Al-serafi, A. [et al.]. Keeping the data lake in form: DS-kNN datasets categorization using proximity mining. A: International Conference on Model and Data Engineering. "Model and Data Engineering, 9th International Conference, MEDI 2019: Toulouse, France, October 28-31, 2019: proceedings". Berlín: Springer, 2019, p. 35-49. The final authenticated version is available online at https://doi.org/10.1007/978-3-030-32065-2_3

Keeping the Data Lake in Form: DS-kNN Datasets Categorization Using Proximity Mining

Ayman Alserafi[1,2], Alberto Abelló[1], Oscar Romero[1], and Toon Calders[3]

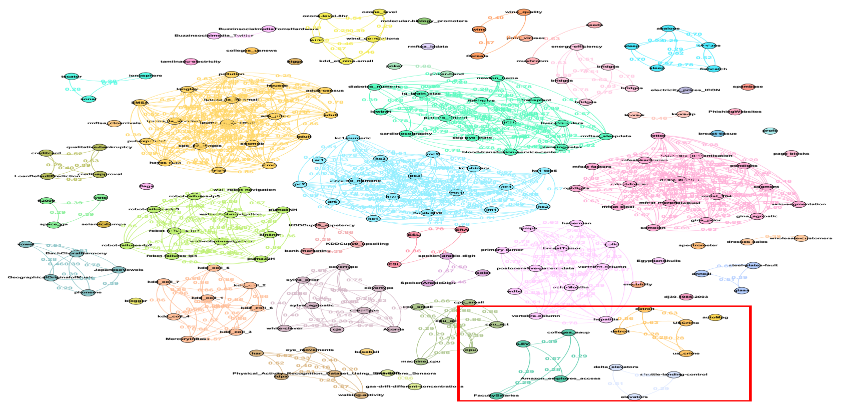
[1] Universitat Politècnica de Catalunya - BarcelonaTech, Barcelona, Catalunya, Spain
{alserafi,aabello,oromero}@essi.upc.edu
[2] Université Libre de Bruxelles (ULB), Brussels, Belgium
[3] Universiteit Antwerpen (UAntwerp), Antwerp, Belgium
toon.calders@uantwerp.be

Abstract. With the growth of the number of datasets stored in data repositories, there has been a trend of using Data Lakes (DLs) to store such data. DLs store datasets in their raw formats without any transformations or preprocessing, with accessibility available using schema-on-read. This makes it difficult for analysts to find datasets that can be crossed and that belong to the same topic. To support them in this DL governance challenge, we propose in this paper an algorithm for categorizing datasets in the DL into pre-defined topic-wise categories of interest. We utilise a k-NN approach for this task which uses a proximity score for computing similarities of datasets based on metadata. We test our algorithm on a real-life DL with a known ground-truth categorization. Our approach is successful in detecting the correct categories for datasets and outliers with a precision of more than 90% and recall rates exceeding 75% in specific settings.

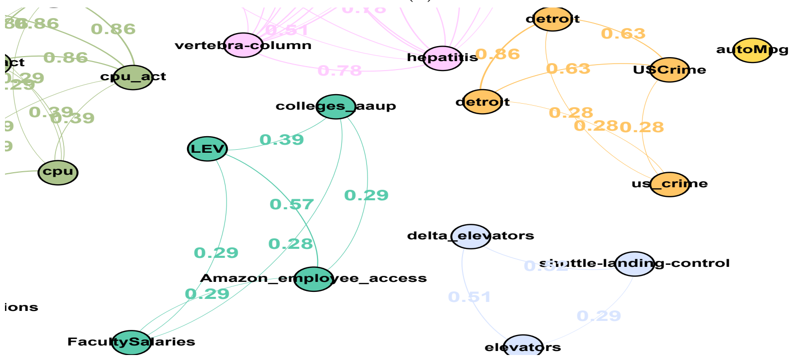
Keywords: Data Lake Categorization · k-Nearest-Neighbour · Metadata Management · Proximity Mining.

1 Introduction

Today, a lot of data is generated covering different heterogeneous topics and domains. Those data are frequently stored as tabular datasets which describe different entities (in the rows) with information about them stored as attributes (in the columns). A collection of such raw datasets which are stored in their original schema without preprocessing or transformations is called a Data Lake (DL) [3, 16]. Over its lifetime, a DL becomes very diverse and can cover different topics, making it difficult to find and retrieve relevant datasets for analysis. Therefore, it is a challenge for the users to govern the DL by detecting the groupings and underlying structures of similar datasets covering relevant topics for analytics [3, 4, 11]. To tackle this challenge, we propose an automated approach called DS-kNN to detect such groupings using k-nearest-neighbour (k-NN). The approach relies on collecting relevant metadata about the datasets when they are ingested, then we compute proximity models of dataset similarities based on supervised machine learning, and apply those models on new datasets to compute their similarity scores with datasets stored in the DL. Once we computed the similarities, we apply a k-NN algorithm to categorize the ingested datasets into the groupings already present in the DL or to classify them as outliers. An example of the expected results can be seen in Fig. 1. Here, we visualise the DL as a *proximity graph* having datasets as nodes and edges connecting the nodes showing the similarity scores ($\mathbb{R} \in [0, 1]$) computed using the proximity model. Datasets in the same category (cluster) are shown in the same colour. We only show edges between datasets in the same category. In Fig. 1 (a) we show an example of a complete DL proximity graph and in (b) we zoom-in on the specific part highlighted with a box for showing more details.



(a)



(b)

Fig. 1: A visualisation of the output from DS-kNN data lake (DL) categorization. A proximity graph shows the datasets as nodes and the proximity scores as edges between nodes. Fig.(a) complete DL and Fig. (b) a zoomed-in view highlighted by the red box in (a)

The main contributions of this paper are: (1) We propose a kNN-based proximity mining algorithm for finding the correct categories for datasets based on existing categories in the DL, (2) we evaluate the algorithm in a real-world setting to prove its effectiveness in assigning correct categories to new datasets ingested in the DL, (3) we experimentally test the effect of different DL settings on the performance of our approach.

In the rest of this paper, we define the DL and the scenario we consider in Section 2, we present the DS-kNN algorithm in Section 3, then we test the algorithm on a real-life DL and we experiment with our algorithm in Section 4, we present related work in Section 5, and we conclude in Section 6.

2 Preliminaries

We consider a DL consisting of tabular datasets. Those are large heterogeneous repositories of *flat structured data* (i.e., CSV, web tables, spreadsheets, etc.). Such datasets are structured as groups of *instances* describing real-world entities, where each instance is expressed as a set of *attributes* describing the properties of the entity. We formally define a dataset D as a set of instances $D = \{I_1, I_2, ...I_n\}$. The dataset has a set of attributes $S = \{A_1, A_2, ...A_m\}$, where each attribute A_i has a fixed type, and every instance has a value of the right type for each attribute. We focus on two types of attributes: continuous ***numeric attributes*** with real numbers and categorical ***nominal attributes*** with discrete values.

For each dataset, we collect different statistics about their content which we call *content meta-features*:

- **Nominal attributes:** their data profile mainly involves frequency distributions of their distinct values.
- **Numeric attributes:** their data profile mainly involves aggregated statistics like mean, min, max, and standard deviations.

We compute similarity scores between pairs of datasets $[D_a, D_b]$, as follows:

- $Sim(D_a, D_b)$: an estimation ($\mathbb{R} \in [0, 1]$) of the similarity based on the comparison of the *content meta-features* we collect about the datasets and their attributes. Typically, the information contained in highly similar datasets would overlap. An example would be a pair of datasets having similar numeric values and distribution of values, or nominal attributes having the same number of values. Alternatively, it could be based on *name string-similarity* between datasets and their attributes.

Scenario. We aim at governing the DL by incrementally maintaining the clusters of datasets defined for them. We consider the scenario where we initially have an existing DL for which we know all clusters of datasets based on their categories. However, given the dynamic nature of DLs, new datasets are frequently ingested. Thus, we need to compare these new datasets against the datasets already in the DL to find their similarity with them, and then to find their most appropriate

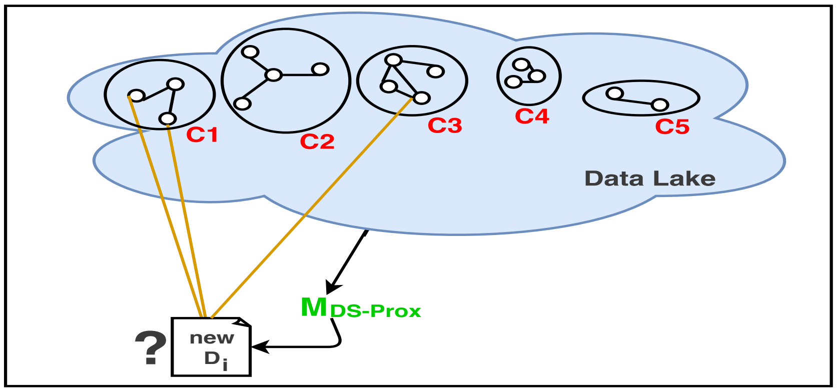


Fig. 2: The data lake categorization scenario using k-NN proximity mining

category based on the similar datasets found in the DL, or to assign them to a separate category as an outlier.

This shapes the main problem for this research paper: given a collection of datasets in a DL and a newly ingested dataset, find all pairs of highly similar datasets, and based on their categories, assign a new category for the new dataset, or if no highly similar datasets are found then indicate that the dataset is an outlier. To compute the similarity between the datasets, we use a proximity model, which we call $M_{DS-Prox}$. We discuss how we create this model in Subsection 2.1.

The scenario discussed is visualized in Fig. 2. Consider that there is a DL having a group of datasets (white circles) which have annotations of all their $Sim(D_a, D_b)$ relationships between pairs (as seen by the lines linking the datasets). Groups of datasets with linkages are segmented into categories (seen by the encompassing black circles). Those categories are the groupings of the subject-areas or domains-of-knowledge we have in the DL. We need to automatically use this DL and its known annotations to create a model $M_{DS-Prox}$ which can automatically annotate relationships of a new dataset D_i with the $Sim(D_i, D_b)$ similarity scores. Therefore, we need to learn a model from the DL and apply it to estimate the similarity between a new dataset and all other datasets already in the DL, in order to find the top-k neighbours.

Based on the similarity scores we assign a category to D_i. The highlighted edges between the new dataset D_i and some nodes in the DL are those having the highest similarity scores computed by the model (in this case, we give an arbitrary example where we use top-3 nearest-neighbours). In our proposed approach, each of those top neighbours proposes its category as the correct one for D_i, and the most proposed category should be assigned, or if no such similar datasets are found then D_i is marked as an outlier without any relevant category found. In the case of tied categories among the proposed ones from the top-k similar datasets, then all of them are assigned to D_i. In the example in Fig. 2, category 'C1' would be assigned as the final category as it has 2 *votes*, compared to only 1 *vote* by category 'C3'.

To learn the $M_{DS-Prox}$ model we use supervised machine learning as described in Subsection 2.1.

2.1 Proximity Mining: Meta-features metrics and models

For all the datasets in the DL, we collect two metadata types: **A.Content-based** and **B. Name-based** meta-features. The name-based techniques are the most commonly used metadata in previous research [7, 11, 13, 14]. In our DS-kNN approach, we propose content-based meta-features as an alternative to name-based metadata when computing similarity scores. Such content meta-features include data profiling statistics about the content of the datasets. Thus, we use two types of metadata for similarity computations:

- *Name-based metadata:* the naming of datasets and their attributes.
- *Content-based metadata:* profiling statistics about the data stored in the datasets. The collected meta-features (described in Table 1) include statistics concerning all attributes collectively, the attribute types found and the overall number of instances. Those form a concise list of meta-features that have been proved in our previous work [4] to be effective in predicting related datasets with similar schemata and stored information. Our purpose for those meta-features is to describe the general structure and content of the datasets for an approximate comparison using our proximity mining classification models.

To compute similarity scores $Sim(D_a, D_b)$ from name-based metadata, we use the Levenshtein distance as a standard string comparison metric [12]. The output from this comparison is considered as the similarity score from $M_{DS-Prox}$. For content meta-features, we construct the $M_{DS-Prox}$ model using the proximity mining approach from our previous work [4]. First, we compute distances for each meta-feature m_i from Table 1 between each pair of datasets $[D_a, D_b]$ using equation 1 which gives the relative difference as a number between 0 and 1. We compute this for all dataset pairs $[D_a, D_b]$ in the training sample.

$$dist_{m_i}(D_a, D_b) = \frac{\max\{m_i(D_a), m_i(D_b)\} - \min\{m_i(D_a), m_i(D_b)\}}{\max\{m_i(D_a), m_i(D_b)\}} \tag{1}$$

Once we have the metadata collected and their distances computed, we feed them to a supervised machine learning algorithm to produce a classification model which identifies those dataset pairs in the same assigned category. This creates the proximity mining model to compute similarity scores. For this initial training sample of datasets we have in the DL, a data analyst should have incrementally assigned a category cluster to each dataset based on their topics. The target variable for those classifiers is a binary value whether the datasets in the pair belong to the same category or not.

We use the two top performing ensemble learning algorithms from [4] to learn the model, which are the boosting machine learning algorithms AdaBoost [15]

Table 1: DS-Prox meta-features

Type	Meta-feature	Description
General	Number of Instances	The number of instances in the dataset
	Number of Attributes	The number of attributes in the dataset
	Dimensionality	The ratio of number of attributes to number of instances
Attributes by Type	Number per Type	The number of attributes per type (Nominal or Numerical)
	Percentage per Type	The percentage of attributes per type (Nominal or Numerical)
Nominal Attributes	Average Number of Values	The average number of distinct values per nominal attribute
	Standard Deviation of Number of Values	The standard deviation in the number of distinct values per nominal attribute
	Minimum/Maximum Number of Values	The minimum and maximum number of distinct values per nominal attribute
Numeric Attributes	Average Numeric Mean	The average of the means of all numeric attributes
	Standard Deviation of the Numeric Mean	The standard deviation of the means of the numeric attributes
	Minimum/Maximum Numeric Mean	The minimum and maximum mean of numeric attributes
Missing Values	Missing Attribute Count	The number of attributes with missing values
	Missing Attribute Percentage	The percentage of attributes with missing values
	Minimum/Maximum Number of Missing Values	The minimum and maximum number of instances with missing values per attribute
	Minimum/Maximum Missing Values Percentage	The minimum and maximum percentage of instances with missing values per attribute
	Mean Number of Missing Values	The mean number of missing values from each attribute
	Mean Percentage of Missing Values	The mean percentage of missing values from each attribute

and LogitBoost [6]. Those algorithms were compared in our previous work to other algorithms and were found to be the best in finding related schemata. The positive-class distribution produced by the ensemble model is used as the similarity score $Sim(D_a, D_b)$ [15]. Finally, we apply the learnt $M_{DS-Prox}$ model on pairs of one new ingested dataset and each existing datasets in the DL to generate the similarity scores. We compare the score against a minimum threshold like in Equation 2. Only pairs passing the threshold are considered as candidate top-k nearest neighbours to a dataset in our DS-kNN algorithm. We discuss this in detail in Section 3. Different similarity thresholds lead to a different performance of the algorithm, so we test multiple threshold values in our experiments to discover the best one to use.

$$Top(D_a, D_b) = \begin{cases} 1, & Sim(D_a, D_b) > c_{rel} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

3 DS-kNN: A proximity mining based k-nearest-neighbour algorithm for categorizing datasets

Algorithm 1: DS-kNN Categorization of a dataset ingested in a Data Lake

```
Input: A new ingested dataset $D_a$, each existing dataset $D_b$ in the data lake
       $DL$, Dataset-level meta-features distance metrics $MF$ for each pair of
       datasets $\{D_a, D_b\}$, the category $Cat_{D_b}$ for each existing dataset in the
       $DL$, the classification model $M_{ds-prox}$, algorithmic parameters: the
       number $k$ of nearest neighbours, and the similarity score threshold $c_{rel}$
Output: The set $SP$ of the ingested dataset and its similarity scores
        $Sim(D_a, D_b)$ and category $Cat_{D_b}$ for each pair $\{D_a, D_b\}$ passing $c_{rel}$,
        the set $SP\text{-}Top$ of top matching $k$ datasets and their categories, the
        assigned category for the new dataset $Cat_{D_a}$
$SP \leftarrow \emptyset$;
$SP\text{-}Top \leftarrow \emptyset$;
foreach $\{D_a, D_b\} \subset DL$ and $a \neq b$ do
    $[D_a, D_b, Sim(D_a, D_b)] = M_{ds-prox}(MF_{\{D_a,D_b\}})$;
    if $Sim(D_a, D_b) > c_{rel}$ then
        $SP \leftarrow SP \cup \{[D_a, D_b, Sim(D_a, D_b), Cat_{D_b}]\}$;
    end
end
$SP\text{-}Top$ = Top-k_Nearest_Neighbours($SP$, k); \\Retrieve the subset of the
  highest ranking k-pairs by similarity score
$Cat_{D_a} = Top\text{-}category(SP\text{-}Top)$; \\Get category with majority vote from Top-k
if $(Cat_{D_a} = NULL)$ then
    $Cat_{D_a} =' Outlier'$;
end
```

We propose an algorithm for computing the categories of an ingested dataset as described in the scenario in Section 2. After learning the classification model $M_{DS-Prox}$, we apply the classifier to each new pair $[D_a, D_b]$ where D_a is any new ingested dataset and D_b is each of the existing datasets in the DL, in order to achieve the similarity score $Sim(D_a, D_b)$ with all datasets in the DL. Then, we apply k-NN in our proposed DS-kNN Algorithm 1 to compute the category for a new dataset. k-NN was also successful in similar categorization problems, like in free-text document categorization [8].

First, our algorithm applies the $M_{DS-Prox}$ model on all the dataset pairs for the new dataset D_a to compute their similarity scores, and those passing the minimum threshold are stored in the set SP. To improve efficiency, a heap data structure could be used to store the datasets with their similarity scores for quick search and retrieval of top-k nearest-neighbours. The next step involves finding those top-k nearest-neighbours which are existing datasets in the DL with the highest similarity scores to D_a. Finally, we assign the category with the most number of pairs in the top-k nearest neighbours as the assigned category based on simple majority voting by top-k nearest neighbours. If no top-k nearest neighbours are found then the dataset is marked as an 'outlier' with no proposed category.

The algorithm has the following parameters as input:

- **The number of neighbours (k):** the top-k number of nearest neighbours which our algorithm uses to predict the new category for an ingested dataset.
- **The proximity model ($M_{ds-prox}$):** this is the proximity mining model created using our approach described in Section 2. We use different models depending on the metadata, i.e. content-based, dataset-name based or attribute-name based.
- **The similarity threshold (c_{rel}):** the minimum allowed similarity score to consider a dataset pair as candidate nearest neighbour.

4 Experimental Evaluation

We test our proposed categorization algorithm on a real-life DL. We describe the dataset used, the experimental setup and our results with a detailed discussion of the performance of DS-kNN.

4.1 Dataset: OpenML DL ground-truth

We created a ground-truth based on manual annotations of 203 datasets from a real-life DL called OpenML[4]. It consists of different datasets covering heterogeneous topics, each having a name and a description. The categories found in the ground-truth are visualised in Fig. 1.

The sample of datasets collected from OpenML is scraped to extract datasets having a description of more than 500 characters. The descriptions helped the manual annotators deciding on the assigned topic for each dataset. Out of the 514 datasets retrieved, we selected 203 with meaningful descriptions (i.e., excluding datasets whose descriptions do not allow to interpret its content and to assign a topic). A domain expert and one of the authors collaborated to manually label the datasets with their topic. The datasets were labelled by both their *broad* **subject** (e.g., '*social demographics*') and their more *specific* **entity** they describe (e.g., '*citizens census data*'). The interested reader can download the two annotated datasets from GitHub[5].

Table 2 shows the number of datasets per category assigned based on topic grouping type. We only show the top 10 categories found by size for each grouping. The total number of categories is also given and the number of categories bigger than a specific size (i.e., with at least this number of members), and the number of outliers (datasets with their own specific category without any other members). As can be seen in the table, the datasets in the DL we use in the experiments cover heterogeneous topics and different category sizes.

[4] http://www.openml.org
[5] https://github.com/AymanUPC/ds-knn

Table 2: A description of the 203 OpenML categorized datasets collected. Datasets are categorized by subject and by entity.

Category Type	**No. of Categories**	**Categories by Type**	**Categories by Size**	**Outliers**
Subject	53	Computer Software (17), Social Demographics (17), Image Recognition (16), Health (14), Robot (11), Disease (11), Natural Water (8), Ecology (8), Computer Hardware (6), Motion Sensing (5)	8+ members (8), 5+ members (14), 3+ members (25)	21
Entity	77	Computer Software defects (16), Citizens Census Data (12), Digit Handwriting Recognition (12), Diseases (11), Robot Motion Measurements (11), Health Measurements (10), Chemical Contamination (8), Plantation Measurements (8), CPU Performance Data (6), Animal Profile (5)	8+ members (8), 5+ members (12), 3+ members (21)	47

4.2 Experimental Setup

Our goal is to test the performance of the DS-kNN algorithm in correctly assigning the right category to datasets. We compare the performance of the DS-Prox content-based models when applied in DS-kNN against the baseline models of dataset-names and attribute-names, which are the commonly used metadata in previous work [7, 11, 13, 14] (see Section 5). We implement DS-kNN based on those different models in Java using a Postgres SQL database as its backend for storing the metadata, and we feed it with the datasets from the OpenML DL. We initially tested the algorithm on a random sample of datasets using different values for $k \in \{3, 5, 7, 9, 11, 13\}$ and found the best performing value under the same settings was $k = 3$, so we conduct all our trials in the experiments using $k = 3$. To test the generalizability and adaptability of DS-kNN under different DL settings, we also conduct trials with the algorithm under the following different settings which affect the ground-truth used in the training and testing of $M_{ds-prox}$:

- **Different category sizes:** we test DS-kNN with all the datasets (including outliers where category size is just 1 dataset) and with categories that at least contain the following number of members (3,5,8). We test different sizes of categories to check if the algorithm is affected by category sizes.
- **Different ground-truth types:** We test the algorithm with the broad (1) subject-based categories and the more detailed (2) entity-based categories.

For each DL setting, we compare the performance of the DS-kNN algorithm using the following input parameters:

- **Different models ($M_{ds-prox}$):** we test the different models generated by different metadata, which are (1) Dataset Name, (2) Attribute Name and (3) DS-Prox Content.

– **Different similarity thresholds:** we use different thresholds for $Sim(D_a, D_b)$ including $(0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)$

We test the different combinations of the above parameters and settings which resulted in the execution of a total of 240 independent trials. We utilise a leave-one-out experimental setup to test our categorization algorithm in each trial, as seen in Fig. 2, so for each experimental trial we train the model under the same settings and with the same parameters 203 times, where for each run we keep a single dataset out from the training of the model and treat it as the new test dataset. We train the proximity model with all the datasets in the DL except the test dataset, we apply the proximity model on the test dataset with all dataset pairs found in the DL, and we run our algorithm to compute its allocated category or to mark it as an outlier. We apply Algorithm 1 on the test dataset and we find the top categories it should be allocated to. The goal is to maximise the number of correctly assigned categories based on top-k nearest neighbours.

To evaluate the effectiveness, we consider our algorithm as an example of a multi-class classification problem. We evaluate whether each dataset gets assigned the correct category based on *top-k nearest neighbours*. We compute the number of correctly annotated categories and outliers by measuring recall, precision and F1-scores which are commonly used for evaluation in similar settings [1, 2, 11]. We compute the F1 score as the harmonic mean of the recall and the precision [12]. The evaluation metrics are described in Equations (3),(4) and (5) respectively. Here, TP means true-positives which are the datasets correctly classified to their category. FN are false negatives, and FP are false positives. We compute the evaluation metrics per category and average the final scores from all categories to achieve macro-averaging scores [12]. For example, consider we have in the ground-truth two categories $C1$ and $C2$ consisting of 10 datasets each. $C1$ had 9 TPs and 1 FP (i.e. a dataset from a different category incorrectly assigned to it by DS-kNN) while $C2$ had 8 TPs and 2 FPs, therefore they will have a precision of 0.9 and 0.8 respectively. Therefore, the macro-precision will be $\frac{0.8+0.9}{2} = 0.85$.

$$recall = \frac{TP}{TP + FN} \tag{3}$$

$$precision = \frac{TP}{TP + FP} \tag{4}$$

$$F1\text{–}score = 2 \times \frac{(Recall \times Precision)}{Recall + Precision} \tag{5}$$

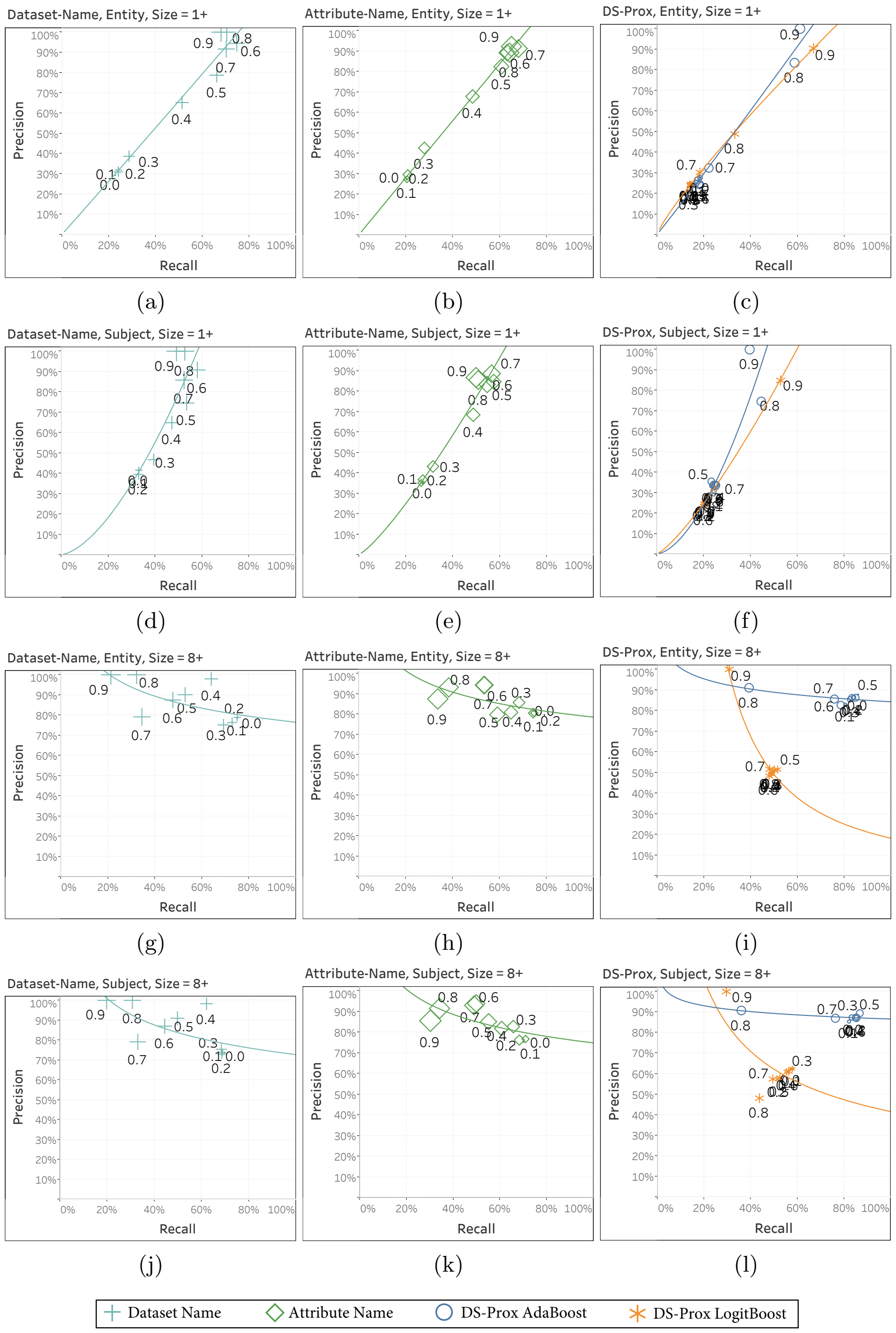


Fig. 3: Performance of DS-kNN using different models, different ground-truths, and different category sizes

4.3 Results

We present the precision-recall curves from our experiments in Fig. 3. Each graph plots the macro-averaging performance resulting from leave-one-out cross-validation of a specific model for a specific ground-truth type and category sizes (which are labelled above the chart). For all our results we use percentages for the performance metrics. Here, we plot recall against precision for each of the different model types used in DS-kNN and the different minimum category sizes we use in the experiment. The numbers annotated on the points indicate the similarity threshold (also indicated by the size of the points, where bigger size indicates a higher similarity threshold). We show the results for the non-restricted (category size = 1+) which includes outliers and the biggest category sizes (category size = 8+). Each model type has a different symbol and colour. For DS-Prox, circles indicate AdaBoost-based model and stars indicate LogitBoost-based model. We also present in Table 3 the evaluation metrics for the top performing parameters for DS-kNN (in terms of F1-scores) for each model type, category sizes, and ground-truth type.

Table 3: The evaluation of DS-kNN for $k = 3$ and different model types, ground-truth types and minimum category sizes. For each setting, we only show here the best performing similarity threshold based on F1-scores.

Model Type	**Ground Truth Type**	**Min. Category Size**	**Similarity Threshold**	**Recall**	**Precision**	**F1-score**
Attribute Name	Entity	1	0.7	67.9	91.3	77.9
Attribute Name	Entity	3	0.3	55.8	75.3	64.1
Attribute Name	Entity	5	0.3	57.5	78.3	66.3
Attribute Name	Entity	8	0	74.6	81.5	77.9
Attribute Name	Subject	1	0.7	56.5	88.6	69
Attribute Name	Subject	3	0.4	47.2	67	55.4
Attribute Name	Subject	5	0.3	55.1	71.4	62.3
Attribute Name	Subject	8	0.1	70.9	76.7	73.7
Dataset Name	Entity	1	0.6	74.7	94.4	83.4
Dataset Name	Entity	3	0.4	49.9	90.1	64.4
Dataset Name	Entity	5	0.4	59.6	98.5	74.3
Dataset Name	Entity	8	0.4	64	98	77.4
Dataset Name	Subject	1	0.6	58	90.8	70.7
Dataset Name	Subject	3	0.4	47.9	74.5	58.3
Dataset Name	Subject	5	0.4	55.7	98.9	71.3
Dataset Name	Subject	8	0.4	62.1	98.3	76.1
DS-Prox Content	Entity	1	0.9	66.8	90.5	76.8
DS-Prox Content	Entity	3	0.2	53.9	61.2	57.4
DS-Prox Content	Entity	5	0.3	68.4	81.9	74.6
DS-Prox Content	Entity	8	0	85.9	87.8	86.8
DS-Prox Content	Subject	1	0.9	52.8	84.8	65.1
DS-Prox Content	Subject	3	0.1	44.5	54.4	48.9
DS-Prox Content	Subject	5	0.2	58.7	70	63.9
DS-Prox Content	Subject	8	0.5	86.8	89.4	88.1

As could be seen from the results, DS-kNN performs comparatively well with the attribute-name and the DS-Prox content-based models for category size 1+, but for larger category sizes the DS-Prox content models are better in assigning the correct categories. For example, DS-Prox content leads to a precision of about 90% and recall higher than 80% for category sizes of at least 8 members and the entity-based ground-truth, while attribute-name model can only achieve 82% precision and 75% recall. Dataset-name based model performs worse in terms of recall with 64% but much better precision with 98%. The results also indicate that the choice of the similarity threshold can affect the performance of DS-kNN.

In general, DS-kNN performs better with bigger category sizes than smaller category sizes as it becomes easier for the algorithm to find relevant top-k nearest neighbours. However, it is still good in detecting outliers and other categories as seen for the performance for *'min. category size'* = 1, for example a recall of 75% and precision of 95% for dataset-name based model. The dataset-name model performs better in detecting outliers as seen from this result. The DS-kNN algorithm performed equally good with both ground-truth types under the same settings and with the same parameters, yet slightly better with the more specific entity-based ground-truth with small category sizes and outliers. This indicates the adaptability of DS-kNN to different DL settings and properties.

5 Related Work

Categorization of datasets from heterogeneous domains is an emerging research topic, and relevant previous research include [11], where they utilise the attribute names to cluster the datasets into categories using a probabilistic model. Datasets are assigned to different categories using different probabilities. They tackle the multi-label classification of datasets and retrieval of datasets from relevant domains by querying systems. Our approach improves this approach by using a machine-learning based approximate proximity mining technique instead of the Jaccard similarity of exact values. We also use content-based metadata for categorizing and not only name-based metadata. This is important for DLs where datasets are not well maintained with meaningful attribute names.

Clustering could also be applied to other types of semi-structured datasets like ontologies [1] and XML documents [2, 10], etc. In [1], they propose an algorithm to cluster instances from different ontologies based on their structural properties in the ontology graphs. Their goal is to facilitate ontology matching rather than domains discovery. Similarly, in [2, 10] they cluster the semi-structured documents based on their structure similarity and linguistic matchers.

Clustering free-text without any structure is also possible. For example, [5] aims to cluster short text messages by computing TF-IDF word similarity between free-text documents. Similarly, [8] categorizes free-text documents using a k-NN based algorithm by first extracting TF-IDF weighted labels and feeding them to the algorithm. Another specific application would be clustering streaming data where a sliding window algorithm could be used [9], where they also use k-NN when finding relevant clusters for a given data instance ingested in a stream of data points.

6 Conclusion

We proposed DS-kNN, a categorization algorithm for classifying datasets into pre-defined topic-wise groups. Our algorithm can be applied in a DL environment to support users in finding relevant datasets for analysis. Our algorithm uses extracted metadata from datasets to compute their similarities to other datasets in the DL using a proximity mining model and name strings comparisons. Those similarity scores are fed to DS-kNN to decide on the most relevant category for a dataset based on its top-k nearest neighbours. Our algorithm was effective in categorizing the datasets in a real-world DL and detecting outliers, yet our results can be improved to achieve better performance. In the future, we will test the same k-NN algorithm but using different proximity models based on finer granularity metadata extracted about the content of attributes in the datasets. We also seek to improve our algorithm with semantic analysis of values found in the attributes to complement the syntactical comparisons we compute in the proximity models.

References

1. Algergawy, A., Massmann, S., Rahm, E.: A Clustering-Based Approach for Large-Scale Ontology Matching. In: East European Conference on Advances in Databases and Information Systems (ADBIS), pp. 415–428. Springer (2011)
2. Algergawy, A., Schallehn, E., Saake, G.: A schema matching-based approach to XML schema clustering. In: Proceedings of the International Conference on Information Integration and Web-based Applications & Services. pp. 131–136. ACM (2008)
3. Alserafi, A., Abelló, A., Romero, O., Calders, T.: Towards Information Profiling: Data Lake Content Metadata Management. In: DINA Workshop, ICDM (2016)
4. Alserafi, A., Calders, T., Abelló, A., Romero, O.: DS-prox: Dataset proximity mining for governing the data lake. In: International Conference on Similarity Search and Applications SISAP. vol. 10609 LNCS, pp. 284–299. Springer (2017)
5. Baralis, E., Cerquitelli, T., Chiusano, S., Grimaudo, L., Xiao, X.: Analysis of twitter data using a multiple-level clustering strategy. In: International Conference on Model and Data Engineering. pp. 13–24. Springer (2013)
6. Friedman, J., Hastie, T., Tibshirani, R., et al.: Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics **28**(2), 337–407 (2000)
7. Gallinucci, E., Golfarelli, M., Rizzi, S.: Schema profiling of document-oriented databases. Information Systems **75**, 13–25 (2018)
8. Han, E.H.S., Karypis, G., Kumar, V.: Text categorization using weight adjusted k-nearest neighbor classification. In: Pacific-asia conference on knowledge discovery and data mining. pp. 53–65. Springer (2001)
9. Hentech, H., Gouider, M.S., Farhat, A.: Clustering heterogeneous data streams with uncertainty over sliding window. In: International Conference on Model and Data Engineering. pp. 162–175. Springer (2013)
10. Lee, M.L., Yang, L.H., Hsu, W., Yang, X.: Xclust: clustering XML schemas for effective integration. In: Proceedings of the international conference on Information and knowledge management. pp. 292–299. ACM (2002)

11. Mahmoud, H.A., Aboulnaga, A.: Schema clustering and retrieval for multi-domain pay-as-you-go data integration systems. In: Proceedings of the ACM SIGMOD International Conference on Management of data. pp. 411–422. ACM (2010)
12. Manning, C.D., Raghavan, P., Schütze, H.: An Introduction to Information Retrieval. No. c (2009)
13. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal **10**(4), 334–350 (2001)
14. Shvaiko, P.: A Survey of Schema-based Matching Approaches. Journal on Data Semantics **3730**, 146–171 (2005)
15. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to data mining. Pearson Education (2006)
16. Terrizzano, I., Schwarz, P., Roth, M., Colino, J.E.: Data Wrangling: The Challenging Journey from the Wild to the Lake. In: 7th Biennial Conference on Innovative Data Systems Research CIDR'15 (2015)