FAIR-DB: FunctionAl DependencIes to discoveR Data Bias

Fabio Azzalini
Politecnico di Milano
Milan, Italy
fabio.azzalini@polimi.it

Chiara Criscuolo
Politecnico di Milano
Milan, Italy
chiara.criscuolo@mail.polimi.it

Letizia Tanca
Politecnico di Milano
Milan, Italy
letizia.tanca@polimi.it

ABSTRACT

Computers and algorithms have become essential tools that pervade all aspects of our daily lives; this technology is based on data and, for it to be reliable, we have to make sure that the data on which it is based on is fair and without bias.

In this context, Fairness has become a relevant topic of discussion within the field of Data Science Ethics, and in general in Data Science. Today's applications should therefore be associated with tools to discover bias in data, in order to avoid (possibly unintentional) unethical behavior and consequences; as a result, technologies that accurately discover discrimination and bias in databases are of paramount importance.

In this work we propose FAIR-DB (FunctionAl dependencIes to discoveR Data Bias), a novel solution to detect biases and discover discrimination in datasets, that exploits the notion of Functional Dependency, a particular type of constraint on the data. The proposed solution is implemented as a framework that focuses on the mining of such dependencies, also proposing some new metrics for evaluating the bias found in the input dataset. Our tool can identify the attributes of the database that encompass discrimination (e.g. gender, ethnicity or religion) and the ones that instead verify various fairness measures; moreover, based on special aspects of these metrics and the intrinsic nature of dependencies, the framework provides very precise information about the groups treated unequally, obtaining more insights regarding the bias present in dataset compared to other existing tools. Finally, our system also suggests possible future steps, by indicating the most appropriate (already existing) algorithms to correct the dataset on the basis of the computed results.

1 INTRODUCTION

In recent years, fairness has become an important topic of interest in the Data Science community. Indeed, computers and algorithms have made our lives efficient and easier, but among the prices we risk to pay is the possible presence of discrimination and unfairness in the decisions we make with their support.

A famous example of unfairness in a data science application regards the Propublica analysis of the COMPAS Recidivism Algorithm [1], a decision-making tool used by judges, probation and parole officers to assess a criminal defendant's likelihood of becoming a recidivist, where their study found that black defendants were far more likely than white defendants to be incorrectly judged to be at a higher risk of recidivism.

Data Science technologies are based on data, and for them to be reliable we have to make sure that the data we feed them are fair and without bias. As a consequence, in these particular applications of data analysis, *data can be considered of good quality only if it conforms to high ethical standard*[8] and, to avoid (possibly unintentional) unethical behaviors and their consequences, data cleaning tools should also include *tools to discover bias in data*, and technologies that accurately discover discrimination and bias to obtain fair databases are badly needed[13].

In this paper we present *FAIR-DB*, a framework that, by discovering and analyzing some special types of functional dependencies, is able to find unfair behaviors in a dataset and guide its correction.

A *Functional Dependency* ($FD : X \rightarrow Y$) is a class of database integrity constraints that hold between two sets X and Y of attributes in a relation of a database. It specifies that the values of the attributes of X uniquely (or *functionally*) determine the values of the attributes of Y. Looking at Table 1 we may spot the following FD: *Education-Num* $\rightarrow$ *Education*, meaning that the number of years already attended at school functionally determines the school level. In a FD, X is called antecedent or left-hand-side (LHS) while Y is called consequent, or right-hand-side (RHS).

The constraints that Functional Dependencies impose are often too strict for real world datasets since they must hold for all the values of the attribute sets X and Y. For this reason, researchers have begun to study generalizations of FDs, called Relaxed Functional Dependencies[4], which relax one or more constraints of canonical FDs.

Among these, we consider Approximate Functional Dependencies (AFDs), that are uncertain FDs, i.e. they hold only on a subset of the tuples, and Conditional Functional Dependencies (or CFDs), where conditions are used to specify the subset of tuples on which a dependency holds.

In particular, a CFD is a pair $(X \rightarrow Y, t_p)$, where X and Y are sets of attributes, $X \rightarrow Y$ is a standard functional dependency and t_p is a *pattern tuple* over the attributes in X and Y; for each A in $X \cup Y$, $t_p[A]$ is a constant *'a'* in *dom(A)*, or an unnamed variable '_'.

Looking at Table 1 we may have the CFD: *Education, Income= '>50K'* $\rightarrow$ *Native-Country*, meaning that, for people whose income is higher than 50K, their education degree functionally determines their native country.

With this type of dependencies we can spot specific concrete patterns in the dataset, and thus we are able to analyze behaviors in correspondence to precise values. Combining the two kinds of relaxed dependencies we obtain the Approximate Conditional Functional Dependencies (ACFDs), i.e., uncertain CFDs.

In this work we use Approximate Conditional Functional Dependencies (ACFDs) to detect biases and discover discrimination in the datasets subject to analysis, by recognizing cases where the value of a certain attribute (e.g. gender, ethnicity or religion) *frequently determines* the value of another one (such as range of the proposed salary or social state).

The paper is organized as follows. Section 2 contains the related work. Section 3 details the methodology. Section 4 presents the experiments and, finally, Section 5 concludes the paper.

[1]https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

2 STATE OF THE ART

Most of the research done in the area of Data Science Ethics is carried out by the Machine Learning community; in this context, researcher start from defining a notion of fairness in data, and then apply it to solve the problem of learning from *unfair data* in a prediction task. This notion of fairness is based on the idea that data have to satisfy three requirements: *Diversity*, *Coverage* and *Stability*.

Three possible approaches can be adopted when trying to enforce fairness in a data analysis application: *(i) preprocessing techniques*, i.e. procedures that before, the application of a prediction algorithm, make sure that the learning data are fair; *(ii) inprocessing techniques*, i.e. procedures that ensure that, during the learning phase, the algorithm does not picks up the bias present in the data, and *(iii) postprocessing techniques*, i.e. procedures that correct the algorithm's decisions with the scope of making them fair.

Most of the machine learning research concerns the second approach; recently, several open-source libraries have been developed to solve the "unfairness problem" during the learning phase of the prediction task. An example is *FairML* [1], which provides an auditing tool for predictive models by quantifying the relative effects providing various different inputs to a model and comparing its predictions. *FairTest* [15], on the other hand, approaches the task of detecting bias by looking for correlations between predicted labels and protected attributes.

A project that provides the implementation of many interesting techniques is *AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias* [2], this work presents a new open-source framework whose aim is to reach algorithmic fairness. The system tries to mitigate algorithmic bias by exploiting techniques such as: *Reweighing* [9], *Optimized Preprocessing* [3], *Learning Fair representations* [10] and *Disparate Impact Remover* [7].

In the machine learning context, the majority of works that try to enforce fairness are related to a prediction task, and more specifically to classification algorithms in decision-making tools. The main difference between these approaches and our framework, that solves unfairness adopting a *preprocessing tecnique*, is that our system does not need a classifier to work, because it is based on finding conditions (constraints) that are already present in the data, even though possibly with some level of approximation. Furthermore, building a classifier to solve the fairness problem requires the policy to be application-oriented, greatly limiting the applicability of these systems to scenarios where other tasks are needed.

A very interesting work on fairness in Machine Learning that employs a *preprocessing tecnique* is *Nutritional Labels for Data and Models* by Stoyanovich and Howe [12]. The authors developed an interpretability and transparency tool based on the concept of *Nutritional Labels*, drawing an analogy to the food industry, where simple and standardized labels convey information about the ingredients and the production processes. Nutritional labels are derived semi-automatically as part of the complex process that gave rise to the data or model they describe. The final system is called *Ranking Facts*, and automatically derives nutritional labels for ranking. *Ranking Facts* is a collection of visual widgets that are based on stability, fairness and diversity concepts. In particular, the *Fairness* widget quantifies whether the ranked output exhibit statistical parity (a particular definition of group fairness) with respect to one or more protected attributes.

Even though *Ranking Facts* and our framework have a similar objective (they both analyze a dataset to discover unfair behaviors), they achieve this goal using two very different strategies. Moreover, the output of the two systems is very different, our framework provides very precise indications of unfairness intended to be used in the correction of the dataset, while *Ranking Facts* employes data visualization tools to display the general unfair behaviors found in the dataset.

3 METHODOLOGY

This section presents the details of the *FAIR-DB* framework.

3.1 Preliminary Notions and Framework Overview

We now introduce some fundamental notions that will accompany us along our discussion.

Given a dataset D the *support* of a CFD $(X \rightarrow Y, t_p)$ is defined as the proportion of tuples t in the dataset D which contain t_p, that is:

$$Support(X \rightarrow Y, t_p) = \frac{|t \in D; t_p \subseteq t|}{|D|}$$

Confidence is an indication of how often the CFD has been found to be true. Let $t_p = (x \cup y)$ where x is a tuple over X and y is a tuple over Y. The confidence value of a CFD $(X \rightarrow Y, t_p)$ is the proportion of the tuples t containing x which also contain y:

$$Confidence(X \rightarrow Y, t_p) = \frac{|t \in D; t_p \subseteq t|}{|t \in D; x \subseteq t|}$$

Figure 1 represents the workflow of FAIR-DB, composed by the following main steps:

(1) **Data Preparation and Exploration:** in this phase we import the data, perform (if needed) data integration and apply the typical data preprocessing steps (solve missing values, apply discretization etc.) needed to clean and prepare the data. During this phase, we also might visualize the attribute features using different *Data Visualization* techniques.
(2) **ACFD Discovery and Filtering**: in this phase we apply an *ACFD Discovery* algorithm to extract Approximate Conditional Functional Dependencies from the dataset. This algorithm takes as input the prepared dataset and three threshold parameters: *minimum support, minimum confidence* and *maximum antecedent size* of the ACFDs that we are searching. From the output of *ACFD Discovery* we discard the dependencies that contain variables. In this phase, we also discard those some dependencies, keeping only those that might reveal unfairness.
(3) **ACFDs Selection:** for each *ACFD*, we compute some metrics capturing the "ethical level" of the dependency, in particular: *(i)* the *Difference* metric is a novel score introduced to discover dependencies that highlight the violation of fairness, and *(ii)* for each protected attribute p we compute its specific *p-Difference.* According to the values of the metrics, we select the most interesting ACFDs.
(4) **ACFDs Ranking:** we rank the ACFDs in descending order of importance, according to the three metrics.
(5) **ACFDs Selection and Scoring:** from the ranking, the user selects the N ACFDs she perceives as the most problematic, and then the system computes some metrics that summarize the level of unfairness of the dataset.

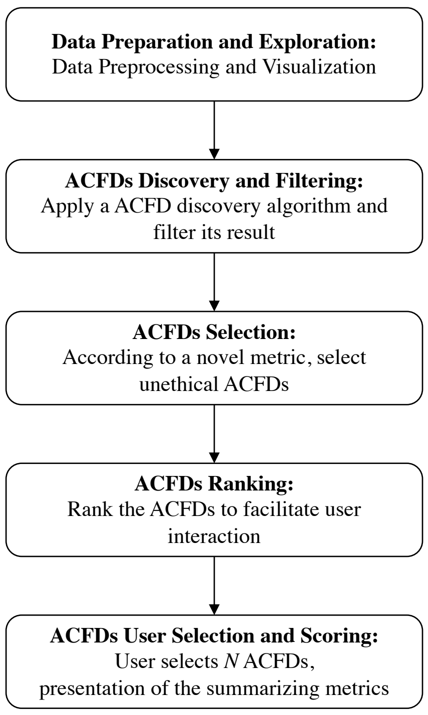
Data Preparation and Exploration:
Data Preprocessing and Visualization
ACFDs Discovery and Filtering:
Apply a ACFD discovery algorithm and filter its result
ACFDs Selection:
According to a novel metric, select unethical ACFDs
ACFDs Ranking:
Rank the ACFDs to facilitate user interaction
ACFDs User Selection and Scoring:
User selects N ACFDs, presentation of the summarizing metrics

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

In the next subsections we give a more detailed description of these phases, with the help of the following running example.

Example 3.1 (Running Example). For a part of our experiments, we have used the U.S. Census Adult Dataset[2] [5], containing information about many social factors of US adults, like 'Income', 'Age', 'Workclass', 'Education', 'Education-Num' (i.e. the number of years already attended at school), 'Marital-Status', 'Race', 'Sex' (i.e. gender), (work) 'Hours-Per-Week', 'Native-Country', and some more. Table 1 contains examples of its tuples.

3.2 Data Preparation and Exploration

Data Preparation starts with *Data Acquisition*, when the user gathers one or more datasets that she will use in the analysis. After Data Acquisition, we compute *Summary Statistics* for each column of the dataset. This primary phase gives a general idea of the dataset and during this step, we can hypothesize the *protected columns* and identify, if present, the *target variable.*

We then perform *Data Cleaning*, since poor data quality could lead to wrong or incomplete results. Depending on the dataset we are considering, we may need to face several different error types. We continue with Data Integration, where, if present, multiple datasets are merged into a single one. After Data Cleaning and Data Integration we may need to perform other steps: *Feature Selection* and *Discretization* [14].

Feature selection refers to choosing the set of features to use that is appropriate for the subsequent analysis. The goal of feature selection is to come up with the *smallest set of features* that best captures the characteristics of the problem being addressed [14].

Another important step is *Discretization*, performed by transforming data *from numeric to nominal data type* [14]. In our case discretization is particularly important, since Functional Dependencies built on numerical attributes that have many different values are very precise, but they do not give an overview over the attribute values. For instance, if we have a dependency that contains a specific value of the attribute *'Age'* (let us assume that it is 18) we do not know if the dependency could be valid also for values near to 18, like 20 or 16; furthermore, the dependency could be not useful if involves only 18-years old people. For this reason, we suggest to use Discretization for numerical attributes that have many different values. A similar procedure is often needed also when a categorical attribute can assume a wide range of values.

[2]https://archive.ics.uci.edu/ml/datasets/Adult

As a last step of this first phase, *Data Visualization* can be of great help in analyzing the characteristics of the data; in fact, from the plots, a user can understand whether groups are present in the dataset and more specifically, if there is a majority class for a certain attribute, and if present can identify minorities. Visualization and summary statistics are very important to find protected attributes and understand if the data contains minorities that need to be analyzed more in details.

Example 3.2 (Data Preparation and Exploration). Before being able to gather useful insights from the running example, we have to perform some preprocessing operations as Data Cleaning, Feature Selection and Discretization. We noticed that the majority of the missing values belong to attributes that are not relevant for our analysis (e.g. 'Marital-Status'), we therefore decided to first perform feature selection and then to remove the few tuples that still had missing values. Regarding feature selection we also removed interesting columns that were related to another one expressing the same meaning; we noticed that two of the columns, 'Education' and 'Education-Num', represented the same information. The latter can be obtained through a numerical encoding of the first one. To extract Functional Dependencies that do not depend on specific values of an attribute, it is useful to group into bins the values of attributes that are continuous. In particular we created five bins for 'Hours-Per-Week' attribute that are: '0-20', '21-40', '41-60', '61-80', '81-100'. We concluded the analysis identifying the protected attribute: 'Sex', 'Race' and 'Native-Country', and selecting 'Income' as target variable. We decided to keep both 'Race' and 'Native-Country' attributes because they are correlated, but not in all cases, for example the offspring of migrants have the same race of their parents, but different native country because in many cases they may be born in U.S.. To have readable and more effective Functional Dependencies, we keep the attribute 'Race' as is in the original dataset and we group the values of the attribute 'Native-Country' using 4 different values: 'NC-US', 'NC-Hispanic', 'NC-Non-US/Hispanic' and 'NC-Asian-Pacific'.

3.3 ACFD Discovery and Filtering

In this phase we extract the ACFDs from the dataset, using the *ACFD Discovery* algorithm of [11]. The algorithm expects as input the following three parameters: *the (minimum) support threshold*, the *(minimum) confidence threshold,* and the size of the largest antecedent *maxSize.*

Given an instance D of a schema R, support threshold δ, confidence threshold ϵ, and maximum antecedent size α, the approximate CFDs discovery problem is to find all ACFDs ϕ: $(X \rightarrow Y, t_p)$ over R with:

- support$(\phi, D) \geq \delta$
- confidence$(\phi, D) \geq \epsilon$
- $|X| \leq \alpha$.

	Age	Workclass	Education	Education-Num	Marital-Status	Race	Sex	Hours-Per-Week	Native-Country	Income
0	90	?	HS-grad	9	Widowed	White	Female	40	United-States	<=50K
1	82	Private	HS-grad	9	Widowed	White	Female	18	United-States	<=50K
2	66	?	Some-college	10	Widowed	Black	Female	40	United-States	<=50K
3	54	Private	7th-8th	4	Divorced	White	Female	40	United-States	<=50K

Table 1: A few exemplar tuples of th U.S. Census Adult dataset

(Education-Degree='Middle-school') → Income='≤ 50K'
(Age-Range='15-30') → income='≤ 50K'
(Education-Degree, Income='≤ 50K') → Native-Country
(Native-Country='NC-Hispanic') → Income='≤ 50K'
(Income='≤ 50K') → Native-Country='NC-US'

Table 2: CFD Discovery output

The ACFDs obtained are in this form:

$$(lhsAttr1 = v1, ..., lhsAttrN = vN) \longrightarrow (rhsAttr = v)$$

The algorithm returns all the dependencies that satisfy the aforementioned criteria. The discovered dependencies may suffer from the following shortcomings:

- It can happen that a dependency *does not involve any protected attribute*, nor the *target attribute*, and thus it is useless for the detection of unfair behavior.
- Some of the dependencies may be *ACFDs* that contain variables; in this case the dependency holds for all the values of each non-constant attribute in its LHS. In our application this type of dependency does not highlight a significant difference corresponding to some specific values of the data, and thus is useless for our aims.

Example 3.3 (CFD Discovery). The algorithm, applied to the dataset resulting from the previous phase, finds 118 ACFDs. Table 2 reports a few of them. We can easily detect the ACFDs that contain variables, for example dependency number ϕ_3: (Education-Degree, Income = '>50K') →(Native-Country), does not specify the values of the attributes 'Education-Degree' *and* 'Native-Country'. *As the user can notice, there are also dependencies that should be removed from the list because they do not contain any protected attribute.*

Now we filter the dependencies, discarding the ones that do not satisfy the following two constraints:

- all the attributes of the dependency must be assigned to a value (this type of ACFDs that consists of only constants in both its LHS and RHS is called *Constant ACFDs*[6]);
- at least one protected attribute and the target variable have to be present inside the dependency, so that the ACFDs might show bias.

During the ACFDs Filtering phase, the user can also add some constraints, such as deciding which *values must appear* in the dependencies, for instance requiring that all the ACFDs must involve only the *'Females'* because the researcher is interested only in peculiar aspects of women, or deciding which *values must not appear* in the dependencies.

Example 3.4 (ACFDs Filtering). From Table 2 we discarded the third dependency for not being a Constant ACFD and the first three dependencies for not containing any protected attribute. After this phase we are left with 84 of the original 118 dependencies.

3.4 ACFDs Selection

This phase is responsible for finding the dependencies that *actually reveal* unfairness in the dataset, in fact, even if the ACFDs identified in the previous step contain protected attributes they do not all necessarily show some unethical behavior.

To do so we have devised two unfairness measures:

- *Difference*: it indicates how much a dependency is *'unethical'*. The more this metric is high, the more the ACFD reveals an unfair behavior.
- *ProtectedAttributeDifference*: it indicates how much the dependency shows bias *with respect to a specific protected attribute.*

In order to assess the unfair behavior of a dependency, we also take into consideration its support, that indicates the *pervasiveness* of the ACFD; unethical dependencies with high support will impact many tuples, and thus will be more important.

For each dependency ϕ, we define the *Difference* metric of ϕ as the difference between the dependency confidence and the confidence of the dependency computed without the protected attributes of the LHS of the ACFD.

Given a dependency in the form:

$$\phi : (X \rightarrow Y, t_p)$$

Let $Z = (X - \{ProtectedAttributes\})$, that is the LHS of the dependency without its *protected attributes*, and let z be the restriction of t to Z. We define the *Difference* as:

$$Difference(\phi) = Confidence(\phi) - NoProtectedAttributeConfidence(\phi)$$

where

$$NoProtectedAttributeConfidence(\phi) = \frac{|t \in D; t_p \subseteq t|}{|t \in D; z \subseteq t|}$$

That is:

$$Difference(\phi) = \frac{|t \in D; t_p \subseteq t|}{|t \in D; x \subseteq t|} - \frac{|t \in D; t_p \subseteq t|}{|t \in D; z \subseteq t|}$$

The *Difference* metric gives us an idea of how much the values of the protected attributes influence the value of Y.

Example 3.5 (Difference score of a dependency). Analyzing the following dependency:

$$\phi_1 : (Sex = \text{'}Female\text{'}, Workclass = \text{'}Private\text{'}) \longrightarrow (Income = \text{'} \leq 50K\text{'})$$

We can compute the Difference *as:*

$$Diff(\phi_1) : Conf(\phi_1) - NoProtAttrConf(\phi_1')$$

where

$$\phi_1' : (Workclass = \text{'}Private\text{'}) \longrightarrow (Income = \text{'} \leq 50K\text{'})$$

Three different behaviors can emerge:

- *If the Difference is close to zero, fairness is respected since it means that females are treated equally to all the elements of the population that have the same characteristics (without specifying the protected attribute).*

ACFD	Support	Difference	RaceDiff	SexDiff	NativeCountryDiff
(Native-Country = ‘NC-Hispanic’) ⟶ (Income = ‘≤50K’)	0.0439	0.1570	0	0	0.1570
(Sex = ‘Female’) ⟶ (Income = ‘≤50K’)	0.2874	0.1352	0	0.1352	0
(Race = ‘Black’) ⟶ (Income = ‘≤50K’)	0.0813	0.1190	0.1190	0	0

Table 3: A few of the selected ACFDs using the *Difference metric*

- *If the Difference is positive it means that the women that work privately and gain less than 50K dollars/year are overall treated worse than the generality of people that work privately and gain less than 50K dollars/year.*
- *If the Difference is negative the opposite situation is detected.*

A dependency could contain in the LHS more than one protected attribute at the same time. For this reason, we introduce the last metric: the *Protected Attribute Difference (P-Difference)*, which is very similar to the *Difference* measure but is computed separately, excluding one protected attribute P at a time ($Z = X - P$).

Finally, we choose the ACFDs whose *Difference* is above the *minThreshold*, which means that there is a significant inequality between the group involved in the ACFD and the general behaviour of the population.

Example 3.6 (Selected ACFDs). Table 3 reports three of the seventeen dependencies that satisfied the selection criteria along with their relevant metrics. From the example, "Hispanic", "Female" and "Black" groups suffer from discrimination with respect to the rest of the population, in fact, people that belong to one or more of these groups have an income that is below the 50000 dollars/year because of their nationality, sex or race.

3.5 ACFDs Ranking

In a real-world dataset, the number of ACFDs selected in the previous step could be very large, even in the order of thousands, therefore for the user to look at all these dependencies would be a very demanding task to complete. Thus, it is necessary to order the dependencies according to some criterion, enabling the user to analyze the most important and interesting ones first, speeding up the process and reducing the cost.

In our framework the user can order the dependencies according to one of the following criteria:

- *Support-based*: the support indicates the proportion of tuples impacted by the dependency: the higher the support, the more tuples are involved by the ACFD. Ordering dependencies by support highlights the *pervasiveness* of the dependency.
- *Difference-based*: this criterion highlights the dependencies where the values of the protected attributes influence most the value of their RHS, therefore this ordering privileges the *unethical aspect* of the dependencies.
- *Mean-based*: this method tries to combine both aspects of a dependency: the unethical perspective and its pervasiveness. Sorting the ACFDs using this criterion results in positioning first the dependencies that have the best trade-off between difference and support.

3.6 ACFDs User Selection and Scoring

In this last phase the user selects from the ranked list N dependencies that are interesting for the research needs. Using only the N selected ACFDs, the framework computes a set of scores that summarize the properties of the entire dataset:

- *Cumulative Support*: is the percentage of tuples in the dataset involved by the selected ACFDs. The more this value is close to 1, the more tuples are impacted by unfair dependencies.
- *Difference Mean*: is the mean of all the *'Difference'* scores of the selected ACFDs. It indicates how much the dataset is unethical according to the dependencies selected. The greater the value, the higher the bias in the dataset.
- *Protected Attribute Difference Mean*: for each protected attribute P, we report the mean of its *P-Difference* over all the selected ACFDs. It indicates how much the dataset is ethical over P according to the selected dependencies.

These summarizing metrics entirely depend on the specific ACFDs selected by the user, thus by selecting different sets of dependencies, the user can highlight different aspects of the dataset.

Note also that the framework allows the user to see some exemplar tuples impacted by the selected ACFDs.

Example 3.7 (ACFDs User Selection and Scoring). The user chooses N = 15 ACFDs that are interesting according to her needs among the dependencies obtained after the ranking step. The total number of tuples involved by the ACFDs is 13296 while the total number of tuples in the dataset is 30169; this results in a Cumulative Support of 0.44. The Difference Mean is 0.16. These two scores indicate that a considerable number of tuples, 44%, show a behavior that is very different, on average 16%, from the fair one. Finally, the P-Difference Mean metrics confirm that the dataset is unfair with respect to all the protected attributes; the groups more discriminated are: 'Female', 'Black', 'NC-Hispanic' and 'Amer-Indian-Eskimo'. Table 4 reports a few interesting ACFDs.

(Sex = ‘Female’) ⟶ Income = ‘≤ 50K’
(Race = ‘Black’) ⟶ Income = ‘≤ 50K’
(Race = ‘Amer-Indian-Eskimo’) ⟶ Income = ‘≤ 50K’
(Native-Country = ‘NC-Hispanic’) ⟶ Income = ‘≤ 50K’

Table 4: A few user-selected dependencies from the U.S. Census Adult Income Dataset

4 EXPERIMENTAL RESULTS

We now present the results obtained by *FAIR-DB* on two real-world datasets, and a comparison between our framework and the *Ranking Facts* system[12].

4.1 Datasets

The first dataset we considered is the *U.S. Census Adult Income Dataset*, already briefly presented in Example 3.1. The version we considered contains 32561 tuples and 13 attributes, 5 of which are numerical and 8 are categorical.

The second dataset we considered is the *Titanic Dataset*, containing information of passengers on the Titanic, a British passenger liner operated by the ‘White Star Line’ that sank in the

North Atlantic Ocean in the early morning hours of 15 April 1912. Of the estimated 2,224 passengers and crew aboard, more than 1,500 died, making the sinking one of modern history's deadliest peacetime commercial marine disasters. The version we considered contains 891 samples and 12 attributes, 8 of which are categorical and 4 are numerical. Specifically the following attributes are present: 'PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'Ticket', 'Fare', 'Cabin' and a few more. We chose 'Sex' and 'Pclass' (the passenger class, that can be first, second or third) as the protected attributes, and selected 'Survived' as the target variable.

4.2 *FAIR-DB* Results

We recall from Example 3.7 the results of *FAIR-DB* on the U.S. Census Adult Income Dataset, which scores a *Cumulative Support* of 0.44 and a *Difference Mean* of 0.16, indicating that many tuples show an unfair behavior. Specifically, the dataset highlights bias towards all the protected attributes: 'Sex', 'Race' and 'Native-Country'; a deeper analysis of the dependencies confirms that the most discriminated groups are: 'Female', 'Black', 'NC-Hispanic' and 'Amer-Indian-Eskimo'.

The Titanic Dataset scores a *Cumulative Support* of 0.74 and a *Difference Mean* of 0.35; indicating that many tuples are subjected to an unfair behavior. Specifically, the dataset shows bias towards the protected attributes 'Sex' and 'Pclass'. Indeed the most discriminated groups are: 'Third class passenger' and 'Male', showing that *(i)* in the third class, to which more than half of the passengers belong, the majority of people did not survive, and *(ii)* the policy 'women and children first' was adopted for the evacuation into the lifeboats. Table 5 reports the corresponding ACFDs.

(Pclass = 1, Sex = Female) $\longrightarrow$ Survived = 1
(Pclass = 2, Sex = Female) $\longrightarrow$ Survived = 1
(Survived = 0, Sex = Female) $\longrightarrow$ (Pclass = 3)
(Pclass = 3, Sex = Male) $\longrightarrow$ (Survived = 0)
(Survived = 1, Sex = Male) $\longrightarrow$ (Pclass = 1)

Table 5: A few user-selected dependencies from the Titanic Dataset

4.3 Comparison with *Ranking Facts*

The results obtained with *Ranking Facts*[12] are in complete accordance with the ones obtained with our framework. Regarding the first dataset, *Ranking Facts* finds unfair behaviors across all the three protected attributes with discrimination against: 'Female', 'Black', 'NC-Hispanic' and 'Amer-Indian-Eskimo'. For what concerns the Titanic datasets, *Ranking Facts* detects unfair behaviors on both the protected attributes with discrimination against: 'Male' and 'Third class passenger'. A deeper comparison with *Ranking Facts* will be included in an extension of this work.

Ranking Facts checks fairness only for one attribute at the time, while, since the ACFD technique can involve more than one attribute at a time, our tool can report information about subgroups fairness, actually detecting unfair behaviors at finer level of granularity. Instead, the results of *Ranking Facts* do not contain information about the existing bias in subgroups or in minorities.

This is very important, because the discrimination might be limited to some specific scenario (e.g. not all the women but only the black women working in the private sector), and this information is very useful to guide the phase of DB repair.

The importance of analyzing more attributes simultaneously it is even more clear if we analyze the ACFDs in Table 5; even though, overall, women obtain a better treatment (both *Ranking Facts* and our framework, for the 'Sex' attribute found discrimination only against men), by analyzing the dependencies we can see that if a woman died she most probably was a third-class passenger, and if a man survived he most probably was a first-class passenger.

5 CONCLUSION AND FUTURE WORKS

We presented *FAIR-DB*, a novel framework, that, through the extraction of a particular type of Functional Dependencies, can discover bias and discrimination present in a datasets.

Future works will include:, *(i)* the addition to the system of a dependency repair phase, that, starting from the selected ACFDs will correct the dataset removing all the unfair behaviors from it *(ii)* the study of dependencies with high confidence and low support to highlight interesting, not necessarily frequent, behaviors, *(iii)* the development of a graphical user interface to facilitate the interaction of the user with the system, *(iv)* the study of other (possibly interesting) classes of functional dependencies[4], *(v)* a deeper comparison with *Ranking Facts* and other similar methods.

REFERENCES

[1] Julius A Adebayo et al. 2016. *FairML: ToolBox for diagnosing bias in predictive modeling.* Ph.D. Dissertation. Massachusetts Institute of Technology.

[2] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, A Mojsilović, et al. 2019. AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development* 63, 4/5 (2019), 4–1.

[3] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. 2017. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems.* 3992–4001.

[4] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. 2015. Relaxed functional dependencies—a survey of approaches. *IEEE Transactions on Knowledge and Data Engineering* 28, 1 (2015), 147–165.

[5] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[6] Wenfei Fan, Floris Geerts, Jianzhong Li, and Ming Xiong. 2010. Discovering conditional functional dependencies. *IEEE Transactions on Knowledge and Data Engineering* 23, 5 (2010), 683–698.

[7] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining.* 259–268.

[8] Donatella Firmani, Letizia Tanca, and Riccardo Torlone. 2019. Ethical Dimensions for Data Quality. *Journal of Data and Information Quality (JDIQ)* 12, 1 (2019), 1–5.

[9] Faisal Kamiran and Toon Calders. 2012. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems* 33, 1 (2012), 1–33.

[10] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. 2018. Learning adversarially fair and transferable representations. *arXiv preprint arXiv:1802.06309.*

[11] Joeri Rammelaere and Floris Geerts. 2018. Revisiting conditional functional dependency discovery: Splitting the "C" from the "FD". In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 552–568.

[12] Julia Stoyanovich and Bill Howe. 2019. Nutritional Labels for Data and Models. *IEEE Data Eng. Bull.* 42, 3 (2019), 13–23.

[13] Julia Stoyanovich, Bill Howe, and HV Jagadish. 2020. Responsible data management. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3474–3488.

[14] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. 2016. *Introduction to data mining.* Pearson Education India.

[15] Florian Tramer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. 2017. FairTest: Discovering unwarranted associations in data-driven applications. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P).* IEEE, 401–416.