Package ‘aif360’

June 23, 2020

Type Package
Title Help Detect and Mitigate Bias in Machine Learning Models
Version 0.1.0
Description The 'AI Fairness 360' <https://aif360.mybluemix.net/> toolkit is an open-source library to help detect and mitigate bias in machine learning models. The AI Fairness 360 R package includes a comprehensive set of metrics for datasets and models to test for biases, explanations for these metrics, and algorithms to mitigate bias in datasets and models.
License Apache License (>= 2.0)
Encoding UTF-8
LazyData true
URL https://github.com/IBM/AIF360/tree/master/aif360/aif360-r
BugReports https://github.com/IBM/AIF360/issues
Imports reticulate, rstudioapi
RoxygenNote 7.1.0
Suggests testthat
NeedsCompilation no
Author Gabriela de Queiroz [aut], Stacey Ronaghan [aut], Saishruthi Swaminathan [aut, cre]
Maintainer Saishruthi Swaminathan <saishruthi.tn@ibm.com>
Repository CRAN
Date/Publication 2020-06-23 14:20:02 UTC

R topics documented:

adult_dataset ................................................. 2
adversarial_debiasing ........................................ 2
aif_dataset .................................................... 3
bank_dataset .................................................. 4
adult_dataset  
*Adult Census Income Dataset*

**Description**

Adult Census Income Dataset

**Usage**

```r
adult_dataset()
```

---

adversarial_debiasing  
*Adversarial Debiasing*

**Description**

Adversarial debiasing is an in-processing technique that learns a classifier to maximize prediction accuracy and simultaneously reduce an adversary's ability to determine the protected attribute from the predictions

**Usage**

```r
adversarial_debiasing(
  unprivileged_groups,
  privileged_groups,
  scope_name = "current",
  sess = tf$compat$V1$Session()
  seed = NULL,
  adversary_loss_weight = 0.1,
  num_epochs = 50,
  batch_size = 128,
  classifier_num_hidden_units = 200,
  debias = TRUE
)
```
Arguments

unprivileged_groups
   a list with two values: the column of the protected class and the value indicating representation for unprivileged group

privileged_groups
   a list with two values: the column of the protected class and the value indicating representation for privileged group

scope_name
   scope name for the tensorflow variables

nenviron
   tensorflow session

seed
   seed to make ‘predict’ repeatable.

adversary_loss_weight
   hyperparameter that chooses the strength of the adversarial loss.

num_epochs
   number of training epochs.

batch_size
   batch size.

classifier_num_hidden_units
   number of hidden units in the classifier model.

debias
   learn a classifier with or without debiasing.

Examples

```r
load_aif360_lib()
ad <- adult_dataset()
p <- list("race", 1)
u <- list("race", 0)
sess = tf$compat$v1$Session()

plain_model = adversarial_debiasing(privileged_groups = p,
unprivileged_groups = u,
scope_name='plain_classifier',
debias=FALSE,
seed=sess)

plain_model$fit(ad)
ad_nodebiasing <- plain_model$predict(ad)
```

Description

Function to create AIF compatible dataset.
Usage

\[
\text{aif\_dataset(data\_path, favor\_label, unfavor\_label, unprivileged\_protected\_attribute, privileged\_protected\_attribute, target\_column, protected\_attribute)}
\]

Arguments

- **data_path**: Path to the input CSV file or a R dataframe.
- **favor_label**: Label value which is considered favorable (i.e. “positive”).
- **unfavor_label**: Label value which is considered unfavorable (i.e. “negative”).
- **unprivileged\_protected\_attribute**: A unprotected attribute value which is considered privileged from a fairness perspective.
- **privileged\_protected\_attribute**: A protected attribute value which is considered privileged from a fairness perspective.
- **target\_column**: Name describing the label.
- **protected\_attribute**: A feature for which fairness is desired.

See Also

More about AIF binary dataset.

Examples

```r
load_aif360_lib()
# Input dataset
data <- data.frame("feat" = c(0,0,1,1,1,1,0,1,1,0), "label" = c(1,0,0,1,0,0,1,0,1,1))
# Create aif compatible input dataset
act <- aif360::aif\_dataset(data\_path = data, favor\_label=0, unfavor\_label=1, unprivileged\_protected\_attribute=0, privileged\_protected\_attribute=1, target\_column="label", protected\_attribute="feat")
```

bank\_dataset

**Bank Dataset**

Description

Bank Dataset

Usage

bank\_dataset()
**Binary Label Dataset Metric**

**Description**

Class for computing metrics on an aif360 compatible dataset with binary labels.

**Usage**

```r
binary_label_dataset_metric(data, privileged_groups, unprivileged_groups)
```

**Arguments**

- `data`: A aif360 compatible dataset.
- `privileged_groups`: Privileged groups. List containing privileged protected attribute name and value of the privileged protected attribute.
- `unprivileged_groups`: Unprivileged groups. List containing unprivileged protected attribute name and value of the unprivileged protected attribute.

**See Also**

Explore available binary label dataset metrics here

Available metrics are: `base_rate`, `consistency`, `disparate_impact`, `mean_difference`, `num_negatives`, `num_positives` and `statistical_parity_difference`.

**Examples**

```r
load_aif360_lib()
# Load the adult dataset
adult_dataset <- adult_dataset()

# Define the groups
privileged_groups <- list("race", 1)
unprivileged_groups <- list("race", 0)

# Metric for Binary Label Dataset
bm <- binary_label_dataset_metric(data = adult_dataset,
                                   privileged_groups = privileged_groups,
                                   unprivileged_groups = unprivileged_groups)

# Difference in mean outcomes between unprivileged and privileged groups
bm$mean_difference()
```
classification_metric  
*Classification Metric*

**Description**

Class for computing metrics based on two BinaryLabelDatasets. The first dataset is the original one and the second is the output of the classification transformer (or similar).

**Usage**

```python
classification_metric(dataset, classified_dataset, unprivileged_groups, privileged_groups)
```

**Arguments**

- `dataset` *(BinaryLabelDataset)* Dataset containing ground-truth labels
- `classified_dataset` *(BinaryLabelDataset)* Dataset containing predictions
- `unprivileged_groups` Unprivileged groups. List containing unprivileged protected attribute name and value of the unprivileged protected attribute.
- `privileged_groups` Privileged groups. List containing privileged protected attribute name and value of the privileged protected attribute.

**See Also**

Explore available classification metrics explanations here

Available metrics:

- accuracy
- average_abs_odds_difference
- average_odds_difference
- between_all_groups_coefficient_of_variation
- between_all_groups_generalized_entropy_index
- between_all_groups_theil_index
- between_group_coefficient_of_variation
- between_group_generalized_entropy_index
- between_group_theil_index
- binary_confusion_matrix
- coefficient_of_variation
- disparate_impact
- equal_opportunity_difference
• error_rate
• error_rate_difference
• error_rate_ratio
• false_discovery_rate
• false_discovery_rate_difference
• false_discovery_rate_ratio
• false_negative_rate
• false_negative_rate_difference
• false_negative_rate_ratio
• false_omission_rate
• false_omission_rate_difference
• false_omission_rate_ratio
• false_positive_rate
• false_positive_rate_difference
• false_positive_rate_ratio
• generalized_binary_confusion_matrix
• generalized_entropy_index
• generalized_false_negative_rate
• generalized_false_positive_rate
• generalized_true_negative_rate
• generalized_true_positive_rate
• negative_predictive_value
• num_false_negatives
• num_false_positives
• num_generalized_false_negatives
• num_generalized_false_positives
• num_generalized_true_negatives
• num_generalized_true_positives
• num_pred_negatives
• num_pred_positives
• num_true_negatives
• num_true_positives
• performance_measures
• positive_predictive_value
• power
• precision
• recall
- selection_rate
- sensitivity
- specificity
- statistical_parity_difference
- theil_index
- true_negative_rate
- true_positive_rate
- true_positive_rate_difference

Examples

load_aif360_lib()
# Input dataset
data <- data.frame("feat" = c(0,0,1,1,1,0,1,1,1,0), "label" = c(1,0,0,1,0,0,1,0,1,1))
# Create aif compatible input dataset
act <- aif360::aif_dataset(data_path = data, favor_label=0, unfavor_label=1,
unprivileged_protected_attribute=0,
privileged_protected_attribute=1,
target_column="label", protected_attribute="feat")
# Classified dataset
pred_data <- data.frame("feat" = c(0,0,1,1,1,0,1,1,1,0), "label" = c(1,0,1,1,0,1,0,0,1,1))
# Create aif compatible classified dataset
pred <- aif360::aif_dataset(data_path = pred_data, favor_label=0, unfavor_label=1,
unprivileged_protected_attribute=0,
privileged_protected_attribute=1,
target_column="label", protected_attribute="feat")
# Create an instance of classification metric
cm <- classification_metric(act, pred, list('feat', 1), list('feat', 0))
# Access metric functions
cm$accuracy()

compas_dataset	Compas Dataset

Description

Compas Dataset

Usage

compas_dataset()
**disparate_impact_remover**

*Disparate Impact Remover*

**Description**

Disparate impact remover is a preprocessing technique that edits feature values to increase group fairness while preserving rank-ordering within groups.

**Usage**

```r
disparate_impact_remover(repair_level, sensitive_attribute)
```

**Arguments**

- `repair_level`  
  Repair amount. 0.0 is no repair while 1.0 is full repair.
- `sensitive_attribute`  
  Single protected attribute with which to do repair.

**Examples**

```r
# An example using the Adult Dataset
load_aif360_lib()
ad <- adult_dataset()
p <- list("race", 1)
u <- list("race", 0)

di <- disparate_impact_remover(repair_level = 1.0, sensitive_attribute = "race")
rp <- di$fit_transform(ad)

di_2 <- disparate_impact_remover(repair_level = 0.8, sensitive_attribute = "race")
rp_2 <- di_2$fit_transform(ad)
```

---

**german_dataset**

*German Dataset*

**Description**

German Dataset

**Usage**

```r
german_dataset()
```
install_aif360 \textit{Install aif360 and its dependencies}

\textbf{Description}

Install aif360 and its dependencies

\textbf{Usage}

\begin{verbatim}
install_aif360(
  method = c("auto", "virtualenv", "conda"),
  conda = "auto",
  version = "default",
  envname = NULL,
  extra_packages = NULL,
  restart_session = TRUE,
  conda_python_version = "3.7",
  ...
)
\end{verbatim}

\textbf{Arguments}

- \texttt{method} \hspace{1cm} Installation method. By default, "auto" automatically finds a method that will work in the local environment. Change the default to force a specific installation method. Note that the "virtualenv" method is not available on Windows. Note also that since this command runs without privilege the "system" method is available only on Windows.

- \texttt{conda} \hspace{1cm} The path to a conda executable. Use "auto" to allow reticulate to automatically find an appropriate conda binary. See \textbf{Finding Conda} for more details.

- \texttt{version} \hspace{1cm} AIF360 version to install. Specify "default" to install the latest release.

- \texttt{envname} \hspace{1cm} Name of Python environment to install within

- \texttt{extra_packages} \hspace{1cm} Additional Python packages to install.

- \texttt{restart_session} \hspace{1cm} Restart R session after installing (note this will only occur within RStudio).

- \texttt{conda_python_version} \hspace{1cm} the python version installed in the created conda environment. Python 3.6 is installed by default.

- ... \hspace{1cm} other arguments passed to \texttt{[reticulate::conda_install()]} or \texttt{[reticulate::virtualenv_install()]}.
load_aif360_lib

Description

load functions

Usage

load_aif360_lib()

prejudice_remover

Prejudice Remover

Description

Prejudice remover is an in-processing technique that adds a discrimination-aware regularization term to the learning objective

Usage

prejudice_remover(eta=1.0, sensitive_attr='race', class_attr='income-per-year')

Arguments

eta fairness penalty parameter
sensitive_attr name of protected attribute
class_attr label name

Examples

# An example using the Adult Dataset
load_aif360_lib()
ad <- adult_dataset()
model <- prejudice_remover(class_attr = "income-per-year", sensitive_attr = "race")
model$fit(ad)
ad_pred <- model$predict(ad)
reject_option_classification

Reject option classification

Description

Reject option classification is a postprocessing technique that gives favorable outcomes to unprivileged groups and unfavorable outcomes to privileged groups in a confidence band around the decision boundary with the highest uncertainty.

Usage

```r
reject_option_classification(
  unprivileged_groups,
  privileged_groups,
  low_class_thresh = 0.01,
  high_class_thresh = 0.99,
  num_class_thresh = as.integer(100),
  num_ROC_margin = as.integer(50),
  metric_name = "Statistical parity difference",
  metric_ub = 0.05,
  metric_lb = -0.05
)
```

Arguments

- **unprivileged_groups**: A list representation for unprivileged group.
- **privileged_groups**: A list representation for privileged group.
- **low_class_thresh**: Smallest classification threshold to use in the optimization. Should be between 0. and 1.
- **high_class_thresh**: Highest classification threshold to use in the optimization. Should be between 0. and 1.
- **num_class_thresh**: Number of classification thresholds between low_class_thresh and high_class_thresh for the optimization search. Should be > 0.
- **num_ROC_margin**: Number of relevant ROC margins to be used in the optimization search. Should be > 0.
- **metric_name**: Name of the metric to use for the optimization. Allowed options are "Statistical parity difference", "Average odds difference", "Equal opportunity difference".
- **metric_ub**: Upper bound of constraint on the metric value
- **metric_lb**: Lower bound of constraint on the metric value
Examples

# Example with Adult Dataset
load_aif360_lib()
ad <- adult_dataset()
p <- list("race", 1)
u <- list("race", 0)

col_names <- c(ad$feature_names, "label")
ad_df <- data.frame(ad$features, ad$labels)
colnames(ad_df) <- col_names

lr <- glm(label ~ ., data=ad_df, family=binomial)
ad_prob <- predict(lr, ad_df)
ad_pred <- factor(ifelse(ad_prob > 0.5, 1, 0))
ad_df_pred <- data.frame(ad_df)
ad_df_pred$label <- as.character(ad_pred)
colnames(ad_df_pred) <- c(ad$feature_names, "label")
ad_ds <- aif_dataset(ad_df, target_column = "label", favor_label = 1, 
                     unfavor_label = 0, unprivileged_protected_attribute = 0, 
                     privileged_protected_attribute = 1, protected_attribute = "race")
ad_ds_pred <- aif_dataset(ad_df_pred, target_column = "label", favor_label = 1, 
                          unfavor_label = 0, unprivileged_protected_attribute = 0, 
                          privileged_protected_attribute = 1, protected_attribute = "race")

roc <- reject_option_classification(unprivileged_groups = u, 
                                    privileged_groups = p, 
                                    low_class_thresh = 0.01, 
                                    high_class_thresh = 0.99, 
                                    num_class_thresh = as.integer(100), 
                                    num_ROC_margin = as.integer(50), 
                                    metric_name = "Statistical parity difference", 
                                    metric_ub = 0.05, 
                                    metric_lb = -0.05)

roc <- roc$fit(ad_ds, ad_ds_pred)
ds_transformed_pred <- roc$predict(ad_ds_pred)

---

**Reweighing**

### Description

Reweighing is a preprocessing technique that weights the examples in each (group, label) combination differently to ensure fairness before classification.
Usage

`reweighing(unprivileged_groups, privileged_groups)`

Arguments

- `unprivileged_groups`: a list with two values: the column of the protected class and the value indicating representation for unprivileged group
- `privileged_groups`: a list with two values: the column of the protected class and the value indicating representation for privileged group

Examples

```r
# An example using the Adult Dataset
load_aif360_lib()
ad <- adult_dataset()
p <- list("race", 1)
u <- list("race", 0)
rw <- reweighing(u, p)
rw$fit(ad)
ad_transformed <- rw$transform(ad)
ad_fit_transformed <- rw$fit_transform(ad)
```
Index

adult_dataset, 2
adversarial_debiasing, 2
aif_dataset, 3
bank_dataset, 4
binary_label_dataset_metric, 5
classification_metric, 6
compas_dataset, 8
disparate_impact_remover, 9
german_dataset, 9
install_aif360, 10
load_aif360_lib, 11
prejudice_remover, 11
reject_option_classification, 12
reweighing, 13