An Approach to Automatically Detect and Visualize Bias in Data Analytics

Ana Lavalle
Lucentia Research (DLSI)
University of Alicante
San Vicente del Raspeig, Spain
Lucentia Lab, Alicante, Spain
alavalle@dlsi.ua.es

Alejandro Maté
Lucentia Research (DLSI)
University of Alicante
San Vicente del Raspeig, Spain
Lucentia Lab, Alicante, Spain
amate@dlsi.ua.es

Juan Trujillo
Lucentia Research (DLSI)
University of Alicante
San Vicente del Raspeig, Spain
Lucentia Lab, Alicante, Spain
jtrujillo@dlsi.ua.es

ABSTRACT

Data Analytics and Artificial Intelligence (AI) are increasingly driving key business decisions and business processes. Any flaws in the interpretation of analytic results or AI outputs can lead to significant economic loses and reputation damage. Among existing flaws, one of the most often overlooked is the use biased data and imbalanced datasets. When unadverted, data bias warps the meaning of data and has a devastating effect on AI results. Existing approaches deal with data bias by constraining the data model, altering its composition until the data is no longer biased. Unfortunately, studies have shown that crucial information about the nature of data may be lost during this process. Therefore, in this paper we propose an alternative process, one that detects data biases and presents biased data in a visual way so that the user can comprehend how data is structured and decide whether or not constraining approaches are applicable in his context. Our approach detects the existence of biases in datasets through our proposed algorithm and generates a series of visualizations in a way that is understandable for users, including non-expert ones. In this way, users become aware not only of the existence of biases in the data, but also how they may impact their analytics and AI algorithms, thus avoiding undesired results.

1 INTRODUCTION

Nowadays, Data Analytics have become a key component of many business processes. Whether driving business decisions or offering new services through Artificial Intelligence (AI) algorithms, data serves as the main resource for improving business performance. Therefore, any flaws within the data or its use will be translated into significant performance and economic loses.

One of such flaws is data bias and the use of imbalanced datasets. When unadverted, data bias can significantly affect the interpretation of data, and has devastating impact on AI results as recently reported by the Gartner Group [6]. One area where biases lead to life-threatening consequences is Healthcare, where identifying as healthy a patient that is incubating a severe illness may delay its treatment [2].

As such, data bias has become an important concern in the community, with Big companies like Amazon, Facebook, Microsoft, Google, etc. investing resources and effort to tackle the problem. Amazon Web Services [23] has published information about fairness in their machine-learning services in terms of accuracy, false positive and false negative rates. Facebook [19] has shown one of its internal anti-bias software tools, "Fairness Flow" which measures how a model interacts with specific groups.

Unfortunately, most approaches developed until now are mainly focused on machine-learning and rebalancing the biased datasets. As [7] argues, the fairness of predictions should be evaluated in context of the data, and unfairness induced by inadequate samples sizes or unmeasured predictive variables should be addressed through data collection rather than by constraining the model. As such, a general approach that automatically warns the user of the existence of biases and lets her analyze the data from different perspectives without altering the dataset is missing.

Therefore, in this paper we focus our work on detecting and presenting in a humanly understandable way the existence of data bias and imbalanced datasets, with a special focus on enabling the analysis through data analytics without altering the dataset.

Our approach complements our previous work [15] [14] where we presented an iterative Goal-Based modeling approach based on the i* language for the automatic derivation of data visualizations and we aligned it with the Model Driven Architecture (MDA) in order to facilitate the creation of the right visual analytics for non-expert users. Now, we include a Biases Detection Process that automatically detects the existence of biases in the datasets and enables users to measure them and select those ones which are relevant to them. Our process includes a novel algorithm that takes into account the scope of the analysis, detects biases, and presents them in a way that is understandable for users, including non-expert ones. In this way, users become aware not only of the existence of biases in their datasets, but also how they may impact their analytics and AI algorithms, thus avoiding unwanted results.

The rest of the paper is structured as follows. Section 2 presents a classification of types of biases. Section 3 summarizes the related work in this area. Section 4 describes our proposed process. Section 5 presents our Biases Detection Approach. Section 6 describes results of the experiments applying our approach. Finally, Section 7 summarizes the conclusions and our future work.

2 BIASES IN DATA

In order to illustrate the negative impact of data bias, in this section, we provide a classification of types of biases. There are different types of biases in datasets, the most common being Class Imbalance and Dataset Shift.

Class Imbalance is the case where classes are not equally represented in the data, this means that one or more categories on the dataset have a higher representation than the rest of the categories. It is usual to find this kind of bias in real word datasets [12]. This bias causes several problems, specially when people are trying to analyze this data and/or applying AI algorithms.

Dataset Shift refers to the case where the distribution of the data within the training dataset does not match the distribution in the test and real datasets. In real word datasets often train and test datasets have not been generated by the same distribution.

Artificial Intelligence Algorithms trained on biased training sets tend not to generalize well on test data that is from the true underlying distribution of the population, which has an negative effect on the quality of a machine learning model. As [18] argue, there are three potential types of dataset shift:

Covariate Shift: It happens when the input attributes have different distributions between the training and test datasets.

Prior Probability Shift: In this case, it happens when the class distribution is different between the training and test datasets.

Concept Shift: It happens when the relationship between the input and class variables changes. Usually occurs when training data is collected at a different point in time than testing data.

Biased datasets are very common and they can cause severe problems if bias are not taken into account and treated properly depending on the type of bias we are facing, the context, and the objective that the dataset is being used for. Therefore, it is paramount to show users how biased their data are, in order to enable them to take into account those biases which are determinant to them. Otherwise, their decisions will likely have unexpected and negative consequences.

3 RELATED WORK

The class imbalance problem has been encountered in multiple areas, some of them with a serious impact, such as in the interpretation of medical data [5]. This problem has been also considered one of the top 10 problems in data mining and pattern recognition [24]. The issue with imbalance in class distribution becomes more pronounced with the applications of the AI algorithms. Mining and learning classifiers from imbalanced datasets are indeed a very important problem from both the algorithmic and performance perspective [13]. Not choosing the right distribution can introduce bias towards the most represented class. Since most AI algorithms expect a balanced class distribution [11], an algorithm trained with imbalanced datasets will tend to unadvertedly return results of the most populated classes.

Different authors have proposed several techniques to handle with these problems. Generally, the approaches to deal with **Imbalanced Data** issues involve three categories [16]:

Data perspective: uses techniques to artificially re-balance the class distribution by sampling the data space to diminish the effect caused by class imbalance. As [10] argues, one intuitive method is undersampling the majority classes by dropping training examples. This approach leads to smaller data sets, but important examples could be dropped during the process. Another method is oversampling the minority classes.

Algorithmic perspective: these solutions try to adapt or modify cost adjustment within the learning algorithm to make it perform better on imbalanced data sets during the training process. For example, [17] proposes an algorithm that is able to deal with the uncertainty that is introduced in large volumes of data without disregarding the learning in the underrepresented class.

Ensemble approach: this type of solutions uses aspects from both perspectives to determine the final prediction. [9] proposes an integrated method for learning large imbalanced dataset. Their approach examines a combination of metrics across different learning algorithms and balancing techniques. The most accurate method is then selected to be applied on real large, imbalanced, and heterogeneous datasets.

In the case of **Dataset Shift** (when the training data and test data are distributed different), a common approach is to reweight data such that the reweighted distribution matches the target distribution [20]. In [22] authors analyze, the relationship between the class distribution of training data to determine the best class distribution for learning. [10] have recently proposed decision tree learning for finding a model that is able to distinguish between training and test distributions.

On the other hand, some works have focused on the impact of data flaws on the visual features of visualization. M. Correll et al. in [8] show how it is possible to create visualizations that seem "plausible" (design parameters are within normal bounds and pass the visual sanity check) but hide crucial data flaws. The biases can be considered as data flaws if the context determines so. It is possible to detect biases in datasets when the classification categories are not approximately equally represented.

As we have shown, most approaches developed until now are mainly focused on machine-learning and rebalancing the biased datasets. However, our goal is not to balance the biased datasets. As [7] argues, the fairness of predictions should be evaluated in context of the data, and unfairness induced by inadequate samples sizes or unmeasured predictive variables should be addressed through data collection rather than by constraining the model. For this reason, we propose an approach that automatically warns the user of the existence of biases and lets her analyze the data from different perspectives without altering the dataset. Since one of the core benefits of visualizations is enabling people to discover visual patterns that might otherwise be hidden [8].

4 PROPOSED PROCESS

In this section, we will describe our proposed process. Fig. 1 summarizes the process followed in our proposal, representing in a red cloud the new elements introduced in this paper. Rest of the elements were introduced in our previous work [15] [14].

In our process, firstly, a sequence of questions guides users in creating a **User Requirements Model** [15] that captures their needs and analysis context. Then, this Model is complemented by the **Data Profiling Model** [15] that analyzes of the features of the data sources selected to be visualized. The user requirements, together with the data profiling information, are translated into a **Visualization Specification** that enables users to derive the best visualization types [15] in each context automatically. This transformation generates a **Data Visualization Model** [14].

The **Data Visualization Model** enables users to specify visualization details regardless of their implementation technology. This model also enables users to determine if the proposed visualization is adequate to satisfy the essential requirements for which it was created or not. If the proposed visualization does not pass the user validation, it will point out the existence of missing or wrongly defined requirements. In this case, a new cycle is started by reviewing the existing model to identify which aspects were not taken into account, generating in turn an updated model. Otherwise, a successful validation will start the **Biases Detection Process**. Once users have validated the visualization, the attributes of the collections that have been selected in the process to be represented in the visualization are analyzed. Our novel algorithm examines the data to automatically detect biases and presents this information to the users. Users may define thresholds to adapt the **Biases Detection Process** to their specific needs. The definition of thresholds is performed in an easy way, adapted for non-expert users by defining two variables through the interface. This new functionality will make users aware of biases that could significantly alter the interpretation of their data, as well as the techniques to be used for the analysis.

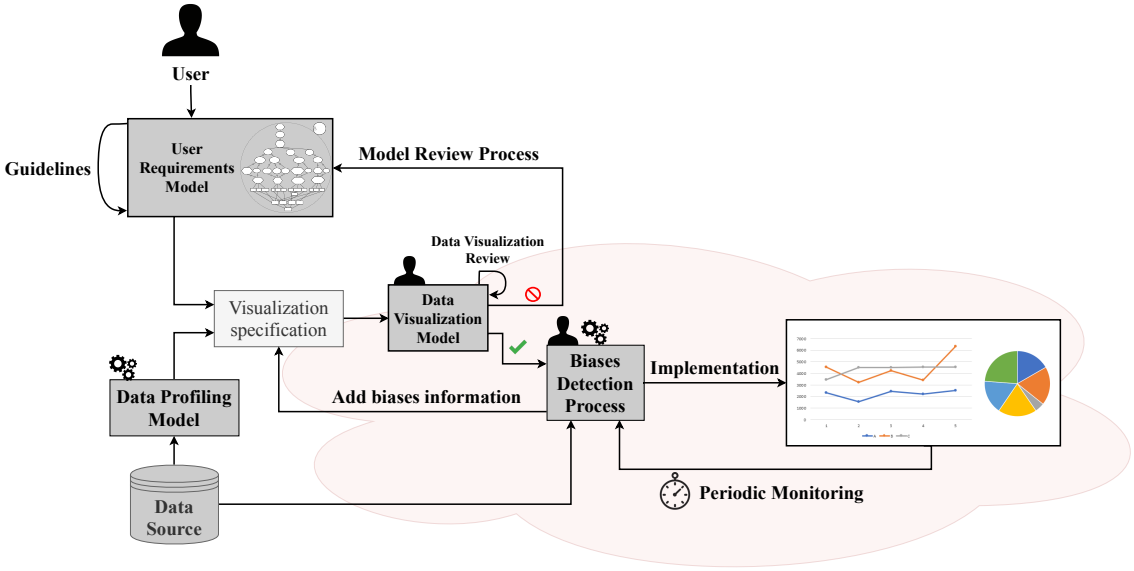

Figure 1: Overall view of the proposed process

As a result of the process, users will obtain a **visual representation** of the bias, being offered the option to include information in their analytics about each of the attributes detected as biased by the algorithm. If they decide to add information about a biased attribute, they can integrate this information within the visualization that they had created for the initial analysis, or, alternatively, in a new visualization that is dynamically connected with the visualization of the process, so that when one of the visualizations is interacted with, the other one is updated.

If users decide to add new information about some biased attribute, a new visualization specification will be generated. Therefore, in the **Data Visualization Model**, users will be able to customize the visualization/visualizations and select how to represent the biases information. Once users validate the new visualizations and do not wish to add further information, the corresponding implementation will be generated.

Finally, when the visualization has been implemented and users are working with it, it is possible to program a Periodic Monitoring. The aim of this continuous monitoring is to ensure that, as new data populates the data sources, no new biases are introduced unadvertedly. The **Periodic Monitoring** event will trigger an execution of our Biases Detection Algorithm with the aim of automatically detect if the data has exceeded the defined thresholds. If a new threshold has been exceed, an alert will be shown to users. This will enable them to return to the Biases Detection Process and choose if they want to edit or add information about this new bias in the visualizations.

By following this process, we facilitate the data analysis and bias awareness for non-expert users in data visualization. Furthermore, all users may benefit from the reduction in time involved in using this approach, since skipping the existing biases will lead to problems, requiring users to manually identify the biases that originated them and requiring to rebuild all the visualizations or re-train their AI algorithms. Therefore, the process enables users to retain control of how data biases affect their data and makes them aware of the impact on their analytics and AI algorithms.

5 BIASES DETECTION

Our proposal starts from the result of our process for automatic derivation of visualizations, shown in Fig. 1. In this sense, we assume the user has defined her requirements, the information that she wants to analyze and that the visualization that best suits her needs has been automatically derived. Once the user has validated the visualization, it is possible that certain elements are changing the interpretation of the data and the user is unaware of them. Therefore, at this point we introduce our novel Biases Detection Process to detect biases in the data, based on the algorithm proposed in this paper that will facilitate this task. It is important to note that, although we assume that the user has followed our previous approach, the process proposed can be applied to visualizations obtained through other tools, as long as the necessary information is facilitated as input to the algorithm.

The first step in our Biases Detection Process proposed is to automatically analyze the attributes of the collections used for the visualization defined in the process through **Algorithm 1**. This algorithm enables us to automatically detect biases in the data by an analysis of the datasets, giving us information as to how biased data are. Users can alter the limits for bias detection in order to tailor the algorithm to their particular case.

It is important to note that, although we exemplify the implementation of our algorithm assuming an existing relational database, our proposal can be applied to any context where structured or semi-structured data is being analyzed.

Algorithm 1 starts with the input of the data tables (**tables_vis**) that are used for the visualization. These tables will come automatically to the algorithm from the previous step of our process. On the other side, the variables **thdCategorical** and **thdBiases** define the thresholds to delimit the biases and attributes, these thresholds do not need to be defined, as they are already assigned default values according to our experience analyzing datasets.

To define the thresholds, we have analyzed different studies. Academic research [11] suggest that there is a situation of class imbalance when the majority-to-minority class ratio is within the

Algorithm 1: Biases Detection Algorithm

```
/* tables_vis comes automatically from the process
   thdCategorical and thdBiases are defined by default, but
   users may personalize it                                  */
Input  : tables_vis[] = list of tables used in the
         visualization, thdCategorical = 0,05; number
         that represents the maximum percentage of the
         total elements of the table to be considered a
         categorical attribute, thdBiases = 8; number
         between 0 and 10 that establishes the admissible
         bias ratio of the attributes (being 0 equally
         distributed and 10 very biased)
Output : biasedAtt = list of attributes and their bias
1  foreach table in tables_vis do
2      Statement stmt = con.createStatement();
       /* Query 1                                           */
3      String rowsQuery = "SELECT COUNT(*) FROM " +
         table;
       /* Query 2                                           */
4      String attributesQuery = "SELECT COLUMN_NAME
         FROM INFORMATION_SCHEMA.COLUMNS
         WHERE TABLE_NAME = " + table;
       /* number of rows from the table                     */
5      ResultSet rsRN = stmt.executeQuery(rowsQuery);
6      int RN = rsRN.getInt(1);
       /* list of attributes from the table                 */
7      ResultSet attributes =
         stmt.executeQuery(attributesQuery);
       /* for each attribute from the table                 */
8      foreach attribute in attributes do
           /* Query 3                                       */
9          String groupAttrQuery = "SELECT COUNT(" +
             attribute + " ) FROM " + table + " GROUP BY " +
             attribute ;
           /* number of times that each different value of the
              attribute appears                             */
10         ResultSet rsGroupAttr =
             stmt.executeQuery(groupAttrQuery);
           /* number of distinct values of the attribute    */
11         rsGroupAttr.last();
12         int RND = rsGroupAttr.getRow();
           /* if it is a categorical attribute              */
13         if RND < RN*thdCategorical then
               /* is extracted the value that is repeated more
                  and less times                            */
14             int max = max(rsGroupAttr);
15             int min = min(rsGroupAttr);
               /* is calculated and normalized the bias in the
                  attribute                                 */
16             float biasAttribute = ((max - min)/max)*10;
               /* if the bias is bigger than the threshold
                  defined by the user                       */
17             if biasAttribute > thdBiases then
18                 biasedAtt.append(attribute, biasAttribute);
19             end
20         end
21     end
22     return (biasedAtt);
23 end
```

range of 100:1 to 10000:1. However, from the viewpoint of effective problem solving, lower class imbalances that make modeling and prediction of the minority class a complex and challenging task (i.e. in the range of 50:1 and lower) are considered high class imbalance by domain experts [21].

In our case, the variable **thdCategorical** is a number that represents the maximum percentage of the total elements of the table to be considered a categorical attribute. An attribute is categorical when it can only take a limited number of possible values. The default threshold for this variable has been defined heuristically, setting the value of this variable to 5% (0,05). This threshold enables us to discover categorical attributes within the data, even when a schema is not available, such as with NoSQL databases or file-based systems.

Moreover, the variable **thdBiases** is a number between 0 and 10 that establishes the admissible bias ratio of the attributes (being 0 equally distributed and 10 very biased). The bias ratio represents the relationship between the values that appear the least and most in an attribute. Therefore, adjusting this variable, users may limit when an attribute is considered biased, i.e. when the difference between the most and least common value is decisive for them. We propose 8 as default value. Therefore, if the most common value has 8 times or more the representation of the least common value, then it will be considered as highly biased.

Finally, the output of this algorithm will be **biasedAtt**, a list with the information about each attribute and its bias ratio.

The algorithm will be executed for each table used for the visualization *(line 1)*. For each table, it stores the number of rows from the table in the variable **RN** *(lines 5-6)*. Then, the attributes of the table are included in the variable **attributes** *(line 7)*. For each attribute in the list *(line 8)*, a ResultSet **rsGroupAttr** with the number of repetitions of each different value is stored *(line 10)*. In *(lines 11-12)*, the number of distinct values of this attribute is calculated and stored in **RND**. Afterwards, the algorithm evaluates whether this attribute is categorical or not *(line 13)*. An attribute is considered categorical when the number of distinct values of this attribute (**RND**) is be lower than the number of rows from the table (**RN**) multiplied by the categorical threshold defined earlier (5%) **thdCategorical**. If this comparison is fulfilled, the values that have the highest (**max**) *(line 14)* and lowest (**min**) *(line 15)* representation are extracted from the ResultSet **rsGroupAttr** that contains the number of times that each different value of the attribute appears. Then, the bias of each attribute is calculated and normalized in **biasAttribute** *(line 16)* using the following formula:

$$\frac{max - min}{max} * 10 \tag{1}$$

We have used Min-Max normalization because it guarantees that all attributes will have the exact same scale and highlights outliers. This is a desirable characteristic in our case, since detecting the existence of these outlier biases and warning the user is one of our main goals. With this normalization, we will have a ratio for each attribute in **biasAttribute** that will provide an indication in the 0 to 10 range how biased is the attribute, 0 being equally distributed and 10 very biased.

If the **biasAttribute** is bigger than the threshold **thdBiases** *(line 17)*, it means that the attribute has a considerable bias that should be analyzed. Then, the name of the attribute and the bias ratio of the attribute previously calculed in **biasAttribute** will be stored in **biasedAtt** *(line 18)*. Therefore, when the algorithm

concludes, the variable **biasedAtt** will contains a list of attributes with their bias ratio.

6 PERFORMANCE ANALYSIS

In order to do an implementation of the experiment, we have downloaded the Fire Department Calls for Service dataset from [1] where we have get an 1,75 GB file.

We have chosen Apache Spark [3] to process this file because its speed, ease of use, advanced and in-memory analytical capabilities. Specifically we have used as a development environment Apache Zeppelin [4] 0.8. The configuration is as default.

We have run the experiment on a single laptop with the following characteristics: Intel Core i5 CPU M 460 @ 2.53GHz × 4, HDD at 7200 rpm, 6GB of RAM and OS: Ubuntu 16.04 LTS.

Although in the definition of **Algorithm 1** we establish connections with the database, since we are running the algorithm on Spark this is not necessary, loading the dataset into the framework using a load instruction instead. We have loaded the *Fire_Department_Calls_for_Service.csv* into the variable **dfCalls** and we run the following queries as part of the algorithm:

(1) **Number of rows from the table:** dfCalls.count()
(2) **List of attributes from the table:** dfCalls.columns
(3) **Number of distinct values from each attribute:**
dfCallsG = dfCalls.groupBy(attribute).count()
dfCallsG.count()

The execution times of our approach over a 5.1 millions of rows and including all the passes to process the 34 columns of the dataset (1,75 GB) are: 46 seconds to be load the data table. Query 1 takes 27 seconds. Query 2 is executed in under 1 second and finally, Query 3 takes 993 seconds. Therefore, the time required to run Algorithm 1 in this experiment is a total of 1066 seconds, i.e. 17 minutes and 46 seconds.

7 CONCLUSIONS AND FUTURE WORK

Data bias is becoming a prominent problem due to its impact in data analytics and AI. Current solutions focus on the problem from an AI outputs perspective, centering their efforts in constraining the model to re-balance the data at hand. The side effect is that the datasets are altered without understanding whether there is a problem at the data gathering step or the data is representing the actual distribution of the sample. In turn, potentially important information about the nature of the data is lost, which can have implications for interpreting the data and finding the root causes of the original imbalance.

Compared to these solutions, in this paper we have presented a Bias Detection Approach. Our proposal complements our previous works [14, 15] by including a novel algorithm that takes into account the scope of the analysis, detects biases, and presents them in a way that is understandable for users, including non-expert ones. The great advantage of our proposal is that we enable users to understand their data and make decisions considering biases from different perspectives without altering the dataset. Furthermore, all users may benefit from the reduction in time required to inspect and understand existing biases within their datasets, while at the same time they avoid biases going unadverted with the problems that it entails.

As a part of our future work, we are continuing our work on new techniques to present biased attributes with a high number of categories. We are also applying our approach to unstructured data and including analytic requirements as an input to estimate the impact of data biases for each particular user.

ACKNOWLEDGMENTS

This work has been co-funded by the ECLIPSE-UA (RTI2018-094283-B-C32) project funded by Spanish Ministry of Science, Innovation, and Universities. Ana Lavalle holds an Industrial PhD Grant (I-PI 03-18) co-funded by the University of Alicante and the Lucentia Lab Spin-off Company.

REFERENCES

[1] 2019. Fire Department Calls for Service dataset. https://data.sfgov.org/Public-Safety/Fire-Department-Calls-for-Service/nuek-vuh3. Accessed: 23/10/2019.
[2] Alaa Althubaiti. 2016. Information bias in health research: definition, pitfalls, and adjustment methods. *Journal of multidisciplinary healthcare* 9 (2016), 211.
[3] Apache. 2019. Apache Spark. https://spark.apache.org/. Accessed: 23/10/2019.
[4] Apache. 2019. Apache Zeppelin. https://zeppelin.apache.org/. Accessed: 23/10/2019.
[5] Colin B Begg and Jesse A Berlin. 1988. Publication bias: a problem in interpreting medical data. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 151, 3 (1988), 419–445.
[6] Kenneth Brant, Moutusi Sau, Anthony Mullen, Magnus Revang, Chirag Dekate, Daryl Plummer, and Whit Andrews. 2017. Predicts 2018: Artificial Intelligence. https://www.gartner.com/en/documents/3827163/predicts-2018-artificial-intelligence. Accessed: 23/10/2019.
[7] Irene Y. Chen, Fredrik D. Johansson, and David Sontag. 2018. Why Is My Classifier Discriminatory?. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. 3543–3554.
[8] Michael Correll, Mingwei Li, Gordon Kindlmann, and Carlos Scheidegger. 2018. Looks Good To Me: Visualizations As Sanity Checks. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 830–839.
[9] Mojgan Ghanavati, Raymond K Wong, Fang Chen, Yang Wang, and Chang-Shing Perng. 2014. An effective integrated method for learning big imbalanced data. In *2014 IEEE International Congress on Big Data*. IEEE, 691–698.
[10] Patrick O. Glauner, Petko Valtchev, and Radu State. 2018. Impact of Biases in Big Data. *CoRR* (2018).
[11] Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* 21, 9 (2009), 1263–1284.
[12] Richard A. Bauder Joffrey L. Leevy, Taghi M. Khoshgoftaar and Naeem Seliya. 2018. A survey on addressing high-class imbalance in big data. *J. Big Data* 5 (2018), 42.
[13] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. 2006. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering* 30, 1 (2006), 25–36.
[14] Ana Lavalle, Alejandro Maté, and Juan Trujillo. 2019. Requirements-Driven Visualizations for Big Data Analytics: a Model-Driven approach. In *International Conference on Conceptual Modeling ER 2019, to appear*. Springer.
[15] Ana Lavalle, Alejandro Maté, Juan Trujillo, and Stefano Rizzi. 2019. Visualization Requirements for Business Intelligence Analytics: A Goal-Based, Iterative Framework. In *27th IEEE International Requirements Engineering Conference RE 2019, to appear.*
[16] Chaoliang Li and Shigang Liu. 2018. A comparative study of the class imbalance problem in Twitter spam detection. *Concurrency and Computation: Practice and Experience* 30, 5 (2018).
[17] Victoria López, Sara Del Río, José Manuel Benítez, and Francisco Herrera. 2015. Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data. *Fuzzy Sets and Systems* 258 (2015), 5–38.
[18] Victoria López, Alberto Fernández, and Francisco Herrera. 2014. On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed. *Information Sciences* 257 (2014), 1–13.
[19] Jerome Pesenti. 2018. TAI at F8 2018: Open frameworks and responsible development. https://engineering.fb.com/ml-applications/ai-at-f8-2018-open-frameworks-and-responsible-development/. Accessed: 23/10/2019.
[20] Sashank Jakkam Reddi, Barnabás Póczos, and Alexander J. Smola. 2015. Doubly Robust Covariate Shift Correction. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.
[21] Isaac Triguero, Sara del Río, Victoria López, Jaume Bacardit, José Manuel Benítez, and Francisco Herrera. 2015. ROSEFW-RF: The winner algorithm for the ECBDL'14 big data competition: An extremely imbalanced big data bioinformatics problem. *Knowledge-Based Systems* 87 (2015), 69–79.
[22] Gary M Weiss and Foster Provost. 2003. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of artificial intelligence research* 19 (2003), 315–354.
[23] Matt Wood. 2018. Thoughts On Machine Learning Accuracy. https://aws.amazon.com/es/blogs/aws/thoughts-on-machine-learning-accuracy/. Accessed: 23/10/2019.
[24] Qiang Yang and Xindong Wu. 2006. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making* 5, 04 (2006), 597–604.