UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

HORTÊNSIA COSTA BARCELOS

Detecção e Fusão de Atributos Duplicados para Mineração de Dados

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dra. Viviane Pereira Moreira
Co-orientador: Prof. Dra. Mariana Recamonde Mendoza

Porto Alegre
2020

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Barcelos, Hortênsia Costa

Detecção e Fusão de Atributos Duplicados para Mineração de Dados / Hortênsia Costa Barcelos. – Porto Alegre: PPGC da UFRGS, 2020.

53 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2020. Orientador: Viviane Pereira Moreira; Coorientador: Mariana Recamonde Mendoza.

1. Fusão de Atributos. 2. Deduplicação. 3. Mineração de Dados. I. Moreira, Viviane Pereira. II. Mendoza, Mariana Recamonde. III. Título.

“The more I study,
the more insatiable do I feel
my genius for it to be.”
— Ada Lovelace

AGRADECIMENTOS

Agradeço a Deus pelas oportunidades colocadas em meu caminho.

Agradeço as minhas orientadoras profª Dra. Viviane Pereira Moreira e profª Dra. Mariana Recamonde Mendoza, que durante este trabalho contribuíram com seus vastos conhecimentos e experiências, me guiando pelo mestrado.

Agradeço a minha mãe, meu pai e irmãos pela paciência e apoio durante todo esse processo, que me permitiu seguir até o fim.

Agradeço aos meus amigos de laboratório, por tornarem meus dias mais leves e por todas as trocas que tivemos.

Agradeço também à toda a comunidade do INF-UFRGS que se empenham pra garantir uma formação de excelência aos seus estudantes.

RESUMO

Atributos duplicados são um problema recorrente em várias bases de dados geradas de fontes de dados similares e decentralizadas. Esta duplicação de atributos resulta em grande dimensionalidade sem aumentar proporcionalmente o valor da informação contida na base de dados. Buscando lidar com esta questão, o presente trabalho procurou aplicar métodos de identificação e fusão de atributos duplicados em conjuntos de dados utilizando algoritmos de aprendizado de máquina para tornar esse processo menos custoso a um especialista. O objetivo foi avaliar a hipótese de que a fusão de atributos duplicados pode melhorar o poder preditivo dos modelos enquanto reduz o seu tempo de treinamento. Um método foi proposto para detecção de atributos duplicados usando classificadores para rotularem pares de atributos como duplicados ou não. Esse método tem como base evidências extraídas da base de dados sendo analisada. As evidências indicam a semelhança entre os atributos de cada par e são separadas em três categorias: baseadas nos nomes dos atributos, no seus conjuntos de valores e na coocorrência de cada par. Ao tornar essa fase de identificação automática, o trabalho dos especialistas se torna menos custoso, visto que é necessário rotular manualmente um conjunto pequeno de pares de atributos para o modelo conseguir rotular os demais. Após a fase de detecção, um método de fusão das duplicatas também foi proposto, de forma que as informações presentes nos atributos duplicados fossem mantidos em um único atributo. Uma avaliação comparando a detecção de duplicados com um *ground truth* gerado manualmente obteve F1 de 0,91. Em seguida, os efeitos da fusão foram medidos em uma tarefa de predição de mortalidade. Assim, observamos que embora nosso modelo tenha sido bem sucedido na tarefa de encontrar pares de atributos duplicados, a fusão destes atributos de acordo com a estratégia implementada não foi efetiva na melhoria da qualidade da classificação. Ainda que tenha sido constatada uma redução do tempo de treinamento com o método de fusão de atributos, a mesma não foi compensada pelo ganho de qualidade na tarefa de classificação. Concluímos que a hipótese investigada não é válida e uma análise foi feita com relação aos pontos a serem aprimorados na fase de detecção de atributos duplicados, que se mostrou como o principal gargalo a ser tratado.

Palavras-chave: Fusão de Atributos. Deduplicação. Mineração de Dados.

Identifying and Fusing Duplicate Features for Data Mining

ABSTRACT

Duplicate features generated from similar and decentralized data sources are a recurring problem found in several databases. This duplication of features results in large dimensionality without proportionally increasing the value of the information contained in the database. Seeking to deal with this issue, the present work sought to apply methods for the identification and fusion of duplicate features in data sets using machine learning algorithms to make this process less costly for a specialist. The goal was to evaluate the hypothesis that the fusion of duplicate features can improve the predictive power of the models while reducing their training time. A method has been proposed for detecting duplicate features using classifiers to label pairs of features as duplicates or not duplicates. This method is based on evidences extracted from the database being analyzed. The evidence indicates the similarity between a pair of attributes and is separated into three categories: based on the feature names, their sets of values, and their co-occurrence. Through automatic identification, the burden on the specialists is reduced, since it is only necessary to manually label a small set of attribute pairs for the model to be able to label the others. After the detection phase, a method for fusing the duplicates was also proposed, so that the information present in the duplicate features is merged into a single feature. An evaluation comparing the detection of duplicates with a manually generated ground truth obtained an F1 of 0.91. Then, the effects of the fusion were measured in a mortality prediction task. Thus, we observed that although our model was successful in finding duplicate pairs of features, the fusion of these attributes according to the strategy implemented was not effective in improving performance. Even though the feature fusion method brought a reduction in training time, it was not compensated by the performance gain in the classification task. We conclude that the investigated hypothesis is not valid and an analysis was made regarding the points to be improved in the phase of detecting duplicate features, which proved to be the main bottleneck to be addressed.

Keywords: Feature Fusion, Deduplication, Data Mining.

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina
MIL	Multiple Instance Learning
NB	Naïve Bayes
OCC	One-Class-Classification
OSVM	OCC-Support Vector Machine
PUL	Positive-Unlabeled Learning
RF	Random Forest
SKC	Set Kernel Classifier
SVDD	Support Vector Data Description
SVM	Support Vector Machines

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Atributos duplicados dificultam a coleta de informações e atrapalham o desempenho de classificadores, uma vez que esse tipo de ruído colabora para a grande dimensionalidade do problema sem proporcionalmente aumentar os valores presentes nas base de dados.

Esse ruído pode aparecer em uma base de dados que foi gerada a partir de fontes de dados similares e descentralizadas, *i.e.,* ao tentar integrar uma base de dados com estrutura semelhante a outra base que também contém dados importantes para o sistema. Esse tipo de cenário é um problema recorrente encontrado também em informática da saúde.

A tarefa de identificar atributos duplicados foi extensivamente estudada no campo de Banco de Dados no contexto da integração de dados. A literatura sobre *Schema Matching* e deduplicação de dados é diversa, com muitas soluções propostas ao longo dos anos (MADHAVAN; BERNSTEIN; RAHM, 2001; DO; RAHM, 2002; STORER et al., 2008; MEISTER et al., 2012; BHATTACHARYA; GETOOR, 2004; CHRISTEN, 2008). No entanto, dentro dos campos de Aprendizado de Máquina (AM) e Mineração de Dados, essa questão não tem recebido muito foco, pois a maioria dos trabalhos é dedicada à criação de técnicas para redução de dimensionalidade e seleção de atributos.

O problema dos atributos duplicados aparece em situações em que os dados vêm de diferentes fontes (como na integração de dados), mas também em grandes conjuntos de dados, como bancos de dados médicos. A base de dados MIMIC-III (JOHNSON et al., 2016), por exemplo, possui dados de saúde muito importantes, usados em centenas de trabalhos científicos. Ela contém muitos eventos que foram registrados como atributos diferentes (*e.g.,* a pressão arterial é registrada como *arterial pressure*, *arterial bp mean*, *arterial blood pressure mean*, e *arterial bp mean #2*), embora compartilhem a mesma semântica. Sempre que diferentes fontes ou sistemas de informação estão envolvidos na coleta de dados, as tarefas de AM estão sujeitas a duplicação e redundância entre os atributos, tornando necessário criar uma representação mais densa das informações com perda mínima de qualidade. Atributos duplicados levam a esparsidade de dados (*i.e.,* atributos com valores faltantes), que é conhecido por ter um impacto negativo em algoritmos de AM. Além disso, não apenas o ajuste de modelos a dados de alta dimensão é computacionalmente caro, mas também é propenso a *overfitting* devido à alta complexidade.

Nesse contexto, a identificação e fusão de atributos duplicados podem trazer ganhos em termos de qualidade de previsão e custos computacionais, reduzindo a esparsi-

dade e a dimensionalidade dos dados. Observamos, no entanto, que a fusão de atributos duplicados difere das técnicas de redução de dimensionalidade com base na extração de atributos (*e.g.,* Análise de Componentes Principais). Enquanto o último produz novos recursos em um espaço de menor dimensão, criando combinações lineares das variáveis originais, independentemente de sua semântica, a fusão de atributos visa preservar a semântica subjacente dos dados, agregando informações em atributos intimamente relacionados de uma maneira específica do domínio.

A avaliação manual de um conjunto de dados em busca de atributos duplicados é inviável, pois são necessárias comparações por pares. Um pequeno conjunto de dados com 100 atributos exigiria que um especialista avaliasse quase 5 mil pares. Portanto, para minimizar o esforço do especialista em domínio, os métodos de detecção de atributos duplicados devem ser capazes de aprender as regras de como identificar possíveis atributos duplicados a partir de uma pequena amostra anotada. O modelo derivado da amostra pode ser aplicado ao conjunto de dados completo.

O objetivo deste trabalho é *avaliar a hipótese de que a fusão de atributos duplicados pode melhorar o poder preditivo dos dados e reduzir o tempo de treinamento*. Para isso, propomos um conjunto de atributos de fusão que captura evidências de diferentes fontes. Essas evidências são fornecidas a um algoritmo de classificação que deve identificar atributos duplicados, gerando uma nova base de dados com pares de atributos rotulados como duplicados ou não. Essa nova base de dados é então utilizada para realizar a fusão dos atributos rotulados como duplicados. Comparamos três tipos de algoritmos: um método tradicional (Floresta Aleatória (BREIMAN, 2001)); e dois métodos que exigem apenas instâncias positivas, o método de Aprendizado Positivo-Não-Rotulado SKC (BAO et al., 2018) e uma Máquina de Vetores de Suporte de Classificação de Classe Única (OSVM).

Os métodos de fusão foram aplicados ao MIMIC-III (JOHNSON et al., 2016) de duas maneiras. Primeiro, uma avaliação intrínseca foi realizada para medir a qualidade da detecção de atributos duplicados. Os resultados mostraram que a detecção de atributos duplicados atingiu um F1 de 0,91 usando os atributos de fusão propostos. Em seguida, uma avaliação extrínseca foi aplicada para avaliar os efeitos da fusão de atributos em uma tarefa de previsão de mortalidade. Com a avaliação extrínseca, a hipótese sobre o benefício da fusão de atributos foi testada. Os resultados obtidos não permitiram sustentar a hipótese investigada, pois o aprendizado do conjunto de dados original (não fundido) produziu melhores resultados de classificação.

Outra contribuição desta dissertação foi um artigo que foi aceito pelo Simpósio Brasileiro de Banco de Dados (BARCELOS; RECAMONDE-MENDOZA; MOREIRA, 2020). Esse artigo apresenta, de forma mais sucinta, os procedimentos e resultados discutidos nesse documento.

Este documento está estruturado da seguinte forma. O Capítulo 2 introduz a fundamentação teórica necessária para a compreensão deste trabalho. O Capítulo 3 discute os trabalhos relacionados, apresentando uma revisão sobre as áreas em que o método de fusão de atributos é utilizado e *schema matching*. O Capítulo 4 apresenta a abordagem utilizada neste trabalho para a detecção e fusão de atributos duplicados. O Capítulo 5 detalha os procedimentos e resultados da avaliação intrínseca, enquanto o Capítulo 6 trata sobre os procedimentos e resultados da avaliação extrínseca. Por fim, o Capítulo 7 sumariza as contribuições deste trabalho e explora possibilidades para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo apresenta os conceitos necessários para compreender este trabalho. Primeiro são definidos os conceitos básicos de Aprendizado de Máquina evidenciando problemas de classificação, seguido de descrições de algoritmos de aprendizado supervisionado utilizados para solucionar esses problemas. Na sequência são discutidas métricas de avaliação de classificadores e estratégias de redução de dimensionalidade.

2.1 Aprendizado de Máquina

O Aprendizado de Máquina (AM) refere-se a um conjunto de métodos computacionais que têm como objetivo otimizar critérios de desempenho que aprimorem a habilidade de generalização do modelo, o qual influenciará em predições acuradas de instâncias desconhecidas, a partir de dados ou experiências prévias, ou, ainda, detectar padrões e regularidades nos dados sem intervenção de um especialista. Esses métodos possuem um modelo no qual são definidos parâmetros, que são otimizados durante o processo de aprendizagem usando dados de treinamento ou experiências passadas. A qualidade e o volume desses dados são cruciais para o sucesso das predições resultantes (ALPAYDM, 2010).

Algoritmos de aprendizado de máquina podem ser categorizados segundo a estratégia utilizada durante o processo de aprendizagem, os quais diferem segundo os tipos de dados disponíveis ao aprendizado do modelo, como os dados de treinamento são recebidos e no modo que os dados de teste são avaliados pelo algoritmo (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012). Na Tabela 2.1 são destacados três tipos de algoritmos de aprendizado e suas diferenças com relação aos dados como entrada dos modelos na fase de treinamento.

Tabela 2.1: Tipos de algoritmos de aprendizado.

Algoritmos de Aprendizado	**Dados de Treinamento**
Supervisionado	Rotulados
Não-Supervisionado	Não rotulados
Semi-Supervisionado	Rotulados e não rotulados

Esses algoritmos são usados em uma variedade de problemas de aprendizagem. Uma categoria de problemas é o de classificação, o qual é definido como uma tarefa de predição do rótulo y, também denominado como *classe* do problema, a partir de um vetor

de entrada x de dimensões K, em que $y \in Y = \{C_1, C_2, ..., C_Q\}$ onde C_i são os rótulos que uma instância do problema pode receber. Esta tarefa é executada usando a função de mapeamento $g : X \Rightarrow Y$ que permite prever a classe de novas instâncias (MARSLAND, 2014). Assim, em problemas de classificação usando algoritmos de aprendizado supervisionado, durante a fase do treinamento a máquina aprendiz usa os atributos de cada instância do problema, os quais descrevem características de cada item da entrada X, junto aos rótulos Y já conhecidos das entradas X para adquirir os conhecimentos sobre o problema a ser solucionado.

Para resolver o problema de classificação, a função de mapeamento é otimizada de acordo com a função de perda, que mede a diferença entre o rótulo predito Y' e o rótulo verdadeiro Y, uma vez que o caso ideal é a máquina aprendiz retornar $Y' = Y$. Essa função então permite identificar o quão errada está a função de mapeamento e possibilita a otimização do algoritmo de aprendizado buscando diminuir o resultado da função de perda. Esse processo de otimização ocorre na fase de treinamento da modelo (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012). Dessa maneira, o vetor de entrada é separado em instâncias de treinamento e teste, sendo a fase de teste o momento de avaliar o desempenho da função de mapeamento otimizado ao usar instâncias que não foram usadas durante a otimização no treinamento (ALPAYDM, 2010).

2.2 Classificação Binária

O contexto deste trabalho envolve algoritmos de aprendizado supervisionado para classificação binária, *i.e.,* uma subclasse de AM, na qual um modelo é aprendido a partir de treinamento sobre instâncias, que são rotuladas segundo uma das duas classes esperadas do problema, sendo então aplicado para classificar dados não vistos. Vários algoritmos foram propostos para esta tarefa e são amplamente empregados (MITCHELL, 1997). Em seguida são revisados brevemente os algoritmos adotados em neste trabalho.

2.2.1 Naïve Bayes

Naïve Bayes (NB) é um classificador probabilístico baseado no teorema de Bayes, o qual calcula a probabilidade de um evento A acontecer dado a probabilidade de um outro evento B acontecer, como estabelecido na Equação 2.1, onde A e B são eventos, $P(A)$ e

$P(B)$ são as probabilidades a priori de cada um dos eventos e $P(A|B)$ é a probabilidade posteriori, ou seja, probabilidade do evento A acontecer dado que evento B acontece.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

Este tipo de classificador supõe que há independência entre os atributos. NB calcula a probabilidade posteriori de cada classe possível y, dado o conhecimento prévio sobre o problema, representado pelo vetor de atributos de entrada $X = (x_1, x_2, ..., x_N)$ a partir do qual as probabilidades condicionais são estimadas. A classe que maximiza a probabilidade posteriori é retornada pelo classificador, que fornece um método para calcular a probabilidade de um evento A acontecer caso outro evento B acontecer. Assim, o algoritmo procura calcular a probabilidade condicional de uma instância ser classificada como uma classe y, uma vez que a entrada seja $X = (x_1, x_2, ..., x_N)$ (MITCHELL, 1997).

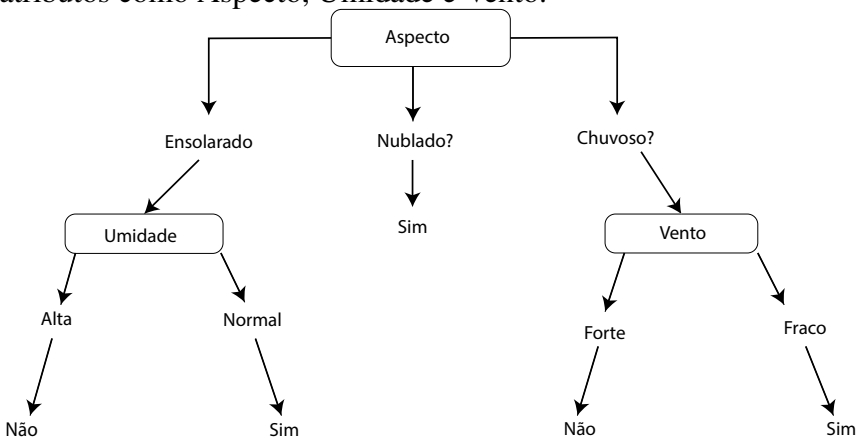
NB apresenta um desempenho relativamente maior do que outros classificadores, por ser mais simples e mais rápido, precisando de um pequeno número de dados de teste para concluir classificação com uma boa precisão. Por este motivo, NB foi utilizado neste trabalho para tratar com a tarefa de classificação na avaliação extrínseca pela grande quantidade de atributos presente nessa tarefa.

2.2.2 Random Forest

Random Forest (RF) é um algoritmo do tipo *ensemble* baseado em Árvore de Decisão, conhecido por seu impacto positivo no desempenho do modelo pela redução da variância. Neste tipo de estrutura, cada nó representa a verificação de uma condição sobre um atributo do problema. Cada ramo da árvore separa os nós de acordo com os resultados esperados da condição estabelecida no nó de onde sai a aresta. Os nós folha representam as classes que uma instância pode ser classificada. O caminho da raiz para a folha descrevem as regras de classificação. A Figura 2.1 apresenta uma Árvore de Decisão para predizer as classes *Sim* ou *Não* quanto à possibilidade de jogar futebol de acordo com os atributos Aspecto, Umidade e Vento.

O RF agrupa várias Árvores de Decisão com diferentes estruturas de ramificação que geram caminhos diferentes. As saídas em árvore são combinadas de forma que a saída de RF seja gerada a partir da classe que apareceu com mais frequência entre o conjunto de saídas das árvores presentes na floresta (BREIMAN, 2001). A Figura 2.2 apresenta

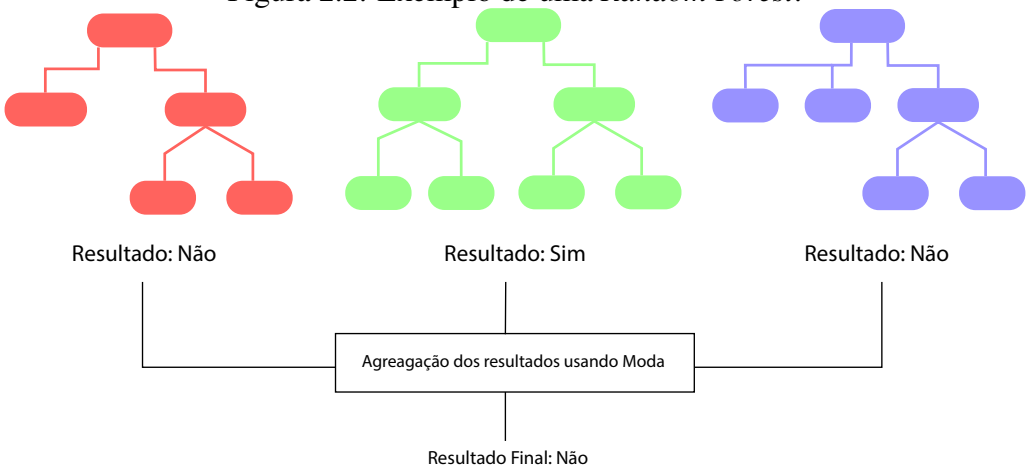
Figura 2.1: Exemplo de uma Árvores de Decisão que busca predizer os rótulos Sim/Não utilizando atributos como Aspecto, Umidade e Vento.



Fonte: Elaborada pela autora.

um exemplo de RF que contém três Árvores de Decisão em que cada uma gerou uma classificação, as quais serão contabilizadas para a geração do resultado da RF. No caso do exemplo apresentado na Figura 2.2 em que a floresta possui três árvores, o resultado da predição seria a classe *Não*, pois é a saída que aparece mais vezes no conjunto de saídas das árvores. Alguns dos parâmetros a serem configurados são (i) número de árvores a serem usadas, (ii) a medida da qualidade das divisões, como índice Gini ou Ganho de Informação e (iii) a profundidade máxima da árvore.

Figura 2.2: Exemplo de uma *Random Forest*.



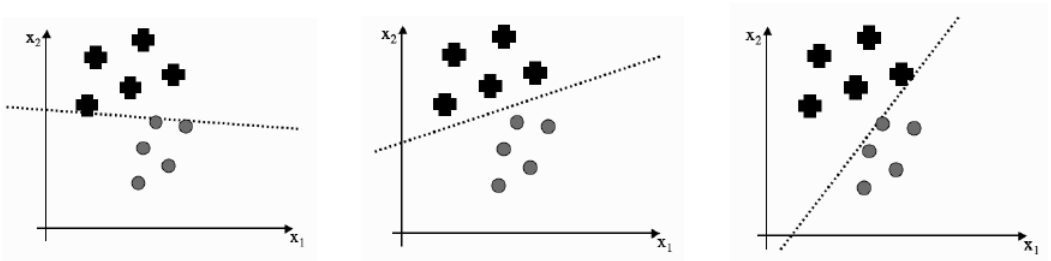
Fonte: Elaborada pela autora.

Pelo agrupamento de várias Árvores de Decisão, o classificador RF consegue evitar o *overfitting*, que é um erro de modelagem que demonstra a dificuldade do modelo de generalizar para dados não vistos após o treinamento. Assim, esse trabalho usou esse classificador na avaliação intrínseca que apresentou bom desempenho em diferenciar as classes do problema.

2.2.3 Máquina de Vetores de Suporte

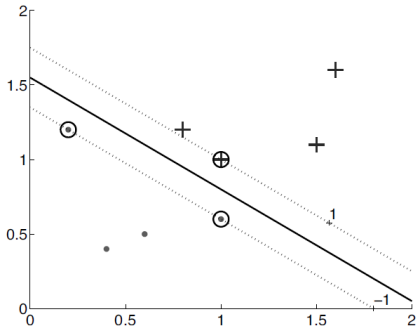
O algoritmo Máquina de Vetores de Suporte (*Support Vector Machine*) (SVM) mapeia o vetor de entrada para um espaço dimensional Z. Nesse espaço dimensional, uma superfície linear é criada, chamada de hiperplano, a qual deve permitir alto nível de generalização. Como pode ser visto na Figura 2.3, o espaço dimensional será muito grande e nem todos os hiperplanos que separarão os dados de treinamento retornarão um alto nível de generalização. Para conseguir isso, o SVM trabalha com Vetores de Suporte, inspiração para o nome do algoritmo, os quais são vetores de ambas as classes em que a distância entre eles maximiza a margem para o hiperplano (CORTES; VAPNIK, 1995), como pode ser visualizado na Figura 2.4, que exemplifica diferentes hiperplanos sendo testados para discriminar as classes utilizando as instâncias rotuladas da fase de treinamento. Ao utilizar o SVM alguns parâmetros podem ser configurados como o tipo de kernel a ser utilizado, o qual pode ser linear ou não-linear, como radial ou polinomial; para alguns desses kernels é necessário definir um coeficiente *gamma* a ser utilizado.

Figura 2.3: Exemplos de hipóteses que permitem discriminar as classes.



Fonte: Marsland (2014).

Figura 2.4: O hiperplano separa as duas classes: pontos e cruzes. As instâncias envolvidas com círculo representam os Vetores de Suporte determinando a margem do hiperplano.



Fonte: Alpaydm (2010).

2.3 Classificação de Classe Única

Em problemas tradicionais de classificação binária, o classificador jamais se depara com um dado que foge do domínio do problema aprendido. Caso isso ocorresse, o classificador não conseguiria discriminar essa nova instância do domínio do problema e a classificaria segundo uma das duas classes conhecidas previamente, gerando um rótulo errado. Nesse caso, o problema de classificação deve ser modificado para identificar se a nova instância pertence a uma classe alvo (rótulo positiva) ou não pertence (rótulo negativa).

Tarefas de *One-Class-Classification* (OCC), ou classificação de classe única, funcionam segundo essa premissa, rotulando instâncias como pertencentes ao conjunto esperado ou não. Porém, problemas do mundo real geralmente apresentam poucos (ou nenhum) casos negativos para auxiliar na fase de treinamento da máquina aprendiz. Isso ocorre principalmente em problemas que instâncias negativas representam falhas, anomalias ou erros, tornando dados negativos custosos de serem obtidos.

Por este motivo, no OCC define-se o limite de classificação ao redor da classe positiva de forma a permitir que o classificador aceite o máximo possível de dados como classe positiva e minimize a probabilidade de aceitar dados fora da classe alvo. A dificuldade está em decidir, usando somente os dados positivos, quão estreito esse limite deve ser ao redor dos dados e quais atributos dessas instâncias devem ser usadas para auxiliar na separação entre as classes positivas e negativas (KHAN; MADDEN, 2014).

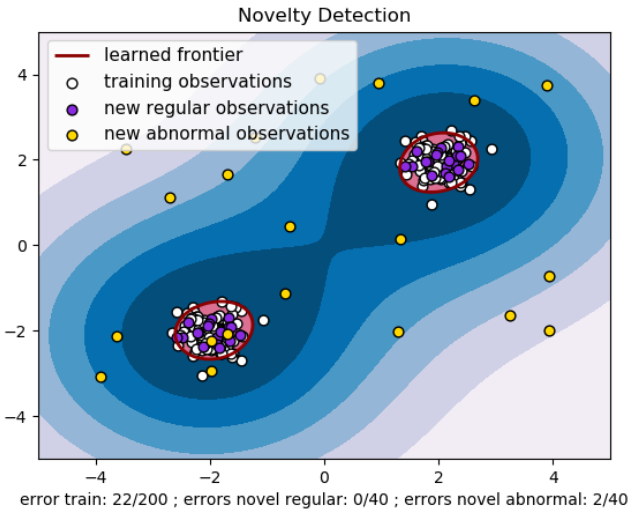
2.3.1 Máquina de Vetores de Suporte para Classe Única

O algoritmo *OCC-Support Vector Machine* (OSVM) é uma modificação do método SVM que funciona segundo a premissa do OCC, sendo usado principalmente em problemas para detectar anomalias, que são instâncias que são inconsistentes com os demais dados do conjunto.

Schölkopf et al. (2000) desenvolveram um algoritmo com uma função que retorna +1 caso a instância esteja dentro de uma região pequena que capture a maioria dos dados, os quais seriam considerados como a classe alvo que compartilham de características similares, e -1 para dados fora dessa região. Desta forma, ao gerar um hiperplano que separa o espaço de atributos, essa função é usada quando a máquina aprendiz recebe uma nova instância e deve avaliar em qual região ela ficará. Dependendo da região, ela será

rotulada como positiva ou negativa. Por lidar com problemas que apresentam poucos casos negativos e ter como prioridade aprender as características da classe positiva, esse tipo de classificador utiliza somente as instâncias positivas durante o treinamento. A Figura 2.5 apresenta um problema de detecção de novidade sendo resolvido utilizando a técnica de OSVM.

Figura 2.5: Exemplo do método OSVM segundo Schölkopf et al. (2000), sendo utilizado para solucionar problema de detecção de novidades.



Fonte: Pedregosa et al. (2011).

Outro método inspirado no SVM e otimizado para trabalhar com problemas OCC é encontrado na literatura como *Support Vector Data Description* (SVDD), o qual é apresentado por Tax and Duin (2001). A diferença está no modo como as classes são discriminadas: ao invés de utilizar um hiperplano, uma hiper-esfera é colocada ao redor dos dados de treinamento, os quais consistem apenas de dados rotulados como positivos. Então o método deve ser capaz de minimizar o volume da hiper-esfera para que não haja erros na discriminação das classes, mantendo as instâncias positivas dentro da hiper-esfera e as instâncias negativas fora da mesma.

Neste trabalho foi utilizado OSVM, permitindo testar se um classificador de classe única com prioridade para aprender as características da classe positiva apresentaria desempenho melhor que um classificador binário tradicional.

2.3.2 Aprendizado Positivo Não-Rotulado

Positive-Unlabeled Learning (PUL) é um método do tipo Classificação de Classe Única, que se diferencia dos métodos OSVM e SVDD vistos anteriormente quanto aos da-

dos utilizados durante a fase de treinamento. Como foi apresentado na Subseção 2.3.1, os métodos OSVM e SVDD utilizam na fase de treinamento somente as amostras positivas presentes no conjunto de dados, ignorando os dados não-rotulados no momento de criação da função discriminante das classes. Métodos PUL são diferentes porque durante a fase de treinamento são aplicadas ambas amostras positivas e não-rotuladas. Deve-se reforçar que esses problemas não possuem amostras negativas e que as amostras não-rotuladas podem consistir de ambas classes (positivas e negativas).

Plessis, Niu and Sugiyama (2015) propõem uma técnica cuja principal ideia é usar funções de perda para amostras positivas e não-rotuladas, sendo uma função de perda convexa comum $l(z)$ para as amostras não-rotuladas e uma função de perda composta $l(z) - l(-z)$ para as amostras positivas, caso essa seja uma função convexa, esse tipo de função tem como propriedade de que dado dois pontos quaisquer A e B desta função, o seu gráfico fica abaixo do segmento AB. Então toda a função objetiva se torna convexa permitindo que a solução global seja obtida. Segundo aos autores, essa proposta aprimora a formulação não convexa e visa impedir o viés de abordagens não-convexas.

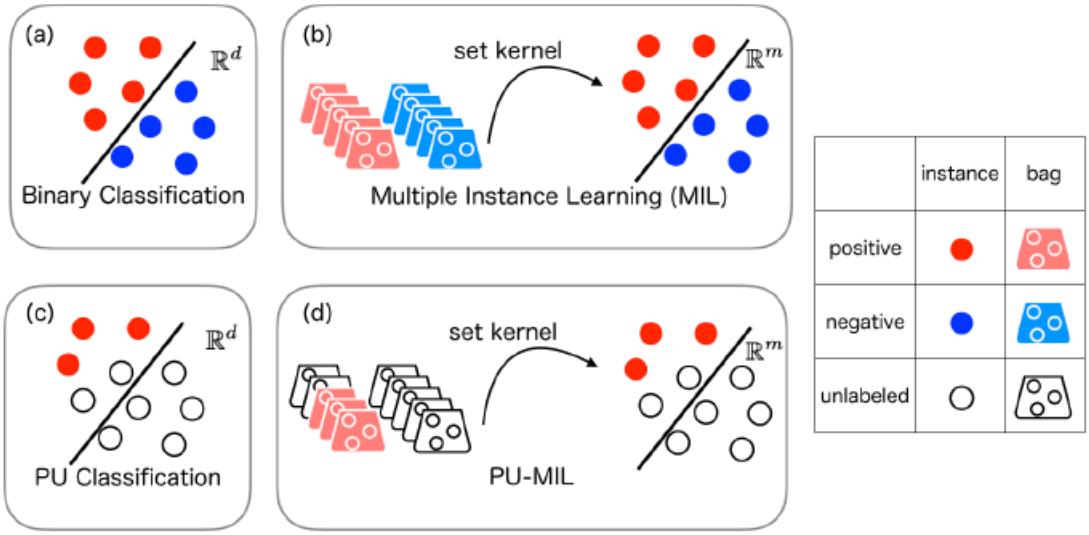
Bao et al. (2018) tratam a classificação PUL junto a outro problema de aprendizagem, denominado *Multiple Instance Learning* (MIL), o qual funciona lidando com conjuntos de instâncias denominadas como bolsas (i.e, *bags*). O rótulo dessas bolsas é definido segundo as seguintes regras.

- Se houver pelo menos uma instância positiva, então a bolsa é rotulada como positiva
- Se não houver instâncias positivas, então a bolsa é rotulada como negativa

A diferença entre ambos os trabalhos relativa ao modo como cada um representou seus dados, como pode ser visualizada na Figura 2.6-c e Figura 2.6-d, visto que Bao et al. (2018) aplicam MIL ao problema de classificação tratado com PUL. Ambos trabalhos utilizam SVM para treinar os seus classificadores, aplicando a minimização empírica de risco proposta pelos autores Plessis, Niu and Sugiyama (2015), que formula a otimização convexa.

Dois parâmetros importantes que devem ser configurados ao usar PUL são (i) a probabilidade de distribuição da classe positiva, chamada de *class prior* e (ii) o termo de regularização (λ). O parâmetro *class prior*, em problemas de PUL não é possível ser calculado, visto que somente uma pequena amostra rotulada como positiva é conhecida durante o treinamento do classificador. Dessa forma, durante os seus experimentos, os autores Plessis, Niu and Sugiyama (2015) utilizaram variações da *class prior*, no entanto,

Figura 2.6: Exemplificação de problemas de aprendizado, sendo (c) abordado por Plessis, Niu and Sugiyama (2015) e (d) por Bao et al. (2018).



Fonte: Bao et al. (2018).

eles sugerem que esse parâmetro seja conhecido no momento do treinamento ou utilizado um método para estimar seu valor.

Além do OSVM como um classificador de classe única, também foi utilizado o SKC como exemplo de um classificador PUL. Esse algoritmo foi escolhido para esse trabalho, pela implementação presente no repositório da biblioteca pywsl[1].

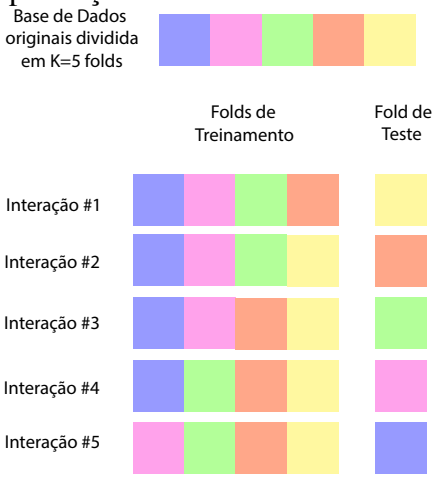
2.4 Avaliação de Classificador

Para avaliar a qualidade do classificador, um método frequentemente usado é a validação cruzada. Esse método permite separar as instâncias em *K-folds*, com K-1 *folds* para treinamento e um *fold* para teste do classificador. Os *folds* de treinamento e teste mudam K vezes para que todos os *folds* sejam utilizados como *folds* de teste, como pode ser visualizado na Figura 2.7, permitindo que todos os dados sejam usados na etapa de teste mas garantindo que os mesmos são independentes dos dados usados no treinamento. Esta estratégia é usada para evitar o *overfitting*, em que o classificador se ajusta tão bem aos dados de treinamento, que apresenta dificuldade em prever corretamente casos desconhecidos.

Há uma variação da validação cruzada intitulada como estratificada. Nessa variação os *folds* são divididos de forma que cada um tem aproximadamente a mesma pro-

[1]Disponível em: <https://github.com/t-sakai-kure/pywsl>

Figura 2.7: Exemplificação do funcionamento da validação cruzada.



Fonte: Elaborada pela autora.

porção de rótulos que a base de dados original (KOHAVI, 1995). A medida de qualidade resultante da validação cruzada dos *K-folds* é então a média e o desvio padrão dos valores de desempenho (*i.e.,* métricas de qualidade) calculados ao alterar esses *folds* em treinamento e teste.

As métricas de qualidade são calculadas de acordo com as saídas geradas pelo classificador, onde verdadeiros positivos ($\#TP$) representam o número de instâncias que foram corretamente classificadas como positivas, enquanto que falsos positivos ($\#FP$) são a quantidade de instâncias que foram incorretamente classificadas como positivas. Da mesma forma, para instâncias classificadas como negativas, em que verdadeiros negativos ($\#TN$) é o total de instâncias classificadas corretamente e falsos negativos ($\#FN$) é a quantidade de instâncias classificadas incorretamente. Utilizando esses valores é possível avaliar a qualidade do classificador ao calcular métricas como precisão, *recall*, *f1-score*, a proporção de falsos positivos (PFP) e a proporção de falsos negativos (PFN) (MARSLAND, 2014), que são calculadas de acordo com as Fórmulas 2.2, 2.3, 2.4, 2.5 e 2.6.

$$Precisao = \frac{\#TP}{\#TP + \#FP} \tag{2.2}$$

$$Recall = \frac{\#TP}{\#TP + \#FN} \tag{2.3}$$

$$F1 - Score = 2 \times \frac{Precisao \times Recall}{Precisao + Recall} \tag{2.4}$$

$$PFP = \frac{\#FP}{\#FP + \#TN} \tag{2.5}$$

$$PFN = \frac{\#FN}{\#FN + \#TP} \tag{2.6}$$

A Precisão avalia quantas instâncias rotuladas pelo classificador foram corretamente identificadas como positivas, enquanto o *Recall* da classe positiva, também conhecido como Proporção de Verdadeiros Positivos, determina a proporção das instâncias que foram rotuladas como positivas corretamente. O *F1-Score* indica o equilíbrio entre Precisão e *Recall*, calculado como a média harmônica entre ambas as métricas. A PFP avalia quantas instâncias marcadas como negativas têm os rótulos incorretos. A PFN avalia quantas das ocorrências marcadas como positivas são identificadas incorretamente.

2.5 Redução de Dimensionalidade

A crescente quantidade de dados sendo gerados diariamente, chegando a quintilhões de bytes diários, representa problemas com milhares de atributos, que são descritos como problemas de alta dimensionalidade (CHEN; MAO; LIU, 2014). A alta dimensionalidade dos problemas pode dificultar a utilização de algoritmos de aprendizado de máquina, sendo muitas vezes necessário realizar um pré-processamento dos dados - processo que inclui métodos de seleção de atributos. Segundo os autores Zhai, Ong and Tsang (2014), métodos de seleção de atributos têm como objetivo encontrar um subconjunto de atributos relevantes dentro do espaço original de atributos, a fim de tornar o modelo mais robusto, acurado e com rápido aprendizado.

Há dois tipos de métodos utilizados para redução de dimensionalidade, i) a seleção de atributos que se preocupa em selecionar do espaço original de atributos apenas as dimensões relevantes ao problema que está sendo solucionado e ii) a extração de atributos que atenta a transformação do espaço original de atributos em um espaço de menor dimensão que preserve o máximo de informação relevante, resultando em uma nova representação dos dados como uma combinação linear da entrada original, por exemplo o método *Principal Component Analysis* (PCA) (Khalid; Khalil; Nasreen, 2014).

A seleção de atributos é um processo que separa os atributos que melhor definem o problema e aprimoram a qualidade do modelo, definidos como atributos relevantes, dos atributos redundantes ou que não auxiliam na predição, sendo esses considerados como irrelevantes, segundo os autores Bolón-Canedo, Sánchez-Maroño and Alonso-Betanzos (2012).

Os autores também apresentam dois tipos de classificação de métodos de seleção de atributos segundo i) a sua abordagem e ii) a sua relação com a máquina aprendiz. Quanto a abordagem do método de seleção de atributos, a categorização é feita com relação ao modo que a avaliação dos atributos é realizada, seja de maneira individual, em que o nível de relevância de cada atributo é avaliado individualmente com a atribuição de pesos, ou levando em consideração o subconjunto sendo criado, o qual é avaliado segundo estratégias de busca. Já segundo a relação entre o método de seleção de atributos e a máquina aprendiz, o método pode ser classificado como:

Filters : quando o processo de seleção de atributos ocorre independente do processo de predição da máquina aprendiz

Wrappers : quando o processo de seleção de atributos faz parte do processo de otimização da predição da máquina aprendiz

Embedded : quando o processo de seleção de atributos acontece como parte integrada do treinamento da máquina aprendiz

Os autores também destacam as vantagens e desvantagens de cada método, uma vez que durante a seleção de atributos informações de cada atributo não são perdidas, porém se um conjunto de atributos é necessário para dar sentido as informações que antes estavam presentes no conjunto original, o qual era diverso, então algumas informações podem ser perdidas ao gerar esse subconjunto. Na extração de atributos a dimensionalidade do problema pode ser reduzida sem grandes perdas de informação. Porém, a combinação linear dos atributos originais pode gerar um espaço de atributos não interpretável e sem informação de quais atributos contribuem mais com a solução do problema.

Os autores então concluem a comparação entre os métodos de redução de dimensionalidade evidenciando como métodos de seleção de atributos *wrapper* geralmente apresentam melhor desempenho comparado com *filters* por serem otimizados pelo algoritmo de predição, no entanto eles não são adequados quando há alta dimensionalidade, uma vez que cada subconjunto deve ser avaliado durante o treinamento. Assim, *filters* são mais apropriados para problemas com alta dimensionalidade. Quando comparados a métodos de extração, os autores destacam como métodos de extração têm sido utilizados durante o pré-processamento para diminuir os efeitos de ruídos no processo de aprendizagem e são sensíveis aos tipos de dados. Já os métodos de seleção de atributos são mais efetivos em tarefas de eliminação de atributos irrelevantes e/ou atributos redundantes ao avaliar métricas de consistência e correlação dos atributos.

2.6 Sumário do Capítulo

Este capítulo apresentou a fundamentação teórica necessária para a compreensão deste trabalho. Inicialmente foram apontados os conceitos de Aprendizado de Máquina com foco em algoritmos de aprendizado supervisionado resolvendo problemas de classificação com abordagens padrão como *Naïve Bayes*, *Random Forest* e Máquina de Vetores de Suporte, bem como variações que lidam com dados não-rotulados, como métodos de Classificação de Classe Única e Aprendizado Positivo Não-Rotulado. Em seguida foi descrito como o método de validação cruzada é utilizado na avaliação da qualidade do classificador e quais métricas de avaliação de classificadores são utilizadas como precisão, *recall* e *F1-score*. Ao final, foi discorrido sobre técnicas de redução de dimensionalidade, como seleção e exportação de atributos, e a importância dessas técnicas no cenário de problemas de alta dimensionalidade.

3 TRABALHOS RELACIONADOS

Este capítulo revisa a literatura relacionada a este trabalho abordando dois tópicos: Fusão de Atributos, utilizada em reconhecimento de padrões em imagens e *Schema Matching*, operação que faz parte da área de integração de dados.

3.1 Fusão de Atributos

A fusão de atributos é uma técnica utilizada em classificação de imagens como uma estratégia para reduzir a dimensionalidade do espaço de atributos, ao fundir múltiplos conjuntos de atributos em único conjunto. Durante a classificação de imagens, o classificador deve ser capaz de reconhecer padrões *i.e.,* dígitos e letras escritos a mão, identificação de traços de humanos como face, digital e gênero (YANG et al., 2003; SUN et al., 2004; SCALZO et al., 2008; PEREZ et al., 2012; YANG; ZHANG, 2012). Essa estratégia de fusão tem como objetivo (i) a eliminação de atributos irrelevantes, que têm correlação fraca com a classe do problema e (ii) a remoção de atributos redundantes, que são atributos que possuem distribuição similar a de outros atributos.

Zhang et al. (2019) revisam a relação dos métodos de fusão com os métodos de seleção de atributos, por considerarem que a essência principal da fusão de atributos está no processo de seleção. Eles complementam, enfatizando a importância do coeficiente de correlação como a medida que fornece informação adicional durante o processo de Fusão de Atributos, visto que ela demonstra a força e direção de um relacionamento linear entre dois atributos. Da mesma forma, Perez et al. (2012) basearam-se no método de seleção de atributos *Mutual Information* para realizar a fusão de atributos de classificação de gênero, que permitiu a representação dos dados em um espaço de atributo de menor dimensão melhorando o desempenho do classificador como também o tempo de execução da classificação.

Yang et al. (2003) classificam as técnicas de fusão de atributos em (i) Serial, quando múltiplos conjuntos de atributos são combinados em um único conjunto, formando um super vetor ou vetor único, ou (ii) Paralela, quando múltiplos conjuntos de atributos são combinados por um vetor complexo ao invés de um vetor único real. Destacando como a utilização de métodos de extração de atributos na fusão paralela colabora com a redução da taxa de erros do classificador. Sun et al. (2009) também propuseram um método de fusão de atributos baseado em um método de extração de atributos cha-

mado *Locally Linear Embedding*, o qual superou os resultados nos experimentos quando comparado ao um método de concatenação de todos os atributos. Sun et al. (2004) também propôs um método de fusão de atributos baseado em extração de atributos, porém utilizando *Canonical Correlation Analysis* (CCA), que usa a correlação de dois grupos de atributos para realizar a fusão e eliminar informações redundantes entre os atributos. Essa redução permitiu que somente atributos essenciais das imagens fossem mantidos, o que influenciou a melhoria do desempenho do classificador. Yang and Zhang (2012) a fim de aprimorar a abordagem do método de extração *Local-preserving CCA*, que é não-supervisionado, criaram uma variante supervisionada usada para fundir diferentes atributos levando a maior confiabilidade no reconhecimento de digitais e veias dos dedos para identificação pessoal.

Este trabalho difere dos anteriormente apresentados nos seguintes pontos, (i) tipo de dado utilizado, visto que os trabalhos anteriores aplicaram a fusão de atributos em imagens, enquanto este trabalho aplicou em dados estruturados e (ii) o objetivo é fundir atributos duplicados, o que se distingue dos trabalhos de reconhecimento de padrões que buscam remover atributos redundantes, uma vez que dados estruturados redundantes podem apresentar mesma similaridade de distribuição, porém representam medidas diferentes. Por exemplo, temperatura em Celsius e Fahrenheit, as quais representam a mesma informação de temperatura, logo são redundantes em uma base de dados, porém indicam medidas diferentes que não podem serem fundidas em um único atributo, logo não são duplicadas.

3.2 *Schema Matching*

A tarefa de Fusão de Atributos tem similaridades com a de *Schema Matching*, que trata-se de uma das operações básicas do processo integração de dados. *Schema Matching* tem como finalidade identificar o relacionamento entre elementos de pares de esquema de origens heterogêneas, em que a entrada é dividida em duas partes, (i) dois esquemas diferentes, sendo um considerado como o original e outro o alvo, e (ii) regras de mapeamento, gerando um esquema que apresenta elementos incorporados de ambos e que deve ser verificado pelo usuário. A intervenção do usuário ainda é necessária durante o *Schema Matching*, uma vez que não se consegue produzir um mapeamento sem erros como esperado pelo usuário, sendo a principal razão para isso a enorme ambiguidade e heterogeneidade dos conceitos de descrição de dados, o que torna irreal a expectativa de

que uma ferramenta de mapeamento conseguirá mapear qualquer conceito presente em um conjunto (GAL, 2006; SUTANTA et al., 2016).

Bilke and Naumann (2005) listam diversas técnicas utilizadas em *schema matching i.e.,*: Correspondência Linguística, que é baseada no nome ou descrição do elemento usando tokenização e comparação de *strings* e *substrings*; Correspondência baseada em Instâncias, em que elementos de esquemas são considerados similares se suas instâncias forem similares segundo estatísticas, metadados ou classificadores; Correspondência baseada em Restrição, em que a comparação de tipos de dados, intervalos de valores, singularidade, elementos nulos e chaves estrangeiras ditam a similaridade entre os elementos.

Bilke and Naumann (2005) apresentaram uma abordagem baseada em instâncias, que testava tuplas da base de dados de modo a detectar as correspondências entre elementos dos esquema sendo mapeados. Avaliando seus resultados, a proposta demonstrou bom desempenho em identificar duplicatas em bases sintéticas. Porém, a avaliação da proposta ocorreu sobre esquemas com poucos atributos e necessita que o usuário tenha uma noção de quantas tuplas duplicadas ele espera encontrar. Já Do and Rahm (2002) propuseram a combinação de diversas técnicas em um sistema para algoritmos de mapeamento de correspondências chamado COMA, o qual suporta diferentes aplicações e múltiplos tipos de esquemas, como XML e bases de dados relacionais. O COMA permite que o usuário escolha as estratégias a serem usadas ou executar em modo automático com estratégias selecionadas de forma padrão na ferramenta. Avaliando o COMA foi verificado que a combinação dessas técnicas aprimorara a qualidade da correspondência entre os esquemas. Do and Rahm (2007) aprimoraram o sistema COMA permitindo trabalhar com esquemas maiores, criando o sistema COMA++, porém mesmo que esse sistema tenha superado outras abordagens genéricas ainda há várias áreas em que eles pretendem estender suas funcionalidades.

Este trabalho também lida com técnicas de verificação de similaridade entre atributos, as quais são similares às utilizadas na comparação de elementos dos esquemas. No entanto, este trabalho difere das abordagens na área de *schema matching*, uma vez que a fusão de atributos duplicados considera apenas elementos que estão presentes em um único esquema.

3.3 Sumário do Capítulo

Neste capítulo, foram analisados trabalhos relacionados a essa dissertação, os quais foram divididos em dois tópicos: Fusão de Atributos e *Schema Matching*. Baseado nesses tópicos, o próximo capítulo abordará a metodologia utilizada na fusão de atributos duplicados com foco em dados estruturados e considerando elementos de uma única base de dados.

4 METODOLOGIA

Esse capítulo apresenta a abordagem utilizada neste trabalho para resolver o problema de detectar atributos duplicados e realizar a fusão desses atributos. Começando pela criação de atributos de fusão que auxiliarão a definir se um par de atributos é duplicado ou não, finalizando o capítulo detalhando como o processo de fusão foi implementado.

4.1 Detecção e Fusão de Atributos Duplicados

A Tabela 4.1 mostra um exemplo de atributos duplicados e não duplicados. Esse exemplo é útil para ilustrar a diferença entre atributos *redundantes* e *duplicados*. Os atributos "heart rate" e "Hr_rate" são *duplicados*, *i.e.,* eles correspondem à mesma medida no mundo real e, desta forma, deveriam ser fundidos. Por outro lado, "temperature C" e "Temp F" (que registram a temperatura corporal do paciente em Celsius e Fahrenheit, respectivamente) não são duplicados, pois eles usam unidades de medidas diferentes. A fusão desses atributos uniria seus valores que são provenientes de diferentes distribuições, que por sua vez adicionaria ruído às tarefas de aprendizado. Esses dois atributos são *redundantes*, uma vez que eles são altamente correlacionados. Deste modo, algoritmos de seleção de atributos provavelmente descartarão um desses atributos. Este trabalho foca em atributos *duplicados* e não em atributos *redundantes*.

O problema abordado neste trabalho pode ser formalmente descrito do seguinte modo. Dado um conjunto de atributos originais $F = \{f_1, f_2, ..., f_n\}$, $\langle f_i, f_j \rangle$ é um par de atributos, onde $f_i \in F, f_j \in F$ e $i \neq j$. Cada atributo original f_i é associado com um nome l_i e um conjunto de valores $V_i = \{v_1, v_2, ..., v_m\}$. A tarefa de *identificar atributos duplicados* deve determinar se f_i e f_j são duplicados, *i.e.,* fazem referência a um mesmo atributo. Então, o *atributo de fusão* é o responsável pela junção dos atributos originais que foram identificados como duplicados. Cada par duplicado $\langle f_i, f_j \rangle$ é fundido para se tornar um novo atributo $f_h = f_i \oplus f_j$, onde $\oplus$ é uma função de agregação previamente definida e que depende do tipo de dados sendo fundido. Assim, V_j é fundido com V_i resultando em um novo conjunto de valores V_h, resultante da agregação pela função $\oplus$. Então um novo

Tabela 4.1: Exemplos de atributos duplicados e não duplicados

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

nome l_h é criado e associado a esse novo atributo resultante da fusão f_h.

A detecção de atributos duplicados depende de um conjunto de atributos que servem de entrada para um classificador, o qual cria um modelo a partir de um conjunto inicial de instâncias rotuladas. Uma vez que o modelo é criado, ele pode ser utilizado para classificar todos os pares de atributos da base de dados. Essa é uma tarefa desafiadora pelas seguintes razões:

- O número de pares de atributos originais a serem analisados pode ser muito grande, o que torna essa tarefa computacionalmente custosa. Para ajudar a atenuar esse problema, a quantidade de atributos a serem comparados deveria ser reduzida.
- O modelo necessitará aprender a partir de um pequeno conjunto de instâncias rotuladas, visto que o processo de rotular as instâncias exige uma mão-de-obra intensiva que precisa ser executada por um especialista do domínio.
- A quantidade de pares negativos (*i.e.,* pares de atributos que não são duplicados) é muito maior do que o número de pares positivos. Essa característica produz uma base de dados extremamente desbalanceada, que tipicamente representa desafios aos algoritmos de aprendizagem. Nesse sentido, o uso de classificadores que somente dependem de instâncias positivas (tais como as apresentadas na Seção 2.3) são desejáveis.
- Capturar a similaridade semântica dos atributos é muito difícil. Por exemplo, o par $\langle f_8, f_9 \rangle$ da Tabela 4.1 possui os nomes similares e a distribuição de valores dentro do mesmo intervalo. No entanto, eles não são duplicados, uma vez que um se refere a Dióxido de Carbono e o outro ao Oxigênio. Esse exemplo reforça a ideia de que usar diferentes fontes de evidências é necessária.

4.1.1 Detecção de Atributos Duplicados

A detecção de duplicatas é modelada como um problema de classificação binária. Para um dado par de atributos originais f_i e f_j, a tarefa dos classificadores é atribuir um rótulo indicando se os atributos f_i e f_j são duplicados. A decisão se algum par de atributos originais é duplicado é baseada em um conjunto de atributos de fusão que depende de evidências de diferentes origens. Essas evidências são explicadas a seguir.

Evidências baseadas nos nomes dos atributos. Em algumas situações, atributos duplicados têm nomes similares. Isso pode ser visto na Tabela 4.1 como f_3 e f_5, por exemplo,

têm nomes similares. A fim de avaliar o quão similares são os nomes desses dois atributos, as métricas de similaridade de *strings* são empregadas de modo semelhante ao que é feito na integração de dados. Entre muitas métricas disponíveis, este trabalho utiliza Levenshtein, Jaro-Winkler, e Soundex. O objetivo foi capturar diferentes aspectos de similaridade (sintáticos e fonético). Levenshtein, também conhecido como distância de edição, calcula quantas mudanças (inserções, remoções, ou substituições) são necessárias para transformar uma *string* em outra. Com a intenção de calcular a similaridade usando essa métrica, foi usada a variação chamada de Levenshtein Normalizada (Eq. 4.1). A similaridade Jaro-Winkler (Eq. 4.2 e Eq. 4.3) também é baseada em caracteres compartilhados. Essa métrica considera que as diferenças entre as primeiras letras das palavras sendo comparadas são mais significantes do que as diferenças entre as últimas letras das palavras. Essa métrica quantifica os caracteres que coincidem e leva a transposição em consideração. Ambas métricas resultam em valores entre 0 e 1. Quanto mais próximo de 1 esse valor é, mais semelhantes são as *strings* sendo comparadas. Diferente dessas duas métricas, Soundex é um algoritmo fonético que produz uma representação das *strings* de acordo com a sua pronúncia e então, a similaridade é calculada ao comparar as representações; se elas forem iguais, então a similaridade é 1; senão é 0.

$$NormLev_{a,b} = 1 - Lev_{a,b} Lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{se } \min(i,j) = 0 \\ min \begin{cases} Lev_{a,b}(i-1,j) + 1 \\ Lev_{a,b}(i,j-1) + 1 \\ Lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{senão.} \end{cases} \tag{4.1}$$

$$sim_j = \begin{cases} 0 & \text{se m} = 0 \\ \frac{1}{3}(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}) & \text{senão} \end{cases} \tag{4.2}$$

$$sim_w = sim_j + \ell p(1 - sim_j) \tag{4.3}$$

Evidências dos valores dos atributos. Se dois atributos são duplicados, então seus valores deveriam estar dentro de um mesmo intervalo (ou dentro de intervalos semelhantes). Essa evidência ajuda a distinguir entre atributos duplicados e redundantes. Como mostrado na Tabela 4.1, os valores dos atributos redundantes não precisam estar no mesmo intervalo. Para avaliar quão similar é a distribuição dos valores entre os atributos originais, é necessário conhecer os tipos dos seus dados (*i.e.,*, numérico ou categórico). Se ambos atributos têm diferentes tipos, então a sua de similaridade será zero. Se ambos atributos

originais são numéricos, então o teste Kolmogorov-Smirnov é utilizado (Eq 4.4). Para atributos categóricos, a similaridade cosseno é usada. O teste Kolmogorov-Smirnov computa se V_i e V_j possuem a mesma distribuição. Caso essa hipótese seja aceita, então o valor p resultante do teste deveria ser próximo de 1. Já o cosseno é comumente usado para medir a similaridade de dois documentos representados por vetores. Para calcular a similaridade cosseno, atributos originais f_i e f_j são transformados em vetores de incidências de termos em um espaço de d dimensões, onde d é o número de valores distintos em $V_i \cup V_j$. Os vetores armazenam a contagem da frequência de cada termo distinto.

$$D_{n,m} = \underset{x}{sup}|F_{1,n}(x) - F_{2,m}(x)| \tag{4.4}$$

Evidência de coocorrência. Se dois atributos originais f_i e f_j são duplicados, então seus valores tendem a coocorrer poucas vezes, *i.e.,* instâncias normalmente não têm ambos valores f_i e f_j simultâneos na base de dados. Isso pode ser visto na Tabela 4.1, como os atributos duplicados $\langle f_3, f_5 \rangle$ e $\langle f_4, f_6 \rangle$ que não coocorrem. Para quantificar a coocorrência dos atributos, é calculada a similaridade Jaccard entre eles, segundo a Eq. 4.5. A intersecção entre f_i e f_j é o número de instâncias na base de dados que possuem valores para ambos f_i e f_j. Quando a base de dados é analisada com dados temporais, a definição da intersecção pode ser adaptada para contabilizar quando os eventos ocorrem dentro do mesmo intervalo de tempo (*i.e.,* o mesmo dia, mesmo horário, *etc.*).

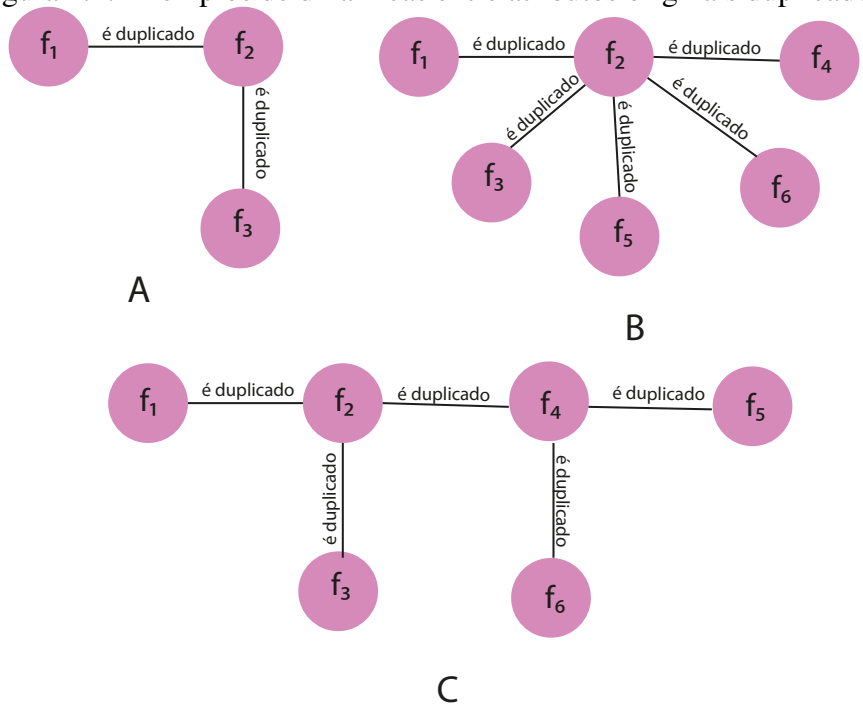
$$Jaccard(f_i, f_j) = \frac{f_i \cap f_j}{f_i \cup f_j} \tag{4.5}$$

4.1.2 Fusão de Atributo

Uma vez que os atributos duplicados tenham sido identificados, o próximo passo é organizar os atributos originais em grafos. Esse tipo de estrutura de dados permite representar a relação entre os atributos rotulados como duplicados aos agrupar em um mesmo grafo, que indica quais atributos duplicados serão fundidos em um único atributo posteriormente. Os vértices são representados pelo código de identificação dos atributos originais rotulados como duplicados e as arestas indicam a relação "é duplicado" entre dois atributos originais. Por exemplo, se os pares $\langle f_1, f_2 \rangle$ e $\langle f_2, f_3 \rangle$ são ambos identificados como duplicados, então os atributos f_1, f_2, e f_3 são colocados em um grafo G_A

para serem fundidos, como mostra a Figura 4.1-A. No entanto, ao seguir os rótulos dados pelos classificadores grafos maiores são gerados. A Figura 4.1-B exemplifica o grafo G_B que possui um atributo central, em que vários atributos originais foram rotulados como duplicados do atributo central f_2. Já a Figura 4.1-C exemplifica o grafo G_C que possui uma aresta central que liga dois grafos menores (G_{C_1} e G_{C_2}). Esses dois últimos exemplos gerariam uma cascata de fusões, transformando esses cinco atributos presentes nos grafos G_B e G_C nos atributos finais $Fused_B$ e $Fused_C$ respectivamente.

Figura 4.1: Exemplos de dinâmicas entre atributos originais duplicados.



Fonte: Elaborada pela autora.

Essa cascata de fusões pode ser indesejável caso os classificadores tenham rotulado erroneamente alguns dos pares presentes nesses conjuntos de atributos duplicados. A fim de evitar esse tipo de cascata de fusões, foi empregado um limiar θ que especifica um número máximo de atributos originais a serem fundidos em um único atributo. Nesse processo, atributos originais que tenham as probabilidades mais altas de serem duplicados terão maior probabilidade de serem inseridos em um grafo e os de probabilidade mais baixa terão menor probabilidade de serem inseridos em um grafo. Logo, a probabilidade de serem fundidos é proporcional a probabilidade do par ser duplicado. A probabilidade de um par de atributos originais ser rotulado como duplicado é calculado a partir de uma pontuação s, a qual representa a média dos valores dos atributos de fusão que são as evidências utilizadas na identificação dos atributos duplicados. Assim, os pares de atributos originais que foram identificados como duplicados são ordenados em ordem decrescente

a partir da pontuação s de cada par.

Após a ordenação decrescente dos pares duplicados por suas pontuações s, o grafo é gerado seguindo a ordem de pares. Desta forma, caso f_x possua uma pontuação s baixa, mesmo que o classificador o tenha rotulado como duplicado de f_y, onde $f_y \in G_i$ e G_i é um grafo, f_x pode não fazer parte de G_i, caso o tamanho de G_i já tenha atingido o valor de θ estabelecido no inicio da geração do grafo.

Com o propósito de juntar os valores dos atributos originais, funções de agregação (*i.e.,* média, mínimo, máximo, moda) são usadas. A partir de um conjunto de instâncias $X = \{x_1, x_2, ..., x_m\}$, o processamento dessas instâncias resulta em $x_p = (f_{p,1}[g], f_{p,2}[g], ..., f_{p,n}[g])$, onde g é a função agregadora usada para cada atributo original e $x_p \in X$.

Após a geração do grafo, é criada a lista dos atributos a serem fundidos, como indicado no pseudocódigo apresentado no Algoritmo 1. Essa lista contém o nome do atributo e o nome do conjunto de duplicados do qual ele fará parte, denominado como Fv. Outra lista a ser criada é a de conjuntos duplicados que armazena quais vértices dos grafos fazem parte de qual conjunto, denominada como Fa. Cada novo conjunto de atributos duplicados é nomeado como $Fused_x$ (linha 6) sendo x o $idFused$ atual e é produzido a partir dos vértices presente no grafo G. Cada vértice $v \in G$ é adicionado no conjunto $Fused_x$ junto com os vértices que compartilham arestas com v, não passando o limite de atributos definidos por θ (linhas 6-12). Caso o vértice v já faça parte de um conjunto de atributos duplicados, então os vértices que compartilham arestas com v são adicionados ao conjunto desde que não ultrapasse o limite θ (linhas 14-18).

Algorithm 1: Criação de Lista de Atributos a Serem Fundidos

Input: G - Grafo de atributos duplicados

Output: Fv - Lista de novos nomes dos atributos fundidos

Fa - Lista de conjuntos de atributos duplicados

```
1  Fv = { }
2  Fa = { }
3  idFused = 1
4  for v in G do
5      if v not in Fv then
6          Fv[v] = "Fused_"+ idFused
7          Fa["Fused_"+ idFused] = [v]
8          for vi in G[v] do
9              if Tamanho de Fa["Fused_"+ idFused] < θ then
10                 if vi not in Fv then
11                     Fv[vi] = "Fused_"+ idFused
12                     concatena vi à lista Fa["Fused_"+ idFused]
13     else
14         for vi in G[v] do
15             if Tamanho de Fa[v] < θ then
16                 if vi not in Fv then
17                     Fv[vi] = Fv[v]
18                     concatena vi à lista Fa[Fv[v]]
19 return Fv, Fa
```

Em seguida é realizada a fusão dos atributos segundo os conjuntos definidos anteriormente, de acordo com o pseudocódigo exibido no Algoritmo 2. A lista nDf é criada de forma que cada elemento represente a fusão dos atributos presentes em cada conjunto de duplicatas. Para isso, percorre-se os elementos de cada conjunto e utiliza-se funções de agregação para atributos numéricos como mínimo e máximo e a verificação quanto ao último valor de atributos categóricos. Para cada um dos elementos é então averiguado qual dos atributos possui o menor valor, o maior valor e qual dado aparece por último na base de dados (linhas 7-10). Após a agregação desses valores dos atributos presente nesse

conjunto de duplicatas sendo fundido, o novo atributo é adicionado a lista nDf segundo a identificação do conjunto da qual ela foi resultado (linha 11). Como alguns dos atributos originais da base de dados podem fazer parte de nenhum conjunto de duplicatas, eles então são inseridos diretamente na lista nDf (linhas 12-14).

Algorithm 2: Fusão de Atributos

Input : Fv - Lista de novos nomes dos atributos fundidos
Fa - Lista de conjuntos de atributos duplicados
Df - Dados dos atributos originais

Output: nDf - Dados dos atributos fundidos

```
1  nDf = { }
2  for v in Fa do
3      auxAtributos = Nulo
4      for p in Fa[i] do
5          if auxAtributos is Nulo then
6              auxAtributos = Df[p]
7          else
8              auxAtributos.set_min(Df.min)
9              auxAtributos.set_max(Df.max)
10             auxAtributos.set_last_value(Df.last_value, Df.last_visit)
11     nDf[v] = auxAtributos
12 for i in Df do
13     if i not in Fv then
14         nDf[i] = Df[i]
15 return nDf
```

4.2 Sumário do Capítulo

Neste capítulo foi apresentada a abordagem proposta para a detecção de atributos duplicados e a fusão dos mesmos. Para a detecção dos atributos duplicados foram estabelecidos atributos de fusão segundo evidências (i) baseadas nos nomes dos atributos, (ii) dos valores dos atributos e (iii) de coocorrência entre os atributos do par sendo investigado. Após a rotulação dos atributos como duplicados e não duplicados, é realizada a fusão dos atributos duplicados agrupando-os em grafos e agregando seus valores usando funções de agregação.

5 AVALIAÇÃO INTRÍNSECA

Esse capítulo apresenta a avaliação intrínseca, que tem como objetivo verificar a qualidade dos métodos de classificação na tarefa de rotular pares de atributos originais como duplicados ou não. A comparação feita utilizou RF (BREIMAN, 2001), SKC (um método PUL) (BAO et al., 2018) e OSVM (SCHÖLKOPF et al., 2000). Primeiro é descrito como os dados foram preparados para esses experimentos, os procedimentos de treinamento do modelo e finalizando com os resultados obtidos.

5.1 Materiais e Métodos

Dados. Os dados utilizados fazem parte da base de dados MIMIC-III v1.4 (JOHNSON et al., 2016), a qual contém mais de 40K pacientes e milhares de variáveis. Os pacientes foram admitidos em uma Unidade de Terapia Intensiva em um hospital em Boston-EUA. Cada paciente tem um número de eventos associados (*i.e.,* medidas de sinais vitais, valores de testes laboratoriais, *etc.*). Esses eventos são considerados como os atributos originais deste trabalho. Essa base de dados foi escolhida por ter eventos armazenados com diferentes identificadores, mas que representam o mesmo evento do mundo real. Este é um problema conhecido dessa base de dados, o que fez com que seus organizadores gerassem uma lista de eventos duplicados e a colocasse em um repositório[1]. Por exemplo, há seis atributos que referem-se à *Pressão Arterial Sistólica*. A base de dados tem 6.460 eventos (*i.e.,* atributos originais), então uma análise completa para determinar quais eventos são duplicados necessitaria de 20.282.570 comparações par a par.

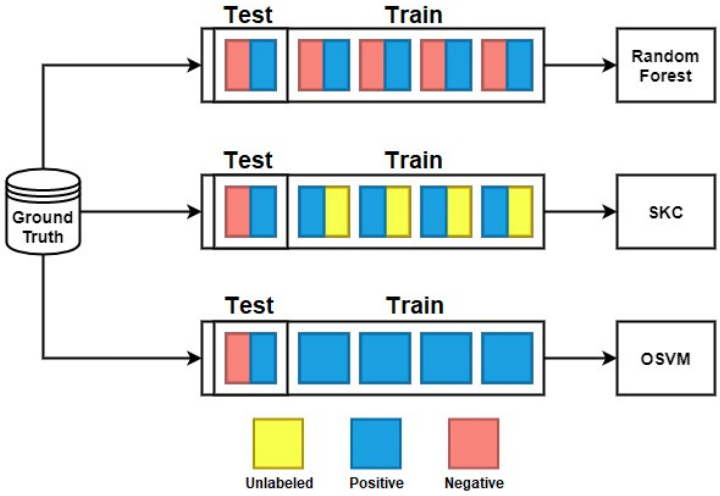
Geração do *Ground Truth*. Dado o alto número de atributos originais no MIMIC-III e consequentemente o enorme número de pares de atributos, é inviável ter um *ground truth* com rótulos para todos os pares possíveis. Assim, foi gerada uma amostra no seguinte formato. Foram usados os 125 pares de atributos que já tinham sido identificados como duplicados por outros pesquisadores e estão disponíveis no repositório do MIMIC-III. Além destes, foram adotados os seguintes procedimentos para identificar outros atributos duplicados. Primeiramente, foi gerado um sistema de pontuação para os atributos de fusão considerando a média dos atributos originais que foi calculada para todos os 20M pares possíveis. Esses pares foram ordenados em ordem decrescente segundo suas pon-

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

tuações. A suposição era de que pares de atributos com mais chance de serem duplicados estariam no topo dessa classificação. Então, 3 mil de pares do topo, meio e do final da classificação dos pares foram selecionados, resultando em 9 mil pares. Destes, foi gerada uma amostra de 2.290 pares que foram manualmente rotulados como *duplicados* ou *não duplicados* por uma especialista médica. Ao final deste processo, o *ground truth* possuía 338 pares rotulados como duplicados e 1.952 pares rotulados como não duplicados. O desbalanceamento entre essas classes (duplicados e não duplicados) não foi tratado. Essa amostra utilizada como *ground truth* equivale a 0,1% dos pares possíveis na base de dados original.

Ferramentas. A fim de calcular os valores para os atributos de fusão, foram utilizados diferentes ferramentas e bibliotecas. A biblioteca `Strsim`[2] foi utilizada para calcular o Levenshtein Normalizado e Jaro-Winkler. A biblioteca Jellyfish[3] foi usada para calcular Soundex. A estatística Kolmogorov-Smirnov foi computada usando SciPy[4]. Finalmente, Scikit-learn[5] foi empregada para computar a similaridade cosseno.

Figura 5.1: Configuração da base de dados separada em dados de treinamento e teste para cada classificador.



Procedimento Experimental. A validação cruzada foi executada usando 5-*folds* estratificados, de acordo com a Figura 5.1, em que as instâncias foram divididas em *folds* seguindo a proporção de instâncias de cada classe da base de dados. O classificador RF levou em consideração instâncias com ambos rótulos de duplicados e não duplicados. Para o OSVM, o treinamento utilizou apenas as instâncias duplicadas. As instâncias com rótulos como não duplicados foram utilizados apenas nos *folds* de teste. Para o SKC,

[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>

as instâncias não duplicadas foram consideradas como sendo instâncias sem rótulos e os *folds* foram divididos em *bags* positivas, que continham três instâncias rotuladas como duplicadas; e não rotuladas, que continham três instâncias sem rótulos. Durante o teste, as instâncias não duplicadas permaneceram com seus rótulos de não duplicadas e o *fold* foi dividido em bolsas com apenas uma instância.

Apenas 5-*folds* foram utilizados para permitir testar todas as configurações para cada classificador mais rapidamente. Foram testadas 243 configurações para o OSVM, 132 para o RF e 45 para o SKC, de modo a identificar os melhores parâmetros. A Tabela 5.1 mostra as melhores cinco configurações para cada classificador. Ao final, a melhor configuração foi escolhida, conforme definido a seguir. OSVM: RBF kernel, coeficiente kernel γ = 0,25; e o limite entre a fração de erros de treinamento e os vetores de suporte nu = 0,35. SKC: taxa de erros do treinamento (λ) = 0,0001 e a probabilidade da classe positiva (prior) = 0,15. RF: número de árvores n_trees = 250, a função que mede a qualidade do critério da divisão = Gini (Index), e max_depth = None.

Tabela 5.1: As cinco melhores configurações de parâmetros para cada classificador considerando Média (Méd) e Desvio Padrão (DP) de cada umas das métricas: F1, Precisão e *Recall*.

Classificador	Parâmetros	Méd F1	DP F1	Méd Precisão	DP Precisão	Méd *Recall*	DP *Recall*
OSVM	**nu=0.35, kernel=rbf gamma=0.25**	**0.705**	**0.040**	**0.795**	**0.142**	**0.648**	**0.053**
	nu=0.35, kernel=rbf gamma=0.2	0.707	0.038	0.798	0.137	0.648	0.053
	nu=0.35, kernel=rbf gamma=0.15	0.706	0.030	0.790	0.125	0.651	0.051
	nu=0.35, kernel=rbf gamma=0.05	0.708	0.034	0.790	0.125	0.654	0.052
	nu=0.35, kernel=rbf gamma=0.1	0.711	0.032	0.799	0.127	0.654	0.052
	nu=0.35, kernel=rbf gamma=0.01	0.713	0.029	0.801	0.119	0.654	0.052
SKC	**prior=0.15, lam=0.0001**	**0.682**	**0.026**	**0.620**	**0.043**	**0.766**	**0.073**
	prior=0.1, lam=1e-05	0.606	0.023	0.468	0.035	0.867	0.055
	prior=0.05, lam=1e-05	0.535	0.104	0.561	0.051	0.521	0.141
	prior=0.15, lam=1e-05	0.499	0.016	0.339	0.016	0.947	0.031
	prior=0.25, lam=0.001	0.447	0.021	0.298	0.018	0.902	0.050
RF	**n_trees=250, criterion=gini max_depth=None**	**0.910**	**0.013**	**0.958**	**0.023**	**0.867**	**0.022**
	n_trees=30, criterion=gini max_depth=10	0.910	0.008	0.958	0.013	0.867	0.022
	n_trees=200, criterion=gini max_depth=7	0.909	0.004	0.967	0.016	0.858	0.013
	n_trees=50, criterion=gini max_depth=7	0.909	0.005	0.967	0.011	0.858	0.013
	n_trees=100, criterion=entropy max_depth=10	0.909	0.006	0.955	0.025	0.867	0.015

Uma alternativa de *baseline* foi testada utilizando a ferramenta COMA (DO; RAHM, 2002). Por limitações computacionais como pouca memória para execução dos testes, uma base de dados com apenas 40 atributos originais foi gerada para esses testes iniciais. Por ser uma ferramenta de *schema matching* era necessário utilizar dois esquemas, para isso foi utilizado a base gerada e sua cópia. No entanto, ao executar o *schema matching*, aplicando todos os cenários sem instâncias disponíveis (\$*AllContextW*) quanto todos os cenários com 50 instâncias (\$*AllContextInstW*), a ferramenta apenas pareou cada atributo original com sua cópia. Por este motivo a ferramenta COMA foi descartada como *baseline* para este trabalho.

Tabela 5.2: Resultados da Avaliação Intrínseca – Fusão de Atributos

	OSVM	**SKC**	**RF**
Proporção TP/Recall	0,654	0,766	**0,867**
Precisão	0,781	0,615	**0,958**
Acurácia	0,922	0,895	**0,975**
PFP	0,032	0,083	**0,007**
PFN	0,346	0,234	**0,133**
F1	0,712	0,682	**0,910**

5.2 Resultados

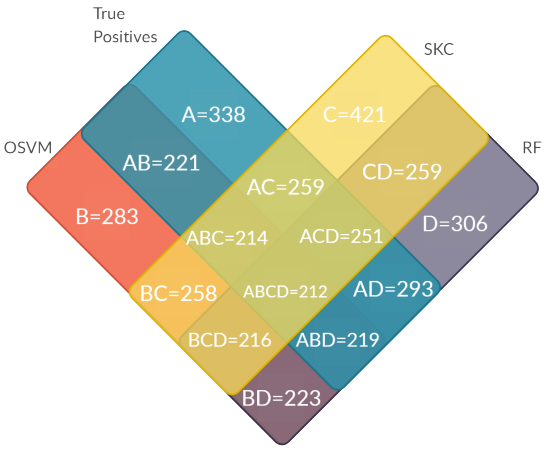
A Tabela 5.2 apresenta os resultados da avaliação intrínseca. RF teve o melhor desempenho em todas as métricas comparado com os demais classificadores. Esses resultados indicam que ter instâncias negativas ajudam na distinção entre atributos duplicados e não duplicados, mesmo existindo desequilíbrio entre as classes. Em uma comparação entre OSVM e SKC, o primeiro é melhor em termos de precisão, acurácia, proporção de falsos positivos e F1, enquanto o último é melhor em termos de proporção de verdadeiros positivo (*Recall*) e proporção de falsos negativos.

A fim de avaliar se os algoritmos concordam sobre as instâncias que eles rotularam como duplicatas/não duplicatas, foram gerados os diagramas de Venn apresentados na Figura 5.2. O diagrama mostra as intersecções das predições de cada método em comparação aos demais e com o *ground truth* (*i.e.,* o grupo *"True Positive"*). Todos os três algoritmos identificaram 63% (212/338) dos atributos duplicados. Já para os atributos não duplicados, a concordância entre os algoritmos foi mais alta, chegando a 91% (1767/1952). RF apresentou a maior concordância com o *ground truth* (AD = 293 para duplicatas e AD = 1939 para não duplicatas). SKC rotulou muito mais instâncias positivas (C = 421) do que as realmente presentes no *ground truth*, levando a uma alta proporção de FP. SKC, por outro lado, teve o maior número de instâncias rotuladas como não duplicadas, logo atingindo uma alta proporção de FN.

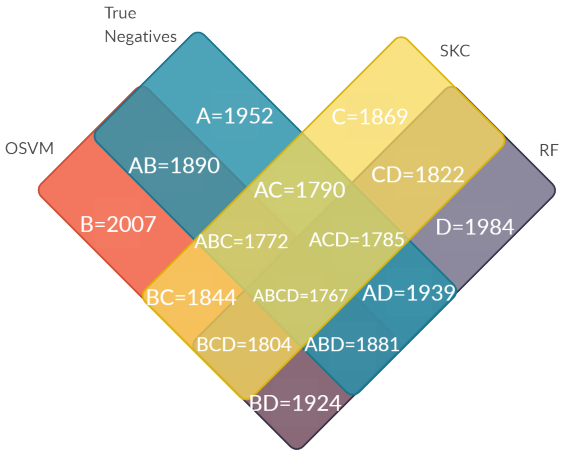
Uma vez que os modelos de classificação foram treinados com os dados anotados, eles foram utilizados para rotular a base de dados completa contendo todos os possíveis pares dos atributos originais, exceto os presentes no *ground truth*. Dos 20.860.280 pares da base completa, o número de instâncias rotuladas em cada classe pelos três modelos é mostrado na Tabela 5.3.

Foi realizada uma análise manual sobre os erros cometidos pelos classificadores. A Tabela 5.4 apresenta as instâncias rotuladas erroneamente por todos, o que demons-

Figura 5.2: Diagramas mostrando a intersecção entre as predições dos classificadores e o *ground truth* para as classes (a) positiva (duplicatas) e (b) negativas (não duplicatas) presentes na base de dados.



(a) Duplicatas



(b) Não Duplicatas

tra os casos que todos os classificadores tiveram dificuldades de rotular. Foi observado que as predições falsas positivas tiveram alta similaridade de nomes e distribuições dos atributos como os pares `temperature celsius` e `temperature f`. Assim como para as falsas negativas, em que a maioria ou apresentou alta similaridade e alta coocorrência como os pares `tpa#mg/hr` e `tpa#2 mg/hr` ou eles tiveram baixa similaridade entre seus nomes, zero coocorrência e alta similaridade de distribuição como os pares `glucose (70-105)` e `bloodglucose`.

Tabela 5.3: Número de instâncias (*i.e.,* pares de atributos originais) rotuladas como duplicadas e não duplicadas por cada classificador.

	OSVM	%OSVM	SKC	%SKC	RF	%RF
Duplicados	4.264.367	20,44%	2.984.812	14,31%	4.730.359	22,67%
Não Duplicados	16.595.913	79,56%	17.875.468	85,69%	16.129.921	77,33%

Também foi feita uma análise sobre os atributos de fusão buscando avaliar o peso de cada atributo ao mensurar o ganho de informação com relação a classe alvo. Assim, foi possível averiguar a importância de cada atributo para a distinção entre classes. A Tabela 5.5 apresenta o peso de cada atributo fusão, sendo o Jarowinkler o de maior importância e o Soundex o de menor importância para a distinção entre as classes. A partir dessa análise é possível notar como as evidências que mediam a distância de edição entre os nomes tiveram um peso maior no processo de identificar os atributos duplicados.

Tabela 5.5: Avaliação da importância dos atributos de fusão

Peso	Atributo
0,3272	Jarowinkler
0,3216	Levenshtein
0,0968	simDistr
0,0314	Jaccard
0	Soundex

5.3 Sumário do Capítulo

Neste capítulo foram apresentados os procedimentos e resultados da avaliação intrínseca, que teve como objetivo avaliar o desempenho na detecção de pares de atributos duplicados e não duplicados. Os resultados demonstraram que há melhoras na detecção quando instâncias negativas também são utilizadas no treinamento. Através da análise de erros foi possível observar os casos complicados em que todos os algoritmos erraram igualmente.

Tabela 5.4: Falsos negativos e falsos positivos dos três algoritmos.

Label #1	Label #2	Levenshtein	Soundex	jarowinkler	Jaccard	simDistr.	GT	OSVM	SKC	RF
temperature celsius	temperature f	0,63	1,00	0,95	0,00	0,96	-1	1	1	1
temperature celsius	temperature fahrenheit	0,59	1,00	0,91	0,00	0,97	-1	1	1	1
temperature fahrenheit	temperature c	0,55	1,00	0,92	0,00	0,98	-1	1	1	1
glucose (70-105)	bloodglucose	0,13	0,00	0,58	0,00	0,97	1	-1	-1	-1
radial abp	bp left arm [mean]	0,17	0,00	0,53	0,00	0,96	1	-1	-1	-1
manual bp mean(calc)	bp left leg [mean]	0,15	0,00	0,52	0,00	0,99	1	-1	-1	-1
bp right leg [mean]	l ulnar abp	0,16	0,00	0,51	0,00	0,97	1	-1	-1	-1
bp right leg [mean]	l fem art bp	0,16	0,00	0,49	0,00	0,97	1	-1	-1	-1
radial abp	bp rt. arm mean	0,20	0,00	0,47	0,00	0,95	1	-1	-1	-1
non invasive blood pressure mean	l ulnar abp	0,13	0,00	0,53	0,00	0,97	1	-1	-1	-1
l ulnar abp	bp rt. leg mean	0,13	0,00	0,53	0,00	0,95	1	-1	-1	-1
bp right leg [mean]	manual bp mean(calc)	0,10	0,00	0,53	0,00	0,98	1	-1	-1	-1
art bp mean	l fem art bp	0,08	0,00	0,56	0,00	0,97	1	-1	-1	-1
l ulnar abp	arterial pressure	0,12	0,00	0,52	0,00	0,96	1	-1	-1	-1
bp left leg [mean]	radial abp	0,17	0,00	0,46	0,00	0,97	1	-1	-1	-1
non invasive blood pressure mean	radial abp	0,13	0,00	0,50	0,00	0,97	1	-1	-1	-1
bp left leg [mean]	arterial pressure	0,11	0,00	0,50	0,00	0,97	1	-1	-1	-1
bp left leg [mean]	l ulnar abp	0,17	0,00	0,44	0,00	0,97	1	-1	-1	-1
radial abp	bp cuff [mean]	0,14	0,00	0,45	0,00	0,98	1	-1	-1	-1
bp rt. leg mean	radial abp	0,13	0,00	0,47	0,00	0,96	1	-1	-1	-1
arterial blood pressure mean	bp cuff [mean]	0,21	0,00	0,38	0,00	0,97	1	-1	-1	-1
arterial pressure	manual bp mean(calc)	0,15	0,00	0,44	0,00	0,97	1	-1	-1	-1
bp lt. leg mean	l ulnar abp	0,13	0,00	0,46	0,00	0,95	1	-1	-1	-1
arterial pressure	bp rt. leg mean	0,12	0,00	0,48	0,00	0,92	1	-1	-1	-1
l ulnar abp	nbp mean	0,09	0,00	0,44	0,00	0,97	1	-1	-1	-1
glucose (70-105)	fingerstick glucose	0,05	0,00	0,46	0,00	0,99	1	-1	-1	-1
nbp mean	arterial pressure	0,12	0,00	0,41	0,00	0,97	1	-1	-1	-1
bp lt. leg mean	radial abp	0,13	0,00	0,39	0,00	0,95	1	-1	-1	-1
arterial pressure	bp cuff [mean]	0,12	0,00	0,35	0,00	0,97	1	-1	-1	-1
bp lt. leg mean	arterial pressure	0,06	0,00	0,42	0,00	0,92	1	-1	-1	-1
art bp mean	l ulnar abp	0,00	0,00	0,40	0,00	0,97	1	-1	-1	-1
retic count (.0-1.5)	retic count(0.5-1.5)	0,85	1,00	0,96	1,00	1,00	1	-1	-1	-1
tank 1	tank 2	0,83	1,00	0,94	1,00	1,00	1	-1	-1	-1
tpa#1 mg/hr	tpa #2 mg/hr	0,83	1,00	0,91	1,00	1,00	1	-1	-1	-1
abg chloirde	abg chloride	0,83	1,00	0,99	0,92	1,00	1	-1	-1	-1
tcco2 [value]	tco2 [value]	0,92	1,00	0,98	0,80	0,96	1	-1	-1	-1
apache iv diagnosis	apache iv diagnosis choice 1	0,68	1,00	0,97	1,00	1,00	1	-1	-1	-1
base excess (cap)	base excess (other)	0,74	1,00	0,95	0,96	0,99	1	-1	-1	-1
pain level	pain level (rest)	0,59	1,00	0,94	0,99	1,00	1	-1	-1	-1

6 AVALIAÇÃO EXTRÍNSECA

Esse capítulo apresenta a avaliação extrínseca, que tem como objetivo avaliar o impacto da fusão de atributos tem sobre a tarefa de classificação. A tarefa usada nessa avaliação foi a de predição de mortalidade, por ser um tópico importante na informática médica.

6.1 Materiais e Métodos

Dados. Do mesmo modo que os atributos originais vêm do MIMIC-III (descrito na Seção 5.1), os dados usados para essa avaliação também são originários da mesma base de dados. Os dados são de pacientes adultos e as instâncias têm somente eventos e medições tomadas nas primeiras 24 horas da última estadia do paciente no hospital. Foi gerada uma amostra aleatória de 13.171 pacientes dentre os 32.507 na base de dados completa. A amostra tem 4.563 pacientes falecidos e 8.608 vivos.
Ferramentas. Weka (HALL et al., 2009) foi utilizado para executar os classificadores.
Procedimento Experimental. O classificador utilizado foi o Naïve Bayes com validação cruzada com 10-*folds*. O limite para a fusão dos atributos usado foi θ = 15. Experimentos foram executados utilizando os dados resultantes de cada um dos métodos de fusão de atributos e dois *baselines* - a base de dados original (sem fusão) e também usando uma base com apenas os pares fundidos identificados no *ground truth*.

6.2 Avaliação

Os resultados para a tarefa de predição de mortalidade são mostrados na Tabela 6.1. Percebe-se que todos os métodos de fusão de atributos tiveram um impacto negativo sobre a qualidade da predição. Um teste-t pareado foi aplicado para comparar os resultados de F1 (cuja distribuição não difere significativamente da distribuição normal) entre os dez *folds* para cada base de dados original e com a fusão. Os resultados indicaram que a redução é estatisticamente significativa usando $\alpha = 0.01$. Mesmo a fusão baseada nos atributos duplicados identificados no *ground truth* não apresentaram bons resultados. Esse fato nos leva à conclusão de que a hipótese que a fusão de atributos duplicados pode melhorar o poder preditivo dos dados *não pode ser validada* em nossos experimentos.

Tabela 6.1: Resultados da Avaliação Extrínseca - Predição de Mortalidade

Métrica	Original	*Ground Truth*	OSVM	SKC	RF
TP Rate/*Recall*	**0,773**	0,772	0,755	0,759	0,762
Precisão	**0,713**	0,712	0,699	0,700	0,700
Acurácia	**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
Tempo Treinamento (segundos)	2,615	2,463	**1,000**	1,107	1,098
#Features	6.108	6.010	2.063	2.119	2.328

O classificador ao utilizar as bases fundidas pode ter perdido atributos que foram importantes na distinção entre as classes na base original. Assim, ao usar as bases fundidas o seu desempenho apresentou uma queda no *Recall*, que indica a quantidade de instâncias rotuladas corretamente como positivas, demonstrando que houve confusão entre instâncias que na base original foram rotuladas corretamente.

Como esperado, como um resultado da redução no número de atributos, o treinamento do modelo foi muito mais rápido nas bases de dados fundidas. O tempo de treinamento foi reduzido para quase um terço do tempo como resultado da redução no número de atributos pela mesma proporção. É possível visualizar que o processo de detecção de duplicatas apresenta muitos falsos positivos uma vez que os atributos duplicados representam uma fração muito menor do que o conjunto de atributos originais.

6.3 Sumário do Capítulo

Neste capítulo foram apresentados os procedimentos e resultados da avaliação extrínseca, que teve como o objetivo avaliar o impacto que da fusão dos atributos sobre a tarefa de classificação. Essa tarefa é a de predição de mortalidade. Os resultados não indicaram melhoras no desempenho do classificador quando utilizado a base de dados com atributos fundidos. Mesmo havendo redução do tempo de execução, colaborado pela redução da dimensionalidade do problema.

7 CONCLUSÃO

A hipótese levantada por este trabalho foi a de que a fusão de atributos duplicados poderia resultar em um melhor poder de generalização dos classificadores.

Este trabalho propôs um método para facilitar o trabalho de especialistas na detecção de atributos duplicados, permitindo que especialistas possam rotular uma amostra pequena de instâncias que será utilizada no treinamento de um classificador, o qual rotulará as demais instâncias da base de dados. Esse método utilizou de evidências projetadas para avaliar a similaridade entre pares de atributos, as quais foram baseadas (i) nos nomes dos atributos, (ii) nos valores dos atributos e (iii) da coocorrência dos atributos. Após rotular os pares de atributos duplicados, o próximo procedimento para investigar a hipótese deste trabalho foi propor um método de fusão de atributos, que permitisse melhorar o desempenho de classificação na tarefa de predição de mortalidade.

Foram criados dois tipos de experimentos que permitissem avaliar o desempenho dos métodos criados. A avaliação intrínseca tinha foco na performance da detecção de atributos duplicados e não-duplicados, enquanto a avaliação extrínseca teve como foco a fusão dos atributos anteriormente rotulados. Mesmo que a avaliação intrínseca tenha apresentado bons resultados, em que o método de detecção utilizando Floresta Aleatória como classificador apresentou o melhor desempenho em rotular os pares de atributos, a avaliação extrínseca mostrou que a fusão foi muito agressiva, indicando que a proporção de atributos duplicados deve ser menor que a evidenciada na Tabela 5.3. Ainda que o tempo de treinamento tenha sido aprimorado pela fusão dos atributos duplicados na avaliação extrínseca, o poder de predição falhou em superar os valores da base de dados original sem fusão de atributos.

Um artigo foi escrito a partir desta dissertação, que será publicado no Simpósio Brasileiro de Banco de Dados (BARCELOS; RECAMONDE-MENDOZA; MOREIRA, 2020). Esse artigo contém a descrição sobre as fases de detecção e fusão de atributos duplicados, assim como os resultados de ambas avaliações.

Para trabalho futuros percebe-se a necessidade de maior investigação da fase de detecção de atributos duplicados e não-duplicados, considerando a sua influência sobre a fase de fusão dos atributos. Essa investigação deve ser focada em encontrar mais evidências que possam auxiliar no processo de rotular os pares de atributos duplicados, permitindo tornar esse processo mais acurado.

Além disso, trabalhos futuros deverão avançar nas limitações que foram encontra-

das nessa pesquisa. Primeiro, os testes foram feitos apenas sob uma única base de dados. Apesar de ser uma base com dados reais, uma base de dados médica desafiadora, experimentos em outras bases de dados são necessários para que a hipótese deste trabalho possa ser mais investigada, aprimorando o entendimento sobre este tópico no futuro. Também é preciso aplicar mais experimentos para explorar diferentes proporções de dados rotulados por um especialista para a identificação de atributos duplicados, visto que neste trabalho apenas 0,1% dos pares possíveis na base original foram rotulados. Assim como também a criação de novos atributos de fusão com o objetivo de reduzir os falsos positivos. Finalmente, embora a baixa dimensionalidade diminua a complexidade do modelo, o que por sua vez contribuiria para melhorar o poder de generalização, é necessário investigar melhor como o processo de fusão influenciou o desempenho do classificador na tarefa de predição de mortalidade e como essa influência pode ser utilizada para aprimorar a sua performance.

REFERÊNCIAS

ALPAYDM, E. **Introduction to Machine Learning**. second. London, England: The MIT Press, 2010.

BAO, H. et al. Convex formulation of multiple instance learning from positive and unlabeled bags. **Neural Networks**, v. 105, p. 132 – 141, 2018.

BARCELOS, H. C.; RECAMONDE-MENDOZA, M.; MOREIRA, V. P. Identifying and fusing duplicate features for data mining. **Brazilian Symposium on Databases**, 2020. A ser publicado.

BHATTACHARYA, I.; GETOOR, L. Iterative record linkage for cleaning and integration. In: **ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery**. [S.l.: s.n.], 2004. (DMKD), p. 11–18.

BILKE, A.; NAUMANN, F. Schema matching using duplicates. In: **International Conference on Data Engineering (ICDE)**. [S.l.: s.n.], 2005. p. 69–80.

BOLóN-CANEDO, V.; SáNCHEZ-MAROñO, N.; ALONSO-BETANZOS, A. A review of feature selection methods on synthetic data. **Knowledge and Information Systems volume**, v. 34, p. 483–519, mar 2012.

BREIMAN, L. Random forests. **Machine learning**, v. 45, n. 1, p. 5–32, 2001.

CHEN, M.; MAO, S.; LIU, Y. Big data: A survey. **Mobile Networks and Applications**, v. 19, p. 171–209, jan 2014.

CHRISTEN, P. 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**. [S.l.: s.n.], 2008. p. 1065–1068.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, Springer, London, England, v. 20, n. 3, p. 273–297, 1995.

DO, H.-H.; RAHM, E. Coma—a system for flexible combination of schema matching approaches. In: **Proceedings of the International Conference on Very Large Databases**. [S.l.: s.n.], 2002. p. 610–621.

DO, H.-H.; RAHM, E. Matching large schemas: Approaches and evaluation. **Information Systems**, Elsevier, v. 32, n. 6, p. 857–885, 2007.

GAL, A. Why is schema matching tough and what can we do about it? **ACM Sigmod Record**, ACM New York, NY, USA, v. 35, n. 4, p. 2–5, 2006.

HALL, M. et al. The weka data mining software: an update. **ACM SIGKDD explorations newsletter**, v. 11, n. 1, p. 10–18, 2009.

JOHNSON, A. E. et al. **MIMIC-III, a freely accessible critical care database**. 2016. 160035 p.

Khalid, S.; Khalil, T.; Nasreen, S. A survey of feature selection and feature extraction techniques in machine learning. In: **2014 Science and Information Conference**. [S.l.: s.n.], 2014. p. 372–378. ISSN null.

KHAN, S. S.; MADDEN, M. G. One-class classification: taxonomy of study and review of techniques. **The Knowledge Engineering Review**, Cambridge University Press, v. 29:3, n. doi:10.1017/S026988891300043X, p. 345–374, jan 2014.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. **Proceedings of the International Joint Conference on Artificial Intelligence**, 1995.

MADHAVAN, J.; BERNSTEIN, P. A.; RAHM, E. Generic schema matching with cupid. In: **VLDB**. [S.l.: s.n.], 2001. v. 1, p. 49–58.

MARSLAND, S. **Machine Learning: an Algorithmic Perspective**. second. London, England: Chapman and Hall CRC, 2014. (Machine Learning & Pattern Recognition Series).

MEISTER, D. et al. A study on data deduplication in hpc storage systems. In: **Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis**. [S.l.: s.n.], 2012. p. 1–11.

MITCHELL, T. M. **Machine Learning**. [S.l.]: McGraw-Hill, 1997.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. **Foundations of Machine Learning**. second. London, England: The MIT Press, 2012. (Adaptive Computation and Machine Learning series).

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PEREZ, C. et al. Gender classification from face images using mutual information and feature fusion. **International Journal of Optomechatronics**, 2012.

PLESSIS, M. C. D.; NIU, G.; SUGIYAMA, M. Convex formulation for learning from positive and unlabeled data. In: **International Conference on Machine Learning**. [S.l.]: Journal of Machine Learning Research (JMLR), 2015. v. 37, p. 1386–1394.

SCALZO, F. et al. Feature fusion hierarchies for gender classification. **International Conference on Pattern Recognition**, 2008.

SCHÖLKOPF, B. et al. Support vector method for novelty detection. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2000. p. 582–588.

STORER, M. W. et al. Secure data deduplication. In: **Proceedings of the International Workshop on Storage Security and Survivability**. New York, NY, USA: [s.n.], 2008. p. 1–10.

SUN, B.-Y. et al. Feature fusion using locally linear embedding for classification. **IEEE transactions on neural networks**, IEEE, v. 21, n. 1, p. 163–168, 2009.

SUN, Q.-S. et al. A new method of feature fusion and its application in image recognition. **Pattern Recognition**, 2004.

SUTANTA, E. et al. Survey: Models and prototypes of schema matching. **International Journal of Electrical and Computer Engineering**, 2016.

TAX, D. M.; DUIN, R. P. Uniform object generation for optimizing one-class classifiers. **Journal of machine learning research**, v. 2, n. Dec, p. 155–173, 2001.

YANG, J. et al. Feature fusion: parallel strategy vs. serial strategy. **Pattern recognition**, Elsevier, v. 36, n. 6, p. 1369–1381, 2003.

YANG, J.; ZHANG, X. Feature-level fusion of fingerprint and finger-vein for personal identification. **Pattern Recognition Letters**, Elsevier, v. 33, n. 5, p. 623–628, 2012.

Zhai, Y.; Ong, Y.; Tsang, I. W. The emerging "big dimensionality". **IEEE Computational Intelligence Magazine**, v. 9, n. 3, p. 14–26, Aug 2014. ISSN 1556-6048.

ZHANG, R. et al. Feature selection with multi-view data: A survey. **Information Fusion**, Elsevier, v. 50, p. 158–167, 2019.