1 How To Use This Document

Highly regulated industries, such as banking and insurance, must comply with government regulations for model validation before a model can be put into production. This includes creating robust model development documentation. DataRobot automates the generation of model documentation, expediting the process required for regulatory compliance and following best practice for reducing model risk.

This document is split into two components: those sections that are automatically produced by DataRobot and those that require further input by the user. The sections in ***blue italicized font*** include specific instructions for the documenter and require additional user input of organization-specific information, such as business use cases, data sources, and implementation details. Once the sections are complete, remove the instructions. The remaining sections in non-blue italicized font are automatically populated by DataRobot and require no further input.

DataRobot

Table of Contents

2 Model Performance Overview

As an additional layer of model validity, DataRobot not only evaluated the statistical metrics underlying the model, but also performed testing on in-sample records.

The performance metric used for this project was LogLoss. The model performance results are presented below for in-sample testing:

Scoring Type	Score (LogLoss)
cross_validation	0.3791*
holdout	0.2942*
validation	0.3801*

3 Model Features Summary

Below are two tables. The first contains a list of the final set of model feature variables, as well as summary statistics for the eXtreme Gradient Boosted Trees Classifier with Early Stopping (learning rate =0.02) model. The second table contains a detailed analysis of missing values.

The Model Features and Summary Statistics table provides a brief overview of the summary statistics of model features. This includes Feature Name, variable type (Var Type), number of unique values (Unique), Number of missing values (Missing), Mean, Standard Deviation (Std Dev), Median, Minimum Value (Min), Maximum Value (Max) and Assessment of target leakage risk (Target Leakage).

3.1 Model Features and Summary Statistics

Feature Name	Var Type	Unique	Missing	Mean	Std Dev	Median	Min	Max	Target Leakage
water_source_clean	Categorical	7	13377	N/A	N/A	N/A	N/A	N/A	Low
water_source_category	Categorical	4	13377	N/A	N/A	N/A	N/A	N/A	Low
age_in_years	Numeric	6246	236	9.48	9.8	7.29	-12.32	106.37	Low
distance_to_city	Numeric	74864	0	43164.58	30577.34	37188.65	55.53	222598.81	Low
distance_to_town	Numeric	74864	0	15840.64	11210.74	13604.69	3.8	58399.95	Low
bgs_recharge	Numeric	252	0	95.096	22.97	94.65	33.26	145.18	Low
water_risk	Numeric	65	0	2.3	0.56	2.086	1.63	4.18	Low
assigned_population	Numeric	2794	16	420.49	554.56	280.0	0.0	30964.0	Low
pressure	Numeric	5242	2491	2.42	4.86	1.29	0.0033	415.82	Low

crucialness	Numeric	60455	2491	0.304	0.27	0.208	0.0042	1.0	Low
population_1km	Numeric	61908	2	2165.047	3245.87	1195.13	-0.0	40327.99	Low
population_10km	Numeric	38844	2	124054.11	175394.47	86397.23	-0.0	2266196.25	Low
men_ratio_10km	Numeric	37795	2	0.48	0.058	0.48	-5.5	5.5	Low
elderly_ratio_10km	Numeric	31741	2	0.041	0.0066	0.041	-0.42	0.43	Low
population_100km	Numeric	1888	0	6844758.85	3078234.17	6327505.5	729392.0	14655119.0	Low
men_ratio_100km	Numeric	1882	0	0.49	0.005	0.49	0.48	0.506	Low
women_of_reproductive_age_ratio_100km	Numeric	1887	0	0.23	0.0032	0.23	0.22	0.25	Low
youth_ratio_100km	Numeric	1832	0	0.205	0.0019	0.206	0.2	0.22	Low
children_under_five_ratio_100km	Numeric	1861	0	0.18	0.0036	0.18	0.15	0.18	Low
years_since_report	Numeric	6012	0	9.18	9.66	6.54	-12.32	106.37	Low

The last column in this table is an assessment of target leakage risk. DataRobot automatically tests for target leakage on a per-feature basis during the Autopilot process. Target leakage, sometimes called data leakage, occurs when a model is trained using a dataset that includes information that would not be available at the time of prediction. This can produce overly optimistic model performance results during training, given a feature will near-completely describe the target (e.g., the number of late payments on a loan as a predictor for loan default at loan application date.)

DataRobot tests for target leakage risk using Alternating Conditional Expectation (ACE) to measure the association between each feature and the target; the ACE score is normalized using the project optimization metric so that its value is in the range [0,1]. If above a certain threshold (see below), DataRobot will create a new feature list with those features flagged and possibly removed, and the user is notified by a banner in the user interface during modeling. Notably, because the definition of target leakage is directly tied with prediction time and not strength of association between a feature and the target, it's possible for DataRobot to not identify all sources of target leakage. Therefore, to reduce the risk for potential target leakage in the feature list, it's important to apply subject matter expertise.

The thresholds for target leakage risk are based on a normalized ACE score:

- High risk: > 0.975, flagged and removed
- Moderate risk: > 0.85, flagged but not removed
- Low risk: < 0.85, no action

The following table provides a summary of missing values. It includes the name of the feature, its type, a summary of the missing value count (both number of rows and as a percentage), and information on the type of imputation applied to the feature.

3.2 Data Quality Handling Report

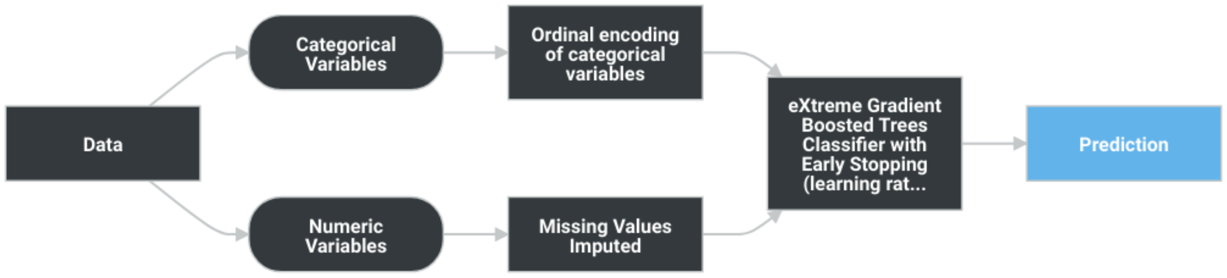
Feature Name	Var Type	Missing Count	Missing Percentage	Imputation Name	Imputation Description

water_source_clean	Categorical	14954	16	Ordinal encoding of categorical variables	Imputed value: -2
water_source_category	Categorical	14954	16	Ordinal encoding of categorical variables	Imputed value: -2
pressure	Numeric	2802	3	Missing Values Imputed	Imputed value: -9999
crucialness	Numeric	2802	3	Missing Values Imputed	Imputed value: -9999
age_in_years	Numeric	547	1	Missing Values Imputed	Imputed value: -9999
assigned_population	Numeric	222	0	Missing Values Imputed	Imputed value: -9999
population_1km	Numeric	2	0	Missing Values Imputed	Missing indicator treated as feature, Imputed value: 1182.2802
population_10km	Numeric	2	0	Missing Values Imputed	Imputed value: 85512.516
men_ratio_10km	Numeric	2	0	Missing Values Imputed	Imputed value: 0.4843
elderly_ratio_10km	Numeric	2	0	Missing Values Imputed	Imputed value: 0.0415
distance_to_city	Numeric	0	0	Missing Values Imputed	Imputed value: 36681.848
distance_to_town	Numeric	0	0	Missing Values Imputed	Imputed value: 14119.197
bgs_recharge	Numeric	0	0	Missing Values Imputed	Imputed value: 96.6499
water_risk	Numeric	0	0	Missing Values Imputed	Imputed value: 2.0858
population_100km	Numeric	0	0	Missing Values Imputed	Imputed value: 6132147
men_ratio_100km	Numeric	0	0	Missing Values Imputed	Imputed value: 0.486
women_of_reproductive_age_ratio_100km	Numeric	0	0	Missing Values Imputed	Imputed value: 0.2334
youth_ratio_100km	Numeric	0	0	Missing Values Imputed	Imputed value: 0.206
children_under_five_ratio_100km	Numeric	0	0	Missing Values Imputed	Imputed value: 0.1769
years_since_report	Numeric	0	0	Missing Values Imputed	Imputed value: 6.3218

4 Model Summary and Description

The particular model referenced in this document: eXtreme Gradient Boosted Trees Classifier with Early Stopping (learning rate =0.02). This model was developed in a project created with vb754f77d02337f3d of DataRobot. This model is denoted within DataRobot by the Project ID: 63fd9f4c81f96be8177a00e9 and the Model ID: 6409695354d5ddc4f653ba26. The project was created on 2023-02-28 06:29:32.

The model development workflow process (i.e., the model blueprint) is detailed in the figure below.


Data
Categorical Variables
Numeric Variables
Ordinal encoding of categorical variables
Missing Values Imputed
eXtreme Gradient Boosted Trees Classifier with Early Stopping (learning rat...
Prediction

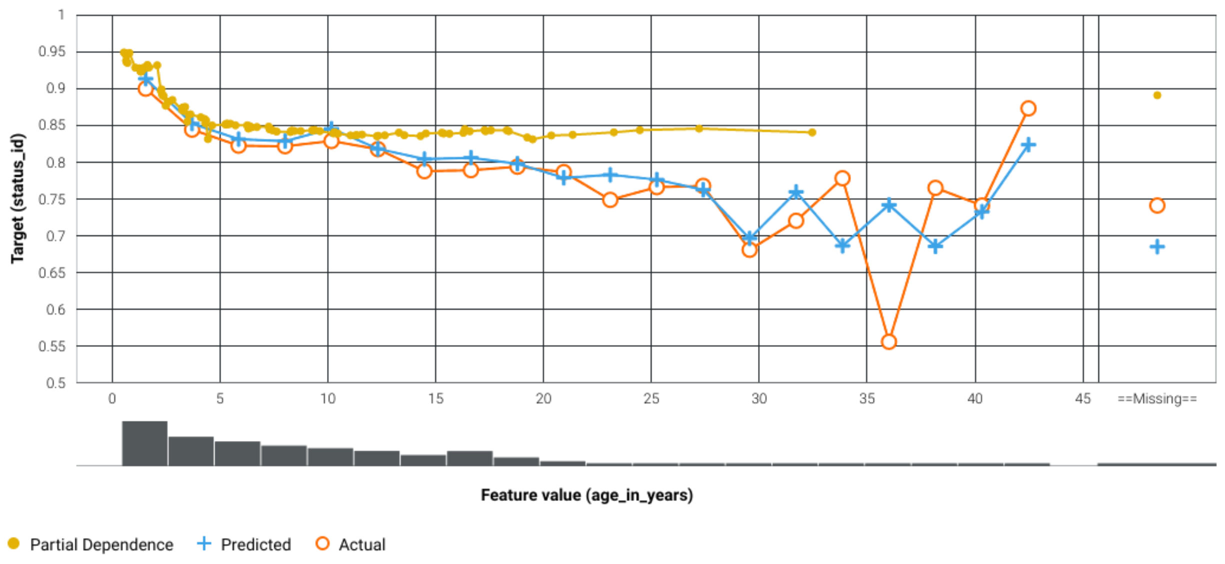
A Blueprint represents the high-level end-to-end procedure for fitting the model, including any preprocessing steps, algorithms, and post-processing. It illustrates the many steps involved in transforming input predictors and targets into a model. Each element (or, “node”) in a blueprint can represent multiple steps.

The following elements connect to create the blueprint:

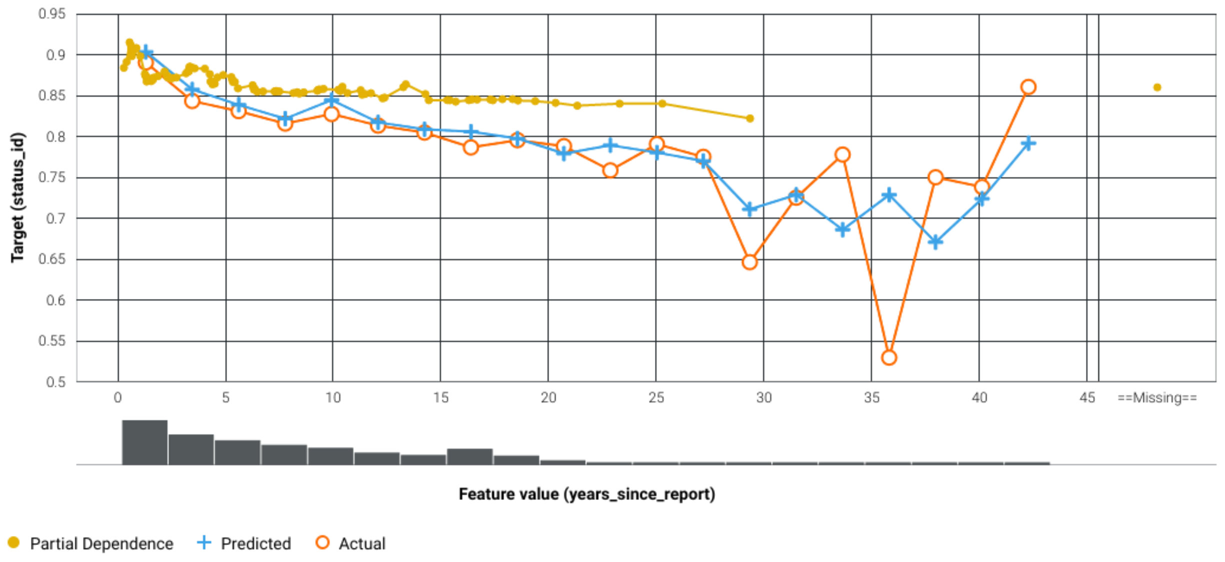
- Ordinal encoding of categorical variables
- Missing Values Imputed
- eXtreme Gradient Boosted Trees Classifier with Early Stopping (learning rate =0.02)

5 Feature Effects

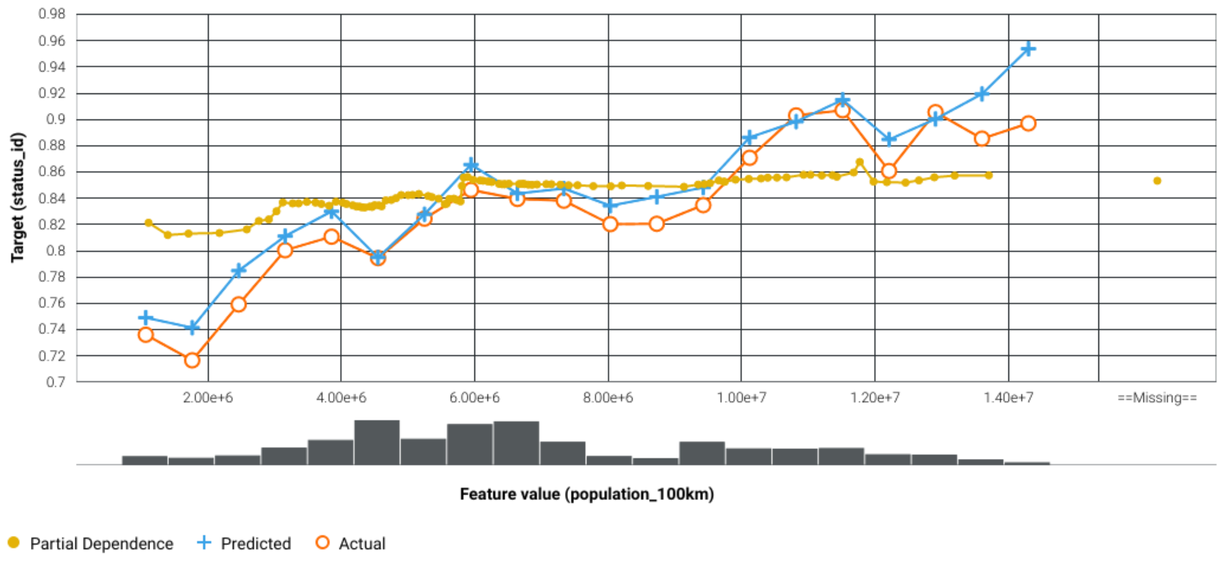
age_in_years



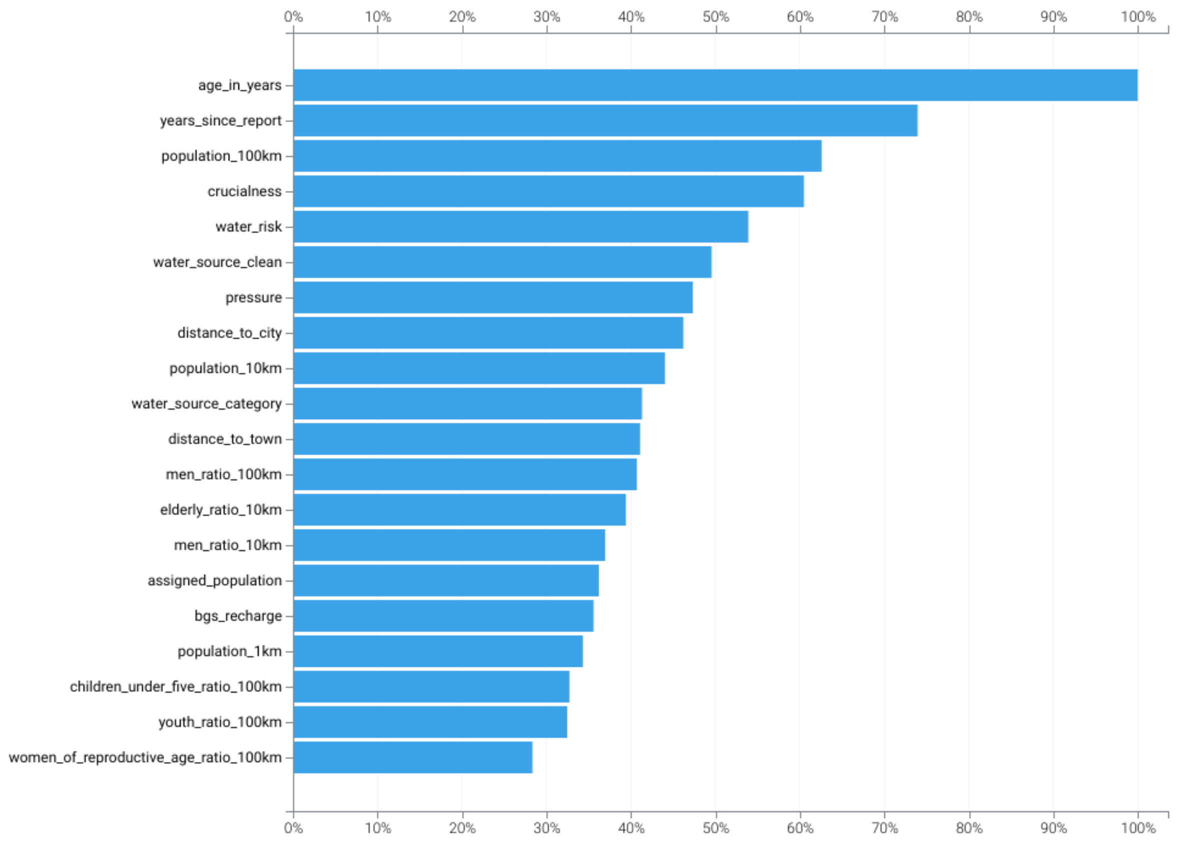
years_since_report



population_100km



6 Feature Impact Chart



7 Feature Impact Table

Feature Name	Impact Normalized	Impact Unnormalized
age_in_years	1.0	0.0526
years_since_report	0.7394	0.0389
population_100km	0.6258	0.0329
crucialness	0.6048	0.0318
water_risk	0.5389	0.0283
water_source_clean	0.4955	0.0261
pressure	0.4733	0.0249
distance_to_city	0.4621	0.0243

population_10km	0.4404	0.0232
water_source_category	0.4132	0.0217
distance_to_town	0.4109	0.0216
men_ratio_100km	0.4071	0.0214
elderly_ratio_10km	0.394	0.0207
men_ratio_10km	0.3695	0.0194
assigned_population	0.3621	0.019
bgs_recharge	0.3558	0.0187
population_1km	0.3431	0.018
children_under_five_ratio_100km	0.3272	0.0172
youth_ratio_100km	0.3246	0.0171
women_of_reproductive_age_ratio_100km	0.2835	0.0149

8 Validation Testing and Stability

To find patterns in a dataset from which it can make predictions, an algorithm must first learn from a historical example – typically from a historical dataset that contains the output variable you want to predict. However, if a model is trained too closely on its training data then it may be overfit. Overfitting is a modeling error that occurs when a model is too closely fit to training data and therefore performs poorly on out-of-sample data (data that was not used to train the model). Overfitting generally results in an overly complex model that explains idiosyncrasies and random noise in the training data, rather than the underlying trends that the model was intended to capture. To avoid overfitting, the best practice is to evaluate model performance on out-of-sample data. If the model performs very well on in-sample data, (the training data) but poorly on out-of-sample data, that may be an indication that the model is overfit.

DataRobot uses standard modeling techniques to validate model performance and ensure that overfitting does not occur. DataRobot used a robust model k-fold cross-validation framework to test the out-of-sample stability of a model's performance. In addition to cross-validation partitioning, DataRobot uses a holdout sample to further test out-of-sample model performance and ensure the model is not overfit.

The following procedure was used during development to insure that overfitting did not occur:

- All values of the feature "fold" represent separate partitions used for cross-validation

DataRobot calculates the Cross Validation scores for each of the training data partitions or folds. The project metric used to calculate the score is LogLoss.

8.1 Cross Validation Scores

Fold	Cross Validation Score (LogLoss)
Fold 1	0.38009
Fold 2	0.37331
Fold 3	0.3803
Fold 4	0.38456
Fold 5	0.37704

9 Model Results

DataRobot runs performance testing during the model development process to evaluate model results and reliability. The validation, cross-validation, and holdout (if applicable) out-of-sample performance scores are presented below, as well as the number of observations for each partition. The performance metric used for this project was LogLoss and the project included a total of 90,678 observations. An asterisk (*) next to a score, whether validation or holdout, indicates that DataRobot used in-sample predictions to derive the score. (In-samples predictions are those that include data from the validation or holdout partitions due to sample size used to build the model.)

Scoring Type	Score (LogLoss)
cross_validation	0.3791*
holdout	0.2942*
validation	0.3801*

10 Bias and Fairness

The Bias & Fairness feature is not a legal compliance tool. Please carefully review the product Documentation to ensure that you fully understand the functionality of the feature before using it. DataRobot recommends that you take local legal advice where you wish to rely on the results of this feature for complying with any legal obligations. Product Documentation can be found here: https://app.datarobot.comdocs/modeling/special-workflows/b-and-f/index.html

DataRobot's Bias and Fairness testing identifies whether the model exhibits biased behavior towards any classes in the dataset's protected features, based on the selected definition of fairness. Protected features and the fairness metric are chosen before Autopilot is started. DataRobot also provides a workflow that guides you towards an appropriate definition of fairness for the specific use case.

DataRobot's Bias and Fairness feature includes two model-level insights:

- Per-Class Bias, which shows whether the model is treating certain protected groups differently as measured by the selected fairness metric. This identifies if there is biased behavior, and if so, how that bias manifests, but not why.
- Cross-Class Data Disparity, which shows how different protected classes differ in their data distribution. This offers deeper insight into why the model is treating groups differently.

Together, these insights can help identify potential mitigation strategies for bias in the dataset and model, such as improving data collection or data sampling for specific groups.

Bias and Fairness testing was used in this project. The selected protected features were clean_adm1. The favorable target outcome was No. The selected fairness metric was favorableAndUnfavorablePredictiveValueParity. The fairness threshold was set at 0.8.

Favorable Predictive Value & Unfavorable Predictive Value Parity (also known as Precision and NPV Parity) measure fairness by Equal Error, and it is best suited for cases when the target is severely unbalanced. These metrics measure whether your model disproportionally discovers favorable cases or omits unfavorable cases correctly for any particular group. It is calculated relative to the model's predictions–-the proportion of predicted favorable or predicted unfavorable cases that are classified correctly. Unlike True Favorable and Unfavorable Rate Parity, Favorable Predictive Value & Unfavorable Predictive Value Parity are calculated relative to the predicted class, rather than the ground truth, which allows it to capture more meaningful information when there are very few predictions for a certain class, such as in imbalanced datasets. Higher values for Favorable Predictive Value and Unfavorable Predictive Value mean that the model is more accurate for that class relative to that metric. It's important to measure both Favorable Predictive Value & Unfavorable Predictive Value Parity together, because poor performance in either one can lead to biased outcomes, often for different stakeholders. These metrics will help you investigate how your model balances both of these types of accuracy across your protected classes.

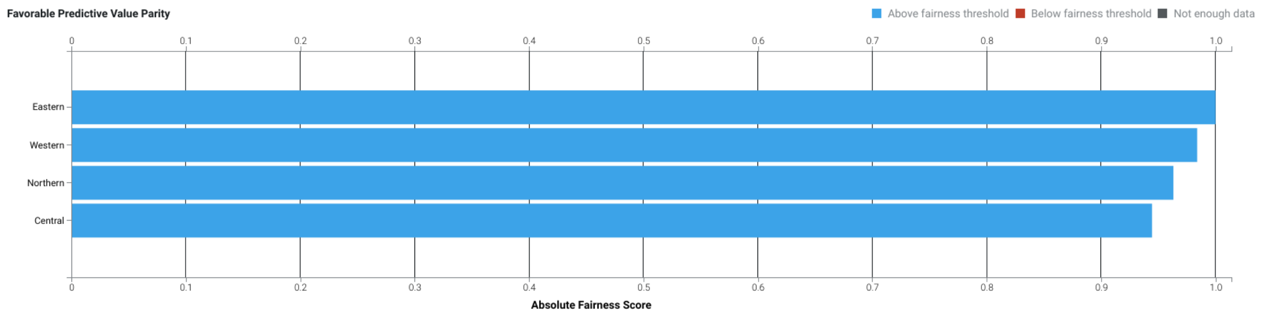
The Per-Class Bias graph shows whether the model exhibits biased behavior across protected features. The top fairness score across each protected class is scaled to 1.0, and the fairness scores for every other class are scaled relative to that value. If the fairness score for a class crosses the selected fairness threshold, the bar for that class is shown in red. If DataRobot used in-sample predictions to derive the model's performance scores (see **Overview of Model Results**), the fairness scores were calculated using in-sample validation data.

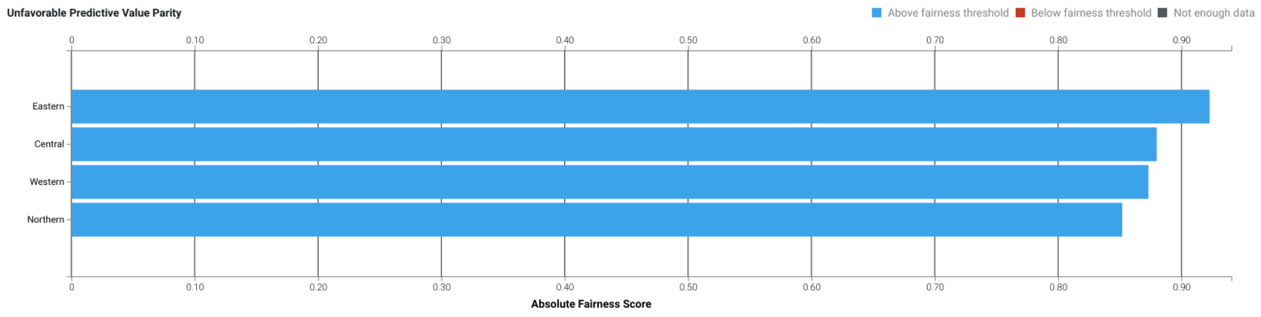
If there is not enough data for a class, its score is still calculated, but the bar for that class is shown in gray. The heuristic for whether a class does not have enough data is the following:

- If the class has <100 rows in the validation data, then it does not have enough data.
- If the class has between 100 and 1,000 rows in the validation data, but has fewer than <10% of the rows of the majority class, then it does not have enough data.
- If the class has >1,000 rows, then it has enough data.

The following figure is the Per-Class Bias graph for each protected feature in this project:

clean_adm1





The Cross-Class Data Disparity graph shows how different protected classes differ in their data distribution, in order to understand why the model treats each class differently based on the dataset. The X-axis depicts the feature importance of each feature in the dataset, while the Y-axis shows the Population Stability Index (PSI) for that feature compared across the two selected classes of the protected feature. The higher the feature importance, the more important that feature is to the model. The higher the PSI, the more differences there are for that feature across each of the two classes.