# themis-ml Documentation

***Release 0.0.2***

**Niels Bantilan**

**Feb 20, 2018**

# Contents

**themis-ml** is an open source machine learning library that implements several fairness-aware methods that comply with the sklearn API.

# CHAPTER 1

# Fairness-aware Machine Learning

**themis-ml** defines discrimination as the preference (bias) for or against a set of social groups that result in the unfair treatment of its members with respect to some outcome.

It defines fairness as the opposite of discrimination, and in the context of a machine learning algorithm, this is measured by the degree to which the algorithm's predictions favor one social group over another in relation to an outcome that holds socioeconomic, political, or legal importance, e.g. the denial/approval of a loan application.

An algorithm is "fair" depending on how we define fairness, the outcome of interest, and the socially sensitive attributes that relate to potentially discriminatory circumstances. For example, if we consider fairness as statistical parity, a fair algorithm is one in which the proportion of approved loans among minorities is equal to the proportion of approved loans among white people.

However, there are many other ways to define and operationalize fairness, and the purpose of **themis-ml** is to attempt to provide an interface that gives users with access to formalized definitions of fairness and discrimination described in the the machine learning and statistics literature. Check out this paper for more details.

# CHAPTER 2

# Install

You can install *themis-ml* with *conda* or *pip*. Currently only Python 2.7 and 3.6 are supported.

```
# conda
conda install -c cosmicbboy themis-ml
```

If you install with pip, you'll need to install scikit-learn, numpy, and pandas with either pip or conda. Version requirements:

- numpy (>= 1.9.0)
- scikit-learn (>= 0.19.1)
- pandas (>= 0.22.0)

```
# pip
pip install themis-ml
```

## 2.1 API

### 2.1.1 Datasets

`themis_ml.datasets.`**`german_credit`**(*raw=False*)

Load German Credit Dataset.

The target variable is "credit_risk", where 0 = bad and 1 = good

**Parameters** **`raw`** (*bool*) – If True, return raw data, otherwise return model-ready data. The model-ready data has columns arranged in the order of:

- numeric features.
- ordered categorical features.
- binary features.

- non-ordered categorical features.
- target.

Note: Raw data does not have this ordering, nor does it have dummified categorical variables.

**Returns** DataFrame of raw or model-ready data.

`themis_ml.datasets.`**`census_income`**`(`*raw=False*`)`

Load Census Income Data from 1994 - 1995.

The target variable is "income_gt_50k" (income above $50,000), where 0 is below and 1 is above.

**Parameters** **`raw`** (*`bool`*) – if True, return raw data, otherwise return model-ready data. The model-ready data has columns arranged in the the order of:

- numeric features.
- ordered categorical features.
- binary features.
- non-ordered categorical features.
- target.

**Returns** DataFrame of raw or model-ready data.

### 2.1.2 Metrics

Module for Fairness-aware scoring metrics.

`themis_ml.metrics.`**`abs_mean_difference_delta`**`(`*y*, *pred*, *s*`)`

Compute lift in mean difference between y and pred.

This measure represents the delta between absolute mean difference score in true y and predicted y. Values are in the range [0, 1] where the higher the value, the better. Note that this takes into account the reverse discrimintion case.

**Parameters**

- **`y`** (*`numpy.array`*) – shape (n, ) containing binary target variable, where 1 is the desireable outcome and 0 is the undesireable outcome.
- **`pred`** (*`numpy.array`*) – shape (n, ) containing binary predicted target, where 1 is the desireable outcome and 0 is the undesireable outcome.
- **`s`** (*`numpy.array`*) – shape (n, ) containing binary protected class variable where 0 is the advantaged groupd and 1 is the disadvantaged group.

**Returns** absolute difference in mean difference score between true y and predicted y

**Return type** float

`themis_ml.metrics.`**`abs_normalized_mean_difference_delta`**`(`*y*, *pred*, *s*`)`

Compute lift in normalized mean difference between y and pred.

This measure represents the delta between absolute normalized mean difference score in true y and predicted y. Values are in the range [0, 1] where the higher the value, the better. Note that this takes into account the reverse discrimintion case. Also note that the normalized mean difference score for predicted y's uses the true target for the normalization factor.

**Parameters**

- **y** (*numpy.array*) – shape (n, ) containing binary target variable, where 1 is the desireable outcome and 0 is the undesireable outcome.
- **pred** (*numpy.array*) – shape (n, ) containing binary predicted target, where 1 is the desireable outcome and 0 is the undesireable outcome.
- **s** (*numpy.array*) – shape (n, ) containing binary protected class variable where 0 is the advantaged groupd and 1 is the disadvantaged group.

**Returns** absolute difference in mean difference score between true y and predicted y

**Return type** float

`themis_ml.metrics.`**`mean_difference`**(*y*, *s*)

Compute the mean difference in y with respect to protected class s.

In the binary target case, the mean difference metric measures the difference in the following conditional probabilities:

mean_difference = p(y+ | s0) - p(y+ | s1)

In the continuous target case, the mean difference metric measures the difference in the expected value of y conditioned on the protected class:

mean_difference = E(y+ | s0) - E(y+ | s1)

Where y+ is the desireable outcome, s0 is the advantaged group, and s1 is the disadvantaged group.

Reference: Zliobaite, I. (2015). A survey on measuring indirect discrimination in machine learning. arXiv preprint arXiv:1511.00148.

**Parameters**

- **y** (*numpy.array*) – shape (n, ) containing binary target variable, where 1 is the desireable outcome and 0 is the undesireable outcome.
- **s** (*numpy.array*) – shape (n, ) containing binary protected class variable where 0 is the advantaged groupd and 1 is the disadvantaged group.

**Returns** mean difference between advantaged group and disadvantaged group with lower and uppoer confidence interval bounds.

**Return type** tuple[float]

`themis_ml.metrics.`**`mean_differences_ci`**(*y*, *s*, *ci=0.975*)

Calculate the mean difference and confidence interval.

**Parameters**

- **y** (*array-like*) – shape (n, ) containing binary target variable, where 1 is the desireable outcome and 0 is the undesireable outcome.
- **s** (*array-like*) – shape (n, ) containing binary protected class variable where 0 is the advantaged group and 1 is the disadvantaged group.
- **ci** (*float*) – % confidence interval to compute. Default: 97.5% to compute 95% two-sided t-statistic associated with degrees of freedom.

**Returns** mean difference between advantaged group and disadvantaged group with error margin.

**Return type** tuple[float]

`themis_ml.metrics.`**`normalized_mean_difference`**(*y*, *s*, *norm_y=None*, *ci=0.975*)

Compute normalized mean difference in y with respect to s.

Same the mean difference score, except the score takes into account the maximum possible discrimination at a given positive outcome rate. Is only defined when y and s are both binary variables.

normalized_mean_difference = mean_difference / d_max

where d_max = min( (p(y+) / p(s0)), ((p(y-) / p(s1)) )

The d_max normalization term denotes the smaller value of either the ratio of positive labels and advantaged observations or the ratio of negative labels and disadvantaged observations.

Therefore the normalized mean difference will report a higher score than mean difference in two cases: - if there are fewer positive examples than there are advantaged

observations.

- if there are fewer negative examples than there are disadvantaged observations.

Reference: Zliobaite, I. (2015). A survey on measuring indirect discrimination in machine learning. arXiv preprint arXiv:1511.00148.

**Parameters**

- **y** (*numpy.array*) – shape (n, ) containing binary target variable, where 1 is the desireable outcome and 0 is the undesireable outcome.
- **s** (*numpy.array*) – shape (n, ) containing binary protected class variable where 0 is the advantaged groupd and 1 is the disadvantaged group.
- **norm_y** (*numpy.array|None*) – shape (n, ) or None. If provided, this array is used to compute the normalization factor d_max.

**Returns** mean difference between advantaged group and disadvantaged group with lower and upper confidence interval bounds

**Return type** tuple(float)

### 2.1.3 Preprocessing

### 2.1.4 Linear Models

**class** themis_ml.linear_model.**LinearACFClassifier**(*target_estimator=LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1, penalty='l2', random_state=None, solver='liblinear', tol=0.0001, verbose=0, warm_start=False), continuous_estimator=LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False), binary_estimator=LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1, penalty='l2', random_state=None, solver='liblinear', tol=0.0001, verbose=0, warm_start=False), binary_residual_type='pearson'*)

**fit**(*X, y, s*)
Fit model.

**predict**(*X, s*)
Generate predicted labels.

**predict_proba**(*X, s*)
Generate predicted probabilities.

### 2.1.5 Postprocessing

### 2.1.6 Meta-estimators

Module for Fairness-aware base estimators.

### 2.1.7 Utilities

Utility functions for computing useful statistics.

themis_ml.stats_utils.**deviance_residuals**(*y, pred*)
Compute Deviance residuals.

Reference: https://web.as.uky.edu/statistics/users/pbreheny/760/S11/notes/4-12.pdf

Formula: d = sign * sqrt(-2 * {y * log(p) + (1 - y) * log(1 - p)}) - where sign is -1 if y = 1 and 1 if y = 0 - y is the true label - p is the predicted probability

**Parameters**

- **y** (*array-like[int]*) – target labels. 1 is positive label, 0 is negative label

- **pred** (*array-like[float]*) – predicted labels.

**Returns** deviance residual.

**Return type** array-like[float]

`themis_ml.stats_utils.`**`pearson_residuals`**(*y*, *pred*)

Compute Pearson residuals.

Reference: https://web.as.uky.edu/statistics/users/pbreheny/760/S11/notes/4-12.pdf

**Parameters**

- **y** (*array-like[int]*) – target labels. 1 is positive label, 0 is negative label
- **pred** (*array-like[float]*) – predicted labels.

**Returns** pearson residual.

**Return type** array-like[float]

Utility functions for doing checks.

# CHAPTER 3

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

# Index