I

Technologies for Information Systems

The problem of Fairness and Data Bias

De Berardinis | Guadagno

FAIRNESS

is not always ensured
by algorithms

data often reflects **discrimination**
which is cause of **unfairnes**

aim

- discover BIAS in data

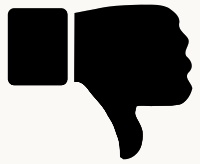
to avoid
unintentional
unethical behaviors

What is a data bias?

It is a systematic error.

privileged groups → advantage

unprivileged groups → disadvantage



Amazon

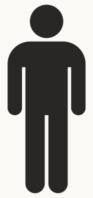
2014

Machine learning experts started working on a system to review job applicants' resumes

scores from 0 to 5

2015

Gender bias towards male candidates



due to *historical discrimination* in training data

Where unfairness can be a problem ?

Unfairness in algorithms can create issues not only in the *hiring process*, but also in other contexts

- College rankings

- Criminal risk assesement

- Finanacial services

What is the problem?

The problem is not the algorithm but
the *training data*
used to train the system

Amazon example:

GENDER

should not influence hiring decisions

- Gender
- Religion
- Race

are the *Protected attributes*

↓

should not be taken into
account by algorithms
that evaluate

qualities

skills

RANKING FACTS

from: ***A nutritional label for ranking***

RANKING FACTS

What is it?

It is a **web-based** application that generates a " *nutritional label* " for rankings.

It is a collection of **visual widgets**

transparency

interoperability

What are ***nutritional labels?***

from Food Industry

standardized labels

conveying information to consumers about the ingredients and production processes

GOAL

explain

- ranked outputs to a user
- how the output is obtained

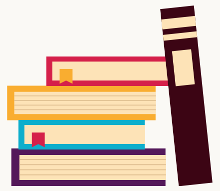
to achieve **transparency** and **fairness**

TECHNIQUE

PRE - PROCESSING

procedure that analyze data before using it to train a classifier

Widgets

◆ Recipe 

-describes **ranking algorithm** listing each attribute together with its **weight**
-describes explicit **intentions** of the designer

◆ Ingredients 

- lists attributes in order of importance
-shows additional attributes associated with high rank

Observe

many attributes in the Recipe do not coincide with those that most impact the ranked outcome in the Ingredients

example: *attribute GRE*

Recipe

Attribute	Weight
PubCount	1.0
Faculty	1.0
GRE	1.0

Ingredients

Attribute	Importance
PubCount	1.0
CSRankingAllArea	0.24
Faculty	0.12

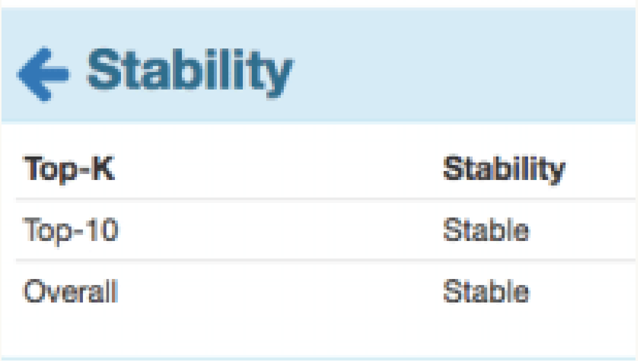
Widgets

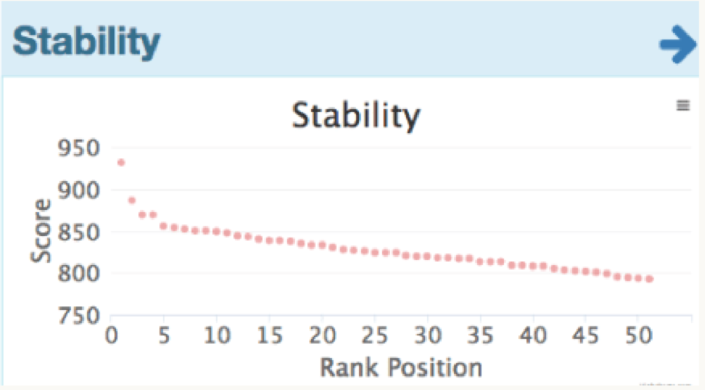
- Stability 

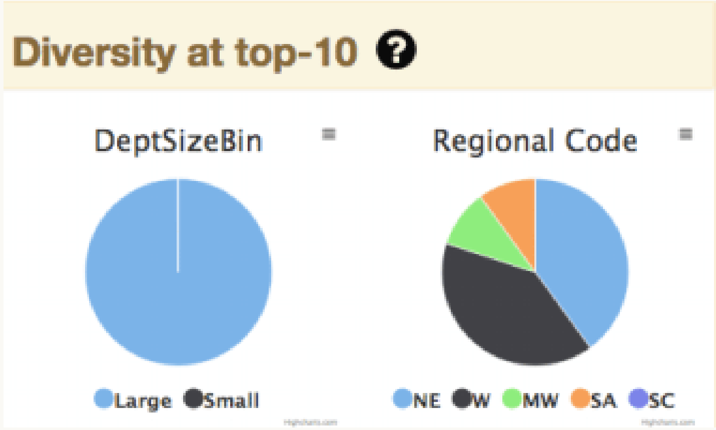
Explains whether the ranking method is *stable:* a slight change in the data should **not** lead to a significant change in the output (*unstable ranking*)

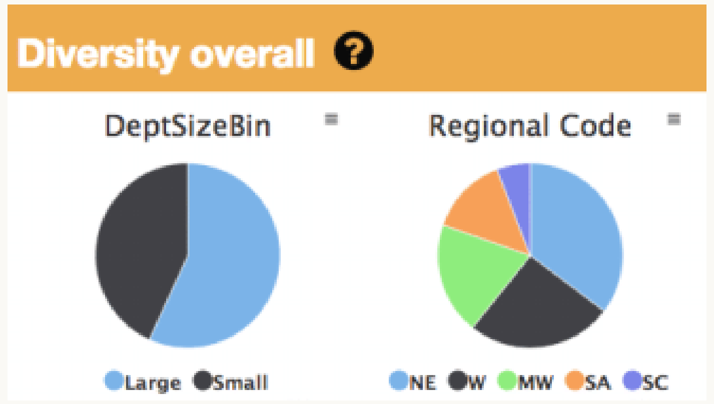
- Diversity 

shows diversity with respect to a set of **demographic categories of individuals** or a set of **categorical attributes** of other kinds of items









Widgets

- Fairness

quantifies whether the ranked output exhibits **statistical parity** with respect to one or more sensitive attributes, like gender or race

Fairness

DeptSizeBin	FA*IR	Pairwise	Proportion
Large	Fair	Fair	Fair
Small	Unfair	Unfair	Unfair

FA*IR

ensures that the proportion of protected candidates of the top-k ranking remains **above a certain minimum**

Proportion

compares the proportion of members of a protected group who received a **positive outcome** to their proportion in the overall population

Pairwise

models the probability that a member of a **protected group is preferred to** a member of the **non-protected group**

FAIR DB

from: **Functional Dependencies to discover Data Bias**

Functional dependencies to discover data bias

FAIR DB

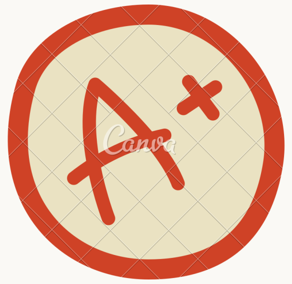
What is it?

framework that uses Approximate conditional Functional Dependendencies (ACFDs) to detect biases and discover discrimination and unfair behaviours in the datasets

What is a Functional Dependency ?

A class of database integrity constraints
that specifies that the values of the attributes of X uniquely
(or functionally) determine the values of the attributes of Y.

{ StudID , ExamID } -> { Grade }



The constraints of Functional Dependencies are often **too strict** for real world datasets

Relaxed Functional Dependencies

Approximate functional dependencies (AFDs)

hold only on a **subset** of tuples in the database

Conditional functional dependencies (CFDs)

conditions are used to specify tuples on which dependencies hold

Approximate Conditional Functional Dependendencies

GOAL

solves unfairness by recognizing cases in which the value of a **certain attribute determines the value of another one**

TECHNIQUE

PRE - PROCESSING

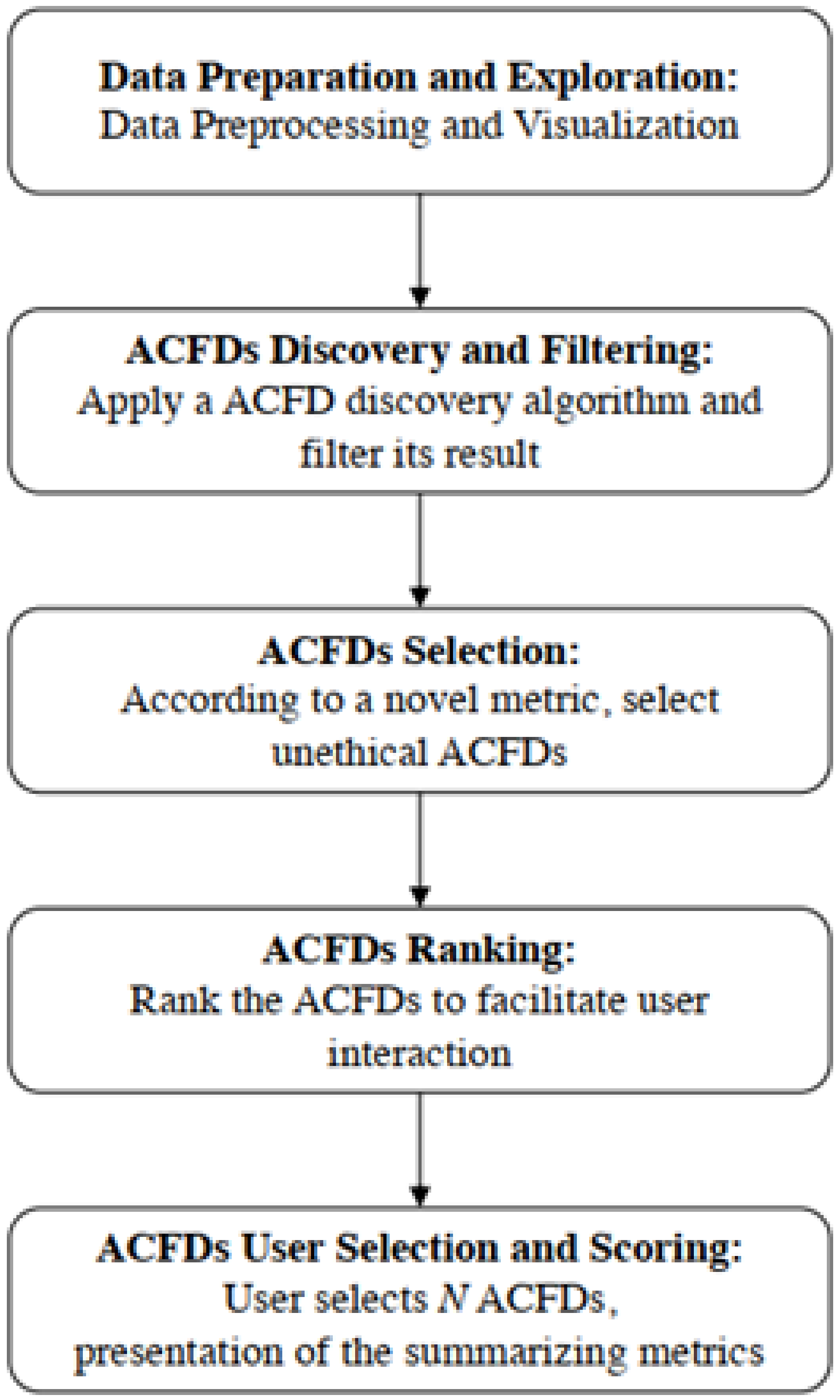


Figure 1: Steps of the *FAIR-DB* framework

1) Data Preparation and Exploration

- Data **acquisition**

- Computation of **Summary Statistics** to have a general idea of the dataset

-> Possibility to hypothesize the protected columns and identify, if present, the **target variable** Y

- Data **cleaning** and data **integration**
- Features **selection** and **discretization**

-> Possibility to visualize the attribute features using different **Data Visualization techniques**

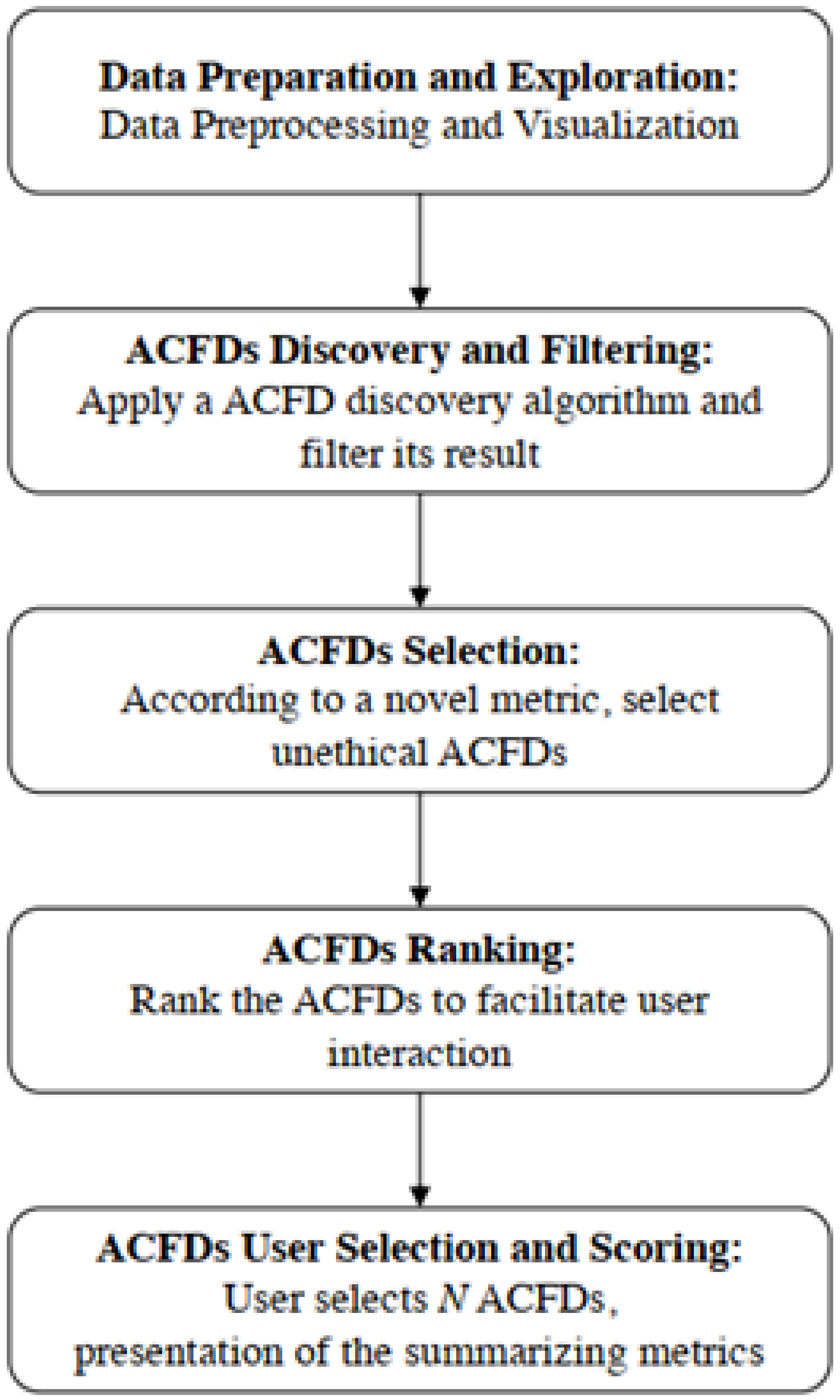


Figure 1: Steps of the *FAIR-DB* framework

2) ACFD Discovery and Filtering

- Extraction of ACFDs from the dataset
- Filtering: in order to select only useful ACFDs.

Given the **CFD X → Y,tp**, 3 inputs are needed :

Minimum support: proportion of tuples in the dataset D which contain tp (respects the condition).

Minimum confidence: proportion of the tuples t containing X that also contain Y

Maximum antecedent size

- Dependencies that contain variables are discarded

3) ACFDs Selection

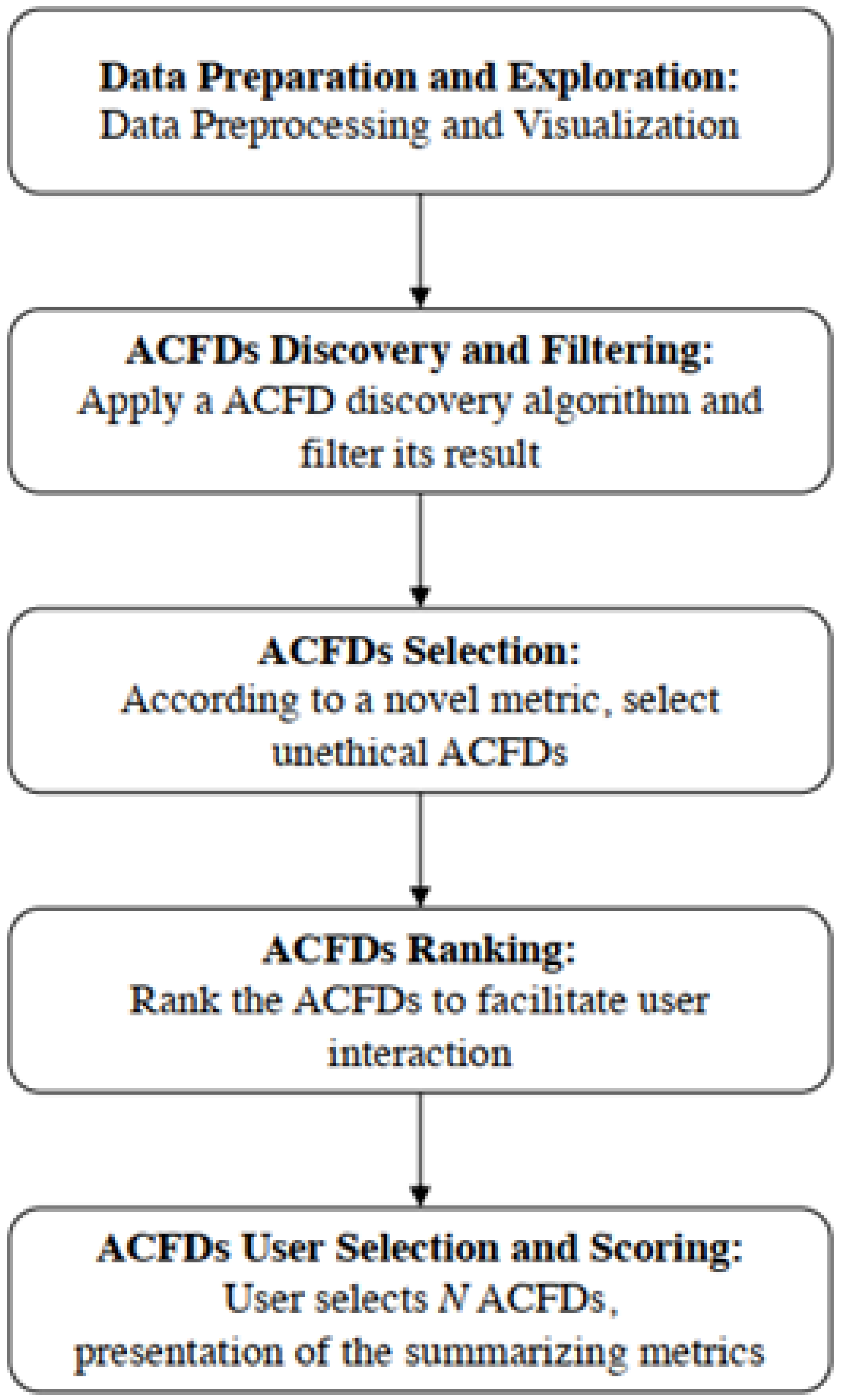


Figure 1: Steps of the *FAIR-DB* framework

Finding the dependencies that actually reveal unfairness in the dataset.

Difference metric: difference between the dependency confidence and the dependency confidence computed without the protected attributes of the ACFD.

High Difference metric = Unfair Behaviour

Since there can be more than one protected attribute at the same time, it is possible to compute for each protected attribute p its specific **p-Difference**

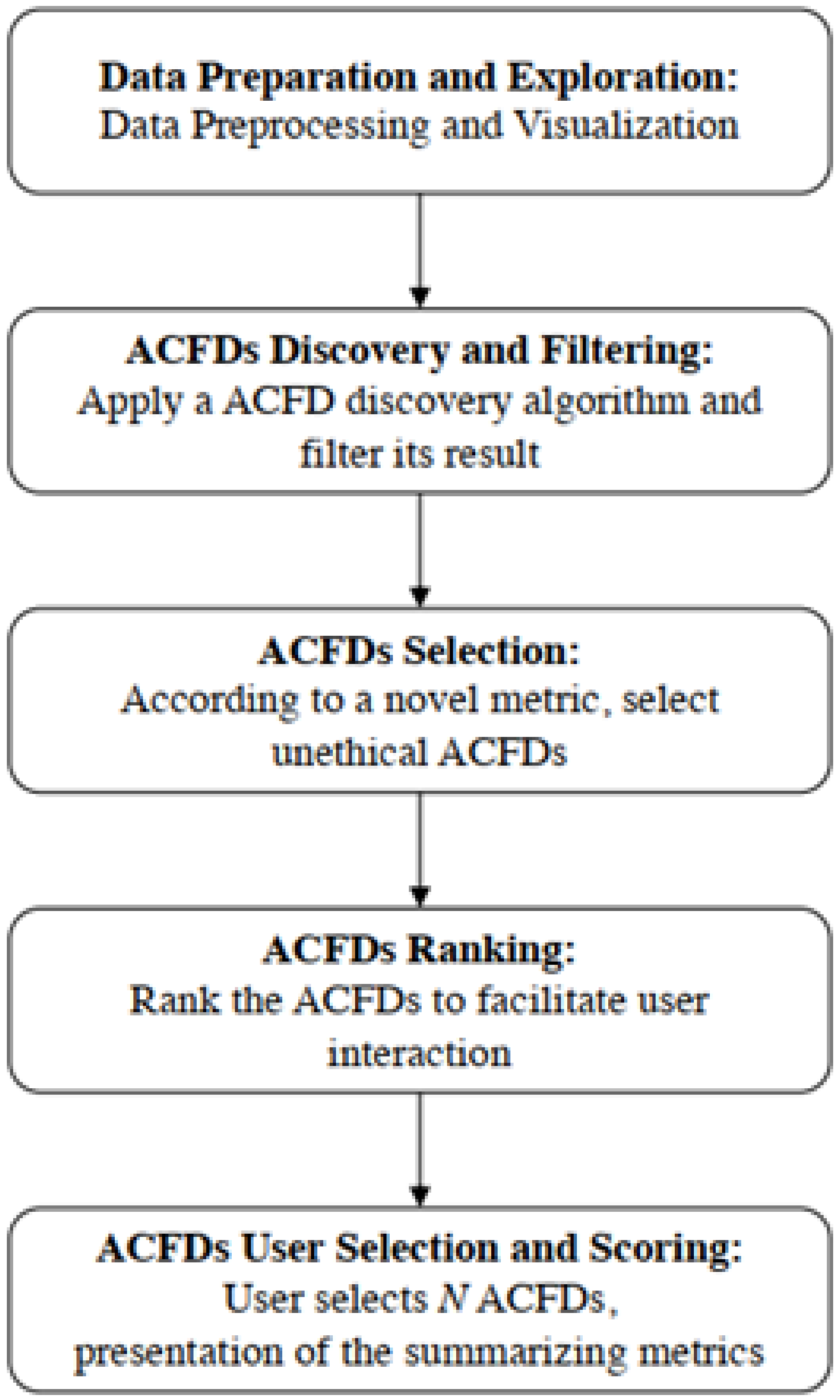


Figure 1: Steps of the *FAIR-DB* framework

4) ACFDs Ranking

ACFDs are ranked in descending order to facilitate **user interaction** according to one of the following criteria:

Support-based

Difference-based

Mean-based (tries to combine both aspects)

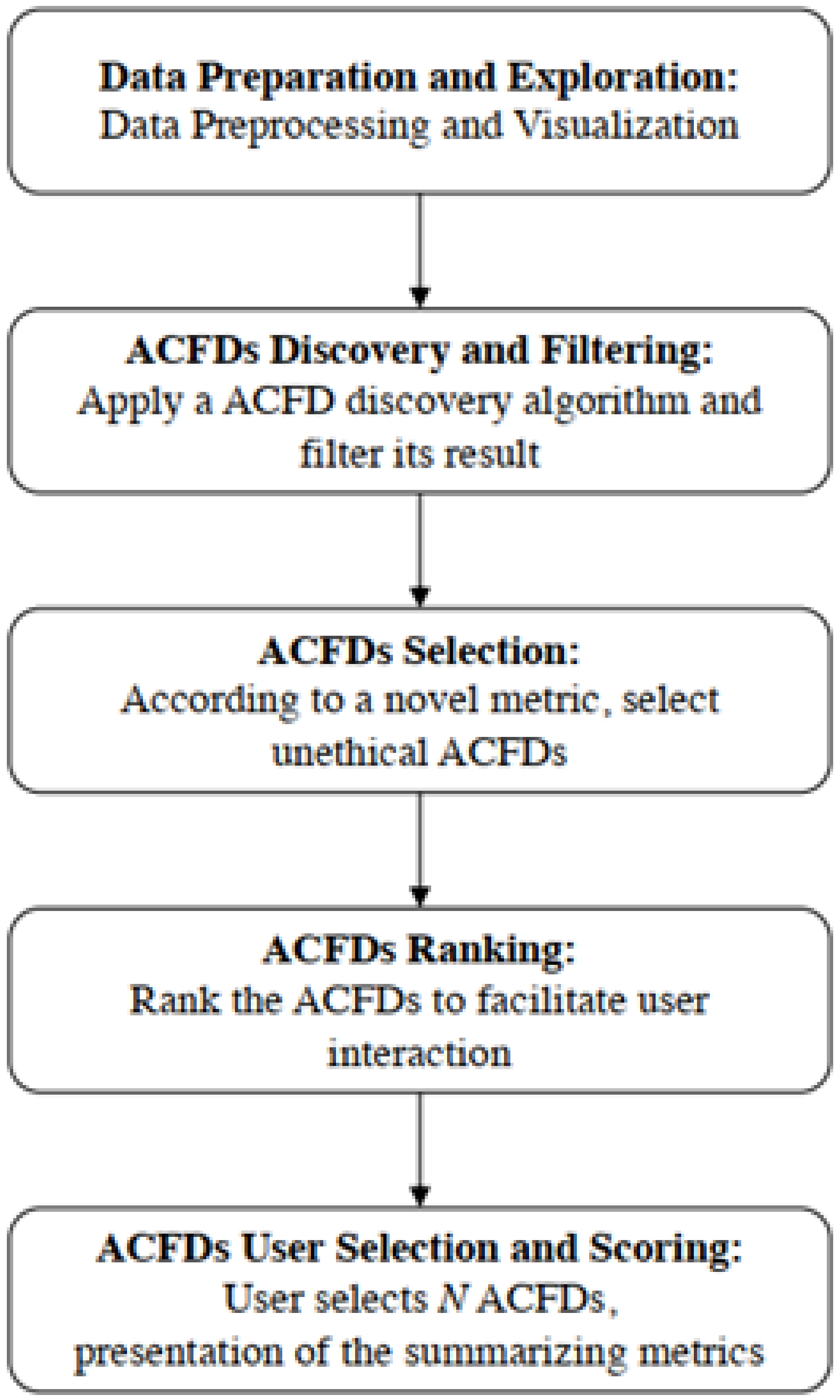


Figure 1: Steps of the *FAIR-DB* framework

5) ACFDs Selection and Scoring

the user selects from the ranked list N dependencies that are interesting for the research needs.

Using the N selected ACFDs, the framework computes:

Cumulative Support

Difference Mean

Protected Attribute p-Difference Mean

CAPUCHIN

from: **Database repair meets algorithmic fairness**

CAPUCHIN

What is it?

Capuchin is a system that interprets the problem of fairness as a **database repair task** with the aim to remove discrimination by repairing the training data that will be used to train a classifier.

Why repairing instead of removing protected attributes?

Simply omitting the protected attributes is an ineffective approach since they are frequently represented implicitly by other attributes: the discrimination remains and its detection becomes harder!

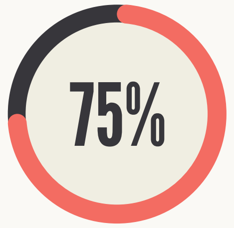
TECHNIQUE

PRE - PROCESSING

We need a definition of Fairness ...

There are many definitions of Fairness

Statistical Fairness



Family of fairness definitions based on **statistical measures** on the variables of interest

Demographic Parity - requires an algorithm to classify both the protected and the privileged group with the same probability
Equalized Odds - requires that both protected and privileged groups must have the same false positive rate and the same false negative rate

It has been shown that these measures are inconsistent!

Causal Fairness

Counterfactual Fairness

A classifier is defined as "counterfactually fair" if **the protected attribute of an individual is not a cause of the outcome** of the classifier for that individual.

But individual-level counterfactuals can not be estimated from data in general!

Proxy Fairness

To avoid individual-level counterfactuals, Proxy Fairness studies the population level rather than an individual level.

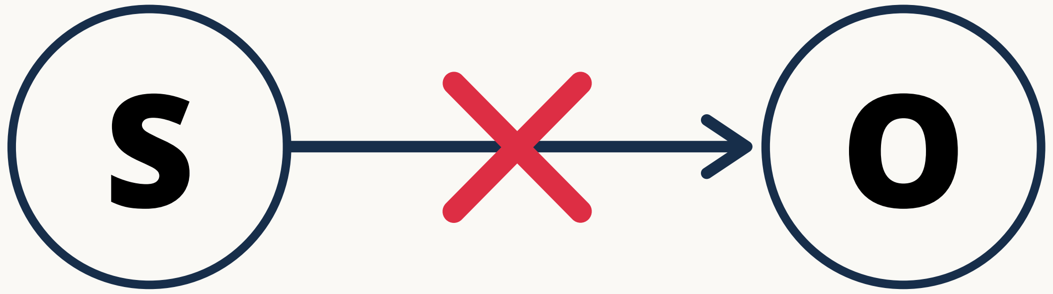
But Proxy Fairness fails to capture group-level discrimination in general.

A new definition of Fairness is needed

Interventional fairness

Unlike proxy fairness, correctly captures group level fairness
Unlike counterfactual fairness is testable from the data.

To ensure interventional fairness, a sufficient condition is
that there exists no path from S (protected attribute) to O (outcome) in the causal graph

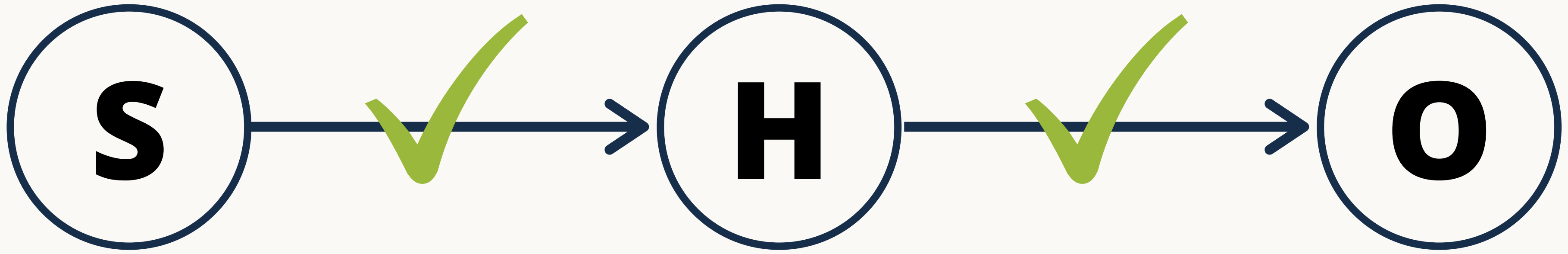


that means that ...

changing the protected attribute, the probability of having a specific outcome is the same.

Justifiable fairness

In Justifiable fairness there can exist a path from S to O only if it goes through an **admissible attribute**



The protected attribute S influences the admissible attribute H and the admissible attribute H influences the outcome O

When do we have Justifiable fairness?

Def. The **Markov Boundary** of Y is the minimal subset of variables V such that Y is indipendent from all the variables that are not contained in MB(Y).
Intuitively, the Markov Boundary of Y shields Y from the influence of other variables

A suffcient condition for a fairness application (A, S, A, I) to be justifiably fair is **MB(O) ⊆ A,** and so that the outcome is influenced only by admissible attributes

How does Capuchin work?

Capuchin performs a sequence of **database updates** (insertions and deletions of tuples) to obtain a new dataset D' that satisfies the condition $(Y \perp\!\!\!\perp I \mid A)$, where *Y is the response variable* of the training dataset.

$(Y \perp\!\!\!\perp I \mid A)$ is considered a integrity constraint that should always hold in training data.

How does Capuchin work?

Capuchin tryes to find another database D' that satisfies the MVD such that the *distance between D and D' is minimized.*

What is an MVD? A Multivalued dependency occurs when two attributes in a table are independent of each other but, both *depend on a third attribute*.

Capuchin reduces the problem of reparing **Conditional Independecies** to the problem of repair MVD, a problem that is well known in licterature.

How does Capuchin work?

D:

	X	Y	Z	Pr
t_1	a	a	c	3/8
t_2	a	b	c	2/8
t_3	b	a	c	2/8
t_4	b	b	d	1/8

B:

X	Y	Z
a	a	c
a	a	c
a	a	c
a	b	c
a	b	c
b	a	c
b	a	c
b	b	d

1) We compute the bag B associated to D (B contains each tuple of D *repeated for the number of times of the numerator of the fraction associated to the tuple itself in D*

D_B :

K	X	Y	Z
1	a	a	c
2	a	a	c
3	a	a	c
1	a	b	c
2	a	b	c
1	b	a	c
2	b	a	c
1	b	b	d

D'_B :

K	X	Y	Z
1	a	a	c
2	a	a	c
1	a	b	c
2	a	b	c
1	b	a	c
1	b	b	c
1	b	b	d

2) Next, we add the new attribute K to the tuples in B and we assign distinct values to **t.K** to all duplicate tuples t, thus converting B into a set DB with attributes K union V.

Then, we repair DB w.r.t. to the MVD **Z ->> KX**, obtaining a repaired database D’B.

How does Capuchin work?

D' :

X	Y	Z	$\Pr'$
a	a	c	2/7
a	b	c	2/7
b	a	c	1/7
b	b	c	1/7
b	b	d	1/7

3) Finally, we construct a new training set D' eliminating the column K and **associating to each tuple the probability distribution** obtained by marginalizing the empirical distribution on D' B to the variables V.

AI FAIRNESS 360

from: **AI FAIRNESS 360: an extensible toolkit for detecting, understanding and mitigating unwanted algorithmic bias**

An extensible toolkit for detecting, understanding and mitigating unwanted algorithmic bias

AI FAIRNESS 360

What is it?

Extensible python Toolkit for

- detecting
- undertsanding
- mitigating

unwanted algorithmic biases

framework to **share** and **evaluate** algorithms

TECHNIQUE

PRE - PROCESSING

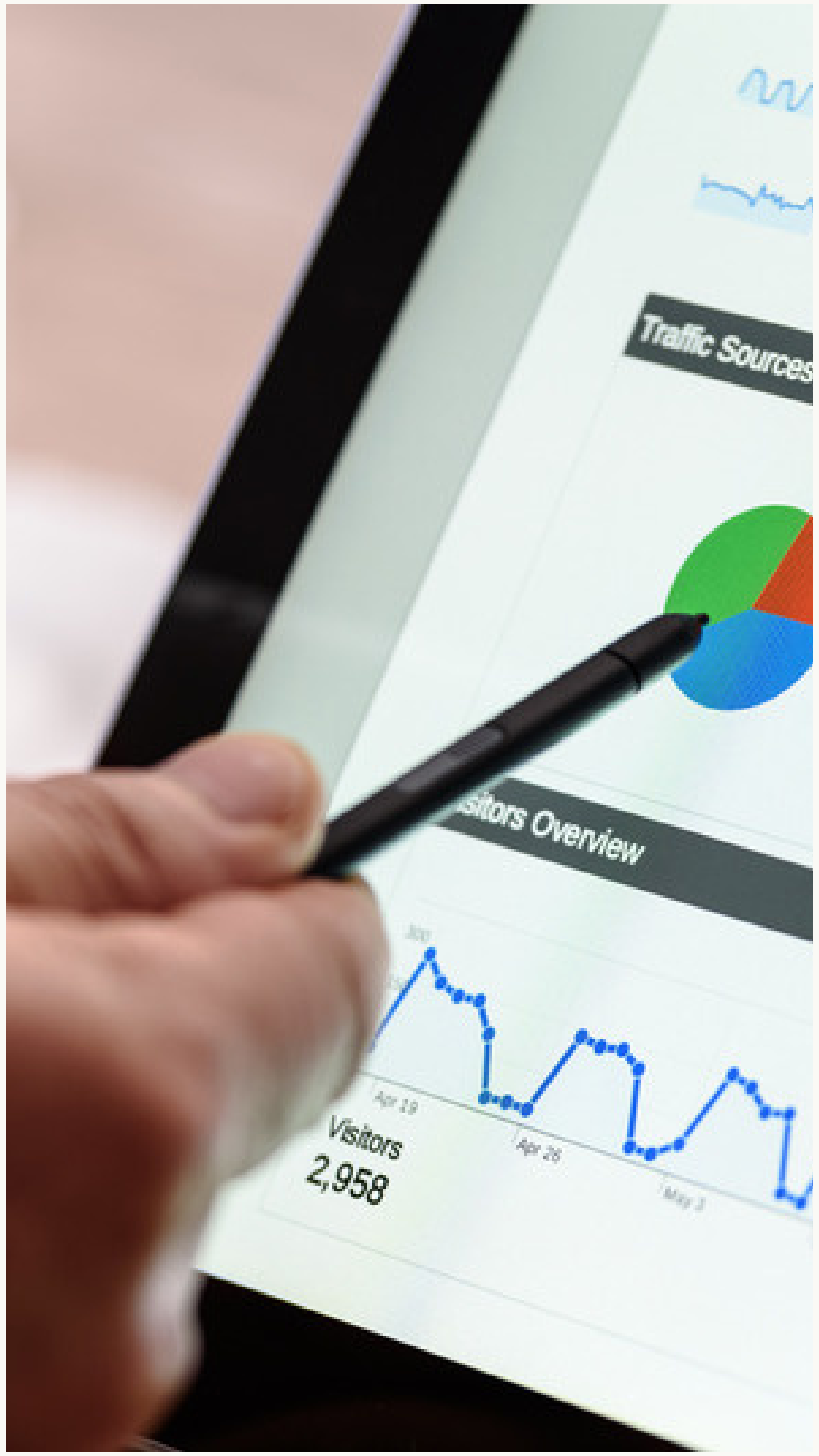
IN - PROCESSING

POST - PROCESSING

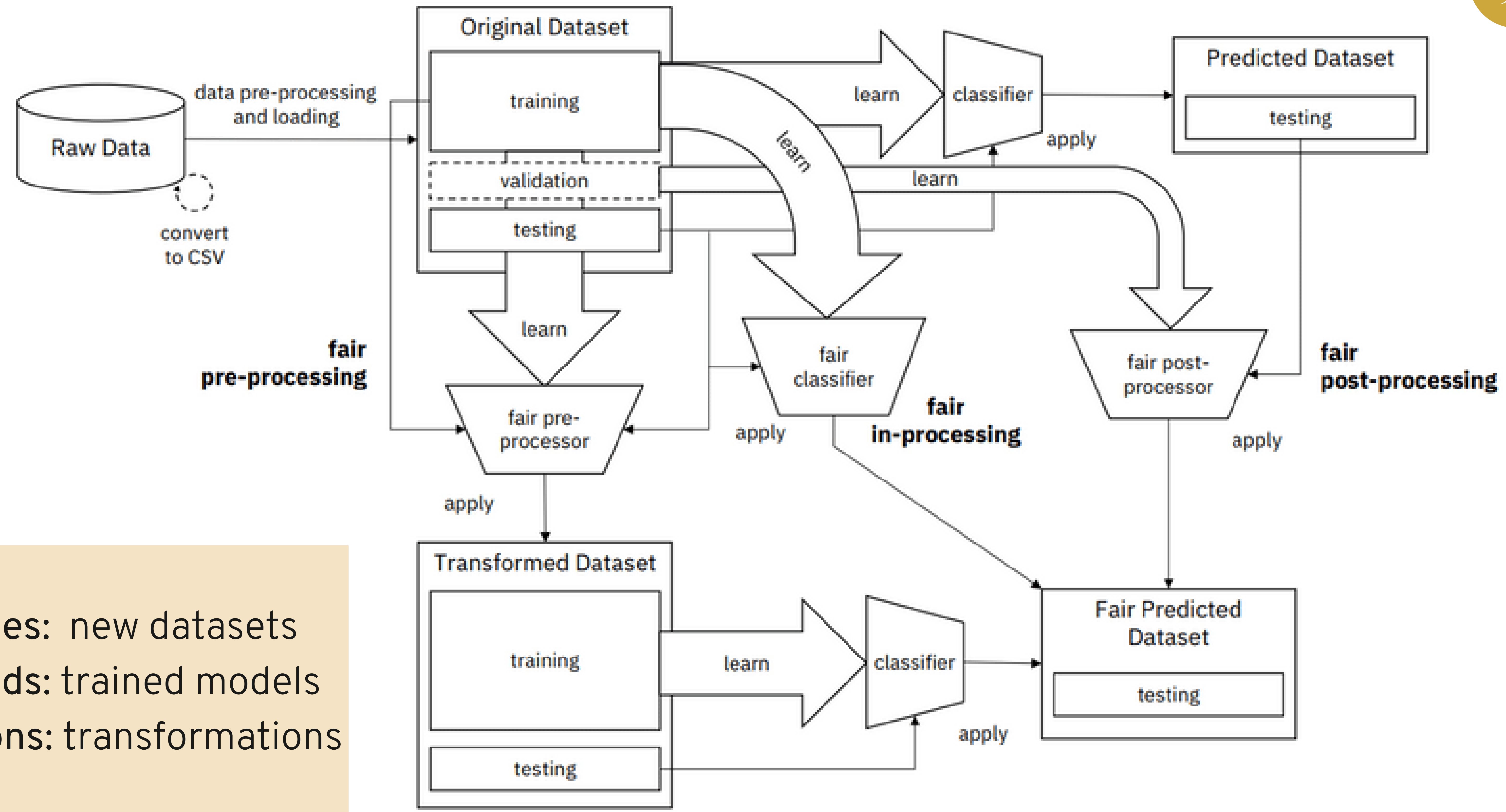
act on
input

↓

produce
output



Pipeline for bias mitigation



- rectangles: new datasets
- trapezoids: trained models
- transitions: transformations

Classes used

Dataset class

- *training data: to* learn classifiers
- *testing data:* make predictions and compare metrics
- *raw data:* must be cleaned

Metric class

compute **fairness metrics** to **detect bias** in datasets and models

Explainer class

provide insights about computed **fairness metrics**

associated with Metric class

Algorithmic class

improve fairness metrics by:

- modifying **training data**
- modifying **learning algorithms**
- modifying the **predictions**

The comparison

	RANKING FACTS	FAIR DB	CAPUCHIN	AI FAIRNESS 360
GOAL	Detect biases and discover unfair behaviors in datasets by analyzing **only one protected attribute at a time**	Detect biases and discover unfair behaviors in datasets by analyzing **one or more protected attribute at a time**	**Repair** the training dataset in order to achieve fairness	Detect, understand and mitigate unwanted algorithmic biases providing a framework to **share and evaluate algorithms**
OUTPUT	Data visualization tools display the general unfair behaviors found in the dataset.	Provides very precise indications of unfairness intended to be used in the correction of the dataset	A repaired dataset	///
TECHNIQUE	Pre processing	Pre processing	Pre processing	Pre/In/Post processing
APPROACH	Uses measures that are statistical tests and determine if the result is fair by using the computed p-value	Uses the Functional dependencies (ACFDs) in order to discover if the protected attributes influence the output	Exploits the techniques used to repair Multivalued dependencies in order to repair the conditional independencies (CI)	It depends on the algorithm
FAIRNESS TYPE	Statistical Fairness	Statistical Fairness	Justifiable Fairness	It depends on the algorithm

Thank you for your attention!