Identifying and Fusing Duplicate Features for Data Mining

Hortênsia C. Barcelos, Mariana Recamonde-Mendoza, Viviane P. Moreira

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

Abstract. *This work addresses the problem of identifying and fusing duplicate features in machine learning datasets. Our goal is to evaluate the hypothesis that fusing duplicate features can improve the predictive power of the data whilst reducing training time. We propose a simple method for duplicate detection and fusion based on a small set of features. An evaluation comparing the duplicate detection against a manually generated ground truth obtained F1 of 0.91. Then, the effects of fusion were measured on a mortality prediction test. The results were inferior to the ones obtained with the original dataset. Thus we concluded that the investigated hypothesis does not hold.*

1. Introduction

The problem of identifying duplicate features has been extensively studied in the Database field in the context of data integration. The literature on schema matching, data deduplication, and record linkage is prolific with many solutions having been proposed throughout the years [Madhavan et al., 2001, Do and Rahm, 2002, Storer et al., 2008, Meister et al., 2012, Bhattacharya and Getoor, 2004, Christen, 2008]. However, within the fields of machine learning (ML) and data mining, this problem is not well studied since most work is devoted to devising techniques for dimensionality reduction and feature selection.

The duplicate features issue appears in situations in which data comes from different sources (as with data integration) but also in large datasets such as medical databases. MIMIC-III [Johnson et al., 2016], for example, is a very important healthcare dataset used in hundreds of scientific works. It contains many events that were recorded as separate features (*e.g.,* arterial pressure is recorded as *arterial pressure*, *arterial bp mean*, *arterial blood pressure mean*, and *arterial bp mean #2*), although they share the same semantics. Whenever different sources or information systems are involved in data collection, ML tasks are subject to duplication and redundancy between features, making it necessary to create a denser representation of information with minimal loss of quality. Duplicate features lead to data sparsity (*i.e.,* missing feature values), which is known to have a negative impact on ML algorithms. Moreover, not only fitting high dimensional data is computationally expensive, but it is also prone to overfitting due to high complexity.

In this context, identifying and fusing duplicate features could potentially bring gains in terms of prediction quality and computational costs by reducing data sparsity and dimensionality. We note, however, that duplicate features fusion differs from dimensionality reduction techniques based on feature extraction (*e.g.,* Principal Component Analysis). Whereas the latter produces new features in a lower-dimensional space by creating linear combinations of the original variables, regardless of their semantics, feature fusion aims at preserving the underlying semantics of the data by aggregating information for closely-related features in a domain-specific fashion.

Manually evaluating a dataset in search for duplicate features is unfeasible, as pairwise comparisons are required. A small dataset with 100 features would require an expert to evaluate almost 5K pairs. Thus, to minimize the effort from the domain specialist, feature fusion methods should be able to learn the fusion rules from a small annotated sample. The model derived from the sample can then be applied to the complete dataset.

The goal of this paper is to *evaluate the hypothesis that fusing duplicate features can improve the predictive power of the data whilst reducing training time*. In order to do that, we propose a set of fusion features that capture evidences from different sources. These evidences are fed to a classification algorithm. We compared three types of algorithms: a traditional method (Random Forest) [Breiman, 2001]; and two methods that require only positive instances, the Positive Unlabeled method SKC [Bao et al., 2018] and One Class Classification Support Vector Machine (OSVM) [Schölkopf et al., 2000].

The fusion methods were applied to MIMIC-III [Johnson et al., 2016] in two ways. First, we performed an intrinsic evaluation to measure the quality of the duplicate feature detection. The results have shown that duplicate detection can achieve an F1 of 0.91 using our proposed features. Then, we ran an extrinsic evaluation to assess the effects of feature fusion on a mortality prediction task. With the extrinsic evaluation, we test our hypothesis regarding the benefit of feature fusion. These results do not support the hypothesis since learning from the original (unfused) dataset yielded better classification results.

2. Background

In this section, we summarize some important concepts in the context of the problem of fusing duplicate features.

2.1. Supervised Learning Algorithms

The context of this work involves supervised learning algorithms, *i.e.,* a subclass of ML in which a model is learned from annotated training instances and is further applied to classify unseen data. Several algorithms have been proposed for this task and are widely employed [Mitchell, 1997]. Below we briefly review the algorithms adopted in our work.

Naïve Bayes (NB) is a probabilistic classifier based on Bayes' theorem, with an assumption of independence among predictors. NB computes the posterior probability of each possible class y given our prior knowledge, represented by the input feature vector $X = (x_1, x_2, ..., x_N)$ from which conditional probabilities are estimated. The class that maximizes the posterior probability is returned by the classifier [Mitchell, 1997].

Random Forest (RF) is a tree-based ensemble algorithm, well known for its positive impact on performance variance. RF groups several Decision Trees with different branch structures that generate different paths. The tree outputs are combined in such a way that the RF output is generated from the class that appeared most often among the set of tree outputs present in the forest [Breiman, 2001]. Thus, it is necessary to configure the parameters such as (i) number of trees to be used, (ii) the measure of the quality of a split, such as *gini* for the Gini impurity and *entropy* for the information gain and (iii) the maximum depth of the tree.

2.2. One-class Classification

One-class Classification (OCC) works with the premise that the classifier should identify whether the new instance belongs to a target class. It is applicable to problems where neg-

ative instances represent failures, anomalies, or errors, making negative instances difficult to obtain. For this reason, OCC defines the classification limit around the positive class in order to allow the classifier to accept as much data as possible as a positive class and to minimize the probability of accepting data outside the target class. The difficulty lies in deciding, using only positive data, how narrow this boundary should be around the data and what attributes of these instances should be used to assist in the separation between positive and negative classes [Khan and Madden, 2014].

The OCC-Support Vector Machine (OSVM) algorithm is a modification of the Support Vector Machine (SVM) method that works according to the OCC premise, being used mainly in problems of anomaly detection. Schölkopf et al. [2000] developed an algorithm that returns +1 if the instance is within a small region that captures most of the data, which would be considered as the target class that share similar characteristics, and -1 for data outside that region. This way, when generating a hyperplane that separates the space, this function is used when the apprentice machine receives a new instance and must evaluate in which region it will be. Depending on the region, it will be labeled as positive or negative. This type of classifier prioritizes learning the characteristics of the positive class; for this reason its training phase uses only positive instances. The parameters that need to be configured for OSVM are the kernel type (linear, poly, rbf, sigmoid), the kernel coefficient (γ), and the limit between the fraction of training errors and the support vectors.

2.3. Positive-Unlabeled Learning

Positive-Unlabeled Learning (PUL) presents a different approach to training data: it assumes that all labeled input data is positive and that the remaining unlabeled data for the problem can be either positive or negative. PUL methods are different because both positive and unlabeled samples are used during the training phase. Bao et al. [2018] proposed the Set Kernel Classifier (SKC) method based on PUL, which dealt with this type of classification along with another learning problem, called Multiple Instance Learning. Their work is based on sets of instances called *bags* that are labeled as follows: if there is at least one positive instance, then the bag is labeled as positive; if there are no positive instances, then the bag is labeled as negative.

Two important parameters that must be configured when using PUL are the class prior, *i.e.,* the probability of the positive class, and the regularization term (λ). The class prior, in PUL problems, cannot be calculated since only a small sample labeled as positive is known during classifier training. Thus, during their experiments, Plessis et al. [2015] used variations of the class prior, however, they suggest that these parameters should be known or estimated at training time.

2.4. Classifier Evaluation

Classifier evaluation is done by comparing the predicted labels against ground truth labels. Some of the most widely used evaluation metrics are:

• Accuracy, which measures the proportion of the correctly classified instances.

• Precision, which measures what proportion of the instances that were classified as belonging to a given class, in fact belong to that class.

• Recall, which measures what proportion of the instances that belong to a given class that were assigned to the class by the classifier. The recall of the positive class is also

called TR_Rate.
• F1, which is the harmonic mean between precision and recall.
• FP_Rate, which measures the proportion of false positives.
• FN_Rate, which measures the proportion of false negatives.

In order to reduce the effects of variability, cross-validation (CV) [Kohavi, 1995] is typically employed. It consists in splitting the dataset into k-folds, with k-1 folds being used for training and the remaining fold used for testing. Training and test folds change k times so that all folds are used for testing. Then results of the k-folds are averaged.

3. Related Work

Feature fusion is a widely used method in the area of image classification and recognition. Sun et al. [2004] proposed a feature fusion method based on Canonical Correlation Analysis, which uses the correlation of two groups of features as for fusion and to eliminate redundant information between features. This reduction allowed only essential features of the image to be maintained, showing good performance in classification tasks. Scalzo et al. [2008] used the Feature Fusion Hierarchies model, which combines feature fusion and decision fusion, generated by an evolutionary algorithm for the classification of gender in images. This model significantly reduced the classification error when compared with PCA. Perez et al. [2012] demonstrated that feature fusion after feature selection using mutual information measures improves the performance of gender classification in images. Lin et al. [2015] proposed a fusion algorithm called Heterogeneous Structure Fusion that deals with the distribution of each feature and similarity between features. It was evaluated in image classification, face recognition, shape analysis, and infrared imagery. Their results revealed that this algorithm outperformed feature fusion methods for classification problems. Our work differs from feature fusion in image classification due to the nature of our features. Here, our scope is structured data. This work is the first to use feature fusion to fuse duplicate features.

The task of feature fusion shares similarities with schema matching, one of the phases of data integration that aims at identifying matching elements across different schemas so that they can be merged. Even though both methods have similar goals, the phases of (i) checking the data to detect the matches between the attributes and (ii) joining the attributes that match; are different. Gal [2006] and Sutanta et al. [2016] point to the dependence that schema matching tools have on the need for user interaction and how attached these tools are to the Data Base Management System for which they were designed. There is a vast literature on this topic [Bilke and Naumann, 2005, Do and Rahm, 2002, Bernstein et al., 2011]. However, schema matching solutions are designed to work with two or more schemas to integrate. In our work, the duplicate features are considered within a single schema.

4. Detecting and Fusing Duplicate Features

Table 1 shows a small example of duplicate and non-duplicate features. The data refers to health measurements of patients. This example is useful to illustrate the difference between *redundant* and *duplicate* features. Features “heart rate” and “Hr_rate” are *duplicate*, *i.e.,* they refer to the same measurement and should thus be fused. On the other hand, “temperature C” and “Temp F” (which record the patient’s body temperature in

Table 1. Examples of duplicate and non-duplicate features

f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
Subject	**temperature C**	**heart rate**	**NBP[systolic]**	**Hr_rate**	**Manual BP [systolic]**	**Temp F**	**Venous PVCO2**	**Venous PVO2**
1	30	71	100				35	38
2				100	120	97	40	42
3	32	75			135		32	41

Celsius and Fahrenheit, respectively) are not duplicate as they use different units of measurement. Fusing them would merge values coming from different distributions, which in turn would add noise to the learning tasks. These two features are *redundant*, as they are likely to be highly correlated. Thus, a feature selection algorithm, such as a wrapper method, is likely to discard one of them. In this paper, we focus on duplicate and not on redundant features.

The problem addressed here can be more formally described as follows. Given a set of original features $F = \{f_1, f_2, ..., f_n\}$, $\langle f_i, f_j \rangle$ is a pair of features, where $f_i \in F, f_j \in F$ and $i \neq j$. Each original feature f_i is associated with a name l_i and a set of values $V_i = \{v_1, v_2, ..., v_m\}$. The task of *identifying duplicate features* determines whether f_i and f_j are duplicate, *i.e.,* refer to the same actual feature. Then, *feature fusion* is responsible for merging the original features that have been identified as duplicates. Each duplicate pair $\langle f_i, f_j \rangle$ is fused to became a new feature $f_h = f_i \oplus f_j$, where $\oplus$ is a previously defined aggregating function and depends on the type of data. A new name l_h is created and associated to this new fused feature f_h.

Duplicate feature detection relies in a set of features that are fed into a classifier, which creates a model from an initial set of labeled instances. Once the model is created, it can be used to classify all pairs of features from the dataset. This is a very challenging task because of a number of reasons:

- The number of pairs of original features to analyze can be very large, which makes the task computationally expensive. To help mitigate this problem, the number of features should be kept small.
- The model will need to learn from a small set of labeled instances, since labeling is a labor-intensive task that needs to be performed by a domain-specialist.
- The number of negative pairs (*i.e.,* pairs of features that are not duplicate) is much larger than the number of positive pairs. This yields an extremely unbalanced dataset, which typically poses challenges to learning algorithms. In this sense, the use of learning schemes that rely solely on positive instances (such as the ones presented in Sections 2.2 and 2.3) is desirable.
- Capturing the semantic similarity of the attributes is very difficult. For example, the pair $\langle f_8, f_9 \rangle$ from Table 1 has similar names and their values lie within the same range. However, they are not duplicate as one refers to Carbon Dioxide and the other to Oxygen. This example reinforces the idea that using different sources of evidence is necessary.

4.1. Duplicate Detection

Duplicate detection is modeled as a binary classification problem. For a given pair of original features f_i and f_j, the task of the classifier is to assign a label stating whether f_i and f_j are duplicates. The decision as to whether any pair of original features is duplicate

is based on a set of fusion features that rely on evidences coming from different sources. These evidences are explained next.

Evidences based on feature names. In some situations, duplicate features have similar names. This can be seen in Table 1 as f_3 and f_5, for example, have similar names. In order to assess how similar two feature names are, we employed string similarity metrics in a similar fashion to what is done in data integration. Among the many metrics available, we chose Levenshtein, Jaro-Winkler, and Soundex. The goal was to capture different aspects of similarity (both syntactic and phonetic). Levenshtein, also known as edit distance, calculates how many changes (insertions, deletions, or substitutions) are required to transform one string into the other one. In order to calculate the similarity using this metric, we used the Normalized Levenshtein (Eq. 1). The Jaro-Winkler similarity (Eq. 2 and Eq. 3) is also based on shared characters. It considers that differences at the beginning of the word are more significant than differences at the end. It counts matching characters and takes transpositions into consideration. Both metrics result in a score between 0 and 1. A score of 1 denotes that the strings are identical and a score of 0 means that the strings do not share any characters. Unlike the other two metrics, Soundex is a phonetic algorithm that produces a representation of strings according to their sound and then, the similarity is calculated comparing the representation of the input strings; if they are equal, the similarity is 1; otherwise it is 0.

$$NormLev_{a,b} = 1 - Lev_{a,b} Lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{if } \min(i,j) = 0 \\ min \begin{cases} Lev_{a,b}(i-1,j)+1 \\ Lev_{a,b}(i,j-1)+1 \\ Lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \quad (1)$$

$$sim_j = \begin{cases} 0 & \text{if m} = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases} \quad (2)$$

$$sim_w = sim_j + \ell p(1 - sim_j) \quad (3)$$

Evidences from feature values. If two features are duplicate, then their values should be within the same range. This evidence helps distinguish between duplicate and redundant features. As shown in Table 1, the values of redundant features do not need to be in the same range. To assess how similar the distribution of values between the original features are, it is necessary to know their data types (*i.e.,*, numeric or categorical). If both features have different types, then their similarity score is zero. If both original features are numerical, then the Kolmogorov-Smirnov test is used. Finally, for categorical features, the cosine similarity is used. The Kolmogorov-Smirnov test computes whether two samples come from the same distribution. Is this hypothesis is accepted, then the p-value resulting from the test should be high. The cosine is commonly used to measure the similarity of two documents represented by their vectors. To calculate the cosine similarity, original features f_i and f_j are transformed into d dimensional term incidence vectors, where d is the number of distinct values in $V_i \cup V_j$. The vectors have the frequency counts of each distinct term.

$$D_{n,m} = \sup_x |F_{1,n}(x) - F_{2,m}(x)| \quad (4)$$

Evidences from co-occurrence. If two original features f_i and f_j are duplicate, then their values tend to co-occur infrequently, *i.e.,* instances do not normally have values for both f_i and f_j simultaneously. This can be seen in Table 1, as duplicate features $\langle f_3, f_5 \rangle$ and $\langle f_4, f_6 \rangle$ do not co-occur. To quantify feature co-occurrence, we compute the Jaccard Similarity between them, according to Eq. 5. The intersection between f_i and f_j is the number of instances in the dataset which have values for both f_i and f_j. When analyzing a database with temporal data, the definition of intersection can be adapted to account for events that occurred within a time interval (*i.e.,* the same day, hour, *etc.*).

$$Jaccard(f_i, f_j) = \frac{f_i \cap f_j}{f_i \cup f_j} \tag{5}$$

4.2. Feature Fusion

Once the duplicate features have been identified, the next step is to fuse them. The fusion process needs to deal with merging a possibly large number of original features. For example, if the pairs $\langle f_1, f_2 \rangle$ and $\langle f_2, f_3 \rangle$ are both identified as duplicate, then features f_1, f_2, and f_3 should be fused. In order to prevent an undesirably large cascading of fusions, we employ a threshold θ that specifies the maximum number of original features to be merged into a single one. In this process, original features that have a higher probability of being duplicate should be prioritized. Hence, the pairs of original features that were identified as duplicate are sorted in decreasing order of a score s that calculates the average of the values of the fusion features. Then, fusion processes the pairs with highest s scores first, merging them in groups of at most θ features.

In order to fuse the values of the original features, aggregating functions (*i.e.,* average, minimum, maximum, mode) are used to transform the feature distribution into a single value. Hence, from a set of instances $X = \{x_1, x_2, ..., x_m\}$, its processing creates $x_p = (f_{p,1}[g], f_{p,2}[g], ..., f_{p,n}[g])$, where g is the aggregating function used for each original feature and $x_p \in X$.

5. Intrinsic Evaluation

This evaluation aims to verify how well the classification methods perform on labeling each pair of original features as duplicate or not *i.e.,* here we perform an *intrinsic* evaluation of the methods. The comparison was made using RF, SKC (a PUL method) [Bao et al., 2018], and OSVM [Schölkopf et al., 2000]. In the next subsections, we describe data preparation for these experiments, model training procedures, and their results.

5.1. Materials and Methods

Data. Our data comes from the MIMIC-III v1.4 database [Johnson et al., 2016], which contains over 40K patients and thousands of variables. The patients are medical and surgical who were admitted to an Intensive Care Unit at a hospital in Boston-USA. Each patient has a number of associated events (*i.e.,* measurements for vital signs, values of laboratory tests, *etc.*). These events are the original features in our setting. This dataset was chosen because it has events stored with different ids but that represent the same event. This is a known issue that has made its organizers list the identified occurrences in a repository[1]. For example, there are six features referring to *Systolic Arterial Blood*

[1] https://github.com/MIT-LCP/mimic-code/blob/master/concepts/firstday/vitals-first-day.sql

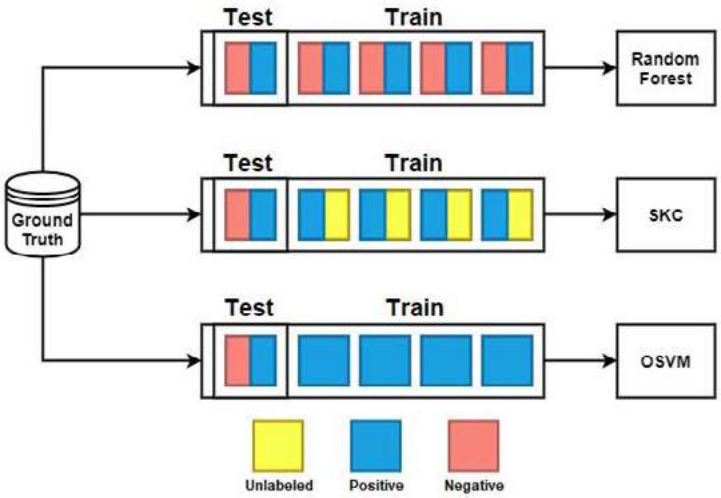


Figure 1. Configuration of training and testing datasets of each classifier.

Pressure. The dataset has 6,460 events (*i.e.,* original features), so a complete analysis to determine duplicate events would require 20,282,570 pairwise comparisons.

Ground Truth Generation. Given the high number of original features in MIMIC-III and the consequently huge number of feature pairs, it is unfeasible to have ground truth labels for every pair. Thus, we worked on a sample in the following manner. We used 125 pairs of features that have already been identified as duplicate by other researchers and are available at the MIMIC-III repository. Besides those, we adopted the following procedure to identify further duplicate features. First, the scores for fusion features (see Section 4) were computed for all 20M possible pairs. These pairs were sorted in decreasing order of the average scores of the fusion features. The assumption was that pairs of features with a higher chance of being duplicate would be at the top of the ranking. Then, 3K pairs from the top, middle, and bottom of the ranking were selected, amounting to 9K pairs. We then took a sample of 2,290 pairs that were manually labeled as *duplicate* or *not duplicate* by a medical doctor. At the end of the process, our ground truth has 338 pairs labeled as duplicate and 1,952 pairs labeled as non-duplicate. The sample for which the ground truth was generated amounts to 0.1% of the possible pairs in the original dataset.

Tools. In order to calculate the values for the fusion features, we used different tools and libraries. The `Strsim` library[2] was used to calculate the Normalized Levenshtein and Jaro-Winkler. The Jellyfish library[3] was used to calculate the scores for Soundex. The Kolmogorov-Smirnov statistic was computed using SciPy[4]. Finally, Scikit-learn[5] was used for the cosine similarity.

Experimental procedure. Cross-validation was performed using five stratified folds, according to Figure 1. The RF classifier considered both duplicate and non-duplicate labels. For OSVM, training used only the duplicate instances. The non-duplicate instances were evenly distributed across the test folds. For SKC, the non-duplicate instances were

[2]https://github.com/luozhouyang/python-string-similarity
[3]https://github.com/jamesturk/jellyfish
[4]https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.ks_2samp.html
[5]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

Table 2. Results for the Intrinsic Evaluation – Feature Fusion

	OSVM	SKC	RF
TP Rate/Recall	0.654	0.766	**0.867**
Precision	0.781	0.615	**0.958**
Accuracy	0.922	0.895	**0.975**
FP Rate	0.032	0.083	**0.007**
FN Rate	0.346	0.234	**0.133**
F1	0.712	0.682	**0.910**

considered unlabeled for training and the folds were divided into positive and unlabeled bags with three instances each. During test, the non-duplicate instances remained negative and the fold was divided into bags with only one instance.

After evaluating several parameters configurations for the algorithms, the best ones were defined as follows. OSVM: RBF kernel, kernel coefficient γ = 0.25, and the limit between the fraction of training errors and the support vectors nu = 0.35. SKC: training error rate (λ) = 0.0001 and the positive class probability (prior) = 0.15. RF: number of trees n_trees = 250, the function that measures the quality of the division criterion = Gini (Index), and max_depth = None.

5.2. Results

Table 2 shows the results for the intrinsic evaluation. RF was the best performer across all metrics. These results indicate that having negative instances helps the distinction between duplicate and non-duplicate features, despite the existing class imbalance. In a comparison between OSVM and SKC, the former is better in terms of precision, accuracy, FP rate, and F1, while the latter is better in terms of TP Rate and FN Rate.

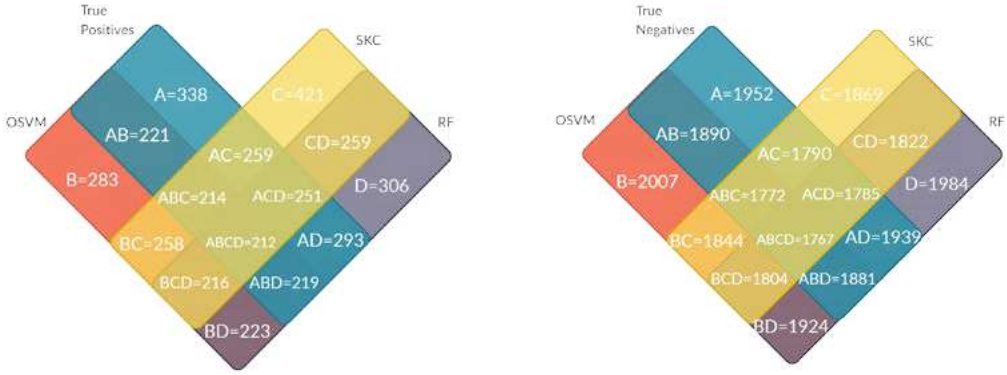
In order to assess whether the algorithms agreed on the instances they labeled as duplicate/not duplicate, we plotted the intersections of the predictions of each method w.r.t each other and the ground truth (*i.e.,* the "True Positive" group). The Venn diagrams are shown in Figure 2. We can see that 63% (212/338) of the duplicate features were identified by all three algorithms. For the non-duplicate features, the agreement was higher, reaching 91% (1767/1952).

Once the classification models were learned using the annotated data, they were used to label the complete dataset containing all possible pairs of original features. The number of instances labeled in each class by the three methods are shown in Table 3.

Table 3. Number of instances (*i.e.,* pair of original features) labeled as duplicate and not duplicate by each classifier.

	OSVM	SKC	RF
Duplicate	4,264,367	2,984,812	4,730,359
Not Duplicate	16,595,913	17,875,468	16,129,921

We also performed a manual analysis of the errors made by the classifiers. As a general tendency, we noticed that false positives tended to have highly similar names and distributions such as `temperature celsius` and `temperature f`. As for false negatives, they typically either have highly similar names and high co-occurrence such



(a) Duplicate (b) Not Duplicate

Figure 2. Diagrams showing the intersection among classifiers' predictions and the ground truth for the (a) positive (duplicate) and (b) negative (not duplicate) examples in our dataset.

as `tpa#mg/hr` and `tpa#2 mg/hr` or they have low similarity between their names, zero co-occurrence, and high similarity of distribution such as `glucose (70-105)` and `bloodglucose`.

6. Extrinsic Evaluation

This evaluation aims to assess the impact that feature fusion has on a classification task, *i.e.,* an *extrinsic* evaluation. The task used in the tests is mortality prediction, an important topic in medical informatics.

6.1. Materials and Methods

Data. As our original features come from MIMIC-III (described in Sec. 5), the data used for this evaluation also comes from this same dataset. Subjects were adult patients and the instances had only events and measurements from the first 24 hours of their last hospital stay. The task is to predict whether a patient will die based on information collected early at the hospital stay. We took a random sample of 13,171 patients (out of the 32,507 in the complete dataset). The sample has 4,563 deceased and 8,608 surviving patients.

Tools. Weka [Hall et al., 2009] was used for running the classifiers.

Experimental Procedure. We used the Naïve Bayes classifier with ten-fold CV. The threshold for feature fusion was θ = 15. Experimental runs were done with each method for feature fusion and two baselines – the original (unfused) dataset and also fusing only the features that we identified as duplicate in the ground truth. Due to computational limitations only Naïve Bayes classifier was used in this evaluation.

6.2. Results

The results for the mortality prediction task are shown in Table 4. We can see that all methods of feature fusion had a negative impact on the quality of the prediction. A paired t-test was used to compare F1 results (which are normally distributed) across the ten folds for each original and fused dataset. The results indicated that the reduction is statistically significant using $\alpha = 0.01$. Even the fusion based on the ground truth lowered the

Table 4. Results for the Extrinsic Evaluation – Mortality Prediction

Metric	Original	Ground Truth	OSVM	SKC	RF
TP Rate/Recall	**0.773**	0.772	0.755	0.759	0.762
Precision	**0.713**	0.712	0.699	0.700	0.700
Accuracy	**0.647**	0.646	0.633	0.633	0.631
FP Rate	**0.419**	0.420	0.432	0.434	0.438
FN Rate	**0.227**	0.228	0.245	0.241	0.238
F1	**0.655**	0.654	0.641	0.640	0.639
Training Time (seconds)	2.615	2.463	**1.000**	1.107	1.098
#Features	6,108	6,010	2,063	2,119	2,328

scores. This fact leads us to conclude that the hypothesis that fusing duplicate features can improve the predictive power of the data *could not be validated* in our experiments.

As expected, as a result of the reduction in the number of features, model training was much faster on the fused datasets. Training time was reduced to almost a third of the time as a result of the reduction in the number of features by the same proportion. We can see here that the duplicate detection process had many false positives since the duplicates actually represent a much smaller fraction of the set of original features.

7. Conclusion

In medical informatics, a recurrent problem found in datasets is duplicate features derived from similar, decentralized data sources, which results in greater dimensionality without proportionally increasing the value of the data. The hypothesis that drove this work was that fusing duplicate features could result in better generalization power of classifiers. Although the results for the intrinsic evaluation were quite high, the extrinsic evaluation showed that fusion was too aggressive. The duplicate detection phase should be further investigated, looking for more evidences that can assist in the process of labeling duplicate feature pairs. Training time was improved for extrinsic evaluation, but predictive power failed in surpassing baselines values.

We note, however, that our work had some important limitations. First, our tests were done over a single dataset. Despite being a real, challenging medical dataset, experiments on other datasets are needed to further investigate our hypothesis and improve our understanding on this topic. Moreover, further experiments could be performed exploring different proportions of labeled data for the duplicate features (which in our work was only 0.1% of the possible pairs in the original dataset), as well as new features aiming at reducing false positives. Finally, whereas lower dimensionality decreases model complexity, which in turn contributes to a better generalization power, we could not investigate more deeply in our scenario due to the lack of independent data.

Acknowledgements. This work was partially supported by CAPES Finance Code 001.

References

Han Bao, Tomoya Sakai, Issei Sato, and Masashi Sugiyama. Convex formulation of multiple instance learning from positive and unlabeled bags. *Neural Networks*, 105:132 – 141, 2018.

Philip A Bernstein, Jayant Madhavan, and Erhard Rahm. Generic schema matching, ten years later. *Proc. of the VLDB Endowment*, 4(11):695–701, 2011.

Indrajit Bhattacharya and Lise Getoor. Iterative record linkage for cleaning and integration. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD, page 11–18, 2004.

Alexander Bilke and Felix Naumann. Schema matching using duplicates. In *International Conference on Data Engineering (ICDE)*, pages 69–80, 2005.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Peter Christen. Febrl -: An open source data cleaning, deduplication and record linkage system with a graphical user interface. In *International Conference on Knowledge Discovery and Data Mining*, page 1065–1068, 2008.

Hong-Hai Do and Erhard Rahm. Coma—a system for flexible combination of schema matching approaches. In *International Conference on Very Large Databases*, pages 610–621, 2002.

Avigdor Gal. Why is schema matching tough and what can we do about it? *ACM Sigmod Record*, 35(4):2–5, 2006.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1): 10–18, 2009.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database, 2016.

Shehroz S Khan and Michael G Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014.

Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *International Joint Conference on Artificial Intelligence*, 1995.

Guangfeng Lin, Guoliang Fan, Xiaobing Kang, Erhu Zhang, and Liangjiang Yu. Heterogeneous feature structure fusion for classification. *Pattern Recognition*, 2015.

Jayant Madhavan, Philip A Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *VLDB*, volume 1, pages 49–58, 2001.

Dirk Meister, Jurgen Kaiser, Andre Brinkmann, Toni Cortes, Michael Kuhn, and Julian Kunkel. A study on data deduplication in hpc storage systems. In *International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–11, 2012.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, mar 1997.

Claudio Perez, Juan Tapia, Pablo Estévez, and Claudio Held. Gender classification from face images using mutual information and feature fusion. *International Journal of Optomechatronics*, 2012.

Marthinus Christoffel Du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *International Conference on Machine Learning*, volume 37, pages 1386–1394, jun 2015.

Fabien Scalzo, George Bebis, Mircea Nicolescu, Leandro Loss, and Alireza Tavakkoli. Feature fusion hierarchies for gender classification. *International Conference on Pattern Recognition*, 2008.

Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

Mark W. Storer, Kevin Greenan, Darrell D.E. Long, and Ethan L. Miller. Secure data deduplication. In *International Workshop on Storage Security and Survivability*, page 1–10, New York, NY, USA, 2008.

Quan-Sen Sun, Sheng-Gen Zeng, Yan Liu, Pheng-Ann Heng, and De-Shen Xia. A new method of feature fusion and its application in image recognition. *Pattern Recognition*, 2004.

Edhy Sutanta, Retantyo Wardoyo, Khabib Mustofa, and Edi Winarko. Survey: Models and prototypes of schema matching. *International Journal of Electrical and Computer Engineering*, 2016.