A scalable decision-tree-based method to explain interactions in dyadic data

Carlos Eiras-Franco[a,*], Bertha Guijarro-Berdiñas[a], Amparo Alonso-Betanzos[a], Antonio Bahamonde[b]

[a] *Universidade da Coruña. CITIC. Grupo LIDIA. Campus de Elviña, 15071 A Coruña, España.*
[b] *Universidad de Oviedo. Gijón, España.*

Abstract

Gaining relevant insight from a dyadic dataset, which describes interactions between two entities, is an open problem that has sparked the interest of researchers and industry data scientists alike. However, the existing methods have poor explainability, a quality that is becoming essential in certain applications. We describe an explainable and scalable method that, operating on dyadic datasets, obtains an easily interpretable high-level summary of the relationship between entities. To do this, we propose a quality measure, which can be configured to a level that suits the user, that factors in the explainability of the model. We report experiments that confirm better results for the proposed method over alternatives, in terms of both explainability and accuracy. We also analyse the method's capacity to extract relevant actionable information and to handle large datasets.

Keywords: dyadic data, machine learning, interpretable machine learning, explainable artificial intelligence, scalable machine learning

1. Introduction

Dyadic data [1] hold information regarding the interaction of two entities of any kind. They are pervasive in a variety of popular problems such as recommender systems [2], social science application analysis, market segmentation [3], computational linguistics, information retrieval, and preference learning [4], but also in more specific areas, such as automatic exam grading [5]. The large number of elements of each entity in such datasets yields an immense number of possible pair-wise interactions that is much larger than the recorded data. The sparsity of measurements can be overcome by using available data points to learn a *utility function* that generalizes the recorded data and predicts the outcome of each possible interaction. The very common problem associated with the learning of this utility function has usually been resolved using matrix factorization [2]. However, this procedure usually results in thousands of parameters and, despite the prediction accuracy for any given pair of elements, offers little insight into the nature of the relationship between two entities. A major problem in dealing with dyadic data therefore consists of identifying groups of entities with similar behaviour, so as to obtain

[*]Corresponding author: carlos.eiras.franco@udc.es

a high-level model of the studied environment. For instance, when analysing data from a book recommender system, a data scientist could look for groups of books that attract similar groups of readers with common reading interests. Characterizing such groups in terms of relevant information is a problem of great commercial interest since information on such small homogeneous groups would enable the development of more effective strategies tailored to specific groups. Such information, highly coveted by companies, is a difficult to obtain in practice, even when there is abundant data to analyse. The algorithms used to process this data should be computationally efficient in their capacity to handle large amounts of data, yet should be accurate enough to retrieve meaningful information and to yield results that are actionable and comprehensible for decision makers.

On a different note, as the complexity of machine learning models grows, predictions are becoming more accurate, yet these models are often hard to interpret and do not provide globally actionable information. It is therefore becoming increasingly important that the results of machine learning algorithms can be understood by a human supervisor. In some cases this is even required by law, as in the case of the right to explanation included in the new General Data Protection Regulation (GDPR) of the European Union[1]. The goal of Explainable Artificial Intelligence (XAI) (apparently the most common moniker, although sometimes also called Interpretable AI[2] AI or Transparent AI) is to obtain models that ideally should [7]:

1. Allow supervisors to interpret results so that it can be confirmed that they verify that the model goals are aligned with the desired goals (for instance, a credit rating system should not have gender or racial biases).
2. Justify predictions so as to enable a supervisor to formulate hypotheses that can be later verified, thereby excluding mere correlations due to randomness or dependence on some external factor,
3. Explain outputs in such a way that their generalizability can be established.
4. be informative, that is, it should offer the supervisor new information regarding the studied variables.

Interpretability can be achieved in one of two ways: (1) in the form of a transparent model that allows the supervisor to follow the "logic" behind every prediction, as in the case of production rules, or (2) as post-hoc interpretability, which consists of justifying a prediction through similar cases or through visualizations or other methods that identify the input features that led to a prediction. While post-hoc interpretability is the most common approach, it is limited to explaining individual cases and does not provide the supervisor with new general information about the modelled environment.

Finally, an additional problem is that, despite the immense amount of data available in some cases holding a great amount of valuable information, processing this data can be challenging because of its sheer volume. As a result, there is a need for scalable algorithms that can deal with very large datasets.

This work tackles the formal problem of obtaining relevant information from a utility function that codes the relationships existing between entities in a dyadic dataset. The proposed method splits the actors into easily interpretable groups with homogeneous behaviour. This

[1] https://ec.europa.eu/info/law/law-topic/data-protection_en

[2] Although some authors [6] make a distinction between the terms *interpretability* and *explainability*, in this work we use them interchangeably.

high-level summary of the data explains existing relationships and provides supervisors with meaningful information that will improve decision making processes. Moreover, implementation in the Apache Spark scalable distributed computing framework [8] enables the processing of large amounts of data to obtain relevant information within a reasonable timeframe.

The rest of the paper is structured as follows. Section 2 gives an account of existing methods in this field, Section 3 contains definitions of key concepts used in the proposed system, Section 4 describes the algorithm, Section 5 describes how the experiments performed to assess the suitability of the method were designed, Section 6 reports on and comments the results, and, finally, Section 7 summarizes the conclusions drawn and indicates future lines of research.

2. Related work

Although XAI is a nascent field, the number of proposed methods is increasing rapidly as shown in the latest survey papers [6, 9, 10]. The goal of obtaining an alternative to an opaque predictor, called the *Open the Black Box* problem, can be tackled in four different ways according to Guidotti et al. [9]:

1. *Transparent box design* consists of obtaining a classifier that uses a logic or methodology that can be directly interpreted by the supervisor.
2. *Model explanation* obtains a surrogate interpretable predictor that mimics the behaviour of the opaque model as closely as possible for any input.
3. *Outcome explanation* yields an explanation for a particular prediction of the classifier, offering post-hoc explainability.
4. *Model inspection* manipulates black box inputs to assess the magnitude of the effect of each variable in the prediction.

Our proposed method can be classified as either a *model explanation* of the utility function that predicts the nature of the relationship between any possible pair of actors, or as a *transparent box design* approach that obtains a grouping of the actors in a dyadic dataset. The resulting model is a shallow decision tree easily interpreted by the supervisor. Decision trees, along with rule systems and linear models, are considered to be easily interpreted, and using a single decision tree model as a surrogate for the black box model requiring explanation is a popular approach that started with the classic Trepan algorithm [11]. While several authors have iterated this approach [12, 13, 14, 15], none of their algorithms can be used to explain dyadic data.

For dyadic data, the most widely used methods to identify large-scale trends are segmentation and clustering techniques. Often problem-specific, they span market segmentation, document clustering and topic modelling, web user clustering and similar related fields. Our problem has been tackled using clustering algorithms [16, 17], self-organizing maps [18, 19], dimensionality reduction algorithms [20], evolutionary algorithms [21] and co-clustering algorithms [22]. Regarding interpretability, the mentioned algorithms have varying features. While evolutionary algorithms and co-clustering methods are not entirely suitable if interpretability is a goal, self-organizing maps and dimensionality reduction algorithms, although they do offer *post-hoc* explanations to the supervisor, do not clearly reveal links between the input variables that describe each entity, which reduces their effectiveness in motivating predictions and explaining outputs.

Lastly, a number of works have tackled explainability in the context of dyadic data. The importance of providing explanations for recommendations in the context of recommender systems has been established, and the effectiveness of different forms of explanation has been studied [23, 24]. An algorithm to obtain explainable clusters of users in the specific context of short text streams has been proposed [25], as a specific version of the topic modelling problem that cannot be used for generic dyadic data. Finally, the most similar work to our own is TEM [26] which consists of an embedding model enhanced with a classification tree in order to obtain explainable outputs. The algorithm uses an attention network to highlight the most relevant embeddings, which are then explained using the classification tree. However, TEM is limited to obtaining post-hoc explanations since the attentive embedding precludes the attainment of global explanations.

In conclusion, although the analysis of dyadic data is a popular field with many different approaches in the literature, the interpretable methods available are limited to post-hoc explanations of outputs. When tasked with obtaining an explanation of the relationships encoded in a utility function, the only possibility available to the practitioner is to use a generic clustering algorithm and then open the black box with an interpretable predictor.

3. Definitions

Dyadic data $\mathcal{X}$ describe interactions between two entities $\mathcal{U}$ and $\mathcal{I}$. Each data point $x \in \mathcal{X}$ is a (u, i, v) tuple, where $u \in \mathcal{U}$ and $i \in \mathcal{I}$ are the elements of each entity involved in the interaction and v is a value that informs of some characteristic of that interaction. For every u, i there is a data point x describing their relationship (x can be observed or predicted). Consequently, $\mathcal{X}$ can be represented with a function $f : (\mathcal{U}, \mathcal{I}) \to \{-1, +1\}$, commonly called a utility function. Note that v can take any value, but here we simplify the problem by transforming v to -1 or 1. Obtaining a prediction of this utility function is a very common problem that has usually been solved in the literature using matrix factorization [2, 27]. In addition, we define a clustering $Cl(\mathcal{U})$ over a dataset $\mathcal{U}$ as a set of m disjoint groups that contain every element in $\mathcal{U}$. The formulation is as follows:

$$Cl(\mathcal{U}) = \{Clu_1, \ldots, Clu_m\}. \tag{1}$$

The homogeneity of the utility function inside each group Clu_k can be used to establish the fitness of the clustering [28]. With this goal, we define the ratio p of positive elements in a group k for a given i_j that represents the j-th element in $\mathcal{I}$ as:

$$p_{kj} = Pr(+1|Clu_k, i_j) = \frac{|\left\{u \in Clu_k : f(u, i_j) = +1\right\}|}{|Clu_k|} \tag{2}$$

A group Clu_k is said to be *consistent* when there is good agreement in the values of f for the elements contained in the group, that is, p approaches 0 or 1. To measure that consistency as intended, we use the entropy of p.

$$H(p) = -p \log_2(p) - (1 - p) \log_2(1 - p). \tag{3}$$

$H(p_{kj})$ measures the consistency of a single group Clu_k with respect to i_j. To extend this measure to the whole clustering $Cl(\mathcal{U})$, every group and every element in $\mathcal{I}$ must be taken into account. Formally, we define the *weighted entropy (WE)* of a clustering $Cl(\mathcal{U})$ as:

$$WE(Cl(\mathcal{U})) = \sum_{k,j} \frac{|Clu_k|}{|\mathcal{U}||\mathcal{I}|} H(p_{kj}). \tag{4}$$

Bearing in mind the interpretability characteristics described in Section 1, here we focus on the ability of algorithms to motivate their predictions in an interpretable way using the input variables. We must therefore add a value to the fitness measure to evaluate the complexity of the explanation required to define each group, which we will estimate with the number of variables that describe the group. Consequently, we define the *quality* of a clustering as:

$$quality(Cl(\mathcal{U})) = -WE(Cl(\mathcal{U})) - \lambda \sum_{Clu_k \in Cl(\mathcal{U})} NV(Clu_k). \tag{5}$$

where NV represents the number of variables needed to describe Clu_k and λ is a hyperparameter that allows the supervisor to balance the entropy of the clustering with its interpretability. It can be shown we can intuitively expect a balance between the complexity of the explanation and the precision of the obtained clustering. Thus, to obtain very uniform groups one will generally need to form a large number of such groups which, in turn, will require a larger number of variables; however, both requirements decrease the interpretability of the clustering. Managing the trade-off between interpretability and accuracy is a desirable feature that avoids obtaining misleadingly oversimplified explanations while keeping interpretability to acceptable levels [6]. Hyperparameter λ allows the supervisor to manage this trade-off. This idea of factoring in both model accuracy and complexity is reminiscent of the well-known Akaike information criterion [29] and the Bayesian information criterion [30], although with significant differences that generalize the model to allow dyadic data handling and explainability through trees.

It is worth noting that it is irrelevant that the *quality* measure is a negative number. The goal of the algorithm is maximizing its value to approach 0. This general quality measure can be applied in any clustering built with whatever method for a dyadic dataset.

4. Proposed algorithm

We describe a new explanatory algorithm for dyadic data that obtains groups that are as homogeneous as possible and that are simultaneously explained using as few of the input variables as possible. To achieve this, a binary decision tree is built and a clustering $Cl(\mathcal{U})$ is defined by considering each leaf node as a separate group that is described by the variables that lead to that node. Note that this contrasts the available alternative, which is to use separate models for the clustering and the explanation tree, producing inferior results. It is also worth noting that the obtained groups are described with a varying number of attributes and the result can therefore be considered as a subspace clustering of the data.

The main process consists of finding the tree that maximizes the *quality* of the resulting clustering defined using Eq. 5. Certain simplifications are needed in order to efficiently explore the solution space. First, as mentioned above, the decision tree is binary because only dichotomous splits are contemplated (a common simplification when building decision trees). Also, to facilitate the calculation of the *quality* of a clustering, given that the number of elements in $\mathcal{I}$ can prevent accurate calculation in a reasonable time, a significant random sample of $\mathcal{I}$ is considered instead of the entire $\mathcal{I}$ set. This is done by performing a previous clustering on $\mathcal{I}$ using a standard algorithm such as K-Means and using centroids ci_j as representative points; in the event that the input space structure does not allow for K-Means to be computed, any sampling procedure that obtains a reduced number of representatives of $\mathcal{I}$ could be used. Once the representatives are computed, the ratio p is estimated using a variation of Eq. 2 where i_j includes only the selected representatives rather than each possible item. Analogously, addends in Eq. 4 are computed for each representative and divided by the number of items belonging to the represented cluster.

In addition, since exploring every possible decision tree built as described is unfeasible, a search strategy is mandatory. First, to prevent the tree from splitting at any possible value of each input variable, the number of split points must be reduced. A maximum number of split points is therefore established for each variable. In the case of numerical variables these points are determined using discretization. We modelled the search procedure after the C4.5 algorithm [31], adapting the entropy calculation to a multi-label context. Our implementation differs from existing multi-label versions of C4.5 [32] in that the weighted entropy measure that we use factors in the size of the groups directly.

A single tree of L_{MAX} levels is built by performing a greedy search in which, for each node, the candidate with the best weighted entropy is selected. This process is recursively repeated for the new groups obtained after the split, until a given L_{MAX} level is reached, as previously selected by the user. To expand the reach of this solution space exploration and so increase the possibilities of achieving a good solution,at each step the proposed algorithm explores not only the candidate with the best weighted entropy but the N best candidates. This spawns N possible trees that, when exploring the next level, will each generate $2 * N$ different possibilities. This process makes the number of explored trees grow exponentially with L_{MAX}. For this reason, the selected values for the N and L_{MAX} hyperparameters should be low. Once all possible trees are generated, the tree that defines the clustering with the highest quality is selected. The resulting clustering defined by this tree will consist of a maximum of $2^{L_{MAX}}$ groups.

This entire process is described in Algorithm 1.

```
Data: $\mathcal{U}, L_{MAX}, N$
Result: Decision tree that determines the clustering.
function buildTree($\mathcal{U}$, $level$, $splitPs$, $L_{MAX}$, $N$) $\rightarrow$ $best$
    if $level > L_{MAX}$ then
        return $\emptyset$
    end
    $candidates \leftarrow$ sorted list with capacity N;
    for $(variable, value) \in splitPs$ do
1       $left \leftarrow \{u \in \mathcal{U} : u[variable] < value\}$;
        $right \leftarrow \{u \in \mathcal{U} : u[variable] > value\}$;
        if WE($left$)*$left.size$+WE($right$)*$right.size$ < $candidates.max$ then
2           $candidates.add((variable, value))$;
        end
    end
    $best \leftarrow \emptyset$;
    for $(variable, value) \in candidates$ do
3       $left \leftarrow \{u \in \mathcal{U} : u[variable] < value\}$;
4       $right \leftarrow \{u \in \mathcal{U} : u[variable] > value\}$;
5       $leftTree \leftarrow$ buildTree($left$, $level + 1$, $splitPs$, $L_{MAX}$, $N$);
6       $rightTree \leftarrow$ buildTree($right$, $level + 1$, $splitPs$, $L_{MAX}$, $N$);
7       $newTree \leftarrow (variable, value, leftTree, rightTree)$;
8       if WE($newTree$) > WE($best$) then
9           $best = newTree$;
        end
    end
    return $best$;
end
$splitPs \leftarrow$ list of split points for every variable;
return buildTree($\mathcal{U}$, 0, $splitPs$, $L_{MAX}$, $N$);
```

Algorithm 1: Explanatory tree construction algorithm.

Lastly, in some cases the clustering defined by the retrieved tree may have less quality than the clustering defined by a subset of that tree. To address this, pruning is implemented; nodes are examined from level $L_{MAX} - 1$ to the root and any splits that do not have a positive impact on the overall *quality* are removed, as described in Algorithm 2.

```
Data: $tree, \lambda$
Result: Pruned tree.
function PRUNE($tree, \lambda$) $\rightarrow tree$
1   if ISLEAF($tree$) then
2       return $tree$;
    end
3   $left \leftarrow$ PRUNE($tree.leftBranch, \lambda$);
4   $right \leftarrow$ PRUNE($tree.rightBranch, \lambda$);
5   $splitEntropy \leftarrow \frac{left.entropy*|left|+right.entropy*|right|}{|tree|}$;
6   $\Delta E = splitEntropy - tree.entropy$;
7   $\Delta NV = left.numVars + right.numVars - tree.numVars$;
    if $-\Delta E - \lambda \Delta NV <= 0$ then
8       $tree.leftBranch \leftarrow \emptyset$;
9       $tree.rightBranch \leftarrow \emptyset$;
    return $tree$;
end
```

Algorithm 2: Pruning algorithm.

This algorithm consists of calculations that can be performed in parallel since they are independent of each other, so, to take advantage of this feature, the algorithm was implemented in the Apache Spark distributed computing framework. By leveraging distributed computing, the scalability of the algorithm is greatly increased, which, in turn, enables the analysis of large datasets in manageable times. The Apache Spark implementation of the clustering algorithm and all data transformation procedures are available for download from `https://github.com/eirasf/Dyadic-Explanation-Tree`.

5. Experimental setup

To assess the validity of the algorithm we performed two sets of experiments. We applied the method first to two real-world large datasets, measured the quality of the obtained explanation and compared the results with those for an alternative approach consisting of using two separate models: a generic clustering algorithm and an interpretable predictor that opens the black box (see Section 2). To the best of our knowledge there is no other method in the literature that can be used to explain dyadic data. Therefore, since the goal of this experiment is to establish whether our method offers better results than using separate clustering and explanation algorithms, we chose K-Means as the most common clustering algorithm and then selected a CART tree as the explanatory model so that the obtained tree is built similarly to our method, which contributes to a fairer comparison of the alternative approaches. Moreover, the use of entropy as the impurity measure for the CART tree assures that the quality obtained with this approach is as high as possible. Another experiment was performed to measure the effect on execution time of adding

Table 1: Dataset description.

Original datasets			
Dataset	Attributes		Samples
Outbrain_DAY1	667		1,445,196
MovieLens 20M	2,154		20,000,263
Transformed datasets			
Dataset	Explanatory	Defining	Samples
Outbrain_DAY1	667	100	1,445,196
UsersEx	1005	100	138,493
UsersExWithPrediction	1005	100	138,493
MoviesEx	1149	100	10,369

further computational nodes to the distributed calculation; this was done to test the scalability of the presented algorithm in the implementation in Apache Spark.

The first dataset we selected was one published by the advertising company Outbrain, made public as the subject of a competition hosted in the popular machine learning site Kaggle.com. Outbrain suggests new news content that may be of interest to readers. The dataset[3] records the page views of a number of users in a variety of news-related websites over the span of 14 days and, since the documents refer to current issues, the recorded views vary in terms of topic at different times; consequently, it was advisable to split the dataset into smaller parts that covered a shorter time span. For the purposes of this research, we used some 1,450,000 records consisting of 667 variables that were collected on the first day. Our second choice was the popular *MovieLens 20M* dataset [33], commonly used to test recommendation systems. This dataset reflects interactions, in the form of some 20 million ratings, between 138,000 users and 27,000 movies. After the required transformations (described in the next subsection), the dataset contained some 20 million examples with 2,154 variables. The dimensions of the datasets are summarized in Table 1.

5.1. Dataset transformation

In order to apply the algorithms, the datasets needed to be formatted appropriately so that each sample represented an interaction between an element u in an entity $\mathcal{U}$ and all the representatives ci_j of the opposing entity $\mathcal{I}$. Therefore, each sample contained the variables that characterize u, which we refer to as *explanatory* variables, and a vector of values $(r(u, ci_0), \ldots, r(u, ci_j))$ where $r(u, ci_j) \in \{-1, 1\}$ qualifies the relationship between u and representative ci_j, which we call *defining* variables.

The Outbrain dataset consists of elements that represent a page view by each user and that contain information about the user, the viewed document and the result of the interaction between the user and the offered sponsored links referring to other documents. While documents are characterized by numerous variables, including the publisher, category, topics covered and entities mentioned, users are solely described by their position (latitude and longitude) and the

[3]Available for download from `https://www.kaggle.com/c/outbrain-click-prediction`

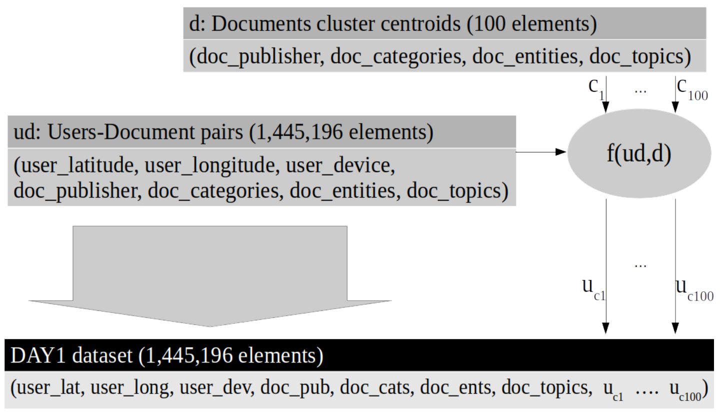

d: Documents cluster centroids (100 elements)
(doc_publisher, doc_categories, doc_entities, doc_topics)
c1 ... c100
ud: Users-Document pairs (1,445,196 elements)
(user_latitude, user_longitude, user_device, doc_publisher, doc_categories, doc_entities, doc_topics)
f(ud,d)
uc1 ... uc100
DAY1 dataset (1,445,196 elements)
(user_lat, user_long, user_dev, doc_pub, doc_cats, doc_ents, doc_topics, uc1 .... uc100)

Figure 1: Transformation performed to obtain the Outbrain_DAY1 dataset.

type of device used. To overcome the lack of information regarding users, we used the transformed dataset obtained by Luaces *et al.* [28] in which the attributes of the viewed document are added to the characterization of the user. Also, although the aim of the original Outbrain competition was to predict the most effective sponsored links in a given situation, the problem tackled here is different, namely, to obtain an explanation of the relationships between the user-document pairs and the sponsored links.

The user-document pairs were thus grouped according to their behaviour in order to obtain a high-level summary. Records corresponding to two consecutive page views by the same user were located and the preference of the user for one document over others was recorded so as to obtain preference tuples. These tuples conformed the dyadic dataset from which the utility function learned using matrix factorization [28]. The interactions of each tuple with a set of document representatives were selected to form the defining vector. These document representatives were the centroids of a $k = 100$ K-Means of the documents. This process is represented in Fig. 1.

The *MovieLens 20M* dataset consists of a large number of ratings that directly describe relationships between users and movies. Movies are characterized by the year of release, a vector of 20 possible non-exclusive genres and a vector of 1,128 tags, totalling 1,149 variables per movie. Users, in contrast, are only identified with an ID. To solve this problem of a lack of user information, we represented users with a vector containing their ratings of the most popular movies, considering popular movies to be those rated at least 5,000 times. This yielded a total of 1,005 popular movies. Users u were therefore described by a vector spanning 1,005 components of the form $\{-1, 0, 1\}$ where -1 represents a negative rating for a movie, 0 represents no rating and 1 represents a positive rating m. We modelled the utility function $f(u, m)$ that predicts ratings to be of the form

$$f(u, m, W, V) = \sigma(< Wu, Vm >) = \frac{1}{1 + e^{-<Wu,Vm>}} \tag{6}$$

where W,V are parameter matrices to be learned that project users and movies in a common space with fewer dimensions than the input space; in this case we selected a space with a dimension

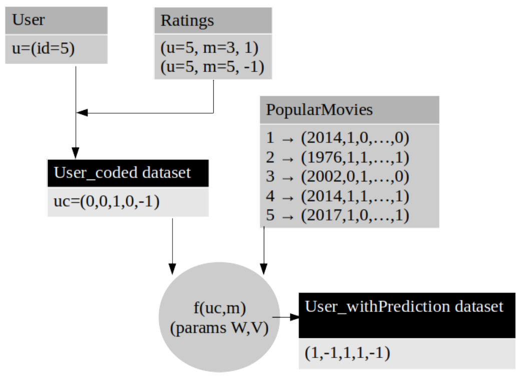


Figure 2: Example demonstrating the two options regarding the codification of a user.

equal to 200. By establishing a cost function:

$$J(W, V) = \sum_{(u,m)\in\mathcal{X}} -\log\sigma(r(u,m)\dot{<}Wu, Vm>) \tag{7}$$

where $r(u,m) \in \{-1, 1\}$ is the rating given to movie m by user u, the parameter matrices can be learnt using Stochastic Gradient Descent, a very common approach to this problem analogous to that used elsewhere [28].

This approach can be used with any dyadic dataset for which there is little or no information describing one of the entities. Nevertheless, in some cases, for datasets where the data available is sparse, the number of zeros in the user coding vector can become too large, increasing the similarity between users and complicating the decision tree task. To circumvent this problem, the user coding can be used to learn the utility function and can then be substituted by the predicted ratings for the most popular movies for that user. For a given popular movie pm_i, the user u coding vector will be 1 in component i if $f(u, pm_i) > 0.5$ and -1 otherwise. Both options are compared in Fig. 2 and their effectiveness for this particular dataset was tested as reported in Section 6.

Once users were coded and the utility function was learned, the projected movies Vm were clustered using K-Means with $k = 100$, yielding 100 movie representatives. A dataset, named *UsersEx*, was constructed by appending, to the coding for each user (explanatory), their rating of each movie representative (*defining*). Once the users were coded using the prediction, the dataset obtained using the same procedure was called *UsersExWithPrediction*. Analogously, 100 user representatives were obtained by clustering the projected users Wu using K-Means with $k = 100$. A third dataset, named *MoviesEx*, was constructed by appending, to the coding of each movie, the utility of each projected user representative. Both these datasets allowed us to obtain two complementary explanations of the data, as will be further explained in Section 6. The complete pipeline is depicted in Fig. 3.

The datasets resulting from the transformations are described in Table 1. Note that the number of movies was reduced to 10,369, since many of them did not have any ratings in the dataset. Also, although the number of explanatory variables depends on the number of attributes characterizing each entity, the number of defining variables was always 100 since in all cases we used the relationship of each element with 100 representatives of the opposing entity to characterize behaviour.

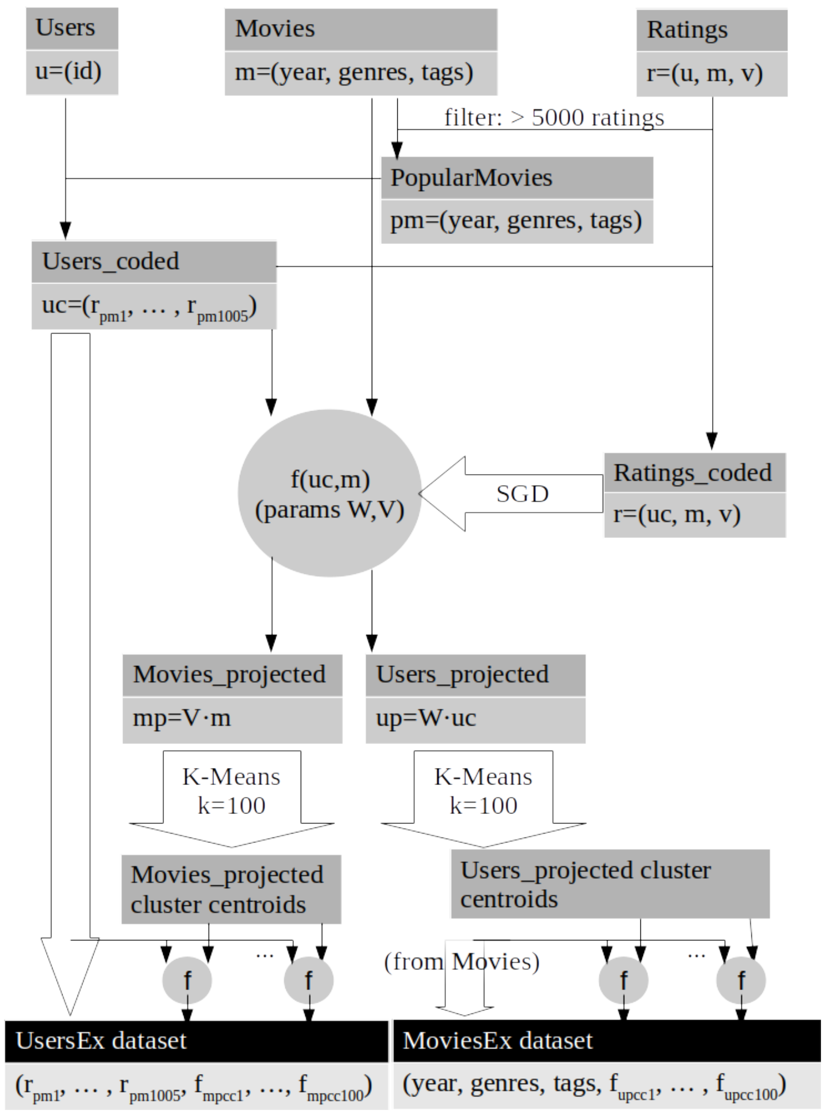


Figure 3: Transformations performed to convert the *MovieLens 20M* dataset in the *MoviesEx* and *UsersEx* datasets.

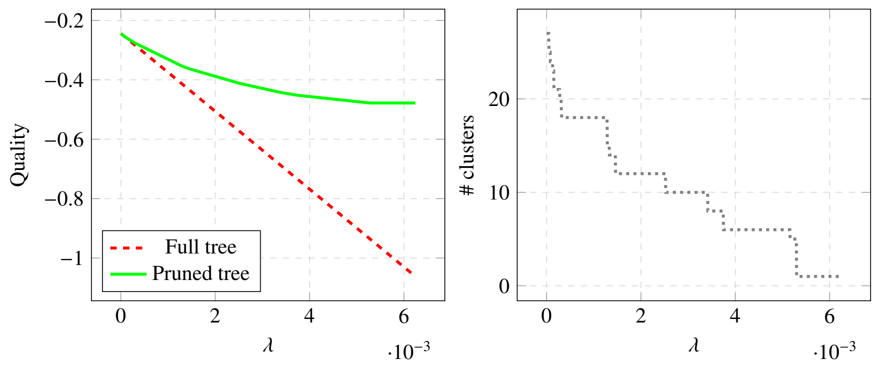


Figure 4: Pruned vs. original tree quality for the *Outbrain_DAY1* dataset (left). Number of nodes in the resulting pruned tree (right).

6. Results

In our first set of experiments we undertook the construction of an explanatory tree for the datasets and compared its quality to that of the explanation obtained by using a clustering algorithm and a separate model explainer. In order to perform these experiments, the values of the hyperparameters λ, N and L_{MAX} needed to be set. We used $N = 5$ and $L_{MAX} = 5$, which originated a 5-level binary tree that, consequently, described 32 clusters characterized by 5 variables each – considered to be a reasonable upper threshold for the complexity of the explanatory tree.

6.1. Effect of the λ hyperparameter

Using $N = 5$ and $L_{MAX} = 5$ in Eq. 5 we obtained $NV = 160$. As described in Section 3, the λ hyperparameter regulates the pruning process, balancing the weighted entropy, which measures the effectiveness of the clustering and is in the $[0, 1]$ range, with $NV(Cl(\mathcal{U}))$, which was in the $[0, 160]$ range. We selected $\lambda = 0.001$ for our comparisons so that the obtained trees would be highly pruned and so could be easily represented. Nonetheless, this process is inexpensive enough to be performed rapidly with hundreds of different λ values. Fig. 4 shows a plot of λ vs. the quality of the explanation for dataset *Outbrain_DAY1*. It can be seen that as λ grows, the quality of the full tree decreases linearly, since the importance of the second component in Eq. 5 becomes larger. When λ is large enough, the pruning process can get rid of nodes that do not decrease the weighted entropy sufficiently to offset the quality penalty associated with having more nodes. Consequently, the number of groups in the clustering decreased, while the quality improved with respect to that of the full tree. Similar results were obtained for datasets *MoviesEx*, *UsersEx* and *UsersExWithPrediction* (see the supplementary material). The λ hyperparameter allows the supervisor to control the aggressiveness of the pruning process and, therefore, the complexity and accuracy of the explanatory model.

6.2. Suitability of the method

The results listed in Table 2 show that the models obtained with our method were both more accurate and more explainable than those obtained using the alternative method. Only for dataset *UsersExWithPrediction* did the $k = 32$ Clustering+Explanation build a model with smaller entropy (0.027) than our method (0.047), although it needed more groups and so resulted in considerably inferior quality (-0.187) than our method (-0.091).Note that our method yielded more

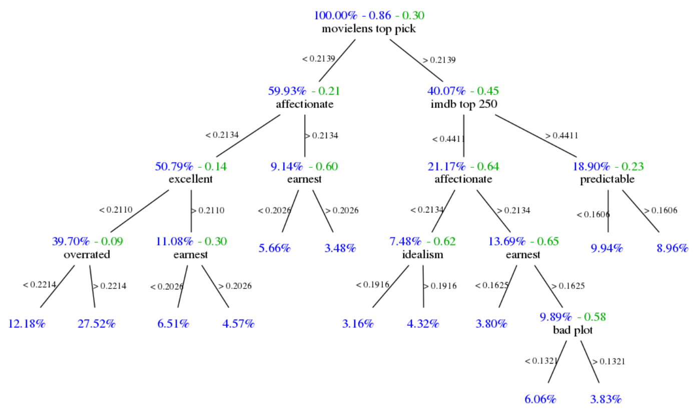
Table 2: Results comparison for the experimental datasets. Best results are highlighted in bold face. $\lambda = 0.001$ was used for the quality measurements.

Dataset	ExplainTree		Clustering+Explanation	
	WE	Quality	WE	Quality
Outbrain_DAY1	**0.244**	**-0.331**	0.359	-0.519
MoviesEx	**0.260**	**-0.353**	0.316	-0.476
UsersEx	**0.290**	**-0.333**	0.301	-0.461
UsersExWithPrediction	0.047	**-0.091**	**0.027**	-0.187

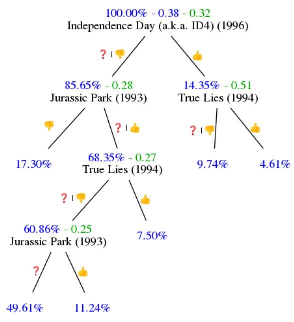
homogeneous groups in most cases than the alternative method, and this advantage was further enhanced when the explainability of the model was taken into account. This highlights the superiority of our approach that couples tree construction with a clustering process over the approach that uses independent algorithms for each step.

6.3. Analysis of the explanations

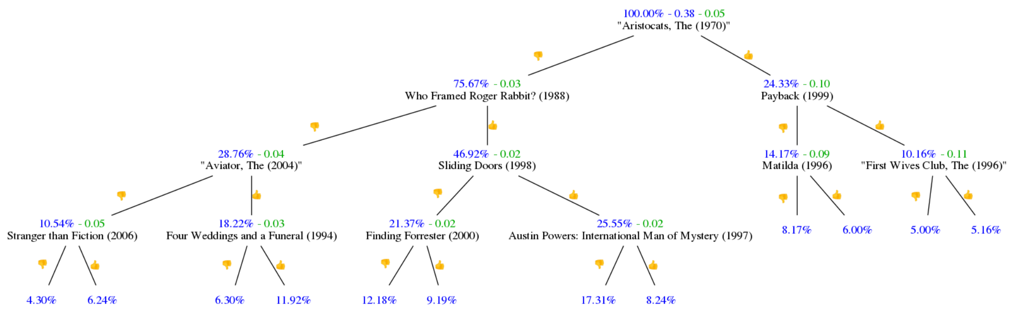
The information that a supervisor can extract regarding the characteristics defining the behaviour of an Outbrain user reading a given document is limited by the fact that the variables that characterize the user-document pair (topic, tags, etc.) are anonymized and only referred to by identifiers. Without the variable names no conclusions can be extracted from the explanatory. In contrast, however, the *MovieLens* dataset contains known variables that help identify trends in the data. Fig. 5 shows the explanatory tree for dataset *MoviesEx* and indicates which characteristics of a movie best define how different user types will react to them. It is apparent that a movie in a top list (variables *"movielens top pick"* and *"imdb top 250"*) was the most defining factor, after which certain movie qualities (variables *"affectionate"*, *"earnest"* or *"predictable"*) determine different user group responses. This information would give a supervisor insight into, for instance, what sort of movies should be added/removed from a catalogue. Another relevant piece of information in Fig.5 is the fact that the initial weighted entropy of the dataset (0.86) decreases greatly (to 0.3) with this clustering, indicating both that the response of users to different movies was very diverse and that the input variables allowed the uncertainty of the user response to a given movie to be decreased; this reflects the high value of the information extracted. The explanatory model for the *UsersEx* dataset represented in Fig. 5 offers additional insights to the same data. The first piece of information that stands out is that the weighted entropy of the full dataset is not very large (0.38), which indicates that users are somewhat homogeneous in their behaviour towards movies. Moreover, clustering does not manage to significantly decrease the weighted entropy of the data and, in consequence, the pruning process was very aggressive, yielding only 6 nodes. This was because, as stated in Section 5.1, the users are characterized by their rating $(-1, 0, 1)$ of the 1,005 most popular movies. However, for a given user, most movies are not rated, so users are defined by very sparse vectors. The large number of coincidences between users (most movies are unrated for a large set of users) made the task of the decision tree a difficult one. Nonetheless, this information could still be used, for instance, to rapidly determine the user type of a new user in a cold-start situation by simply asking them to rate a few movies selected from this decision tree. Furthermore, more information could be extracted from this approach by using the learned utility function to eliminate undetermined values in the characterization of users, as described in Section 5.1. Using the predicted values, the clustering tree



(a) *MoviesEx*



(b) *UsersEx*



(c) *UsersExWithPrediction*

Figure 5: Explanatory trees for the *MovieLens* datasets after pruning with $\lambda = 0.001$. Each node shows, respectively, the proportion of elements that it represents w.r.t. the full dataset (shown in blue), the weighted entropy at that node (shown in green) and the variable used in the next split (black). Split values are shown next to each split line. The root node also indicates (in blue) the weighted entropy of the whole dataset.

corresponding to dataset *UsersExWithPrediction*, represented in Fig. 5, was much more effective and decreased the weighted entropy to 0.05. Such a decision tree could be useful in a cold-start scenario, similar to that described above but in which one could expect an unambiguous rating of each presented item, as could be the case when items can be rated on the spot.

6.4. Scalability of the method

We performed an experiment with the aim of measuring the scalability of the method and of the Apache Spark distributed implementation. The same computation was performed for varying numbers of computing nodes. The experiments were run in a computer cluster formed by 8 machines with 12 computing cores each. The technical specifications for each node are provided in Table 3. The Spark version used was 2.4.0, on Hadoop 3.0.0-cdh6.1.0. The operating system of the machines was CentOS Linux release 7.4.1708.

The times invested to compute a three-level clustering tree for dataset *Outbrain_DAY1*, listed in Table 4, indicate that the Spark implementation takes advantage of the fact that most calculations are mutually independent and can, therefore, be performed in parallel. Consequently, the

Table 3: Computer cluster overview.

8 nodes with the following characteristics:	
Processor:	2 × Intel Xeon E5-2620 v3 at 2.40Ghz
Cores:	6 per processor (12 per node)
Threads:	2 per core (24 total per node)
Storage:	12 × 2TB NL SATA 6Gbps 3.5” G2HS
RAM:	64 GB
Network:	1x10Gbps + 2x1Gbps

Table 4: Execution time for computing a three-level tree for varying numbers of computing nodes.

Time (H:M:S)			
# cores	12	2x12	4x12
Outbrain_DAY1	3:19:46	2:35:33	1:33:45

execution time decreased at a similar rate to the increase in the number of computing nodes, which would be the ideal. The implementation allows the processing of large amounts of data in a reasonable time, provided the user supplies enough computing resources for the calculation.

7. Conclusions

We describe a method to obtain a global explanation of the information encoded in a dyadic dataset. The computed model consists of a single decision tree that partitions one of the entities into groups with homogeneous behaviour; this decision tree is computed using an adaptation of a measure documented in the literature. The presented method is formulated in such a way that it can be applied to any dyadic dataset in any field that processes dyadic data. For instance, for data representing interactions in a market it could be used to perform large-scale market segmentation that provides supervisors with valuable insights for informed decision making. It could also be used to identify global trends in the data corresponding to recommender systems, topic modelling, social network analysis and other similar data problems.

Also described is an an implementation of the presented algorithm in the popular Apache Spark distributed computing framework, which allows the processing of large volumes of data. Our experiments point to both the validity and the scalability of the approach, while also demonstrating various approaches to analysing diverse dyadic data. A brief analysis of how the retrieved information can be used is also presented.

In the future we plan to adapt this algorithm so that it becomes incremental and so allows the use of a previously trained model to accelerate the calculation of an updated version when new data becomes available. We also plan to revise the search strategy used to build the tree so that the algorithm can be interrupted at any time and still yield meaningful results.

Acknowledgements

This research has been supported in part by the Spanish Ministerio de Economía y Competitividad (projects TIN 2015-65069-C2-1-R and 2-R), partially funded by FEDER funds of the EU and by the Xunta de Galicia (projects ED431C 2018/34 and Centro Singular de Investigación de Galicia, accreditation 2016-2019). The authors wish to thank the Fundación Pública Galega Centro Tecnolóxico de Supercomputación de Galicia (CESGA) for the use of their computing resources. We also thank Jorge Díez, Pablo Pérez and Oscar Luaces for providing the data needed to construct the *(Outbrain_DAY1)* dataset and for their assistance in an early stage of this project.

References

[1] T. Hofmann, J. Puzicha, and M. I Jordan. Learning from dyadic data. In *Advances in Neural Information Processing Systems*, pages 466–472, 1999.

[2] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[3] P. Kotler and K. K. Cox. *Marketing management and strategy*. Prentice Hall, 1980.

[4] O. Luaces, J. Díez, A. Alonso-Betanzos, A. Troncoso, and A. Bahamonde. A factorization approach to evaluate open-response assignments in MOOCs using preference learning on peer assessments. *Knowledge-Based Systems*, 85:322–328, 2015.

[5] O. Luaces, J. Díez, A. Alonso-Betanzos, A. Troncoso, and A. Bahamonde. Content-based methods in peer assessment of open-response questions to grade students as authors and as graders. *Knowledge-Based Systems*, 117:79–87, 2017.

[6] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89. IEEE, 2018.

[7] Z. C. Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[8] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[9] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)*, 51(5):93, 2018.

[10] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

[11] M. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30, 1996.

[12] R. Krishnan, G. Sivakumar, and P. Bhattacharya. Extracting decision trees from trained neural networks. *Pattern recognition*, 32(12), 1999.

[13] O. Boz. Extracting decision trees from trained neural networks. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 456–461. ACM, 2002.

[14] J. Basak and R. Krishnapuram. Interpretable hierarchical clustering by constructing an unsupervised decision tree. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):121–132, 2005.

[15] U. Johansson and L. Niklasson. Evolving decision trees using oracle guides. In *2009 IEEE Symposium on Computational Intelligence and Data Mining*, pages 238–244. IEEE, 2009.

[16] P. Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[17] J. Liu, X. Liao, W. Huang, and X. Liao. Market segmentation: A multiple criteria approach combining preference analysis and segmentation decision. *Omega*, 2018.

[18] M. Y. Kiang, M. Y. Hu, and D. M. Fisher. An extended self-organizing map network for market segmentation—a telecommunication example. *Decision Support Systems*, 42(1):36–47, 2006.

[19] P. Hanafizadeh and M. Mirzazadeh. Visualizing market segmentation using self-organizing maps and fuzzy delphi method–adsl market of a telecommunication company. *Expert Systems with Applications*, 38(1):198–205, 2011.

[20] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[21] O. Cordon F. Marcelloni A. Fernandez, M.J. del Jesus and F. Herrera. Evolutionary fuzzy systems for explainable artificial intelligence: Why, when, what for, and where to? *IEEE Computational Intelligence Magazine*, 14(1):69–81, 2019.

[22] H. Shan and A. Banerjee. Bayesian co-clustering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 530–539. IEEE, 2008.

[23] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250. ACM, 2000.

[24] P. Kouki, J. Schaffer, J. Pujara, J. O'Donovan, and L. Getoor. Personalized explanations for hybrid recommender systems. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pages 379–390. ACM, 2019.

[25] Y. Zhao, S. Liang, Z. Ren, J. Ma, E. Yilmaz, and M. de Rijke. Explainable user clustering in short text streams. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 155–164. ACM, 2016.

[26] X. Wang, X. He, F. Feng, L. Nie, and T. Chua. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1543–1552. International World Wide Web Conferences Steering Committee, 2018.

[27] C. Liu, H. Yang, J. Fan, L. He, and Y. Wang. Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. In *Proceedings of the 19th international conference on World wide web*, pages 681–690. ACM, 2010.

[28] J. Díez, P. Pérez, O. Luaces, and A. Bahamonde. Readers segmentation according to their preferences to click promoted links in digital publications. Technical report, Universidad de Oviedo, 2018.

[29] H Akaike. Information theory and an extension of the maximum likelihood principle. In *2nd International Symposium on Information Theory*, pages 267–281. Akademiai Kiado, 1973.

[30] G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[31] J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[32] A. Clare and R. D. King. Knowledge discovery in multi-label phenotype data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 42–53. Springer, 2001.

[33] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS)*, 5(4):19, 2016.