SolasAI

Release

SolasAI LLC

Jun 13, 2023

Table of Contents

Chapter 1

Examples

1.1 SolasAI Disparity Introduction

1.1.1 Quick Start Summary

This example serves as an introductory text aiming to teach a user how to measure disparities with the SolasAI library. While the example in this notebook is based on the calculation of a single metric, the Adverse Impact Ratio (AIR), there are numerous other metrics available in SolasAI, as well as a generic metric interface for developing custom disparity metrics. Usage of the other metrics is discussed in other examples.

This notebook provides:

1. A short background on terms used throughout the SolasAI library.
2. An explanation of how to import and call functionality in the SolasAI disparity testing library.
3. An example of how to calculate an Adverse Impact Ratio using U.S. Home Mortgage Disclosure Act (HMDA) data.
4. An overview of how to make customized or formatted tables and charts using the SolasAI disparity testing library.

1.1.2 Background

Before jumping into the code, we provide a short explanation of several terms used in the code.

"Protected" and "Reference" Groups

In SolasAI's current list of curated metrics, there is an assumption that one will test whether some group(s) achieve outcomes that are *at least as good* as another group's outcomes. The first group(s), identified in SolasAI as `protected_groups`, typically consists of people who have been or are continuing to experience some form of discrimination or disadvantage. The outcomes of these groups are then compared to those of the `reference_groups`, which typically have not been subject to the same type of discrimination or disadvantages.

Outside SolasAI, there are numerous ways that these classifications are made and described. For example, often the word "protected" may be replaced with "minority", "disadvantaged", "test", or some other description, while "reference" may be replaced with "majority", "advantaged", "control", or some other description. Our use of the "protected" and "reference" categorization comes from our experience working in litigation and regulatory compliance in the United States. It is not meant to imply any particular value judgement regarding the classifications used.

A third attribute, `group_categories` is also used throughout the SolasAI library to delineate between different protected-reference group combinations. For example, `group_categories` might be race, sex, age, medical diagnoses, etc. By way of example, classifications most commonly used by

SolasAI's U.S. customers are:

`group_categories`	`protected_group`	`reference_group`
Race	Black	White
Race	Hispanic	White
Race	Asian	White
Sex / Gender	Female	Male
Age	Age >= X	Age < X

Outcomes, Labels, and Segments

SolasAI can be used to measure disparities created as the result of automated systems, semi-automated systems, or entirely subjective processes. Regardless of the use case, when the user calculates disparity, they will need to specify the `outcome` attribute. In some cases, this will be a binary (i.e., "Yes" or "No") outcome, such as whether a person was offered a job, loan, or sent a marketing offer. In other cases, it may be a continuous value, such as a model's probability of loan default, the amount of time it takes for a person to be promoted, or an employee's pay rate. Of the metrics SolasAI provides, some are appropriate for the binary case (e.g., `adverse_impact_ratio`), while others are appropriate for analyses of continuous values (e.g., `standardized_mean_difference`).

When measuring disparities that arise from the use of a model, and when the true outcome is known, a user can specify the `label` attribute. Certain metrics, such as the `residual_standardized_mean_difference`, require the label to be present because the disparity measurement incorporates the label.

Some metrics, including the `segmented_adverse_impact_ratio` perform analyses on subsets of the data and then aggregate the results. SolasAI refers to these subsets using the `segment` attribute. Examples of segments might be different store locations, job openings, or job types. Care should be used when deciding whether to incorporate segmentation into an analysis.

Statistical Significance and Practical Significance

SolasAI uses thresholds of statistical significance and "practical" significance to determine whether any potential disparities found are sufficiently large to warrant further review. The SolasAI does not provide guidance as to which standards are appropriate. What constitute appropriate and reasonable standards may be driven by regulatory and legal requirements, business decisions, or other factors. We suggest consulting with one's compliance department, legal advisors, or consultants such as BLDS, LLC (the consultancy from which SolasAI was founded) for advice.

That said, after one decides what thresholds to use, SolasAI allows users to test whether the disparity metrics exceed those thresholds. In the case of statistical significance, each metric uses a measurement of statistical significance that is appropriate or commonly used for that metric. As an example, in the case of the AIR, statistical significance is calculated using either a Fisher's Exact test or a Chi-Squared test (depending on the size of the data). For the SMD, a t-test is used. In certain cases, the user can specify either the test itself or certain attributes of the test. In United States legal and regulatory standards, a two-sided p-value less than 5% (or, equivilently, a one-sided p-value less than 2.5%) is generally considered statistically significant. SolasAI does not provide guidance on whether this standard is reasonable or appropriate for any particular use case.

Importantly, for SolasAI to identify that a result is "practically significant", it must *both* be found to be statistically significant and exceed the practical significance thresholds set. Most metrics in SolasAI provide the option to specify two thresholds. The first is a `percent_difference_threshold`. If a user sets a value greater than zero, then the raw difference in outcomes between the protected and reference groups must exceed a particular value before a result is considered practically significant. For example, if `percent_difference_threshold = 0.02` for the AIR, and we find that Black applicants receive offers 1% of the time, but White applicants receive them 2.5% of the time, then this disparity will not be significant because the difference in outcomes is only 1.5%, which is less thna the 2% threshold.

The second practical significance metric relates to the value of the metric itself. For example, when calling the `adverse_impact_ratio`, one sets the `air_threshold` to a particular value. In the example below, we use air_threshold=0.80. This means that only results that are statistically significantly different from parity *and* that have AIR values less than 0.80 will be considered practically significant.

1.1.3 Importing the Library and Data for Use in the Analysis

Below, we import the pandas and plotly libraries used to prepare and graph the data. SolasAI relies on the plotly library for graphing. The final line of code, pio.renderers.default = 'svg' is only necessary because this workbook is hosted in GitHub, which cannot render plotly graphics in their native format.

```
import pandas as pd
from pathlib import Path
```

Certain notebook environments have limited rendering functionality. Uncomment this cell as a potential workaround if plots are not displaying.

```
# import plotly.io as pio
# pio.renderers.default = "png"
```

It's preferable to explicitly and specifically handle warnings. For the purposes of this notebook, we will filter out all warnings.

```
from warnings import simplefilter
simplefilter("ignore")
```

SolasAI's disparity library is imported just like any standard Python package. Here, we import the library itself, which will be accessed as sd. Within sd, we will also access several other types of functions, including:

1. The interface functionality, sd.ui, which allows users to do things like create nicely formatted tables.
2. The SolasAI constants file, sd.const, which allows users to customize numerous settings including column names and plot headings.
3. Access to a set of utility functions, sd.util, which provide additional useful functionality.

```
import solas_disparity as sd
```

Data Preparation

The following code imports a sample of the 2018 Home Mortgage Disclosure Act (HMDA) data. The HMDA dataset includes information about nearly every home mortgage application in the United States. This dataset includes information about the mortgage itself, such as the loan term and APR; information about credit characteristics of the borrowers themselves, including the borrowers income and debt-to-income (DTI) ratio; and information about the home being purchased, such as its location and the value of the property. Importantly, it also includes information about each borrower's race, gender, and ethnicity. The data we are using is based only for applications where the borrower was approved for the loan.

```
df = pd.read_csv("hmda.csv.gz", index_col="id")
df.sample(random_state=161803, n=5)
```

We next specify the groups that we will use to test disparities. In SolasAI, each protected and reference group must be its own variable in the input data. These variable names are then included in the protected_group, reference_group, and group_categories lists that are used throughout the SolasAI library.

For example, if we are going to test for evidence of disparities between Black and White applicants, we must have one variable that identifies whether the person represented by the observation is Black and one variable that identifies whether the person is White. Importantly, missing values for these characteristics are generally allowed in SolasAI.

While more groups are available for analysis in the HMDA data, we limit the analysis in order to make the output more tractable. The categorization used in this example is as follows:

group_categories	protected_group	reference_group
Race	Black	White
Race	Native American	White
Race	Asian	White
Sex	Female	Male
Ethnicity	Hispanic	Non-Hispanic

The three lists must all be the same length, with each element of the list corresponding to the same comparison (e.g., the first element of the lists below have `Black`, `White` and `Race`, meaning that Black applicants are being compared to White applicants, which is a comparison by race. The fifth elements of each list are `Female`, `Male`, and `Sex`, which means that women are being compared to men, and the type of comparison is by Sex).

```
protected_groups = ["Black", "Asian", "Native American", "Hispanic", "Female"]
reference_groups = ["White", "White", "White", "Non-Hispanic", "Male"]
group_categories = ["Race", "Race", "Race", "Ethnicity", "Sex"]
```

While not demonstrated in this notebook, a key benefit to the SolasAI library is its ability to calculate disparities on groups where the characteristics are estimated, rather than known.

In this case, each person's probability estimates are put into the fields identifying group membership. For example, if the estimation procedure finds that a person has a 75% chance of being Black and a 25% chance of being White, then that person's Black and White variables would have values of 0.75 and 0.25, respectively.

A common example of this occurs in race and ethnicity estimation, where a person's home address and last name are used to calculate the probability that the person is Black or White (This is known as the Bayesian Improved Surname Geocoding ("BISG") method. See here for more detail).

1.1.4 Calculating the Adverse Impact Ratio on Prior Lending Decisions

Determining whether there is evidence of discrimination requires, to the extent possible, testing a model or process before it is put into use as well as testing it when it is being used in production (i.e., having an effective monitoring process). In this example, we focus on how an organization would monitor a process that is already in production. Here, we use the HMDA data to test whether there is evidence that members of the protected groups were less likely to receive low-priced loans than members of the reference groups.

This type of analysis can be performed on subjective decisions, such as loan officer decisions to underwrite a loan, or a manager deciding whom to promote. It can also be performed on the outcomes of automated decisioning processes, such as the use of a model to screen applicants, give job offers, or some other similar process. It can also be performed on a model's training or validation datasets prior to the model being used in production.

Below, we use the `sd.adverse_impact_ratio` function to calculate the AIR. More detail about the AIR can be found in the API documentation for solas_disparity.adverse_impact_ratio.

```
air = sd.adverse_impact_ratio(
    group_data=df,  # dataset containing the protected and reference group
information
    protected_groups=protected_groups,
    reference_groups=reference_groups,
    group_categories=group_categories,
    outcome=df["Low-Priced"],
    sample_weight=None,
    air_threshold=0.80,
    percent_difference_threshold=0,
)
```

1.1.5 Overview of SolasAI Disparity Objects with the AIR as an Example

Before jumping into the results of the analysis, below we discuss common elements of the SolasAI disparity objects.

Disparity Object Output

In a notebook, a user can display a formatted summary of results by displaying the results object using standard IPython notebook methods such as referencing the object in the last line of a cell or by calling the `display` function. This summary includes three elements:

1. The disparity card, which summarizes information about the inputs and results of the test run.
2. A summary table, which prints more detailed results (and is discussed below).
3. A Plotly graph of the AIR metric.

Two examples are shown below.

```
air
```

```
from IPython.display import display

display(air)
```

```
from solas_disparity import ui
ui.show(air.summary_table)
```

Disparity Summary Table Output

Nearly all the important information about the results of the analysis is contained in the `summary_table`. The discussion below describes how to access, format, and graph the information in the `summary_table`.

The table can be accessed as a pandas DataFrame by referencing the `summary_table` attribute of the results object:

```
air.summary_table
```

It can also be viewed as a styled Pandas dataframe by using the SolasAI `sd.ui.show` command.

```
sd.ui.show(air.summary_table)
```

The user can generate plots for specific columns of the summary table by using the `plot` method of the results object. Below, we show examples of plotting the `percent_favorable` and `air` values.

```
air.plot(column="Percent Favorable")
```

```
air.plot(column="AIR")
```

Results can also be exported. Examples of these commands are shown below.

```
output = Path(".output")
output.mkdir(exist_ok=True)
```

```
air.to_excel(file_path=output / "air_summary_table.xlsx")
```

1.1.6 Utility Functions

`utils.pgrg_ordered` returns a unique and ordered list of protected AND reference groups. This can be helpful when working with group data or with SolasAI results outside of SolasAI.

```
protected_and_reference_groups = sd.utils.pgrg_ordered(
    protected_groups=protected_groups,
    reference_groups=reference_groups,
)
protected_and_reference_groups
```

1.1.7 Customizing Output

A user can change column names used in all downstream results using the `solas_disparity.const` interface. This can be very helpful when customizing output for a particular use case. As an example, a lender might want to make the "Percent Favorable" column display as "Loans Underwritten" or "Accepted Applications", while an employer might want to make the "Percent Favorable" column be "Job Offers" or "Promotions."

Below is an example of how column names can be customized for the HMDA data use case.

```
sd.const.FAVORABLE = "Total Loan Offers"
sd.const.PERCENT_FAVORABLE = "Loan Offer Percent"
sd.const.OBSERVATIONS = "Obs. with Data"
sd.const.PERCENT_MISSING = "Pct Obs. Missing Data"
sd.const.PERCENT_DIFFERENCE_FAVORABLE = "Offer Percent Difference"
sd.const.AIR_VALUES = "Adverse Impact Ratio"
```

Formatting in the table can also be changed, as occurs in the code below. Here, `sd.ui.AUTO_FORMATTERS` is a dictionary that contains the formats for each of the attributes in the summary table. When changing the formats, one specifies the key of the dictionary as the attribute from the const file (e.g., "FAVORABLE" when changing `sd.const.FAVORABLE`), and the value as desired python formatter.

```
sd.ui.AUTO_FORMATTERS["TOTAL"] = "0,.0f"
sd.ui.AUTO_FORMATTERS["FAVORABLE"] = "0,.0f"
sd.ui.AUTO_FORMATTERS["P_VALUES"] = "0.1%"
sd.ui.AUTO_FORMATTERS["PERCENT_MISSING"] = "0.1%"
```

```
const_modified_air = sd.adverse_impact_ratio(
    group_data=df,
    protected_groups=protected_groups,
    reference_groups=reference_groups,
    group_categories=group_categories,
    outcome=df["Low-Priced"],
    sample_weight=None,
    air_threshold=0.80,
    percent_difference_threshold=0.0,
)
```

The summary table with new variable names and formatted values is printed below.

```
sd.ui.show(const_modified_air.summary_table)
```

1.2 SolasAI Disparity Calculations

```
import solas_disparity as sd
import pandas as pd
```

Certain notebook environments have limited rendering functionality. Uncomment this cell as a potential workaround if plots are not displaying.

```
# import plotly.io as pio
# pio.renderers.default = "png"
```

It's preferable to explicitly and specifically handle warnings. For the purposes of this notebook, we will filter out all warnings.

```
from warnings import simplefilter
simplefilter("ignore")
```

Some predictions were created using a tree model run on an HMDA dataset.

```
label = "Interest Rate"
data = pd.read_parquet("hmda_test.parquet")
```

Store commonly reused function arguments.

```
protected_groups = ["Black", "Asian", "Native American", "Hispanic", "Female"]
reference_groups = ["White", "White", "White", "Non-Hispanic", "Male"]
groups = sd.pgrg_ordered(
    protected_groups=protected_groups,
    reference_groups=reference_groups,
)
reused_arguments = dict(
    group_data=data[groups],
    protected_groups=protected_groups,
    reference_groups=reference_groups,
    group_categories=["Race", "Race", "Race", "Ethnicity", "Sex"],
    sample_weight=None,
)
binary_outcome = data["Prediction"] <= data["Prediction"].quantile(0.5)
binary_label = data[label] <= data[label].quantile(0.5)
```

1.2.1 Adverse Impact Ratio (AIR)

```
air = sd.adverse_impact_ratio(
    outcome=binary_outcome,
    air_threshold=0.8,
    percent_difference_threshold=0.0,
    **reused_arguments,
)
```

```
air
```

1.2.2 Standardized Mean Difference (SMD)

```
smd = sd.standardized_mean_difference(
    outcome=data["Prediction"],
    label=data[label],
    smd_threshold=30,
    lower_score_favorable=True,
    **reused_arguments,
)
```

```
smd
```

1.2.3 Adverse Impact Ratio by Quantile

```
airq = sd.adverse_impact_ratio_by_quantile(
    outcome=data["Prediction"],
    air_threshold=0.8,
    percent_difference_threshold=0.0,
    quantiles=[decile / 10 for decile in range(1, 11)],
    lower_score_favorable=True,
    **reused_arguments,
)
```

```
airq
```

1.2.4 Odds Ratio

```
odds_ratio = sd.odds_ratio(
    outcome=binary_outcome,
    odds_ratio_threshold=0.68,
    percent_difference_threshold=0.0,
    **reused_arguments,
)
```

```
odds_ratio
```

1.2.5 Categorical Adverse Impact Ratio

```
# Generate an example categorical outcome.
categorical_outcome = pd.qcut(data["Prediction"], q=[0.0, 0.25, 0.5, 0.75, 1.0])
categories = categorical_outcome.cat.categories.to_series()
categories = pd.Series(["Best", "Great", "Good", "Bad"], index=categories.index)
categorical_outcome.replace(categories.to_dict(), inplace=True)
```

```
cair = sd.categorical_adverse_impact_ratio(
    outcome=categorical_outcome,
    category_order=list(reversed(categories.tolist())),
    air_threshold=0.8,
    percent_difference_threshold=0.0,
    **reused_arguments,
)
```

```
cair
```

1.2.6 Residual Standardized Mean Difference

```
rsmd = sd.residual_standardized_mean_difference(
    prediction=data["Prediction"],
    label=data[label],
    residual_smd_threshold=30,
    lower_score_favorable=True,
    **reused_arguments,
)
```

```
rsmd
```

1.2.7 Segmented Adverse Impact Ratio

```
# Generate example income segments.
segments = pd.qcut(data["Income"], q=[0.0, 1 / 3, 2 / 3, 1.0])
categories = segments.cat.categories.to_series()
categories = pd.Series(
    ["Low Income", "Mid Income", "High Income"], index=categories.index
)
segments.replace(categories.to_dict(), inplace=True)
```

```
sair = sd.segmented_adverse_impact_ratio(
    outcome=binary_outcome,
    air_threshold=0.8,
    percent_difference_threshold=0.0,
    fdr_threshold=0.2,
    segment=segments,
    **reused_arguments,
)
```

```
sair
```

1.2.8 Custom Disparity Metric

We can recreate the AIR as an example of how custom disparity metrics can be used. Many more advanced disparity metrics can benefit from the framework and additional validation provided by the custom disparity metric interface.

```
# Define a function for calculating perecent favorable.
def percent_favorable(outcome, sample_weight):
    return (outcome.mul(sample_weight, axis=0)).sum(
        axis=0, min_count=1
    ) / sample_weight.sum(axis=0, min_count=1)
```

```
custom_air = sd.custom_disparity_metric(
    outcome=binary_outcome,
    metric=percent_favorable,

difference_calculation=sd.types.DifferenceCalculation.REFERENCE_MINUS_PROTECTED,
    difference_threshold=lambda value: value > 0.0,
    ratio_calculation=sd.types.RatioCalculation.PROTECTED_OVER_REFERENCE,
    ratio_threshold=lambda value: value < 1.0,
    statistical_significance_test=sd.types.StatSigTest.FISHERS_OR_CHI_SQUARED,
    **reused_arguments,
)

custom_air
```

Note that the values from the custom disparity metric summary correspond to those of the original AIR calculation.

```
pd.concat(
    (
        custom_air.summary_table["PERCENT FAVORABLE"],
        air.summary_table[sd.const.PERCENT_FAVORABLE],
    ),
    axis=1,
)
```

```
pd.concat(
    (
        custom_air.summary_table[sd.const.RATIO],
        air.summary_table[sd.const.AIR_VALUES],
    ),
    axis=1,
)
```

1.2.9 Confusion Matrix Metrics

SolasAI-provides ready-made implementations of confusion matrix metrics by wrapping around the custom disparity metric.

```
(
    sd.false_discovery_rate,
    sd.false_negative_rate,
    sd.false_positive_rate,
    sd.precision,
    sd.true_negative_rate,
    sd.true_positive_rate,
)
```

```
precision_arguments = reused_arguments.copy()
precision_arguments["group_data"] = precision_arguments["group_data"].fillna(0.0)
precision = sd.precision(
    outcome=binary_outcome,
    label=binary_label,
    ratio_threshold=1.0,
    difference_threshold=0.0,
    **precision_arguments,
)
precision
```

`sd.precision` is essentially a convience wrapper for the following call to the custom disparity metric.

```
def precision(y_true, y_pred, sample_weight):
    from sklearn.metrics import confusion_matrix

    tn, fp, fn, tp = confusion_matrix(
        y_true=y_true,
        y_pred=y_pred,
        sample_weight=sample_weight,
    ).ravel()

    return tp / (tp + fp)


sd.custom_disparity_metric(
    outcome=binary_outcome,
    metric=precision,
    label=binary_label,

difference_calculation=sd.types.DifferenceCalculation.REFERENCE_MINUS_PROTECTED,
    difference_threshold=lambda difference: difference > 0.0,
    ratio_calculation=sd.types.RatioCalculation.PROTECTED_OVER_REFERENCE,
    ratio_threshold=lambda ratio: ratio < 1.0,
    statistical_significance_test=sd.types.StatSigTest.BOOTSTRAPPING,
    p_value_threshold=0.05,
    **precision_arguments,
)
```

Additionally, statistical significance can be set to `None` for custom disparity metrics, causing statistical significance calculations to be skipped.

```
sd.custom_disparity_metric(
    outcome=binary_outcome,
    metric=precision,
    label=binary_label,

difference_calculation=sd.types.DifferenceCalculation.REFERENCE_MINUS_PROTECTED,
    difference_threshold=lambda difference: difference > 0.0,
    ratio_calculation=sd.types.RatioCalculation.PROTECTED_OVER_REFERENCE,
    ratio_threshold=lambda ratio: ratio < 1.0,
    statistical_significance_test=None,
    p_value_threshold=0.05,
    **precision_arguments,
)
```

1.3 SolasAI Disparity Plots

```python
import solas_disparity as sd
import pandas as pd
```

Certain notebook environments have limited rendering functionality. Uncomment this cell as a potential workaround if plots are not displaying.

```python
# import plotly.io as pio
# pio.renderers.default = "png"
```

It's preferable to explicitly and specifically handle warnings. For the purposes of this notebook, we will filter out all warnings.

```python
from warnings import simplefilter
simplefilter("ignore")
```

Some predictions have already been created using a tree model run on an HMDA dataset.

```python
label = "Interest Rate"
data = pd.read_parquet("hmda_test.parquet")
```

Store commonly reused function arguments.

```python
protected_groups = ["Black", "Asian", "Native American", "Hispanic", "Female"]
reference_groups = ["White", "White", "White", "Non-Hispanic", "Male"]
groups = sd.pgrg_ordered(
    protected_groups=protected_groups,
    reference_groups=reference_groups,
)
reused_arguments = dict(
    group_data=data[groups],
    protected_groups=protected_groups,
    reference_groups=reference_groups,
    group_categories=["Race", "Race", "Race", "Ethnicity", "Sex"],
    sample_weight=None,
)
binary_outcome = data["Prediction"] <= data["Prediction"].quantile(0.5)
binary_label = data[label] <= data[label].quantile(0.5)
```

1.3.1 Single-Level Plots

Certain disparity functions provide a result for each group. Their associated plots are single figures and are referred to as single-level plots.

Calculate Disparity

Let's use a result from the AIR function as an example for single-index plots.

```python
air = sd.adverse_impact_ratio(
    outcome=binary_outcome,
    air_threshold=0.8,
    percent_difference_threshold=0.0,
    **reused_arguments
)
```

Output Results

The default output for a disparity calculation result object includes a default plot.

```
air
```

The `.plot()` method on the result object returns the plotly figure directly.

```
figure = air.plot()
type(figure)
```

```
figure
```

In the case of AIR, the plot function also takes an `column` argument, allowing specification of a different column in the summary table to be plotted.

```
air.plot(column=sd.const.TOTAL)
```

The `.plot()` method is simply a convenience wrapper for the associated plot function in the `solas_disparity.plots` namespace. For further information, reference this plot function in rendered documention. To have stronger linting support, one can optionally call this function directly.

```
sd.plots.plot_adverse_impact_ratio(disparity=air)
```

1.3.2 Multi-Level Plots

Certain other disparity functions provide a result for each secondary level for each group.

Calculate Disparity

Use AIR by quantile as an example for multi-level plots.

```
airq = sd.adverse_impact_ratio_by_quantile(
    outcome=data["Prediction"],
    air_threshold=0.8,
    percent_difference_threshold=0.0,
    quantiles=[decile / 10 for decile in range(1, 11)],
    **reused_arguments,
)
```

Output Results

The default output for a disparity calculation result object includes a default plot. Note that a new subplot is created for each quantile.

```
airq
```

The `.plot()` method on the result object returns the plotly figure directly.

```
type(airq.plot())
```

The `.plot()` method also takes an optional argument `column` just like a single-index plot.

```
airq.plot(column=sd.const.PERCENT_DIFFERENCE_FAVORABLE)
```

A user can also specify a single group to extract a single by-level plot for.

```
airq.plot(group="Black")
```

```
airq.plot(group="Black", column=sd.const.PERCENT_DIFFERENCE_FAVORABLE)
```

`.plot()` also has a quantile argument to return a figure for a single quantile. The `quantile` argument is specific to AIR by quantile. For example, the equivalent argument for a categorical AIR calculation would be `category`.

```
airq.plot(quantile=0.1)
airq.plot(quantile=0.5)
```

Another argument exposed by multi-level plots is `separate`. It is used to separate a single plotly figure containing multiple subplots into a list of separate plotly figures for each level. It is convenience argument equivalent to calling `.plot()` with the `quantile` argument for every quantile.

```
airq_figures = airq.plot(separate=True)
type(airq_figures)
```

```
airq_figures[0]
```

```
airq_figures[4]
```

As with any other plot, the full documentation and typing support can be found in the `solas_disparity.plots` namespace.

```
sd.plots.plot_adverse_impact_ratio_by_quantile
```

1.3.3 More Plot Functionality

Since the figures returned by plot functions are plotly figures, reference the plotly documentation for more functionality. https://plotly.com/python-api-reference/generated/plotly.graph_objects.Figure.html#plotly.graph_objects.Figure

The `update_layout` method to modify overall attributes of the figure, including its height and width. https://plotly.com/python-api-reference/generated/plotly.graph_objects.Figure.html#plotly.graph_objects.Figure.update_layout

```
air.plot().update_layout(height=500, width=500)
```

Plots can be saved as images using the `write_image` method. Here's an example saving a plot as an svg file. https://plotly.com/python-api-reference/generated/plotly.graph_objects.Figure.html#plotly.graph_objects.Figure.write_image

```
air.plot().write_image("air.svg")
```

Or as a png…

```
air.plot().write_image("air.png")
```

The size of plot when being saved to an image can also be controlled without affecting the original figure object.

```
air.plot().write_image("air_resized.svg", height=800, width=1100)
```

Clean up files.

```
from pathlib import Path

to_clean = ["air.svg", "air_resized.svg", "air.png"]
for name in to_clean:
    if Path(name).exists():
        Path(name).unlink()
```

Chapter 2

Python API Reference

`solas_disparity`

2.1 solas_disparity

Functions

`adverse_impact_ratio`	Calculate the Adverse Impact Ratio (AIR) for a given set of protected and reference groups.
`adverse_impact_ratio_by_quantile`	Calculate the Adverse Impact Ratio for specified quantiles.
`categorical_adverse_impact_ratio`	Calculate the Adverse Impact Ratio for a set of favorability-ordinal categorical outcomes.
`custom_disparity_metric`	Provide a modular format that to create a custom disparity metric.
`false_discovery_rate`	This method calculates the False Discovery Rate (FDR) for a given set of protected and reference groups.
`false_negative_rate`	This method calculates the False Negative Rate (FNR) for a given set of protected and reference groups.
`false_positive_rate`	This method calculates the False Positive Rate (FPR) for a given set of protected and reference groups.
`odds_ratio`	Calculate the Odds Ratio for a given set of protected and reference groups.
`pgrg_ordered`	Create an ordered list of protected and reference groups.
`precision`	This method calculates the Precision for a given set of protected and reference groups.
`residual_standardized_mean_difference`	Calculate the Standardized Mean Difference of residuals for a given set of protected and reference groups.
`scoring_impact_ratio`	Calculate the scoring impact ratio, an impact ratio where the boolean outcome is whether an individual scored above the median.

`segmented_adverse_impact_ratio`	Calculates within-segment and cross-segment disparate impact and statistical significance statistics for dichotomous outcomes when it is appropriate to aggregate segment results to a single cross-segment measurement of disparity.
`selection_impact_ratio`	Calculate the selection impact ratio.
`standardized_mean_difference`	Calculate the Standardized Mean Difference (SMD) for a given set of protected and reference groups.
`true_negative_rate`	This method calculates the True Negative Rate (TNR) for a given set of protected and reference groups.
`true_positive_rate`	This method calculates the True Positve Rate (TPR) for a given set of protected and reference groups.

2.1.1 adverse_impact_ratio

`solas_disparity.`**`adverse_impact_ratio`**(...)

Calculate the Adverse Impact Ratio (AIR) for a given set of protected and reference groups.

AIR is defined as the percentage of favorable outcomes of the protected group divided by the percentage of favorable outcomes of the reference group. For example, if 10% of Black or African American applicants were to receive a loan offer, but 20% of Non-Hispanic White applicants were to receive a loan offer, the AIR is equal to 10% / 20% = 0.50.

\text{AIR}_\text{Protected Group} = \frac{\text{% Favorable Outcome}_\text{Protected Group}}{\text{% Favorable Outcome}_\text{Reference Group}}

Or, in terms of the confusion matrix:

\text{AIR}_\text{Protected Group} = \frac{\frac{P_{\text{Protected Group}}}{(P_{\text{Protected Group}} + N_{\text{Protected Group}})}}{\frac{P_{\text{Reference Group}}}{(P_{\text{Reference Group}} + N_{\text{Reference Group}})}}

Where `P` represents the positive outcomes (i.e., `TP + FP`) and `N` represents the negative outcomes (i.e., `TN + FN`). Importantly, for this implementation of the AIR, we consider `P` to be favorable **from the perspective of the person being scored by the model**.

In some academic literature, the AIR is called the "disparate impact" metric. While it is true that the AIR may be used as **a** measure of disparate impact, other metrics are used to measure disparate impact in the legal sense of the word. Additionally, the AIR can also be used to measure other forms of discrimination, such as disparate treatment.

An AIR is considered practically significant if the AIR is:

1. less than a chosen `air_threshold`,
2. statistically significantly different than parity,
3. AND if its percent difference greater than a chosen `percent_difference_threshold`.

Parameters

- **`group_data`** (*`pd.DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.

- **outcome** (*pd.Series*) – Boolean outcome series where a value of `1` is assumed to be favorable.
- **air_threshold** (*float*) – Adverse Impact Ratio threshold value.
- **percent_difference_threshold** (*float*) – Percent difference threshold value. For example, a 20% difference is input as `percent_difference_threshold=0.2`.
- **label** (*Optional[pd.Series], optional*) – Boolean label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **sample_weight** (*Optional[pd.Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **max_for_fishers** (*int, optional*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.
- **shortfall_method** (*Optional[types.ShortfallMethod], optional*) – Method used for shortfall calculation. Defaults to ShortfallMethod.TO_REFERENCE_MEAN.

Returns Object containing results of the disparity calculation.

Return type types.Disparity

2.1.2 adverse_impact_ratio_by_quantile

`solas_disparity.`**`adverse_impact_ratio_by_quantile`**`(...)`

Calculate the Adverse Impact Ratio for specified quantiles.

AIR is defined as the percentage of favorable outcomes of the protected group divided by the percentage of favorable outcomes of the reference group.

\text{AIR}_\text{Protected Group} = \frac{\text{% Favorable Outcome}_\text{Protected Group}}{\text{% Favorable Outcome}_\text{Reference Group}}

An AIR is considered practically significant if the AIR is:

1. less than a chosen `air_threshold`,
2. statistically significantly different than parity,
3. AND greater than a chosen `percent_difference_threshold`.

Parameters

- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Outcome series.
- **air_threshold** (*float*) – Adverse Impact Ratio threshold value.
- **percent_difference_threshold** (*float*) – Percent difference threshold value. For example, a 20% difference is input as `percent_difference_threshold=0.2`.
- **quantiles** (*List[float]*) – Set of quantiles at which the AIR will be calculated (e.g. `[0.2, 0.4, 0.6, 0.8, 1.0]`).
- **label** (*Optional[Series], optional*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **sample_weight** (*Optional[Series], optional*) – Sample weight series.

Has the same length as `group_data`. Defaults to None.

- **`max_for_fishers`** (*`int, optional`*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.
- **`lower_score_favorable`** (*`bool, optional`*) – Whether a lower value of `outcome` is favorable. Defaults to True.
- **`merge_bins`** (*`bool, optional`*) – Whether quantiles with same cutoff are merged into one. Defaults to True.

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.3 categorical_adverse_impact_ratio

`solas_disparity.`**`categorical_adverse_impact_ratio`**`( ... )`

Calculate the Adverse Impact Ratio for a set of favorability-ordinal categorical outcomes.

AIR is defined as the percentage of favorable outcomes of the protected group divided by the percentage of favorable outcomes of the reference group.

\text{AIR}_\text{Protected Group} = \frac{\text{% Favorable Outcome}_\text{Protected Group}}{\text{% Favorable Outcome}_\text{Reference Group}}

An AIR is considered practically significant if the AIR is:

1. less than a chosen `air_threshold`,
2. statistically significantly different than parity,
3. AND greater than a chosen `percent_difference_threshold`.

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series of elements of the set `category_order`.
- **`air_threshold`** (*`float`*) – Adverse Impact Ratio threshold value.
- **`percent_difference_threshold`** (*`float`*) – Percent difference threshold value. For example, a 20% difference is input as `percent_difference_threshold=0.2`.
- **`category_order`** (*`List[str]`*) – Series of outcome categories in ascending order of favorability (e.g. `["bad", "good", "great", "best"]`).
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`max_for_fishers`** (*`int, optional`*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.4 custom_disparity_metric

`solas_disparity.`**`custom_disparity_metric`** (…)

Provide a modular format that to create a custom disparity metric.

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series.
- **`metric`** (*`Callable[..., Union[int, float]]`*) – A function of an outcome, label, and/or sample weight that returns a scalar. Typical argument names such as those used by sklearn.metrics or SolasAI disparity testing functions.
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`difference_calculation`** (*`Optional[ DifferenceCalculation ], optional`*) – Method for calculating the difference between group metrics. Defaults to DifferenceCalculation.REFERENCE_MINUS_PROTECTED.
- **`difference_threshold`** (*`Optional[Callable[[Union[int, float]], bool]], optional`*) – A function of the difference between protected and reference group metrics that returns whether the difference is significant. This will be factored into practical significance if not `None`. Defaults to lambda difference:(difference < 0.0).
- **`ratio_calculation`** (*`Optional[ RatioCalculation ], optional`*) – Method for calculating the ratio between group metrics. Defaults to RatioCalculation.PROTECTED_OVER_REFERENCE.
- **`ratio_threshold`** (*`Optional[Callable[[Union[int, float]], bool]], optional`*) – A function of the ratio between protected and reference group metrics that returns whether the difference is significant. This will be factored into practical significance if not `None`. Defaults to None.
- **`statistical_significance_test`** (*`Optional[StatSigTest], optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **`p_value_threshold`** (*`float, optional`*) – P-value threshold for statistical significance. Defaults to 0.05.
- **`statistical_significance_arguments`** (*`Dict[str, Any], optional`*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.5 false_discovery_rate

`solas_disparity.`**`false_discovery_rate`** (...)

This method calculates the False Discovery Rate (FDR) for a given set of protected and reference groups.

False Discovery Rate (FDR) = False Positives / (False Positives + True Positives)

An TPR is considered "practically significant" if the FDR is:

1. higher than *ratio_threshold*.
2. statistically significantly different than parity, AND
3. lesser than *difference_threshold*.

Note Ratio is defined as (FDR of Reference Group) / (FDR of Protected Group)

Difference is defined as (FDR of Protected Group) - (FDR of Reference Group)

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series.
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`ratio_threshold`** (*`float`*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*`float`*) – Threshold at which a difference is considered significant.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`statistical_significance_test`** (*`Optional[StatSigTest], optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **`p_value_threshold`** (*`float, optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None. Defaults to 0.05.
- **`statistical_significance_arguments`** (*`Dict[str, Any], optional`*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.6 false_negative_rate

`solas_disparity.`**`false_negative_rate`**`( ... )`

This method calculates the False Negative Rate (FNR) for a given set of protected and reference groups.

Also know as Fall-out.

False Negative Rate (FNR) = False Negatives / (False Negatives + True Positives)

An TPR is considered "practically significant" if the FPR is:

1. higher than *ratio_threshold*.
2. statistically significantly different than parity, AND
3. lesser than *difference_threshold*.

Note Ratio is defined as (FNR of Reference Group) / (FNR of Protected Group)

Difference is defined as (FNR of Protected Group) - (FNR of Reference Group)

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series.
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`ratio_threshold`** (*`float`*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*`float`*) – Threshold at which a difference is considered significant.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`statistical_significance_test`** (*`Optional[StatSigTest], optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **`p_value_threshold`** (*`float, optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None. Defaults to 0.05.
- **`statistical_significance_arguments`** (*`Dict[str, Any], optional`*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.7 false_positive_rate

`solas_disparity.`**`false_positive_rate`**(...)

This method calculates the False Positive Rate (FPR) for a given set of protected and reference groups.

Also know as Fall-out.

False Positive Rate (FPR) = False Positives / (False Positives + True Negatives)

An TPR is considered "practically significant" if the FPR is:

1. higher than *ratio_threshold*.
2. statistically significantly different than parity, AND
3. lesser than *difference_threshold*.

Note Ratio is defined as (FPR of Reference Group) / (FPR of Protected Group)

Difference is defined as (FPR of Protected Group) - (FPR of Reference Group)

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series.
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`ratio_threshold`** (*`float`*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*`float`*) – Threshold at which a difference is considered significant.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`statistical_significance_test`** (*`Optional[StatSigTest], optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **`p_value_threshold`** (*`float, optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None. Defaults to 0.05.
- **`statistical_significance_arguments`** (*`Dict[str, Any], optional`*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.8 odds_ratio

`solas_disparity.`**`odds_ratio`**(...)

Calculate the Odds Ratio for a given set of protected and reference groups.

The odds of a favorable outcome is the probability of a favorable outcome divided by its complement.

\text{Favorable Outcome Odds} = \frac{P(\text{Favorable Outcome})}{P(\text{Favorable Outcome})^\complement}

The Odds Ratio is defined as the odds of a favorable outcome of the protected group divided by that of the reference group.

\text{Odds Ratio}_\text{Protected Group} = \frac{\text{Favorable Outcome Odds}_\text{Protected Group}}{\text{Favorable Outcome Odds}_\text{Reference Group}}

An Odds Ratio is considered practically significant if `lower_score_favorable=False` and the Odds Ratio is:

1. lesser than a chosen `odds_ratio_threshold`,
2. statistically significantly different than parity,
3. AND greater than a chosen `percent_difference_threshold`.

Alternatively, an Odds Ratio is considered practically significant if `lower_score_favorable=True` and the Odds Ratio is:

1. greater than a chosen `odds_ratio_threshold`,
2. statistically significantly different than parity,
3. AND greater than a chosen `percent_difference_threshold`.

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Boolean outcome series.
- **`odds_ratio_threshold`** (*`float`*) – Odds Ratio threshold value.
- **`percent_difference_threshold`** (*`float`*) – Percent difference threshold value. For example, a 20% difference is input as `percent_difference_threshold=0.2`.
- **`lower_score_favorable`** (*`bool, optional`*) – Whether a lower value of `outcome` (i.e. `0`) is favorable. Defaults to False.
- **`label`** (*`Optional[Series], optional`*) – Boolean label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`max_for_fishers`** (*`int, optional`*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.9 pgrg_ordered

`solas_disparity.`**`pgrg_ordered`**(...)

Create an ordered list of protected and reference groups.

Parameters

- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups associated with each protected group.
- **`group_categories`** (*`Optional[List[str]], optional`*) – Group categories associated with each protected group. If None, ordered group categories are returned alongside unique group names. Defaults to None.

Returns List of ordered protected and reference groups. May also include group categories if group_categories is set.

Return type List[str]

Examples

```
>>> protected_groups = ["black", "asian", "female", "hispanic"]
>>> reference_groups = ["white", "white", "male", "white"]
>>> pgrg_orderd(protected_groups, reference_groups)
["black", "asian", "hispanic", "white", "female", "male"]
```

2.1.10 precision

`solas_disparity.`**`precision`**(...)

This method calculates the Precision for a given set of protected and reference groups.

Also know as Precision.

Precision = True Positives / (True Positives + False Positives)

An TPR is considered "practically significant" if the Precision is:

1. less than a chosen *ratio_threshold*.
2. statistically significantly different than parity, AND
3. greater than *difference_threshold*.

Note Ratio is defined as (Precision of Protected Group) / (Precision of Reference Group)

Difference is defined as (Precision of Reference Group) - (Precision of Protected Group)

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series.
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`ratio_threshold`** (*`float`*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*`float`*) – Threshold at which a difference is

considered significant.

- **`sample_weight`** (*`Optional[Series]`, `optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`statistical_significance_test`** (*`Optional[StatSigTest]`, `optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **`p_value_threshold`** (*`float`, `optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None. Defaults to 0.05.
- **`statistical_significance_arguments`** (*`Dict[str, Any]`, `optional`*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.11 residual_standardized_mean_difference

`solas_disparity.`**`residual_standardized_mean_difference`** (...)

Calculate the Standardized Mean Difference of residuals for a given set of protected and reference groups.
A residual is the label value minus the predicted value.
\text{Residual} = y - \hat{y}
The Residual SMD is the mean residual of a protected group minus the mean residual of a reference group all divided by the standard deviation of residuals.
\text{Residual SMD}_\text{Protected Group} = \frac{\text{Mean Residual}_{\text{Protected Group}} - \text{Mean Residual}_{\text{Reference Group}}}{s}

A Residual SMD is considered practically significant if `lower_score_favorable=True` and the Residual SMD is:

1. greater in magnitude than a chosen `residual_smd_threshold`
2. AND statistically significantly different than zero.

Alternatively, a Residual SMD is considered practically significant if `lower_score_favorable=False` and the Residual SMD is:

1. lesser in magnitude than a chosen `residual_smd_threshold`
2. AND statistically significantly different than zero.

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups.
- **`group_categories`** (*`List[str]`*) – List of group categories.
- **`prediction`** – (Series): Predictions series.
- **`label`** (*`Series`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`residual_smd_threshold`** (*`float`*) – Residual Standardized Mean Difference threshold value.
- **`lower_score_favorable`** (*`bool`, `optional`*) – Whether a lower value of `label - prediction` is favorable. Defaults to True.

- **`sample_weight`** – Optional[Series]: Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`residual_smd_denominator`** (*`Union[ResidualSMDDenominator, str], optional`*) – Residual Standardized Mean Difference denominator calculation. Defaults to ResidualSMDDenominator.POPULATION.

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.12 scoring_impact_ratio

`solas_disparity.`**`scoring_impact_ratio`**(...)

Calculate the scoring impact ratio, an impact ratio where the boolean outcome is whether an individual scored above the median.

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`race_ethnicity_groups`** (*`List[str]`*) – A list of race/ethnicity groups corresponding to individual columns in group_data (e.g. *["Hispanic or Latino", "White", "Black or African American", "Native Hawaiian or Pacific Islander", "Asian", "Native American or Alaska Native", "Two or More Races"]*).
- **`gender_groups`** (*`List[str]`*) – A list of gender groups corresponding to individual columns in group_data (e.g. *["Male", "Female"]*).
- **`outcome`** (*`Series`*) – Outcome series of scores.
- **`ratio_threshold`** (*`float`*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*`float`*) – Threshold at which a difference is considered significant.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`max_for_fishers`** (*`int, optional`*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.
- **`shortfall_method`** (*`Optional[types.ShortfallMethod], optional`*) – Method used for shortfall calculation. Defaults to ShortfallMethod.TO_REFERENCE_MEAN.
- **`drop_small_groups`** (*`Optional[bool], optional`*) – Whether to separate and return a table of groups that comprise less than 2% of individuals. Defaults to True.

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.13 segmented_adverse_impact_ratio

`solas_disparity.`**`segmented_adverse_impact_ratio`**(...)

Calculates within-segment and cross-segment disparate impact and statistical significance statistics for dichotomous outcomes when it is appropriate to aggregate segment results to a single cross-segment measurement of disparity. A `segment` might be a market, location, type of job, or some other discrete unit.

The Adverse Impact Ratio (AIR) is calculated within each segment and cross-segment.

AIR is defined as the percentage of favorable outcomes of the protected group divided by the percentage of favorable outcomes of the reference group.

\text{AIR}_\text{Protected Group} = \frac{\text{% Favorable Outcome}_\text{Protected

Group}}{\text{% Favorable Outcome}_\text{Reference Group}}

The cross-segment AIR attempts to control for the fact that the protected class may be distributed across segments differently from the reference class. In order to account for this, this method keeps the distribution of protected group members constant across the segments, but reallocates the reference group members according to the protected group members' distribution. It does keep the protected group and reference group favorable outcome rates constant within each segment. In this way, the method is able to adjust for different distributions of class members by segment, but test whether the overall acceptance rates differ across classes.

Two cross-segment tests for statistical significance are performed: the Cochran-Mantel-Haenszel, CMH, test, which tests whether all segment-level odds ratios are equal to 1.0; and the Breslow-Day test, which tests whether segment level odds ratios are equal to each other.

When one can reject the Breslow-Day test, the p-values from the segment level tests (i.e., the Fisher's Exact or Chi-Squared tests) are assessed against critical values derived from the Benjamani-Hochberg Procedure for multiple testing, rather than the standard 2-tailed 5% rule.

When one cannot reject the Breslow-Day test, then the CMH practical significance results can be applied to within-segment practical significance. If `overwrite_segment_results=True`, then the within-segment practical significance results are overwritten with the CMH practical significance results. By setting `overwrite_segment_results=False`, within-segment results will not be overwritten. For the case where group count is zero in a segment, the practical significance will remain `""` and not be overwritten.

Parameters

- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Boolean outcome series where a value of `1` is assumed to be favorable.
- **air_threshold** (*float*) – Adverse Impact Ratio threshold value.
- **percent_difference_threshold** (*float*) – Percent difference threshold value. For example, a 20% difference is input as `percent_difference_threshold=0.2`.
- **fdr_threshold** (*float*) – False discovery rate (fdr) threshold value used in calculating whether segment-level results are statistically significant using the Benjamani-Hochberg Procedure.
- **segment** (*Series*) – Segment series.
- **label** (*Optional[Series], optional*) – Boolean label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **max_for_fishers** (*int, optional*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.
- **shift_zeros** (*bool, optional*) – If `True`, if any cell count in a contingency table used in the CMH and Breslow-Day tests is zero, then 0.5 is added to all values in the contingency table so an odds ratio is able to be calculated. Note: In large sample sizes, this correction will not make a significant difference. In small sample sizes, this correction has the potential to impact the significance determination. Defaults to True.

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.14 selection_impact_ratio

`solas_disparity.`**`selection_impact_ratio`**(...)

Calculate the selection impact ratio.

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`race_ethnicity_groups`** (*`List[str]`*) – A list of race/ethnicity groups corresponding to individual columns in group_data (e.g. *["Hispanic or Latino", "White", "Black or African American", "Native Hawaiian or Pacific Islander", "Asian", "Native American or Alaska Native", "Two or More Races"]*).
- **`gender_groups`** (*`List[str]`*) – A list of gender groups corresponding to individual columns in group_data (e.g. *["Male", "Female"]*).
- **`outcome`** (*`Series`*) – Boolean outcome series.
- **`ratio_threshold`** (*`float`*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*`float`*) – Threshold at which a difference is considered significant.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`max_for_fishers`** (*`int, optional`*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.
- **`shortfall_method`** (*`Optional[types.ShortfallMethod], optional`*) – Method used for shortfall calculation. Defaults to ShortfallMethod.TO_REFERENCE_MEAN.
- **`drop_small_groups`** (*`Optional[bool], optional`*) – Whether to separate and return a table of groups that comprise less than 2% of individuals. Defaults to True.

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.15 standardized_mean_difference

`solas_disparity.`**`standardized_mean_difference`**(...)

Calculate the Standardized Mean Difference (SMD) for a given set of protected and reference groups.

The SMD is the mean outcome of a protected group minus the mean outcome of a reference group all divided by the standard deviation of outcomes.

\text{SMD}_\text{Protected Group} = \frac{\text{Mean Outcome}_{\text{Protected Group}} - \text{Mean Outcome}_{\text{Reference Group}}}{s}

An SMD is considered practically significant if `lower_score_favorable=True` and the SMD is:

1. greater than a chosen `smd_threshold`
2. AND statistically significantly different than zero.

Alternatively, an SMD is considered practically significant if `lower_score_favorable=False` and the SMD is:

1. lesser than a chosen `smd_threshold`
2. AND statistically significantly different than zero.

Parameters

- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Outcome series.
- **smd_threshold** (*float*) – Standardized Mean Difference threshold value.
- **lower_score_favorable** (*bool, optional*) – Whether a lower value of `outcome` is favorable. Defaults to True.
- **label** (*Optional[Series], optional*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **smd_denominator** (*SMDDenominator, optional*) – Standardized Mean Difference denominator calculation. Defaults to SMDDenominator.POPULATION.

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.16 true_negative_rate

`solas_disparity.`**`true_negative_rate`**`( ... )`

This method calculates the True Negative Rate (TNR) for a given set of protected and reference groups.

Also know as Specificity or Selectivity.

True Negative Rate (TNR) = True Negatives / (True Negatives + False Positives)

An TPR is considered "practically significant" if the TNR is:

1. less than a chosen *ratio_threshold*.
2. statistically significantly different than parity, AND
3. greater than *difference_threshold*.

Note Ratio is defined as (TNR of Protected Group) / (TNR of Reference Group)

Difference is defined as (TNR of Reference Group) - (TNR of Protected Group)

Parameters

- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Outcome series.
- **label** (*Optional[Series], optional*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.

- **ratio_threshold** (*float*) – Threshold at which a ratio is considered significant.
- **difference_threshold** (*float*) – Threshold at which a difference is considered significant.
- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **statistical_significance_test** (*Optional[StatSigTest], optional*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **p_value_threshold** (*float, optional*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None. Defaults to 0.05.
- **statistical_significance_arguments** (*Dict[str, Any], optional*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.17 true_positive_rate

`solas_disparity.`**`true_positive_rate`**`( ... )`

This method calculates the True Positve Rate (TPR) for a given set of protected and reference groups.

Also know as Sensitivity, Recall or Hit Rate.

True Positive Rate (TPR) = True Positives / (True Positives + False Negatives)

An TPR is considered "practically significant" if the TPR is:

1. less than a chosen *ratio_threshold*.
2. statistically significantly different than parity, AND
3. greater than *difference_threshold*.

Note Ratio is defined as (TPR of Protected Group) / (TPR of Reference Group)

Difference is defined as (TPR of Reference Group) - (TPR of Protected Group)

Parameters

- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Outcome series.
- **label** (*Optional[Series], optional*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **ratio_threshold** (*float*) – Threshold at which a ratio is considered significant.
- **difference_threshold** (*float*) – Threshold at which a difference is

considered significant.

- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **statistical_significance_test** (*Optional[StatSigTest], optional*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **p_value_threshold** (*float, optional*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None. Defaults to 0.05.
- **statistical_significance_arguments** (*Dict[str, Any], optional*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

Modules

`solas_disparity.const`	Constants used throughout solas-disparity library.
`solas_disparity.disparity`	
`solas_disparity.plots`	
`solas_disparity.statistical_significance`	
`solas_disparity.types`	
`solas_disparity.utils`	

2.1.18 solas_disparity.const

Constants used throughout solas-disparity library.

2.1.19 solas_disparity.disparity

Functions

`adverse_impact_ratio`	Calculate the Adverse Impact Ratio (AIR) for a given set of protected and reference groups.
`adverse_impact_ratio_by_quantile`	Calculate the Adverse Impact Ratio for specified quantiles.
`categorical_adverse_impact_ratio`	Calculate the Adverse Impact Ratio for a set of favorability-ordinal categorical outcomes.
`custom_disparity_metric`	Provide a modular format that to create a custom disparity metric.
`false_discovery_rate`	This method calculates the False Discovery Rate (FDR) for a given set of protected and reference groups.
`false_negative_rate`	This method calculates the False Negative Rate (FNR) for a given set of protected and reference groups.
`false_positive_rate`	This method calculates the False Positive Rate (FPR) for a given set of protected and reference groups.
`odds_ratio`	Calculate the Odds Ratio for a given set of protected and reference groups.

`precision`	This method calculates the Precision for a given set of protected and reference groups.
`residual_standardized_mean_difference`	Calculate the Standardized Mean Difference of residuals for a given set of protected and reference groups.
`scoring_impact_ratio`	Calculate the scoring impact ratio, an impact ratio where the boolean outcome is whether an individual scored above the median.
`segmented_adverse_impact_ratio`	Calculates within-segment and cross-segment disparate impact and statistical significance statistics for dichotomous outcomes when it is appropriate to aggregate segment results to a single cross-segment measurement of disparity.
`selection_impact_ratio`	Calculate the selection impact ratio.
`standardized_mean_difference`	Calculate the Standardized Mean Difference (SMD) for a given set of protected and reference groups.
`true_negative_rate`	This method calculates the True Negative Rate (TNR) for a given set of protected and reference groups.
`true_positive_rate`	This method calculates the True Positve Rate (TPR) for a given set of protected and reference groups.

adverse_impact_ratio

`solas_disparity.disparity.`**`adverse_impact_ratio`**(...)

Calculate the Adverse Impact Ratio (AIR) for a given set of protected and reference groups.

AIR is defined as the percentage of favorable outcomes of the protected group divided by the percentage of favorable outcomes of the reference group. For example, if 10% of Black or African American applicants were to receive a loan offer, but 20% of Non-Hispanic White applicants were to receive a loan offer, the AIR is equal to 10% / 20% = 0.50.

\text{AIR}_\text{Protected Group} = \frac{\text{% Favorable Outcome}_\text{Protected Group}}{\text{% Favorable Outcome}_\text{Reference Group}}

Or, in terms of the confusion matrix:

\text{AIR}_\text{Protected Group} = \frac{\frac{P_{\text{Protected Group}}}{(P_{\text{Protected Group}} + N_{\text{Protected Group}})}}{\frac{P_{\text{Reference Group}}}{(P_{\text{Reference Group}} + N_{\text{Reference Group}})}}

Where `P` represents the positive outcomes (i.e., `TP` + `FP`) and `N` represents the negative outcomes (i.e., `TN` + `FN`). Importantly, for this implementation of the AIR, we consider `P` to be favorable **from the perspective of the person being scored by the model**.

In some academic literature, the AIR is called the "disparate impact" metric. While it is true that the AIR may be used as **a** measure of disparate impact, other metrics are used to measure disparate impact in the legal sense of the word. Additionally, the AIR can also be used to measure other forms of discrimination, such as disparate treatment.

An AIR is considered practically significant if the AIR is:

1. less than a chosen `air_threshold`,
2. statistically significantly different than parity,
3. AND if its percent difference greater than a chosen `percent_difference_threshold`.

Parameters
- **`group_data`** (*`pd.DataFrame`*) – Dataframe containing columns for group data.

- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*pd.Series*) – Boolean outcome series where a value of `1` is assumed to be favorable.
- **air_threshold** (*float*) – Adverse Impact Ratio threshold value.
- **percent_difference_threshold** (*float*) – Percent difference threshold value. For example, a 20% difference is input as `percent_difference_threshold=0.2`.
- **label** (*Optional[pd.Series], optional*) – Boolean label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **sample_weight** (*Optional[pd.Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **max_for_fishers** (*int, optional*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.
- **shortfall_method** (*Optional[types.ShortfallMethod], optional*) – Method used for shortfall calculation. Defaults to ShortfallMethod.TO_REFERENCE_MEAN.

Returns Object containing results of the disparity calculation.

Return type types.Disparity

adverse_impact_ratio_by_quantile

`solas_disparity.disparity.`**`adverse_impact_ratio_by_quantile`**`(...)`

Calculate the Adverse Impact Ratio for specified quantiles.

AIR is defined as the percentage of favorable outcomes of the protected group divided by the percentage of favorable outcomes of the reference group.

\text{AIR}_\text{Protected Group} = \frac{\text{% Favorable Outcome}_\text{Protected Group}}{\text{% Favorable Outcome}_\text{Reference Group}}

An AIR is considered practically significant if the AIR is:

1. less than a chosen `air_threshold`,
2. statistically significantly different than parity,
3. AND greater than a chosen `percent_difference_threshold`.

Parameters

- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Outcome series.
- **air_threshold** (*float*) – Adverse Impact Ratio threshold value.
- **percent_difference_threshold** (*float*) – Percent difference threshold value. For example, a 20% difference is input as

`percent_difference_threshold=0.2`.

- **`quantiles`** (*`List[float]`*) – Set of quantiles at which the AIR will be calculated (e.g. `[0.2, 0.4, 0.6, 0.8, 1.0]`).
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`max_for_fishers`** (*`int, optional`*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.
- **`lower_score_favorable`** (*`bool, optional`*) – Whether a lower value of `outcome` is favorable. Defaults to True.
- **`merge_bins`** (*`bool, optional`*) – Whether quantiles with same cutoff are merged into one. Defaults to True.

Returns Object containing results of the disparity calculation.

Return type Disparity

categorical_adverse_impact_ratio

`solas_disparity.disparity.`**`categorical_adverse_impact_ratio`**(...)

Calculate the Adverse Impact Ratio for a set of favorability-ordinal categorical outcomes.

AIR is defined as the percentage of favorable outcomes of the protected group divided by the percentage of favorable outcomes of the reference group.

\text{AIR}_\text{Protected Group} = \frac{\text{% Favorable Outcome}_\text{Protected Group}}{\text{% Favorable Outcome}_\text{Reference Group}}

An AIR is considered practically significant if the AIR is:

1. less than a chosen `air_threshold`,
2. statistically significantly different than parity,
3. AND greater than a chosen `percent_difference_threshold`.

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series of elements of the set `category_order`.
- **`air_threshold`** (*`float`*) – Adverse Impact Ratio threshold value.
- **`percent_difference_threshold`** (*`float`*) – Percent difference threshold value. For example, a 20% difference is input as `percent_difference_threshold=0.2`.
- **`category_order`** (*`List[str]`*) – Series of outcome categories in ascending order of favorability (e.g. `["bad", "good", "great", "best"]`).
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.

- **max_for_fishers** (*int, optional*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.

Returns Object containing results of the disparity calculation.

Return type Disparity

custom_disparity_metric

`solas_disparity.disparity.`**`custom_disparity_metric`** (...)

Provide a modular format that to create a custom disparity metric.

Parameters
- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Outcome series.
- **metric** (*Callable[..., Union[int, float]]*) – A function of an outcome, label, and/or sample weight that returns a scalar. Typical argument names such as those used by sklearn.metrics or SolasAI disparity testing functions.
- **label** (*Optional[Series], optional*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **difference_calculation** (*Optional[DifferenceCalculation], optional*) – Method for calculating the difference between group metrics. Defaults to DifferenceCalculation.REFERENCE_MINUS_PROTECTED.
- **difference_threshold** (*Optional[Callable[[Union[int, float]], bool]], optional*) – A function of the difference between protected and reference group metrics that returns whether the difference is significant. This will be factored into practical significance if not `None`. Defaults to lambda difference:(difference < 0.0).
- **ratio_calculation** (*Optional[RatioCalculation], optional*) – Method for calculating the ratio between group metrics. Defaults to RatioCalculation.PROTECTED_OVER_REFERENCE.
- **ratio_threshold** (*Optional[Callable[[Union[int, float]], bool]], optional*) – A function of the ratio between protected and reference group metrics that returns whether the difference is significant. This will be factored into practical significance if not `None`. Defaults to None.
- **statistical_significance_test** (*Optional[StatSigTest], optional*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **p_value_threshold** (*float, optional*) – P-value threshold for statistical significance. Defaults to 0.05.
- **statistical_significance_arguments** (*Dict[str, Any], optional*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

false_discovery_rate

`solas_disparity.disparity.`**`false_discovery_rate`**(...)

This method calculates the False Discovery Rate (FDR) for a given set of protected and reference groups.

False Discovery Rate (FDR) = False Positives / (False Positives + True Positives)

An TPR is considered "practically significant" if the FDR is:

1. higher than *ratio_threshold*.
2. statistically significantly different than parity, AND
3. lesser than *difference_threshold*.

Note Ratio is defined as (FDR of Reference Group) / (FDR of Protected Group)

Difference is defined as (FDR of Protected Group) - (FDR of Reference Group)

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series.
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`ratio_threshold`** (*`float`*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*`float`*) – Threshold at which a difference is considered significant.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`statistical_significance_test`** (*`Optional[StatSigTest], optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **`p_value_threshold`** (*`float, optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None. Defaults to 0.05.
- **`statistical_significance_arguments`** (*`Dict[str, Any], optional`*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

false_negative_rate

`solas_disparity.disparity.`**`false_negative_rate`** (...)

This method calculates the False Negative Rate (FNR) for a given set of protected and reference groups.

Also know as Fall-out.

False Negative Rate (FNR) = False Negatives / (False Negatives + True Positives)

An TPR is considered "practically significant" if the FPR is:

1. higher than *ratio_threshold*.
2. statistically significantly different than parity, AND
3. lesser than *difference_threshold*.

Note Ratio is defined as (FNR of Reference Group) / (FNR of Protected Group)

Difference is defined as (FNR of Protected Group) - (FNR of Reference Group)

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series.
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`ratio_threshold`** (*`float`*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*`float`*) – Threshold at which a difference is considered significant.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`statistical_significance_test`** (*`Optional[StatSigTest], optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **`p_value_threshold`** (*`float, optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None. Defaults to 0.05.
- **`statistical_significance_arguments`** (*`Dict[str, Any], optional`*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

false_positive_rate

`solas_disparity.disparity.`**`false_positive_rate`** (...)

This method calculates the False Positive Rate (FPR) for a given set of protected and reference groups.

Also know as Fall-out.

False Positive Rate (FPR) = False Positives / (False Positives + True Negatives)

An TPR is considered "practically significant" if the FPR is:

1. higher than *ratio_threshold*.
2. statistically significantly different than parity, AND
3. lesser than *difference_threshold*.

Note Ratio is defined as (FPR of Reference Group) / (FPR of Protected Group)

Difference is defined as (FPR of Protected Group) - (FPR of Reference Group)

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series.
- **`label`** (*`Optional[Series]`*, *`optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`ratio_threshold`** (*`float`*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*`float`*) – Threshold at which a difference is considered significant.
- **`sample_weight`** (*`Optional[Series]`*, *`optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`statistical_significance_test`** (*`Optional[StatSigTest]`*, *`optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **`p_value_threshold`** (*`float`*, *`optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None. Defaults to 0.05.
- **`statistical_significance_arguments`** (*`Dict[str, Any]`*, *`optional`*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

odds_ratio

`solas_disparity.disparity.`**`odds_ratio`** (...)

Calculate the Odds Ratio for a given set of protected and reference groups.

The odds of a favorable outcome is the probability of a favorable outcome divided by its complement.
\text{Favorable Outcome Odds} = \frac{P(\text{Favorable Outcome})}{P(\text{Favorable Outcome})^\complement}
The Odds Ratio is defined as the odds of a favorable outcome of the protected group divided by that of the reference group.
\text{Odds Ratio}_\text{Protected Group} = \frac{\text{Favorable Outcome Odds}_\text{Protected Group}}{\text{Favorable Outcome Odds}_\text{Reference Group}}

An Odds Ratio is considered practically significant if `lower_score_favorable=False` and the Odds Ratio is:

1. lesser than a chosen `odds_ratio_threshold`,
2. statistically significantly different than parity,
3. AND greater than a chosen `percent_difference_threshold`.

Alternatively, an Odds Ratio is considered practically significant if `lower_score_favorable=True` and the Odds Ratio is:

1. greater than a chosen `odds_ratio_threshold`,
2. statistically significantly different than parity,
3. AND greater than a chosen `percent_difference_threshold`.

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Boolean outcome series.
- **`odds_ratio_threshold`** (*`float`*) – Odds Ratio threshold value.
- **`percent_difference_threshold`** (*`float`*) – Percent difference threshold value. For example, a 20% difference is input as `percent_difference_threshold=0.2`.
- **`lower_score_favorable`** (*`bool, optional`*) – Whether a lower value of `outcome` (i.e. `0`) is favorable. Defaults to False.
- **`label`** (*`Optional[Series], optional`*) – Boolean label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`max_for_fishers`** (*`int, optional`*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.

Returns Object containing results of the disparity calculation.

Return type Disparity

precision

`solas_disparity.disparity.`**`precision`** (...)

This method calculates the Precision for a given set of protected and reference groups.
Also know as Precision.

Precision = True Positives / (True Positives + False Positives)

An TPR is considered "practically significant" if the Precision is:

1. less than a chosen *ratio_threshold*.
2. statistically significantly different than parity, AND
3. greater than *difference_threshold*.

Note Ratio is defined as (Precision of Protected Group) / (Precision of Reference Group)

Difference is defined as (Precision of Reference Group) - (Precision of Protected Group)

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series.
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`ratio_threshold`** (*`float`*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*`float`*) – Threshold at which a difference is considered significant.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`statistical_significance_test`** (*`Optional[StatSigTest], optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **`p_value_threshold`** (*`float, optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None. Defaults to 0.05.
- **`statistical_significance_arguments`** (*`Dict[str, Any], optional`*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

residual_standardized_mean_difference

`solas_disparity.disparity.`**`residual_standardized_mean_difference`**`(...)`

Calculate the Standardized Mean Difference of residuals for a given set of protected and reference groups.

A residual is the label value minus the predicted value.

\text{Residual} = y - \hat{y}

The Residual SMD is the mean residual of a protected group minus the mean residual of a reference group all divided by the standard deviation of residuals.

\text{Residual SMD}_\text{Protected Group} = \frac{\text{Mean Residual}_{\text{Protected

Group}} - \text{Mean Residual}_{\text{Reference Group}}}{s}

A Residual SMD is considered practically significant if `lower_score_favorable=True` and the Residual SMD is:

1. greater in magnitude than a chosen `residual_smd_threshold`
2. AND statistically significantly different than zero.

Alternatively, a Residual SMD is considered practically significant if `lower_score_favorable=False` and the Residual SMD is:

1. lesser in magnitude than a chosen `residual_smd_threshold`
2. AND statistically significantly different than zero.

Parameters
- **`group_data`** (*DataFrame*) – Dataframe containing columns for group data.
- **`protected_groups`** (*List[str]*) – List of protected groups.
- **`reference_groups`** (*List[str]*) – List of reference groups.
- **`group_categories`** (*List[str]*) – List of group categories.
- **`prediction`** – (Series): Predictions series.
- **`label`** (*Series*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`residual_smd_threshold`** (*float*) – Residual Standardized Mean Difference threshold value.
- **`lower_score_favorable`** (*bool, optional*) – Whether a lower value of `label` - `prediction` is favorable. Defaults to True.
- **`sample_weight`** – Optional[Series]: Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`residual_smd_denominator`** (*Union[ResidualSMDDenominator, str], optional*) – Residual Standardized Mean Difference denominator calculation. Defaults to ResidualSMDDenominator.POPULATION.

Returns Object containing results of the disparity calculation.

Return type Disparity

scoring_impact_ratio

`solas_disparity.disparity.`**`scoring_impact_ratio`**(...)

Calculate the scoring impact ratio, an impact ratio where the boolean outcome is whether an individual scored above the median.

Parameters
- **`group_data`** (*DataFrame*) – Dataframe containing columns for group data.
- **`race_ethnicity_groups`** (*List[str]*) – A list of race/ethnicity groups corresponding to individual columns in group_data (e.g. *["Hispanic or Latino", "White", "Black or African American", "Native Hawaiian or Pacific Islander", "Asian", "Native American or Alaska Native", "Two or More Races"]*).
- **`gender_groups`** (*List[str]*) – A list of gender groups corresponding to individual columns in group_data (e.g. *["Male", "Female"]*).
- **`outcome`** (*Series*) – Outcome series of scores.
- **`ratio_threshold`** (*float*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*float*) – Threshold at which a difference is considered significant.

- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **max_for_fishers** (*int, optional*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.
- **shortfall_method** (*Optional[types.ShortfallMethod], optional*) – Method used for shortfall calculation. Defaults to ShortfallMethod.TO_REFERENCE_MEAN.
- **drop_small_groups** (*Optional[bool], optional*) – Whether to separate and return a table of groups that comprise less than 2% of individuals. Defaults to True.

Returns Object containing results of the disparity calculation.

Return type Disparity

segmented_adverse_impact_ratio

`solas_disparity.disparity.`**`segmented_adverse_impact_ratio`** (...)

Calculates within-segment and cross-segment disparate impact and statistical significance statistics for dichotomous outcomes when it is appropriate to aggregate segment results to a single cross-segment measurement of disparity. A `segment` might be a market, location, type of job, or some other discrete unit.

The Adverse Impact Ratio (AIR) is calculated within each segment and cross-segment.

AIR is defined as the percentage of favorable outcomes of the protected group divided by the percentage of favorable outcomes of the reference group.

\text{AIR}_\text{Protected Group} = \frac{\text{% Favorable Outcome}_\text{Protected Group}}{\text{% Favorable Outcome}_\text{Reference Group}}

The cross-segment AIR attempts to control for the fact that the protected class may be distributed across segments differently from the reference class. In order to account for this, this method keeps the distribution of protected group members constant across the segments, but reallocates the reference group members according to the protected group members' distribution. It does keep the protected group and reference group favorable outcome rates constant within each segment. In this way, the method is able to adjust for different distributions of class members by segment, but test whether the overall acceptance rates differ across classes.

Two cross-segment tests for statistical significance are performed: the Cochran-Mantel-Haenszel, CMH, test, which tests whether all segment-level odds ratios are equal to 1.0; and the Breslow-Day test, which tests whether segment level odds ratios are equal to each other.

When one can reject the Breslow-Day test, the p-values from the segment level tests (i.e., the Fisher's Exact or Chi-Squared tests) are assessed against critical values derived from the Benjamani-Hochberg Procedure for multiple testing, rather than the standard 2-tailed 5% rule.

When one cannot reject the Breslow-Day test, then the CMH practical significance results can be applied to within-segment practical significance. If `overwrite_segment_results=True`, then the within-segment practical significance results are overwritten with the CMH practical significance results. By setting `overwrite_segment_results=False`, within-segment results will not be overwritten. For the case where group count is zero in a segment, the practical significance will remain `""` and not be overwritten.

Parameters

- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Boolean outcome series where a value of `1` is assumed to be favorable.

- **air_threshold** (*float*) – Adverse Impact Ratio threshold value.
- **percent_difference_threshold** (*float*) – Percent difference threshold value. For example, a 20% difference is input as `percent_difference_threshold=0.2`.
- **fdr_threshold** (*float*) – False discovery rate (fdr) threshold value used in calculating whether segment-level results are statistically significant using the Benjamani-Hochberg Procedure.
- **segment** (*Series*) – Segment series.
- **label** (*Optional[Series], optional*) – Boolean label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **max_for_fishers** (*int, optional*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.
- **shift_zeros** (*bool, optional*) – If `True`, if any cell count in a contingency table used in the CMH and Breslow-Day tests is zero, then 0.5 is added to all values in the contingency table so an odds ratio is able to be calculated. Note: In large sample sizes, this correction will not make a significant difference. In small sample sizes, this correction has the potential to impact the significance determination. Defaults to True.

Returns Object containing results of the disparity calculation.

Return type Disparity

selection_impact_ratio

`solas_disparity.disparity.`**`selection_impact_ratio`**`( ... )`

Calculate the selection impact ratio.

Parameters

- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **race_ethnicity_groups** (*List[str]*) – A list of race/ethnicity groups corresponding to individual columns in group_data (e.g. *["Hispanic or Latino", "White", "Black or African American", "Native Hawaiian or Pacific Islander", "Asian", "Native American or Alaska Native", "Two or More Races"]*).
- **gender_groups** (*List[str]*) – A list of gender groups corresponding to individual columns in group_data (e.g. *["Male", "Female"]*).
- **outcome** (*Series*) – Boolean outcome series.
- **ratio_threshold** (*float*) – Threshold at which a ratio is considered significant.
- **difference_threshold** (*float*) – Threshold at which a difference is considered significant.
- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **max_for_fishers** (*int, optional*) – Maximum value of samples for Fisher's exact test to be used. Defaults to MAX_FOR_FISHERS.
- **shortfall_method** (*Optional[types.ShortfallMethod], optional*) – Method used for shortfall calculation. Defaults to ShortfallMethod.TO_REFERENCE_MEAN.
- **drop_small_groups** (*Optional[bool], optional*) – Whether to separate and return a table of groups that comprise less than 2% of individuals. Defaults to True.

Returns Object containing results of the disparity calculation.

Return type Disparity

standardized_mean_difference

`solas_disparity.disparity.`**`standardized_mean_difference`**`( ... )`

Calculate the Standardized Mean Difference (SMD) for a given set of protected and reference groups.

The SMD is the mean outcome of a protected group minus the mean outcome of a reference group all divided by the standard deviation of outcomes.

\text{SMD}_\text{Protected Group} = \frac{\text{Mean Outcome}_{\text{Protected Group}} - \text{Mean Outcome}_{\text{Reference Group}}}{s}

An SMD is considered practically significant if `lower_score_favorable=True` and the SMD is:

1. greater than a chosen `smd_threshold`
2. AND statistically significantly different than zero.

Alternatively, an SMD is considered practically significant if `lower_score_favorable=False` and the SMD is:

1. lesser than a chosen `smd_threshold`
2. AND statistically significantly different than zero.

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series.
- **`smd_threshold`** (*`float`*) – Standardized Mean Difference threshold value.
- **`lower_score_favorable`** (*`bool, optional`*) – Whether a lower value of `outcome` is favorable. Defaults to True.
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`smd_denominator`** (*`SMDDenominator, optional`*) – Standardized Mean Difference denominator calculation. Defaults to SMDDenominator.POPULATION.

Returns Object containing results of the disparity calculation.

Return type Disparity

true_negative_rate

`solas_disparity.disparity.`**`true_negative_rate`**`( ... )`

This method calculates the True Negative Rate (TNR) for a given set of protected and reference groups.

Also know as Specificity or Selectivity.

True Negative Rate (TNR) = True Negatives / (True Negatives + False Positives)

An TPR is considered "practically significant" if the TNR is:

1. less than a chosen *ratio_threshold*.
2. statistically significantly different than parity, AND
3. greater than *difference_threshold*.

Note Ratio is defined as (TNR of Protected Group) / (TNR of Reference Group)

Difference is defined as (TNR of Reference Group) - (TNR of Protected Group)

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series.
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`ratio_threshold`** (*`float`*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*`float`*) – Threshold at which a difference is considered significant.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`statistical_significance_test`** (*`Optional[StatSigTest], optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **`p_value_threshold`** (*`float, optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None. Defaults to 0.05.
- **`statistical_significance_arguments`** (*`Dict[str, Any], optional`*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

true_positive_rate

`solas_disparity.disparity.`**`true_positive_rate`**`( ... )`

This method calculates the True Positve Rate (TPR) for a given set of protected and reference groups.

Also know as Sensitivity, Recall or Hit Rate.

True Positive Rate (TPR) = True Positives / (True Positives + False Negatives)

An TPR is considered "practically significant" if the TPR is:

1. less than a chosen *ratio_threshold*.
2. statistically significantly different than parity, AND

3. greater than *difference_threshold*.

Note Ratio is defined as (TPR of Protected Group) / (TPR of Reference Group)

Difference is defined as (TPR of Reference Group) - (TPR of Protected Group)

Parameters

- **`group_data`** (*`DataFrame`*) – Dataframe containing columns for group data.
- **`protected_groups`** (*`List[str]`*) – List of protected groups.
- **`reference_groups`** (*`List[str]`*) – List of reference groups with the same length as `protected_groups`.
- **`group_categories`** (*`List[str]`*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **`outcome`** (*`Series`*) – Outcome series.
- **`label`** (*`Optional[Series], optional`*) – Label, true outcome, and/or target series evaluated alongside `outcome`. Defaults to None.
- **`ratio_threshold`** (*`float`*) – Threshold at which a ratio is considered significant.
- **`difference_threshold`** (*`float`*) – Threshold at which a difference is considered significant.
- **`sample_weight`** (*`Optional[Series], optional`*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **`statistical_significance_test`** (*`Optional[StatSigTest], optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None.
- **`p_value_threshold`** (*`float, optional`*) – Statistical significance test that will be factored into practical significance if not `None`. Defaults to None. Defaults to 0.05.
- **`statistical_significance_arguments`** (*`Dict[str, Any], optional`*) – A dictionary of arbitrary arguments to the statistical significance test function. Every function corresponds to a member of `StatSigTest`. Defaults to dict().

Returns Object containing results of the disparity calculation.

Return type Disparity

2.1.20 solas_disparity.plots

Functions

`plot_adverse_impact_ratio`	Generate a plot for an AIR result object.
`plot_adverse_impact_ratio_by_quantile`	Generate a plot for an AIR-by-quantile result object.
`plot_categorical_adverse_impact_ratio`	Generate a plot for a categorical AIR result object.
`plot_custom_disparity_metric`	Generate a plot for a custom disparity metric result object.
`plot_false_discovery_rate`	Generate a plot for an FDR result object.
`plot_false_negative_rate`	Generate a plot for an FNR result object.
`plot_false_positive_rate`	Generate a plot for an FPR result object.

plot_odds_ratio	Generate a plot for an odds ratio result object.
plot_precision	Generate a plot for a precision result object.
plot_residual_standardized_mean_difference	Generate a plot for an residual SMD result object.
plot_scoring_impact_ratio	Generate a plot for a categorical AIR result object.
plot_segmented_adverse_impact_ratio	Generate a plot for a segmented AIR result object.
plot_selection_impact_ratio	Generate a plot for a categorical AIR result object.
plot_standardized_mean_difference	Generate a plot for an SMD result object.
plot_true_negative_rate	Generate a plot for a TNR result object.
plot_true_positive_rate	Generate a plot for an TPR result object.
plot_wasserstein	Generate a plot for a Wasserstein result object.

plot_adverse_impact_ratio

solas_disparity.plots.**plot_adverse_impact_ratio** (...)

Generate a plot for an AIR result object.

Parameters
- **disparity** (*Disparity*) – Disparity result.
- **column** (*str, optional*) – Column to plot. Defaults to const.AIR_VALUES.

Returns An interactive plotly figure.

Return type Figure

plot_adverse_impact_ratio_by_quantile

solas_disparity.plots.**plot_adverse_impact_ratio_by_quantile** (...)

Generate a plot for an AIR-by-quantile result object.

Parameters
- **disparity** (*Disparity*) – Disparity result.
- **column** (*str, optional*) – Column to plot. Defaults to const.AIR_VALUES.
- **quantile** (*Optional[Union[str, float]], optional*) – Quantile slice of the summary table to plot. If None, plot all quantiles. Defaults to None.
- **separate** (*bool, optional*) – Whether to generate a separate figure for each quantile. Defaults to False.
- **group** (*Optional[str], optional*) – Group slice of the summary table to plot. If None, plot all groups. Defaults to None.

Returns A figure or list of interactive plotly figures if separate is set to True.

Return type Union[Figure, List[Figure]]

plot_categorical_adverse_impact_ratio

solas_disparity.plots.**plot_categorical_adverse_impact_ratio** (...)

Generate a plot for a categorical AIR result object.

Parameters
- **disparity** (*Disparity*) – Disparity result.
- **column** (*str, optional*) – Column to plot. Defaults to const.AIR_VALUES.
- **category** (*Optional[Union[str, float]], optional*) – Category

slice of the summary table to plot. If None, plot all quantiles. Defaults to None.

- **separate** (*bool, optional*) – Whether to generate a separate figure for each category. Defaults to False.
- **group** (*Optional[str], optional*) – Group slice of the summary table to plot. If None, plot all groups. Defaults to None.

Returns A figure or list of interactive plotly figures if separate is set to True.

Return type Union[Figure, List[Figure]]

plot_custom_disparity_metric

`solas_disparity.plots.`**`plot_custom_disparity_metric`**`( ... )`

Generate a plot for a custom disparity metric result object.

Parameters
- **disparity** (*Disparity*) – Disparity result.
- **column** (*Optional[str], optional*) – Column to plot. If None, the function will try to plot the ratio, difference, and metric in that order. Defaults to None.

Returns An interactive plotly figure.

Return type Figure

plot_false_discovery_rate

`solas_disparity.plots.`**`plot_false_discovery_rate`**`( ... )`

Generate a plot for an FDR result object.

Parameters
- **disparity** (*Disparity*) – Disparity result.
- **column** (*str, optional*) – Column to plot. Defaults to const.RATIO.

Returns An interactive plotly figure.

Return type Figure

plot_false_negative_rate

`solas_disparity.plots.`**`plot_false_negative_rate`**`( ... )`

Generate a plot for an FNR result object.

Parameters
- **disparity** (*Disparity*) – Disparity result.
- **column** (*str, optional*) – Column to plot. Defaults to const.RATIO.

Returns An interactive plotly figure.

Return type Figure

plot_false_positive_rate

`solas_disparity.plots.`**`plot_false_positive_rate`**`( ... )`

Generate a plot for an FPR result object.

Parameters
- **disparity** (*Disparity*) – Disparity result.
- **column** (*str, optional*) – Column to plot. Defaults to const.RATIO.

Returns An interactive plotly figure.

Return type Figure

plot_odds_ratio

`solas_disparity.plots.`**`plot_odds_ratio`** (...)

Generate a plot for an odds ratio result object.

Parameters
- **`disparity`** (*`Disparity`*) – Disparity result.
- **`column`** (*`str, optional`*) – Column to plot. Defaults to const.ODDS_RATIO.

Returns An interactive plotly figure.

Return type Figure

plot_precision

`solas_disparity.plots.`**`plot_precision`** (...)

Generate a plot for a precision result object.

Parameters
- **`disparity`** (*`Disparity`*) – Disparity result.
- **`column`** (*`str, optional`*) – Column to plot. Defaults to const.RATIO.

Returns An interactive plotly figure.

Return type Figure

plot_residual_standardized_mean_difference

`solas_disparity.plots.`**`plot_residual_standardized_mean_difference`** (...)

Generate a plot for an residual SMD result object.

Parameters
- **`disparity`** (*`Disparity`*) – Disparity result.
- **`column`** (*`str, optional`*) – Column to plot. Defaults to const.RESIDUAL_S-MD_VALUES.

Returns An interactive plotly figure.

Return type Figure

plot_scoring_impact_ratio

`solas_disparity.plots.`**`plot_scoring_impact_ratio`** (...)

Generate a plot for a categorical AIR result object.

Parameters
- **`disparity`** (*`Disparity`*) – Disparity result.
- **`column`** (*`str, optional`*) – Column to plot. Defaults to const.IMPACT_RATIO.
- **`group_category`** (*`Optional[Union[str, float]], optional`*) – Category slice of the summary table to plot.
- **`None`** (*`If`*) – "Race/Ethinicty", "Gender" and "Intersectional". Defaults to None
- **`categories`** (*`plot all group`*) – "Race/Ethinicty", "Gender" and "Intersectional". Defaults to None
- **`separate`** (*`bool, optional`*) – Whether to generate a separate figure for each category. Defaults to False.

Returns A figure or list of interactive plotly figures if separate is set to True.

Return type Union[Figure, List[Figure]]

plot_segmented_adverse_impact_ratio

`solas_disparity.plots.`**`plot_segmented_adverse_impact_ratio`** (...)

Generate a plot for a segmented AIR result object.

Parameters

- **`disparity`** (*`Disparity`*) – Disparity result.
- **`column`** (*`str, optional`*) – Column to plot. Defaults to const.AIR_VALUES.
- **`segment`** (*`Optional[str], optional`*) – Segment slice of the summary table to plot. If None, plot all segments. Defaults to None.
- **`separate`** (*`bool, optional`*) – Whether to generate a separate figure for each segment. Defaults to False.
- **`group`** (*`Optional[str], optional`*) – Group slice of the summary table to plot. If None, plot all groups. Defaults to None.

Returns A figure or list of interactive plotly figures if separate is set to True.

Return type Union[Figure, List[Figure]]

plot_selection_impact_ratio

`solas_disparity.plots.`**`plot_selection_impact_ratio`** (...)

Generate a plot for a categorical AIR result object.

Parameters

- **`disparity`** (*`Disparity`*) – Disparity result.
- **`column`** (*`str, optional`*) – Column to plot. Defaults to const.IMPACT_RATIO.
- **`group_category`** (*`Optional[Union[str, float]], optional`*) – Category slice of the summary table to plot.
- **`None`** (*`If`*) – "Race/Ethinicty", "Gender" and "Intersectional". Defaults to None
- **`categories`** (*`plot all group`*) – "Race/Ethinicty", "Gender" and "Intersectional". Defaults to None
- **`separate`** (*`bool, optional`*) – Whether to generate a separate figure for each category. Defaults to False.

Returns A figure or list of interactive plotly figures if separate is set to True.

Return type Union[Figure, List[Figure]]

plot_standardized_mean_difference

`solas_disparity.plots.`**`plot_standardized_mean_difference`** (...)

Generate a plot for an SMD result object.

Parameters

- **`disparity`** (*`Disparity`*) – Disparity result.
- **`column`** (*`str, optional`*) – Column to plot. Defaults to const.SMD_VALUES.

Returns An interactive plotly figure.

Return type Figure

plot_true_negative_rate

`solas_disparity.plots.`**`plot_true_negative_rate`** (...)

Generate a plot for a TNR result object.

Parameters

- **`disparity`** (*`Disparity`*) – Disparity result.

- **column** (*str, optional*) – Column to plot. Defaults to const.RATIO.

Returns An interactive plotly figure.

Return type Figure

plot_true_positive_rate

`solas_disparity.plots.`**`plot_true_positive_rate`**`( ... )`

Generate a plot for an TPR result object.

Parameters
- **disparity** (*Disparity*) – Disparity result.
- **column** (*str, optional*) – Column to plot. Defaults to const.RATIO.

Returns An interactive plotly figure.

Return type Figure

plot_wasserstein

`solas_disparity.plots.`**`plot_wasserstein`**`( ... )`

Generate a plot for a Wasserstein result object.

Parameters
- **disparity** (*Disparity*) – Disparity result.
- **column** (*str, optional*) – Column to plot. Defaults to const.UNFAVORABLE_DISPARITY.

Returns An interactive plotly figure.

Return type Figure

2.1.21 solas_disparity.statistical_significance

Functions

`bootstrapping`	Conduct bootstrapping for statistical significance.
`chi_squared_test`	Conduct a chi-squared test for contingency tables.
`fishers_exact`	Conduct Fisher's exact test for contingency tables.
`fishers_or_chi_squared`	Conduct either a Fisher's exact test or a chi-squared test.
`stacked_regression`	Create a stacked regression dataset and use it to conduct a two-sample t-test.
`two_sample_t_test`	Conduct a two-sample t-test.

bootstrapping

`solas_disparity.statistical_significance.`**`bootstrapping`**`( ... )`

Conduct bootstrapping for statistical significance.

Parameters
- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Outcome series.
- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.

- **resamples** (*int, optional*) – The number of independent resamples. Defaults to const.RESAMPLES.
- **sample** (*Union[float, int], optional*) – The sample size or sample fraction. Defaults to const.SAMPLE.
- **seed** (*Optional[int], optional*) – Random seed passed through to numpy.random.default_rng. Defaults to None.
- **replace** (*bool, optional*) – Whether to sample with replacement. Defaults to False.

Raises **NotImplementedError** – Bootstrapping will be implemented soon.

Returns Statistical significance result object.

Return type StatSig

chi_squared_test

`solas_disparity.statistical_significance.`**`chi_squared_test`** (...)

Conduct a chi-squared test for contingency tables.

Parameters

- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Outcome series.
- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.

Returns Statistical significance result object.

Return type StatSig

fishers_exact

`solas_disparity.statistical_significance.`**`fishers_exact`** (...)

Conduct Fisher's exact test for contingency tables.

Parameters

- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Outcome series.
- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.

Returns Statistical significance result object.

Return type StatSig

fishers_or_chi_squared

solas_disparity.statistical_significance.**fishers_or_chi_squared** (...)

Conduct either a Fisher's exact test or a chi-squared test. If max_for_fishers is greater than any value of the expected or observed contingency table, the Fisher's exact test is conducted. Else, the chi-squared test is conducted.

Parameters
- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Outcome series.
- **sample_weight** (*Optional[pd.Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.
- **max_for_fishers** (*Union[int, float], optional*) – Maximum value of samples for Fisher's exact test to be used. Defaults to const.MAX_FOR_FISHERS.

Returns Statistical significance result object.

Return type StatSig

stacked_regression

solas_disparity.statistical_significance.**stacked_regression** (...)

Create a stacked regression dataset and use it to conduct a two-sample t-test.

Parameters
- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Outcome series.
- **regression_type** (*StatSigRegressionType, optional*) – Type of regression to be performed. Defaults to StatSigRegressionType.GAUSSIAN.
- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Defaults to None.

Returns Statistical significance result object.

Return type StatSig

two_sample_t_test

solas_disparity.statistical_significance.**two_sample_t_test** (...)

Conduct a two-sample t-test.

Parameters
- **group_data** (*DataFrame*) – Dataframe containing columns for group data.
- **protected_groups** (*List[str]*) – List of protected groups.

- **reference_groups** (*List[str]*) – List of reference groups with the same length as `protected_groups`.
- **group_categories** (*List[str]*) – List of group categories to which each protected and reference group pair belongs to (e.g. race, gender, age, etc.). Has the same length as `protected_groups`.
- **outcome** (*Series*) – Outcome series.
- **sample_weight** (*Optional[Series], optional*) – Sample weight series. Has the same length as `group_data`. Defaults to None.

Raises **NotImplementedError** – Bootstrapping will be implemented soon.

Returns Statistical significance result object.

Return type StatSig

2.1.22 solas_disparity.types

Classes

`DifferenceCalculation`	Enum class denoting possible difference calculations.
`Disparity`	Dataclass for disparity objects.
`DisparityCalculation`	Enum class denoting possible disparity calculations.
`DisparityResponse`	
`EnumBase`	Enum base class.
`RatioCalculation`	Enum class denoting possible ratio calculations.
`ResidualSMDDenominator`	Enum class denoting possible SMD denominators.
`SMDDenominator`	Enum class denoting possible SMD denominators.
`ShortfallMethod`	Enum class denoting possible shortfall methods.
`StatSig`	Dataclass for statistical significance test outputs.
`StatSigRegressionType`	Enum class denoting possible regression types to be performed by stacked regression.
`StatSigTest`	Enum class denoting possible statistical significance tests.

DifferenceCalculation

class solas_disparity.types.**DifferenceCalculation** (...)

Enum class denoting possible difference calculations.

Attributes

`REFERENCE_MINUS_PROTECTED`	
`PROTECTED_MINUS_REFERENCE`	

Disparity

class solas_disparity.types.**Disparity** (...)

Dataclass for disparity objects.

Methods

`__init__`	Method generated by attrs for class Disparity.
`show`	
`to_excel`	Export summary table as an XLSX file.

Attributes

`affected_categories`	Group categories that correspond to practically significant groups.

`affected_groups`	Protected groups that have practically significant adverse disparities.
`affected_reference`	Reference groups that correspond to practically significant groups.
`plot`	
`report`	Data in NYC Department of Consumer and Worker Protection format.
`disparity_type`	Type of disparity calculation.
`summary_table`	Summary table of disparity calculation results.
`protected_groups`	Protected group names.
`reference_groups`	Reference group names.
`group_categories`	Group category names.
`statistical_significance`	StatSig object.
`smd_threshold`	Standardized mean difference threshold.
`residual_smd_threshold`	Residual Standardized mean difference threshold.
`smd_denominator`	Standardized mean difference denominator.
`residual_smd_denominator`	Residual standardized mean difference denominator.
`lower_score_favorable`	Is a lower pre-transformation prediction favorable? If True, then the model's predictions are assumed to be more favorable the lower the value.
`odds_ratio_threshold`	Odds ratio threshold.
`air_threshold`	AIR threshold.
`percent_difference_threshold`	Percent difference threshold value.
`max_for_fishers`	Max value of samples for Fishers Exact test to be used.
`shortfall_method`	Shortfall method.
`fdr_threshold`	False discovery rate threshold for use when calculating segment-level results are statistically significant using the Benjamani-Hochberg Procedure.
`metric`	Metric Requested
`difference_calculation`	Difference Calculation.
`difference_threshold`	Difference threshold as a float set only for curated custom disparity metrics such as FDR.
`ratio_calculation`	Ratio Calculation.
`ratio_threshold`	Ratio threshold as float set only for curated custom disparity metrics such as FDR.
`statistical_significance_test`	Statistical Significance Method
`p_value_threshold`	Statistical Significance P-value Threshold
`shift_zeros`	If `True`, if any cell count in a contingency table used in the CMH and Breslow-Day tests is zero, then 0.5 is added to all values in the contingency table so an odds ratio is able to be calculated.
`drop_small_groups`	Whether to separate and return a table of groups that comprise less than 2% of individuals.
`small_group_table`	Groups comprising less than 2% of individuals.
`unknown_table`	Summary of individuals with unknown demographic information.

property `affected_categories`: Optional[List[str]]

Group categories that correspond to practically significant groups.

property `affected_groups`: List[str]
Protected groups that have practically significant adverse disparities.

property `affected_reference`: List[str]
Reference groups that correspond to practically significant groups.

`air_threshold`: Optional[float]
AIR threshold. Set by the user, this takes a float that represents the AIR level below which Solas will identify as being indicative of a practically significant disparity. Legal and compliance counsel should be sought for the appropriate AIR threshold in a given use case.

`difference_calculation`: Optional[solas_disparity.types._difference_calculation.DifferenceCalculation]
Difference Calculation.

`difference_threshold`: Optional[float]
Difference threshold as a float set only for curated custom disparity metrics such as FDR.

`disparity_type`: solas_disparity.types._disparity_calculation.DisparityCalculation
Type of disparity calculation.

`drop_small_groups`: bool
Whether to separate and return a table of groups that comprise less than 2% of individuals.

`fdr_threshold`: Optional[float]
False discovery rate threshold for use when calculating segment-level results are statistically significant using the Benjamani-Hochberg Procedure.

`group_categories`: List[str]
Group category names. Same length as `protected_groups`. Set by the user, this takes a list of strings which represent the reference groups (also known as control groups) being analyzed. There must be a one-to-one correspondence between reference groups and protected_groups. Note that the protected groups and reference groups are aligned by index in the lists.

`lower_score_favorable`: Optional[bool]
Is a lower pre-transformation prediction favorable? If True, then the model's predictions are assumed to be more favorable the lower the value. If False, then the model's predictions are assumed to be more favorable the higher the value. Optional. If omitted, defaults to True.

`max_for_fishers`: Optional[int]
Max value of samples for Fishers Exact test to be used. Defaults to const.MAX_FOR_FISHERS. Set by the user, this takes an integer and defaults to 100.

`metric`: Callable[[...], Union[int, float]]
Metric Requested

`odds_ratio_threshold`: Optional[float]
Odds ratio threshold.

`p_value_threshold`: float
Statistical Significance P-value Threshold

`percent_difference_threshold`: Optional[float]
Percent difference threshold value. For example, if percent_difference_threshold = 0.2, then the difference in percent favorable will need to exceed 20% for a result to be practically significant.

`protected_groups`: List[str]
: Protected group names. Set by the user, this takes a list of strings which represent the protected groups being analyzed. There can be as few as one protected group and there is no upper limit to the number of protected groups that can be analyzed.

`ratio_calculation`: Optional[solas_disparity.types._ratio_calculation.RatioCalculation]
: Ratio Calculation.

`ratio_threshold`: Optional[float]
: Ratio threshold as float set only for curated custom disparity metrics such as FDR.

`reference_groups`: List[str]
: Reference group names. Same length as `protected_groups`. Set by the user, this takes a list of strings which represent the reference groups (also known as control groups) being analyzed. There must be a one-to-one correspondence between reference groups and protected_groups. Note that the protected groups and reference groups are aligned by index in the lists.

property `report`: Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]
: Data in NYC Department of Consumer and Worker Protection format.

Raises	**`ValueError`** – If the disparity calculation is not supported
Returns	Gender, Race/Ethnicity, and Intersectional tables.
Return type	Tuple[DataFrame, DataFrame, DataFrame]

`residual_smd_denominator`: Optional[str]
: Residual standardized mean difference denominator. Defaults to ResidualSMDDenominator.POPULATION.

`residual_smd_threshold`: Optional[float]
: Residual Standardized mean difference threshold.

`shift_zeros`: bool
: If `True`, if any cell count in a contingency table used in the CMH and Breslow-Day tests is zero, then 0.5 is added to all values in the contingency table so an odds ratio is able to be calculated. Note: In large sample sizes, this correction will not make a significant difference. In small sample sizes, this correction has the potential to impact the significance determination. Defaults to True.

`shortfall_method`: Optional[solas_disparity.types._shortfall_method.ShortfallMethod]
: Shortfall method. Set by the user. Determines the value of the `const.SHORTFALL` column in `Disparity.summary_table`. Defaults to `ShortfallMethod.TO_REFERENCE_MEAN`.

`small_group_table`: pandas.core.frame.DataFrame
: Groups comprising less than 2% of individuals.

`smd_denominator`: Optional[str]
: Standardized mean difference denominator. Defaults to SMDDenominator.POPULATION. This defines how the standard deviation of scores is set. Options include either “population” or “pooled”.

`smd_threshold`: Optional[float]
: Standardized mean difference threshold. Set by the user, this takes a float that represents the SMD level which Solas will identify as being indicative of a practically significant disparity. Legal and compliance counsel should be sought for the appropriate SMD threshold in a given use case.

`statistical_significance`: Optional[solas_disparity.types._stat_sig.StatSig]
StatSig object. Contains statistical significance information stated in `Disparity.summary_table` and sometimes more than the former.

`statistical_significance_test`: Optional[solas_disparity.types._stat_sig_test.StatSigTest]
Statistical Significance Method

`summary_table`: pandas.core.frame.DataFrame
Summary table of disparity calculation results. Provided as a Pandas DataFrame. This is a stand-alone version of the summary table provided by .disparity

`to_excel` (*file_path: Union[str, pathlib.Path]*)
Export summary table as an XLSX file.

Parameters `file_path` (*`Union[str, Path]`*) – Path to file.

`unknown_table`: pandas.core.frame.DataFrame
Summary of individuals with unknown demographic information.

DisparityCalculation

class `solas_disparity.types.`**`DisparityCalculation`** (...)
Enum class denoting possible disparity calculations.
Attributes

`ADVERSE_IMPACT_RATIO`	
`SEGMENTED_ADVERSE_IMPACT_RATIO`	
`ADVERSE_IMPACT_RATIO_BY_QUANTILE`	
`CATEGORICAL_ADVERSE_IMPACT_RATIO`	
`STANDARDIZED_MEAN_DIFFERENCE`	
`RESIDUAL_STANDARDIZED_MEAN_DIFFERENCE`	
`ODDS_RATIO`	
`CUSTOM_DISPARITY_METRIC`	
`TRUE_POSITIVE_RATE`	
`TRUE_NEGATIVE_RATE`	
`FALSE_POSITIVE_RATE`	
`FALSE_NEGATIVE_RATE`	
`PRECISION`	
`FALSE_DISCOVERY_RATE`	
`SCORING_IMPACT_RATIO`	
`SELECTION_IMPACT_RATIO`	
`NOT_SET`	

DisparityResponse

class `solas_disparity.types.`**`DisparityResponse`** (...)
Methods

`__init__`	Create a new model by parsing and validating input data from keyword arguments.
`construct`	Creates a new model setting __dict__ and __fields_set__ from trusted or pre-validated data.
`copy`	Duplicate a model, optionally choose which fields to include, exclude and change.

`dict`	Generate a dictionary representation of the model, optionally specifying which fields to include or exclude.
`from_disparity`	
`from_orm`	
`json`	Generate a JSON representation of the model, *include* and *exclude* arguments as per *dict()*.
`parse_file`	
`parse_obj`	
`parse_raw`	
`schema`	
`schema_json`	
`update_forward_refs`	Try to update ForwardRefs on fields based on this Model, globalns and localns.
`validate`	

Attributes

`metadata`	
`disparity_type`	Type of disparity calculation performed.
`summary_table_json`	Summary table of disparity calculation results.
`summary_table_json_flat`	Summary table of disparity calculation results.
`protected_groups`	Protected group names.
`reference_groups`	Reference group names.
`group_categories`	Group category names.
`outcome`	Column containing ordinal value of each category
`air_threshold`	AIR threshold.
`percent_difference_threshold`	Percent difference threshold value.
`label`	A string representing the name of the label column in *group_data*.
`sample_weight`	A string representing the name of the column in *group_data*.
`max_for_fishers`	The maximum number of groups that can be used for Fisher's Exact Test.
`shortfall_method`	
`smd_threshold`	Standardized mean difference threshold.
`lower_score_favorable`	Is a lower pre-transformation prediction favorable? If True, then the model's predictions are assumed to be more favorable the lower the value.
`smd_denominator`	The denominator used for the SMD calculation.
`plot_json`	Plot of disparity calculation results.

`air_threshold`: Optional[float]

AIR threshold. Set by the user, this takes a float that represents the AIR level below which Solas will identify as being indicative of a practically significant disparity. Legal and compliance counsel should be sought for the appropriate AIR threshold in a given use case.

`disparity_type`: solas_disparity.types._disparity_calculation.DisparityCalculation

Type of disparity calculation performed.

`group_categories`: List[str]

Group category names. Same length as `protected_groups`. Set by the user, this takes a list of strings which represent the reference groups (also known as control groups) being analyzed. There must be a one-to-one correspondence between reference groups and protected_groups. Note that the protected groups and reference groups are aligned by index in the lists.

`label`: Optional[str]
A string representing the name of the label column in *group_data*.

`lower_score_favorable`: Optional[bool]
Is a lower pre-transformation prediction favorable? If True, then the model's predictions are assumed to be more favorable the lower the value. If False, then the model's predictions are assumed to be more favorable the higher the value. Optional. If omitted, defaults to True.

`max_for_fishers`: Optional[int]
The maximum number of groups that can be used for Fisher's Exact Test. If the number of groups is greater than this value, Fisher's Exact Test will not be used. Instead, the chi-squared test will be used.

`outcome`: Optional[str]
Column containing ordinal value of each category

`percent_difference_threshold`: Optional[float]
Percent difference threshold value. For example, if percent_difference_threshold = 0.2, then the difference in percent favorable will need to exceed 20% for a result to be practically significant.

`plot_json`: Optional[str]
Plot of disparity calculation results. Provided as a Plotly json definition.

`protected_groups`: List[str]
Protected group names. Set by the user, this takes a list of strings which represent the protected groups being analyzed. There can be as few as one protected group and there is no upper limit to the number of protected groups that can be analyzed.

`reference_groups`: List[str]
Reference group names. Same length as `protected_groups`. Set by the user, this takes a list of strings which represent the reference groups (also known as control groups) being analyzed. There must be a one-to-one correspondence between reference groups and protected_groups. Note that the protected groups and reference groups are aligned by index in the lists.

`sample_weight`: Optional[str]
A string representing the name of the column in *group_data*.

`smd_denominator`: Optional[solas_disparity.types._smd_denominator.SMDDenominator]
The denominator used for the SMD calculation.

`smd_threshold`: Optional[float]
Standardized mean difference threshold. Set by the user, this takes a float that represents the SMD level which Solas will identify as being indicative of a practically significant disparity. Legal and compliance counsel should be sought for the appropriate SMD threshold in a given use case.

`summary_table_json`: str
Summary table of disparity calculation results. Provided as a Pandas DataFrame in json format.

`summary_table_json_flat`: str
Summary table of disparity calculation results. Provided as a list of dictionaries. Each dictionary represents a row in the table. Provides a flat representation of the table.

EnumBase

class `solas_disparity.types.`**`EnumBase`** (...)

Enum base class.

RatioCalculation

class `solas_disparity.types.`**`RatioCalculation`** (...)

Enum class denoting possible ratio calculations.

Attributes

`REFERENCE_OVER_PROTECTED`	
`PROTECTED_OVER_REFERENCE`	

ResidualSMDDenominator

class `solas_disparity.types.`**`ResidualSMDDenominator`** (...)

Enum class denoting possible SMD denominators.

Attributes

`POOLED`	
`POPULATION`	

SMDDenominator

class `solas_disparity.types.`**`SMDDenominator`** (...)

Enum class denoting possible SMD denominators.

Attributes

`POOLED`	
`POPULATION`	

ShortfallMethod

class `solas_disparity.types.`**`ShortfallMethod`** (...)

Enum class denoting possible shortfall methods.

Attributes

`TO_REFERENCE_MEAN`	
`TO_COMBINED_MEAN`	

StatSig

class `solas_disparity.types.`**`StatSig`** (...)

Dataclass for statistical significance test outputs.

Methods

`__init__`	Method generated by attrs for class StatSig.

Attributes

`stat_sig_test`	Statistical significance test.
`summary_table`	Summary table of results.

`stat_sig_test:` solas_disparity.types._stat_sig_test.StatSigTest

Statistical significance test.

summary_table: pandas.core.frame.DataFrame
Summary table of results.

StatSigRegressionType

class solas_disparity.types.**StatSigRegressionType** (...)

Enum class denoting possible regression types to be performed by stacked regression.

Attributes

GAUSSIAN	
LOGISTIC	

StatSigTest

class solas_disparity.types.**StatSigTest** (...)

Enum class denoting possible statistical significance tests.

Attributes

FISHERS_OR_CHI_SQUARED	
FISHERS_EXACT	
CHI_SQUARED_TEST	
TWO_SAMPLE_T_TEST	
STACKED_REGRESSION	
BOOTSTRAPPING	

2.1.23 solas_disparity.utils

Functions

pgrg_ordered	Create an ordered list of protected and reference groups.

pgrg_ordered

solas_disparity.utils.**pgrg_ordered** (...)

Create an ordered list of protected and reference groups.

Parameters

- **protected_groups** (*List[str]*) – List of protected groups.
- **reference_groups** (*List[str]*) – List of reference groups associated with each protected group.
- **group_categories** (*Optional[List[str]], optional*) – Group categories associated with each protected group. If None, ordered group categories are returned alongside unique group names. Defaults to None.

Returns List of ordered protected and reference groups. May also include group categories if group_categories is set.

Return type List[str]

Examples

```
>>> protected_groups = ["black", "asian", "female", "hispanic"]
>>> reference_groups = ["white", "white", "male", "white"]
>>> pgrg_orderd(protected_groups, reference_groups)
["black", "asian", "hispanic", "white", "female", "male"]
```

Chapter 3

As a PDF

Chapter 3

Chapter 4

Indices & Tables

- genindex
- modindex
- search