Fairness for Scikit-Learn Pipelines with Lale

Martin Hirzel, IBM Research

PyData NYC November 2022

Motivation

- Goal: fair machine-learning pipelines
  - Avoid bias based on race, gender, age, …
  - Reasons: laws, regulations, values, reputation, business, …
- In scope: algorithmic fairness metrics and mitigators
  - Tabular data with protected attributes
  - Binary classification
  - Noisy trade-offs
- Important but out of scope: societal issues
  - No consensus on which techniques are right
  - Conflicting world views
  - This talk lists options, but the choice is yours

Fairness and Data

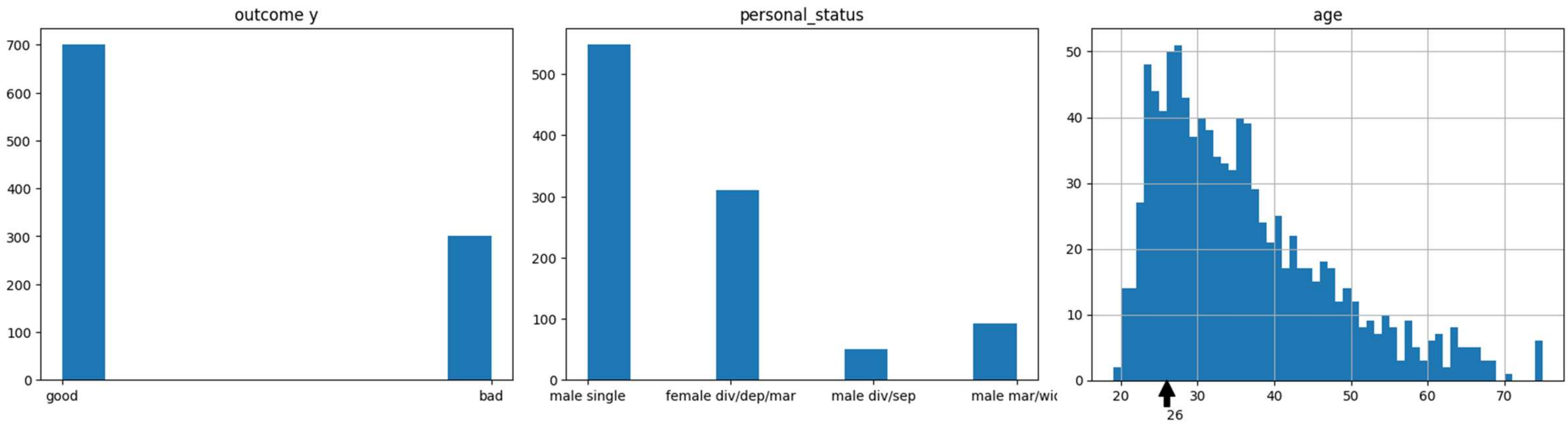
credit-g Dataset

	y	X									
	label	protected attributes									
	class	personal_status	age	checking_status	duration	credit_history	purpose	credit_amount	savings_status	employment	...
0	good	male single	67.0	<0	6.0	critical/other existing credit	radio/tv	1169.0	no known savings	>=7	...
1	bad	female div/dep/mar	22.0	0<=X<200	48.0	existing paid	radio/tv	5951.0	<100	1<=X<4	...
2	good	male single	49.0	no checking	12.0	critical/other existing credit	education	2096.0	<100	4<=X<7	...
3	good	male single	45.0	<0	42.0	existing paid	furniture/equipment	7882.0	<100	4<=X<7	...
4	bad	male single	53.0	<0	24.0	delayed previously	new car	4870.0	<100	1<=X<4	...
...	...	...	...	...	...	...	...	...	...	...	...
995	good	female div/dep/mar	31.0	no checking	12.0	existing paid	furniture/equipment	1736.0	<100	4<=X<7	...
996	good	male div/sep	40.0	<0	30.0	existing paid	used car	3857.0	<100	1<=X<4	...
997	good	male single	38.0	no checking	12.0	existing paid	radio/tv	804.0	<100	>=7	...
998	bad	male single	23.0	<0	45.0	existing paid	radio/tv	1845.0	<100	1<=X<4	...
999	good	male single	27.0	0<=X<200	45.0	critical/other existing credit	used car	4576.0	100<=X<500	unemployed	...

1000 rows × 21 columns

Fairness Meta-Data

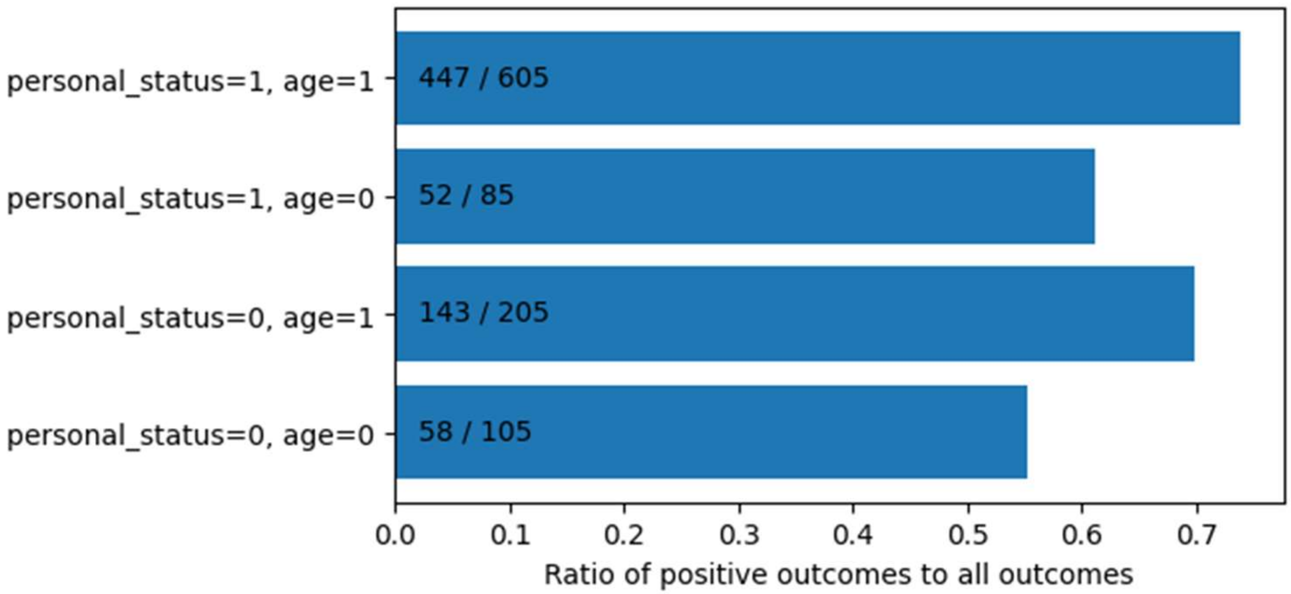
```
fairness_info = {
  'favorable_labels': ['good'],
  'protected_attributes': [
    { 'feature': 'personal_status',
      'reference_group': ['male div/sep', 'male mar/wid', 'male single']},
    { 'feature': 'age', 'reference_group': [[26, 1000]]}]}
```



Groups and Intersections

```
fairness_info = {
  'favorable_labels': ['good'],
  'protected_attributes': [
    { 'feature': 'personal_status',
      'reference_group': ['male div/sep', 'male mar/wid', 'male single']},
    { 'feature': 'age', 'reference_group': [[26, 1000]]}]}
```

Groups based on binary encoding of protected attributes and outcomes



Axiomatic Assumptions

- Source: Friedler/Scheidegger/Venkatasubramanian. *The (Im)Possibility of Fairness: Different Value Systems Require Different Mechanisms for Fair Decision Making.* CACM 2021.
- WAE (We're All Equal)
  - All groups are essentially the same
  - If groups differ in the dataset, that is caused by structural bias
  - Motivates group fairness
- WYSIWYG (What You See Is What You Get)
  - Features and labels in the dataset accurately reflect construct space
  - Motivates individual fairness
- Algorithms cannot guarantee both WYSIWIG and WAE

Protected Attributes Influence Outcomes

sklearn.inspection.permutation_importance

	feature	importance	std
0	checking_status	0.086000	0.011189
1	credit_amount	0.076400	0.004923
2	duration	0.063600	0.006681
3	credit_history	0.053600	0.006681
4	purpose	0.044800	0.002713
5	age	0.027600	0.002154
6	savings_status	0.021800	0.004534
7	other_parties	0.014800	0.001166
8	other_payment_plans	0.014600	0.002332
9	residence_since	0.014000	0.002608
10	personal_status	0.009400	0.001020
11	employment	0.009200	0.003709
12	housing	0.007600	0.001497
13	job	0.006400	0.001744
14	property_magnitude	0.005600	0.002577
15	existing_credits	0.005600	0.001356
16	installment_commitment	0.004400	0.002332
17	own_telephone	0.004200	0.002135
18	foreign_worker	0.003400	0.000490
19	num_dependents	-0.001800	0.001327

Other Attributes can Predict Protected Attributes

- Derived dataset where the binarized protected attribute is the target and removed from features
- Balanced accuracy 0.612 to predict personal_status group
- Balanced accuracy 0.640 to predict age group
- Redaction would avoid disparate treatment but not disparate impact

personal_status

sklearn.inspection.permutation_importance

	feature	importance	std
0	credit_amount	0.055200	0.007250
1	age	0.051200	0.007250
2	employment	0.038000	0.006481
3	purpose	0.036600	0.001200
4	housing	0.032800	0.001720
5	installment_commitment	0.024200	0.002926
6	residence_since	0.018800	0.003187
7	num_dependents	0.018400	0.004224
8	checking_status	0.012600	0.003200
9	duration	0.011800	0.001939
10	credit_history	0.008400	0.002154
11	savings_status	0.005600	0.002245
12	existing_credits	0.004600	0.002245
13	property_magnitude	0.004400	0.003611
14	job	0.002600	0.002059
15	own_telephone	0.002200	0.000400
16	other_payment_plans	0.002000	0.001673
17	other_parties	0.000800	0.000400
18	foreign_worker	0.000400	0.001625

age

sklearn.inspection.permutation_importance

	feature	importance	std
0	housing	0.052000	0.005550
1	personal_status	0.034800	0.007679
2	credit_amount	0.031200	0.001470
3	employment	0.025000	0.003406
4	purpose	0.014400	0.003611
5	credit_history	0.012000	0.001414
6	job	0.012000	0.003578
7	checking_status	0.010200	0.001720
8	residence_since	0.009200	0.002786
9	foreign_worker	0.006600	0.001855
10	num_dependents	0.006200	0.003059
11	own_telephone	0.006200	0.004956
12	duration	0.005200	0.003187
13	property_magnitude	0.004400	0.002577
14	installment_commitment	0.002600	0.000800
15	existing_credits	0.002200	0.001720
16	savings_status	0.001600	0.001356
17	other_payment_plans	0.000800	0.000400
18	other_parties	0.000200	0.000400

Fairness and Metrics

Scikit-Learn Metrics and Scoring APIs

```
# scorer
ba_scorer = sklearn.metrics.make_scorer(sklearn.metrics.balanced_accuracy_score)
ba_scorer(trained, test_X, test_y)

# cross-validation
sklearn.model_selection.cross_val_score(
    sklearn.base.clone(trainable), train_X, train_y,
    scoring=ba_scorer, cv=StratifiedKFold(3))

# grid search
grid_search = sklearn.model_selection.GridSearchCV(
    sklearn.base.clone(trainable),
    param_grid={
        "gradientboostingclassifier__n_estimators": [1, 10, 100, 1000]},
    scoring=ba_scorer, cv=StratifiedKFold(3))
grid_search = grid_search.fit(train_X, train_y)
```

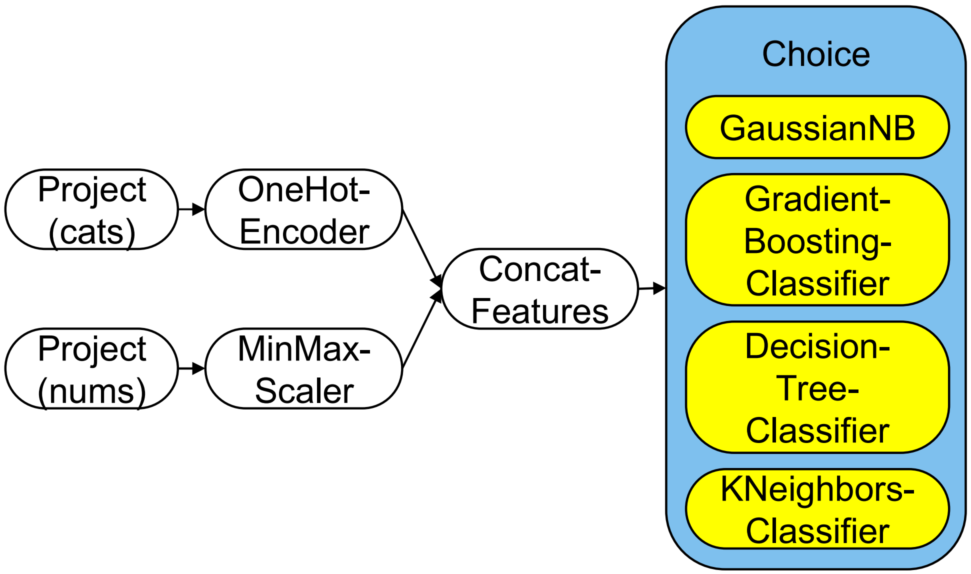
Fairness Metrics

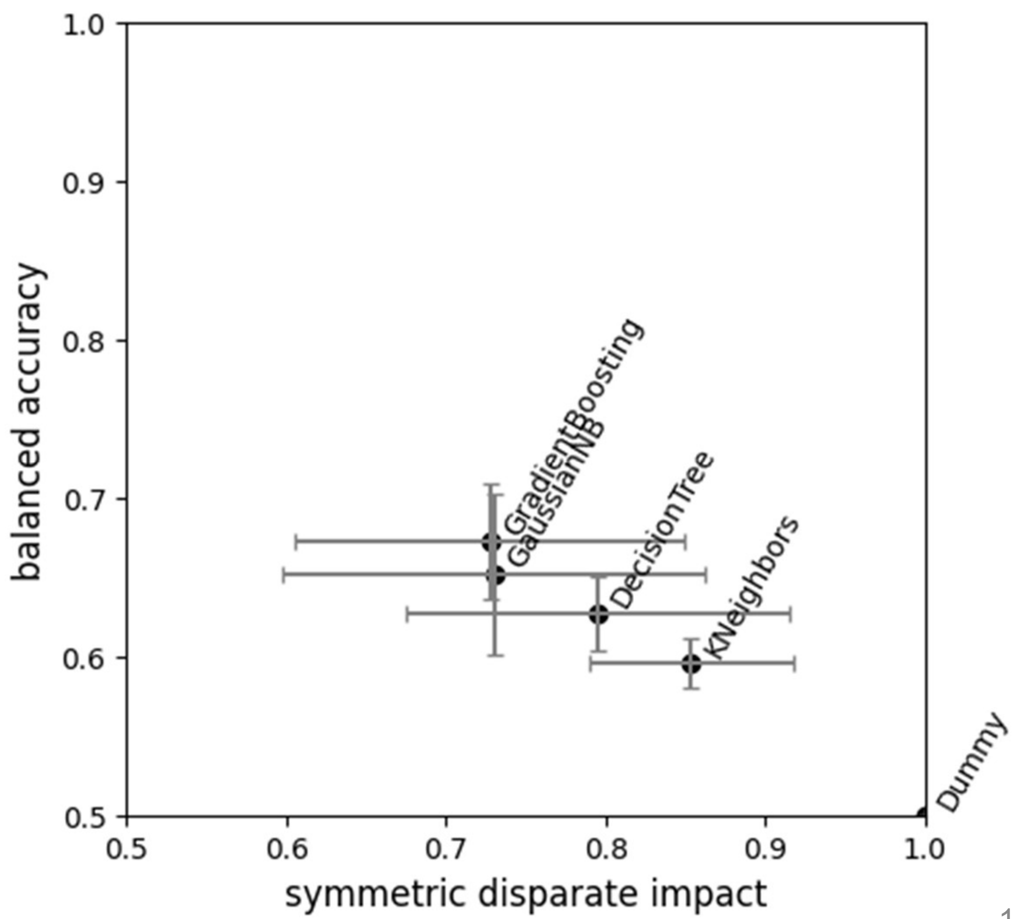
Metric	Formula	Inputs	Ideal	Thresholds
Disparate impact	$\Pr(y = 1 \mid X_p = 0)$ / $\Pr(y = 1 \mid X_p = 1)$	Protected attributes X_p, Labels y	1	Unfair to $X_p = 0$: <0.8 Unfair to $X_p = 1$: >1.25
Symmetric disparate impact	di **if** $di \leq 1$ **else** $1/di$	Protected attributes X_p, Labels y	1	Unfair: <0.8
Statistical parity difference	$\Pr(y = 1 \mid X_p = 0)$ $- \Pr(y = 1 \mid X_p = 1)$	Protected attributes X_p, Labels y	0	Unfair to $X_p = 0$: <–0.1 Unfair to $X_p = 1$: >+0.1
Equal opportunity difference	$\Pr(\hat{y}=y=1 \mid X_p = 0)$ $- \Pr(\hat{y}=y=1 \mid X_p = 1)$	Protected attributes X_p, Ground-truth labels y, Predicted labels $\hat{y}$	0	Unfair to $X_p = 0$: <–0.1 Unfair to $X_p = 1$: >+0.1
Theil index	$\mathrm{E}(\hat{y} - y + 1)$	Ground-truth labels y, Predicted labels $\hat{y}$	1	Too much benefit: >>1 Too little benefit: <<1

Scikit-Learn Compatible Fairness Metrics

```
# dataset fairness meta-data
fairness_info = {
  'favorable_labels': ['good'],
  'protected_attributes': [
    { 'feature': 'personal_status',
      'reference_group': ['male div/sep', 'male mar/wid', 'male single']},
    { 'feature': 'age', 'reference_group': [[26, 1000]]}]}

# scorer
di_scorer = lale.lib.aif360.symmetric_disparate_impact(**fairness_info)

# … use like other scorer for things like cross-validation, grid-search, …
```

Pipelines without Mitigators





Error bars based on 10 random 75% : 25% splits

Fairness and Pipelines

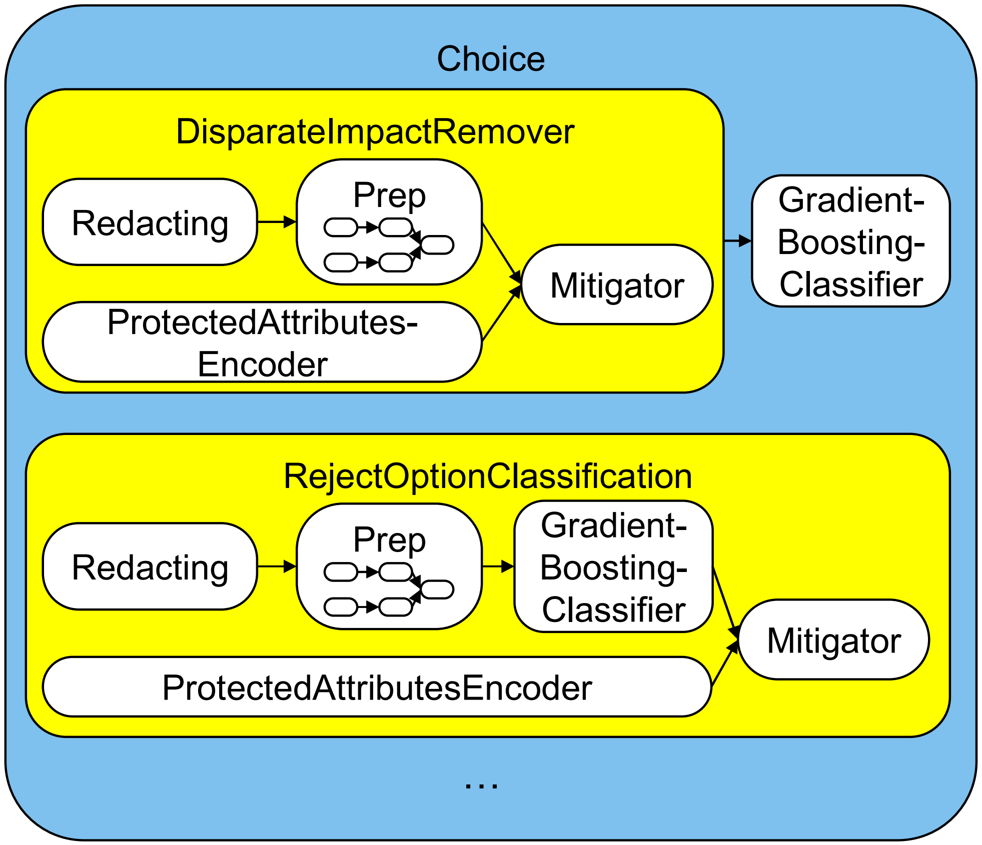
Bias Mitigators

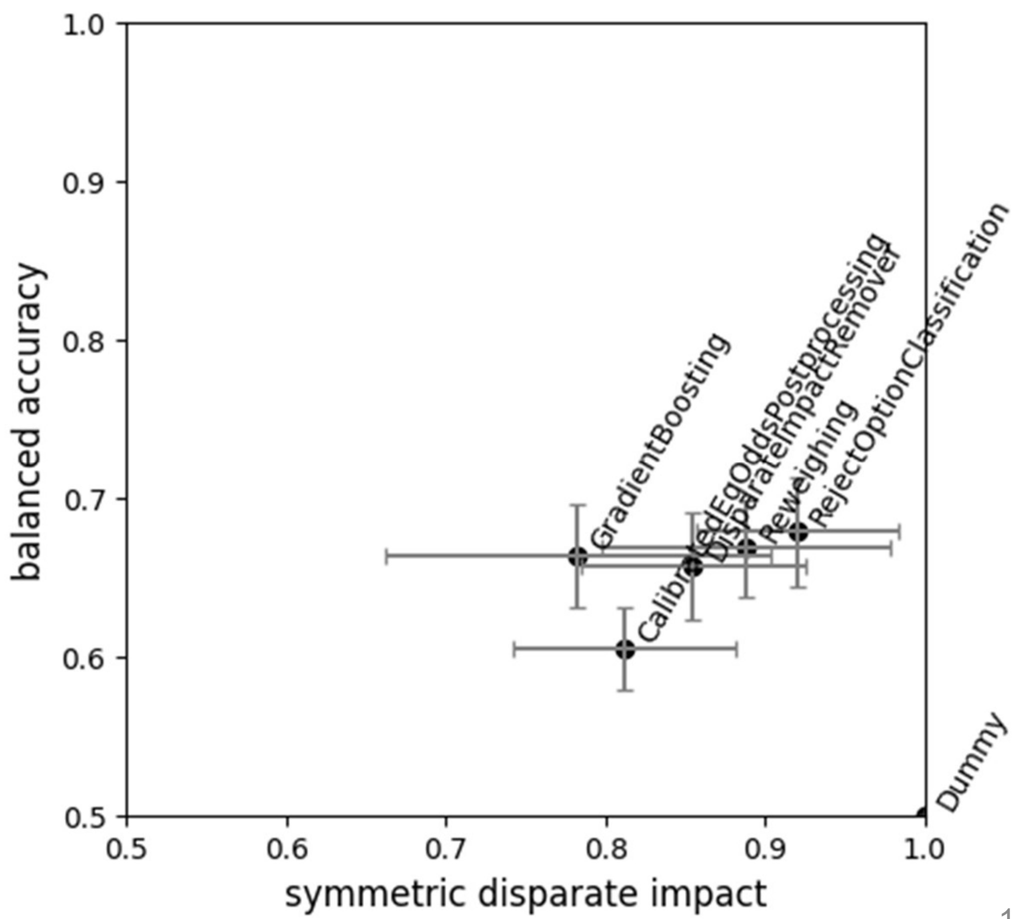
Mitigator	Kind	Description	Reference
Disparate impact remover	Pre-estimator	Separately shift distribution of each non-protected attribute so it is not correlated with protected attributes	Feldman/Friedler/Moelle/Scheidegger/Venkatasu-bramanian 2015
Reweighing	Pre-estimator	Increase sample weights for training data instances so the groups have equal total positive instance weight	Kamiran/Calders 2012
Calibrated equalized odds postprocessing	Post-estimator	Randomly flip some predictions near the decision boundaries based on group membership	Pleiss/Raghavan/Wu/Kleinberg/Weinberger 2017
Reject-option classification	Post-estimator	Deterministically flip some predictions near the decision boundaries based on group membership	Kamiran/Karim/Zhang 2012

Bias Mitigators in Python

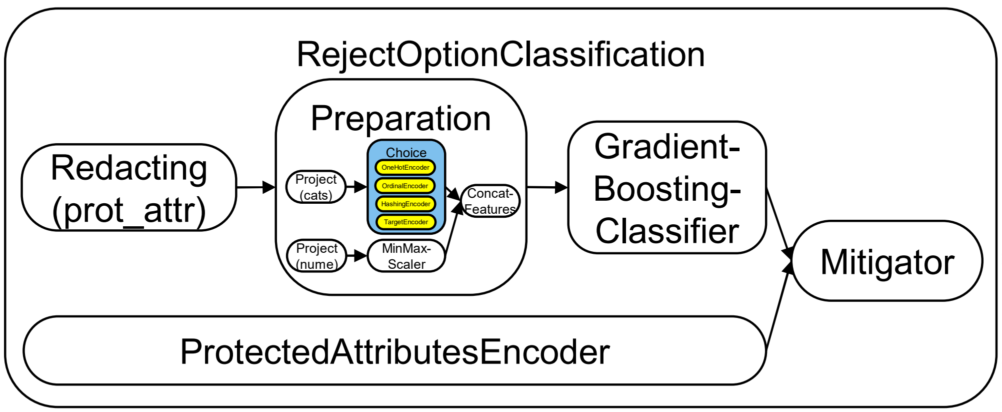
```
# dataset fairness meta-data
fairness_info = {
  'favorable_labels': ['good'],
  'protected_attributes': [
    { 'feature': 'personal_status',
      'reference_group': ['male div/sep', 'male mar/wid', 'male single']},
    { 'feature': 'age', 'reference_group': [[26, 1000]]}]}

# pipeline with bias mitigator, data preparation, and final estimator
trainable = lale.lib.aif360.DisparateImpactRemover(
    **fairness_info,
    preparation=(
        (Project(columns=cat_columns) >> OneHotEncoder(handle_unknown="ignore"))
        & (Project(columns=num_columns) >> MinMaxScaler())
      ) >> ConcatFeatures
  ) >> GradientBoostingClassifier()
```

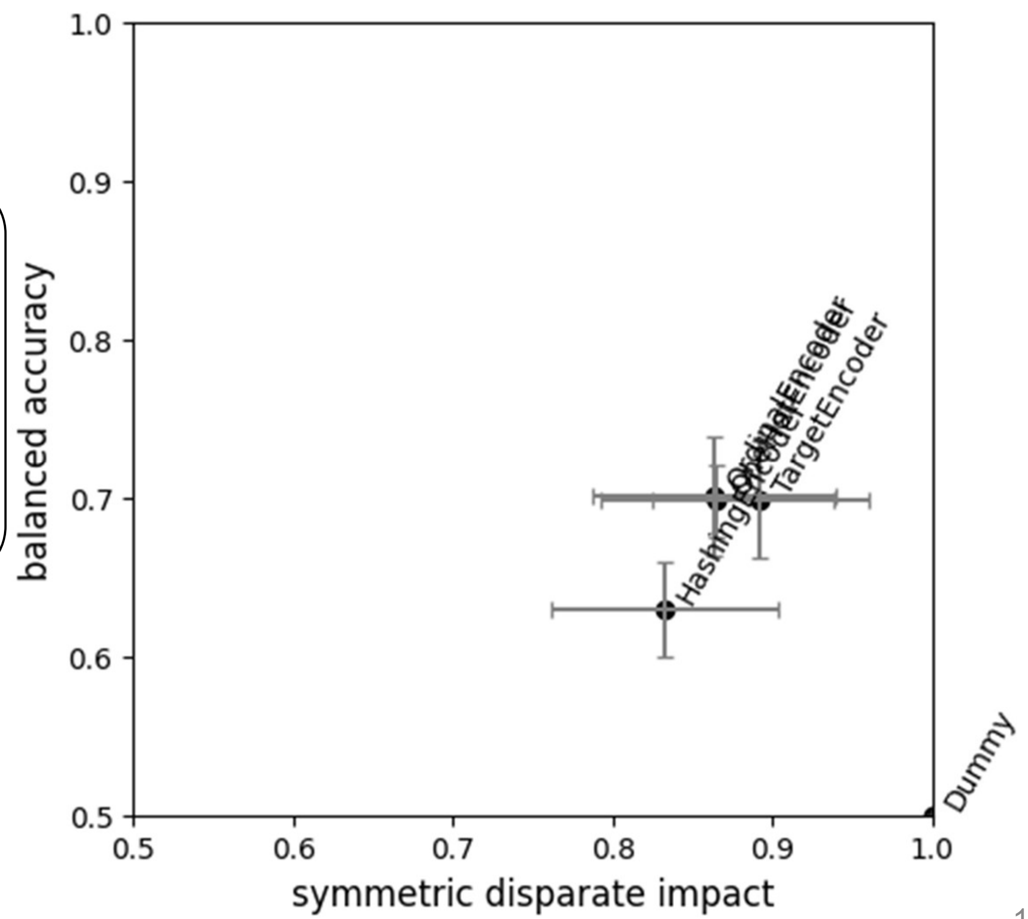
Bias Mitigators



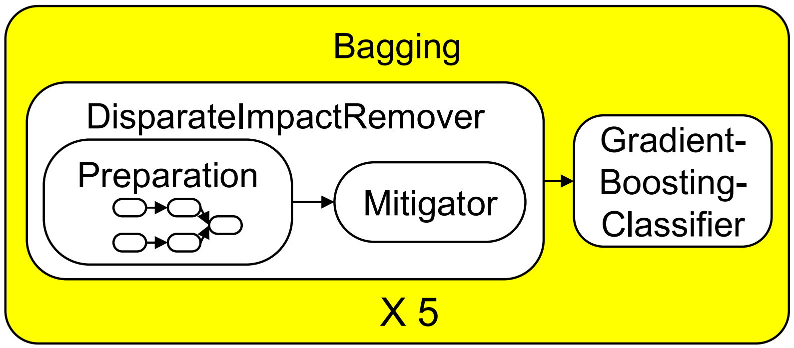


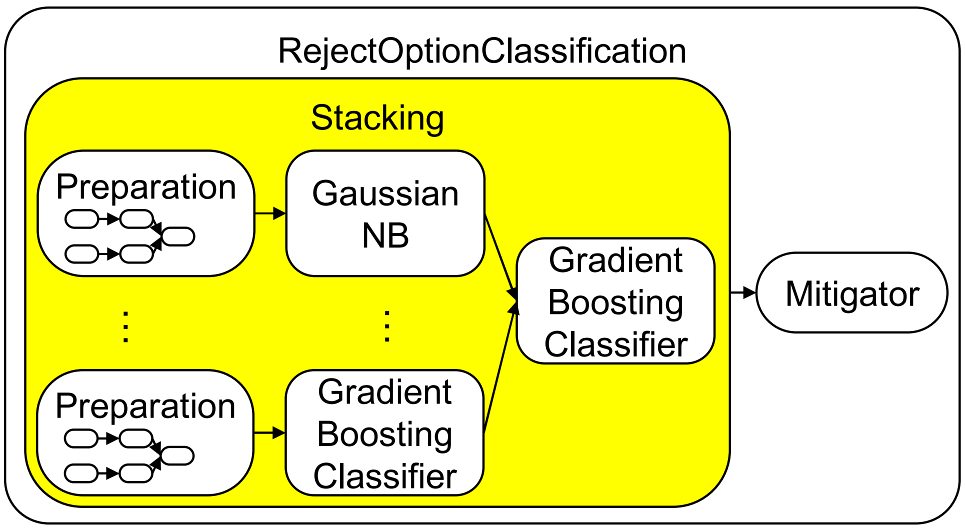
Mitigators and Preprocessing

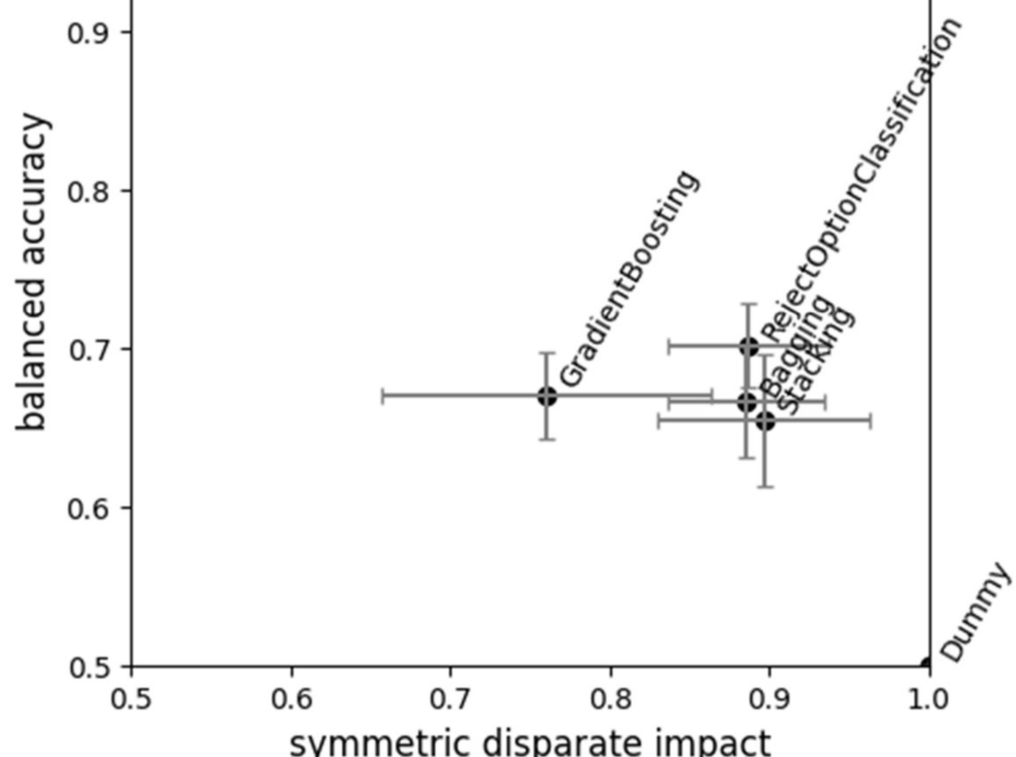




Mitigators and Ensembles







Fairness and AutoML

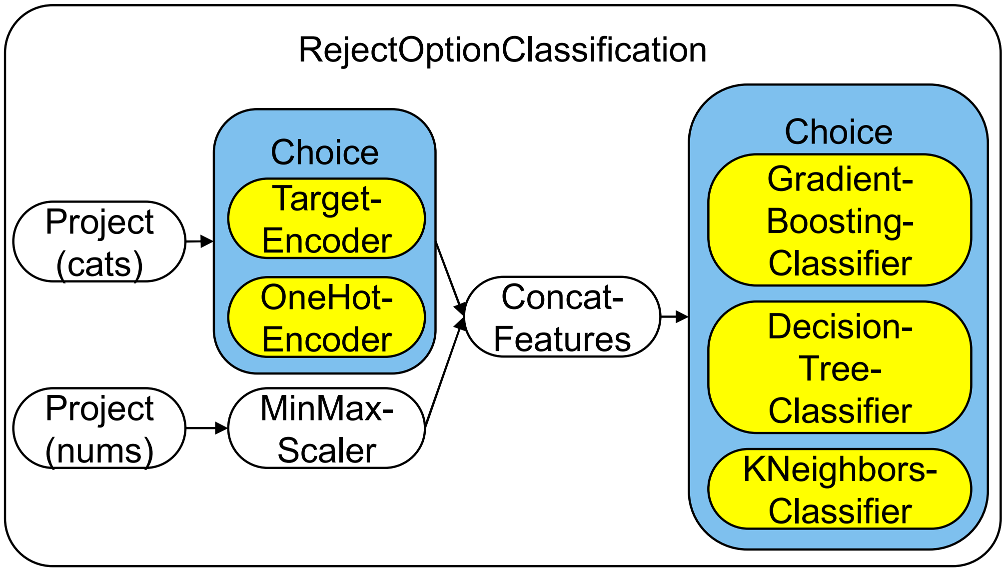
Challenges for Fairness and AutoML

Challenge	Solution in this talk	Other solutions
Multiple objectives	• Blend into single objective via harmonic mean • Show x-y scatter	• Multi-objective optimizer • Different blending strategies • Maximize one objective and threshold the other
Noise	• Show error bars • Repeated k-fold cross validation • Stratification by both outcomes and protected attribute groups	• Use larger ensembles

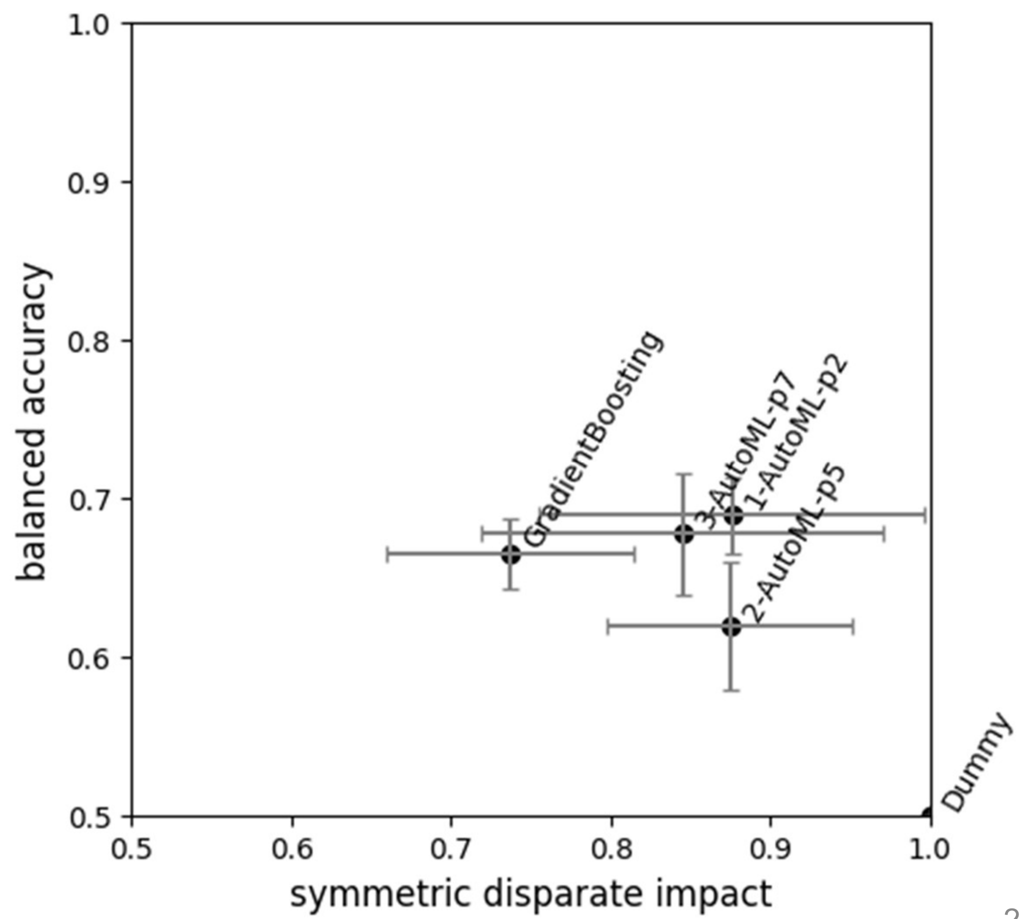
AutoML Search Space in Python

```
planned_est_choice = lf.RejectOptionClassification(
    **fairness_info,
    estimator=
        (   (     Project(columns=cat_columns)
                >> (TargetEncoder | OneHotEncoder(handle_unknown="ignore"))
            )
          & (     Project(columns=num_columns)
                >> MinMaxScaler
            )
        )
      >> ConcatFeatures
      >> (   DecisionTreeClassifier
          | KNeighborsClassifier
          | GradientBoostingClassifier
        )
)
```

Mitigators and AutoML



Rank	Name	Encoder	Estimator
1	p2	TargetEncoder	GradientBoostingClassifier
2	p5	TargetEncoder	KNeighborsClassifier
3	p7	OrdinalEncoder	GradientBoostingClassifier



Conclusion

- Fairness value judgements are beyond the scope of this talk
  - E.g., WYSIWYG vs. WAS, disparate treatment vs. disparate impact, …
  - Useful to know algorithms for accomplishing fairness goals
- This talk discussed various fairness metrics and bias mitigators
  - Fairness and data
  - Fairness metrics
  - Bias mitigators
  - Fairness and AutoML
- Lale library: https://github.com/ibm/lale
  - Try it out
  - Contribute