ExPAD: An Explainable Distributed Automatic Anomaly Detection Framework over Large KGs

Farshad Bakhshandegan Moghaddam
University of Bonn
Bonn, Germany
farshad.moghaddam@uni-bonn.de

Jens Lehmann
TU Dresden, Amazon
(work done outside of Amazon)
Dresden, Germany
jens.lehmann@tu-dresden.de

Hajira Jabeen
GESIS-Leibniz Institute for the Social Sciences
Cologne, Germany
hajira.jabeen@gesis.org

***Abstract*—RDF data is playing an important role in publishing and integrating heterogeneous data in data lakes. However, since the data is generally created utilizing liberal curation approaches such as crowd-sourcing and automatic extraction tools with no cross-validation on input data, the data is prone to errors that can be hidden in several dimensions. The types of errors which can be considered as outliers may occur in any part of RDF statements, especially in literal objects. Although some scientific studies have revealed anomalies in knowledge graphs, none of the current approaches has the ability to explain the anomalies that have been discovered. In this paper, we present ExPAD, a scalable and distributed framework for explainable numeric anomaly detection on very large RDF knowledge graphs. Inspired by OutlierTree, ExPAD generates human-readable explanations for a given numeric result being an outlier by following and assessing supervised decision tree splits. The proposed framework ExPAD is open-source, well-documented, and fully integrated into the Semantic Analytics Stack (SANSA). Experiments on real-world use cases and synthetic datasets disclose that the framework can not only handle high volumes of RDF data but also efficiently generate explanations for hidden anomalies discovered in KGs.**

***Index Terms*—RDF, Explainable Anomaly Detection, Big Data, Distributed Computing, SANSA, Decision Tree**

I. Introduction

Anomaly Detection (AD) (interchangeably known as Outlier Detection) is a field of Machine Learning (ML) that tries to identify rare items, events or observations which deviate significantly from the majority of the data [1], [2]. Although AD is a well-studied field in the area of machine learning and statistics, there is a major gap of AD in the area of Semantic Web and large-scale heterogeneous Knowledge Graphs (KGs). Especially when it comes to interpretability, there is no research work in the area of explainable anomaly detection in KGs.

The Semantic Web enables a structural view of existing data on the Web and provides machine-readable formats such as the Resource Description Framework (RDF)[3]. RDF data are a collection of triples $\langle subject, predicate, object \rangle$ which tend to have rich relationships, forming a potentially very large and complex graph-like structure. These graphs (KGs) are being generated in a variety of ways. Some KGs such as Wikidata [4] have been created by crowd-sourcing. DBpedia [5] has been created by automatically extracting knowledge tools and NELL [6] has been created by natural language processing techniques. As the entered data are generally neither constrained nor cross-validated, KGs are prone to various types of errors because of the range of approaches and freedom in inserting the input data. These errors can happen at *Subject*, *Predicate* or *Object* part in the RDF format. For example there can be errors in the *predicate* part, such as a person has n birth places[1] which $n > 1$ (normally, a person has only one birthplace). Or there can be extraction errors like parsing errors, e.g. in some cases the extraction tool can not interpret "-" and converts "1989-2000" in Wikipedia[2] to "19892000" in DBpedia[3].

Although numerous techniques for detecting outliers and anomalies in unstructured collections of multiple dimension points have been developed in recent years, with the current interest in large-scale heterogeneous data in Knowledge Graphs, most of the traditional algorithms are no longer directly applicable on KGs due to scalability and RDF complex data structure. Furthermore, uncovering why a reported anomaly have occurred (explanation discovery), forms a crucial capability for any anomaly detection system.

In this paper, we propose ExPAD, a generic, distributed, and scalable software framework that can automatically detect numeric anomalies in KGs and produce human-readable explanations for why a given value of a variable in an observation can be considered as outlier. ExPAD is inspired by OutlierTree [7] and works by evaluating and following distributed supervised decision tree splits on variables. This helps to detect and explain anomalous cases which can not be detected without considering other features. For example, a person's age can be 2 and it is reasonable, however, the age of a person who is a CEO of a company can usually not be 2. Under this logic, it is possible to produce human-readable explanations for why a given value of a variable in an observation can be considered as outlier, by considering the decision tree branch conditions.

Furthermore, given our main focus on distributed computing and the ability to perform on large KGs, ExPAD is integrated into the SANSA Stack [8]. This integration ensures that the

[1]https://dbpedia.org/page/Alireza_Afzal
[2]https://en.wikipedia.org/wiki/Ian_Turbott
[3]https://dbpedia.org/page/Ian_Turbott

platform is maintained and continuously updated due to the community involvement in the project's operations.

To summarize, the main contributions of this paper are as follows:

- A scalable distributed anomaly detection framework that can automatically detect numeric anomalies in a large RDF graph and provide clear explanations for why certain values are identified as anomalies.
- Integration of ExPAD into the holistic SANSA stack and making it open-source and publicly available on GitHub [9]
- Covering the code by unit and integration tests, documenting it, and providing a tutorial within Databricks [9]
- Evaluation of the results over multiple datasets and empirical evaluation of scalability

The remainder of the paper is organized as follows: Section II provides an overview of the related work. Section III describes the architecture and implementation of ExPAD. Section IV is devoted to the experimental setup, the datasets, and scalability evaluation. Finally, Section V concludes the paper and discusses the prospects for further development of ExPAD.

II. Related Work

This section presents prior related studies on Interpretable Machine Learning, Anomaly Detection, and Anomaly Detection on KGs.

*a) **Interpretable Machine Learning.**:* Interpretable machine learning has recently gained a lot of attention [10] and there are two broad categories of relevant techniques: *interpretable models* and *model-agnostic methods*. Interpretable models such as linear regression and decision trees construct a human-readable model directly from data [10]. Model-agnostic approaches, on the other hand, decouple explanations from the ML model. There are several methods in the model-agnostic family. LIME [11] explains every classifier prediction by approximating it locally with an interpretable linear model. In [12] authors improved upon LIME by replacing its linear model with a logical rule for explaining a data instance. SHAP [13] and RESP [14] scores are both instance-level explanations that provide a numerical value to each attribute based on its relevance in the result.

As ExPAD generates the explanation directly from data, it can be classified under *interpretable models* family.

*b) **Anomaly Detection.**:* Anomaly Detection (AD) has a long history of study. Because of the wide range of data properties, anomaly types, and application domains, AD methodologies range from statistical methods [15], [16], distance-based [17], density-based [18], and isolation forests [19] to deep learning methods [20]. A comprehensive review of AD methods is beyond the scope of this paper; we refer the reader to previous survey publications for a thorough discussion [2], [20], [21].

We should mentioned that the mentioned approaches are generally not scalable and more importantly can not be directly applied on native RDF datasets.

*c) **Anomaly Detection Over KGs.**:* Although Anomaly Detection is already a well-studied field, to the best of our knowledge there has not been much dedicated research work on anomaly detection on big RDF data. [22] is one of the pioneers in the detection of erroneous numerical data in DBpedia. The authors conducted the search for numerical outliers in two phases. They began by categorizing the *subjects* depending on the *type*, and outliers are found using Interquartile Range in the next phase. The authors vectorized the entities using the FeGeLOD framework [23] and stated that the run-time on datasets with only two predicates - *DBpedia-owl:populationTotal* and *DBpedia-owl:elevation* is more than 24 hours due to the sluggish clustering approach. Another work [24] offered two independent anomaly detection approaches. They created a large sub-population in the first step and then used outlier detection on the sub-population. The *owl:sameAs* property was utilized in the second step to affirm or reject outliers and determine if an outlier is a natural outlier or not. Their solution, however, required manual querying of data to retrieve the information. CONOD [25] is the first scalable approach for detecting numerical outliers in DBpedia. It created cohorts using *rdf:type* and Linked Hypernyms Dataset (LHD) [26]. Cohorts, unlike clusters, have the potential to overlap. CONOD employed a scalable clustering technique based on Locality Sensitive Hashing (LSH) [27], however, as it uses *rdf:type* and LHD for cohorting, it is only applicable to DBpedia. DistAD [28] is another framework for anomaly detection on large RDF knowledge graphs. DistAD is scalable and provides a granularity for the end-users to select from a vast number of different algorithms, methods, and hyper-parameters to detect outliers. Although extraction part of ExPAD is partially inspired by DistAD, these frameworks are thoroughly distinctive. For example, DistAD does not provide any explanation for the detected anomalies. Moreover, the execution pipelines and the way that anomalies have been detected are completely different from the algorithmic perspective.

In summary, all the above-mentioned methods (except [25], [28]) are not scalable to large-scale knowledge graphs, are complex, and require manual intervention. Moreover, none of them provide explanation for the detected anomalies.

III. ExPAD as a Resource

We now present ExPAD, a generic distributed framework for Explainable Anomaly Detection in KGs. The main goal of the framework is to detect numeric anomalies in KGs and to provide human-readable explanations for why a given value can be considered as an outlier. ExPAD vectorizes RDF data (extracting literals from RDF which is explained in Section III-A2), then to detect anomalies on a given numerical variable v, it fits a Distributed Decision Tree[4], choosing v as the target and the rest of attributes as the features. Based on the depth of the tree, v is divided to two or more partitions. By applying any anomaly detection method (eg. IQR (Section III-A5a)) over the

[4]https://spark.apache.org/docs/latest/mllib-decision-tree.html

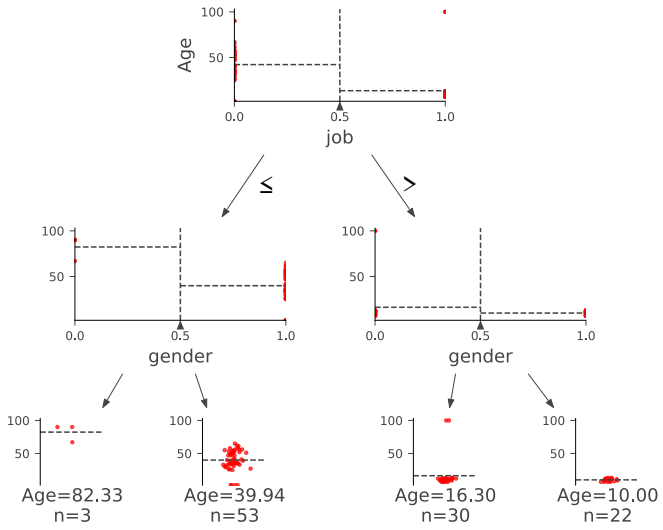


Fig. 1: Trained decision tree with the target variable *age*

partitioned data and finding anomalies, it is possible to produce explanations for why a given value of the target variable is identified as outlier, by considering the decision tree branches and their corresponding variables. An example is sketched here to shed the lights on working of ExPAD. Consider an RDF dataset containing information about the class *Person* with sample predicates *age (numerical)*, *job (URI)*, and *gender (boolean)* with 100 triples. After vectorizing and indexing this data (explained in Section III-A2), ExPAD considers one predicate (feature) as a target variable and rest of predicates as features for training a decision tree. Figure 1 depicts a fitted decision tree model with target feature *age*. As it can be seen each tree node (internal or leaf) partitions data. For example the root node partitions data based on *job* or the second leaf node from right partitions data based on *job* and *gender*. Based on the second leaf node from right, the estimated age for a person with job type 1.0 (this is a mapping index which is explained in Section III-A2) and gender 0.0 is 16.30, however based on this partitioning there is a person with age of $>= 100$ which is an anomaly in this scenario. Now this anomaly can be explained as "*the age value* $>= 100$ *is suspicious given job* $= 1.0$ *and gender* $= 0.0$" or after mapping indices to the original values "*the age value* $>= 100$ *is suspicious given job=student and gender=female*". By this logic, the fitted decision tree can be parsed and for each node an SQL query can be extracted (e.g. `SELECT * FROM data WHERE job=1.0 AND gender=0.0`). We consider these queries as rules. Finally applying these extracted rules on the dataset and applying anomaly detection technique on the target values can yield explanations for the possible detected outliers.

a) Availability: To make the framework available for the community, all the components of it have been integrated into the SANSA ecosystem. The framework is fully available for the community as an open-source GitHub repository [9]. ***b) Impact:*** ExPAD offers Semantic Web practitioners a tool

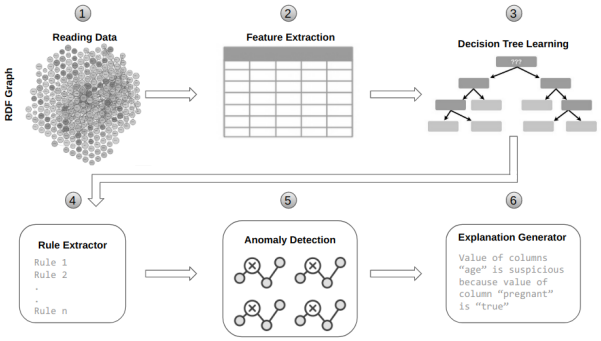


Fig. 2: System architecture high-level overview

to not only detect anomalies in their RDF dataset, but also provide an explanation and improve the quality of the existing dataset. Besides this, as it can handle huge RDF data, it can be a suitable option for enterprise companies in the field of energy and IoT to detect anomalous events in the RDFized sensor data[5]. ***c) Usability:*** Although the framework is fully documented but we also provided a tutorial and a sample in Databricks to assist end-users and significantly decrease the try-out barrier [9].

Figure 2 depicts the high-level system architecture overview of ExPAD. Moreover, Table I lists configurable parts of the framework. In the following, each step of the framework is explained in detail based on Figure 2.

A. Components

1) Step 1: Reading Data: The first step of the pipeline is reading and loading the RDF data. The output of this step will be a Spark dataframe. ExPAD supports *N-Triple* and *Turtle* file formats and the RDF data can reside in normal file systems or Hadoop File System (HDFS).

2) Step 2: Feature Extractor: The output of the first step is a dataframe with 3 columns which stores $\langle subject, predicate, object \rangle$, however, to be able to apply any anomaly detection algorithm on the $object$ part, the dataframe should be vectorized (Prepositionalized). This steps moves each $predicate$ to a separate column and reshape the dataframe accordingly. In this step we borrowed two integrated approaches from DistAD to apply vectorization.

a) Pivoting/Grouping: A pivot is an aggregation where one (or more in the general case) of the grouping columns has its distinct values transposed into individual columns. Spark provides a pivoting mechanism over dataframes.

b) Literal2Feature: Literal2Feature [29] transform a given RDF dataset to a standard feature matrix by deep traversing the RDF graph and extracting literals to a given depth. It generates a SPARQL query which is executed by the SANSA built-in SPARQL engine. The result of the SPARQL query execution is the vectorized RDF dataframe.

Besides above mentioned approaches, in ExPAD we implemented a new transformation pipeline entitled "*SmartDataFrame*". *SmartDataFrame* not only reads the RDF data

[5]https://platoon-project.eu

into a dataframe, but also extract features based on objects and after detection of data types, casts the literal to the primitive data-types. This transformer converts all features corresponding to their feature type into numeric representations. For example, non-categorical strings (e.g. URIs) are transformed into the label indexer mapping or boolean-type will be converted to $\{0, 1\}$. This transformation is necessary to make the corresponding decision tree learning possible. Listing 1 and 2 depicts a native RDF dataset featurized via *SmartDataFrame* transformer:

```
dbr:Person0 dbo:age "13"^^http://www.w3.org/2001/XMLSchema#integer
dbr:Person0 dbo:gender "false"^^http://www.w3.org/2001/XMLSchema#boolean
dbr:Person0 dbo:job dbr:Student
dbr:Person1 dbo:age "8"^^http://www.w3.org/2001/XMLSchema#integer
dbr:Person1 dbo:gender "true"^^http://www.w3.org/2001/XMLSchema#boolean
dbr:Person1 dbo:job dbr:Teacher
```

Listing 1: Sample RDF data

3) Step 3: Decision Tree Learning: The third step of the framework is fitting a distributed decision tree. For this reason, one of the originally numeric columns (features) in the dataframe will be considered as a *target* variable and the rest of columns as features. We only consider the original numeric features as target variables, because anomaly detection will happen on only these values in the next steps, however, the categorical ones will be involved as a feature in the decision tree training process. The maximum depth of the tree is a hyper-parameter, the deeper the tree goes, the more complex explanation will be produced due to the number of reported variables. This value is configurable by the end user. After the training is done, the decision tree model will be ready to be parsed in the next step.

4) Step 4: Rule Extractor: This component is responsible for parsing the fitted decision tree model and for each tree node (either internal node or leaf node) extracting an SQL rule. The parsing process uses DFS (Depth First Search) [30] to traverse the whole tree. The process keeps track of each variable in every split and generates an SQL query for each path. Consider the example at Figure 1, the generated rules are shown in Listing 3.

```
SELECT * from df WHERE job <= 0.5
SELECT * from df WHERE job > 0.5
SELECT * from df WHERE job <= 0.5 AND gender <= 0.5
SELECT * from df WHERE job <= 0.5 AND gender > 0.5
SELECT * from df WHERE job > 0.5 AND gender <= 0.5
SELECT * from df WHERE job > 0.5 AND gender > 0.5
```

Listing 3: Generated SQL rules

Worth noting that, instead of SQL, SPARQL rules could also be generated, however, as SQL is a native query language implemented over Spark dataframe, the execution time of SQL over dataframe is much shorter than running SPARQL on the RDF data in SANSA.

5) Step 5: Anomaly Detection: In ExPAD, we have provided anomaly detection methods for only numerical literals. For detecting anomalies in the numerical literals, we have integrated IQR, MAD, and Z-Score from DistAD [28].

a) Interquartile Range: The Interquartile Range (IQR) [15] technique is a statistical dispersion metric. It is based on determining the first quartile ($Q1$), median ($Q2$), and third quartile ($Q3$) of a numerical dataset. The difference between $Q3$ and $Q1$ is the IQR. Outliers are data points that are less than $Q1 - 1.5 \times IQR$ and more than $Q3 + 1.5 \times IQR$.

b) Median Absolute Deviation: The Median Absolute Deviation (MAD) [16] is a reliable measure of the variability of a univariate sample of quantitative data. It is more resistant to data set outliers than the standard deviation technique. The MAD for a univariate data collection $X_1, X_2, ..., X_n$ is defined as the median of the absolute deviations from the median of the data. So if $\widetilde{X} = median(X)$ then:

$$MAD = b \times median(|X_i - \widetilde{X}|)$$

where b is a constant scaling factor that changes with the distribution. Outliers are values in the input set X that are more than $\widetilde{X} + 2.5 \times MAD$ and less than $\widetilde{X} - 2.5 \times MAD$.

c) Z-Score: The number of standard deviations by which the value of a raw score (i.e., an observed value or data point) is above or below the mean value of what is being observed or measured is referred to as the Z-Score. Raw scores that are higher than the mean have positive standard scores, whereas those that are lower than the mean have negative standard scores. For example a Z-Score of 2.5 indicates that the data point is 2.5 units away from the mean, indicating that it is an outlier. The Z-score is defined as:

$$z - score = \frac{x - mean}{standarddeviation}$$

6) Step 6: Explanation Generator: Now that rules have been generated, one can apply each rule to the dataset to filter (fetch cluster) the data. This is possible due to Spark SQL[6] which is a Spark module for running SQL over dataframes. The result will be a dataframe as well. The target column will be selected to be checked for anomalies with an anomaly detection algorithm. In case any anomaly is detected, it will be reported alongside with the variables and their corresponding values in the SQL rule. For clarification, consider example at Figure 1. By applying `SELECT * from df WHERE job > 0.5 AND gender <= 0.5` and afterward running IQR over the filtered age values, one can see that the value 100 will be detected as anomaly. This value can be reported as "*the age value* 100 *is suspicious given job* $= 1.0$ *and gender* $= 0.0$" or after mapping indices to the original values "*the age value* $= 100$ *is suspicious given job=student and gender=female*".

The final output of the framework is the list of anomalous RDF triples and corresponding explanations that can be saved as a normal file on a file system or on HDFS.

B. Implementation

As Scala[7] is the programming language of SANSA, we chose the same and its APIs in Apache Spark[8] to provide the

[6]https://spark.apache.org/sql
[7]https://www.scala-lang.org
[8]https://spark.apache.org

```
+-----------+------------+---------------+-----------+--------------+
|s          |age__integer|gender__boolean|job__url   |job__url_index|
+-----------+------------+---------------+-----------+--------------+
|dbo:Person0|13          |0              |dbr:Student|0             |
|dbo:Person1|8           |1              |dbr:Teacher|1             |
+-----------+------------+---------------+-----------+--------------+
```

Listing 2: Transformed dataframe via SmartDataFrame

distributed implementation of ExPAD. Moreover, we benefit from SANSA IO layer for reading/writing RDF data. Technically, ExPAD can be divided into the following steps 1) read RDF data as a data frame with *SmartDateFrame* 2) generate feature matrix 3) train decision tree 4) traverse and parse fitted decision tree and generate rules 5) perform anomaly detection 6) generate explanation. Algorithm 1 depicts the framework execution pipeline.

Algorithm 1 ExPAD Workflow

Data: $G(E, V)$ Knowledge Graph as a RDF serialization
Result: possible anomalies and their explanation
$df = SmartDataFrame.read(G)$
$df = featureExtractor(df)$
foreach *column* $c \in df.cols$ **do**
  if c *is a numeric literal* **then**
    $target = c$
    $features = df.cols \setminus c$
    $dt = DecisionTree(df, target, features)$
    $rules = parse(dt)$
    foreach *rule* $r \in rules$ **do**
      $filteredDF = df.select(r)$
      $targetData = filteredDF(col = target)$
      $anomalies = detectAnomalies(targetData)$
      if *anomalies is not empty* **then**
        $report(anomalies, r)$

IV. Experimental Results

In this section, two sets of experiments will be conducted to analyze two particular aspects of ExPAD, effectiveness and computational performance. In the first experiment, the correctness of the extracted anomalies explanation will be analyzed over different datasets and in the second experiment, we will investigate the scalability of the framework. To the best of our knowledge there is no other work to simultaneously detect and explain anomalies on KGs via distributed computing stack. Therefore, in this section we introduce multiple scenarios to examine the quality of the ExPAD solely.

A. Dataset

To the best of our knowledge, there is no RDF dataset bench mark for anomaly detection and especially for explainability. Moreover, authors in [28] reported that most of the detected numeric outliers in DBpedia are found in the date/time related values and the reason is that the extraction tool can not always extract correctly the information related to date/time. For example `198?-85` in Wikipedia[9], parsed to `198^^xsd:integer` in DBpedia[10]. We should note that, although this type of anomalies can be successfully detected by ExPAD, however explaining them with respect to other variables may not be informative due to independentness. Therefore DBpedia can not be an adequate option for investigation because it contains many independent predicates which do not correlate with each other. To this end, and as there is no adequate RDF dataset benchmark, three datasets have been exploited. Engie-accident dataset[11], Titanic dataset[12], and our generated synthetic dataset. Engie SA[13] is a French multinational electric utility company that operates in the fields of energy transition, generation, and distribution, natural gas, nuclear, renewable energy, and petroleum. The accident RDF dataset contains data about $\sim$ 57K car accidents that occurred in France in 2018 (such as geo-coordinate, weather condition, ...). Titanic dataset contains information about passengers of the Titanic shipwreck. The dataset contains 12 features for 891 passengers. Beside this and to be able to have different size of datasets, we implemented an RDF data simulator which generates synthetic RDF graph. For the RDF data generator, we consider *Person* class with 5 synthetic properties and respective distribution listed in Table II.

B. Experiment A: Assessment of the detected Anomalies Explanation

In this section, we analyze the detected anomalies and explanations for all three mentioned datasets. A synthetic data set with manually added anomalies, Engie accident dataset, and RDFized version of Titanic dataset. For the synthetic data we generated $100K$ triples with the mentioned distribution in Table II. For the anomalies, we added 0.02% (= 20 cases) anomalous data; 10 pregnant women with age $\in [90, 100]$, 5 presidents with age $\in [4, 10]$, and 5 students with age $\in [70, 90]$. The reason why we only added a few anomalous cases is that anomalies should rarely happen, and in case they occur frequently they can not be detected. It is worth noting that the 5 generated predicates in the simulated data are deliberately defined dependent (e.g. pregnancy is related to the gender). Although adding other independent predicates (such as weather condition) won't cause any issue for the workflow,

[9]https://en.wikipedia.org/wiki/Steve_Walters
[10]https://dbpedia.org/page/Steve_Walters
[11]Can not be publicly published due to Intellectual Property concerns
[12]https://www.kaggle.com/c/titanic
[13]https://www.engie.com

TABLE I: ExPAD configurable components

Feature	Options	Comments
Feature Extraction	`Pivoting/Grouping`	Basic operation for extracting feature from RDF data
	`Literal2Feature[29]`	Sophisticated method for extracting features from RDF data
Decision Tree	`MaxDepth`	Maximum depth of the tree (nonnegative)
	`MaxBins`	Maximum number of bins used for discretizing continuous features
Anomaly Detection	`Interquartile Range[15]`	Used for numeric values
	`Median Absolute Deviation[16]`	Used for numeric values
	`Z-score`	Used for numeric values
General	`verbose`	Boolean to generate logs
	`AnomalyListSize`	The minimum number of samples required for anomaly detection method

TABLE II: Predicates used for generating synthetic RDF graph for class *Person*

Predicate	Value Type	Example	Distribution
id	non negative integer	{0,1,2,...}	incremental starting from 0
gender	boolean	{male,female}	50% male, 50% female
job	URI	{Student,President}	50% student, 50% president
pregnant	boolean	{true,false}	if male then false, if job=student then false, if job=president and age>55 then false, if job=president and age<=55 then 40%
age	positive integer	{1,2,3,...}	if job=student in [7,14], if job=president in [25,70]

however, our approach naturally (due to the nature of decision tree algorithm) can not generate meaningful explanation for the independent variables (a age of a person is not correlated to the weather condition and therefore can not be explained meaningfully via it).

In this scenario we set the maximum depth of tree to two, used Literal2Feature (depth=1) for extracting features, and used IQR for anomaly detection. Note that all of these values are configurable for the end user. Moreover, in this scenario, we only focused on the *age* predicate, although ExPAD performs on all the numeric variables. Figure 3 depicts the fitted decision tree on the $100K$ synthetic data with *age* as the target variable.

For the accident dataset, we used Literal2Feature with depth=5 for extracting features (due to the nested architecture of RDF in this dataset), used IQR for anomaly detection, and set the maximum depth of tree to two. Literal2Feature extracted 43 features from this dataset. In this case we executed ExPAD on geo-coordinates of the accidents.

For the Titanic dataset, we used RDFizer [31] to transform CSV data to RDF. It is beyond the scope of this paper to explain how RDFizer works, however, it uses RDF Mapping Language (RML)[14] rules for transformation of (un)structured data into RDF knowledge graphs. For this scenario we only used the train dataset provided by Kaggle[15]. For feature extraction we used pivoting approach, for anomaly detection IQR has been used, and set the maximum depth of tree to two.

After applying ExPAD on all of three datasets, all the 20 cases from synthetic data, 9 cases from Titanic dataset, and 5 cases from accident dataset have been detected. Table III shows some sample detected anomalies with the corresponding explanation. It is worth noting that there is a positive correlation between decision tree depth and the complication of explanation (number of variable reported in the explanation). Obviously as going deeper, the explanation may contain more variables. Comparing the detected anomalies from Titanic dataset with OutlierTree [7] results on the same dataset, revealed that ExPAD detected the anomalies correctly. Moreover, manual inspection of the detected anomalies in the accident dataset, reveal that although accidents location in the dataset are 'France Métropole', however, their geo-coordinates show that they lay outside France, in Asia and eastern Europe (Figure 4).

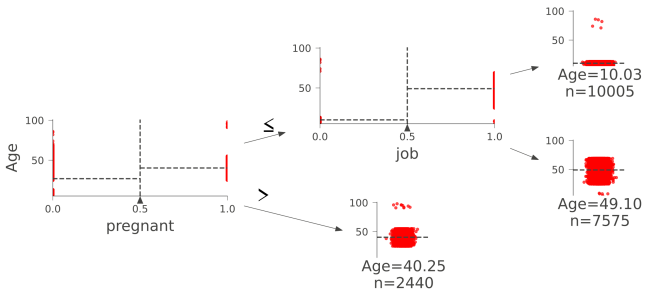


Fig. 3: Trained decision tree with the target variable *age*

Fig. 4: Detected anomalies on the Accident Dataset

[14]https://rml.io/specs/rml

[15]https://www.kaggle.com/competitions/titanic/data?select=train.csv

TABLE III: Sample of detected anomalies and their explanation

Dataset	Anomaly	Explanation
$100k$ Synthetic	$age = 100$	$age = 100$ is anomalous given pregnant = $true$
	$age = 80$	$age = 80$ is anomalous given pregnant = $false$ and job = $student$
	$age = 5$	$age = 5$ is anomalous given pregnant = $false$ and job = $president$
Titanic	$Fare = 0$	$Fare = 0$ is anomalous given Pclass = 3 and SibSp = 0
Accident	$longitude = 50$	$longitude = 50$ is anomalous given location='$France$'

TABLE IV: Synthetic dataset description

Dataset	File Size	#Triples
$DS1$	$500MB$	$1M$
$DS2$	$2.6GB$	$5M$
$DS3$	$5.2GB$	$10M$
$DS4$	$26GB$	$50M$
$DS5$	$52GB$	$100M$

C. Experiment B: Scalability

In this experiment we evaluate the scalability of ExPAD by different data sizes and varying cluster processing setups. To be able to have different size of datasets, we use the same RDF data simulator explained earlier to generates big synthetic RDF graphs. Table IV listed the generated datasets and their characteristics (Worth to mention that the German DBpedia size is 48GB). As running time of reading data from HDFS is same for all the configuration, and as the scalability of Literal2Feature and Pivoting has been intensively investigated in [29] and [28] respectively, for these experiments, we neglect the execution time of these components.

1) Scalability over number of cores: To adjust the distributed processing power, the number of available cores was regulated. In this experiment, we selected *DS3* (5.2GB) as a pilot dataset and the number of cores was increased starting from $2^2 = 4$ up to $2^7 = 128$. The experiments were carried out on a small cluster of 4 nodes (1 master, 3 workers): AMD Opteron(TM) CPU Processor 6376 @ 2300MHz (64 Cores), 256 GB RAM. Moreover, the machines were connected via a Gigabit network. All experiments are executed three times and the average value is reported in the results. Moreover, in these experiments we set the maximum depth of tree to 2, used pivoting for extracting features, and used MAD for anomaly detection. Figure 5(a) shows the scalability over different computing cluster setups. It is clear that increasing the computational power horizontally, decreases the execution time. Initially, doubling the number of cores reduces the execution time by nearly a factor of two. However, by adding more cores, the execution time only slightly decreases. This phenomenon is caused by the overhead of moving data between nodes as well as network latency. The maximum speed up is 5.1x.

2) Scalability over dataset size: To analyze the scalability over different datasets, we fix the computational power to 32 cores and run the experiments for all datasets introduced in Table IV. By comparing the run-time as shown in Figure 5(b), we note that the execution time does not increase neither linear nor exponentially. This behavior is due to the distribution among available resources e.g. (memory) and partition size. It can be seen that by increasing the dataset size from 500 MB to 52 GB ($\sim$ 100 times bigger), the execution time is almost only 19 times higher.

3) Scalability over number of features: For checking anomaly in each feature, a decision tree should be trained, therefore the complexity of ExPAD with respect to the number of features is linear. However, to show how decision tree training of ExPAD behaves when the data set dimensionality increases, we added 1000 uninformative numeric features to the *DS3* dataset which are drawn from a Gaussian distribution. Dealing with these random numbers can be much problematic for decision trees. Also we fixed the computational power to 32 cores, max decision tree depth to 3, and set *age* as the target variable. Figure 5(c) shows that increasing the dimensionality, increases the running time almost linearly. This is happening due to efficient implementation of decision tree in Spark. So it can be seen the framework is able to cope with even with the huge number of features.

V. Conclusion

In this paper, we introduced ExPAD, a generic, distributed, and scalable framework for explainable numeric anomaly detection in KGs. ExPAD is open-source, available on GitHub, and integrated into SANSA Stack. Inspired by OutlierTree, ExPAD generates human-readable explanations for outlier identification, by traversing supervised decision tree splits.

As ExPAD operates on univariate data, it may not be able to detect all multi-dimensional outliers (as opposed to other methods such as Isolation Forest that consider multiple variables simultaneously), however, it generates meaningful explanations for the dependent features. Moreover, our experiments show that the framework by detecting and explaining abnormalities can help in enhancing the data quality in KGs. Furthermore, our results indicate that ExPAD can be successfully scaled across a cluster of nodes for very large data sets.

In the future, we intend to work on explainable categorical anomaly detection in KGs. Moreover, in multivariate anomaly detection scenarios, it is important to be able to explain why a specific data point is considered as an anomaly.

Acknowledgment

This work was partly supported by the EU Horizon 2020 project PLATOON (GA no. 872592).

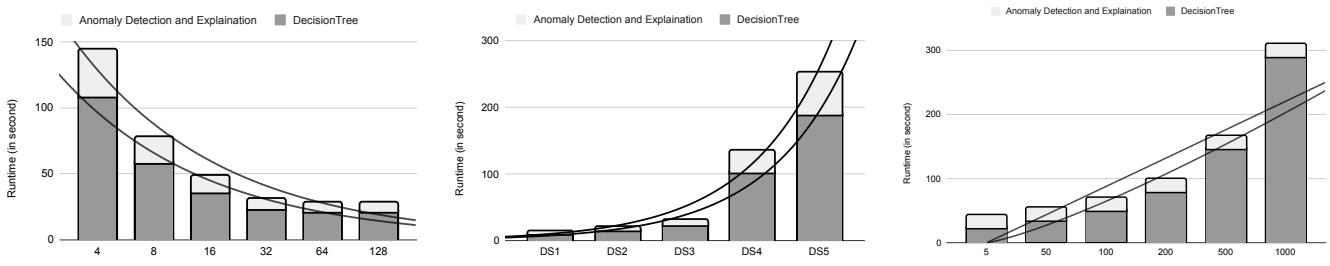
References

[1] D. Hawkins, *Identification of Outliers*. Chapman and Hall, 1980.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009.

[3] T. Berners-Lee, "A roadmap to the semantic web," http://www.w3.org/DesignIssues/Semantic.html, 1998.

[4] D. Vrandecic, "Wikidata: a new platform for collaborative data collection," in *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*. ACM, 2012, pp. 1063–1064. [Online]. Available: https://doi.org/10.1145/2187980.2188242



(a) (b) (c)

Fig. 5: (a) Processing power scalability on *DS3* (b) Size-up performance evaluation over 32 Core (c) Running time of ExPAD over large number of features

[5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives, "Dbpedia: A nucleus for a web of open data," in *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, ser. Lecture Notes in Computer Science, vol. 4825. Springer, 2007, pp. 722–735.

[6] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling, "Never-ending learning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

[7] D. Cortes, "Explainable outlier detection through decision tree conditioning," *arXiv preprint arXiv:2001.00636*, 2020.

[8] J. Lehmann, G. Sejdiu, L. Bühmann, P. Westphal, C. Stadler, I. Ermilov, S. Bin, N. Chakraborty, M. Saleem, A.-C. N. Ngonga, and H. Jabeen, "Distributed semantic analytics using the sansa stack," in *Proceedings of 16th International Semantic Web Conference - Resources Track (ISWC'2017)*. Springer, 2017, pp. 147–155.

[9] "Sansa team," https://github.com/SANSA-Stack/SANSA-Stack/releases/tag/v0.8.5_ExPAD, 2022.

[10] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[11] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[12] M. T. Ribeiro, S. Singh, and C. Guestri, "Anchors: High-precision model-agnostic explanations," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press, 2018, pp. 1527–1535. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982

[13] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[14] L. Bertossi, J. Li, M. Schleich, D. Suciu, and Z. Vagena, "Causality-based explanation of classification outcomes," in *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, 2020, pp. 1–10.

[15] R. McGill, J. W. Tukey, and W. A. Larsen, "Variations of box plots," *The American Statistician*, vol. 32, no. 1, pp. 12–16, 1978.

[16] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, 2013.

[17] L. Tran, L. Fan, and C. Shahabi, "Distance-based outlier detection in data streams," *Proc. VLDB Endow.*, vol. 9, no. 12, p. 1089–1100, aug 2016.

[18] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 93–104.

[19] T. Zemicheal and T. G. Dietterich, "Anomaly detection in the presence of missing values for weather data quality control," in *Proceedings of the 2nd ACM SIGCAS Conference on Computing and Sustainable Societies*, ser. COMPASS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 65–73.

[20] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, mar 2021.

[21] C. C. Aggarwal, "Outlier ensembles: Position paper," *SIGKDD Explor. Newsl.*, vol. 14, no. 2, p. 49–58, Apr. 2013.

[22] D. Wienand and H. Paulheim, "Detecting incorrect numerical data in dbpedia," in *The Semantic Web: Trends and Challenges*. Cham: Springer International Publishing, 2014, pp. 504–518.

[23] H. Paulheim and J. Fürnkranz, "Unsupervised generation of data mining features from linked open data," in *2nd International Conference on Web Intelligence, Mining and Semantics, WIMS '12, Craiova, Romania, June 6-8, 2012*. ACM, 2012, pp. 31:1–31:12.

[24] D. Fleischhacker, H. Paulheim, V. Bryl, J. Völker, and C. Bizer, "Detecting errors in numerical linked data using cross-checked outlier detection," in *The Semantic Web – ISWC 2014*. Cham: Springer International Publishing, 2014, pp. 357–372.

[25] H. Jabeen, R. Dadwal, G. Sejdiu, and J. Lehmann, "Divided we stand out! forging cohorts for numeric outlier detection in large scale knowledge graphs (conod)," in *Knowledge Engineering and Knowledge Management*. Cham: Springer International Publishing, 2018, pp. 534–548.

[26] T. Kliegr, "Linked hypernyms: Enriching dbpedia with targeted hypernym discovery," *Journal of Web Semantics*, vol. 31, pp. 59–69, 2015.

[27] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, ser. SCG '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 253–262.

[28] F. B. Moghaddam, J. Lehmann, and H. Jabeen, "Distad: A distributed generic anomaly detection framework over large kgs," in *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, 2022, pp. 243–250.

[29] F. B. Moghaddam, C. F. Draschner, J. Lehmann, and H. Jabeen, "Literal2feature: An automatic scalable rdf graph feature extractor," in *Proceedings of the 17th International Conference on Semantic Systems, SEMANTICS 2021, Amsterdam, The Netherlands, September 6-9, 2021*, 2021.

[30] D. C. Kozen, *Depth-First and Breadth-First Search*. New York, NY: Springer New York, 1992, pp. 19–24. [Online]. Available: https://doi.org/10.1007/978-1-4612-4400-4_4

[31] E. Iglesias, S. Jozashoori, D. Chaves-Fraga, D. Collarana, and M. Vidal, "Sdm-rdfizer: An RML interpreter for the efficient creation of RDF knowledge graphs," *CoRR*, vol. abs/2008.07176, 2020.