TAPIA: G: A Data Quality Test Approach for Automated Constraint Discovery and Fault Detection

Hajar Homayouni
Advisors: Sudipto Ghosh, Indrakshi Ray
{hhajar,ghosh,iray}@colostate.edu
Department of Computer Science, Colorado State University

ABSTRACT

The quality of data is extremely important for data analytics. Data quality tests typically involve checking constraints specified by domain experts. Existing approaches detect trivial constraint violations and identify outliers without explaining the constraints that were violated. Moreover, domain experts may specify constraints in an ad hoc manner and miss important ones. We describe an automated data quality test approach, ADQuaTe2, which uses an autoencoder to (1) discover constraints that may have been missed by experts, (2) label as suspicious those records that violate the constraints, and (3) provide explanations about the violations. An interactive learning technique incorporates expert feedback, which improves the accuracy. We evaluate the effectiveness of ADQuaTe2 on real-world datasets from health and plant domains. We also use datasets from the UCI repository to evaluate the improvement in the accuracy after incorporating ground truth knowledge.

KEYWORDS

Big data; Data quality tests, Explainable learning, Interactive learning, Unsupervised learning.

1 INTRODUCTION

Enterprises use databases, data warehouses, and big data appliances to store, manage, and query data for making critical decisions. Records can get corrupted because of how the data is collected, transformed, and managed, and also because of malicious activities. Inaccurate data can lead to incorrect conclusions. Thus, rigorous data quality testing approaches are required.

Data quality tests rely on constraint specifications. Syntactic constraints check for the conformance of an attribute with the structural specifications in the data model. For example, in a health data store, *patient_age* must take numeric values. Semantic constraints check for the conformance of the attribute values with the specifications stated by domain experts. Semantic constraints can exist over single attributes (e.g., *patient_age* >= 0) or multiple attributes (e.g., *pregnancy_status = true* → *patient_gender = female*).

Constraints are typically defined by domain experts but often in an ad hoc manner based on their knowledge of the application domain and stakeholders requirements. For example, a data record in a health data store may contain an incorrect value for the day's supply of a drug. However, the constraint that restricts the values for the drug may be missing. Incorrect values in attributes pertaining to medications and prescriptions can have disastrous consequences for both patients and research outcomes if the data is used for patient treatment and in medical research [1]. Tools that automatically generate syntactic constraints also exist, but they only check for trivial ones, such as the not-null check [2]. Existing machine learning approaches can automatically discover non-trivial constraints from the data and report the faulty records as outliers [3]. However, these approaches do not explain which constraints are violated by these records. Thus, experts are forced to inspect a large number of outliers to determine and explain which ones are actually faulty and why. Moreover, these techniques can potentially learn incorrect constraints pertaining to the invalid data and generate false alarms, which can overwhelm domain experts [4].

This paper describes an automated data quality test approach called ADQuaTe2 that we previously proposed [5, 6] to address the above issues. ADQuaTe2 automatically discovers complex semantic constraints from the data, marks records that violate the constraints as suspicious, and explains the violations. ADQuaTe2 uses an unsupervised deep learning technique called autoencoder [7] to discover the constraints associated with the unlabeled records (i.e., records whose validity is not known in advance). We use an autoencoder because its deep architecture can model constraints involving both linear and non-linear relationships among data attributes.

ADQuaTe2 assigns a suspiciousness score (*s*-score) to each record. Records whose *s*-score is greater than a threshold are flagged as suspicious. To reduce the time needed to inspect a large number of suspicious records, the Self Organizing Map (SOM) [8] clustering technique is used to identify a small number of groups. Each group contains records that are likely to violate the same constraints. Decision trees are generated using a Random Forest classifier [9] to identify the constraints violated by each group.

ADQuaTe2 minimizes false alarms through an interactive learning process [10]. Domain experts use a web-based interface to mark groups of records that are actually faulty. This feedback is incorporated to retrain the machine learning model and improve the accuracy of constraint discovery and fault detection.

We evaluated ADQuaTe2 using datasets from a health data warehouse and a plant diagnosis database. We demonstrated that our approach can discover new constraints that were missed by domain experts and can also detect new faults in these datasets. We evaluated the improvements in the accuracy of ADQuaTe2 using datasets with ground truth data (i.e., a set of known faults) from the UCI repository [11]. We demonstrated that the true positive rate increases and the false negative rate decreases after incorporating the ground truth knowledge and retraining the learning model.

2 PROPOSED APPROACH

Figure 1 presents an overview of the approach. Data records are provided as input. The output is a list of groups of suspicious records accompanied with an explanation of the violated constraints. The five components in the approach are explained below.

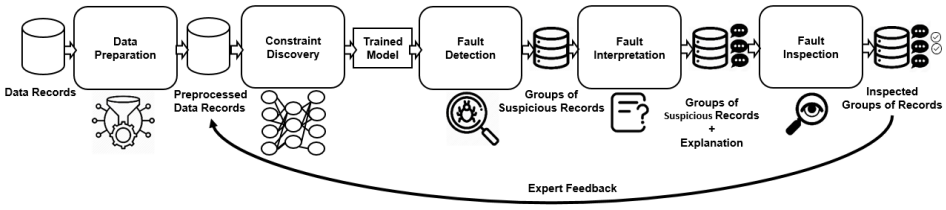


Figure 1: ADQuaTe2 Approach

2.1 Data Preparation

This component prepares the data by transforming it from its raw format into a form that is suitable for analysis. Typical machine learning algorithms cannot be directly applied to certain data types, such as categorical, which is treated as numeric. The data attributes need to be preprocessed based on their type and values. One-hot encoding [12] is used for preprocessing the categorical attributes and the standardization [13] method for the numeric attributes.

2.2 Constraint Discovery

This component obtains a trained model that best represents different types of constraints in the unlabeled input data. It uses an unsupervised deep neural network called autoencoder [7] that is known to be effective for attribute representation learning [14]. An autoencoder is composed of an *encoder* and a *decoder*. The encoder compresses the data from the input layer into a short representation, which is a non-linear combination of the input elements. The decoder decompresses this representation into a new representation that closely matches the original data. The network is trained to minimize the reconstruction error (RE), which is the average squared distance between the original data and its reconstruction [7]. The constraints represented in the trained autoencoder model are in the form of complex equations that formulate the associations among data attributes. However, these constraints are not human-interpretable. Further steps are required to explain the identified constraints to the domain experts.

2.3 Fault Detection

This component detects suspicious records that do not conform to the constraints represented by the trained model. Each record is assigned a suspiciousness score (*s*-score), which is equal to the reconstruction error of the record. Records whose *s*-score is greater than a threshold are flagged as suspicious.

Since individually inspecting a large number of suspicious records is infeasible, we group them based on their similarity using a clustering approach called Self Organizing Map (SOM) [8], which preserves the relationships among attributes in its clusters [15]. However, additional information about the reason behind the invalidity of each group is needed for determining which groups are actually faulty.

2.4 Fault Interpretation

This component helps a domain expert interpret the suspicious groups by generating visualization plots of two types, namely, *s*-score per attribute and decision tree. We use the trained autoencoder model to calculate the *s*-score per attribute. The higher the value of *s*-score for an attribute, the more likely is the attribute to contribute to the invalidity of the group. For each group we generate a plot showing the average *s*-score values for each of the attributes in the group. Figures 2 and 3 show these plots for two suspicious groups in the plant diagnosis dataset. The major causes of invalidity are the attributes *Host*, *Diagnosis ID*, and *Suspected Problem* for Group 1, and the attribute *Genus Confirmation* for Group 2.

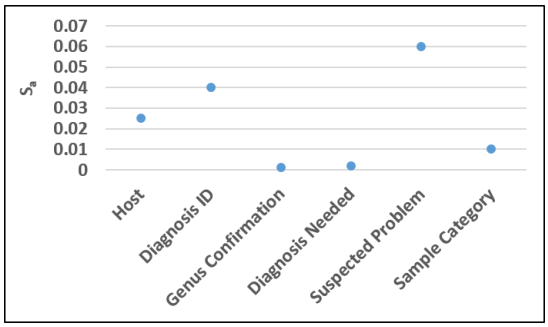


Figure 2: *S*-score Per Attribute for Suspicious Group 1

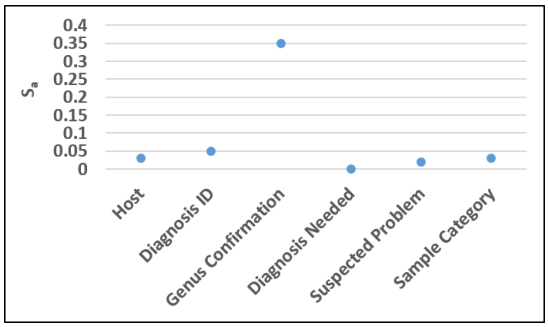
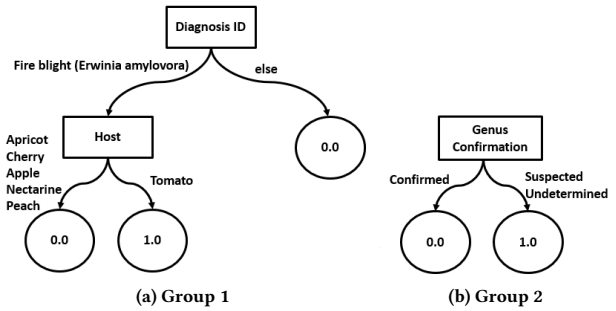


Figure 3: *S*-score Per Attribute for Suspicious Group 2

We use a *decision tree*-based technique [16] called random forest classifier [9] to determine the constraints that are violated by each group of suspicious records. The non-leaf nodes in the decision tree correspond to the attributes, the edges correspond to the possible values of the attributes, and every leaf node contains the label of the path described by the attribute values from the root to that leaf node. The label value lies between 0 and 1, where 1 signifies invalid records. Domain experts decide whether or not a suspicious group is actually faulty by analyzing the constraints represented in the decision trees. In Figure 4 (a), the path from the root (*Diagnosis ID*) to the leaf node labeled 1.0 represents a constraint, which states that if the plant *Diagnosis ID* is 'Fire blight (Erwinia amylovora)' and the *Host* is 'Tomato', then the record is invalid.



(a) Group 1 (b) Group 2

Figure 4: Decision Trees for the Plant Diagnosis Dataset

2.5 Fault Inspection

Domain expert feedback is obtained through a web-based user interface. Check boxes are used to select groups that are actually faulty. The feedback is used to label the training data records. The labels are used by the extended constraint discovery and fault detection components to improve the accuracy of the approach.

2.6 Retraining Using Domain Expert Feedback

2.6.1 Update Training Dataset for Retraining. The label can take four possible values (1: faulty, 0.5: suspicious, 0: unknown, and -1: valid) to each record in the input dataset. The default value is 0. In the initial dataset for the training phase, the labels are all zero. The label value of each data record is updated based on expert feedback. The fault detection component modifies some of the labels from 0 (unknown) to 0.5 (suspicious). The domain expert marks some of the suspicious records as actually faulty, and their labels are updated to 1. Suspicious records not marked by the domain expert are updated to -1. The updated dataset is used to retrain the autoencoder.

2.6.2 Incorporate Domain Expert Feedback for Constraint Discovery. This is done by (1) defining the reconstruction error of autoencoder based on the label value and (2) initializing the network parameters for the retraining phase.

Define reconstruction error based on label value. We provide the record label and the record attributes as input to the autoencoder. The network is trained to (1) minimize the difference between the record and its reconstruction and (2) minimize the difference between the record label and the label predicted by the network.

Initialize autoencoder parameters for retraining. The parameters are initialized with the values learned in the previous execution, which ensures that the network does not lose any information from the previous execution and the accuracy of the network is at least as much as the accuracy of the previous trained network.

2.6.3 Incorporate Domain Expert Feedback for Fault Detection. The fault detection component defines the *s*-score based on the label values using ($s\_score(X) = RE(X) + l(X)$, where $RE(X)$ is the reconstruction error of record X and $l(X)$ is the label assigned to X). This ensures that all the records detected in the retraining phase are either faulty (i.e., records that were flagged by the expert in previous executions) or unknown (i.e., records that have not been reported by ADQuaTe2 in previous executions). Since $0 \leq RE(X) \leq 1$ and $l(X) \in \{-1, 0, 0.5, 1\}$, this definition ensures that the range of the *s*-scores of faulty records is [1, 2], that of valid records is [-1, 0], and that of unknown records is [0, 1]. Setting the threshold to a value greater than zero ensures that ADQuaTe2 never reports as suspicious any valid record in subsequent executions. Moreover, all the records marked as faulty by the expert in previous executions are reported as suspicious in subsequent executions, but with a higher *s*-score value (in the range [1,2]).

3 EVALUATION

We evaluated the constraint discovery, fault detection, and fault interpretation effectiveness of ADQuaTe2 using real-world data from health and plant domains. We used seven datasets created using multiple table joins in the health data warehouse and one dataset from the plant diagnosis database. We also evaluated the improvements in the accuracy of the approach using datasets with ground truth data from the UCI repository [11]. Finally, we evaluate the time taken by the approach.

3.1 Effectiveness In the Absence of Retraining

We evaluated the constraint discovery and fault detection effectiveness on real-world datasets when expert feedback is not used for retraining. The effectiveness was measured along two dimensions: the fault detection ability and the generation of new constraints that were not previously specified by the experts. The new constraints discovered by the approach help detect new faults.

RQ1.a: Can ADQuaTe2 detect the faults in real-world data that were already detected by existing tools that rely on manual specification of constraints by domain experts?

Given E, the set of faulty records detected by an existing data quality test approach, and A, the set of suspicious records detected by ADQuaTe2, we define the metrics *Previously Detected* (PD), *Suspicious Detected* (SD), and *UnDetected* (UD). $PD = \frac{|E \cap A|}{|E|}$ is the percentage of faulty records detected by an existing approach that could also be detected by ADQuaTe2. $SD = \frac{|A-E|}{|A|}$ is the percentage of suspicious records detected by ADQuaTe2 that were not previously detected. $UD = \frac{|E-A|}{|E|}$ is the percentage of faulty records detected by an existing approach that could not be detected by ADQuaTe2. Table 1 shows the values of *PD*, *SD*, and *UD* of ADQuaTe2 with respect to the known faults in the real-world datasets. The first dataset ($ID = 1$) is from the plant science dataset and the rest are from the health datasets. In this table, $|E|$ is the number of known faults and $|A|$ is the total number of suspicious records detected by ADQuaTe2. There were no previously known faults for dataset *ID*s 1 and 5 and every record reported for these two sets corresponded to a suspicious record not previously detected. ADQuaTe2 detected between 96.14% and 100% of faults that were previously detected by Achilles [17] and Murdock [18] testing tools for the health datasets. In the worst case, ADQuaTe2 could not detect 3.86% of faults that were previously detected by these tools, indicating that the autoencoder could not discover all the associations among the attributes.

RQ1.b: Can ADQuaTe2 detect the faults in real-world data that were missed by domain experts?

Table 1: Known Faults and Suspicious Records in Real-world Datasets Detected by ADQuaTe2

Dataset ID	|E|	|A|	PD	SD	UD
1	0	89	0.00	100.00	0.00
2	19	19	100.00	0.00	0.00
3	5650	6848	98.25	17.50	0.017
4	4810	5070	96.77	8.50	0.032
5	0	5026	0.00	100.00	0.00
6	109980	132174	99.99	16.75	0.01
7	246000	249724	97.21	4.24	2.79
8	700	753	96.14	10.63	3.86

We asked domain experts to inspect the suspicious records. Given *AF* as the set of faulty records that are flagged by the domain expert as actually faulty, we define the *Newly Detected* (*ND*) as the percentage of suspicious records detected by ADQuaTe2 that are real faults not previously detected. ADQuaTe2 could detect between 33.33% to 35.63% actual faults that were not previously detected for a plant science and a health dataset. The experts interpreted the remaining detected records as *unusual* but possible, *suspicious* if they needed more investigations, and *valid* if they were not actually faulty. It took one hour for the plant domain expert to inspect the 16 groups of 89 reported records. It also took one hour for the health domain expert to inspect 23 groups of 6848 reported records.

3.2 Fault Interpretation Effectiveness

RQ2: To what extent do the generated visualization plots correctly explain the reason behind the invalidity of faults?

We define the *Visualization Efficiency* (*VE*) as the percentage of plots that could correctly explain the reason behind the invalidity of the suspicious groups to answer this question. Between 53.33% to 100.00% of the plots for the plant science and the health datasets correctly explained the reasons behind invalidity of the records.

3.3 Accuracy Improvement Using UCI Datasets

RQ3.a: Does the number of correctly detected faults increase after retraining the learning model with the help of the expert feedback?

Given *E*, the set of faulty records detected by an existing data quality test approach, *A*, the set of faulty records detected by ADQuaTe2, and *AF*, the set of faulty records that are flagged by domain expert as actually faulty, we use the *True Positive Rate* (TPR) and *Number of Runs* (NR). $TPR = \frac{|AF|}{|A|}$ is the percentage of actual faulty records that are correctly identified as faulty and *NR* is the total number of times an expert revalidates data until reaching the desired TPR.

RQ3.b: Can the actual faults detected by ADQuaTe2 in the previous runs still be detected after retraining the model?

To answer this question, we use *False Negative Rate* (FNR). $FNR = \frac{|AF_{old} - AF_{new}|}{|AF_{old}|} + UD$ is the percentage of undetected faults (UD) plus percentage of actual faulty records detected in previous runs that could not be detected in the current run, where AF_{old} is the set of actual faults detected in previous runs and AF_{new} is the one detected in the current run.

We implemented a script that automatically executes ADQuaTe2 against a dataset and detects suspicious records. The script updates the value of the label from 0.5 to 1 for the suspicious records that are actually faulty (i.e., within the previously known faults) and from 0.5 to -1 for the ones that are valid. Next, the script retrains the constraint discovery model and reruns the fault detection component (Section 2.5) and measures the accuracy of ADQuaTe2 based on the metrics described in this section. The whole process is performed 10 times. Figures 5 and 6 show how TPR increases and FNR decreases over time during the retraining process for the UCI datasets. These results show that the fault detection effectiveness of ADQuaTe2 improves after retraining the machine learning model. Moreover, the TPR and FNR are almost stabilized after four iterations, which shows that the payoff is not worth the cost of running ADQuaTe2 after four iterations for these datasets.

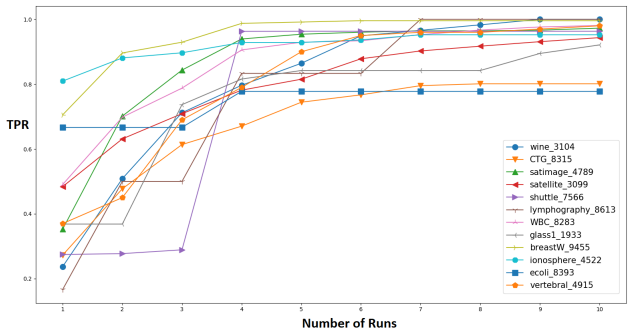


Figure 5: Improvement in TPR for UCI Datasets

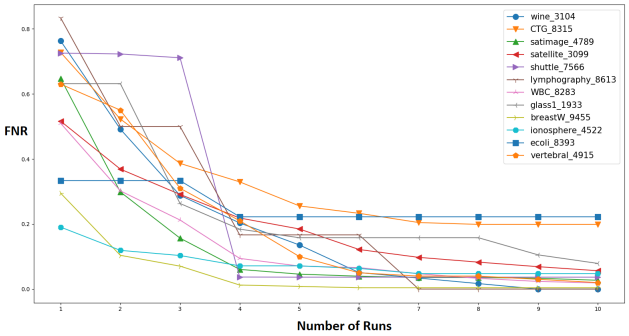


Figure 6: Improvement in FNR for UCI Datasets

3.4 Performance Evaluation

RQ5: How does the time it takes to execute ADQuaTe2 change based on the size of the datasets?

We measure the Total Time (*TT*) it takes to perform the automated steps of ADQuaTe2 to answer this question. The time spent by domain experts is not included because different experts do the fault inspection in different ways. Figure 7 shows values of *TT* for one execution of ADQuaTe2 for all the datasets based on their $size = NRe * NAt$, where *NRe* is the number of records and *NAt* is the number of attributes. It took between 0.138 to 27 minutes to execute the automated steps of ADQuaTe2 for these datasets. As

the results show, *TT* is not necessarily greater for the datasets with larger sizes. This shows that dataset characteristics other than size, such as data types and data sparseness may have also affected the results. The analysis of other effective factors on the performance of ADQuaTe2 is the subject of our future work.

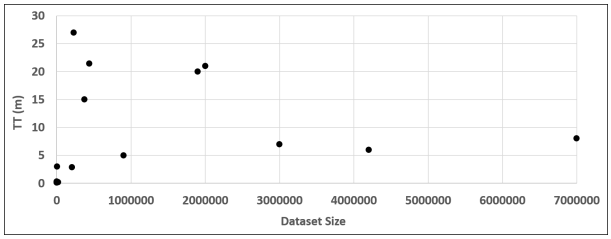


Figure 7: Total Time (TT) for Different Dataset Sizes

4 RELATED WORK

Machine learning-based approaches can detect the outliers [3] that violate semantic constraints in the data. Depending on the availability of labeled data, these techniques are classified as supervised (e.g., Naive Bayesian [19], Support Vector Machine (SVM) [20], and Artificial Neural Network (ANN) [21]), semi-supervised (e.g., OC-SVM [22]), and unsupervised (e.g., clustering [23]). Supervised techniques require domain experts to manually label training data, which is not scalable for big datasets. The training phase is restricted to a set of labeled data that are biased towards the expert's knowledge. The Naive Bayesian approach assumes a strong independence between the record attributes and cannot discover constraints that involve relationships among multiple attributes. SVM trains a hyperplane in the attribute space that best divides a labeled dataset into valid/invalid classes but the trained hyperplane is an equation over data attributes that is not human interpretable. ANN trains a network of information processing units that best classifies the records as valid/invalid but the network is not human interpretable. OC-SVM requires providing a clean dataset for the training phase. This technique is biased towards the definition of valid records by the experts. Distance-based clustering algorithms cannot derive relationships among attributes in their clusters [23].

5 CONCLUSIONS

We described ADQuaTe2, which is a data quality test approach for complex constraint discovery, fault detection, and explanation. ADQuaTe2 allows domain experts to inspect the reported suspicious records and feeds the constraint discovery and fault detection components with the information received from them to minimize false alarms. ADQuaTe2 uses ground truth knowledge to tune its parameters. Our approach discovered new constraints in the attributes of health and plant science data that were missed by domain experts and detected faults that were not previously detected by the existing tools. Using ground truth UCI datasets that contain a set of known faults, we demonstrated that the true positive and false negative rates improved after incorporating the ground truth knowledge and retraining the learning model. Using the same datasets, we demonstrated that the training process must be stopped before the network is overfitted on the training data. We plan to extend ADQuaTe2 to support constraint discovery over multiple records using time series analysis techniques. We also plan to automatically generate data quality test assertions from the discovered constraints.

ACKNOWLEDGMENT

This work was supported in part by funding from NSF under Award Numbers CNS 1650573, CNS 1822118, OAC 1931363, CableLabs, Furuno Electric Company, SecureNok, AFRL, and NIST.

REFERENCES

[1] J. Bresnick, "Patient Safety Errors are Common with Electronic Health Record Use," https://healthitanalytics.com/news/patient-safety-errors-are-common-with-electronic-health-record-use (Accessed 2020-04-13).
[2] "Informatica Data Quality Test Tool," https://www.informatica.com/ (Accessed 2020-04-11).
[3] C. C. Aggarwal, "An introduction to outlier analysis," in *Outlier Analysis*. Springer International Publishing, 2017, pp. 1–34.
[4] B. N. Saha, N. Ray, and H. Zhang, "Snake Validation: A PCA-based Outlier Detection Method," *IEEE Signal Processing Letters*, vol. 16, no. 6, pp. 549–552, 2009.
[5] H. Homayouni, S. Ghosh, and I. Ray, "ADQuaTe: An Automated Data Quality Test Approach for Constraint Discovery and Fault Detection," in *IEEE 20th International Conference on Information Reuse and Integration for Data Science*, Los Angeles, CA, 2019, pp. 61–68.
[6] H. Homayouni, S. Ghosh, I. Ray, and M. Kahn, "An interactive data quality test approach for constraint discovery and fault detection," in *IEEE Big Data*, Los Angeles, CA, 2019, pp. 200–205.
[7] C. Zhou and R. C. Paffenroth, "Anomaly Detection with Robust Deep Autoencoders," in *23rd ACM International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 665–674.
[8] T. Kohonen, "The Self-Organizing Map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.
[9] B. Kaminski, M. Jakubczyk, and P. Szufel, "A Framework for Sensitivity Analysis of Decision Trees," *Central European Journal of Operations Research*, vol. 26, no. 1, pp. 135–159, 2018.
[10] R. M. Konijn and W. Kowalczyk, "An Interactive Approach to Outlier Detection," in *Rough Set and Knowledge Technology*, J. Yu, S. Greco, P. Lingras, G. Wang, and A. Skowron, Eds. Springer Berlin Heidelberg, 2010, vol. 6401, pp. 379–385.
[11] "UCI ML Repository," https://archive.ics.uci.edu/ml/index.php (Accessed 2020-04-12).
[12] W. Zhang, T. Du, and J. Wang, "Deep Learning over Multi-field Categorical Data," in *Advances in Information Retrieval*, ser. Lecture Notes in Computer Science, N. Ferro, F. Crestani, M.-F. Moens, J. Mothe, F. Silvestri, G. M. Di Nunzio, C. Hauff, and G. Silvello, Eds. Springer International Publishing, 2016, pp. 45–57.
[13] L. A. Shalabi and Z. Shaaban, "Normalization as a Preprocessing Engine for Data Mining and the Approach of Preference Matrix," in *International Conference on Dependability of Computer Systems*, 2006, pp. 207–214.
[14] G. Zhong, L.-N. Wang, X. Ling, and J. Dong, "An Overview on Data Representation Learning: From Traditional Feature Learning to Recent Deep Learning," *Journal of Finance and Data Science*, vol. 2, no. 4, pp. 265–278, 2016.
[15] V. Cherkassky and F. M. Mulier, *Learning from Data: Concepts, Theory, and Methods*, 2nd ed. Wiley-IEEE Press.
[16] P. B. de Laat, "Algorithmic Decision-Making based on Machine Learning from Big Data: Can Transparency Restore Accountability," *Philosophy & Technology*, vol. 31, no. 4, pp. 525–541, 2018.
[17] "Achilles," https://github.com/OHDSI/Achilles (Accessed 2019-02-12).
[18] "Murdock," https://murdock-study.com/ (Accessed 2019-06-24).
[19] P. Lam, L. Wang, H. Y. T. Ngan, N. H. C. Yung, and A. G. O. Yeh. (2017) Outlier Detection in Large-Scale Traffic Data by Naive Bayes Method and Gaussian Mixture Model Method.
[20] C. Cortes and V. Vapnik, "Support-vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
[21] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion Detection by Machine Learning: A Review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.
[22] S. Agrawal and J. Agrawal, "Survey on Anomaly Detection using Data Mining Techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.
[23] G. Gan, C. Ma, and J. Wu, *Data Clustering: Theory, Algorithms, and Applications*. Society for Industrial and Applied Mathematics, 2007.