UNIVERSIDADE FEDERAL DO RIO GRANDE
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

Dissertação de Mestrado

Uma Abordagem *Transfer-learning* para agrupamento de dados

Igor Avila Pereira

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Computação

Orientador: Prof. Dr. Leonardo Emmenrdorfer
Coorientadora: Prof[a]. Dr[a]. Karina Machado
Colaboradores: Prof. Msc. Eduardo Borges
Prof. Msc. André Vargas

Rio Grande, 2014

Dados de catalogação na fonte:
Biblioteca Central - FURG

Pereira, Igor Avila

Uma Abordagem *Transfer-learning* para agrupamento de dados / Igor Avila Pereira. – Rio Grande, 2014. – 74 f: gráf. – Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande. Centro de Ciências Computacionais. Rio Grande, 2014. – Orientador Leonardo Emmenrdorfer; Coorientadora Karina Machado.

1. Agrupamento de dados. 2. Machine Learning. 3. Aprendizado de máquina. 4. Clustering. I. Emmenrdorfer, Leonardo. II. Machado, Karina. III. Título.

CDD:

UNIVERSIDADE FEDERAL DO RIO GRANDE
Centro de Ciências Computacionais
Programa da Pós-Graduação em Computação
Curso de Mestrado em Engenharia de Computação

DISSERTAÇÃO DE MESTRADO

Uma Abordagem Transfer-learning para agrupamento de dados

IGOR AVILA PEREIRA

Banca examinadora:

Prof. Dr. Cleo Billa

Prof. Dr. Rodrigo Barros

Prof. Dr. Ricardo Rodrigues

Prof. Dr. Leonardo Emmenrdorfer

Orientador(a)

Karina Machado

Profa. Dra. Karina Machado

Coorientadora

AGRADECIMENTOS

Agradecimento especial à minha família, minha namorada Marina e meus amigos pela compreensão e auxílio durante o Trabalho.

Agradecimentos aos professores Leonardo Emmerdorfer, Eduardo Borges, Karina Machado e André Prisco pela ajuda dada no desenvolvimento do trabalho de mestrado. Obrigado pelas inúmeras revisões e reuniões.

Agradeço aos meus colegas de mestrado e aos funcionários do Centro de Ciências Computacionais - C3 que também me apoiaram durante o período de mestrado.

Pra quem gosta de nós é um prato cheio — HUMBERTO GESSINGER

RESUMO

PEREIRA, Igor Avila. **Uma Abordagem *Transfer-learning* para agrupamento de dados**. 2014. 74 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande, Rio Grande.

Na vida cotidiana, as pessoas agrupam objetos de forma inconsciente, sem saber exatamente como a seleção de objetos que compõem um determinado grupo é feita. Assim, é uma característica da aprendizagem humana considerar o conhecimento obtido a partir de grupos anteriores para gerar novos agrupamentos. No entanto, a maioria dos algoritmos de agrupamento não considera esse tipo de conhecimento prévio. Neste contexto, este trabalho propõe um novo algoritmo de agrupamento que aplica técnica de *Transfer Learning*, a fim de transferir o conhecimento de agrupamentos anteriores para um agrupamento futuro. Todo *dataset* incorporado é submetido a um processo de pré-processamento, que combina as instâncias em pares e mapeia seus atributos em novas características, a fim de que esteja em um domínio comum. Em uma etapa de treinamento, aplica-se um classificador que extrai o conhecimento presente nos *datasets*. Deste modo, o conhecimento extraído dos *datesets* anteriores é repassado à tarefa de agrupar um novo *dataset*. Assim, a transferência de conhecimento contribui para o processo de formação de grupos, melhorando o resultado final do agrupamento. Experimentos foram realizados com dois conjuntos de *datasets*. O primeiro contém 10 *datasets* e o segundo 5 *datasets*. No primeiro conjunto, há *datasets* com diferentes atributos e número de instâncias. Todavia, o segundo conjunto possui apenas *datasets* de mesmas dimensões (dois atributos) e número de instâncias diferentes. Para ambos, utilizou-se o procedimento de validação cruzada. A cada iteração, um *dataset* do conjunto era definido com sendo o *dataset* teste, ou seja, que se deseja agrupar e os demais eram utilizados como bases auxiliares de conhecimento. No conjunto de 10 *datasets* aplicou-se um classificador baseado em árvores de decisão e, para o segundo conjunto um classificador baseado em regressão logística.

Palavras-chave: Agrupamento de dados, Machine Learning, aprendizado de máquina, clustering.

ABSTRACT

PEREIRA, Igor Avila. **Transfer Learning approach to data clustering**. 2014. 74 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande, Rio Grande.

In everyday life, people group objects unconsciously, without knowing exactly how the selection of objects that comprise a given group is made. Thus, it is a characteristic of human learning to consider the knowledge gained from earlier groups to generate new clusters. However, most clustering algorithms do not consider this type of prior knowledge. In this context, this work proposes a new clustering algorithm applying the Learning Transfer techniques in order to transfer knowledge from previous partitions for a future partition. The datasets undergo a process of pre-processing that combines instances in pairs and maps their attributes into new features in order to be in a homogeneous domain. In a training phase, we apply a classifier that extracts the datasets knowledge. Thus, the knowledge extracted from previous datesets is passed to the task of clustering a new dataset. The knowledge transfer process contributes to the formation of groups, improving the outcome of the clustering algorithm. Experiments were performed with two sets of datasets. The first contains 10 datasets and the second 5 datasets. In the first set there are datasets with different attributes and number of instances. However, the second set has only datasets with the same dimensions (two attributes) and different number of instances. For both used the cross-validation procedure. At each iteration, a dataset was defined with the set and the test dataset, that is, a dataset we want cluster and others were used as auxiliary knowledge bases. In all 10 datasets a classifier based on decision trees was applied, and for the second set, a classifier based on logistic regression was employed.

Keywords: clustering, transfer learning, machine learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FURG	Universidade Federal do Rio Grande
C3	Centro de Ciências Computacionais
WEKA	Waikato Environment for Knowledge Analysis
KS	Método Kolmogorov-SmirNov
KL	Método Kullback-Leibler
ARFF	Attribute-Relation File Format
LCE	Learning from Cluster Examples

SUMÁRIO

1 INTRODUÇÃO

No cotidiano, pessoas agrupam objetos de modo inconsciente, não sabendo exatamente de que modo é feita a seleção de objetos que compõem um determinado grupo. Por exemplo, a necessidade de agrupar um conjunto de animais quem contém um ornitorrinco, duas aves e dois mamíferos. O ornitorrinco poderia ser agrupado juntamente as aves pelo fato de possuir um atributo em comum: o bico. Entretanto, analisando outro atributo, a produção de leite, ele poderia ser agrupado com os mamíferos.

É característica do aprendizado humano considerar, mesmo sem perceber, o conhecimento obtido de agrupamentos anteriores para gerar novos agrupamentos, mais precisamente as relações entre os objetos e os procedimentos adotados. Entretanto, na computação e, mais precisamente, nos algoritmos de agrupamento, não é isto o que ocorre; o conhecimento obtido em um agrupamento não é transferido para novos agrupamentos, podendo reduzir a capacidade de agrupar destes algoritmos. Logo, a transferência de conhecimento a partir de um conjuntos de dados previamente agrupados pode permitir que o algoritmo de agrupamento melhore a qualidade de seus resultados. Os algoritmos de agrupamento ou *clustering* possuem um objetivo em comum: separar um conjunto de objetos em grupos de tal maneira que os objetos que compartilham um mesmo grupo sejam similares e que sejam diferentes dos objetos contidos em outros grupos (Tan et al., 2005) (Barbakh et al., 2009).

Do ponto de vista prático, esta abordagem pode ser atrativa uma vez que a escolha de um algoritmo de agrupamento não é uma tarefa simples. Normalmente, usuários buscam alternativas algorítmicas conhecidas de agrupamento até escolher a mais adequada para seus dados. A existência de uma técnica de agrupamento que melhore seu desempenho através de tarefas anteriores é de grande valia.

Neste contexto, torna-se interessante construir um algoritmo de agrupamento capaz de utilizar o conhecimento adquirido em agrupamentos anteriores para novos agrupamentos, explorando a independência de domínio dos atributos e o aprendizado supervisionado (classificação). Assim, este trabalho de mestrado propõe uma nova técnica de agrupamento que aplica os conceitos de *Transfer Learning* (Pan and Yang, 2010) para transferir o conhecimento adquirido de agrupamentos anteriores para novos conjuntos de objetos.

Para permitir a transferência de aprendizado entre agrupamentos, elaborou-se um conjunto de atributos sintéticos, que representam as relações entre os objetos/instâncias e são definidos mediante um espaço comum de características.

A técnica de *Transfer Learning* já vem sendo aplicada com sucesso em tarefas onde já se sabe quais classes estão presentes no conjunto de dados (aprendizado supervisionado) (Pan and Yang, 2010). Neste trabalho, o aprendizado por transferência é aplicado em uma etapa de construção de conhecimento.

A Figura 1 resume o trabalho proposto. Todo novo *dataset* incorporado é submetido a um processo de transformação ou *pré-processamento* (etapa 1), que combina suas instâncias originais em pares e mapeia seus atributos em novos atributos (ou conjunto de atributos sintéticos). O processo de transformação faz com que, através do uso e cálculo dos atributos sintéticos, o *dataset* possua ao final do processo, independente da origem, os mesmos atributos dos demais *datasets* já previamente adicionados à arquitetura. Assim, em uma etapa de Treinamento (etapa 2), o foco passa a ser de extrair o conhecimento contido neste *dataset*. A extração é feita por meio de um método *classificador*. Deste modo, quando um novo *dataset* é adicionado (etapa 3 de Teste) e o objetivo é agrupá-lo, o conhecimento extraído do classificador é repassado à tarefa de agrupar.

Figura 1: Etapas do algoritmo de agrupamento proposto.

1.1 Objetivos Geral

Desenvolver e especificar um novo método de agrupamento de dados baseado na técnica *Transfer Learning*.

1.2 Objetivos Específicos

São objetivos específicos:

- Realizar um estudo sobre técnicas de agrupamento de dados encontradas na literatura.

- Elaborar atributos sintéticos com objetivo de mapear as instâncias de todo *dataset* submetido à proposta para um mesmo domínio.

- Agregar técnicas de classificação de dados à proposta de agrupamento (árvores de decisão e regressão logística).
- Utilizar bases de conhecimento auxiliares para agrupar novas bases de dados.
- Avaliar o desempenho da proposta por meio de índices de validação.
- Comparar o método de agrupamento de dados proposto com as técnicas já existentes na literatura.

1.3 Organização do Documento

O restante do texto está organizado da seguinte forma: o capítulo 2 apresenta a fundamentação teórica sobre agrupamento e *Transfer Learning*; trabalhos que descrevem algoritmos de agrupamento relacionados são analisados no capítulo 3; o método proposto é apresentado no capítulo 4; a descrição dos experimentos está presente no capítulo 5 e o capítulo 6 trata das conclusões e dos trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Um grupo é uma coleção de objetos (ou instâncias) de dados que são semelhantes entre si e diferentes de objetos existentes em outros grupos (Barbakh et al., 2009). Há muitos algoritmos de agrupamento na literatura (Xu and Wunsch, 2005), o que ocasiona uma difícil categorização dos métodos e das abordagens de agrupamento existentes. Por esta razão, as próximas seções apresentam uma breve descrição de algumas destas categorizações, a fim de facilitar a compreensão das características contempladas pelas técnicas da literatura e pelo trabalho proposto.

2.1 Agrupamento

Os algoritmos de agrupamento ou *clustering* são utilizados para separar objetos de uma base de dados em grupos, de tal maneira que os objetos que compartilham um mesmo grupo tenham alguma similaridade entre si e que sejam distintos dos objetos contidos em outros grupos (Tan et al., 2005). As próximas seções apresentam os tipos e as abordagens utilizadas em agrupamentos de dados.

2.1.1 Tipos de Agrupamento

É possível classificar um agrupamento como sendo do tipo particional quando, realizando uma simples divisão dos objetos dentro do conjunto de objetos, cada objeto permaneça em um único grupo. Ao passo que, se o objetivo for que os grupos tenham subgrupos, tem-se assim, um agrupamento do tipo hierárquico. Os agrupamentos hierárquicos organizam-se em forma de uma árvore. Ocasionalmente, as folhas destas árvores podem ser formadas de grupos de um único objeto (Tan et al., 2005). A Figura 2 ilustra um exemplo de agrupamento hierárquico.

Há outras formas de classificar um agrupamento. Um agrupamento pode ser exclusivo, interseccionado ou difuso (Tan et al., 2005). Agrupamentos exclusivos são agrupamentos onde cada objeto está presente exclusivamente em um único grupo. Classificamos um agrupamento como sendo interseccionado quando um mesmo objeto faz parte de mais de um grupo ao mesmo tempo. Em um agrupamento difuso, cada objeto pertencente a

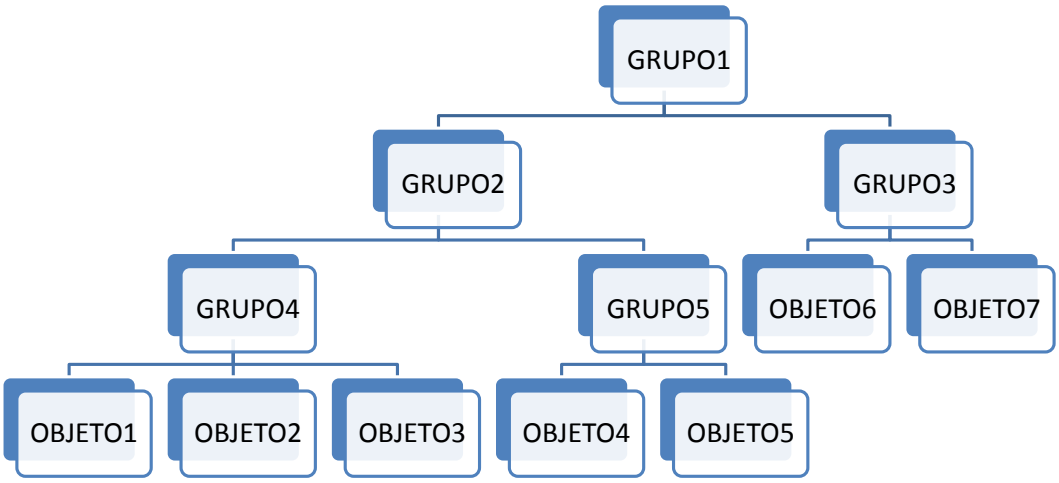


Figura 2: Agrupamento Hierárquico.

um grupo e possui um grau de pertinência por ser membro deste grupo (o peso tem valor que varia entre 0 e 1) (Tan et al., 2005). Na Figura 3 é possível observar um exemplo de agrupamento particional e exclusivo, já que cada objeto ficou estritamente em um grupo e nenhum grupo é composto de subgrupos. Na Figura 4 há um exemplo de agrupamento interseccionado, onde o objeto *objeto* está presente simultaneamente nos grupos 1, 2, 3 e 4.

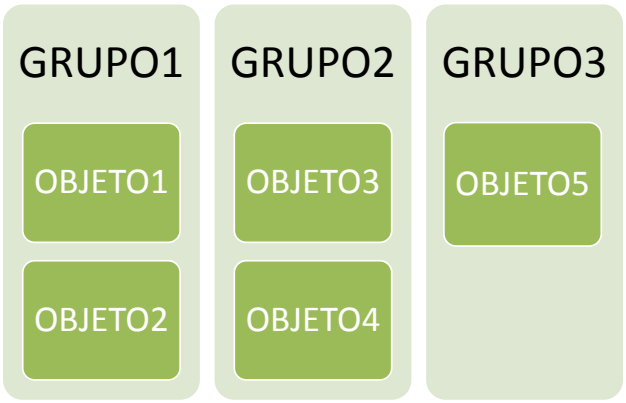


Figura 3: Exemplo de agrupamento particional e exclusivo.

Além das formas de classificação dos algoritmos de agrupamento já citadas, um agrupamento pode também ser completo ou parcial, sendo completo o agrupamento que atribui todos objetos do conjunto para algum grupo. O parcial é o inverso, isto é, podem existir objetos dentro do conjunto que não estejam em nenhum grupo ao final do agrupamento (Tan et al., 2005). A figura 5 ilustra um exemplo de agrupamento parcial (o objeto 5 não pertence a nenhum grupo).

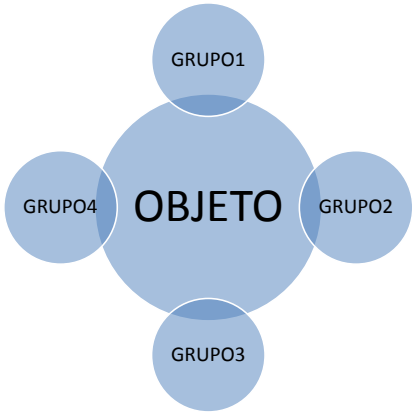


Figura 4: Agrupamento Interseccionado.

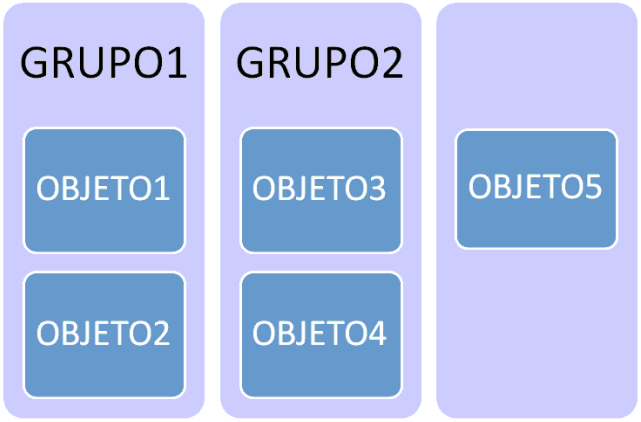


Figura 5: Agrupamento Parcial.

2.1.2 Abordagens de Agrupamento

As principais abordagens utilizadas para agrupamentos de dados são as abordagens baseadas em protótipos e baseadas em densidades. Na medida que for possível determinar protótipos a partir de um conjuntos de objetos, esses objetos reconhecidos como protótipos poderão agrupar outros objetos do conjunto mediante a existência de alguma semelhança entre eles (entre o protótipo e os outros objetos do conjunto). A determinação dos protótipos deve ser realizada a partir do cálculo do centróide ou da média do conjunto (Tan et al., 2005). Na Figura 6 há grupos compostos a partir da definição de 2 protótipos.

No entanto, há conjunto de objetos em que a determinação dos protótipos torna-se impraticável, em virtude da necessidade do cálculo da média (conjuntos de objetos compostos por dados categóricos) (Luz, 2003). Para estes conjuntos é preciso mudar a abordagem e, em vez de determinar os protótipos, determinar zonas (ou regiões) de alta densidade para realizar o agrupamento. Assim, grupos são formados a partir da divisão entre zonas de alta e de baixa densidade (Tan et al., 2005) (Figura 7).

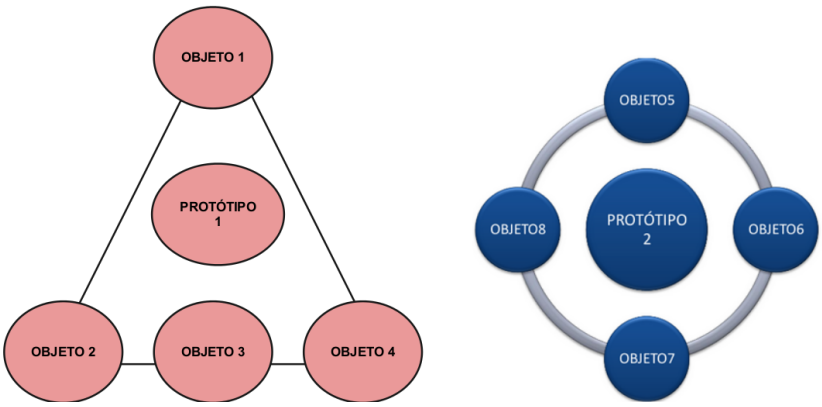


Figura 6: Abordagem de agrupamento baseada em protótipo.

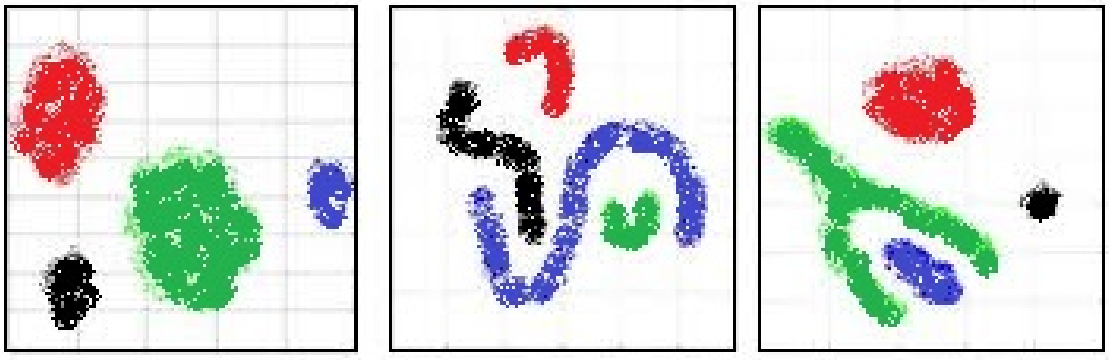


Figura 7: Abordagem de agrupamento por densidade.

2.1.3 Algoritmos de Agrupamento

Dentre a gama de técnicas existentes de agrupamento de dados, elencou-se um algoritmo representante de cada abordagem. Para a abordagem por protótipo selecionou-se o algoritmo K-means e para abordagem baseada em densidade o algoritmo DBSCAN. Além do K-means e do DBSCAN é possível citar outras técnicas como, por exemplo, OPTICS, DIANA e CLARANS (Tan et al., 2005).

2.1.3.1 K-means

O algoritmo K-means é definido como uma técnica de agrupamento de tipo particional baseada em protótipos que tenta encontrar o número de grupos especificado pelo usuário, representados pelos seus centróides (Hartigan and Wong, 1979) (Tan et al., 2005).

O algoritmo começa quando ocorre a seleção (aleatória ou através de uma heurística) dos K centróides iniciais (base de cada grupo), onde K é um parâmetro determinado pelo usuário. Cada objeto é atribuído ao centróide mais próximo, e cada conjunto de objetos atribuídos a um centróide é um grupo. O centróide de cada grupo é então atualizado baseado nos objetos atribuídos ao grupo. Devemos repetir os passos de atribuição e atualização

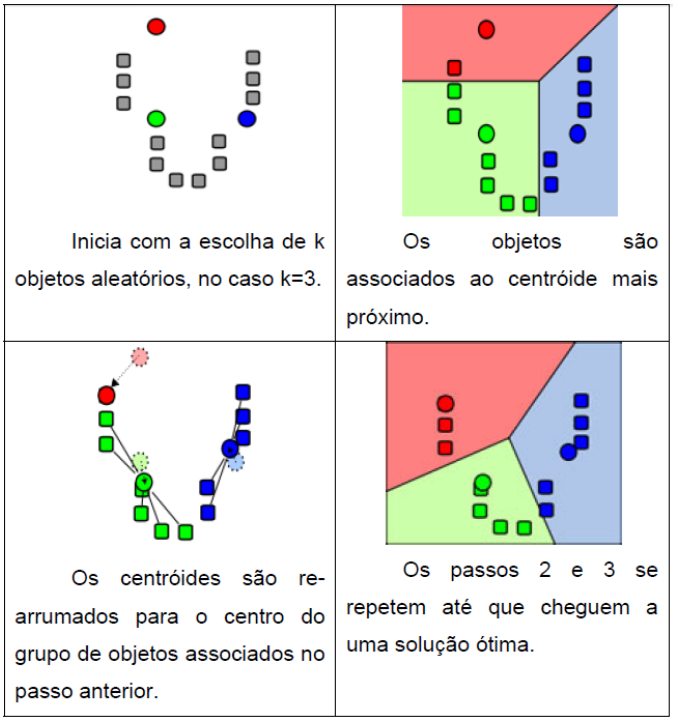


Figura 8: Exemplo K-means – Figura extraída de (Nunes, 2009).

até o momento que nenhum objeto mude de grupo, ou até que os centróides permaneçam os mesmos. Entretanto, o algoritmo K-means possui um problema: sua dependência da inicialização dos protótipos, que podem provocar, dependendo da escolha dos protótipos iniciais, grupos diferentes (Barbakh et al., 2009). A Figura 8 apresenta as etapas realizadas pelo algoritmo K-means em um *dataset* de exemplo.

2.1.3.2 DBSCAN

Agrupamentos baseados em densidade localizam regiões de alta densidade que estejam separadas entre si por regiões de baixa densidade (Ester et al., 1996).

No algoritmo DBSCAN, a densidade é avaliada para um determinado objeto do conjunto contando-se a quantidade de vizinhos deste objeto dentro de um raio Eps. Obtendo a quantidade de vizinhos de cada objeto, devemos rotulá-los como objeto central, limite ou ruído.

Um objeto é central se o número de vizinhos dentro de sua vizinhança, conforme uma função de distância e um parâmetro de distância especificado pelo usuário Eps, ultrapassar um limite $MinPts$, que é também um parâmetro especificado pelo usuário. Entretanto, um objeto é um objeto limite quando este não é um objeto central, mas fica dentro da vizinhança de um objeto central. Por último, um objeto é um objeto ruído quando não é nem objeto central nem objeto limite.

Resumidamente, o algoritmo DBSCAN pode ser descrito pelos seguintes passos:

1. Rotular todos os objetos (centro, limite ou ruído);
2. Eliminar objetos rotulados como objetos ruído;
3. Ligar todos os objetos de centro que estejam dentro da Eps uns dos outros;
4. Agrupar em mesmo grupo objetos centrais que estejam dentro do *Eps* de outros objetos centrais;
5. Atribuir cada objeto limite a um dos grupos dos seus objetos de centro associados;

A complexidade do algoritmo no caso base é *O(m * tempo para encontrar os objetos na vizinhança Eps)*, onde m é o numero de objetos. No pior caso $O(m^2)$.

Tendo em vista que o DBSCAN usa a abordagem baseada em densidade para agrupar, caracterizamos o DBSCAN como sendo um algoritmo relativamente imune a ruídos, que pode lidar com grupos de tamanhos e formas arbitrárias (Tan et al., 2005). Logo, o DBSCAN encontra muitos grupos que não poderiam ser encontrados pelo algoritmo K-means (seção 2.1.3.1). Todavia, o DBSCAN tem problemas quando os grupos tem densidades muito variadas. Ele também tem problemas com dados de alta dimensionalidade porque a densidade é mais difícil de ser definida para tais dados. Além disso, o DBSCAN pode ser custoso quando calcula os vizinhos mais próximos, já que requer o cálculo de proximidades entre pares (Tan et al., 2005). Outra característica negativa do DBSCAN é que, embora ele seja capaz de descobrir grupos em dados com ruído, ele é sensível aos dois parâmetros de entrada (o raio Eps e o número mínimo de elementos que constituem um grupo $MinPts$). Para descobrir os grupos, o DBSCAN depende que o usuário informe os parâmetros adequados (Barbakh et al., 2009), o que muitas vezes pode consumir muito tempo pela necessidade de executar os passos várias vezes até chegar a um resultado.

2.2 Validação de Agrupamentos

O trabalho de (Shao et al., 2007) argumenta sobre a impossibilidade de definir uma métrica universal de avaliação de grupos. Deste modo, dentro do universo de possibilidades, foram selecionadas apenas as métricas de avaliação de grupos utilizadas durante os experimentos (capítulo 5). É importante destacar que as métricas de avaliação utilizadas em métodos de classificação de dados também podem ser aplicadas para avaliação de agrupamentos (neste caso é preciso utilizar os rótulos dos grupos) (Tan et al., 2005). Além disso, as métricas de avaliação também podem ser divididas em dois tipos: métricas externas e métricas internas. As métricas externas avaliam os resultados dos agrupamentos comparando-os com classes pré-definidas, que podem ser rótulos previamente classificados por um usuário/sistema. Por outro lado, as métricas internas focam no próprio resultado do agrupamento, ou seja, não necessitam de outra fonte de informação externa.

2.2.1 DBI

A métrica de avaliação de agrupamentos DBI pode ser definida pela equação 1 (Davies and Bouldin, 1979). Na equação, k é o número de grupos, σ_i é a distância média de todas as instâncias (ou objetos) do grupo i para o seu centróide c_i, σ_j é a distância média de todas as instâncias do grupo j para seu centróide c_j e $d(c_i, c_j)$ é a distância entre os centróides c_i e c_j. Quanto menor o valor do DBI para agrupamento, mais compactos são seus grupos e o centróide de cada grupo está mais distante dos demais.

$$DBI = \frac{1}{n}\sum_{i=1}^{n} \max_{j:i\neq j}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right) \tag{1}$$

A Tabela 1 e a Figura 9 apresenta o *dataset-base* utilizado para exemplificar o cálculo da métrica DBI. Um método de agrupamento determinou para o *dataset* um agrupamento com 3 grupos: o grupo 1 (vermelho), o grupo 2 (azul) e o grupo 3 (verde).

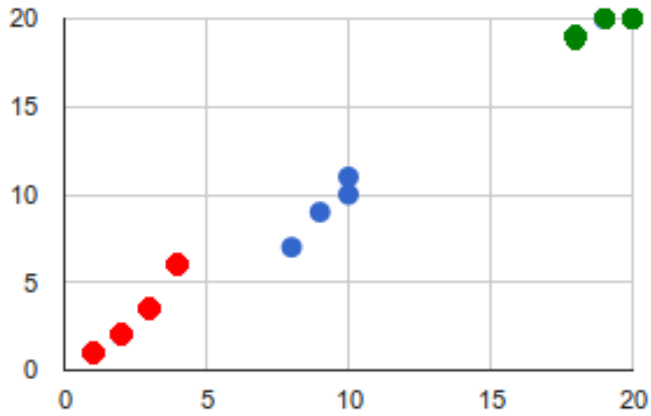


Figura 9: Dataset - Exemplo DBI.

Tabela 1: Dataset - Exemplo DBI.

Instância	Atributo x	Atributo y
0	1	1
1	2	2
2	4	6
3	3	3,5
4	10	11
5	10	10
6	8	7
7	9	9
8	20	20
9	18	19
10	19	20

O passo (1) calcula, para cada grupo encontrado no processo de agrupamento, seu centróide (Tabela 2) e a distância euclidiana de seu centróide em relação aos centróides dos outros grupos. No passo (2) (Tabela 3), para cada instância, é calculado a distância euclidiana em relação ao centróide que representa seu grupo. Ao final, calcula-se o valor da distância média, obtida por meio da razão do somatório das distâncias obtidas (para cada instância) comparada ao seu centróide pela quantidade de instâncias pertencentes ao grupo.

Tabela 2: DBI - (1.1) Centróides.

Centróide	x	y
Grupo 1	2,5	3,125
Grupo 2	9,25	9,25
Grupo 3	19	19,67

Tabela 3: DBI - (1.2) Distância entre Centróides.

Centróides	Distância
Grupo 1 - Grupo 2	9,12
Grupo 1 - Grupo 3	23,37
Grupo 2 - Grupo 3	14,27

Tabela 4: DBI (2) - Grupo 1 Distância Média de todas as instâncias em relação ao seu centróide.

Instância	Distância
0	2,60108151
1	1,24
2	3,25
3	0,625
Distância Média	**1,92499229**

O passo (3) é subdividido em duas etapas. Na primeira etapa (passo 3.1 - Tabela 7), pares de grupos dentre os grupos encontrados são relacionados por meio de um cálculo que soma as distâncias médias das instâncias destes grupos em relação aos seus centróides e divide o valor pela distância entre os centróides pertencentes aos grupos. O maior valor obtido é utilizado para o passo seguinte (passo 3.2 - Tabela 8).

Tabela 5: DBI (2) - Dist. Média de Todas as instâncias em relação ao seu centróide - Grupo 2.

Instância	Distância
4	1,90
5	1,06
6	2,58
7	0,36
Distância Média	**1,48**

Tabela 6: DBI (2) - Dist. Média de Todas as instâncias em relação ao seu centróide - Grupo 3.

Instância	Distância
8	1,05
9	1,20
10	0,34
Distância Média	**0,87**

Tabela 7: DBI (3.1) - DBI - Cálculo Máximo e Resultado Final.

Grupos	Cálculo
1-2	$\frac{distMediaGrupoI + distMediaGrupoJ}{d(c_I, c_J)}$ = 0,3728
1-3	0,119332
2-3	0,163733

Tabela 8: DBI (3.2) - DBI - Cálculo Máximo.

max	resultado
max1	1-2 = 0,3728
max2	1-2 = 0,3728
max3	2-3 = 0,163733
Somatório	**0,909342**

O resultado final é dado pelo *somatório* dividido pela quantidade de grupos encontrados. No exemplo o resultado final é dado por $(1/3)*$**Somatório**$= (1/3) * (0,909342) = 0,303114$.

2.2.2 Silhouette

A métrica de validação de agrupamentos *Silhouette* (Rousseeuw, 1987) define a qualidade dos agrupamentos tendo por base a proximidade entre os objetos de um determinado grupo e a proximidade desses objetos ao grupo mais próximo. Os valores variam entre $[-1, 1]$. Quanto mais próximo de 1, melhor é a alocação do objeto no grupo. Em contrapartida, quanto mais próximo de -1, pior é alocação do objeto. *Silhouette* é obtida pela equação 2.

$$s(v_i) = \frac{d(v_i, C_h) - d(v_i, C_j)}{\max(d(v_i, C_j), d(v_i, C_h))} \tag{2}$$

onde v_i representa o objeto pertencente ao grupo C_j e C_h é o grupo mais próximo a v_i. Além disso, para realizar o cálculo da métrica para todos os objetos/instâncias que fazem parte do agrupamento, deve-se calcular, primeiramente, a métrica para o grupo (S_i) e, logo em seguida, para todo o agrupamento (GS). Esses valores são obtidos pelas equações 3 e 4. Nas equações, N é a quantidade de objetos do grupo J e K é o número total de grupos.

$$S_j = \frac{\sum_{i=1}^{N_j} s(v_i)}{N_j} \tag{3}$$

$$GS = \frac{\sum_{j=1}^{K} S_j}{K} \tag{4}$$

2.2.3 MicroF e MacroF

MicroF e MacroF são métricas externas de validação de agrupamento que medem a qualidade dos grupos em função das classes conhecidas a *priori*.

A $macroF$ é uma média aritmética simples de todas as F (média harmônica de *precision* e *recall*), que atribui a cada classe uma mesma importância. Ela mede a qualidade geral do resultado com N classes. A $microF$ é uma média que atribuí a mesma importância para cada instância. Ela também mede a qualidade geral do resultado com N classes, entretanto é uma métrica mais confiável quando as classes são bastante desbalanceadas. Para duas classes A e B, $microF_h1 = (|g1 \cap cA| + |g2 \cap cB|)/(|A| + |B|)$, ou seja, a porcentagem de acertos no total. É a mesma acurácia do modelo como um todo.

Para calcular tanto a $microF$ quanto a $macroF$ é necessário definir todas as possíveis hipóteses, ou seja, com posse das classes e dos grupos, elenca-se todas as hipóteses onde grupos do agrupamento podem representar, parcialmente ou em sua totalidade, as classes do *dataset* conhecidas a $priori$. O resultado final é dado pelo valor que, dentre o conjunto de hipóteses, maximiza o valor de *microF* e *macroF*.

Por exemplo, para 2 grupos $G = \{g1, g2\}$ e 3 classes $C = \{cA, cB, cC\}$, o número de hipóteses é calculado a partir do arranjo das classes nos grupos A_G^C, que é igual a $A_2^3 = 3!/(3-2)! = 6$ hipóteses que são:

- $h1 : cA = g1$ e $cB = g2$
- $h2 : cA = g1$ e $cC = g2$
- $h3 : cB = g1$ e $cA = g2$

- $h4 : cB = g1$ e $cC = g2$
- $h5 : cC = g1$ e $cA = g2$
- $h6 : cC = g1$ e $cB = g2$

Além das hipóteses, as métricas precisam responder questões similares as questões abaixo:

- Quantos elementos de $g1$ pertencem a $cA(|g1 \cap cA|)$?
- Qual a quantidade de elementos em $g1(|g1|)$?
- Qual a quantidade de elementos em $A(|A|)$?

As figuras 10 e 11 exemplificam o cálculo das métricas *microF1* e *macroF1* respectivamente para agrupamentos com duas classes (conhecidas a *priori*) e com 2 grupos resultantes.

Exemplo: *Dataset*
- Possui 2 classes = {cA, cB}
- Encontrou 2 grupos = {g1, g2}

Fórmula
- |g1 e cA| = Quantidade de instâncias que estão em g1 e são da classe cA
- |g2 e cB| = Quantidade de instâncias que estão em g2 e são da classe cB
- micro F1_h1 =
  - (|g1 e cA| + |g2 e cB|) / (total de instâncias do *dataset*)
- micro F1_h2 =
  - (|g1 e cB| + |g2 e cA|) / (total de instâncias do *dataset*)
- o maior valor de microF é a hipótese escolhida.

Figura 10: Exemplo *microF*.

Além do que já foi dito, é preciso também conhecer duas outras métricas que servem de pré-requisito para o cálculo de *microF* e de *macroF*: precisão (*precision*) e revocação (*recall*). Precisão e Revocação são métricas comumente utilizadas em sistemas de recuperação de informação. Precisão é definida como sendo a proporção dos documentos/instâncias/objetos recuperados que são relevantes para uma dada consulta/-grupo em relação ao total recuperado e revocação como sendo a razão entre o número de documentos/instâncias/objetos recuperados que são relevantes para uma consulta/grupo e o total do conjunto que é considerado relevante (Tan et al., 2005).

Exemplo:
- dataset
  - Possui 2 classes = {cA, cB}
  - Entretanto, encontrou 2 grupos = {g1, g2}

Fórmula
- numeradorMacroF1 =
  - Somatório de medidaF's
    - ((2 *precisao(gX, classeY) * revocação(grupoX, classeY)) / (precisao(gX, classeY) + revocação(gX, classeY)));
- macro F1 =
  - numeradorMacroF1 / Qtde de classes

Figura 11: Exemplo *macroF*.

2.3 Transfer Learning

É possível definir a técnica de *Transfer Learning* como sendo a habilidade de um sistema reconhecer e aplicar conhecimentos aprendidos em tarefas anteriores para a solução de novos problemas (Pan and Yang, 2010). Assim, o objetivo do *Transfer Learning* consiste em extrair o conhecimento a partir de uma ou mais tarefas fontes e aplicar o conhecimento para uma tarefa de destino (figura 12).

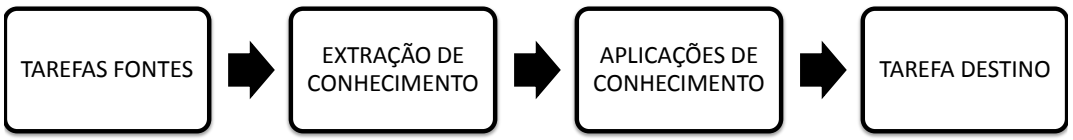


Figura 12: *Transfer Learning*.

Tradicionalmente, as técnicas de *Machine Learning* tentam aprender cada tarefa a partir do zero, enquanto o *Transfer Learning* tenta transferir o conhecimento de algumas tarefas anteriores para uma tarefa de destino. Um exemplo da técnica de *Transfer Learning* é transferir o conhecimento adquirido ao tocar violão para o aprendizado de guitarra.

Além de aplicar os conhecimentos aprendidos em tarefas anteriores para novos problemas, o *Transfer Learning* é norteado por três importantes questões (Pan and Yang, 2010):

- O que transferir?
- Como transferir?
- Quando transferir?

O que transferir? corresponde a determinar que parte do conhecimento pode ser transferido através das tarefas fontes e destino. As tarefas podem ter duas formas de

conhecimento: o específico e o comum. O último conhecimento é o mais indicado a ser transferido, já que ajuda a melhorar o desempenho e por ter relação direta com a tarefa de destino. Assim, deve-se invocar algoritmos de aprendizagem que são responsáveis por realizar a transferência destes conhecimentos e responder a questão de *Como transferir?* (Pan and Yang, 2010).

A questão de *Quando transferir?* trata das situações de quando a transferência de competências deve ser feita. Trata também de conhecer em quais situações o conhecimento não deve ser transferido. Em alguns casos, quando as tarefas fontes e destino não são relacionadas, a transferência por força bruta pode ser sem sucesso. No pior caso, pode prejudicar o desempenho de aprendizagem da tarefa de destino, sendo uma situação de transferência negativa. A maioria dos trabalhos atuais focam em *O que transferir?* e *Como transferir?* assumindo que as tarefas fontes e a tarefa destino são relacionadas uma com a outra. No entanto, como evitar uma transferência negativa é uma questão em aberto que irá atrair mais e mais atenção no futuro (Pan and Yang, 2010).

2.4 Kolmogorov - Smirnov e Kullback - Leibler

Os métodos *Kolmogorov - Smirnov - KS* e *Kullback - Leibler - KL* calculam a distância entre duas curvas acumuladas. Para o trabalho de mestrado os métodos *KS* e *KL* foram utilizados durante o processamento dos *Atributos Sintéticos* desenvolvidos (seção 4.2). As curvas acumuladas funcionam como entrada para as métricas. *KS* e *KL* calculam, cada um a sua maneira, um valor de distância existente entre duas curvas. A Figura 13 ilustra a distância encontrada entre duas curvas acumuladas quaisquer.

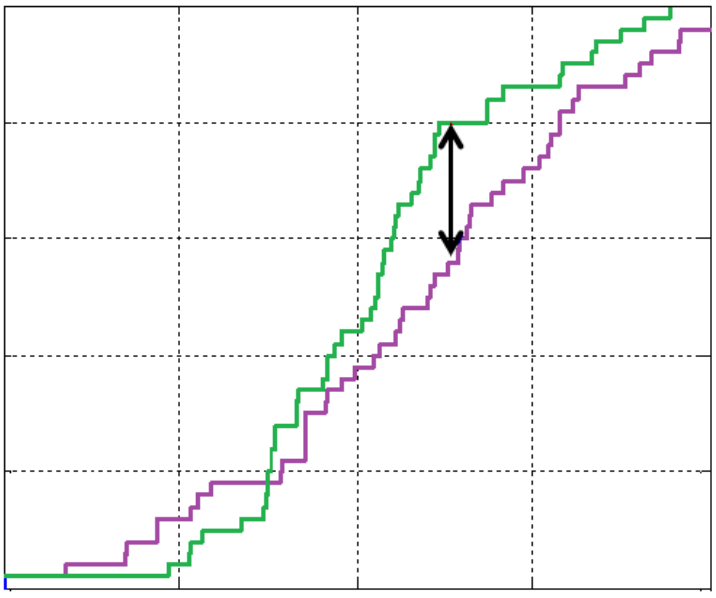


Figura 13: Métodos *Kolmogorov - Smirnov* e *Kullback - Leibler*.

Tabela 9: Exemplo de curvas utilizadas nos métodos Kolmogorov - Smirnov e Kullback - Leibler.

Vetor 1	1	10	20	15	7	8	16	13	5	6
Vetor 2	5	4	1	15	12	11	8	9	10	11

Para exemplificar o funcionamento das métricas, elencou-se 2 vetores de pontos expressos pela Tabela 9. Independe da origem os dados, os mesmos devem passar pela função *CDF* (Função de Distribuição Acumulada). A função *CDF* transforma os dados (vetores) em distribuições acumuladas. O pseudocódigo da função é ilustrado pela Figura 14. Nesse código, duas variáveis *valor mínimo* e *valor máximo* são inicializadas e incrementadas a cada iteração com objetivo de restringir o intervalo de valores. Assim, enquanto *valor máximo* não ultrapassar a distância euclidiana presente entre os vetores *1* e *2* (no exemplo *d(vet1,vet2) = 24.02*), armazena-se em uma outra variável o número de elementos compreendidos entre as variáveis *valor mínimo* e *valor máximo*.

valor mínimo = 0;
valor máximo = d(vet1,vet2)/**10**;
Enquanto (**valor máximo** <= d(vet1, vet2)){
- Obter a QTDE de elementos de **vetor1/vetor 2** que estão entre:
  - **valor mínimo**
  - **valor máximo**
- **valor mínimo** = **valor máximo;**
- **valor máximo**+= d(vet1,vet2)/10;

}

Figura 14: Função *CDF*.

Segundo o Teorema de Nyquist (Farrow et al., 2011), a frequência de amostragem de um sinal analógico, para que possa posteriormente ser reconstituído com o mínimo de perda de informação, deve ser igual ou maior a duas vezes a maior frequência do espectro desse sinal. Para a abordagem *Trasnfer Learning* de agrupamento de dados, verificou-se que a frequência máxima é $5Hz$ (Figura 15). Assim, definiu-se 10 para incremento da variável *valor máximo* e do número de elementos resultantes retornados da função *CDF*. Na Tabela 10, observa-se os valores *mínimo* e *máximo*, os elementos compreendidos entre as variáveis e a quantidade encontrada de cada intervalo para o $vetor1$. Assim, o vetor quantidade é transformado em vetor-quantidade-acumulado. A Tabela 11 mostra o vetor 1 original, o vetor quantidade encontrado, o vetor quantidade-acumulado e o vetor 1 final para a função *CDF*. Da mesma forma, a Tabela 12 refere-se ao vetor 2.

O próximo passo é encaminhar os vetores resultantes para os métodos *KS* e *KL*. O método *KS* retorna a maior diferença absoluta entre os valores dos dois vetores resultantes da função *CDF*. Para o exemplo, o resultado seria expresso na Tabela 13. O método *KL* pode ser ilustrado pela Figura 16 e pela Tabela 14.

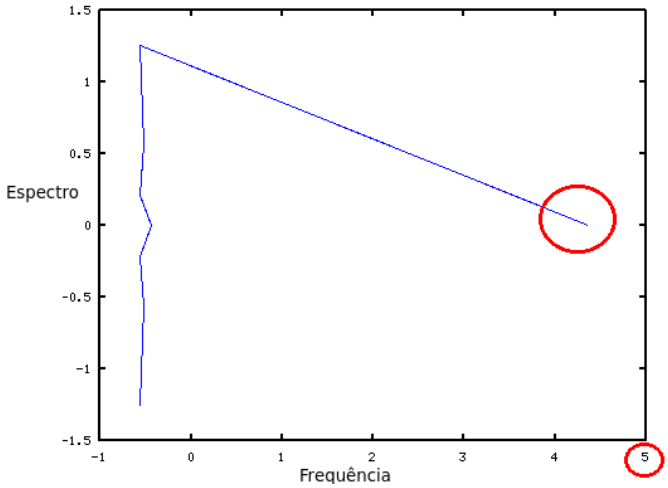


Figura 15: Aplicação do Teorema de Nyquist para a abordagem *Transfer Learning* para agrupamento de dados.

Tabela 10: Exemplo - Função *CDF* - Vetor 1.

Iteração	**Valor Mínimo**	**Valor Máximo**	**Elementos**	**Qtde**
1)	0.0	2.4	{**1**,10,20,15,7,8,16,13,5,6}	**1**
2)	2.4	4.8	{1,10,20,15,7,8,16,13,5,6}	**0**
3)	4.8	7.2	{1,10,20,15,**7**,8,16,13,**5**,**6**}	**3**
4)	7.2	9.6	{1,10,20,15,7,**8**,16,13,5,6}	**1**
5)	9.6	12.01	{1,**10**,20,15,7,8,16,13,5,6}	**1**
6)	12.01	14.41	{1,10,20,15,7,8,16,**13**,5,6}	**1**
7)	14.41	16.81	{1,10,20,**15**,7,8,**16**,13,5,6}	**2**
8)	16.81	19.21	{1,10,20,15,7,8,16,13,5,6}	**0**
9)	19.21	21.61	{1,10,**20**,15,7,8,16,13,5,6}	**1**
10)	21.61	**24.02**	{1,10,20,15,7,8,16,13,5,6}	**0**

Tabela 11: Exemplo - Função *CDF* - Vetor 1 Resultante.

Vetor 1	{1,10,20,15,7,8,16,13,5,6}
Vetor 1 - Qtde	{**1**,**0**,**3**,**1**,**1**,**1**,**2**,**0**,**1**,**0**}
Vetor 1 - Qtde Acumulado	{**1**,**1**,**4**,**5**,**6**,**7**,**9**,**9**,**10**,**10**}
Vetor 1 - CDF	{**0.1**,**0.1**,**0.4**,**0.5**,**0.6**,**0.7**,**0.9**,**0.9**,**1**,**1**}

Tabela 12: Exemplo - Função *CDF* - Vetor 2 Resultante.

Vetor 2	{5,4,1,15,12,11,8,9,10,11}
Vetor 2 - Qtde	{**1**,**1**,**1**,**2**,**4**,**0**,**1**,**0**,**0**,**0**}
Vetor 2 - Qtde Acumulado	{**1**,**2**,**3**,**5**,**9**,**9**,**10**,**10**,**10**,**10**}
Vetor 2 - CDF	{**0.1**,**0.2**,**0.3**,**0.5**,**0.9**,**0.9**,**1**,**1**,**1**,**1**}

Tabela 13: Exemplo - Método *KS* (*Kolmogorov - Smirnov*) - Execução.

Iteração	Diferença	Resultado
1)	$\|0.1 - 0.1\|$	0.0
2)	$\|0.1 - 0.2\|$	0.1
3)	$\|0.4 - 0.3\|$	0.1
4)	$\|0.5 - 0.5\|$	0.0
5)	$\|0.6 - 0.9\|$	**0.3**
6)	$\|0.7 - 0.9\|$	0.2
7)	$\|0.9 - 1.0\|$	0.1
8)	$\|0.9 - 1.0\|$	0.1
9)	$\|1.0 - 1.0\|$	0.0
10)	$\|1.0 - 1.0\|$	0.0

resultado = 0; i = 0;

somatórioVetorCDF1 = somarVetor(vetorCDF1);

somatórioVetorCDF2 = somarVetor(vetorCDF2);

Para cada elemento de vetor1 e vetor2

- p = vetorCDF1(i) / somatórioVetorCDF1;
- q = vetorCDF2(i) / somatórioVetorCDF2;
- resultado += p * log (p / q);
- i++;

resultado = resultado/ log (2);

p e q = representam o valor normalizado da curva acumulada a cada ponto

Figura 16: Método *KL* - (*Kullback - Leibler*).

Tabela 14: Método *KL* - (*Kullback - Leibler*) - Cálculos.

Iteração	p	q
1)	0.1/6.2 = **0.0161**	0.1/6.9 = **0.0144**
2)	0.1/6.2 = **0.0161**	0.2/6.9 = **0.0289**
3)	0.4/6.2 = **0.0645**	0.3/6.9 = **0.0434**
...	...	...
10)	1.0/6.2 = **0.16129**	1.0/6.9 = **0.1449**

Tabela 15: Método *KL* - (*Kullback - Leibler*) - Execução e Resultado.

Iteração	Resultado
1)	0.0+0.**0161***log(**0.0161**/**0.0144**) = 0.0017
2)	0.0017+**0.0161***log(**0.0161**/**0.0289**) = -0.0077
3)	-0.0077+**0.0645***log(**0.0645**/**0.0434**) = 0.0177
...	
10)	-0.00110+**0.16129***log(**0.16129**/**0.1449**) = **0.016151**
Resultado	**0.016151**$/log(2) = 0.023$

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados que buscam resolver o problema de agrupamento de dados. Cada trabalho foca de forma diferente o problema, combinando várias técnicas com o objetivo de transferir o conhecimento. Assim, os trabalhos possuem o mesmo objetivo de nosso algoritmo proposto. Ao fim da seção, há uma tabela comparativa.

3.1 *Learning from Cluster Examples*

Learning from Cluster Examples - LCE (Kamishima and Motoyoshi, 2003) é uma tarefa composta que combina características das técnicas de aprendizagem a partir de exemplos e de *clustering*, propondo uma extensão destas duas abordagens conhecidas. O objetivo não é encontrar uma regra para classificar objetos únicos, ou um agrupamento particular, mas sim, encontrar uma regra de particionamento para um *dataset* através de um outro *dataset* de exemplo. Cada exemplar presente no conjunto de exemplos de formação é um *par* que consiste em um conjunto de objetos acompanhados por uma partição verdadeira (um conjunto de *clusters*) para este conjunto. A regra obtida através da aprendizagem deste conjunto é utilizada para estimar uma partição real em um conjunto de objetos invisível. Desta forma, em contraste com a aprendizagem por meio de exemplos, que procura encontrar uma regra para classificar um objeto, o LCE adquire uma regra para particionar um conjunto de objetos. Entretanto, embora o objetivo do agrupamento seja a partição de um conjunto de objetos, a regra de particionamento produzido pelo LCE é então aplicável a qualquer objeto definido a partir de um mesmo domínio. Em suma, o LCE tem a natureza de supervisionar o aprendizado através de exemplos, e traz isso para a tarefa de agrupamento.

A entrada para o algoritmo é um conjunto de valores de características de *linha-segmentos*, ou seja, o comprimento ou a conectividade entre eles. Então, para aplicar um algoritmo de agrupamento, o usuário deve também especificar o conhecimento, como uma função potencial ou uma medida de dissimilaridade. Desta maneira, é difícil especificar tal conhecimento, porque há uma dificuldade em detectar a correspondência entre

os valores. Mesmo em tal caso, os autores consideram fácil atribuir as partições. Assim, eles supõem que ela é útil para derivar regras de particionamento a partir de exemplos de partição, sendo importante para a tarefa de LCE como um todo.

Na Figura 17 observamos que o LCE é dividido em 2 etapas: a etapa de aprendizado e a etapa de particionamento. A regra de particionamento, obtida na etapa de aprendizado, funciona como uma *ponte-de-ligação* entre as duas etapas. Na etapa de aprendizado há um conjunto de descrição dos objetos e um conjunto de objetos de treinamento. Além disso, na etapa final, o subconjunto de objetos ainda não visualizados (presente no *dataset*) são particionados pela regra inferida na etapa inicial deste processo.

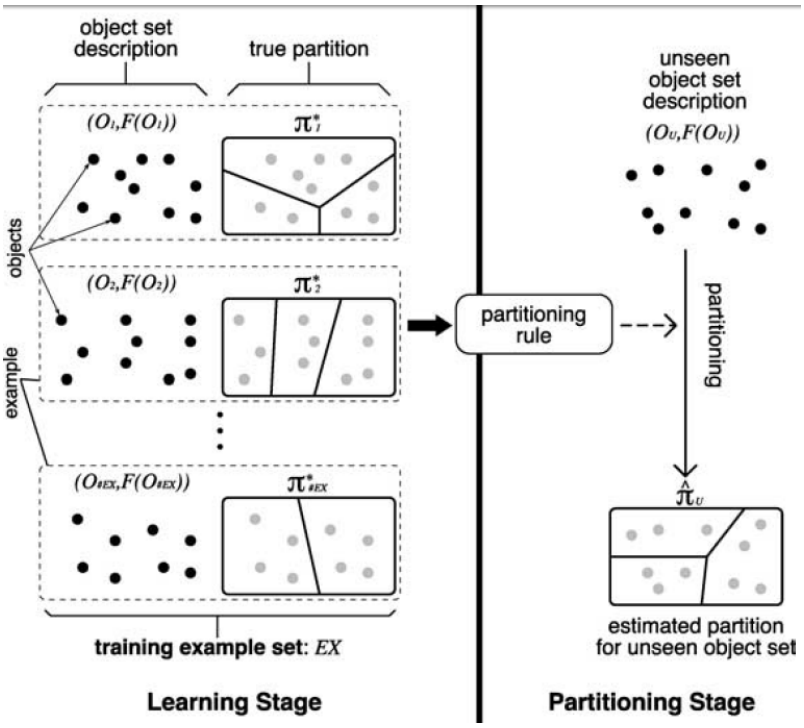


Figura 17: Exemplo *Learning from Cluster Examples* – Figura extraída de (Kamishima and Motoyoshi, 2003).

3.2 *Learning to Cluster using Local Neighborhood Structure*

Rosales et al. (2004) realiza a tarefa de agrupamento de dados por meio da análise de vizinhança dos objetos do *dataset*. Objetos que compartilham vizinhanças possivelmente fazem parte de um mesmo grupo, logo, o contrário é verdadeiro, vizinhanças heterogêneas podem indicar que esses objetos não compartilham nenhum vizinho em comum e, por consequência, serem elementos de grupos distintos. Além disso, o artigo destaca que para alguns problemas de classificação, a estrutura local pode ser mais relevante do que outras medidas de similaridade pois, nestes problemas, cada classe pode ter funções de similaridade diferente e que a função global seria o resultado de uma sobreposição

aproximada de todas as funções de similaridade de cada classe. Somado à dificuldade de definir as funções de similaridade, há a premissa de que a tarefa de agrupar é por si só um problema mal definido, pois o melhor agrupamento seria cada objeto em um grupo de um único elemento (ele próprio). É preciso ainda restringir os critérios utilizados para determinar e avaliar se houve ou não um bom resultado de agrupamento.

A Figura 18 ilustra a ideia de que, se duas vizinhanças próximas possuírem classes diferentes, a compatibilidade entre elas diminui com o número de vezes que os seus elementos em comum discordam (variações poderiam também ser de interesse). Note-se que usando esta definição, pares de objetos que possuem a mesma classe de vizinhança são igualmente compatíveis, independentemente do número de elementos compartilhados.

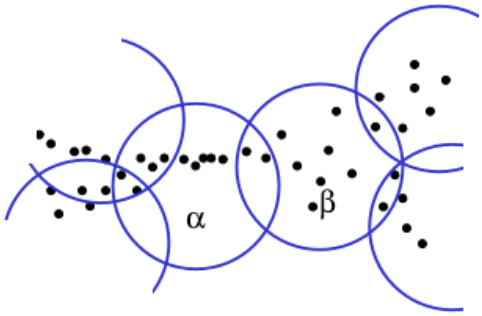


Figura 18: Exemplo *Learning to Cluster using Local Neighborhood Structure* – Figura extraída e adaptada de (Rosales et al., 2004).

No objetivo de caracterizar a estrutura local do *dataset*, o trabalho faz uso de atributos de vizinhança, salientando que, extensas vizinhanças podem tornar a análise desta vizinhança um problema complexo. Entretanto, para que não ocorra problema de intratabilidade computacional, um método de vizinhança baseado no Modelo de Markov é incorporado ao modelo proposto no artigo. Este método de vizinhança baseado no Modelo de Markov considera que duas vizinhanças são probabilisticamente relacionadas quando existe, pelo menos, um objeto compartilhado entre elas. Além disso, para conseguir aprender a estrutura das vizinhanças, seria necessário aplicar o conceito de aprendizagem, realizando assim o agrupamento por meio de dados já rotulados. A aprendizagem através de dados já rotulados somente poderia ser feita na medida que fosse possível solucionar o problema de *classe-dependência* entre as vizinhanças, ou seja, assumir que, ao lado de um não-rotulado *dataset* Z, exista um ou mais *datasets* rotulados L^i, com correspondência entre os rótulos (ou classes). Na parte de inferência dos *clusters* (ou grupos) existe a adição de um parâmetro *factor graph*, que descreve a relação entre as variáveis aleatórias definidas no modelo. Esse *factor graph* permite que análises complexas e algoritmos por inferências possam ser facilmente derivados a partir daí, emergindo assim, os grupos (Rosales et al., 2004).

3.3 *Self-taught clustering*

O trabalho (Dai et al., 2008) propõe um algoritmo que visa agrupar um *dataset-alvo não-rotulado* com o auxílio de um outro *dataset-auxiliar* também *não-rotulado*. O *dataset-auxiliar* pode ajudar na descoberta da melhor representação dos dados do *dataset-alvo*. Desta forma, o problema pode ser considerado uma instância de um problema de *Transfer Learning* (aprendizagem por transferência), já que se faz uso do conhecimento obtido a partir de uma tarefa de aprendizagem para aperfeiçoar o desempenho de uma outra, ainda que essas tarefas de aprendizagem sejam de domínios ou distribuições diferentes. Essa característica agrega valor à proposta do artigo já que não é comumente encontrada nos algoritmos clássicos de agrupamento. No algoritmo, os dois co-agrupamentos são desenvolvidos simultaneamente, sobre o *dataset-alvo* e *dataset-auxiliar* com objetivo de encontrar características compartilhadas (ou semelhantes) entre eles.

A Figura 19 demonstra que diferentes objetos podem compartilhar algumas características relevantes ou em comum, ou seja, anéis de diamante compartilham características com os objetos feitos em diamante; os anéis que possuem partes de platina compartilham características com os objetos de platina, além disso, a platina e o titânio compartilham muitas partes e uma série de outras características com o metal. Neste caso, os dados auxiliares podem ser utilizados para ajudar a identificar uma melhor forma de representação de dados, beneficiando assim, o conjunto de objetos do *dataset-alvo*.

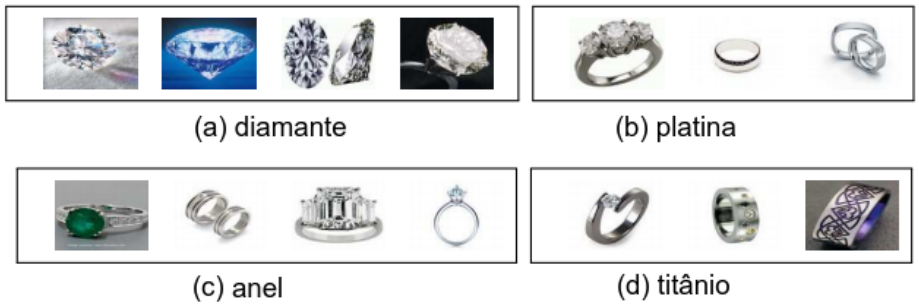


Figura 19: Exemplo *Self-taught Clustering* – Figura extraída e traduzida de (Dai et al., 2008).

A função objetivo proposta no artigo é:

$$J = I(X, Z) - I(X^q, Z^q) + \lambda[I(Y, Z) - I(Y^q, Z^q)]$$

Os parâmetros X e Y correspondem aos conjuntos de valores dos *datasets alvo* e *auxiliar*, respectivamente, e Z, o conjunto que corresponde ao que há de comum no espaço de características entre os conjuntos. I denota a informação mútua existente entre os conjuntos X e Y. Além disso, $I(X^q)$ e $I(Y^q)$ representam as junções de distribuições de probabilidades.

O parâmetro λ presente na função objetivo tem a função de balancear a influência entre os *dataset-alvo* e o *dataset-auxiliar*. Nos experimentos descritos no artigo, os autores observaram empiricamente que, quando o $\lambda > 1$, o desempenho do algoritmo proposto é insensível ao parâmetro, indicando que o *dataset-auxiliar* pode ajudar o *dataset-alvo* na tarefa de agrupar. Nos experimentos foi utilizado $\lambda = 1$, já que foi mostrado que é difícil determinar teoricamente seu valor. Além disso, com o objetivo de avaliar a qualidade dos resultados, foi utilizada a medida de *entropia*, ou seja, o valor de informação contida em cada grupo resultante dos agrupamentos. Ainda durante a realização dos experimentos, verificou-se que 10 iterações são suficientes para a convergência do algoritmo, possuindo complexidade algorítmica linear.

A função objetivo trabalha com os dois *datasets* (*dataset-alvo* e *dataset-auxiliar*), incitando a transferência de conhecimento (*Transfer Learning*) entre eles. Como forma de melhor o desempenho, o artigo aplica o método *Kullback-Leibler* (Cover and Thomas, 1991) sobre a mesma função com o objetivo de otimizá-la.

3.4 Kolmogorov Complexity: Clustering Objects and Similarity

(Nasution, 2012) propõe uma métrica de similaridade baseada na complexidade de *Kolmogorov*. Segundo o autor do trabalho, *clustering* é um tema de grande estudo e em poucos trabalhos da área a similaridade automática é utilizada para agrupar as instâncias do *dataset*. No entanto, ele ressalta que há pesquisas que utilizam a *Complexidade de Kolmogorov* como meio de adquirir a informação presente nos objetos, como por exemplo, das *páginas web* (lugar de difícil extração de informação).

A complexidade de *Kolmogorov* é uma tarefa profunda e sofisticada que trata da quantidade de informação de objetos individuais, medida através do tamanho de sua descrição algorítimica. Ela é uma noção moderna de aleatoriedade e refere-se a conceito pontual de aleatoriedade, ao invés de uma aleatoriedade média como faz a teoria das probabilidades. A complexidade de *Kolmogorov* é definida por meio do conceito de máquina de *Turing*. A existência da máquina de *Turing* garante que a complexidade de *Kolmogorov* seja um conceito objetivo em virtude de que a complexidade não é dependente da máquina escolhida como referência para medi-la.

Sobre os problemas matemáticos, o autor fala que o mais importante é que os objetos sejam definidos de uma forma uniforme, como por exemplo, de conjuntos. Trata ainda sobre os estudos de *Turing*, que demonstrou ser impossível escrever um programa de computador que saiba predizer se um outro programa entrará em *loop* ou não. No entanto, o autor esclarece, dizendo que há estudos de métodos de aproximação para o problema e a complexidade de *Kolmogorov* permite a execução de um pequeno programa, onde os objetos podem ser dados de forma similar como ”um ser humano pode ser representado por DNA”.

3.5 Resumo dos Trabalhos Relacionados

A Tabela 16 apresenta um resumo das principais técnicas de agrupamento estudadas. K-means (MacQueen, 1967) e Learning From Cluster Examples (Kamishima and Motoyoshi, 2003) utilizam variações do método de agrupamento particional, ou seja, ambos buscam regras de particionamento, sendo através dos centróides do *dataset* (K-means) ou através de um subconjunto presente no conjunto de objetos. Além destes trabalhos, o algoritmo DBSCAN (Ester et al., 1996) aplica a abordagem de agrupamento por densidade, pois através da classificação dos objetos como centrais, o DBSCAN age como um fluido, agrupando todos os objetos que estejam na margem dos objetos centrais e dos objetos limite.

Por outro lado, (Nasution, 2012) e (Dai et al., 2008) buscam em seus trabalhos incorporar à tarefa de agrupar algum tipo de conhecimento prévio. (Nasution, 2012) obtém este conhecimento através da Complexidade de *Kolmogorov* e (Dai et al., 2008) do *dataset-auxiliar* e da técnica de *Transfer Learning*.

Tabela 16: Tabela comparativa dos Trabalhos pesquisados.

Trabalho	Extração de Conhecimento	Independência de Domínio	Estrutura Local	Dados Rotulados
K-means	Protótipos (centróides)	Não	Sim	Não
DBSCAN	Regiões de baixa e alta densidade	Não	Sim	Não
Learning from Cluster Examples	Regra de particionamento para um dataset através de um outro dataset de exemplo	Não	Não	?
Learning to Cluster using Local Neighborhood Structure	Análise de Vizinhança dos objetos do dataset	Não	Sim	Sim
Self-Taught Clustering	Dataset-Alvo e Dataset-Auxiliar são desenvolvidos e simultaneamente com objetivo de encontrar características compartilhadas (ambos não rotulados)	Não	Não	Não
Kolmogorov Complexity: Clustering Objects and Similarity	Medida de similaridade baseada na complexidade de Kolmogorov	Não	?	Não
TRABALHO PROPOSTO	Transferência de conhecimento capturado pelas árvores de decisão ou por método de regressão logística	Sim	Sim	Sim (Bases de Conhecimento)

4 ALGORITMO PROPOSTO

Este capítulo trata do funcionamento do algoritmo proposto. A Figura 20 resume seu funcionamento.

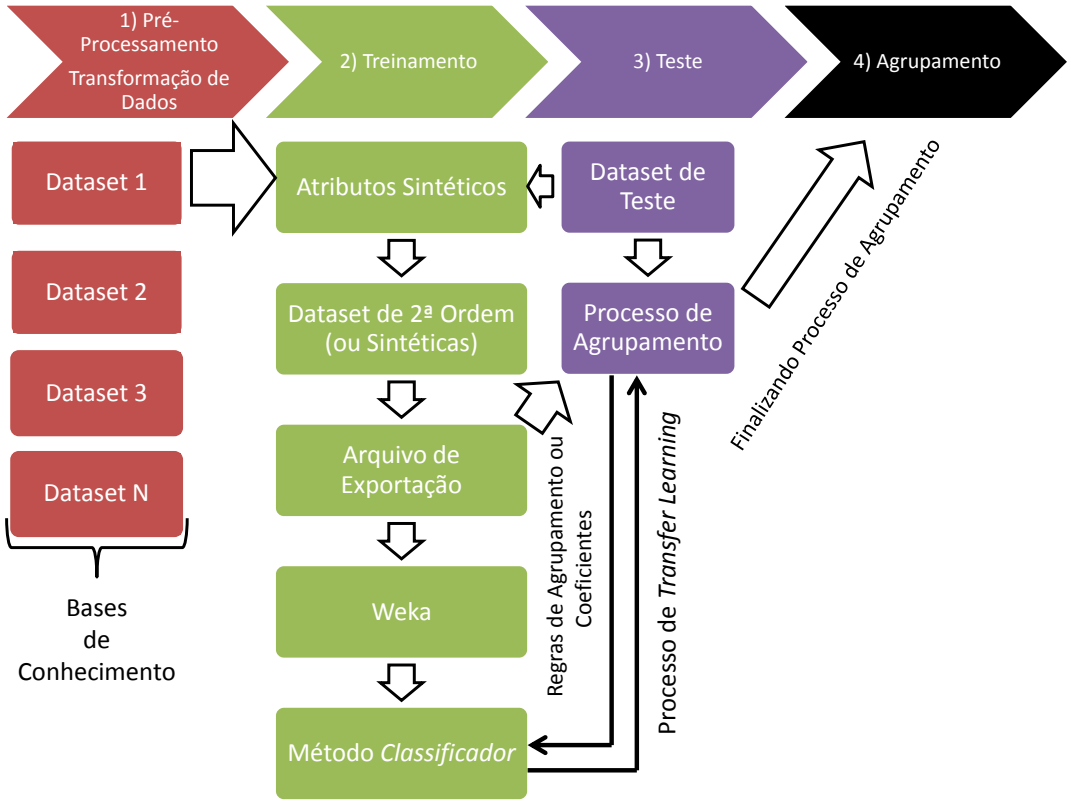


Figura 20: Funcionamento do algoritmo proposto.

A primeira etapa é denominada *Pré-Processamento*. As instâncias do *dataset* são combinadas em pares. Cada *par* de instâncias submetido ao conjunto de *Atributos Sintéticos* gera uma instância sintética (ou de 2ª ordem). As instâncias sintéticas são persistidas em um arquivo de exportação. O processo de geração do arquivo de exportação mapeia o conteúdo de todo *dataset* para um único domínio comum (o domínio dos atributos sintéticos). A uniformização de domínio favorece a transferência de conhecimento entre as bases envolvidas no processo de agrupamento.

De posse dos arquivos de exportação (etapa 2 - *Treinamento*) e com auxílio do software WEKA, aplica-se um método *classificador* com o objetivo de encontrar regras de agrupamento ou coeficientes de importância para os atributos sintéticos. Assim, quando se deseja agrupar um novo *dataset de Teste* (etapa 3 - *Teste*), esse *dataset* também terá seus atributos mapeados pelos *atributos sintéticos*, entretanto, em vez de encaminhar seu arquivo de exportação ao WEKA, como ocorre com os *datasets* de *Treinamento*, o mesmo será encaminhado para a etapa responsável pelo processo de agrupamento (etapa 3 - *Teste*). O processo de agrupamento decide se duas instâncias são ou não parte de um mesmo grupo, consultando um conjunto de regras de agrupamento ou buscando o valor numérico da instância sintética derivada do *par* quando a mesma foi submetida à função logística (etapa 4 - *Agrupamento*). Tanto o conjunto de regras de agrupamento como o resultado logístico são produzidos por meio do método *classificador* que recebe os *datasets de treinamento* e retorna um conjunto de árvores de decisão ou um vetor de coeficientes de importância para cada atributo sintético. Neste momento é que ocorre o *Transfer Learning*, já que o aprendizado anterior adquirido é transferido para o agrupamento do *dataset de Teste*.

As próximas seções deste capítulo descrevem detalhadamente as etapas do trabalho desenvolvido.

4.1 Etapa de Pré-Processamento

Durante a etapa de *Pré-Processamento*, os dados dos *datasets* participantes do processo são transformados de modo que o domínio de cada *dataset* é mapeado para um novo espaço comum de dimensões constantes, independentemente da dimensionalidade dos dados originais. Assim, cria-se uma homogeneização de todos os *datasets* envolvidos no processo e desenvolve-se um ambiente favorável para a transferência de conhecimento. A Figura 21 ilustra as ações realizadas nesta etapa.

Em um primeiro momento, duas ações são realizadas praticamente em paralelo no *dataset*: a geração de seu arquivo de exportação (onde são calculadas todas as instâncias sintéticas) e a gravação das instâncias do *dataset* no banco de dados. O *dataset* é persistido no banco de dados com objetivo de ser utilizado em futuras análises de qualidade. A ação de gerar o arquivo de exportação é subdividida em outras duas: *armazenar na base de dados* e *armazenar em um arquivo texto*. O arquivo de exportação em modo texto é utilizado como entrada para o método *classificador*. O arquivo de exportação também é persistido na base dados. Persistir o arquivo de exportação é importante para eliminar a busca sequencial que anteriormente era realizada no arquivo de exportação em modo texto e otimizar a busca de instâncias sintéticas durante o processo de agrupamento. O arquivo de exportação é composto de instâncias sintéticas que são combinações de *pares* de instâncias do *dataset*.

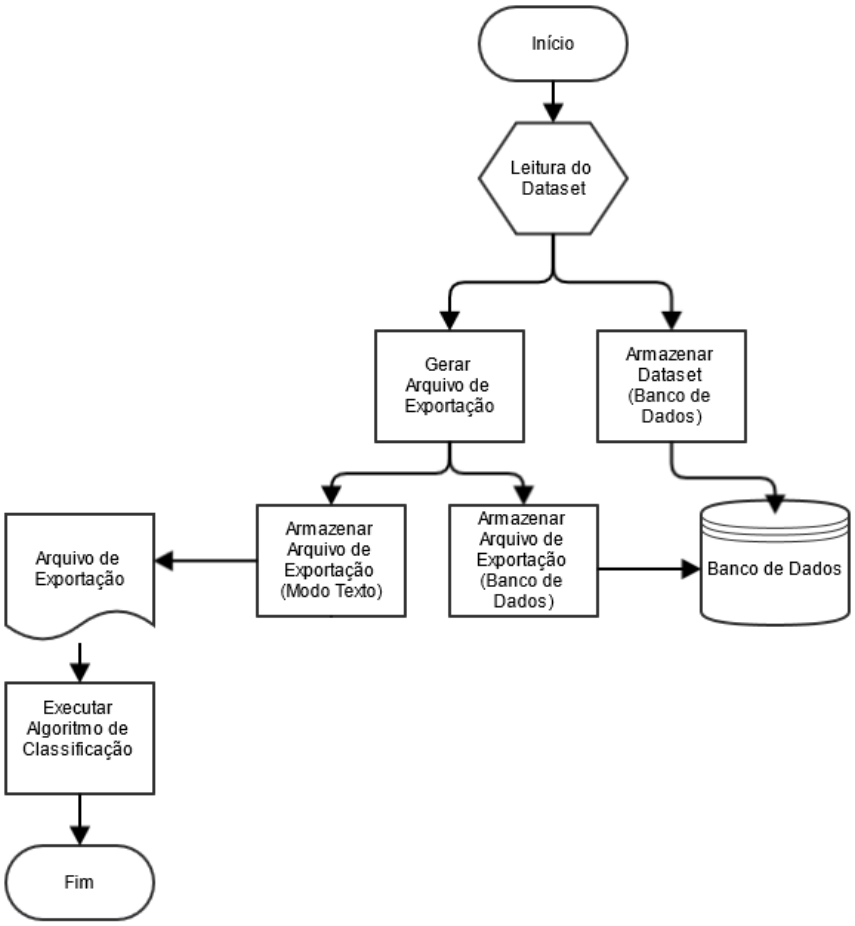


Figura 21: Etapa de Pré-Processamento.

4.2 Atributos Sintéticos

Nesta seção são descritos os atributos sintéticos desenvolvidos. Os atributos sintéticos tem a função de mapear os atributos originais das instâncias dos *datasets* da etapa de *Treinamento* e dos *Datasets de Teste* (Figura 20) em um novo conjunto de atributos. Assim, independente da origem dos *datasets*, todos devem ter ao final do pré-processamento dos *atributos sintéticos*, novos atributos pertencentes a um mesmo domínio. Para cada *dataset* mapeado pelos *atributos sintéticos*, existe um arquivo de exportação resultante, que contém os valores dos novos atributos. Todo arquivo de exportação possui suas instâncias reunidas *par-a-par*, já que todo atributo sintético recebe um *par* de instâncias e retorna um valor. Deste modo, é preciso estipular todos os possíveis pares de instâncias (genericamente A e B), pertencentes ao *dataset*, a fim de que os atributos sintéticos possam invocar todas as possíveis combinações de pares de instâncias. Além disso, os atributos sintéticos consideram tanto a ida como a volta, ou seja, iniciam tanto partindo de A até B como de B até A, realizando em ambos os casos, os procedimentos e os cálculos devidos. A Figura 22 apresenta graficamente os passos de pré-processamento de um atributo sintético genérico X.

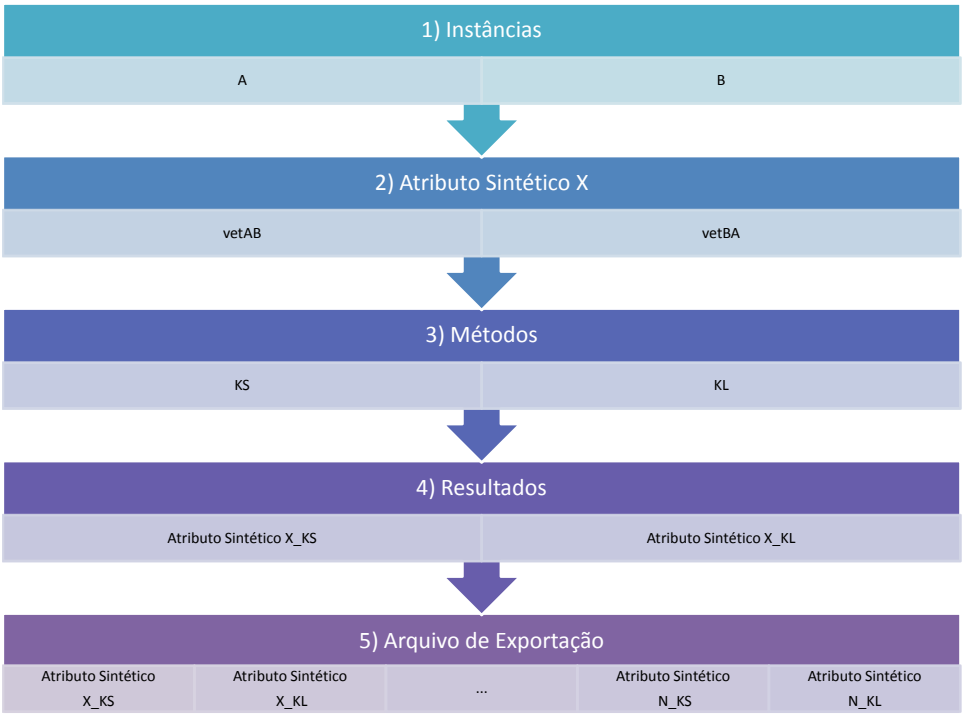


Figura 22: Funcionamento dos *Atributos Sintéticos*.

No passo 1, *X* recebe o *par* de instâncias *A* e *B*. No passo 2, são criados os vetores *vetAB* e *vetBA*, que armazenam os valores correspondentes ao que *X* representa, sendo *vetAB* os valores obtidos através da execução de *X* partindo de *A* até *B* e *vetBA* os valores obtidos da execução partindo de *B* até *A*. Em seguida (passo 3), os vetores são encaminhados aos métodos *Kolmogorov-Smirnov* (*KS*) (Massey, 1951) e *Kullback-Leibler* (*KL*) (Kullback and Leibler, 1951). No entanto, *vetAB* e *vetBA* devem ser transformados antes em distribuições acumuladas, devido aos métodos *KS* e *KL* exigirem distribuições acumuladas como parâmetros de entrada (*KS* e *KL* são comumente utilizados quando é necessário comparar distribuições de dados muito diferentes). No passo 4, obtém-se os resultados finais de *X* para ambos os métodos (a única exceção é o atributo sintético *K-Vizinhos* - seção 4.2.3 que não envia os vetores *vetAB* e *vetBA* para os métodos *KS* e *KL*, porque retorna diretamente a quantidade de vizinhos presente no conjunto resultante da intersecção destes vetores). Ao final (passo 5), os resultados finais do atributo sintético *X* são gravados no arquivo de exportação referente ao *dataset*.

A reprodução dos passos de pré-processamento para todos os *atributos sintéticos* com todas as possíveis combinações de pares de instâncias, resulta no arquivo de exportação completo do *dataset*. O arquivo de exportação é utilizado em dois momentos na proposta: o primeiro momento ocorre no software WEKA, com objetivo de encontrar regras de agrupamento ou coeficientes de regressão (resultado do método *classificador*) e; no segundo momento, na etapa de agrupamento, pois quando um novo *Dataset de Teste* é submetido à arquitetura, este também é submetido aos *Atributos Sintéticos*. Os *Atribu-*

tos Sintéticos geram para o *Dataset de Teste* seu arquivo de exportação, mapeando seus atributos para os mesmos atributos dos *datasets* de *Treinamento*, quando estes foram submetidos aos *Atributos Sintéticos*. Entretanto, nesta etapa, o novo arquivo de exportação é enviado para a etapa de *Teste*, a fim de que ocorra efetivamente o agrupamento. O processo de agrupamento decide se duas instâncias fazem ou não parte de um mesmo grupo através de consultas às regras de agrupamento ou à função logística (resultado da aplicação do método *classificador* que teve como parâmetro de entrada os *Datasets de Treinamento*). Assim, desta forma, ocorre o *Transfer Learning*, em virtude de que, todo conhecimento obtido por meio dos *Datasets de Treinamento* é utilizado no agrupamento do *Dataset de Teste*.

4.2.1 Distância Por Vizinho

Para todo *par* de instâncias A e B pertencentes ao *dataset*, o atributo sintético *DistânciaPorVizinho* retorna a distância dos vizinhos de *A* e *B* em ordem crescente, calculados a partir dos seguintes passos:

1. Calcular a distância euclidiana entre as instâncias A e B.
2. Calcular $RaioA = RaioB = \frac{d(A,B)}{2}$.
3. Buscar instâncias vizinhas de A que tenham valores de distância em relação a A menores ou iguais a $RaioA$.
4. Buscar instâncias vizinhas de B que tenham valores de distância em relação a B menores ou iguais a $RaioB$.
5. Atribuir aos vetores $vetAB$ e $vetBA$ os valores de distância que respeitaram a condição anterior.
6. Ordenar os vetores $vetAB$ e $vetBA$ em ordem crescente.

A Figura 23 apresenta o *dataset* utilizado para exemplificar o cálculo do atributo sintético *DistânciaPorVizinho*. Na Tabela 17 são exibidos os valores de distância euclidiana das instâncias A e B em relação as demais instâncias existentes no *dataset* de exemplo.

Os valores de distância da Tabela 17 determinam que a distância entre A e B é 20 ($d(A,B) = 20$) e consequentemente que $RaioA$ e $RaioB$ são iguais a 10. Assim, é possível estabelecer o conjunto de instâncias vizinhas de A como sendo $\{C, D, E, F\}$ e de B como sendo $\{G, H\}$, em virtude de que os valores de distância das instâncias C, D, E e F em relação a A serem menores que $RaioA$ e das instâncias G e H em relação a B serem menores que $RaioB$. As instâncias I e J não são vizinhas nem de A nem de B porque estão fora tanto da abrangência de $RaioA$ como da abrangência de $RaioB$.

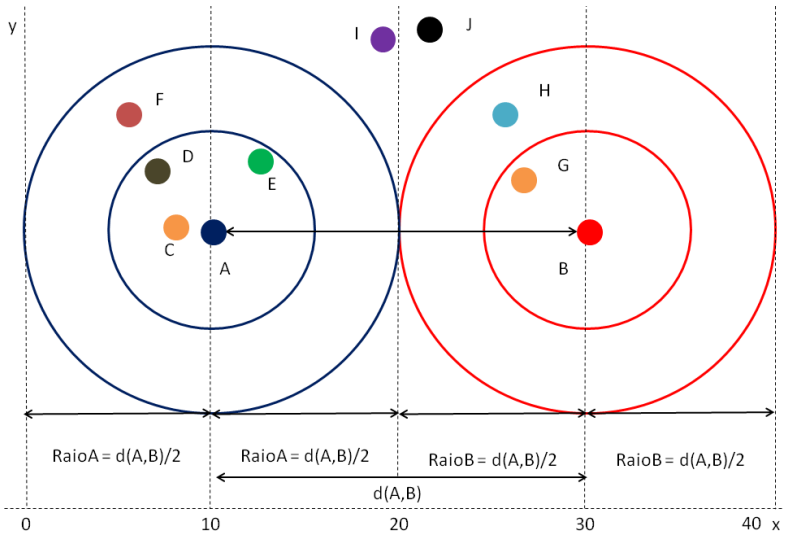


Figura 23: Atributo Sintético *DistânciaPorVizinho* - *dataset* de exemplo.

Tabela 17: Valores de distância euclidiana das instâncias A e B em relação as demais instâncias do *dataset* de exemplo.

Instâncias	**A**	**B**
A	0	**20**
B	**20**	0
C	**2**	21
D	**5**	22
E	**6**	21
F	**9**	25
G	22	**7**
H	22	**9**
I	22	15
J	22	16

Com posse dos conjuntos ($A = \{2, 5, 6, 9\}$ e $B = \{7, 9\}$), calcula-se a distância entre as curvas através dos métodos *Kolmogorov-SmirNov* e *Kullback-Leibler*.

4.2.2 Vizinhos Por Raio

O atributo sintético *VizinhosPorRaio* obtém, para um par de instâncias A e B a quantidade de vizinhos dentro de um raio de abrangência. Inicialmente, a variável $raio$ recebe o menor valor de distância entre duas instâncias encontrado no *dataset* ($\Delta raio$). A cada iteração, $raio$ é incrementada com o mesmo valor inicial ($\Delta raio$). A condição de parada ocorre quando $raio$ for maior que a metade da distância entre A e B ($\frac{d(A,B)}{2}$), calculados a partir dos seguintes passos:

1. Determinar a menor distância euclidiana não nula entre duas instâncias presente no

dataset ($\Delta raio$)

2. $raio = \Delta raio$
3. Determinar a distância euclidiana entre as instâncias A e B ($d(A, B)$)
4. Obter a quantidade de instâncias vizinhas de A e B que tenham valores de distância d, onde $d <= raio$
5. A cada iteração $raio = raio + \Delta raio$, repetir passo 4 e verificar condição de parada ($raio > \frac{d(A,B)}{2}$)

A Figura 24 apresenta um exemplo de aplicação do atributo sintético *VizinhosPorRaio*. Para o *dataset* de exemplo, o atributo sintético realizou duas iterações, isto é, houve dois deslocamentos de $\Delta raio$. No primeiro, A mapeou 3 instâncias vizinhas (C, D e E) e B mapeou 1 instância (G). No segundo, tanto A como B mapearam apenas 1 instância (A mapeou F e B mapeou H).

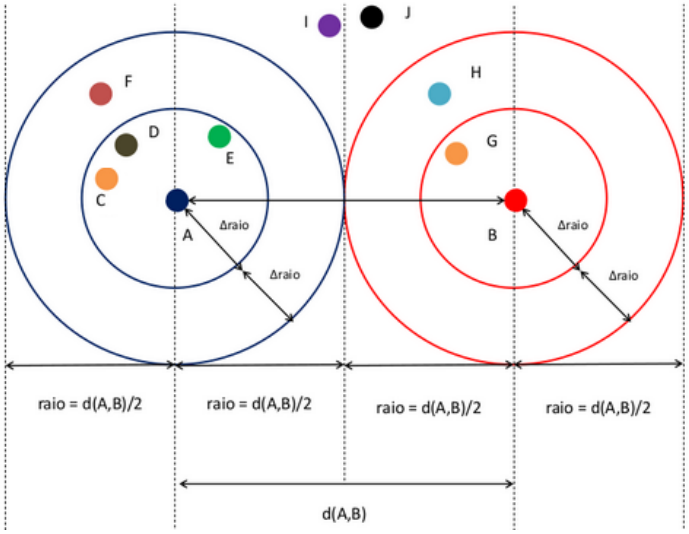


Figura 24: Atributo Sintético *VizinhosPorRaio* - *dataset* de exemplo.

Atribui-se os valores obtidos pelas instâncias A ($\{3, 1\}$) e B ($\{1, 1\}$) para os vetores $vetAB$ e $vetBA$ respectivamente, e calcula-se a diferença entre as distribuições acumuladas através dos métodos *KS* e *KL*, sendo os valores das diferenças os resultados finais do atributo sintético *VizinhosPorRaio* para o par de instâncias A e B.

4.2.3 K-Vizinhos

Para todo *par* de instâncias (A e B) do *dataset*, o atributo sintético *K-Vizinhos* obtém a porcentagem de vizinhos em comum, através da intersecção dos conjuntos de instâncias das *K-Vizinhas* mais próximas de A e B, sendo K o número máximo de elementos presente em cada conjunto de instâncias vizinhas mais próximas.

Com o objetivo de exemplificar o funcionamento do atributo sintético, é possível determinar $K = 2$, e obter para A e B os valores de distância em relação aos seus vizinhos (as demais instâncias do *dataset*). Neste exemplo, considera-se um *dataset* com 4 instâncias: A, B, C e D. Os valores de distância de A e B em relação as demais instâncias do *dataset* estão na Tabela 18.

Tabela 18: Atributo Sintético *K-Vizinhos* - *dataset* de exemplo.

Instâncias	**A**	**B**	C	D
A	0	**10**	**20**	30
B	10	0	**1**	**5**

Considerando $K = 2$, é preciso selecionar as duas instâncias mais próximas tanto A e como de B. Para compor o conjunto de *K-Vizinhos* de A, as instâncias B e C foram as selecionadas. Por outro lado, o conjunto de *K-Vizinhos* de B é composto pelas instâncias C e D. Assim, o conjunto intersecção resultante dos conjuntos de *K-Vizinhos* de A e de B é composto por apenas **1** instância (instância C). Logo, o resultado final do atributo sintético *K-Vizinhos* para este exemplo é $1/K = 0.5$ $(1/2 = 0.5)$.

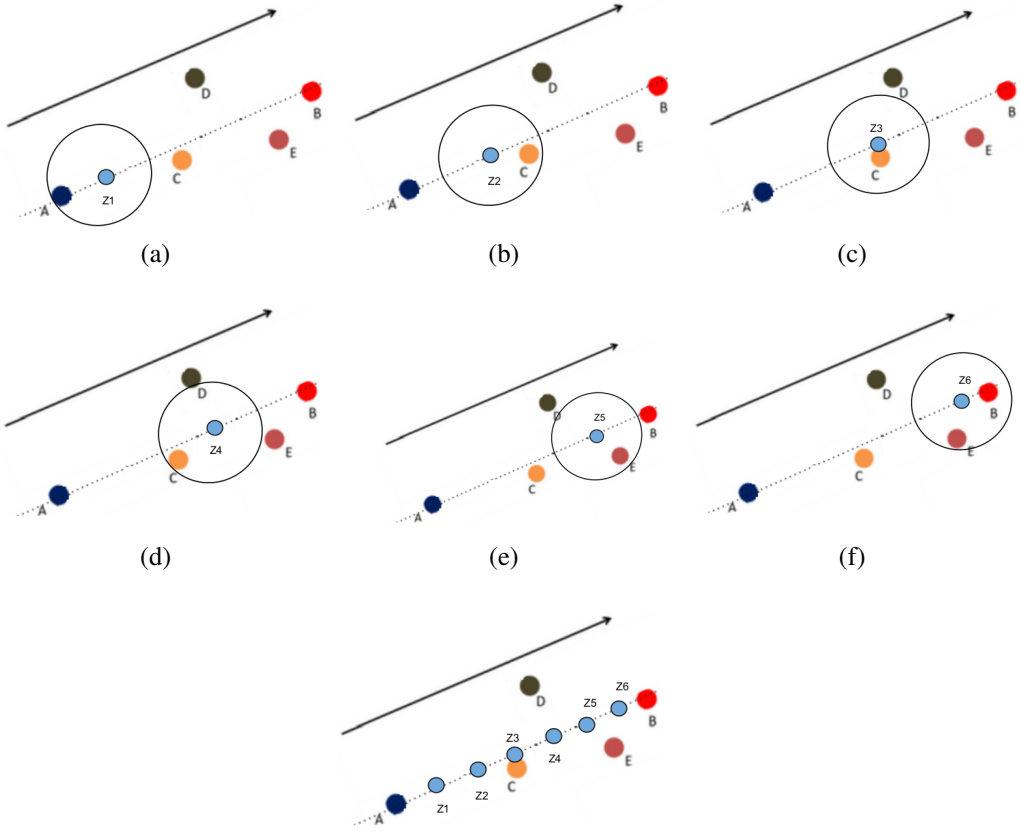
Obs: Este atributo sintético desconsidera a relação da instância com ela mesmo $(d(A, A) = 0)$.

4.2.4 Deslocamento Linear

O atributo sintético *DeslocamentoLinear* funciona da seguinte maneira: partindo da instância A vamos até a instância B. A cada passo, busca-se uma instância Z (entre A e B) e captura-se a quantidade de vizinhos dentro do seu raio de abrangência. Em seguida, desloca-se um valor Δp, buscando uma nova instância Z e sua quantidade de vizinhos. Assim, cria-se entre A e B um conjunto de instâncias artificiais $Z = \{Z1, Z2, Z3, ..., Zn\}$, armazenando, para cada elemento do conjunto, a quantidade de vizinhos. A Figura 25 mostra a instância *A* deslocando-se até a instância B. As instâncias *C*, *D* e *E* estão localizadas em uma região intermediária entre as instâncias *A* e *B*.

Para obter as instâncias do conjunto Z é preciso considerar:

- $m = d(A, B)$, sendo $d(A, B)$ a distância euclidiana entre as instâncias A e B
- $raio_{abrangencia} = \dfrac{m}{2}$
- $\Delta =$ menor distância entre duas instâncias encontrada no *dataset*
- $n = \dfrac{m}{\Delta}$ e $\Delta p = \dfrac{1}{n}$
- A cada iteração $P = P + \Delta p$



(a) (b) (c)

(d) (e) (f)

(g) Conjunto Z

Figura 25: Etapas - Deslocamento Linear.

A cada iteração n é preciso determinar os valores dos atributos da instância Zn correspondente. Os valores são definidos pela equação 5.

$$(x_Z, y_Z, .., n_Z) = (x_A, y_A, .., n_A) + P * (x_B - x_A, y_B - y_A, .., n_B - n_A) \quad (5)$$

Para exemplificar o funcionamento do atributo sintético é preciso considerar o *dataset* da Tabela 19 que possui instâncias com dois atributos (x e y). Logo, considerando as instâncias de $Id = 0$ e $Id = 8$ como A e B, respectivamente, e realizando os cálculos devidos, cria-se a Tabela 20.

A Tabela 20 pode ser apresentada graficamente pela Figura 26. Nesta figura, as instâncias de *Id=0* e *Id=8* estão na cor *vermelha*, enquanto as demais instâncias do conjunto Z estão em *azul*.

O atributo sintético *DeslocamentoLinear* retorna a quantidade de vizinhos que cada elemento do conjunto Z de instâncias encontra em sua periferia. Neste exemplo, com posse da quantidade de vizinhos da Tabela 20, são criados os seguintes vetores resultantes:

Tabela 19: Atributo Sintético *DeslocamentoLinear* - *dataset* de exemplo.

Instância Id	Atributo x	Atributo y
0	**1**	**1**
1	2	2
2	4	6
3	3	3,5
4	10	11
5	10	10
6	8	7
7	9	9
8	**20**	**20**
9	18	19
10	19	20

Tabela 20: Conjunto Z de instâncias intermediárias entre as instâncias *Id=0* e *Id=8*.

Zi	x	y	Quantidade de Vizinhos
1	1,71	1,71	8
2	2,41	2,41	8
3	3,12	3,12	8
4	3,83	3,83	8
5	4,54	4,54	8
6	5,24	5,24	8
7	5,95	5,95	8
8	6,66	6,66	8
9	7,36	7,36	8
10	8,07	8,07	8
11	8,78	8,78	8
12	9,49	9,49	9
13	10,19	10,19	10
14	10,90	10,90	10
15	11,61	11,61	9
16	12,31	12,31	9
17	13,02	13,02	8
18	13,73	13,73	8
19	14,44	14,44	8
20	15,14	15,14	7
21	15,85	15,85	7
22	16,56	16,56	7
23	17,26	17,26	6
24	17,97	17,97	6
25	18,68	18,68	5
26	19,38	19,38	5

$$vetAB = \{8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 9.0,$$

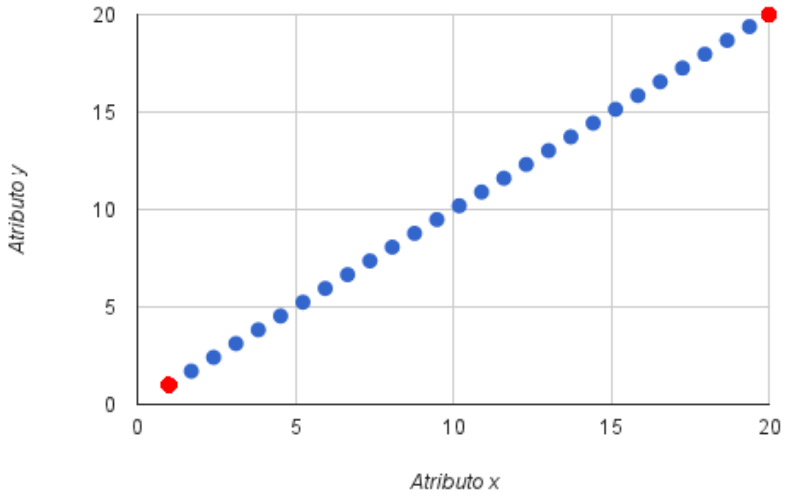


Figura 26: Conjunto Z de instâncias intermediárias entre as instâncias *Id=0* e *Id=8*.

$$10.0, 10.0, 9.0, 9.0, 8.0, 8.0, 8.0, 7.0, 7.0, 7.0, 6.0, 6.0, 5.0, 5.0\}$$

$$vetBA = \{5.0, 5.0, 6.0, 6.0, 7.0, 7.0, 7.0, 8.0, 8.0, 8.0, 9.0, 9.0, 10.0, 10.0,$$

$$9.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0\}$$

Todavia, o atributo sintético *DeslocamentoLinear* possui uma diferença em relação aos demais: os vetores *AB* e *BA* não são comparados entre si, mas comparados com uma curva acumulada fixa. Com posse dos valores resultantes, ou seja, o *vetAB* comparado com a curva acumulada fixa e o vetor *vetBA* também comparado com a mesma curva acumulada, o valor final do atributo sintético é dado pelo maior valor obtido (o procedimento é independente do método que calcula a distância entre curvas). Essa proposta foi utilizada devido ao fato de que, comparar as curvas $vetAB$ e $vetBA$ como ocorre com aos demais atributos sintéticos, especificamente para o atributo sintético *DeslocamentoLinear*, provocou durante os experimentos uma quantidade significativa de valores zerados. Isso dificultava a utilização do atributo sintético na arquitetura, pois o mesmo não tornava-se um atributo importante quando decidia-se agrupar ou não agrupar um *par* qualquer de instâncias.

4.3 Etapa de Treinamento

O conteúdo dos *datasets* dos módulos de *Treinamento* e de *Teste* é mapeado para um mesmo domínio através do conjunto de *Atributos Sintéticos*. Assim, para cada *dataset* submetido ao módulo de *Atributos Sintéticos*, há um arquivo de exportação, que contém os valores dos novos atributos calculados pelos *Atributos Sintéticos*. Na Figura 27 é possível visualizar graficamente o processo de geração do arquivo de exportação dos *datasets* submetidos ao método. Além disso, na Figura o *dataset* servirá de base de co-

nhecimento já que possui como último atributo a confirmação (*true*) ou a negativa (*false*) de que o par de instância que resultou a instância sintética que corresponde cada linha do arquivo de exportação fazia ou não parte do mesmo grupo. O *dataset de teste* não possui rótulos, logo seu arquivo de exportação não terá esse último atributo.

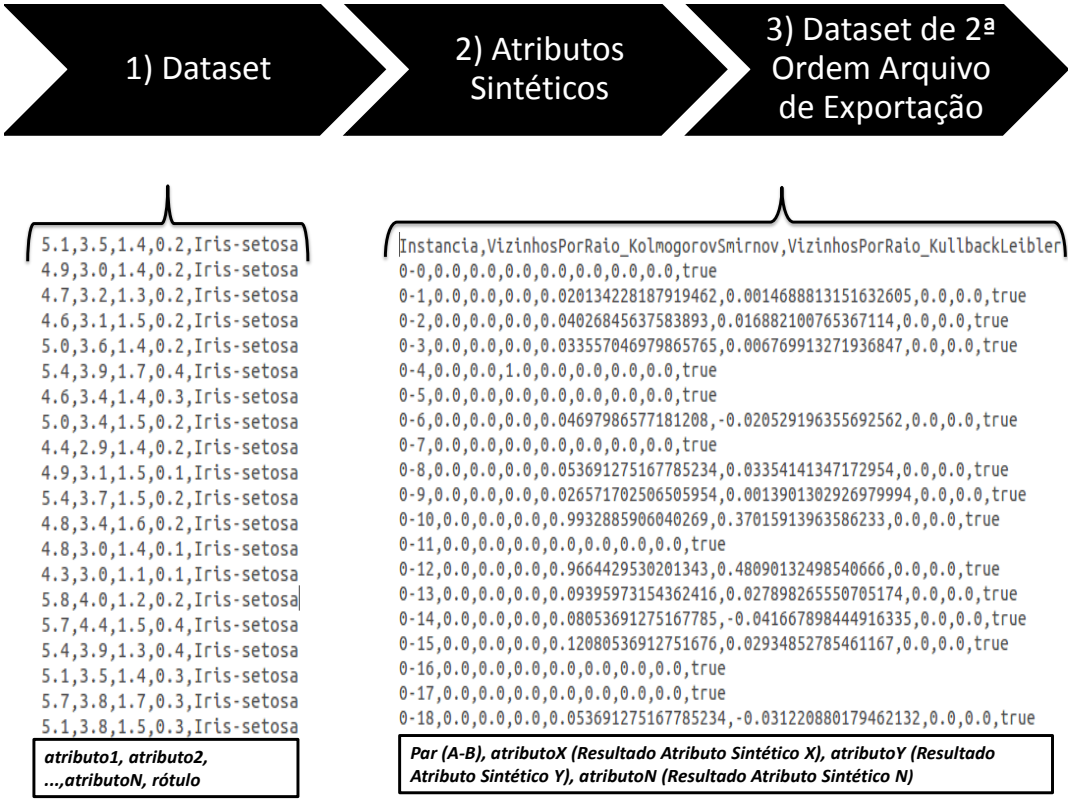


Figura 27: Funcionamento - Geração do Arquivo de Exportação através dos *Atributos Sintéticos*.

De posse dos arquivos de exportação dos *datasets* da etapa de *Treinamento*, aplica-se um método *classificador*. Logo, quando se deseja agrupar um novo *dataset de teste*, o *dataset teste* também terá seus atributos mapeados pelos mesmos *Atributos Sintéticos* e seu arquivo de exportação será desta vez encaminhado para a etapa responsável pelo *Processo de Agrupamento*.

4.4 Etapa de Teste e de Agrupamento

Esta seção tem o objetivo de explicar e exemplificar o processo de teste, dando ênfase às metodologias de agrupamento propostas nesta dissertação.

4.4.1 Metodologia de Agrupamento com Árvores de Decisão

A metodologia adotada durante os experimentos realizados com o *classificador* que expressa o conhecimento extraído dos *datasets* por meio de árvores decisão é esquematizada abaixo:

1. Rotular as instâncias do *dataset*. Cada instância é rotulada com um rótulo diferente. Ao final do processo, cada instância forma um grupo de um único elemento (a própria instância);

   Inicializar variáveis:

   int $qtdeTroca = 0$;

   int $rotuloNovo$;

2. Continuar até a *Condição de Parada* seja respeitada;

3. Sortear uma instância alvo:

   Sortear um índice (de 0 a n (número de instâncias))

4. Sortear instâncias vizinhas (o tamanho da vizinhança deve ser no máximo 10% do número total de instâncias).

   O sorteio da vizinhança é auxiliado pela função desenvolvida $randvet$. A função recebe um vetor dos valores inversos de distância em relação a todos seus vizinhos, ponderando as instâncias mais próximas da instância alvo sorteada. A função $randvet$ retorna, de forma aleatória, índices de instâncias vizinhas. Obs: Há possibilidade de que a vizinhança fique em número menor que 10%, devido ao fato de que a função $randvet$ retornar índices de instâncias vizinhas repetidos. As repetições são desconsideradas e os índices repetidos são contabilizados uma única vez.

5. Estabelecer a transferência de conhecimento através da vizinhança selecionada para a instância alvo sorteada.

6. Definir um possível novo rótulo para a instância alvo. O novo rótulo deverá ser, dentre os rótulos dos vizinhos selecionados que tiveram resultado verdadeiro no processo de eleição, o de maior quantidade de instâncias. Ou seja:

   Para cada rótulo presente na vizinhança - Armazenar a quantidade de vizinhos de mesmo rótulo.

7. Sendo o rótulo da instância alvo diferente do rótulo determinado pelo processo de definição do possível novo rótulo, o rótulo da instância alvo é modificado. Além disso, a variável $qtdeTrocas$ é incrementada. Caso contrário (negativo), ou seja, os rótulos sejam iguais, a variável $qtdeTrocas$ é zerada.

A metodologia é dividida em 7 etapas. Na etapa 1, cada instância presente no *dataset* é rotulada com um rótulo diferente, fazendo com que, no início do processo de agrupamento, cada instância participe de um grupo diferente com uma única instância (ela própria). A seguir, as variáveis $qtdeTroca$ e $rotuloNovo$ são inicializadas. A etapa 2 verifica se a condição de parada proposta foi respeitada ou não. Caso a condição parada ainda não seja respeitada o processo continua. Na etapa 3, um índice é sorteado aleatoriamente e a instância correspondente ao índice é definida como sendo a instância-alvo da iteração. Na etapa 4, é sorteada um conjunto de instâncias vizinhas da instância-alvo que servem de parâmetro para o método que determina o possível novo rótulo da instância-alvo. Na etapa 5, as instâncias vizinhas testadas em conjunto com a instância-alvo que tiveram resultado desfavorável durante o processo de eleição são excluídas deste conjunto, permanecendo no conjunto somente as instâncias de resultado favorável, ou seja, instâncias que pareadas com a instância alvo tiverem êxito no processo de eleição, ou seja, onde o processo de eleição decidiu que deveriam ficar no mesmo grupo. Assim, o conjunto (com apenas as instância com resultado favorável) é encaminhado para a próxima etapa (etapa 6). Na etapa 6, há um método que verifica, dentre as instâncias vizinhas selecionadas do conjunto, o rótulo que contém o maior número de instâncias. O rótulo que possuir o maior número de instâncias é retornado ao método principal. Esse rótulo é definido como sendo o *possível novo rótulo* da instância-alvo da iteração corrente. Na última etapa (etapa 7), compara-se o *possível novo rótulo* com o atual rótulo da instância-alvo. Caso os rótulos sejam diferentes, modifica-se o rótulo da instância alvo (o rótulo da instância alvo passa a ser *o novo possível rótulo* e contabiliza-se o contador de trocas). Caso os rótulos sejam iguais, nada ocorre e a condição de parada é verificada novamente.

No capítulo que compreende os experimentos (capítulo 5) é possível observar variações propostas de novas formas de condição de parada e de mecanismos de troca de rótulos e seus respectivos resultados. Observa-se também que o núcleo da metodologia foi mantido e que dependendo do proposto, houve variações nos resultados. Além disso, foi proposto utilizar em vez de árvores de decisão como fonte de conhecimento, coeficientes de importância para cada *Atributo Sintético* (coeficientes obtidos por meio de um método *classificador* de regressão logística). O método de regressão elimina também o processo de eleição, pois a função logística presente no método, por si só, decide se há ou não necessidade de agrupar um par qualquer de instâncias, excluindo assim, a exigência de obter os votos das bases de conhecimento (através de suas árvores de decisão).

4.4.2 Metodologia de Agrupamento com Regressão Logística

A metodologia adotada durante os experimentos realizados com o *classificador* de regressão logística é ilustrada abaixo.

1. Leitura do *dataset*.
2. Inicializar a variável max_iter com $1.5 * n$ (n é o número de instâncias) .
3. Obter a matriz de distâncias euclidianas para todo par de instâncias
4. Selecionar aleatoriamente uma instância alvo a.
5. Enquanto número de iterações for menor que max_iter

   Montar a lista com as distâncias euclidianas em relação a instância-alvo. Deve-se retirar a própria instância da lista.

   Obter a instância b. b é a instância mais perto de a.

   Obter o valor obtido pela função logística do par $a - b$.

   Caso o valor seja maior ou igual a média de todos os valores, a instância b recebe o rótulo da instância alvo a.

   Caso o valor obtido seja maior ou igual a média de todos os valores a próxima instância alvo será b (b será a nova instância alvo a).

   Caso o valor seja menor que a média de todos os valores, seleciona-se aleatoriamente uma nova instância alvo a e não realiza-se nenhuma mudança de rótulo (nem da instância b, nem da instância a).

Na primeira etapa realiza-se a leitura do *dataset* original. Além da leitura do *dataset*, inicializa-se a variável max_iter. A max_iter é responsável por armazenar o número máximo de iterações (praticou-se o durante os experimentos o valor de %50 maior que o número de instâncias e o valor manteve-se constante independente do *dataset de teste* e do número de instâncias).

Na etapa 2 cria-se uma matriz com, inicialmente, todos os valores zerados. A matriz possui a dimensão m x m, sendo m o número de instâncias. Em seguida, calcula-se para cada par de instâncias, a distância euclidiana e o valor obtido é armazenado na matriz. A distância euclidiana é utilizada também durante a execução do processo de agrupamento.

Na etapa 3 cada instância recebe um rótulo entre 0 e m e a definição dos rótulos é feita de modo sequencial (de 0 a $m - 1$). Além disso, determina-se qual será a instância que iniciará o processo. Em seguida, uma variável que contém o valor da média de todos os valores obtidos anteriormente pelos pares de instância é inicializada, quando os mesmos foram submetidos a função logística. Esse valor funciona como limitante. decidindo ou

não o agrupamento de pares de instâncias em mesmos grupos. A variável a recebe a instância inicial do processo.

Na etapa 4 é onde o processo de agrupamento acontece, ou seja, os rótulos definitivos das instâncias são atribuídos. Neste trecho há um *loop* que vai até o número máximo de iterações (definido anteriormente). A cada iteração se define uma lista de vizinhos da instância corrente e elimina-se desta lista a própria instância. Logo em seguida, obtém-se o vizinho mais próximo. Assim, compara-se o valor obtido da função logística do par composto pela instância corrente e seu vizinho mais próximo sorteada com o valor do limitante. Caso o valor seja maior ou igual, o rótulo da instância corrente é definido como sendo o mesmo rótulo de seu vizinho e a próxima instância da iteração seguinte é o vizinho. Caso contrário, obtém-se dentro da lista de instâncias, uma outra instância aleatória e o processo segue até o número final de iterações. O ponto principal da metodologia é a exploração da localidade entre as instâncias, ou seja, quando define-se a instância corrente com rótulo do vizinho e o processo de agrupamento segue com a instância vizinha, localmente localizada próximo da instância alvo anterior.

4.4.3 Processo de Agrupamento

O processo de agrupamento pode ser realizado tanto por árvores de decisão e seu respectivo processo de eleição como também pelo método de regressão e sua função logística. Nesta seção, será descrito de que forma cada proposta de agrupamento funciona e suas características.

4.4.3.1 Árvore de Decisão e o Processo de Eleição

O processo de eleição atua quando um novo *Dataset de Teste* é submetido ao algoritmo. No momento em que o *Dataset de Teste* tem suas instâncias mapeadas pelos *Atributos Sintéticos*, seu arquivo de exportação é encaminhado para a etapa de *Teste* (Método de Agrupamento). Na etapa de *Teste* existe um processo de eleição que determina se duas instâncias fazem ou não parte de um mesmo grupo mediante consultas no conjunto de regras de agrupamento, obtidas pelas árvores de decisão dos *Datasets de Treinamento*. Neste momento é que ocorre o *Transfer Learning*, já que todo conhecimento anterior adquirido é utilizado na tarefa de agrupar o *Dataset de Teste*. A Figura 28 ilustra a metodologia proposta.

O processo de eleição tem seu funcionamento ilustrado pela figura 29. Basicamente, cada instância sintética do arquivo de exportação (que é uma combinação de um par de instâncias do *Dataset de Teste*) é testada para cada *dataset de Treinamento*, com o objetivo de determinar se, para cada *dataset*, o par de instâncias faz ou não parte de um mesmo grupo. O teste é realizado percorrendo as árvores de decisão desses *datasets* e observando se para cada árvore há um resultado positivo ou negativo, sendo que, se houver no processo de eleição resultado final favorável, ou seja, se a maioria dos testes realizados nas

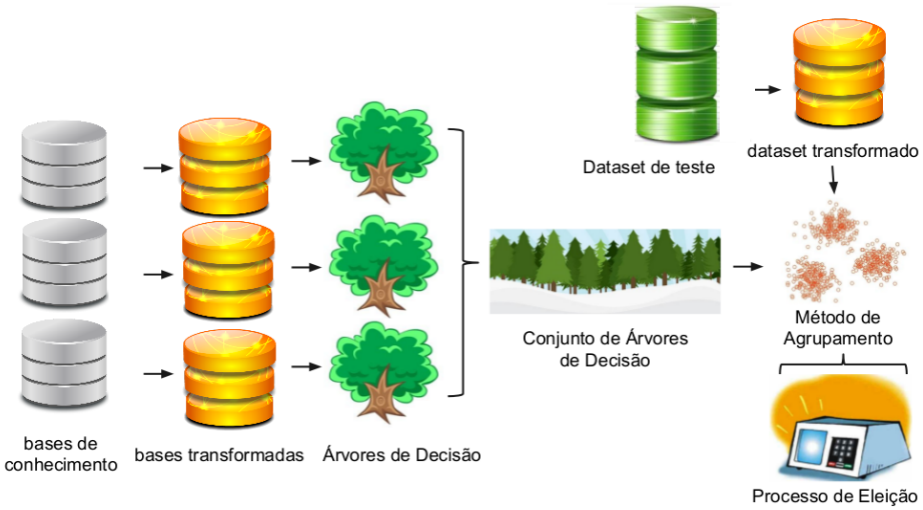


Figura 28: Metodologia de agrupamento baseada em Árvores de Decisão.

árvores de decisão for positivo, existe a confirmação de que o *par* de instâncias analisado deve fazer parte de um mesmo grupo. Assim, provoca-se entre os *datasets* a transferência de conhecimento, pois o processo de agrupamento do *dataset de Teste* ocorre auxiliado/-precedido pelo conhecimento dos *Datasets de Treinamento*. Na Figura 29, há três *datasets* de *Treinamento* que retornam dois resultados positivos e um negativo para a instância sintética *A-B*. Neste exemplo, o método proposto agruparia o *par* de instâncias *A* e *B* em um mesmo grupo.

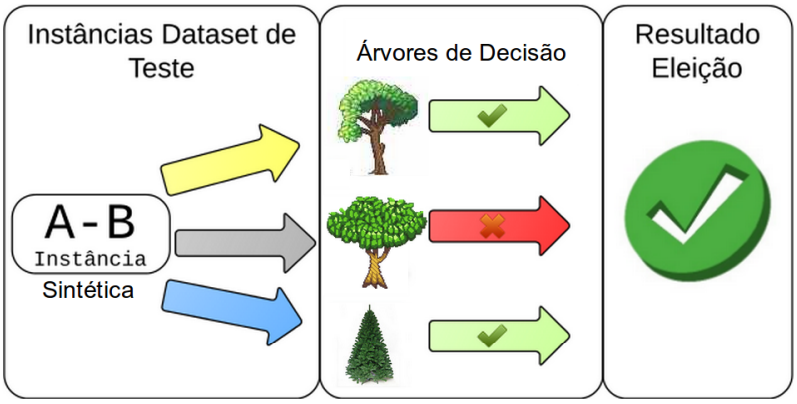


Figura 29: Processo de Agrupamento - Metodologia baseada em um Processo de *Eleição* por meio de Árvores de Decisão.

Outro ponto importante é a recomendação do uso de um número ímpar de árvores de decisão, obtidas dos *Datasets de Treinamento*, eliminando a possibilidade de haver empate durante processo de eleição, no quesito de dizer se qualquer duas instâncias estão ou não juntas em um mesmo grupo. Caso isso não ocorra, ou seja, haja um número par de *datasets de treinamento* e, por consequência, de árvores de decisão, em caso de empate, o processo de eleição decidirá de modo aleatório.

4.4.3.2 *Coeficientes de Importância - Atributos Sintéticos e o Método de Regressão Logística*

No processo de agrupamento com regressão logística, cria-se um grande *dataset único*, que possui as instâncias sintéticas de todas as bases de conhecimento. Deste modo, aplica-se no WEKA um método *classificador* de regressão logística e este retorna coeficientes de importância para cada atributo sintético que compõe o grande *dataset único* de conhecimento. Os coeficientes são incorporados à função logística de pertinência que calcula, para qualquer par de instâncias, a necessidade ou não de estarem em um mesmo grupo. A pertinência baseia-se em um limiar que varia entre 0 e 1, onde valores menores ou iguais a média representam, na proposta, decisões de não agrupamento e, valores maiores, resultados favoráveis de agrupamento.

A Figura 30 ilustra a proposta de agrupamento com a utilização do método de regressão logística. Na figura, as bases de conhecimento (*cinza*) são encaminhadas à etapa de pré-processamento e transformadas em um grande *dataset transformado único (ou global)*. Deste modo, aplica-se o método de regressão. Os coeficientes obtidos da regressão são encaminhados à função logística. Assim, quando o *dataset de teste* (*verde*) também é transformado, a função logística calcula a matriz de todos valores dos pares de instância desse *dataset*. Ao final, a matriz é enviada para o método de agrupamento e é utilizada como parâmetro de entrada do processo. No método de agrupamento, a ideia central, ou seja, a ação de agrupar ou não agrupar um *par* de instâncias, baseia-se na média dos valores contidos na matriz.

No capítulo 5 que trata da avaliação experimental, discute-se mais sobre a metodologia de agrupamento com a regressão logística e os problemas enfrentados devido ao desbalanceamento das classes nas bases de conhecimento (ou *treinamento*) .

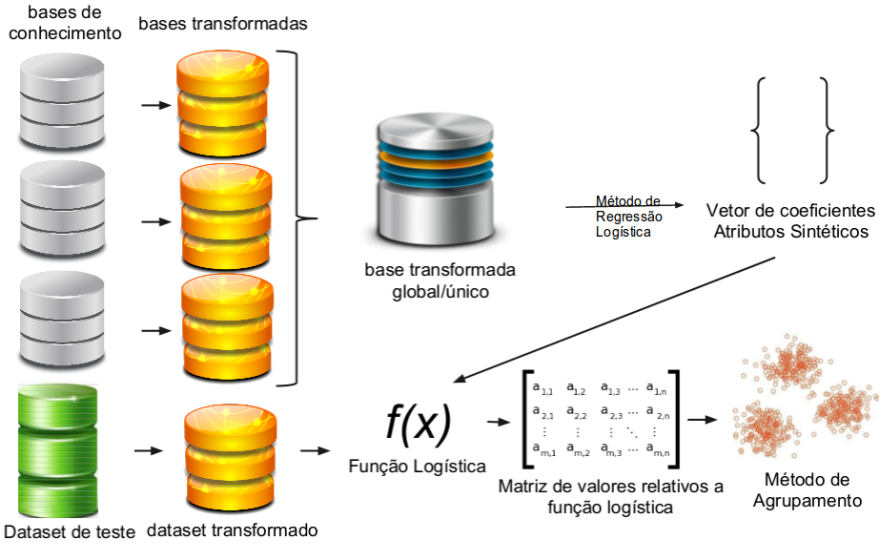


Figura 30: Metodologia de agrupamento baseada em Regressão Logística.

4.5 Decisões de Implementação

A seção discute as decisões relativas a implementação da proposta que foram tomadas no decorrer do desenvolvimento do trabalho.

Uma das principais medidas foi substituir o modo de busca das instâncias sintéticas durante o processo de eleição. Nos primeiros protótipos era necessária uma busca sequencial e lenta (*linha* por *linha*) para cada instância sintética. Assim, a busca sequencial foi substituída por consultas a uma base de dados. A base dados contém as mesmas instâncias sintéticas que estão no arquivo de exportação (modo texto).

Além de persistir o arquivo de exportação, outra importante solução que otimizou o método de agrupamento proposto foi a decisão de implementar o padrão de projetos *SINGLETON* (Freeman and Freeman, 2007) na classe responsável pela conexão ao banco de dados. Anteriormente, um novo objeto da classe era criado a cada consulta, ocasionando uso de memória e diminuindo desempenho (principalmente durante a etapa de pré-processamento). Com a implementação do padrão de projetos, há somente um objeto da classe de conexão em todo o programa, reduzindo a utilização da memória e, consequentemente, melhorando o desempenho.

Durante os experimentos, algumas metodologias de como processar o agrupamento foram testadas. Deste modo, verificou-se que o processo de agrupamento está ligado com a forma de obtenção/extração das regras de agrupamento ou de alguma outra forma de conhecimento, ou seja, ligado a como o *classificador* expressa o conhecimento extraído das bases de conhecimento. Por exemplo, quando se utiliza um *classificador* que expressa o conhecimento em forma de árvores de decisão, se faz necessário para o método de agrupamento um *parser*, que extrai das árvores de decisão (armazenadas em modo texto) suas regras de agrupamento. Além disso, existe a necessidade de um processo de eleição, pois se tem para cada base de conhecimento uma árvore de decisão correspondente e a decisão de agrupar um par de instâncias qualquer baseia-se no resultado favorável da maioria das árvores de decisão (ou bases de conhecimento) consultadas.

Um exemplo de uma árvore de decisão utilizada pela proposta de agrupamento pode ser visto na Figura 32. O *dataset* deste exemplo é o IRIS, encontrado no repositório de *datasets* UCI[1]. Para geração da árvore de decisão do *dataset* foram utilizados os valores padrões dos parâmetros sugeridos pelo WEKA, sendo o parâmetro *M* (*minNumObj*) a única exceção (alterado para *350*), com o objetivo de gerar uma árvore de decisão com menor altura. O *minNumObj* controla o número mínimo de ocorrências nas folhas da árvore de decisão. A Figura 31 mostra a tela de configuração do software WEKA e os parâmetros utilizados no algoritmo J48 (versão Java do algoritmo C4.5).

[1] http://archive.ics.uci.edu/ml/

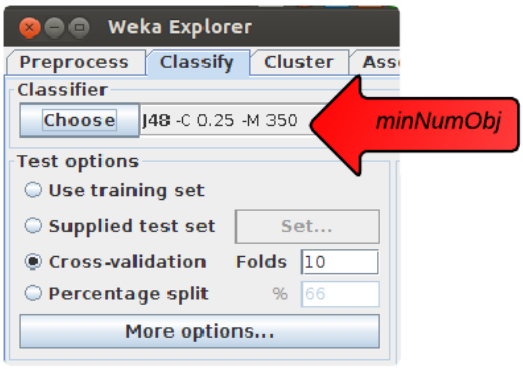


Figura 31: Parâmetros do Algoritmo J48 para o *dataset* IRIS.

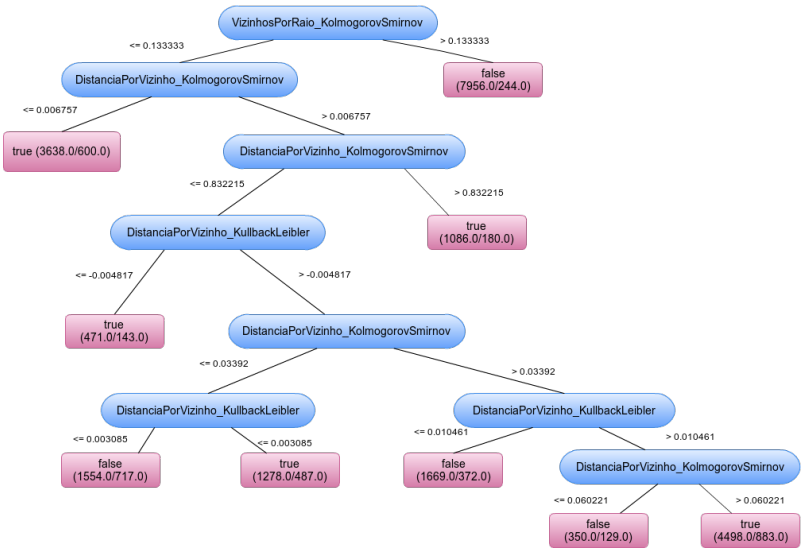


Figura 32: Árvore de Decisão Resultante do Arquivo de Exportação do *dataset* IRIS.

Além do uso de árvores de decisão, outras formas de aprendizagem podem ser usadas. Dentre elas existe, por exemplo, classificadores de regressão logística. Este tipo de *classificador* expressa o conhecimento extraído através de coeficientes que determinam a importância de cada atributo sintético no processo de definição de um par qualquer de instância estar ou não estar presente em um mesmo grupo. A Figura 33 ilustra a utilização da técnica de regressão logística no software WEKA. Além disso, na figura e, mais especificamente, na área denominada *Classifier Output*, observa-se os valores de importância definidos para cada atributo que compõe as instâncias pertencentes ao *dataset* representado no exemplo (atributos *duration*, *wage-increase-first-year* e etc).

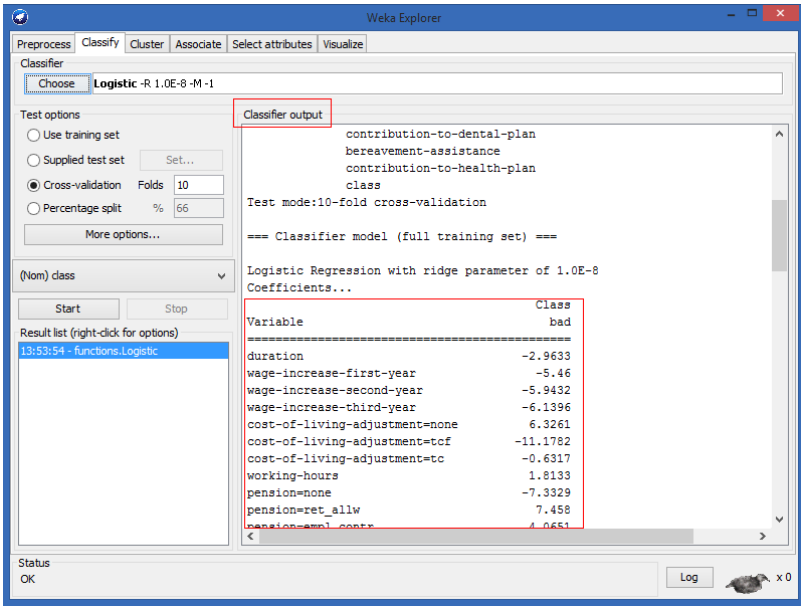


Figura 33: Regressão Logística.

5 AVALIAÇÃO EXPERIMENTAL

As próximas seções discutem as metodologias propostas e os resultados dos experimentos realizados.

5.1 Método Aglomerativo baseado em Árvores de Decisão

Um novo conjunto de 10 *datasets* foi selecionado dentre os *datasets* encontrados do repositório *http://faculty.washington.edu/kayee/cluster*. Além disso, para contornar o número reduzido de bases de conhecimento e para gerar um número maior de árvores de decisão e, por consequência, aumentar a capacidade de transferência de conhecimento, alterou-se a etapa de pré-processamento. Em vez de cada *dataset* gerar um *dataset transformado* e este *dataset transformado* resultar uma árvore de decisão, retirou-se 10 amostras de cada *dataset transformado* e a cada amostra foi submetida ao método *classificador*. Como resultado final do pré-processamento, 100 novas árvores de decisão foram obtidas do conjunto original de 10 *datasets*. Assim, realizou-se uma sequência de experimentos, considerando, a cada nova iteração, um *dataset* como sendo *dataset de teste* e os demais, com as suas respectivas árvores de decisão, *datasets* de treinamento.

Observação: A quantidade de instâncias das amostras foi definida como sendo $(\text{Número de Instâncias}/20)$ e as instâncias foram selecionadas aleatoriamente por meio de consultas à base de dados que armazena os *datasets* transformados (ou sintéticos).

5.1.1 Condição de Parada: 100 Não Trocas de Rótulo em Sequência

Neste conjunto de experimentos a condição de parada escolhida foi a de que o método terminaria sua execução quando atingisse 100 não trocas seguidas do rótulo das instâncias-alvo. A cada iteração sorteou-se um índice de modo aleatório e a instância do índice correspondente era definida como sendo a instância-alvo da iteração. No momento em que, independente da instância-alvo sorteada, houvesse 100 iterações seguidas onde o rótulo atual da instância-alvo não mudasse, ou seja, o método que determina o próximo possível rótulo da instância-alvo determinasse o mesmo rótulo atual da instância alvo, o processo de agrupamento encerrava.

Na Tabela 21 observa-se os resultados obtidos para cada *dataset*. Na tabela, a primeira coluna identifica os *datasets*; a segunda descreve o total de instâncias; a terceira identifica o número de classes. As colunas 4 e 5 identificam, respectivamente, os valores de *DBI* e de grupos encontrados. Nas colunas 5,6 e 7 tem-se a quantidade de instâncias por grupo e os valores de *microF* e *macroF*. O pior resultado de agrupamento encontrado foi, dentre os *datasets* do experimento, obtido pelo *dataset monk2*. Para o *dataset monk2* encontrou-se *13* grupos, sendo a grande maioria dos grupos compostos de uma única instância. Por outro lado, no mesmo *dataset*, encontrou-se um grande grupo que aglomerou, praticamente, todas as instâncias do *dataset*. O melhor resultado de agrupamento ocorreu com o *dataset zoo*. O experimento realizado no *dataset zoo* encontrou *6* grupos.

Tabela 21: Metodologia Aglomerativa com Árvores de Decisão - Condição de Parada: 100 não trocas de rótulo e Porcentagem Vizinhança: 10%.

dataset	atributos	instâncias	classes	dbi	grupos	instâncias por grupo	macroF	microF
sonar	60	208	2	0.76	3	(206/1/1)	0,36	0,54
zoo	16	101	7	1.1964	6	(47/44/3/3/2/2)	0,26	0,55
hamditolga	5	403	4	2.37	2	(395/8)	0,12	0,33
balance	4	625	3	1.0567	14	(612/1/*)	0,23	0,46
saheart	9	462	2	?	2	(438/24)	0,42	0,63
monk2	6	432	2	1.1598	13	(419/2/1/*)	0,35	0,52
bupa	6	345	2	1.8581	6	(319/18/2/2/2/2)	0,39	0,58
cleveland	13	297 (303)	5	2.7749	3	(293/3/1)	0,00	0,53
spectheart	44	267	2	1.1451	2	(266/1)	0,18	0,79
housevotes	16	232 (435)	2	1.3794	5	(174/21/17/12/8)	0,48	0,50

* diversos grupos com o mesmo número de instâncias.

5.1.2 Condição de Parada: 1000 Iterações

Neste experimento, devido a novas experimentações e ao tempo desprendido no experimento de 100 não trocas de rótulo consecutivas da instância-alvo, foi estabelecido que o número de 1000 iterações, independente do número de trocas de rótulo, principalmente devido as características do conjunto de *datasets* do experimento (número de instâncias, número de atributos), resultaria nos mesmos ou melhores resultados em comparação aos resultados obtidos anteriormente. Na Tabela 22 observa-se os resultados finais para cada *dataset* (a Tabela 22 têm a mesma estrutura da Tabela 21). Além disso, observa-se que, mesmo com a mudança da condição de parada, não ocorreu nenhum tipo de melhora nos resultados. Em alguns casos como, por exemplo, do *dataset balance* encontrou-se *129* sendo que, originalmente, o mesmo possui *3* grupos.

Tabela 22: Metodologia Aglomerativa com Árvores de Decisão - Condição de Parada: 1000 e Porcentagem Vizinhança: 10%.

dataset	instâncias	classes	dbi	grupos	instâncias por grupo
sonar	208	2	1.02	4	(205/1/1/1)
zoo	101	7	?	10	(58/11/16/2/2/4/2/2/2/2)
hamditolga	403	4	?	51	(314/18/14/3/3/2/2/2/2/2/1/*)
balance	625	3	1.634	129	(463/13/12/6/3/2/2/2/2/1/*)
saheart	462	2	1.83	71	(226/66/47/21/8/7/4/4/4/3/3/3/2/2/2/2/2/2/2/2/1/*)
monk2	432	2	1.38	48	(379/3/3/2/2/1/*)
bupa	345	2	1.769	41	(120/74/71/20/6/5/3/3/2/2/2/2/2/2/2/2/2/2/1/*)
cleveland	297 (303)	5	1.84	28	(234/14/8/6/4/3/3/3/2/2/1/*)
spectheart	267	2	1.76	9	(257/3/1/1/1/1/1/1/1)
housevotes	232 (435)	2	1.147	12	(156/24/18/9/7/5/4/3/2/2/1/1)

* diversos grupos com o mesmo número de instâncias.

5.1.3 Condição de Parada: 1000 Iterações considerando o valor do próximo rótulo e do próximo não rótulo

Nos experimentos anteriores observou-se que utilizar as condições de *não-agrupamento* obtidas durante o processo de eleição poderiam também ser válidas, devido ao fato de que, o conhecimento adquirido mesmo em não agrupar um *par* de instâncias poderia ser considerado também uma forma alternativa de conhecimento ou aprendizagem. Nos experimentos anteriores, a informação de *não-agrupar* foi desconsiderada e somente os resultados de eleição positivo (ou verdadeiro) foram utilizados para a determinação de novo possível rótulo da instância-alvo. Neste conjunto de experimentos, a determinação do novo possível rótulo da instância-alvo eliminou, dentre os possíveis rótulos, aquele que dificilmente poderia vir a ser o novo rótulo, pois o mesmo é rótulo da maioria das instâncias testadas no processo de eleição que tiveram resultado desfavorável. Por exemplo: o método que elege o possível novo rótulo recebe, além do vetor de vizinhos, o rótulo que está menos presente em sua periferia, eliminando-o do conjunto dos possíveis rótulos. As Tabelas 23 e 24 descrevem os resultados obtidos para cada *dataset* tendo como bases de conhecimento todas as árvores ou somente as derivadas do próprio *dataset teste*. Nas tabelas, observa-se que, dependendo do *dataset*, a utilização de somente suas próprias bases derivadas de conhecimento foi benéfica e para outros foi justamente o oposto. Independente do conjunto de bases de conhecimento os resultados foram insatisfatórios, possivelmente devido a falhas da metodologia e/ou ao desbalanceamento entre rótulos dos pares de instância das bases auxiliares de conhecimento. Além disso, observa-se que, para os *datasets saheart*, *spectheart* e *monk2*, os grupos mantiveram a maioria das instâncias em um único grupo, enquanto que os demais grupos tiveram apenas *1* ou duas instâncias.

Tabela 23: Metodologia Aglomerativa com árvores de decisão (todas) - modificada - Condição de Parada: 1000 e Porcentagem Vizinhança: 10%.

dataset	instâncias	classes	dbi	grupos	instâncias por grupo
sonar	208	2	1.9871	3	(205/2/1)
zoo	101	7	1.55	12	(25/18/17/15/5/5/4/4/2/2/2/2)
hamditolga	403	4	2.16	93	(64/50/43/20/15/12/11/8/7/6/*5/4*3*/2*/1*)
balance	625	3	?	168	(349/22/18/11/7/7/6*5/3/*2/*1/*)
saheart	462	2	?	11	(257/1/1/1/1/1/1/1/1/1/1)
monk2	432	2	1.1941	57	(369/3/2/2/2/2/2/1/*)
bupa	345	2	2.7	83	(28/24/23/16/15/10/*9/8/*7*/6*5/*4/*3/*2/*1/*)
cleveland	297 (303)	5	3.45	71	(17/17/13/13/12/11/9/9/8/7/7/7/6/*5/*4/*3/*2/*1/*)
spectheart	267	2	1.17	11	(257/1/1/1/1/1/1/1/1/1/1)
housevotes	232 (435)	2	1.33	23	(127/17/14/14/10/8/6/6/4/4/3/3/3/2/2/2/1/*)

* diversos grupos com o mesmo número de instâncias.

Tabela 24: Metodologia Aglomerativa com árvores de decisão (somente as suas) - modificada - Condição de Parada: 1000 e Porcentagem Vizinhança: 10%.

dataset	instâncias	classes	grupos	instâncias por grupo	dbi
sonar	208	2	13	(194/2/2/1/1/1/1/1/1/1/1/1/1)	1.54
zoo	101	7	17	(24/14/12/11/7/6/5/4/4/4/3/2/1/1/1/1/1)	1.25
hamditolga	403	4	105	(26/26/24/16/14/14/11/8/7/7/6/*5/*4/*/3/*2*/1*)	2.26
balance	625	3	156	(441/11/6/4/3/3/2/3/2/*1/*)	?
saheart	462	2	70	(369/15/3/3/2/2/2/2/1/*)	2.32
monk2	432	2	51	(373/4/1/*)	1.34
bupa	345	2	30	(312/2/2/2/2/1/*)	1.43
cleveland	297 (303)	5	282	(5/3/3/3/3/2/2/2/1/*)	2.19
spectfheart	267	2	14	(254/1/*)	1.28
housevotes	232 (435)	2	26	(97/43/12/11/9/8/8/4/*3/3/3/2/1/*)	1.6294

* diversos grupos com o mesmo número de instâncias.

5.1.4 Problema de Desequilíbrio e a Amostragem das bases de conhecimento

Os resultados, independente da condição de parada, não foram satisfatórios, possivelmente devido ao problema de desequilíbrio entre as classes (*true* e *false*) das bases de conhecimento (Chawla, 2005) (Tan et al., 2005). Nas bases de conhecimento há um número maior de pares com rótulo/classe *false* em comparação com os pares *true*. O maior número de pares *false* foi observado em virtude de existir uma maior quantidade de instâncias que não possuem o mesmo rótulo (ou classe) em relação aos pares que têm o mesmo rótulo. Os rótulos *true* e *false* determinam se as instâncias que compõem os pares das bases de conhecimento possuíam ou não o mesmo rótulo quando ainda não tinham sido pareadas (etapa de pré-processamento). Assim, cogitou-se a utilização de um número balanceado de pares com rótulo *true* e *false*, pois uma hipótese do insucesso dos experimentos anteriores poderia ser devido ao desbalanceamento entre o número de pares de cada classe (ou rótulo). O desbalanceamento poderia ser um obstáculo para obtenção de uma base de conhecimento de qualidade.

5.2 Método Aglomerativo baseado em Regressão Logística

Analisando os resultados obtidos do experimento realizado com o conjunto de *datasets* utilizando árvores de decisão, buscou-se uma segunda opção de extração de conhecimento. A regressão logística mostrou-se ser uma boa opção já que, para qualquer *dataset*, é preciso somente realizar um conjunto de operações aritméticas para incorporá-la ao processo de análise dos pares (ou instâncias sintéticas). Assim, o *parser* do processo de agrupamento que anteriormente decidia se um par de instâncias deveria ou não estar agrupado em um mesmo grupo mediante consultas às árvores de decisão foi desconsiderado. Um outro ponto importante foi eliminar a etapa que determinava o possível novo rótulo da instância-alvo. A eliminação da etapa provocou a exclusão do parâmetro que definia a quantidade de instâncias vizinhas a ser utilizada.

5.2.1 Funcionamento

Para obedecer a exigência e construir um processo que continuasse permitindo a transferência de conhecimento, foi construído um grande *dataset* único, que continha todos os *datasets transformados* do conjunto de bases de conhecimento. Assim, para o método de regressão, há apenas um *dataset* e, para a transferência de conhecimento, o método de agrupamento poderia continuar usufruindo de todo conhecimento. Deste modo, elaborou-se um processo de amostragem continha o objetivo de balancear o número de pares *true* e *false* de todas as bases de conhecimento. A amostragem foi realizada através da distância euclidiana entre as instâncias. Assim, deixaria de existir uma discrepância geométrica que pudesse impedir um bom resultado de agrupamento. Além disso, para *datasets* esparsos, ou seja, para *datasets* com instâncias bem afastadas uma das outras, o processo de agrupamento proposto poderia encontrar dificuldades, em virtude de que as bases de conhecimento poderiam ser de instâncias mais compactas, resultando em árvores de decisão e atributos sintéticos insuficientes para os *datasets* esparsos, ou seja, árvores de decisão que não auxiliariam no agrupamento porque teriam atributos sintéticos discrepantes com os atributos do *dataset de teste transformado*. Assim, cada *dataset* foi dividido em 100 intervalos e contabilizou-se a quantidade acumulada de pares *true* e *false*. Logo, foi possível determinar o ponto de cruzamento entre as curvas acumuladas. Assim, somente os pares que tinham valores de distância menores que o valor do ponto de cruzamento foram selecionados para compor as *subamostras* de cada base de conhecimento. O *dataset único/global* é o resultado de todas as *subamostras*. Concluída a etapa de amostragem, o *dataset único* foi encaminhado para o método de regressão logística.

O processo de regressão logística tem como resultado um conjunto de coeficientes que são utilizados na função de ligação logística e no método proposto, em substituição ao processo de eleição, com objetivo de determinar se um par deve ou não ficar em um mesmo grupo. Caso positivo, o rótulo da segunda instância do par é modificado.

Para a proposta de agrupamento com regressão logística utilizou-se 5 *datasets*: *compound*, *flame*, *spiral*, *jain* e *pathbased*. A Tabela 25 ilustra a quantidade de instâncias, os atributos e a quantidade de classes de cada *dataset* do conjunto.

Tabela 25: Datasets utilizados na metodologia baseada em regressão logística.

dataset	instâncias	atributos	classes
Compound	399	2	6
Pathbased	300	2	3
Spiral	312	2	3
Jain	373	2	2
Flame	240	2	2

5.2.2 Regressão Logística: Nova metodologia

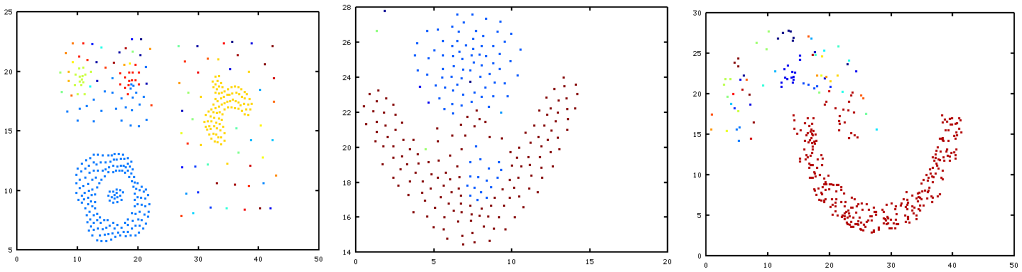
Independente da metodologia e do método *classificador* utilizado até o momento, os resultados não foram satisfatórios. Assim, combinou-se o melhor das opções testadas até o momento: a escolha aleatória da instância-alvo (explorando a localidade espacial) e o método de regressão logística.

A escolha aleatória da instância-alvo não tenciona o início do processo de agrupamento a nenhuma instância em especial. Assim, não há vícios no início do processo de agrupamento (ex: começar sempre pela primeira instância). Além disso, dentre as alternativas propostas de base de conhecimento, o método de regressão foi o que obteve um melhor desempenho comparado com o classificador que utilizava árvores de decisão. Adicionado à isso, o agrupamento por regressão logística não necessita nenhum *parser textual*. O *parser* percorria as árvores de decisão (armazenadas em modo texto) e processava todo o processo de eleição.

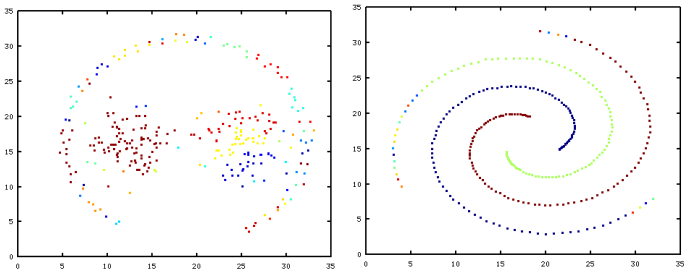
Agregado à escolha aleatória e o método de regressão, explorou-se ainda a localidade espacial entre vizinhos. Essa característica é explorada em pares de instâncias onde a função logística de regressão determina a permanência do *par* em um mesmo grupo, ou seja, se o processo de agrupamento acertou, o mais recomendável é que continue na periferia das instâncias que já agrupou anteriormente. Deste modo, elimina-se a necessidade de uma determinação aleatória de uma nova instância alvo para a iteração seguinte, pois o processo de agrupamento continua com a instância vizinha. Em contra-partida, a determinação aleatória permanece em caso negativo, ou seja, em decisões de não agrupamento do *par* selecionado. Assim, o método proposto pode gerar resultados diferentes através execuções diferentes.

A decisão de *agrupar* ou *não agrupar* em um mesmo grupo o *par* funciona mediante a comparação do valor da função de regressão logística calculado para o par de instâncias em relação a média de todos os valores calculados. O processo deve agrupar instâncias caso seu valor seja maior ou igual a média dos valores. A Figura 34 ilustram graficamente

os resultados obtidos para o conjunto de 5 *datasets* testados. Os *datasets* apresentaram regiões de alta densidade, ou seja, regiões onde instâncias localmente próximas tinham o mesmo rótulo. Essas regiões de alta densidade são indícios de bons resultados de agrupamento.



(a) Compound (b) Flame (c) Jain

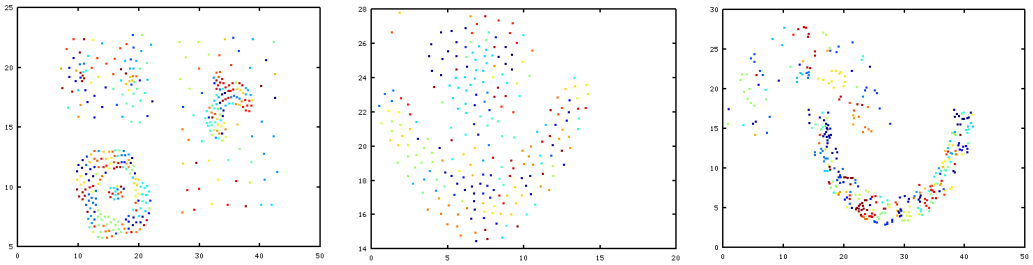


(d) PathBased (e) Spiral

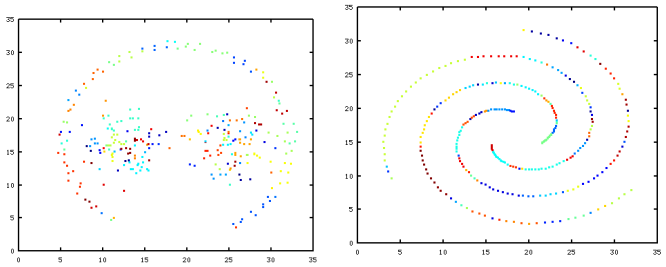
Figura 34: Resultados - Metodologia - *classificador* Regressão Logística - *SEM* a própria base participando do conjunto de bases de conhecimento.

Além disso, elaborou-se também um novo experimento que realizava o procedimento de *agrupar* ou *não-agrupar* de forma aleatória, ou seja, desconsiderando o valor obtido por meio da função logística e fixando o limiar decisório em 0.5. A Figura 35 ilustra os resultados obtidos. Neste caso, houve uma piora acentuada nos resultados, possivelmente, devido a inexistência de alguma forma de conhecimento anterior. Além disso, nenhum *dataset* testado apresentou regiões de alta densidade, ou seja, aglomerados de instâncias de mesmo rótulo (na figura rótulos iguais possuem a mesma cor). Deste modo, existiu somente grupos esparsos e, por consequência, piores resultados de agrupamento.

Na Tabela 26 têm-se os resultados das métricas de validação para cada *dataset* do experimento (com ou sem o próprio *dataset* como base de conhecimento). Na Tabela 26, observa-se pouca variação nos resultados quando se inclui ou se exclui o *dataset de teste transformado* das demais bases de conhecimento (as exceções são os *datasets flame* e *spiral*). Na Tabela 27 há também os resultados para o experimento proposto com modo aleatório de agrupamento. Nessa tabela, observa-se o mesmo que graficamente já foi mostrado, ou seja, uma piora nos resultados.



(a) Compound (b) Flame (c) Jain



(d) PathBased (e) Spiral

Figura 35: Resultados - Metodologia - *classificador* Regressão Logística - *SEM* a própria base como base de conhecimento - Decidindo *agrupar* ou *não-agrupar* de modo aleatório.

Tabela 26: Datasets e os valores de métricas obtidos.

Dataset	Base de Conhecimento	Instâncias	DBI	silhoutte
flame	sem sua própria base de conhecimento	240	0.74552	0.85696
flame	com sua própria base de conhecimento	240	1.1918	0.81212
jain	sem sua própria base de conhecimento	373	0.59846	0.91011
jain	com sua própria base de conhecimento	373	0.58278	0.86909
path	sem sua própria base de conhecimento	300	0.50198	0.87197
path	com sua própria base de conhecimento	300	0.84454	0.89286
compound	sem sua própria base de conhecimento	399	0.44969	0.90743
compound	com sua própria base de conhecimento	399	0.44958	0.94506
spiral	sem sua própria base de conhecimento	312	1.0617	0.88937
spiral	com sua própria base de conhecimento	312	1.395	0.78009

Tabela 27: Datasets e os valores de métricas obtidos - Decisão aleatória de agrupar.

dataset	com ou sem a própria base de conhecimento	dbi	silhoutte
flame	com a própria a base de conhecimento	0.7904	0.71221
	sem a própria a base de conhecimento	0.74501	0.73071
jain	com a própria a base de conhecimento	0.62628	0.76793
	sem a própria a base de conhecimento	0.54009	0.74836
path	com a própria a base de conhecimento	0.5575	0.76983
	sem a própria a base de conhecimento	0.61068	0.74867
compound	com a própria a base de conhecimento	0.65123	0.74751
	sem a própria a base de conhecimento	0.79716	0.71007
spiral	com a própria a base de conhecimento	0.40475	0.73299
	sem a própria a base de conhecimento	0.42467	0.74051

Mediante os resultados obtidos, mostra-se interessante a proposta de agrupar conjuntos de dados através de bases auxiliares. Além disso, observa-se que a ideia de mapear para um mesmo domínio todas as bases envolvidas no processo de agrupamento funciona como um artifício para incitar o processo de transferência de conhecimento.

6 CONCLUSÕES E TRABALHOS FUTUROS

Nesta dissertação de mestrado foi apresentado um novo método de agrupamento de dados que aplica os conceitos de *Transfer-Learning*. A transferência de conhecimento é obtida através do mapeamento dos atributos originais dos *datasets* envolvidos em novos atributos (atributos sintéticos) e de bases de conhecimentos. As bases de conhecimento funcionam como bases auxiliares que favorecem o agrupamento de um novo *dataset*. Deste modo, a arquitetura de agrupamento desenvolvida contemplou o objetivo inicial proposto. Os resultados obtidos demonstram a importância da continuidade do estudo de técnicas alternativas de agrupamento de dados que usufruem de algum conhecimento prévio como forma de auxílio para o processo de agrupamento. Além disso, o projeto derivou as seguintes publicações:

- XII Mostra de Produção Universitária - XV Encontro de Pós-Graduação - Universidade Federal do Rio Grande - MPU/FURG - 2013 com o trabalho intitulado *Um Método de Agrupamento de Dados por Transferência*. O trabalho reporta a ideia inicial do algoritmo e as observações dos membros da banca foram úteis para a elaboração de novas ideias para o trabalho.

- ENIAC 2013 com o trabalho intitulado *A Transfer-learning Approach for Data Clustering* - rejeitada. As revisões reportadas foram importantes para a construção da nova metodologia independe do algoritmo DBSCAN.

- Em paralelo ao desenvolvimento da dissertação, ocorre o processo de escrita de um artigo para um periódico internacional.

Além dos experimentos realizados durante o período de mestrado, poderiam ser também investigados os seguintes tópicos:

- Realizar novos experimentos com a metodologia que obteve os melhores resultados e comparar a proposta com as demais encontradas na literatura.

- Propor mecanismos de avaliação individual de cada atributo sintético.

- Desenvolver novos atributos sintéticos que possam agregar qualidade à metodologia.
- Construir mecanismos de implementação que melhorem o desempenho da etapa de *Pré-Processamento* e de construção da matriz resultante da função logística.
- Elaborar e submeter outros artigos para conferências que tratam de *Machine Learning* e *clustering*.

REFERÊNCIAS

Barbakh, W. A., Wu, Y., and Fyfe, C. (2009). Review of clustering algorithms. *Studies in Computational Intelligence*, 249:7–28.

Chawla, N. (2005). Data mining for imbalanced datasets: An overview. In Maimon, O. and Rokach, L., editors, *Data Mining and Knowledge Discovery Handbook*, pages 853–867. Springer US.

Cover, T. and Thomas, J. (1991). *Elements of information theory*. Wiley, New York.

Dai, W., Yang, Q., Xue, G.-R., and Yu, Y. (2008). Self-taught clustering. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 200–207, New York, NY, USA. ACM.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1(2):224–227.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231.

Farrow, C. L., Shaw, M., Kim, H., Juhás, P., and Billinge, S. J. L. (2011). Nyquist-shannon sampling theorem applied to refinements of the atomic pair distribution function. *Phys. Rev. B*, 84:134105.

Freeman, E. and Freeman, E. (2007). *Use a Cabeça! Padrões de Projetos (Design Patterns) - 2ª Ed. Revisada*. Alta Books.

Hartigan, J. A. and Wong, M. A. (1979). A K-means clustering algorithm. *Applied Statistics*, 28:100–108.

Kamishima, T. and Motoyoshi, F. (2003). Learning from cluster examples. In *Mach. Learn.*, volume 53-3, pages 199–233, Hingham, MA, USA. Kluwer Academic Publishers.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):pp. 79–86.

Luz, D. (2003). *Implementação de uma Ferramenta de Data Mining para o Auxílio à Tomada de Decisão - Caso de uma Cadeia de Suprimentos*. Trabalho de Conclusão de Curso - Engenharia de Controle e Automação Industrial, Universidade Federal de Santa Catarina - UFSC, Santa Catarina, SC, Brasil.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In Cam, L. M. L. and Neyman, J., editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press.

Massey, F. J. (1951). The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78.

Nasution, M. K. M. (2012). Kolmogorov complexity: Clustering objects and similarity. *CoRR*, abs/1207.6188.

Nunes, B. P. (2009). *Classificação Automática de Dados Semi-Estruturados*. PUC-RIO - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, Brasil.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Rosales, R., Achan, K., and Frey, B. J. (2004). Learning to Cluster using Local Neighborhood Structure. *International Conference on Machine Learning (ICML)*.

Rousseeuw, P. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65.

Shao, J., Tanner, S. W., Thompson, N., and Cheatham, T. E. (2007). Clustering molecular dynamics trajectories: 1. characterizing the performance of different clustering algorithms. *Journal of Chemical Theory and Computation*, 3(6):2312–2334.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Xu, R. and Wunsch, D., I. (2005). Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678.