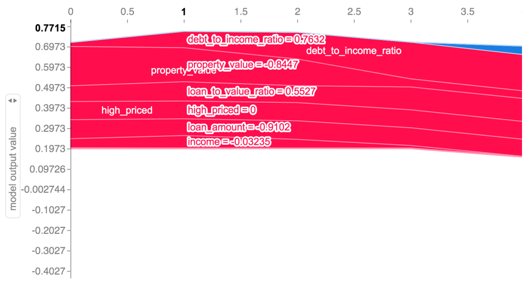


**Fig. 19.** Overprediction Multiple Samples Attribution

**Residual Drivers.** Looking at all observations what is the features driving over and under prediction. Higher property value and intro rate period leads to over-prediction, higher no intro rate period and higher loan amount leads to under-prediction. Additional variables related to the top causes might be needed to place them into better context when making predictions.

| Top Feature | REL SHAP Value | Causes Overprediction (CO) | Effect Size (CO) |
|---|---|---|---|
| loan_to_value_ratio | 1.000000 | intro_rate_period | 0.001398 |
| debt_to_income_ratio | 0.517486 | property_value | 0.001284 |
| loan_amount | 0.439055 | debt_to_income_ratio | 0.000553 |
| property_value | 0.349000 | term_360 | 0.000311 |
| income | 0.300727 | conforming | 0.000010 |

**Fig. 20.** Causes of Overprediction

**Benchmark Competition.** Benchmark models are an excellent model debugging tool. They can be used at training time to understand how a new model differs from an established, trusted model. They can also be used at scoring time to understand if a newer or more complex model is giving different predictions from a previously deployed trusted model or simpler model. If a prediction from a new model is too different from a prediction from a trusted model, this could be indicative of potential accuracy, fairness, or security problems. See the notebook for multiple model comparisons.

**Benchmark Disagreement.** One interesting place to start comparing a more complex model to a simpler model is when the simple model is right and the complex model is wrong. For a range of probabilities between 0.1 and 0.9 there exists a group of customers where a GLM model gives more correct predictions than the more complex GBM model. In the plot in the notebook, the yellow points represent the known target labels, the pink line is the sorted GBM model predictions, and the blue line is the GLM predictions for the same customers and target labels. For this group of people the GLM is obviously able to better represent some attribute of the customer's demographics or behaviors. Can the differences between this group of people and the rest of the customers be identified and leveraged to make better predictions?

**Adversarial Examples.** We can artificially adjust values hundreds of thousands of times to see if we can artificially induce non-sensible predictions.

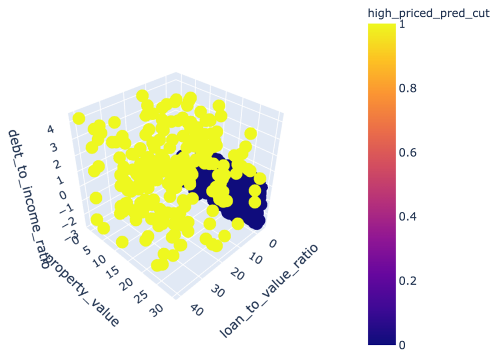


**Fig. 21.** Plotting All Binary Prediction

From this vantage point, we can further investigate if some sample predictions behave in an unexpected fashion.
The next plot might be more informative as it presents the user with continuous values. We can see that there are cases

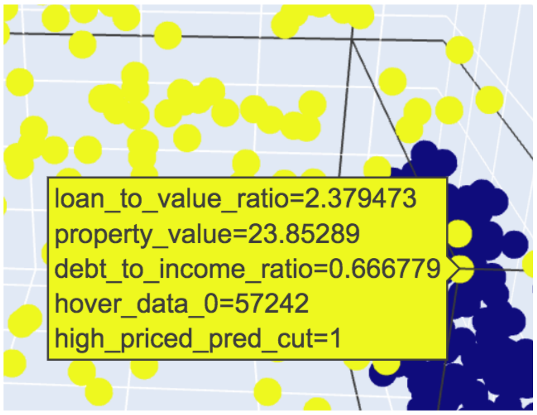


**Fig. 22.** Vicinity of Potential Adversarial Examples

where distinct colours appear close to eachother, but it does not look systematic.

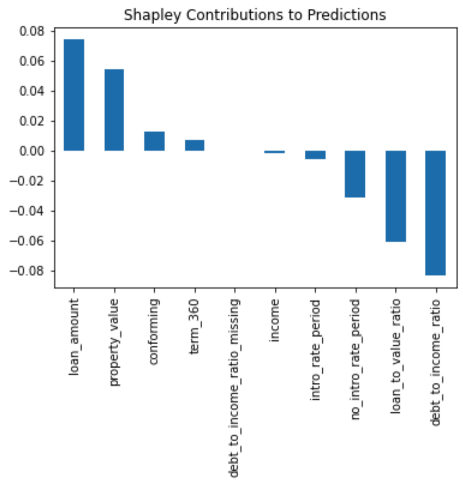


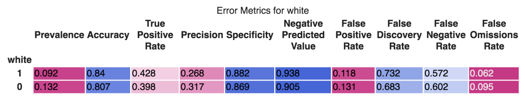
**Fig. 23.** Vicinity with Continuous Values

## Fairness

Group fairness, in its broadest sense, partitions a population into groups defined by protected attributes and seeks for some statistical measure to be equal across groups. Individual fairness, in its broadest sense, seeks for similar individuals to be treated similarly.

## Group Level

**Disparate Error Analysis (DEA).** A quantitative technique to address group fairness.

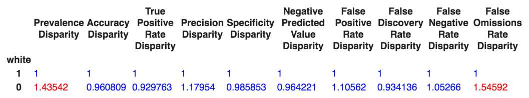


Error Metrics for white

| white | Prevalence | Accuracy | True Positive Rate | Precision | Specificity | Negative Predicted Value | False Positive Rate | False Discovery Rate | False Negative Rate | False Omissions Rate |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.092 | 0.84 | 0.428 | 0.268 | 0.882 | 0.938 | 0.118 | 0.732 | 0.572 | 0.062 |
| 0 | 0.132 | 0.807 | 0.398 | 0.317 | 0.869 | 0.905 | 0.131 | 0.683 | 0.602 | 0.095 |

**Fig. 24.** DEA Analysis

Prevalance: there are about 40% more defaults for non-white groups in the dataset –Bad Sign For Minority. Accuracy: the accuracy in prediction is 4% lower for non-white groups. TPR: the model better predicted those who would default for white rather than non-white groups (7pp difference) – Good Sign For Minority. Precision: when the model predicted that someone would default, it was more precise in predicting that for non-white groups. – Good Sign For Minority. Specificty: of the people that did not default, the model correctly predicted those non-defaults in a similar ratio between groups. NPV: for model predicted non-defaults, there was an equal ratio of correctly predicted non-defaults between white and non-white groups. FPR (Important): out of the people that did not default, the model incorrectly predicted that more non-whites would proportionately default (10pp) – Bad Sign For Minority. FDR (Important): The model incorrectly predicted that more whites would default that did not default (7pp) – Good Sign For Minority. FNR (Important): For the amount of people that would default, the model predicted more incorrectly that non-whites would not default – Good Sign For Minority. FOR (Important): 50% more of the time the model incorrectly predicted that non-whites would not default, but they did default – Good Sign For Minority.



| white | Prevalence Disparity | Accuracy Disparity | True Positive Rate Disparity | Precision Disparity | Specificity Disparity | Negative Predicted Value Disparity | False Positive Rate Disparity | False Discovery Rate Disparity | False Negative Rate Disparity | False Omissions Rate Disparity |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1.43542 | 0.960809 | 0.929763 | 1.17954 | 0.985853 | 0.964221 | 1.10562 | 0.934136 | 1.05266 | 1.54592 |

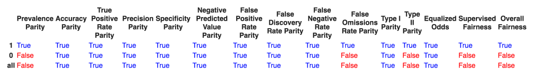
**Fig. 25.** DEA Threshold

There are generally regulation that prescribes the level of suitable deviation, here it has been set at 20%. Here we will highlight some of the important metrics used in industry. FPR (Important): out of the people that did not default, the model incorrectly predicted that more males would proportionately default (30pp) – Good Sign For Minority. FDR (Important): The model incorrectly predicted that more females would default that did not default (4pp) – Bad Sign For Minority. FNR (Important): For the amount of people that would default, the model predicted more incorrectly that females would not default – Good Sign For Minority. FOR (Important): 12% more of the time the model incorrectly predicted that men would not default, but they did default – Bad Sign For Minority.

**Majoritarian Fairness.** This is a novel technique we have developed to address fairness. Train the model on all feature, including sensitive feature, then choose the model demographic intersection as the only prediction class, thus making the model superficially colour blind, i.e. change the test set to all have the same demographical characteristics. See the notebook for additional details.

**Parity Indicators.** A binary indication of parity for metrics is reported by simply checking whether disparity values are within the user-defined thresholds. Further parity indicators are defined as combinations of other disparity values: Type I Parity: Both FDR Parity and FPR Parity. Type II Parity: Both FOR Parity and FNR Parity. Equalized Odds: Both FPR Parity and TPR Parity. Supervised Fairness: Both Type I and Type II Parity Overall Fairness: Fairness across all metrics.

**Model Agnostic Pre-and Post-Processing.** Among preprocessing algorithms, reweighing only changes weights applied to training samples; it does not change any feature or



| | Prevalence Parity | Accuracy Parity | True Positive Rate Parity | Precision Parity | Specificity Parity | Negative Predicted Value Parity | False Positive Rate Parity | False Discovery Rate Parity | False Negative Rate Parity | False Omissions Rate Parity | Type I Parity | Type II Parity | Equalized Odds | Supervised Fairness | Overall Fairness |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | True | True | True | True | True | True | True | True | True | True | True | True | True | True | True |
| 0 | False | True | True | True | True | True | True | True | True | False | True | False | True | False | False |
| all | False | True | True | True | True | True | True | True | True | False | True | False | True | False | False |

**Fig. 26.** Parity Indicators

label values. Therefore, it may be a preferred option in case the application does not allow for value changes. Disparate impact remover and optimized pre-processing yield modified datasets in the same space as the input training data, whereas LFR's pre-processed dataset is in a latent space. If the application requires transparency on the transformation, then disparate impact remover and optimized pre-processing may be preferred options. Moreover, optimized pre-processing addresses both group fairness and individual fairness.
Among post-processing algorithms, the two equalized odds post-processing algorithms have a randomized component whereas the reject option algorithm is deterministic, and may be preferred for that reason.
The current AIF360 implementations of some algorithms take arguments on which fairness metric to optimize (e.g. optimized pre-processing and reject option) and some do not (e.g. disparate impact remover and equalized odds post-processing), which may imply better and worse performance by some algorithms with respect to some metrics.

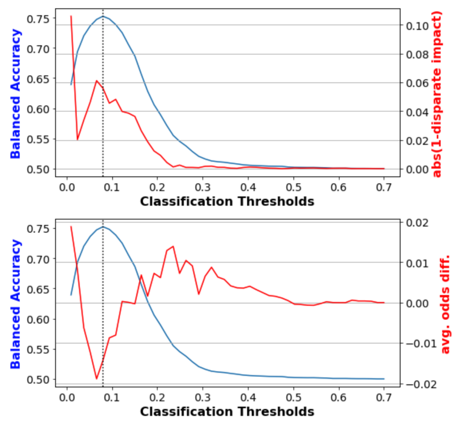


**Fig. 27.** Calibrated Odds Postprocessing

**Calibrated Odds Postprocessing** changes predictions from a classifier to make them fairer. Provides favorable outcomes to unprivileged groups and unfavorable outcomes to privileged groups in a confidence band around the decision boundary with the highest uncertainty. Among post-processing algorithms, the two equalized odds post-processing algorithms have a randomized component whereas the reject option algorithm is deterministic, and may be preferred for that reason.
Other techniques that are found in the notebook and GitHub repository is Disparate Impact Preprocessing, and a Reweighing Preprocessing Technique. It is best to start with preprocessing to mediate bias, and then to use post-processing, the earliest your can mediate it, the greatest level of flexibility is left in later stages for further correction. Among pre-processing algorithms, reweighing only changes weights applied to training samples; it does not change any feature or label values. Therefore, it may be a preferred option in case the application does not allow for value changes. Disparate impact remover is a preprocessing technique that edits feature values increase group fairness while preserving rank-ordering within groups.Learning fair representations is a preprocessing technique that finds a latent representation which encodes the data well but obfuscates information about protected attributes.
What does the privileged group have that contributes more off and less off to the final prediction. You could also include the sensitive features and see if the order of importance change. If the relative importance change, it could be evidence of a biased variable.

**Feature Decomposition.** Here our primary focus is on the equity doctrine and looking at the structural biases by decomposing the features. Decomposing quantitative fairness metrics using SHAP can reduce their opacity when the metrics are driven by measurement biases effecting only a few features.
The original half-mono LightGBM model is going to be used because it's performance on the fairness metrics is reasonable. Let's retrain the model here. What does the privileged group have that contributes more off and less off to the final prediction. You could also include the sensitive features and see if the order of importance change. If the relative importance change, it could be evidence of a biased variable.

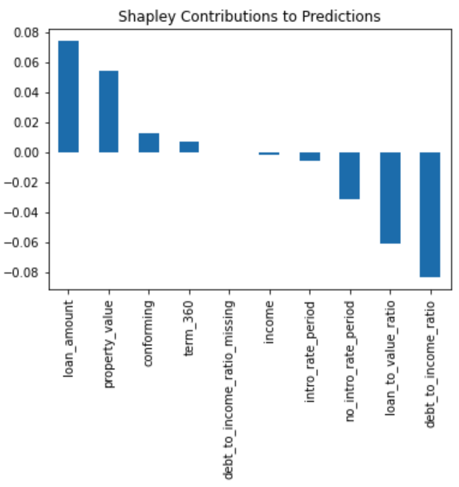
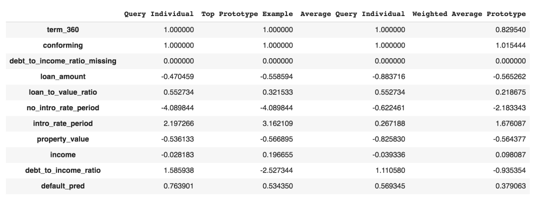


**Fig. 28.** Privileged Feature Outcome

Here we can clearly see the factors driving the differences between a privileged and non-privileged group's prediction outcomes.

## Individual Level

**Prototypical Explanations.** Attempts to find examples in the database that are most like a sample given. A way to see if the sample was treated unfairly or different. It is comforting to know that the top prototype would get the same predicted outcome. So this individual is not entirely unfairly treated. They also have higher debt to income and lower income than the prototype so the differences in the default prediction also seems intuitive.



| | Query Individual | Top Prototype Example | Average Query Individual | Weighted Average Prototype |
|---|---|---|---|---|
| term_360 | 1.000000 | 1.000000 | 1.000000 | 0.829540 |
| conforming | 1.000000 | 1.000000 | 1.000000 | 1.015444 |
| debt_to_income_ratio_missing | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| loan_amount | -0.470459 | -0.558594 | -0.883716 | -0.565262 |
| loan_to_value_ratio | 0.552734 | 0.321533 | 0.552734 | 0.218675 |
| no_intro_rate_period | -4.089844 | -4.089844 | -0.622461 | -2.183343 |
| intro_rate_period | 2.197266 | 3.162109 | 0.267188 | 1.676087 |
| property_value | -0.536133 | -0.566895 | -0.825830 | -0.564377 |
| income | -0.028183 | 0.196655 | -0.039336 | 0.098087 |
| debt_to_income_ratio | 1.585938 | -2.527344 | 1.110580 | -0.935354 |
| default_pred | 0.763901 | 0.534350 | 0.569345 | 0.379063 |

**Fig. 29.** Prototypical Examples

**Counterfactual Explanations.** Simply answers what should be minimally changed to change the prediction. What should be changed to go from a default to a no-default prediction. A model is not robust or fair if changes occur easily and the prediction are flipped from small perturbations. The results in the notebook show that as little as a 10% change in one variable led to an almost 60% change in outcome probability.

**Contrastive Explanations.** *Pertinent Negative* (PN) identifies what features should be minimally and necessarily absent from the instance to be explained in order to maintain the original prediction class. The aim of PN's is not to provide a full set of characteristics that should be absent in the explained instance, but to provide a minimal set that differentiates it from the closest different class. Here it gives the same result as the counterfactual. *Pertinent Positive* (PP) finds the features that should be minimally and sufficiently present (e.g. important pixels in an image) to predict the same class as on the original instance.