UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RODOLFO JOSÉ DE OLIVEIRA SOARES

sPerC: ***Um Classificador Baseado em Perturbação para Dados Multimodais***

Recife

2020

RODOLFO JOSÉ DE OLIVEIRA SOARES

sPerC: *Um Classificador Baseado em Perturbação para Dados Multimodais*

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Inteligência computacional

Orientador (a): George Darmiton da Cunha Cavalcanti

Coorientador (a): Edson Leite Araújo

Recife

2020

Catalogação na fonte
Bibliotecária Mariana de Souza Alves CRB4-2105

S676s Soares, Rodolfo José de Oliveira
sPerC: um classificador baseado em perturbação para dados multimodais/ Rodolfo José de Oliveira Soares. – 2020.
75f., il., fig., tab.

Orientador: George Darmiton da Cunha Cavalcanti.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da computação, Recife, 2020.
Inclui referências e apêndices.

1. Inteligência computacional. 2. Perturbações. 3. Multimodalidade. 4. Agrupamento de dados. I. Cavalcanti, George Darmiton da Cunha. (orientador) II. Título.

006.31 CDD (22. ed.) UFPE-CCEN 2021-06

Rodolfo José de Oliveira Soares

"sPerC: Um Classificador Baseado em Perturbação para Dados Multimodais"

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 07/10/2020.

BANCA EXAMINADORA

_____

Prof. Dr. Hansenclever de França Bassani
Centro de Informática/ UFPE

_____

Prof. Dr. Thiago Oliveira dos Santos
Departamento de Informática / UFES

_____

Prof. Dr. George Darmiton da Cunha Cavalcanti
Centro de Informática / UFPE
(Orientador)

Dedico este trabalho a minha família e minha noiva que foram porto seguro perante as dificuldades durante este percurso.

AGRADECIMENTOS

Primeiramente, agradeço a minha família, meu pai Reinaldo, minha mãe Mércia, e meus irmãos Ronald e Ruan, por todo apoio e motivação durante todo esses anos. À Vitória, minha noiva, que esteve comigo lado a lado nessa jornada, mostrando ser companheira para as horas fáceis e difíceis da vida. Agradeço também ao meu orientador, Prof. George Darmiton, pela paciência, confiança, compreensão, apoio e pelas inúmeras discussões que direcionaram o desenvolvimento da pesquisa. Agradeço aos meus amigos, em especial Bruno Oliveira, Carlos Fonsêca, José Henrique Davino, Hugo Rodrigues e Milton Gama, pelo companheirismo e apoio que ofertaram durante o mestrado, além das várias descontrações promovidas. Por fim, a todos aqueles que de alguma forma ajudaram nesta caminhada.

RESUMO

Na literatura, a teoria bayesiana é amplamente utilizada como base para a geração de novos modelos supervisionados, dada sua robustez em diversos contextos. Entre as diferentes técnicas embasadas no modelo estatístico, o classificador baseado em perturbações (PerC, do inglês ***Per****turbation-based* ***C****lassifier*) utiliza as alterações ocorridas nos parâmetros $\hat{\mu}$ e $\hat{\Sigma}$, chamadas de $\Delta\hat{\mu}$ e $\Delta\hat{\Sigma}$, para rotular novas amostras, tendo seu desempenho comprovado em vários cenários. Entretanto, quando o PerC é submetido a problemas que seguem uma distribuição multimodal, sua performance, e seu poder de generalização, degradam. Neste contexto, a multimodalidade pode ser observada quando exemplos de uma mesma classe formam grupos (*clusters*) dispersos no espaço de características. Assim, a multimodalidade aumenta a complexidade dos dados, reduzindo o nível de discernimento dos vetores médios e matrizes de covariâncias estimadas pelo classificador PerC. Para preencher essa lacuna, este trabalho apresenta uma nova abordagem de classificação para o algoritmo PerC, intitulada sPerC (do inglês, ***s****ubconcept* PerC), no qual utiliza o algoritmo *K-Means* para particionar os dados do problema em *clusters*, fornecendo-os como entrada para a técnica PerC, com o intuito de aprimorar o poder de generalização do modelo. A partir da combinação de 4 diferentes *clustering validation* índices, o valor do parâmetro K é estimado para o algoritmo *K-Means*. Vale destacar que nenhum conhecimento prévio, sobre os domínios em estudo, é levado em consideração ao definir o valor do parâmetro. Experimentos foram conduzidos usando 30 bases de dados disponíveis nos repositórios KEEL e *UCI Machine Learning*, comparando o desempenho do método proposto em relação as técnicas PerC (versão original), Árvore de Decisão, k-NN, *Multilayer Perceptron* (MLP), *Naïve Bayes*, *Random Forest* e *Support Vector Machine* (SVM). Os resultados demonstraram a eficácia do trabalho proposto, alcançando desempenho competitivos em relação aos métodos *Random Forest* e Árvore de Decisão, e sendo significativamente superior aos algoritmos PerC, MLP e SVM, segundo os testes estatísticos de Wilcoxon e Friedman. Por fim, 22 medidas de complexidade foram adotadas na extração de características das bases de dados e utilizadas na construção de um *meta-learning dataset* para descrever os cenários favoráveis para a aplicação do método sPerC, a partir de um meta-classificador.

Palavras-chaves: Perturbações. Multimodalidade. Agrupamento de dados. Validação de *clusters*. Classificação.

ABSTRACT

Bayesian theory is widely used as a basis for generation of new supervised models, due to its robustness in different contexts. Among the different techniques based on the statistical model, the **Per**turbation-based **C**lassifier (PerC) uses the changes in the parameters $\hat{\mu}$ and $\hat{\Sigma}$, called $\Delta\hat{\mu}$ and $\Delta\hat{\Sigma}$, to label new samples. The PerC classifier has had its performance proven in several scenarios. However, when PerC is subjected to problems with multimodal distribution, its performance, and its power of generalization, degrades. In this context, multimodality can be observed when samples of the same class form clusters dispersed in the feature space. Thus, multimodality increases data complexity, reducing the level of discernment of the mean vectors and matrices of covariance estimated by the PerC classifier. To fill this gap, this paper presents a new classification approach for the PerC algorithm, namely sPerC (subconcept PerC), that uses the K-Means algorithm to partition the problem data in clusters and provide them as inputs to the PerC technique aiming to improve the generalization power of the model. From the combination of 4 different clustering validation indexes, the value of the K parameter is estimated for the K-Means algorithm. It is worth noting that no prior knowledge about the study's domains is taken into consideration when setting the parameter value. Experiments were conducted using 30 datasets available in KEEL and UCI Machine Learning repositories to compare the performance of the proposed method with the PerC techniques (original version), Decision Tree, k-NN, Multilayer Perceptron (MLP), Naïve Bayes, Random Forest, and Support Vector Machine (SVM). The results of the experiments show the effectiveness of the proposed work as it achieved competitive performance in comparison to Random Forest and Decision Tree methods, and it was significantly superior to PerC, MLP, and SVM algorithms, according to Wilcoxon and Friedman statistical tests. Finally, 22 complexity measures were applied in the extraction of characteristics from the datasets and used in the construction of a meta-learning dataset to describe the favorable scenarios for the application of the sPerC method from a meta-classifier.

Keywords: Perturbation. Multimodality. Clustering. Clustering validation. Classification.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

b_j	*Index* limite entre os *large* e *small clusters* da classe j
f_j	Conjunto de classificadores PerC da classe j
f_{ji}	*i*-ésimo classificador PerC da classe j
G_j	Conjunto de partições gerados a partir do subconjunto Γ_j
G_j^*	Conjunto de partições gerados a partir do subconjunto Γ_j resultado da remoção dos *small clusters*.
h	Número de atributos presentes na base de dados Γ
j	Índice da classe
K	Parâmetro do algoritmo *K-Means*
n	Número de amostras presentes na base de dados Γ
m	Número de classes presentes na base de dados Γ
x_j	Amostra de treinamento da classe j
Γ	Conjunto de dados de treinamento
Γ_j	Subconjunto de treinamento pertencente à classe j
μ_j	Vetor médio da classe j
$\hat{\mu_j}$	Estimativa do vetor média da classe j
$\Delta\hat{\mu_j}$	Perturbação ocorrida no vetor médio da classe j
Σ_j	Matriz de covariância da classe j
$\hat{\Sigma_j}$	Estimativa da matriz de covariância da classe j
$\Delta\hat{\Sigma_j}$	Perturbação ocorrida na matriz de covariância da classe j

SUMÁRIO

1 INTRODUÇÃO

Em aprendizagem de máquina, o reconhecimento de padrões é uma abordagem utilizada para tomar decisões futuras a partir do conhecimento presente nos dados (BISHOP, 2006). Comumente, os padrões pertencem a diferentes classes (conceitos) do problema e quando analisados em conjunto, constituem regiões de separação no espaço de características. Posto isto, os modelos de aprendizado são construídos com o objetivo de mapear o conjunto de características (padrão) em uma determinada classe, assumindo que objetos de um mesmo grupo compartilham características em comum (DUDA; HART; STORK, 2012; FUKUNAGA, 2013; ARAÚJO; CAVALCANTI; REN, 2020). Portanto, as máquinas de aprendizagem são modelos preditivos que buscam aprender os conceitos existentes com base nas informações disponíveis.

Sendo uma das alternativas para modelar problemas de reconhecimento de padrões, a *teoria bayesiana* utiliza conceitos estatísticos com a finalidade de produzir classificadores, formulando a tomada de decisão em termos probabilísticos. Tais modelos buscam minimizar o erro de classificação, quantificando o custo/benefício entre diferentes decisões de classificação e o seu custo associado (BISHOP, 2006).

Em geral, os métodos baseados na abordagem estatística utilizam a função de densidade de probabilidade (*fdp*), estimada a partir dos exemplos, para inferir as fronteiras de separação das diferentes classes existentes em um conjunto de dados. Contudo, estimar a função de densidade de probabilidade das classes é um problema complexo, devido à sua dependência do número de amostras pré-classificadas (em geral pequeno), e da dimensionalidade do vetor de características (em geral alta). Como solução da problemática explanada, a suposição da normalidade dos dados é visto como uma estimativa eficaz. A distribuição normal é parametrizada por um vetor médio, μ, e uma matriz de covariância, Σ. Portanto, para estimar a *fdp*, faz-se necessário estimar os parâmetros μ e Σ que são desconhecidos.

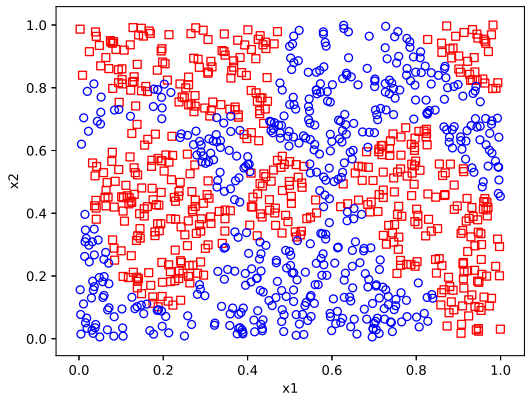
Entre as diversas aplicações baseadas na abordagem estatística, o trabalho proposto em (ARAÚJO, 2017; ARAÚJO; CAVALCANTI; REN, 2020), intitulado *Um classificador baseado em perturbações* (PerC, do inglês, ***Per****turbation-based* ***C****lassifier*), foi desenvolvido com o objetivo de avaliar as perturbações, denominadas $\Delta\hat{\mu}$ e $\Delta\hat{\Sigma}$, provocadas nas estimativas $\hat{\mu}$ e $\hat{\Sigma}$ após a inserção de uma amostra de teste no conjunto de treinamento. Seu desempenho é mensurado em vários cenários, demonstrando ser um classificador bastante eficiente. Entretanto, seu desempenho acaba sendo degradado quando submetido a dados que seguem uma distribuição

multimodal.

A ocorrência da multimodalidade nos dados pode ser observada quando objetos de uma mesma classe formam grupos (*clusters*), também conhecidos como subconceitos (SHARMA; SOMAYAJI; JAPKOWICZ, 2018), dispersos no espaço de características (TAHERI et al., 2019). No mundo real, a classificação de dígitos em números pares e ímpares, ou até mesmo no diagnóstico de doenças, dada a existência de diversas causas para uma mesma enfermidade, são exemplos de aplicações nas quais a multimodalidade pode ser encontrada. A multimodalidade também pode aparecer quando problemas de classificação multiclasses são resolvidos aplicando a abordagem conhecida como "um-contra-todos" (OVA, do inglês, *one-vs-all*) (WANG et al., 2009; SUGIYAMA, 2007; JR; CAVALCANTI; REN, 2016; SUGIYAMA, 2006).

Para analisar o comportamento do classificador PerC, o banco de dados sintético *P2 Problem* (VALENTINI, 2005) foi utilizado. Tal *dataset* é definido como um problema bidimensional composto por 2 classes, multimodais, delimitadas por funções polinomiais e trigonométricas. Foram geradas 1.000 instâncias, com 500 exemplos em cada classe. A Figura 1 apresenta a distribuição dos dados.

Figura 1 – Conjunto de dados, *P2 Problem*, gerado artificialmente.



Fornecendo a base de dados *P2 Problem* como entrada para o classificador PerC, as estimativas $\hat{\mu}$ e $\hat{\Sigma}$ são determinadas, encontrando os seguintes valores:

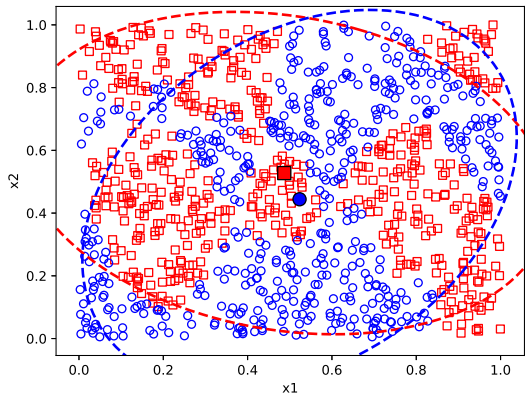
$$\hat{\mu_1} = (0,486955; 0,527565), \quad \hat{\mu_2} = (0,523176; 0,444084)$$

e

$$\hat{\Sigma_1} = \begin{bmatrix} 0,098865 & -0.015138 \\ -0,015138 & 0,066025 \end{bmatrix}, \quad \hat{\Sigma_2} = \begin{bmatrix} 0,066346 & 0,025245 \\ 0,025245 & 0,091052 \end{bmatrix}$$

A Figura 2 apresenta os pontos (■) e (●) que representam, respectivamentes, os vetores médios $\hat{\mu_1}$ e $\hat{\mu_2}$. As elipses de confiança são determinadas com 95% de certeza, e para estas elipses a distância $|\hat{\mu_1} - \hat{\mu_2}|$, entre os vetores médios, foi computada utilizando a distância euclidiana, obtendo o valor $|\hat{\mu_1} - \hat{\mu_2}| = 0{,}091$. Já para as matrizes de covariâncias $\hat{\Sigma_1}$ e $\hat{\Sigma_2}$, a norma de frobenius foi utilizada para computar a diferença entre as matrizes, alcançando o valor $|\hat{\Sigma_1} - \hat{\Sigma_2}| = 0{,}070$.

Figura 2 – Execução do algoritmo PerC, sobre o *dataset P2 Problem*. Elipses tracejadas delimitam a região de confiança com 95% de certeza.



Como podemos observar na Figura 2, devido à presença de vários subconceitos dispersos no espaço de características, o modelo PerC alcança um baixo grau de generalização para cada uma das classes existentes, uma vez que os vetores médios ($|\mu_1 - \mu_2| = 0{,}091$), assim como suas matrizes de covariâncias ($|\Sigma_1 - \Sigma_2| = 0{,}070$), possuem baixo nível de discernimento. Dessa forma, o alto grau de complexidade contido no domínio do problema, acaba afetando o desempenho do algoritmo PerC.

1.1 MOTIVAÇÃO

Em problemas multimodais, as classes do domínio são segregadas em vários subconceitos, aumentando a complexidade do dados e, consequentemente, afetando a capacidade de gerar classificadores efetivos (LIU et al., 2020; SHARMA; SOMAYAJI; JAPKOWICZ, 2018).

De acordo com Sharma, Somayaji e Japkowicz (2018), na presença de subconceitos nos dados, os classificadores são mais eficazes quando treinados sobre cada grupo, em vez da classe como um todo. Como visto anteriormente, a difícil separabilidade dos vetores médios, assim como das matrizes de covariâncias, influênciam diretamente o desempenho do algoritmo

PerC. Visando aumentar o grau de discernimento dos dados, buscando encontrar associações baseadas nas distribuições das amostras, técnicas de agrupamentos de dados podem ser empregadas com o objetivo de encontrar os subconceitos (*clusters*) existentes no dados (LIU et al., 2020; SHARMA; SOMAYAJI; JAPKOWICZ, 2018; LIU; JAPKOWICZ; WANG, 2020).

Métodos baseado em *clusters* constroem grupos de objetos com base na semelhança (ou diferença) entre eles de tal maneira que os grupos obtidos sejam os mais homogêneos e separados. Tais grupos, ou partições, podem ser definidos como um aglomerado de dados que possuem coesão interna e isolamento externo. Usualmente as partições são construídas otimizando um critério de adequação entre as classes e seus representantes. Portanto, deseja-se encontrar, entre todas as possíveis partições, aquela que otimize determinado critério definido a *priori*.

Na literatura, é possível encontrar diversos algoritmos baseados nas abordagens de agrupamentos (ROKACH; MAIMON, 2005), como: Algoritmo hierárquico aglomerativo, Algoritmo de transferência, Algoritmo de Centro móveis, baseados em lógica *Fuzzy*, etc. Dentre eles, o método mais empregado é o *K-Means* (MACQUEEN, 1967; LLOYD, 1982; FORGY, 1965; ARTHUR; VASSILVITSKII, 2007). O *K-Means* é um algoritmo iterativo que busca associar instâncias num determinado número K, pré-definido, de grupos (*clusters*), definindo representantes (centróides) para cada uma das partições, minimizando a soma de todas as distâncias euclidianas entre cada amostra e seu respectivo centróide, segundo o critério dos mínimos quadrados.

Cada partição gerada representa uma característica do problema e a quantidade de amostras presentes em cada grupo varia dependendo do domínio em estudo. Nesse contexto, alguns *clusters* podem ter uma baixa representatividade nos dados (*small clusters*), sendo afetados pelo problema, conhecido na literatura como, *small disjuncts problem* (HOLTE; ACKER; PORTER, 1989; WEISS, 2010; WEISS, 1995).

1.2 OBJETIVO

Como objetivo este trabalho apresenta uma nova abordagem de classificação para o algoritmo PerC, intitulada sPerC (do inglês, ***s**ubconcept* PerC), particionando os dados do problema em *clusters*, fornecendo-os como entrada para o classificador PerC. A partir disto, são propostas 3 diferentes formas para agregar as perturbações provocadas pela inserção das instâncias de teste, em cada um dos *clusters* definidos.

1.2.1 Objetivos Específicos

- Empregar o algoritmo *K-Means* com o intuito de determinar os subconceitos (*clusters*) existentes nos conjuntos de dados. Como nenhum conhecimento a *priori* sobre a distribuição dos dados é levado em consideração, o valor do parâmetro K é estimado com base na combinação de 4 métricas de avaliação dos *clusters*, sendo elas: *Calinski-Harabasz* (CALIŃSKI; HARABASZ, 1974), *Davies-Bouldin* (DAVIES; BOULDIN, 1979), *Gap Statistic* (TIBSHIRANI; WALTHER; HASTIE, 2001) e *Silhouette* (ROUSSEEUW, 1987);
- Analisar a influência do tamanho dos *clusters* durante o processo de treinamento do classificador PerC;
- Definir, e avaliar, novas regras para combinar as perturbações provocadas pelas instâncias de teste;
- Avaliar o desempenho da abordagem proposta a partir de dados disponibilizados dos repositórios *Knowledge Extraction based on Evolutioary Learning* (KEEL) (ALCALÁ-FDEZ et al., 2011) e *UCI Machine Learning* (UCI) (DUA; GRAFF, 2017), comparando os resultados obtidos com os algoritmos do estado-da-arte Árvore de Decisão (BREIMAN, 2017), k-NN (WANG; NESKOVIC; COOPER, 2007), *Multilayer Perceptron* (MLP) (CHENG; TITTERINGTON, 1994), *Naïve Bayes* (DOMINGOS; PAZZANI, 1997), *Random Forest* (BREIMAN, 2001) e *Support Vector Machine* (SVM) (SCHOLKOPF; SMOLA, 2001).

1.3 VISÃO GERAL DA PROPOSTA

Este trabalho está organizado da seguinte forma. No Capítulo 2 a fudamentação teórica é apresentada, trazendo conceitos necessários para a compreensão deste trabalho. O Capítulo 3 descreve a arquitetura proposta, detalhando a aplicação do algoritmo *K-Means* para realizar o agrupamento dos dados. No Capítulo 4 os experimentos são definidos, especificando as bases de dados utilizadas. O Capítulo 5 apresenta os resultados obtidos, seguidos por uma discussão. Por fim, o Capítulo 6 conclui o estudo e fornece possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO

Esse capítulo descreve os principais conceitos abordados ao longo desta dissertação. A Seção 2.1 apresenta o conceito de aprendizagem de máquina, detalhando as principais abordagens e paradigmas da área. A Seção 2.2 fundamenta o classificador PerC. Na Seção 2.3 a técnica *K-means* é apresentada. Na Seção 2.4 são apresentadas as considerações finais.

2.1 APRENDIZAGEM DE MÁQUINA

Subcampo da Inteligência Artificial (IA), Aprendizagem de Máquina (do inglês, *Machine Learning*) têm como objetivo o desenvolvimento de técnicas computacionais, conhecidas como modelos, que aprendem e identificam padrões a partir dos dados, visando encontrar *insights* ocultos, com mínima intervenção humana. Segundo Tom Mitchell, em (MITCHELL et al., 1997), o conceito de *machine learning* pode ser entendido como o estudo de técnicas que aprendem a partir da experiência E, em relação a uma classe de tarefas T, com medida de desempenho P, se seu desempenho em T, medido por P, melhora em E.

Localizada na interseção da ciência da computação e estatística (JORDAN; MITCHELL, 2015), *machine learning* busca gerar hipóteses através dos dados, de forma automática, sendo empregada em uma variedade de tarefas computacionais onde criar e programar algoritmos é impraticável. Em geral, sistemas de aprendizagem de máquina visam aproximar uma função f, não definida, processando um conjunto de dados (chamado de conjunto de treinamento) que auxiliará a compreender o comportamento de eventos ocorridos, e generalizar para eventos futuros.

Tipicamente, para solucionar um problema utilizando artifícios de aprendizagem de máquina, faz-se necessário categorizá-lo. Conhecer a natureza de um problema possibilita pensar sobre quais dados são necessários e que tipos de técnicas podem ser exploradas. Dessa maneira, é possível distinguir os problemas de aprendizagem de máquina nos seguintes tipos:

- Aprendizagem supervisionado
- Aprendizagem não supervisionado
- Aprendizagem por reforço

A partir de um conjunto de dados rotulados, ou seja, a saída correta para cada padrão de entrada é conhecida, as técnicas de *aprendizado supervisionado* são treinadas definindo uma função $f : x \rightarrow y$, que mapeia cada padrão *x* em uma classe *y*. O objetivo é aprender uma regra geral que mapeia as entradas para as saídas, ajustando os parâmetros dos modelos para que possa prever rótulos desconhecidos para novos padrões de entrada (conjunto de teste, por exemplo). Sendo conhecido como as técnicas de aprendizado de máquina mais utilizadas (HASTIE; TIBSHIRANI; FRIEDMAN, 2009), os métodos de aprendizado supervisionado são categorizados em algoritmos para classificação e regressão (ALPAYDIN, 2014; KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007). De modo geral, problemas de classificação de padrões tem o objetivo prever categorias distintas, usando modelos de aprendizado de máquina para distinguir várias classes de uma só vez, combinar múltiplos preditores e obter probabilidades para cada associação realizada (BZDOK; KRZYWINSKI; ALTMAN, 2018). Para problemas de regressão, a tarefa consiste em estimar valores reais, visando determinar como uma variável se comporta na medida em que outra variável sofre oscilações.

No *aprendizado não supervisionado*, os rótulos de um conjunto de dados são desconhecidos. Dessa maneira, o objetivo de tais problema envolve a análise de dados não rotulados buscando encontrar relações estruturais nos dados (JORDAN; MITCHELL, 2015). Nesse cenário, todas as variáveis usadas na análise são fornecidas como entradas e, devido à abordagem, os métodos são adequados para técnicas de mineração em *cluster* e associação (BERRY; MOHAMED; YAP, 2019). De acordo com Gentleman e Carey (2008), a definição do algoritmo a ser utilizado, a escolha da métrica de similaridade empregada, assim como a seleção de amostras a serem agrupadas são pré-requisitos para a execução das técnicas de aprendizado não supervisionado. Devido a ausência de dados rotulados, avaliar um modelo de aprendizado não supervisionado torna-se mais complexo em relação às técnicas de aprendizado supervisionado.

No *aprendizado por reforço*, os modelos de aprendizagem tentam aprender qual é a melhor ação a ser tomada, por meio de interações de tentativa e erro (KAELBLING; LITTMAN; MOORE, 1996), dependendo das circunstâncias em que essa ação será executada. De acordo com Duda, Hart e Stork (2012), tal aprendizado é análogo a um crítico que apenas afirma que algo está certo ou errado, mas não diz especificamente como está errado. Como muitas vezes é impraticável obter o comportamento desejado de todas as situações, espera-se que um modelo deve ser capaz de aprender com sua própria experiência (SUTTON; BARTO, 2018).

Como citado anteriormente, o aprendizado de um algoritmo de AM é realizado a partir do fornecimento de um conjunto de dados ou com base em suas percepções e ações.

Nesse contexto, um dos objetivos centrais de um modelo é generalizar a partir de suas experiências (BISHOP, 2006). A generalização pode ser entendida como a capacidade de responder corretamente a exemplos não vistos. Muitos métodos de aprendizagem constroem uma descrição geral e explícita da função alvo a partir de exemplos de treinamento. Assim, os diversos sistemas de AM possuem características que possibilitam sua classificação considerando o paradigma no qual foram projetados, tais como o paradigma simbólico, baseado em instâncias, estatístico, conexionista, evolucionário, entre outros (MITCHELL et al., 1997).

O aprendizado simbólico tem por característica aprender construindo representações simbólicas de um conceito, em uma linguagem de fácil interpretação, tipicamente expressa na forma lógica, através da análise de exemplos e contraexemplos. As representações simbólicas, geralmente, assumem a forma de árvores de decisão (QUINLAN, 1986), regras de produção ou redes semânticas (SOWA, 1987).

Os métodos de aprendizado baseado em instâncias (do inglês, *Instance-based learning*) (AHA; KIBLER; ALBERT, 1991), rotulam novos dados a partir de exemplos similares. Tais algoritmos assumem que as instâncias podem ser representadas como pontos em um espaço euclidiano. Denominados de *lazy learning* (aprendizado preguiçoso), não possuem uma etapa de treinamento do modelo, consistindo somente no armazenamento dos dados de treinamento em memória, cuja a generalização é realizado na etapa de teste. O algoritmo *Nearest Neighbours* (WANG, 2006) é o mais comum em tal paradigma.

No aprendizado estatístico a tomada de decisão é formulada em termos probabilísticos, utilizando modelos estatísticos para encontrar uma boa aproximação do problema. Em geral, os métodos estatísticos são baseados na regra de Bayes (DUDA; HART; STORK, 2012), que utilizam um modelo probabilístico baseado no conhecimento prévio do problema, o qual é combinado com exemplos de treinamento para determinar a probabilidade final de uma hipótese.

O paradigma conexionista está diretamente ligado às Redes Neurais (BEALE; JACKSON, 1990), estruturas distribuídas formadas por um grande número de unidades de processamento conectadas entre si, inspiradas no modelo biológico do sistema nervoso.

O aprendizado evolucionário (ou aprendizado evolutivo) consiste em técnicas utilizadas para a resolução de problemas de modelagem ou otimização, empregando modelos computacionais baseados na teoria da evolução natural de Darwin, no qual somente os mais adaptados sobrevivem (MITCHELL et al., 1997). Entre os algoritmos baseado neste paradigma, pode-se destacar os algoritmos genéticos (WHITLEY, 1994) e algoritmos de Inteligência de Enxame (KENNEDY, 2006).

Nos últimos anos, o progresso recente em aprendizado de máquina foi impulsionado pelo desenvolvimento de novos algoritmos e teoria de aprendizado e pela explosão contínua na disponibilidade de dados *online* e computação de baixo custo (JORDAN; MITCHELL, 2015). Sua utilização pode ser encontrada em diversas aplicações, como reconhecimento de imagem (TAIGMAN et al., 2014) e fala (BERARD et al., 2016; CARLINI; WAGNER, 2018), detecção de fraudes (AWOYEMI; ADETUNMBI; OLUWADARE, 2017; RANDHAWA et al., 2018), diagnóstico médico (NILASHI et al., 2017; KUMAR; GANDHI, 2018), entre outras (SANTOS et al., 2019; FONSÊCA; MACIEL, 2019; NETO et al., 2019).

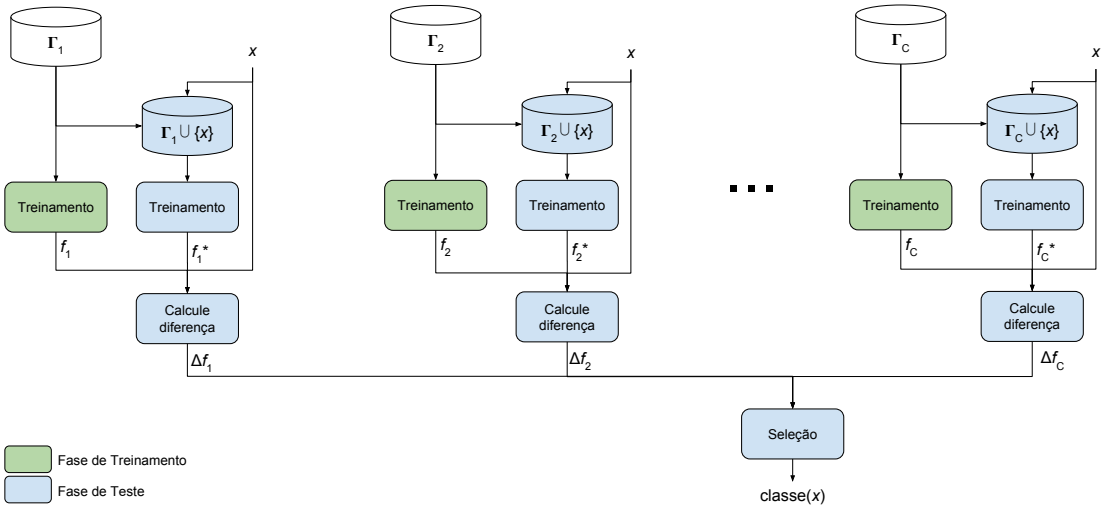
2.2 *PERTURBATION-BASED CLASSIFIER (PERC)*

Os métodos desenvolvidos a partir da inferência bayesiana ou derivados do classificador de Bayes, possuem forte dependência dos vetores médios, μ_i, e das matrizes de covariância, Σ_i, de cada uma das classes (grupos) existentes nos dados de treinamento (DUDA; HART; STORK, 2012; FUKUNAGA, 2013; JAIN; DUIN; MAO, 2000). Dessa forma, estimativas de máxima verossimilhança, baseadas exclusivamente nas amostras disponíveis para o treinamento, denotadas por $\hat{\mu}_i$ e $\hat{\Sigma}_i$ (JOHNSON; WICHERN et al., 2002), são as mais utilizadas (ARAÚJO; CAVALCANTI; REN, 2020).

De acordo com Edson et al., em (ARAÚJO; CAVALCANTI; REN, 2020), se um dado exemplo de teste pertence a uma determinada classe, a inserção dessa amostra fará com que seus parâmetros $\hat{\mu}_i$ e $\hat{\Sigma}_i$ sejam alterados, provocando uma perturbação mínima na distribuição de sua respectiva classe. Dessa forma, é possível avaliar as perturbações $\Delta\hat{\mu}_i$ e $\Delta\hat{\Sigma}_i$ para cada uma das classes presentes no banco de treinamento e distinguir qual classe sofrerá maior ou menor perturbação (ARAÚJO; CAVALCANTI; REN, 2020).

Dada as explanações acima, o trabalho intitulado *Um classificador baseado em perturbações* (PerC, do inglês, ***Per**turbation-based **C**lassifier*) (ARAÚJO; CAVALCANTI; REN, 2020) foi proposto. O PerC é um modelo de aprendizagem de máquina supervisionado, baseado no classificador de Bayes. Dada uma instância de teste *x*, o modelo PerC atribui ao exemplo à classe em que ocorrer a menor perturbação nas estimativas do vetor médio, $\hat{\mu}_i$, e matriz de covariância, $\hat{\Sigma}_i$, ou numa combinação de ambos, calculando as perturbações ocorridas em todas as classes, existentes nos dados de treinamento, simulando uma possível inserção da amostra a ser rotulada. A Figura 3, apresenta o diagrama de bloco do classificador PerC.

Figura 3 – Diagrama de Blocos - Algoritmo PerC.



Fonte: Adaptado de Araújo, Cavalcanti e Ren (2020).

De modo geral, o modelo PerC consiste de duas fases, treinamento e teste. Durante a fase de treinamento, para cada classe existente nos dados de treinamento ($\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_C\}$, sendo C o número de classes), um subconjunto Γ_i, constituído por amostras de uma mesma classe, é fornecido como entrada para a técnica descrita, de modo que para cada classe seja treinado um classificador f_i. De acordo com o classificador de Bayes, o treinamento consiste em calcular o vetor médio, $\hat{\mu}_i$, e a matriz de covariância, $\hat{\Sigma}_i$, utilizando, respectivamente, as equações 2.1 e 2.2 definidas a seguir:

$$\hat{\mu}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} x_{i,j}, \tag{2.1}$$

$$\hat{\Sigma}_i = \frac{1}{N_i - 1} \sum_{j=1}^{N_i} (x_{i,j} - \hat{\mu}_i)(x_{i,j} - \hat{\mu}_i)^T \tag{2.2}$$

com $x_{i,j}$ sendo o j-ésimo exemplo da classe w_i e N_i o número de amostras da classe w_i.

Na fase de teste, um novo classificador f_i^* é treinado a partir do novo banco de dados formado pela adição do padrão de teste x ao subconjunto de treinamento Γ_i. Em seguida, o método proposta irá calcular as perturbações Δf_i provocada pela inserção do padrão de teste x, para cada uma das classes. Por fim, a classe do padrão de teste x é determinada selecionando a classe que sofre menor perturbação.

Baseado no modelo proposto, três variações para computar a diferença entre f_i e f_i^* foram propostas, sendo elas:

- $f_i = f(\hat{\mu}_i)$ - Intitulado de *PerC(Mean)*, esse modelo descreve a distribuição das classes analisando exclusivamente as perturbações ocorridas nos vetores médios, $\hat{\mu}_i$, decorrentes da inserção da instância de teste x;
- $f_i = g(\hat{\Sigma}_i)$ - Chamado de *PerC(Cov)*, as distribuição das classes são descritas averiguando apenas as perturbações ocorridas nas matrizes de covariâncias, $\hat{\Sigma}_i$;
- $f_i = h(\hat{\mu}_i, \hat{\Sigma}_i)$ - Classificador mais poderoso, intitulado de *PerC(Comb)*, que descreve as distribuições das classes, analisando as perturbações provocadas em ambos os parâmetros, $\hat{\mu}_i$ e $\hat{\Sigma}_i$, decorrentes da inserção da amostra de teste x.

Um ponto de atenção provocado pelos autores é a necessidade de armazenar o conjunto de treinamento Γ (parâmetro de entrada do algoritmo), para o cálculo das perturbações ocorridas nos parâmetros $\Delta\hat{\mu}_i$ e $\Delta\hat{\Sigma}_i$. Baseado nisso, os autores demonstraram um modo eficiente de calcular as perturbações, sem a necessidade de armazenar o conjunto de treinamento Γ. Assim, as alterações sofridas pelos parâmetros $\Delta\hat{\mu}_i$ e $\Delta\hat{\Sigma}_i$, após a inserção da amostra de teste x, é dada, respectivamente, pelas equações 2.3 e 2.4, a seguir:

$$\Delta\hat{\mu}_i = \frac{1}{N_i + 1}(x - \hat{\mu}_i) \tag{2.3}$$

$$\Delta\hat{\Sigma}_i = -\frac{1}{N_i}\hat{\Sigma}_i + \frac{1}{N_i + 1}(x - \hat{\mu}_i)(x - \hat{\mu}_i)^T \tag{2.4}$$

A partir dos métodos definidos acima, os autores propuseram, também, um modo eficiente para computar as alterações resultante da combinação dos parâmetros $\hat{\mu}_i$ e $\hat{\Sigma}_i$, de acordo com a equação 2.5 a seguir:

$$\Delta d_i^{'} \cong [-2\hat{\Sigma}_i^{-1}(x - \hat{\mu}_i)]^T \ \Delta\hat{\mu}_i + tr\{[\hat{\Sigma}_i^{-1} - \hat{\Sigma}_i^{-1}(x - \hat{\mu}_i)(x - \hat{\mu}_i)^T\hat{\Sigma}_i^{-1}] \ \Delta\hat{\Sigma}_i\} \tag{2.5}$$

sendo $\Delta d_i^{'}$ a perturbação ocorrida na classe w_i, decorrente da inserção do vetor de teste x, $\hat{\mu}_i$ a estimativa do vetor médio, $\hat{\Sigma}_i^{-1}$ a matriz inversa da estimativa $\hat{\Sigma}_i$, e $\Delta\hat{\mu}_i$ e $\Delta\hat{\Sigma}_i$ as perturbações provocadas no vetor médio e matriz de covariância, respectivamente.

De acordo com o autor, a equação 2.5 depende fortemente de $\hat{\Sigma}_i^{-1}$ que, em alguns casos não existe. Nesse cenário, visando evitar possíveis problemas, utilizou-se a pseudo-inversa da matriz $\hat{\Sigma}_i$ que funciona como uma aproximação numérica de $\hat{\Sigma}_i^{-1}$ (ARAÚJO; CAVALCANTI; REN, 2020).

Por fim, outro ponto de destaque, ressaltado pelos autores, refere-se ao método PerC não ser categorizado como um modelo de aprendizagem *online* (ADE; DESHMUKH, 2013; KIVINEN;

SMOLA; WILLIAMSON, 2004; LÜTZ; RODNER; DENZLER, 2011; LÜTZ; RODNER; DENZLER, 2013) (do inglês, *online learning*). Durante a fase de teste do classificador, a amostra de teste x não é inserida permanentemente no conjunto de treinamento Γ, ou seja, é realizado uma simulação da inserção da amostra a ser rotulada, sendo descartada ao término do processo de classificação. Na há portanto, alterações dos valores de $\hat{\mu}_i$ e $\hat{\Sigma}_i$ em nenhuma das classes durante ou após o processo de classificação (ARAÚJO; CAVALCANTI; REN, 2020).

2.3 *K-MEANS*

A análise de agrupamento (do inglês, *Cluster Analysis*) (TRYON, 1939) é o estudo formal de métodos e algoritmos, usados para classificar objetos em grupos relativamente homogêneos chamados de agrupamento (ou conglomerados), de acordo com características ou similaridade intrínsecas medidas ou percebidas (JAIN, 2010). Assim, organizar os dados em agrupamentos sensíveis é um dos mais fundamentais modos de entendimento e aprendizado (JAIN, 2010).

A análise de agrupamento visa encontrar grupos "naturais" de objetos para um conjunto de dados não rotulados, sendo assim categorizada como um tipo de aprendizado não supervisionado. Os *clusters*, conjunto de pontos compactos e isolados, compartilham as características de cada objeto em uma medida de similaridade, podendo diferenciar-se um do outro em termos de formato, tamanho e densidade (JAIN, 2010). Nesse caso, a similaridade pode ser vista como a distância entre os objetos.

Proposto a mais de 50 anos, o algoritmo *K-Means* (MACQUEEN, 1967; LLOYD, 1982; FORGY, 1965; ARTHUR; VASSILVITSKII, 2007) é o método de agrupamento de dados mais conhecido na literatura, dada sua simplicidade na implementação, eficiência, e sucesso empírico (JAIN, 2010). O algoritmo *K-Means* busca particionar os dados em K grupos, pré-definidos, definindo representantes (centroides) para cada uma das partições, minimizando o erro quadrático (SSE, do inglês, *Sum of Squared Errors*) entre o ponto médio de um *cluster* e os pontos no *clusters*, segundo o critério dos mínimos quadrados.

Seja $\overline{x}_i$ o representante do *cluster* C_i. Podemos definir o erro quadrático entre $\overline{x}_i$ e os pontos x_i, presentes no cluster C_i, de acordo com a equação 2.6:

$$J(C_i) = \sum_{x_i \in C_i} ||x_i - \overline{x}_i||^2 \tag{2.6}$$

O objetivo do *K-Means* é minimizar a soma do erro quadrático médio sobre todos os *clusters*.

Assim, a função objetivo do algoritmo pode ser descrita conforme a equação 2.7:

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in C_i} ||x_i - \overline{x}_i||^2 \tag{2.7}$$

sendo $C = \{C_1, C_2, \ldots, C_k\}$ o conjunto de K *clusters*. O *K-Means* é um algoritmo de abordagem gulosa, convergindo em um número finito de iterações. Contudo, a minimização da função objetivo é um problema computacionalmente conhecido como NP-difícil (até para K=2) (DRINEAS et al., 2004).

De acordo com (JAIN; DUBES, 1988), as principais etapas do algoritmo *K-Means* podem serem descritas a seguir:

1. Selecione representantes (centroides) para os K *clusters* iniciais;
2. Gere uma nova partição atribuindo cada amostra ao centroide mais próximo;
3. Calcule os novos representantes dos *clusters*.

As etapas 2 e 3 são repetidas até que o critério de parada seja atinjido (por exemplo, não ocorra modificações nas partições geradas ou o número máximo de iterações seja alcançado). A Figura 4 demonstra a execução do algoritmo *K-Means*.

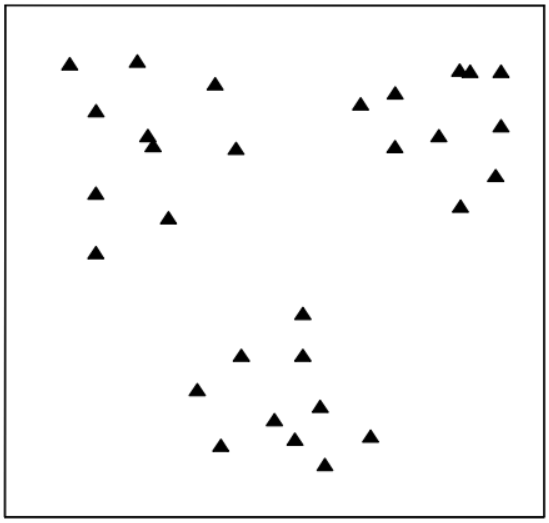
Dado um conjunto de dados, bidimensional, com 3 *clusters* (Figura 4a), o algoritmo *K-Means* seleciona 3 centroides, aleatoriamente, e define os agrupamentos iniciais, associando cada objeto ao seu representante mais próximo (Figura 4b). Em seguida, durante uma etapa iterativa, o algoritmo atualiza os centroides dos 3 *clusters* gerados, e, posteriormente, reagrupa os dados, de acordo com os novos representantes (Figuras 4c e 4d). Por fim, após o algoritmo convergir, é obtido como resultado as 3 partições geradas a partir do conjunto de dados inicial, assim como seus respectivos centroides, que podem ser utilizados para agrupar novos pontos (Figura 4e). Durante a execução do *K-Means*, vale destacar, todos os pontos são atribuídos exclusivamente a um e apenas um dos *clusters* gerados (BISHOP, 2006).

Entretanto, de acordo com Jain (2010), existem 3 parâmetros essenciais, que precisam serem definidos, para a execução do algoritmo *K-Means*, sendo eles: (i) Os representantes iniciais; (ii) a medida de similaridade; e (iii) o número K de *clusters*.

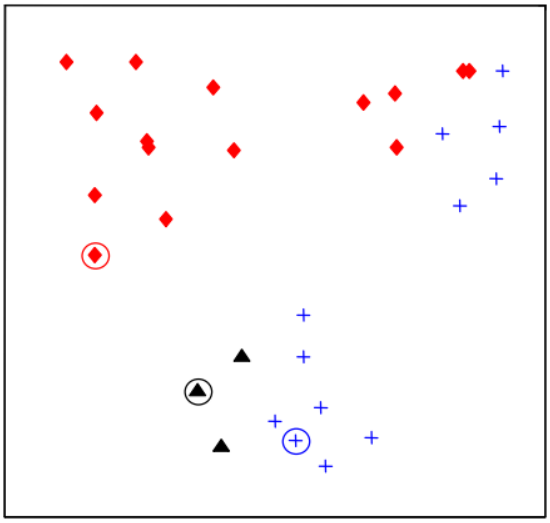
Como visto, ao iniciar sua execução, o *K-Means* seleciona aleatoriamente os *representantes* das K partições. Dessa forma, executar o algoritmo diversas vezes sobre os mesmos dados pode resultar em diferentes agrupamento. Portanto, uma má seleção dos representantes iniciais pode acarretar em um resultado ruim.

Figura 4 – Execução do algoritmo *K-Means*.

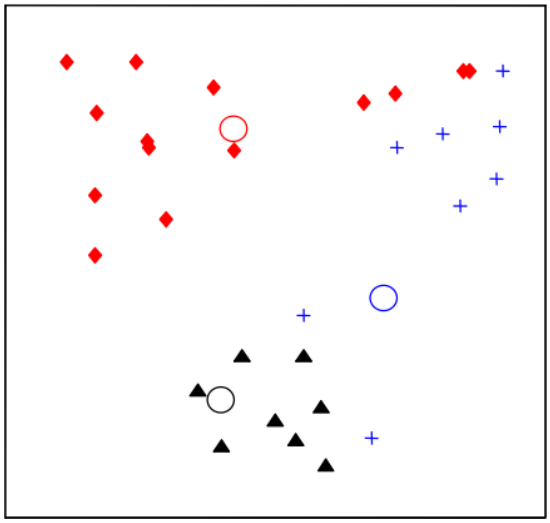
(a) Conjunto de dados inicial



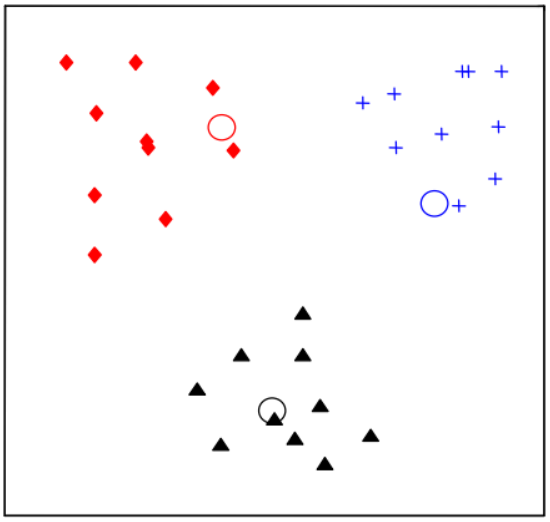
(b) Seleção dos centroides



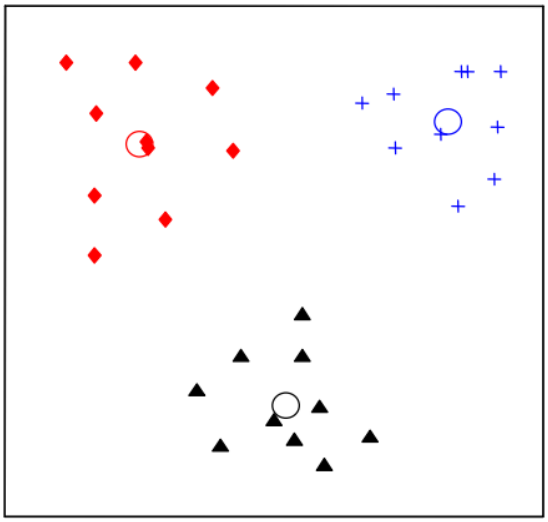
(c) Iteração 2



(d) Iteração 3



(e) Agrupamento final



Fonte: Adaptado de Jain (2010).

Tipicamente, para computar a *medida de similaridade* entre os centroides e os pontos de um *cluster*, emprega-se a distância euclidiana (BISHOP, 2006). Uma característica obtida pelo *K-means*, ao utilizar essa distância, é a busca por agrupamentos esféricos (JAIN, 2010). Outra medida utilizada é a distância de Mahalanobis. Nesse cenário, o *K-means* adquire a característica de detectar *clusters* elipsóides (MAO; JAIN, 1996).

Ao executar o método *K-means*, faz-se necessário definir o parâmetro K, ou seja, a *quantidade* de *clusters* que o algoritmo deve encontrar. Contudo, raramente o número correto de partições é conhecido (KODINARIYA; MAKWANA, 2013). Em conjuntos de dados com alta dimensionalidade, definir o valor de K torna-se mais difícil, mesmo quando os *clusters* estão bem separados (HAMERLY; ELKAN, 2004). Nesse cenário, estimar o valor ideal de *clusters* K torna-se uma tarefa crítica (JAIN, 2010).

A qualidade dos resultados alcançados pelo algoritmo *K-means* está diretamente correlacionada a escolha do número de *clusters* (PATIL; BAIDARI, 2019). Uma forma para determinar o número correto de partições é a partir da utilização do processo conhecido como *cluster validation*.

2.3.1 *Cluster validation*

Cluster validation (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001) é o processo comumente utilizado para avaliar a qualidade dos *clusters* gerados, através de um algoritmo de agrupamento, determinando o melhor valor do parâmetro K combinando *compactness* e *separation*. O critério de *compactness* respalda a importância dos elementos em uma mesma partição estarem o mais próximos uns dos outros. Já o critério *separation* determina a distância entre os elementos de dois *clusters* distintos. Essa medida tem sido amplamente utilizada devido à sua eficiência e efetividade computacional para partições esféricas (RENDÓN et al., 2011).

Avaliar os resultados das técnicas de agrupamentos é um dos problemas mais vitais na análise de *clusters* (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001). Assim, os *clustering validation* ganharam força, sendo substanciais para o sucesso dos algoritmos de agrupamento (LIU et al., 2010). Por via de regra, os métodos de *clustering validation* são divididos em duas famílias, a saber:

- *External validation*

- *Internal validation*

Os *external validation* utilizam o conhecimento prévio do conjunto de dados como artefato para averiguar a qualidade das partições produzidas, ou seja, os *clusters* resultantes dos algoritmos de agrupamento são comparados com uma partição referência (definida a *priori*) da base de dados. Essa abordagem pode ser vista como um tipo de medição de erro (BRUN et al., 2007). Entretanto, em problemas do mundo real, torna-se difícil ser aplicada, dado a inexistência de conhecimento prévio sobre os dados (RENDÓN et al., 2011). *Adjusted Rand* (HUBERT; ARABIE, 1985) e *Jaccard* (JACCARD, 1908) são exemplos de técnicas que constituem a família *external validation*.

Amplamente utilizada para definir o número ideal de *clusters* (LIU et al., 2010), os *internal validation* não requisitam conhecimento prévio do problema para analisar as partições produzidas, realizando a tomada de decisão apenas com informações contidas nos dados (HÄMÄLÄINEN; JAUHIAINEN; KÄRKKÄINEN, 2017). Essa família de técnicas baseia-se no pressuposto de que os algoritmos devem procurar *clusters* cujos membros estejam próximos um do outro e distantes dos membros de outros *clusters* (BRUN et al., 2007). São exemplos dessa família, técnicas como *Silhouette* (ROUSSEEUW, 1987), *Dunn index* (DUNN, 1973), *Score function* (SAITTA; RAPHAEL; SMITH, 2007), entre outros.

Ainda é possível encontrar técnicas baseada em uma terceira família de *clustering validation*, intitulada *relative validation*. Essa abordagem realiza a comparação de partições geradas pelo mesmo algoritmo de agrupamento (BRUN et al., 2007). Enquanto as demais famílias possuem como base o teste estatístico, essa categoria é baseada em critérios relativos, diminuindo assim o custo computacional durante a tomada de decisão (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001). *Modified Hubert index* (HUBERT; ARABIE, 1985) e o método de *Elbow* (NG, 2012) são exemplos de técnicas pertencentes a família *relative validation*.

2.4 CONSIDERAÇÕES FINAIS

Neste capítulo, foram apresentados uma visão geral dos fundamentos teóricos utilizados como base para a pesquisa. A Seção 2.1 introduziu os diversos conceitos da área de Aprendizagem de Máquina, detalhando as abordagens existentes de acordo com a naturaza do problema e categorizando os sistemas de aprendizagem de máquina baseados no paradigma no qual foram projetados. A Seção 2.2 descreveu as principais definições da técnica PerC,

detalhando seu funcionamento e apresentando suas diferentes variações. Na Seção 2.3, o algoritmo K-Means foi brevemente descrito, sendo detalhado sua execução, assim como os parâmetro necessário para seu funcionamento. Por fim, abordagens para avaliação de *clusters* foram discutidas na Seção 2.3.1, detalhando as principais abordagens existentes.

3 ARQUITETURA PROPOSTA

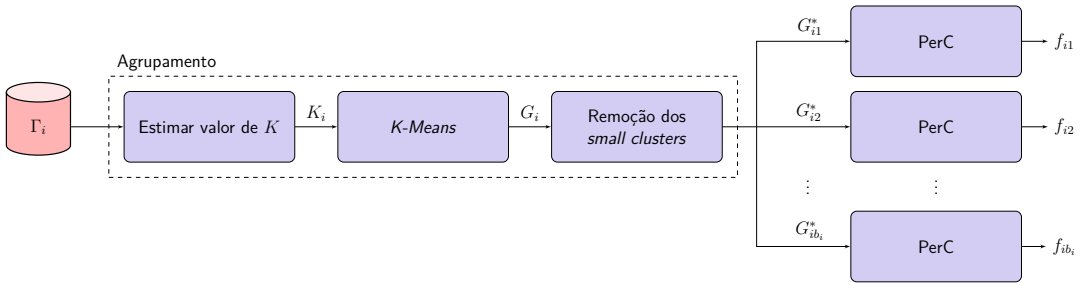
Nesse capítulo, o método proposto *subconcept PerC* (sPerC) é descrito, apresentando suas principais definições. A técnica proposta possui duas fases, treinamento e teste. Durante a etapa de treinamento, um pré-processamento dos dados é realizado aplicando o algoritmo *K-Means*. Na fase de teste, as perturbações estimadas pelo classificador PerC são combinadas para predizer o rótulo da amostra de teste. As Seções 3.1 e 3.2, a seguir, detalham as fases de treinamento e teste do método proposto. Na Seção 3.3 é ilustrada a execução da arquitetura proposta usando um *toy problem*.

Seja $\Gamma = \{(x_j, y_j)\}_{j=1}^{n}$ um conjunto de treinamento, sendo $y_j \in \{w_1, \ldots, w_m\}$ a classe correspondente da amostra x_j e m o número de classes do problema. Define-se formalmente $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \ldots \cup \Gamma_m$, sendo $\Gamma_i = \{(x_i, y_i) \mid (x_i, y_i) \in \Gamma \text{ e } y_i = w_i\}$ o subconjunto de treinamento formado por instâncias pertencentes à classe w_i.

3.1 TREINAMENTO DO CLASSIFICADOR USANDO OS DADOS PARTICIONADOS

A arquitetura proposta busca particionar o espaço de características, para cada subconjunto de treinamento Γ_i, em *clusters*, de modo que amostras de classes distintas não sejam agrupadas em uma mesma partição. Em seguida, um classificador PerC é treinado para cada grupo. A Figura 5 exibe com mais detalhes o processo de agrupamento dos dados e o treinamento do classificador PerC.

Figura 5 – Treinamento do classificador PerC usando os dados particionados.



Inicialmente cada subconjunto Γ_i, separadamente, é fornecido como entrada para o algoritmo de agrupamento. O método *K-Means* é empregado para realizar o particionamento dos dados, dado sua ampla utilização na literatura e por apresentar sólidos resultados em diversos cenários de uso. Ao empregar a técnica *K-Means*, faz-se necessário definir, *a priori*, o número de partições K que o método deve gerar. Escolher o valor ideal para K é

um problema complexo (ÜNLÜ; XANTHOPOULOS, 2019), uma vez que tal valor depende do conjunto de dados utilizado. Nessa dissertação, para estimar o valor do parâmetro K, foram adotadas medidas baseadas em *internal validation*, uma vez que nenhum conhecimento prévio é levado em consideração para estimar o valor de K. Sendo assim, o valor de K pode ser estimado executando o algoritmo *K-Means* para diferentes valores de K, computando o *internal validation* índice para cada partição resultante, e selecionando o valor de K que melhor satisfaz o critério de seleção da medida empregada.

Uma vez estimado o número de *clusters* K_i para o subconjunto Γ_i, o algoritmo *K-Means* é executado 100 vezes sobre o mesmo subconjunto, de maneira independente. Ao final de cada execução, um conjunto de partições $G_i = \{G_{i1}, G_{i2}, \ldots, G_{iK_i}\}$ é obtido. Tais repetições foram estipuladas para evitar que a escolha dos centroides iniciais interfiram negativamente na geração das partições, dado que tais centroides são selecionados aleatoriamente. Após o término de todas as execuções do algoritmo *K-Means*, o conjunto de partições G_i que minimiza a soma de todas as distâncias entre cada amostra e seu representante é escolhido.

As partições geradas pelo algoritmo *K-Means* possuem cardinalidades diferentes, de modo que alguns grupos podem ser formados por um número reduzido de amostras (*small clusters*), sendo afetados pela problemática conhecida como *small disjuncts problem* (WEISS, 2010; WEISS, 1995). Tais *disjuncts*, em geral, apresentam elevadas taxas de erro sobre o conjunto de teste. Esse comportamento pode influênciar diretamente o desempenho do classificador PerC, além da baixa representatividade dos dados intrínsecas nos *small clusters*, uma vez que o modelo estima o vetor médio, $\hat{\mu}$, e a matriz de covariância, $\hat{\Sigma}$, a partir de um conjunto de dados informado. Neste quesito, visando ampliar o desempenho do classificador PerC, o próximo passo do método proposto consiste em realizar a remoção dos *small clusters* do processo de treinamento do modelo. Contudo, antes da remoção, faz-se necessário quantificar tal categorização a partir de uma regra.

Nesse cenário, no trabalho proposto em (HE; XU; DENG, 2003), os autores classificam os *clusters* gerados em duas categorias, grandes e pequenos *clusters*, baseado na definição a seguir: Seja $C_i = \{C_{i1}, C_{i2}, \ldots, C_{iK_i}\}$ um conjunto de *clusters* ordenado, tal que $|C_{i1}| \geq |C_{i2}| \geq \cdots \geq |C_{iK_i}|$. Dessa maneira define-se como b_i o índice limite entre os *large* e *small clusters*, de acordo com a Equação 3.1:

$$b_i = \arg\min_{K} \left\{ \sum_{t=1}^{K} |C_{it}| \geq |\Gamma_i| \times \tau \right\} \tag{3.1}$$

sendo τ um parâmetro definido *a priori*. Assim, cada *cluster* $C_{iK_i} \in C_i$ é categorizado como

large cluster ou *small cluster* de acordo com a Equação 3.2, a seguir:

$$categoria(C_{ij}) = \begin{cases} \textit{large cluster}, & j \leq b_i \\ \textit{small cluster}, & j > b_i \end{cases} \tag{3.2}$$

sendo j o índice do *cluster* e $j \leq K_i$. Dessa forma, as partições cujos índices sejam menores (maiores) que b_i são categorizadas como *large clusters* (*small clusters*). Uma vez realizada a categorização de cada partição $G_{iK_i} \in G_i$ (usando a Equação 3.2), os *clusters* considerados *small clusters* são removidos da etapa de treinamento. Assim, um novo conjunto de partições G_i^* é formado, contendo apenas os *large clusters*.

De posse do conjunto de partições G_i^*, o passo final do método proposto é treinar o classificador PerC. Dessa maneira, para cada partição $G_{ib_i}^* \in G_i^*$, treina-se um modelo f_{ib_i}, computando as estimativas do vetor médio $\hat{\mu}_i$ e a matriz de covariância $\hat{\Sigma}_i$, formando o conjunto de classificadores $f_i = \{f_{i1}, f_{i2}, \ldots, f_{ib_i}\}$. Como saída da arquitetura proposta são obtidos o conjunto de partições G_i^* e o conjunto de classificadores treinados f_i, para cada subconjunto Γ_i.

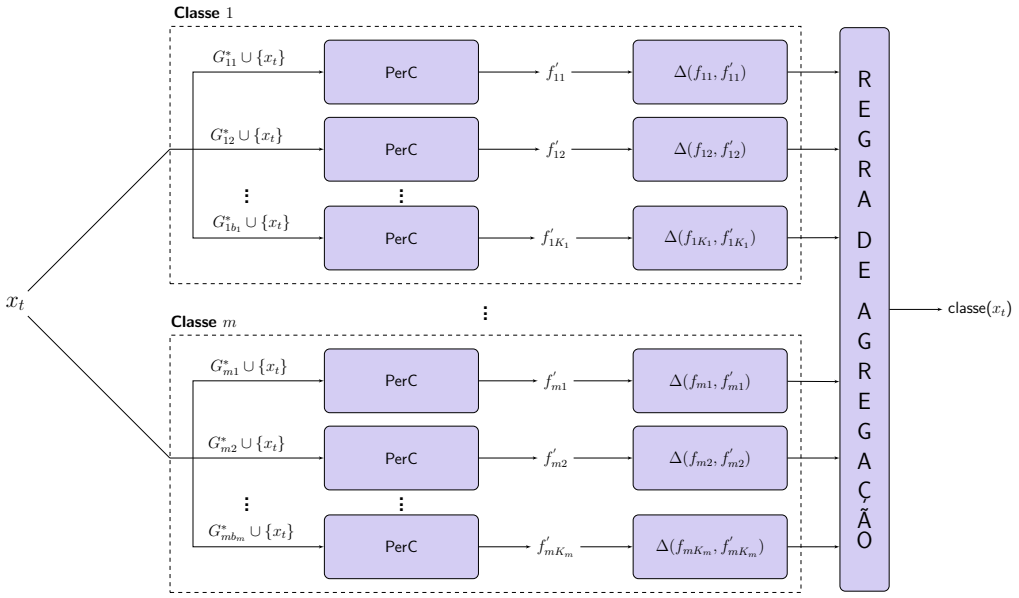
3.2 PREDIÇÃO DOS EXEMPLOS DE TESTES

Ao término da fase de treinamento (Seção 3.1), para cada classe w_i do *dataset* é obtido um conjunto de partições $G_i^* = \{G_{i1}^*, G_{i2}^*, \ldots, G_{ib_i}^*\}$ e um conjunto de classificadores $f_i = \{f_{i1}, f_{i2}, \ldots, f_{ib_i}\}$, sendo $i = \{1, \ldots, m\}$, m o número de classes e b_i a quantidade de partições geradas para a classe w_i, após a remoção dos *small clusters*. Dado um exemplo de teste x_t, a arquitetura proposta tem como propósito predizer sua classe, utilizando das perturbações que tal amostra provoca ao ser inserida nos *clusters* gerados. A Figura 6 demonstra a abordagem utilizada para predizer as instâncias de teste.

O primeiro passo do método proposto é inserir o exemplo de teste x_t em cada uma das partições geradas $G_{ib_i}^* \in G_i^*$, obtendo como resultado uma nova partição $G_{ib}^{*'} = G_{ib_i}^* \cup \{x_t\}$. Após a inserção da amostra de teste nos grupos gerados, a técnica PerC é aplicada sobre as novas partições. Assim, um novo classificador $f_{ib_i}^{'}$ é treinado para cada *cluster* $G_{ib_i}^{*'}$. De posse dos classificadores $f_{ib_i}^{'}$ treinados sobre cada partição $G_{ib_i}^{*'}$, o próximo passo da arquitetura proposta é computar as perturbações provocadas pela inserção da instância de teste x_t. Para tal, é calculada a perturbação, $\Delta(f_{ib_i}, f_{ib_i}^{'})$, entre o classificador f_{ib_i}, gerado na etapa de treinamento, e o classificador $f_{ib_i}^{'}$ gerado na etapa de teste. O procedimento mencionado é

executado para cada classe do problema, de modo que todo o conjunto de partições G_i^* e o conjunto de classificadores f_i, gerados na fase de treinamento, sejam empregados.

Figura 6 – Predição da instância de teste, combinando as perturbações provocadas após sua inserção nos *clusters* gerados.



Ao término do cômputo de todas as perturbações, as medidas são combinadas aplicando uma regra de agregação (definida na Seção 3.2.1). Como saída, a classe $y_t \in \{w_1, w_2, \ldots, w_m\}$ com menor perturbação é utilizada para rotular a instância de teste x_t.

3.2.1 Regras de agregação

Para rotular uma dada amostra de teste x_t, o método proposto calcula suas respectivas perturbações provocadas ao inserir tal instância sobre os *clusters* definidos. Dado tal fato, as perturbações ocorridas em ambos os parâmetros $\hat{\mu}_i$ e $\hat{\Sigma}_i$ podem ser agregadas aplicando diferentes regras de combinação. Nesse estudo foram selecionadas 4 regras para agregação das perturbações produzidas, a saber:

- **Regra do Mínimo (Mínimo)** - Dada uma instância de teste x_t, sua respectiva classe é definida com base na partição $G_{iK_i}^*$ que sofre menor perturbação, provocada após a inserção da amostra x_t.
- **Regra da Média (Média)** - Para cada classe w_i, a média das perturbações associadas ao seu respectivo conjunto de partições G_i^* é calculada. Assim, a classe da amostra de teste x_t é definida com base na classe que apresenta menor perturbação média.

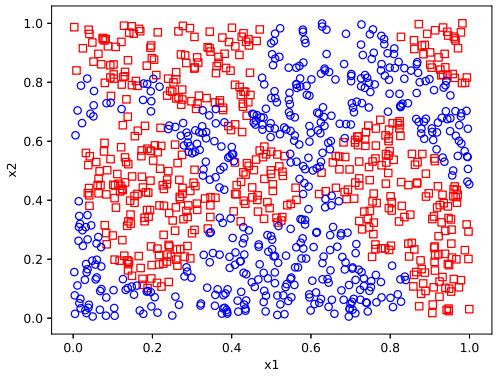
- **Regra da Mediana (Mediana)** - Para cada classe, a mediana das perturbações associadas a cada partição $G^*_{iK_i} \in G^*_i$ é computada. Em seguida, a classe da instância de teste x_t é estimada de acordo com a classe w_i que apresenta menor mediana.

- **Regra do *Cluster* Mais Próximo (*Cluster*)** - Para cada classe, a partição $G^*_{iK_i}$ que apresenta menor distância euclidiana entre seu centroide e a amostra de teste x_t é selecionada, e a perturbação associada a partição $G^*_{iK_i}$ computada. A predição da instância x_t é dada pela classe w_i que possuir menor perturbação.

3.3 *TOY PROBLEM*

Nessa seção, é apresentado o processo da técnica sPerC sobre um *toy problem*. Para o conjunto de treinamento Γ foi utilizado o *dataset* sintético *P2 Problem* (VALENTINI, 2005). Tal conjunto define um problema bidimensional, multimodal, composto por duas classes $\{w_1, w_2\}$, onde cada classe é delimitada por funções polinomiais e trigonométricas. Para o experimento, foram geradas 1.000 instâncias, distribuídas igualmente entre as classes. A Figura 7 exibe a distribuição dos dados no espaço de característica.

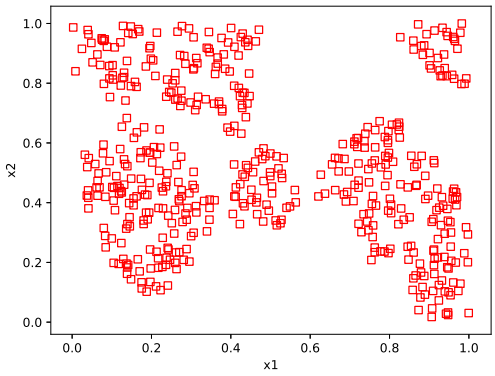
Figura 7 – Distribuição do conjunto de dados *P2 Problem*.



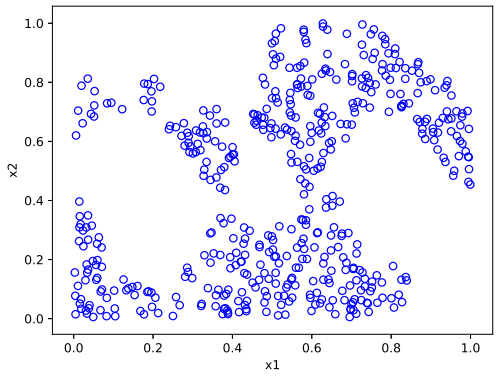
O processo de treinamento do sPerC é executado sobre cada subconjunto Γ_i, formado por instâncias pertencentes a classe w_i. Sendo assim, o primeiro passo da arquitetura proposta é dividir as instâncias do conjunto de treinamento Γ de acordo com as classes w_1 e w_2. A Figura 8 apresenta a separação dos dados para os subconjuntos Γ_1 (Figura 8a) e Γ_2 (Figura 8b), sendo $\Gamma = \Gamma_1 \cup \Gamma_2$ e $\Gamma_1 \cap \Gamma_2 = \emptyset$, são obtidos.

Figura 8 – Distribuição dos dados para os subconjunto Γ_1 (a) e Γ_2 (b), que constituem a base de dados *P2 Problem*.

(a) Subconjunto de dados Γ_1



(b) Subconjunto de dados Γ_2

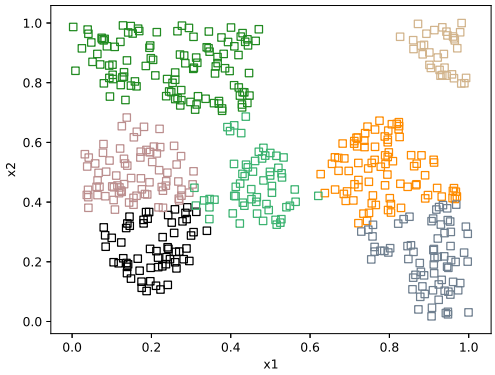


Uma vez definidos os subconjuntos Γ_1 e Γ_2, o próximo passo da arquitetura proposta é estimar o número K_i de *clusters* que a técnica K-Means deve gerar, para cada uma das classes.

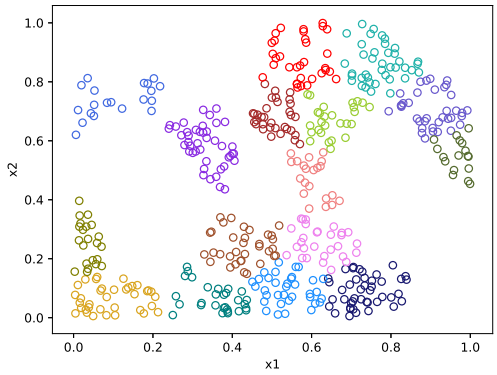
Analisando as distribuições dos dados para os subconjuntos Γ_1 (Figura 8a) e Γ_2 (Figura 8b), é possível visualizar, previamente, agrupamentos intrínsecos para cada uma das classes w_1 e w_2. Contudo, as técnicas baseadas em *internal validation* não levam em consideração conhecimento prévio para estimar o valor do parâmetro K.

Figura 9 – *Clusters* gerados pelo algoritmo *K-Means*, para os subconjuntos Γ_1 (a) e Γ_2 (b).

(a) Resultado do algoritmo *K-Means* ($K = 7$) para o subconjunto Γ_1



(b) Resultado do algoritmo *K-Means* ($K = 14$) para o subconjunto Γ_2



Como alternativa para estimar o número K_i de partições, o *internal validation* índice *Davies-Bouldin* foi aplicado, variando o valor de K_i no intervalo $[2, \sqrt{n_i/2}]$, sendo $n_i = |\Gamma_i|$ a cardinalidade do subconjunto Γ_i, e $i = \{1, 2\}$. Ao término da execução, a técnica sPerC obtém como resultado os valores $K_1 = 7$ e $K_2 = 14$, para os subconjuntos Γ_1 e Γ_2

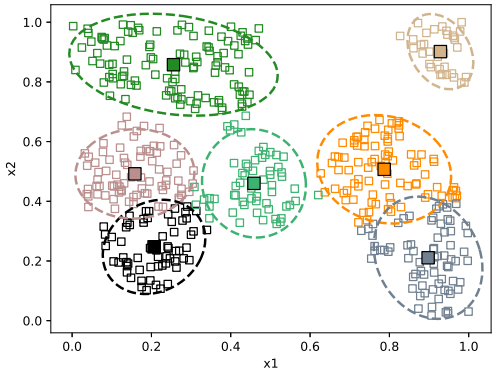
respectivamente. Uma vez estimado os parâmetros K_i, para as classes w_1 e w_2, o algoritmo *K-Means* é executado, 100 vezes, sobre os dados. Ao término da execução da técnica de agrupamento, os conjuntos de *clusters* G_1 e G_2 são obtidos. A Figura 9 apresenta os resultados da execução da técnica *K-Means* sobre os subconjuntos Γ_1 (Figura 9a) e Γ_2 (Figura 9b).

Durante a execução do *toy problem*, foi definido $\tau = 100\%$. Dessa forma, todos os *clusters* gerados são categorizados como *large clusters*. Consequentemente, nenhum *cluster* é descartado pela arquitetura proposta.

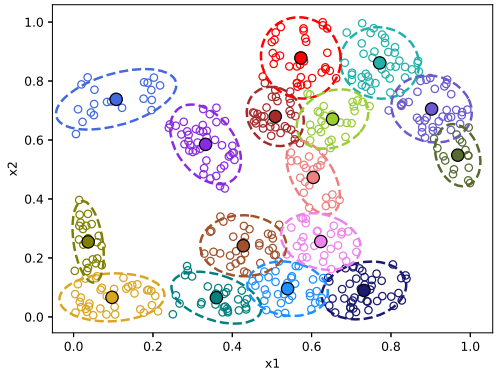
Para finalizar a fase de treinamento, os conjuntos de partições G_1 e G_2, sucessivamente, são fornecidos como entrada do classificador PerC. Desse modo, para cada *cluster* $G_{ij} \in G_i$, sendo $i = \{1, 2\}$ e $j \leq K_i$, um classificador f_{ij} é gerado. Como resultado da arquitetura proposta são obtidos os conjuntos de *clusters* G_1 e G_2, assim como os conjuntos de classificadores treinados f_1 e f_2, respectivamente para as classes w_1 e w_2. A Figura 10 apresenta a saída obtida para os subconjuntos Γ_1 (Figura 10a) e Γ_2 (Figura 10b), respectivamente.

Figura 10 – Saída do método proposto, para os subconjuntos Γ_1 (a) e Γ_2 (b).

(a) sPerC aplicado ao subconjunto Γ_1



(b) sPerC aplicado ao subconjunto Γ_2



Por fim, um experimento foi realizado para comparar o desempenho da técnica sPerC em relação a versão original do algoritmo PerC. Ambas as técnicas foram executadas através de uma validação cruzada estratificada 10-*folds*, com 10 repetições. Média da acurácia foi computada. Ao término do experimento, a técnica sPerC alcançou uma acurácia de 86,86%, enquanto o algoritmo PerC (versão original) obteve um desempenho de 67,25%. Como pode-se observar, a técnica sPerC alcança um alto grau de generalização para as classes existentes (vide as Figuras 10a e 10b), uma vez que a arquitetura proposta treina um algoritmo PerC para cada subconceito (*cluster*) definido, diferentemente da versão original do modelo PerC que enfoca apenas na classe como um todo, justificando o desempenho superior da técnica sPerC.

4 METODOLOGIA DOS EXPERIMENTOS

Nesse capítulo, o objetivo é descrever os experimentos conduzidos para avaliar o método sPerC em diferentes cenários. A implementação da arquitetura proposta e execução dos experimentos, foi realizada utilizando a linguagem de programação *Python*, em conjunto com a biblioteca *scikit-learn* (PEDREGOSA et al., 2011), que oferece suporte a diversas implementações dos modelos de aprendizagem de máquina.

Todos os experimentos foram executados através de uma validação cruzada estratificada 10-*folds*, com 10 repetições, para cada uma das bases de dados definidas (Seção 4.1). Nessa abordagem, os modelos são treinados com 9-*folds* (conjunto de treinamento) e avaliados com o *fold* remanescente (conjunto de teste).

A métrica AUC (do inglês, *Area Under Curve*) (BRADLEY, 1997), amplamente utilizada para avaliar o desempenho de modelos a partir de dados desbalanceados (LÓPEZ et al., 2013), foi escolhida como métrica de avaliação dos experimentos. A medida AUC é definida para problemas de classificação binária. Para problemas multiclasses, o valor AUC do modelo foi determinado como a média dos valores entre as classes existentes.

Com o intuito de averiguar possíveis diferenças significativas entre os valores de AUC computados, foram utilizados os testes de hipóteses não paramétricos de Friedman (FRIEDMAN, 1937), empregado para comparações entre múltiplas técnicas, e Wilcoxon (WILCOXON, 1992), para comparações entre duas técnicas.

4.1 BASES DE DADOS

Os experimentos foram conduzidos usando 30 bases de dados disponíveis nos repositórios *Knowledge Extraction based on Evolutionary Learning* (KEEL) (ALCALÁ-FDEZ et al., 2011) e UCI *Machine Learning* (UCI) (DUA; GRAFF, 2017). Os conjuntos de dados utilizados, com exceção da base de dados *vowel*, apresentam como característica o desbalanceamento dos dados. A Tabela 1 exibe as propriedades dos conjuntos utilizados: número de atributos (Atributos), número de amostras (Amostras), quantidade de classes (Classes) e a taxa de desbalanceamento (IR, do inglês, *imbalance ratio*).

A métrica IR é adotada em diversos trabalhos (OLIVEIRA; CAVALCANTI; SABOURIN, 2017; LÁZARO; HERRERA; FIGUEIRAS-VIDAL, 2020) para descrever o grau de desbalanceamento da

Tabela 1 – Descrição das 30 bases de dados utilizadas nos experimentos. Os *datasets* destacados (*) são oriundos do repositório UCI.

Dataset	Atributos	Amostras	Classes	IR	Dataset	Atributos	Amostras	Classes	IR
balance	4	625	3	5,88	newthyroid2	5	215	2	5,14
banana	2	5300	2	1,23	page-blocks0	10	5472	2	8,79
banknote*	4	1372	2	1,25	poker-8-9vs5	10	2075	2	82,00
bupa	6	345	2	1,38	poker-8-9vs6	10	1485	2	58,40
cleveland	13	297	5	12,31	shuttle-6vs2-3	9	230	2	22,00
cleveland-0vs4	13	173	2	12,31	transfusion*	4	748	2	3,20
ecoli-0-1-4-7vs2-3-5-6	7	336	2	10,59	vertebral2c*	6	310	2	2,10
ecoli1	7	336	2	3,36	vowel	13	990	11	1,00
ecoli2	7	336	2	5,46	winequality-red-4	11	1599	2	29,17
ecoli3	7	336	2	8,60	winequality-red-8vs6-7	11	855	2	46,50
glass-0-1-5vs2	9	172	2	9,12	winequality-white-3-9vs5	11	1482	2	58,28
hayes-roth	4	160	3	2,10	wisconsin	9	683	2	1,86
indian-liver-patient*	10	579	2	2,51	yeast-0-5-6-7-9vs4	8	528	2	9,35
led7digit-0-2-4-5-6-7-8-9vs1	7	443	2	10,97	yeast-1-4-5-8vs7	8	693	2	22,10
new-thyroid1	5	215	2	5,14	yeast-2vs4	8	514	2	9,08

distribuição dos dados, sendo resultante da proporção entre o número de instâncias da classe majoritária em relação ao número de amostras da classe minoritária. Para cada base de dados, seus respectivos atributos foram normalizados para um valor compreendido no intervalo $[0, 1]$.

4.2 *CLUSTER VALIDATION ÍNDICES*

Estimar o valor de K (número de *clusters*), para o algoritmo *K-Means*, que melhor particiona os dados para uma base de dados é um dos pontos crucias do sPerC, influenciando diretamente o treinamento do classificador PerC e, consequentemente, a predição das instâncias de testes. Vale ressaltar, que para as bases de dados definidas nos experimentos, não sabe-se qual a partição ideal para os dados, assim como o valor ideal do parâmetro K.

Nesse cenário, *internal validation* índices foram empregadas com o objetivo de avaliar as partições geradas, para diferentes valores de K, e selecionar o número de *clusters* que melhor divide o espaço de características. Entre as diversas métricas existentes na literatura (ARBELAITZ et al., 2013), 4 foram selecionadas e avaliadas nesse trabalho, definidas a seguir:

- ***Calinski–Harabasz* (CH)** (CALIŃSKI; HARABASZ, 1974) - Conhecida também como *variante ratio criterion*, analisa as partições geradas de acordo com a razão entre a soma das distâncias entre os objetos de um *cluster* e seu respectivo centroide, e a soma das distâncias entre cada centroide e o ponto médio da base de dados. Assim, o valor de K que maximiza o índice *Calinski-Harabasz* é selecionado.

- ***Davies–Bouldin* (DB)** (DAVIES; BOULDIN, 1979) - As partições geradas são avaliadas

determinando os pares de *clusters* mais similares entre si, identificando os grupos mais distantes, calculando a distância entre os centroides, e compactas, baseado na distância entre os pontos de um *cluster* ao seu respectivo centroide. Assim sendo, estima-se o valor do parâmetro K que minimiza a métrica DB.

- ***Gap Statistic* (Gap)** (TIBSHIRANI; WALTHER; HASTIE, 2001) - Avalia o resultado de um algoritmo de agrupamento, comparando o valor da dispersão de dados, geradas sobre o conjunto resultante, e seu valor esperado sobre uma distribuição uniforme aleatória, escolhendo o menor valor de K que satisfaz:

$$Gap(K) \geq Gap(K+1) - S_{K+1}, \tag{4.1}$$

  onde $Gap(K)$ é conhecida como a *gap statistic*, e S_K, seu respectivo desvio padrão.

- ***Silhouette* (Sil)** (ROUSSEEUW, 1987) - Estima o valor de K baseado nos princípios de *compactness*, determinada a partir da distância entre todos os objetos de um mesmo *cluster*, e *separation*, calculado a partir da distância entre um determinado objeto e o seu *cluster* vizinho mais próximo. Dessa forma, o número de *clusters* é estimado selecionando o valor de K que maximiza a métrica *silhouette*.

Os métodos acima mencionados foram selecionados por serem frequentemente empregados para estimar o valor de K, para o algoritmo *K-Means*, e apresentarem desempenhos satisfatórios em vários cenários (ARBELAITZ et al., 2013).

Cada *clustering validation* índice define sua estratégia para determinar o número de *clusters* ideal. Assim, estimar o parâmetro K, para o algoritmo *K-Means*, torna-se subjetivo, dependendo diretamente do índice aplicado. Baseado nessa explanação, foi definida uma abordagem, chamada de **Combinação**, no qual o melhor índice é selecionado de acordo com o desempenho do classificador PerC sobre o conjunto de treinamento.

4.3 DEFINIÇÃO DO PARÂMETRO τ

Para categorizar uma dada partição como *small* (*large*) *cluster*, faz-se necessário definir o valor do parâmetro τ e, à medida que seu valor é incrementado, o sPerC tende a considerar uma quantidade maior de *clusters* durante a fase de treinamento. Dessa maneira, diferentes valores para o parâmetro $\tau = 10\%, 20\%, 30\%, \ldots, 100\%$ foram considerados.

Vale destacar que a geração dos *clusters* é um processo não determinístico, dependendo da escolha inicial dos centroides. Sendo assim, foi definido um procedimento, chamado de **Auto**, para determinar o valor do parâmetro τ que maximiza o desempenho do classificador PerC, sobre o conjunto de treinamento.

4.4 ALGORITMOS DO ESTADO DA ARTE

Para comparar o desempenho do sPerC, foram utilizados 6 diferentes classificadores: Árvore de Decisão (BREIMAN, 2017), k-NN (WANG; NESKOVIC; COOPER, 2007), *Multilayer Perceptron* (CHENG; TITTERINGTON, 1994) , *Naïve Bayes* (DOMINGOS; PAZZANI, 1997), *Random Forest* (BREIMAN, 2001) e *Support Vector Machine* (SCHOLKOPF; SMOLA, 2001). Tais métodos foram selecionados por serem conhecidos na literatura e empregados em diversos trabalhos. Um breve resumo sobre cada algoritmo é apresentado a seguir:

- **Árvore de Decisão (DT)** -Técnica de aprendizado supervisionado capaz de extrair regras de decisão simples sobre as características das amostras, representadas na forma de disjunções *SE ENTÃO*, inferidas a partir do conjunto de dados de entrada.
- **k-NN** - Algoritmo baseado em instâncias, utilizando como medida de similaridade a distância Euclidiana. Utiliza os k vizinhos mais próximos para predizer a classe de um instância de teste.
- ***Multilayer Perceptron* (MLP)** - Rede neural composta por mais de uma camada de neurônios, conectados entre si através de um peso. Em geral, para treinar a rede, utiliza-se o algoritmo *backpropagation*.
- ***Naïve Bayes* (NB)** - Utiliza de conceitos probabilísticos para predizer as classes de cada instância de teste. Possui como premissa a independência entre as características.
- ***Random Forest* (R. Forest)** - Técnica supervisionada que cria diversas árvores de decisão, de modo aleatório, e as combina com o intuito de obter a predição de uma instância de teste, com maior estabilidade e acurácia.
- ***Support Vector Machine* (SVM)** - Visa encontrar um hiperplano, em um espaço n-dimensional, para classificar instâncias de teste.

Para cada técnica utilizada, foram adotados os parâmetros definidos em seus respectivos artigos. A Tabela 2 apresenta os parâmetros para cada algoritmo.

Tabela 2 – Parâmetros utilizados para as técnicas k-NN, MLP, Random Forest e SVM

Algoritmo	Parâmetro	Valor
k-NN	k	5
	distância	euclidiana
MLP	Camadas Escondidas	1
	N. de Neurônios	100
	Função de Ativação	*relu*
	Taxa de aprendizagem	10^{-3}
	Iterações	200
R. Forest	*pool*	100
SVM	*kernel*	Gaussiano
	C	1.0
	Critério de parada	10^{-3}
	gamma	$\frac{1}{n.atributos + Var[X]}$

4.5 MEDIDAS DE COMPLEXIDADE

Lorena et al. (2018), Lorena et al. (2019) propuseram a biblioteca ECoL[1] (do inglês, ***E**xtended **Co**mplexity **L**ibrary*), que oferece a implementação das medidas de complexidades (do inglês, *complexity measures*) propostas por Ho e Basu (2002), assim como outras propostas na literatura. As medidas de complexidade são utilizadas para determinar o grau de dificuldade dos problemas de classificação a partir das perspectivas da geometria e distribuições das fronteiras de separação das classes (HO; BASU, 2002; LORENA et al., 2019).

Segundo Basu e Ho (2006), a complexidade dos problemas de classificação são atribuídas à 3 principais pilares: (i) a ambiguidade das classes; (ii) a complexidade das fronteiras de separação das classes; e (iii) a esparsidade e dimensionalidade dos dados. Essas métricas possibilitam a análise de cenários no qual um dado modelo preditivo obtém sucesso ou não (ALI; SMITH, 2006; FLORES; GÁMEZ; MARTÍNEZ, 2014; LUENGO; HERRERA, 2015; MUÑOZ et al., 2018). A Tabela 3 apresenta as propriedades das medidas de complexidades empregadas: valor mínimo (Mín.), valor máximo (Máx.) e custo computacional (Custo), sendo n o número de amostras presentes na base de dados, h o número de atributos e m a quantidade de classes.

[1] https://github.com/lpfgarcia/ECoL

Tabela 3 – Descrição das medidas de complexidades disponibilizadas a partir da biblioteca ECoL. Os valores das propriedades Mín., Máx. e Custo foram extraídos do trabalho (LORENA et al., 2019).

ID	Nome	Mín.	Máx.	Custo
F1	*Maximum Fisher's discriminant ratio*	≈ 0	1	$O(h \cdot n)$
F1v	*The directional-vector maximum Fisher's discriminant ratio*	≈ 0	1	$O(h \cdot n \cdot m + h^3 \cdot m^2)$
F2	*The overlap of the per-class bouding boxes*	0	1	$O(h \cdot n \cdot m)$
F3	*The maximum (individual) feature efficiency*	0	1	$O(h \cdot n \cdot m)$
F4	*The collective feature efficiency*	0	1	$O(h^2 \cdot n \cdot m)$
L1	*The minimized sum of the error distance of a linear classifier*	0	≈ 1	$O(n^2)$
L2	*The training error of a linear classifier*	0	1	$O(n^2)$
L3	*The nonlinearity of a linear classifier*	0	1	$O(n^2 + h \cdot l \cdot m)$
N1	*The fraction of points on the class boundary*	0	1	$O(h \cdot n^2)$
N2	*The ratio of average intra/inter class nearest neighbor distance*	0	≈ 1	$O(h \cdot n^2)$
N3	*The leave-one-out error rate of the one-nearest neighbor classifier*	0	1	$O(h \cdot n^2)$
N4	*The nonlinearity of the one-nearest neighbor classifier*	0	1	$O(h \cdot n^2 + h \cdot l \cdot n)$
T1	*The fraction of maximum covering spheres*	0	1	$O(h \cdot n^2)$
LSC	*Local set average cardinality*	0	1 - $\frac{1}{n}$	$O(h \cdot n^2)$
Density	*Density*	0	1	$O(h \cdot n^2)$
ClsCoef	*Clustering Coefficient*	0	1	$O(h \cdot n^2)$
Hubs	*Hubs*	0	1	$O(h \cdot n^2)$
T2	*The average number of points per dimension*	≈ 0	h	$O(h + n)$
T3	*Average number of PCA dimensions per points*	≈ 0	h	$O(h^2 \cdot n + h^3)$
T4	*Ratio of the PCA dimension to the original dimension*	0	1	$O(h^2 \cdot n + h^3)$
C1	*Entropy of classes proportions*	0	1	$O(n)$
C2	*Imbalance ratio*	0	1	$O(n)$

Ao definirem algumas medidas de complexidade, Ho e Basu (2002) limitaram sua aplicabilidade para problemas binários. Nesse contexto, para problemas multiclasses, Lorena et al. (2018) realizam uma análise para cada par de classes, ou seja, o problema multiclasse é decomposto em vários subproblemas binários, utilizando a abordagem "um-contra-um" (OVO, do inglês, *one-versus-one*). Portanto, as medidas de complexidade para problema multiclasse são definidas como a média dos valores entre os diferentes subproblemas (LORENA et al., 2019).

5 RESULTADOS

Nesse capítulo, os resultados obtidos após as execuções dos experimentos são analisados sobre as bases de dados definidas (Seção 4.1). Na Seção 5.1, avaliou-se a influência das técnicas utilizadas para estimar a quantidade de *clusters* para cada *dataset*. Na Seção 5.2 é investigado o procedimento de remoção dos *small clusters*, antes do treinamento do classificador PerC, para a melhor estimativa do parâmetro K. A Seção 5.3 apresenta uma comparativo entre as diferentes regras de agregação definidas. Na Seção 5.4, o sPerC tem seu desempenho comparado com os algoritmos do estado-da-arte. Por fim, na Seção 5.5, apresentamos características presentes nas bases de dados avaliadas, buscando justificar quais cenários favorecem a utilização da técnica sPerC em relação à técnica PerC.

5.1 ANÁLISE DA ESTIMATIVA DO VALOR DE K

Nessa seção, vamos analisar a influência dos *clustering validation* índices, definidos na Seção 4.2, sobre o sPerC. Para esse cenário, todos os *clusters* gerados ($\tau = 100\%$) são utilizados para treinar o classificador PerC. As perturbações produzidas a partir da inserção de uma amostra são agregadas através da regra do mínimo, conforme definido originalmente para o classificador PerC. A Tabela 4 apresenta os resultados para a métrica AUC, sobre o conjunto de treinamento, e o número de *clusters* gerados para os *clustering validation* índices. Os resultados encontrados para a estratégia de Combinação encontram-se na última coluna da tabela.

Ao averiguar os resultados apresentados na Tabela 4, nota-se que a abordagem de Combinação (AUC = 0,768) alcança, em média, desempenho superior ao ser comparada com os índices *Calinski-Harabasz* (AUC = 0,685), *Davies-Bouldin* (AUC = 0,657), *Gap Statistic* (AUC = 0,690) e *Silhouette* (AUC = 0,706). Analisando o número de *clusters* gerados, a estratégia de Combinação (13,84), em média, estima valores de K superiores quando comparada aos métodos de validação de *clusters Calinski-Harabasz* (08,78), *Davies-Bouldin* (12,77) e *Silhouette* (08,82), sendo inferior apenas a estimativa *Gap Statistic* (20,26).

Um ponto de destaque pode ser reparado ao investigar o comportamento da abordagem Combinação. Com exceção de 5 conjuntos de dados, sendo eles *banknote, ecoli3, indian-liver-patient, poker-8-9vs5* e *shuttle-6vs2-3*, observa-se que a estratégia mencionada

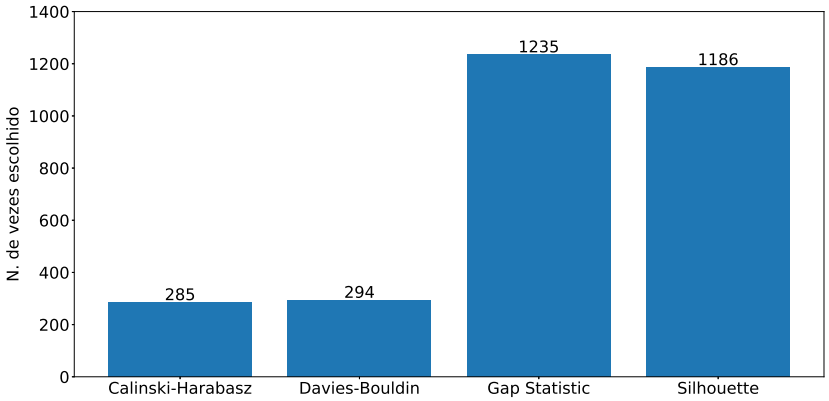
Tabela 4 – Valores da métrica AUC no conjunto de treinamento para os *clustering validation* índices *Calinski-Harabasz, Davies-Bouldin, Gap Statistic, Silhouette* e Combinação, assim como o número médio de *clusters* gerados (entre parênteses). Os melhores resultados encontram-se em destaque.

Dataset	*Calinski-Harabasz*	*Davies-Bouldin*	*Gap Statistic*	*Silhouette*	Combinação
balance	0,822 (11,91)	0,826 (22,83)	0,826 (11,90)	0,816 (12,65)	**0,839 (16,76)**
banana	0,883 (31,09)	0,892 (08,92)	0,888 (26,65)	0,885 (06,02)	**0,893 (14,71)**
banknote	0,990 (11,88)	0,995 (12,84)	**1,000 (31,56)**	0,997 (04,00)	**1,000 (26,86)**
bupa	0,654 (04,00)	0,647 (10,95)	0,748 (14,69)	0,654 (04,00)	**0,761 (13,84)**
cleveland	0,537 (11,96)	0,503 (24,60)	0,595 (18,66)	0,511 (23,95)	**0,602 (17,17)**
cleveland-0vs4	0,617 (04,80)	0,512 (11,70)	0,831 (09,52)	0,576 (11,50)	**0,846 (08,28)**
ecoli-0-1-4-7vs2-3-5-6	0,755 (04,17)	0,512 (15,31)	0,690 (12,55)	0,734 (04,79)	**0,809 (06,75)**
ecoli1	0,597 (07,68)	0,533 (07,96)	0,540 (12,70)	0,785 (06,60)	**0,802 (06,94)**
ecoli2	0,685 (04,94)	0,652 (08,11)	0,554 (12,91)	0,782 (06,76)	**0,825 (07,04)**
ecoli3	**0,500 (07,73)**	**0,500 (06,00)**	**0,500 (12,34)**	**0,500 (06,00)**	**0,500 (06,00)**
glass-0-1-5vs2	0,509 (05,50)	0,503 (09,43)	0,509 (09,56)	0,503 (09,51)	**0,517 (09,29)**
hayes-roth	0,718 (07,87)	0,685 (13,77)	0,726 (09,53)	0,718 (11,38)	**0,757 (10,16)**
indian-liver-patient	0,500 (04,00)	0,500 (04,00)	**0,531 (19,30)**	0,500 (04,00)	**0,531 (18,54)**
led7digit-0-2-4-5-6-7-8-9vs1	0,517 (11,45)	0,507 (15,16)	0,652 (15,00)	0,511 (18,90)	**0,663 (16,40)**
new-thyroid1	0,964 (04,50)	0,949 (06,08)	0,958 (09,06)	0,956 (04,88)	**0,992 (08,42)**
newthyroid2	0,964 (04,54)	0,950 (06,13)	0,961 (08,99)	0,956 (04,90)	**0,991 (08,34)**
page-blocks0	0,910 (05,94)	0,766 (14,74)	0,572 (59,09)	0,905 (05,81)	**0,917 (14,41)**
poker-8-9vs5	0,500 (04,00)	0,500 (04,01)	**0,512 (32,06)**	0,500 (04,00)	**0,512 (14,68)**
poker-8-9vs6	0,500 (04,00)	0,973 (23,17)	0,963 (24,95)	0,500 (04,00)	**0,984 (24,17)**
shuttle-6vs2-3	0,983 (11,30)	0,713 (09,77)	0,985 (10,47)	**0,986 (09,30)**	**0,986 (09,31)**
transfusion	0,571 (06,29)	0,586 (08,89)	0,607 (21,89)	0,622 (04,00)	**0,640 (14,57)**
vertebral2c	0,693 (04,00)	0,522 (08,05)	0,525 (13,49)	0,693 (04,00)	**0,698 (04,74)**
vowel	0,993 (54,82)	0,986 (56,78)	0,985 (62,78)	0,993 (53,15)	**0,995 (53,53)**
winequality-red-4	0,500 (04,90)	0,506 (09,11)	0,522 (28,39)	0,506 (04,01)	**0,527 (07,64)**
winequality-red-8vs6-7	0,495 (04,00)	0,501 (06,38)	0,613 (20,06)	0,495 (04,00)	**0,615 (19,54)**
winequality-white-3-9vs5	0,600 (04,76)	0,708 (05,82)	0,718 (26,85)	0,629 (05,05)	**0,756 (15,67)**
wisconsin	0,939 (05,00)	0,737 (18,08)	0,578 (22,44)	0,943 (04,89)	**0,961 (08,54)**
yeast-0-5-6-7-9vs4	0,547 (06,15)	0,522 (11,07)	0,552 (16,12)	0,701 (05,92)	**0,724 (07,37)**
yeast-1-4-5-8vs7	0,499 (04,97)	0,510 (10,66)	0,560 (18,04)	0,504 (08,87)	**0,561 (17,80)**
yeast-2vs4	0,616 (05,29)	0,508 (12,88)	0,512 (16,21)	0,820 (07,77)	**0,830 (07,72)**
Média	0,685 (08,78)	0,657 (12,77)	0,690 (20,26)	0,706 (08,82)	0,768 (13,84)

encontra os resultados mais satisfatórios para as bases de dados avaliadas. Visando demonstrar a influência dos *clustering validation* índices no desempenho da estimativa de Combinação, a Figura 11 expõe a escolha dos índices, em cada ciclo de iteração, para todas as bases de dados avaliadas no experimento. O eixo x apresenta as estimativas Calanski-Harabasz, Davies-Bouldin, Gap-Statistic e Silhouette, enquanto o eixo y exibe a quantidade de ocorrências que os métodos de validação de *clusters* foram escolhidos, para cada *fold* executado.

Analisando a Figura 11, o índice Gap Statistic apresenta maior influência, sobre o método proposto, quando confrontado as demais estimativas. Entretanto, tal comportamento não é refletido no desempenho médio do índice, onde a estimativa Silhouette apresenta valor

Figura 11 – Escolha do *clustering validation* índice, para cada *fold* do conjunto de treinamento, ao utilizar a estratégia de Combinação.

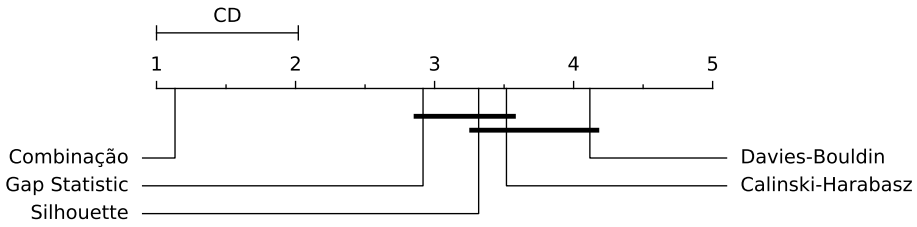


médio para a métrica AUC superior ao índice Gap Statistic. Da mesma forma, a estimativa Calinsk-Harabasz possui menor influência quando aplicada a estratégia de Combinação. Contudo, seu desempenho médio está muito próximo do resultado obtido para o *clustering validation* índice Gap Statistic.

Dado as explanações, nenhuma conclusão pode ser inferida sem a aplicação de um teste estatístico. Para tal, o teste de Friedman (com nível de significância $\alpha = 5\%$) foi empregado para verificar se existe diferença significativa entre os resultados encontrados para os *clustering validation* índices avaliados, em conjunto com a estratégia de Combinação. O resultado do teste de Friedman (*p-value* $= 1{,}680 \times 10^{-13}$) indica que existe diferença significativa para os resultados alcançados.

A Figura 12 apresenta o diagrama CD para o *post-hoc* teste de Bonferroni-Dunn, cujo menor *ranking* apresenta um melhor desempenho entre as estimativas avaliadas, e índices conectados por um segmento de reta não possuem diferença estatística significativa.

Figura 12 – Diagrama CD do *post-hoc* de Bonferroni-Dunn para os *clustering validation* índices sobre as 30 bases de dados, sendo CD $= 1{,}020$.



Como pode-se observar, os resultados obtidos para a estratégia de Combinação apresentam diferenças significativas em relação aos índices *Calinski-Harabasz*, *Davies-Bouldin*, *Gap Statistic* e *Silhouette*. Além disso, a abordagem de Combinação (1,133) obtém menor *ranking* entre os índices avaliados, seguida dos índices *Gap Statistic* (2,917), *Silhouette* (3,317), *Calinski-Harabasz* (3,517) e *Davies-Bouldin* (4,117). Ou seja, a abordagem de Combinação alcança um melhor desempenho para as bases de dados utilizadas. Portanto, por apresentar menor *ranking*, e desempenho médio superior aos demais índices, a estratégia de *Combinação* dos *clustering validation* índices será empregada nos demais experimentos.

5.2 ANÁLISE DA REMOÇÃO DOS *SMALL CLUSTERS*

Nessa seção avaliou-se a influência da remoção dos *small clusters* no desempenho da arquitetura proposta. A estratégia de Combinação, definida na Seção 5.1, foi utilizada para selecionar o *clustering validation* índice mais satisfatório a cada ciclo de iteração. Para agregar as perturbações provocadas pelas instâncias de teste, a *regra do mínimo* foi novamente empregada. Os resultados obtidos encontram-se na Tabela 5, para a métrica AUC, sobre o conjunto de treinamento, e na Tabela 6, para o número médio de *clusters* usados durante a fase de treinamento, assim como o número médio de grupos removidos. Em ambas as tabelas, a última coluna ($\tau =$ "Auto") apresenta os resultados alcançados pelo procedimento de seleção automática do parâmetro τ, definido na Seção 4.3.

Analisando os dados da Tabela 5, em média, para os valores de $\tau = 10\%, 20\%, 30\%, \ldots,$ 100%, a arquitetura proposta alcança desempenho médio superior à medida que o valor do parâmetro é incrementado. Ou seja, conforme o método sPerC categoriza mais grupos como *large clusters*, o classificador PerC reduz sua taxa de erro (exceto para as bases de dados *cleveland, ecoli-0-1-4-6vs2-3-5-6, ecoli1, ecoli2, ecoli3, glass-0-1-5vs2, indian-liver-patient, led7digit-0-2-4-5-6-7-8-9vs1, page-blocks0, vertebral2c, winequality-red-4, yeast-0-5-6-7-9vs4, yeast-2vs4*). Tal resultado mostra-se coerente aos trabalhos publicados na literatura (HOLTE; ACKER; PORTER, 1989; WEISS; HIRSH, 2000), onde os autores justificam que a remoção de todos os *small disjuncts* (*small clusters*) acaba degradando a performance global dos modelos preditivos.

Ao analisar os resultados para seleção automática para o valor do parâmetro τ (coluna $\tau =$ "Auto"), a arquitetura proposta obtém desempenho médio superior à configuração $\tau = 100\%$. Observando com mais detalhes os resultados apresentados para a configuração

τ = "Auto", nota-se que em 21 das 30 bases de dados avaliadas, seu desempenho foi superior ao ser comparada a configuração $\tau = 100\%$. Nesses casos, a remoção de alguns *small clusters* (*small disjuncts*) incrementou o desempenho do classificador PerC. Para os conjuntos de dados *balance*, *banana*, *banknote*, *poker-8-9vs5*, *poker-8-9vs6*, *shuttle-6vs2-3*, *winequality-red-8vs6-7*, *winequality-white-3-9vs5* e *yeast-1-4-5-8vs7*, ambas as configurações alcançaram desempenho similares.

Investigando os resultados apresentados para o número médio de *clusters* removidos (Tabela 6), a configuração τ = "Auto", em média, descarta 2,94 *clusters* ($\approx$ 3 partições) ao ser comparada a configuração $\tau = 100\%$, no qual utiliza todos os *clusters* durante a fase de treinamento. Vale observar que o número médio de *clusters* removidos não é inversamente proporcional a medida que o valor do parâmetro τ é incrementado. Este comportamento pode ser justificado pela aplicação da estratégia de Combinação dos *clustering validation* índices, visto que a cada ciclo de iteração as medidas estimam diferente valores do parâmetro K, para o algoritmo *K-Means*. Por outro lado, a quantidade média de *clusters* utilizados na etapa de treinamento incrementa conforme o valor do parâmetro τ aumenta (com exceção da configuração τ = "Auto").

Após as explanações mencionadas anteriormente, nenhuma conclusão a cerca dos resultados encontrados pode ser realizada sem a aplicação de um teste de hipótese não paramétrico. Para tal, o teste de Friedman (com nível de significância $\alpha = 5\%$) foi empregado para verificar se existe dominância entre os resultados exibidos na Tabela 5. O resultado do teste de Friedman (*p-value* = 6,242 $\times 10^{-45}$) apresenta evidências de dominância entre os resultados analisados.

A Figura 13 apresenta o diagrama CD para o *post-hoc* teste de Nemenyi, cujo menor *ranking* apresenta um melhor desempenho entre as configurações avaliadas, e parâmetros conectadas por um segmento de reta não possuem diferenças estatísticas significativas.

Analisando a Figura 13, a configuração τ = "Auto" não apresenta diferença estatística significativa em relação as configurações $\tau = 90\%$ e $\tau = 100\%$. Nesse cenário, a configuração τ = "Auto" (1,150) obtém menor *ranking* comparada as demais configurações. As configurações $\tau = 10\%, 20\%, 30\%, \ldots, 80\%$ apresentam *ranking* superiores as configurações mencionadas inicialmente, apontando diferenças estatísticas significativas entre seus resultados (com exceção das configurações τ = 70% e 80%). Sendo assim, por apresentar menor *ranking*, e maior valor médio para a métrica AUC, a configuração τ = "Auto", que determina automaticamente o valor do parâmetro τ, será empregada nos demais experimentos.

Figura 13 – Diagrama CD do *post-hoc* teste de Nemenyi para os diferentes valores de τ sobre as 30 bases de dados, sendo CD $= 2{,}756$.

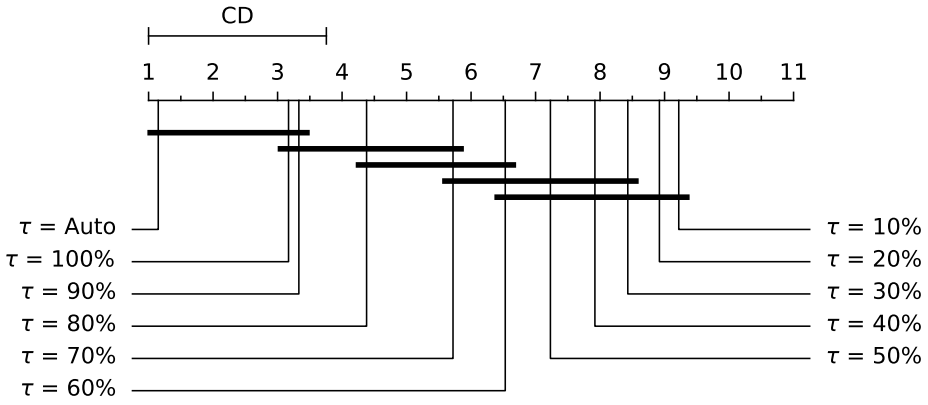


Tabela 5 – Valores da métrica AUC no conjunto de treinamento para a variação do parâmetro τ. Os melhores resultados encontram-se em destaque.

Dataset	$\tau = 10\%$	$\tau = 20\%$	$\tau = 30\%$	$\tau = 40\%$	$\tau = 50\%$	$\tau = 60\%$	$\tau = 70\%$	$\tau = 80\%$	$\tau = 90\%$	$\tau = 100\%$	$\tau =$ "Auto"
balance	0,545	0,660	0,686	0,713	0,727	0,746	0,767	0,782	0,797	**0,839**	**0,839**
banana	0,500	0,500	0,578	0,650	0,689	0,761	0,763	0,852	0,858	**0,893**	**0,893**
banknote	0,553	0,834	0,861	0,918	0,946	0,962	0,971	0,980	0,997	**1,000**	**1,000**
bupa	0,500	0,553	0,582	0,607	0,635	0,663	0,688	0,713	0,737	0,761	**0,768**
cleveland	0,500	0,507	0,534	0,592	0,631	0,665	0,699	0,726	0,737	0,602	**0,741**
cleveland-0vs4	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,505	0,589	0,846	**0,861**
ecoli-0-1-4-7vs2-3-5-6	0,500	0,500	0,500	0,500	0,500	0,500	0,548	0,838	0,879	0,809	**0,895**
ecoli1	0,500	0,535	0,583	0,645	0,713	0,777	0,865	0,874	0,880	0,802	**0,902**
ecoli2	0,500	0,500	0,500	0,503	0,511	0,554	0,649	0,839	0,901	0,825	**0,903**
ecoli3	0,500	0,500	0,500	0,513	0,531	0,605	0,693	0,783	0,808	0,500	**0,820**
glass-0-1-5vs2	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,556	0,668	0,517	**0,669**
hayes-roth	0,506	0,529	0,562	0,579	0,613	0,633	0,664	0,701	0,734	0,757	**0,765**
indian-liver-patient	0,500	0,500	0,500	0,503	0,512	0,520	0,529	0,536	0,541	0,531	**0,543**
led7digit-0-2-4-5-6-7-8-9vs1	0,500	0,500	0,500	0,500	0,500	0,500	0,511	0,798	0,839	0,663	**0,847**
new-thyroid1	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,981	0,987	0,992	**0,994**
newthyroid2	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,977	0,983	0,991	**0,993**
page-blocks0	0,500	0,500	0,500	0,502	0,543	0,648	0,677	0,695	0,922	0,917	**0,929**
poker-8-9vs5	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500	**0,512**	**0,512**
poker-8-9vs6	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500	**0,984**	**0,984**
shuttle-6vs2-3	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500	**0,986**	**0,986**
transfusion	0,500	0,503	0,532	0,568	0,576	0,600	0,599	0,611	0,633	0,640	**0,655**
vertebral2c	0,500	0,506	0,760	0,754	0,744	0,755	0,776	0,791	0,823	0,698	**0,827**
vowel	0,553	0,605	0,659	0,710	0,761	0,810	0,857	0,905	0,953	0,995	**0,997**
winequality-red-4	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,539	0,527	**0,558**
winequality-red-8vs6-7	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500	**0,615**	**0,615**
winequality-white-3-9vs5	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500	**0,756**	**0,756**
wisconsin	0,500	0,500	0,500	0,500	0,585	0,966	0,969	0,969	0,957	0,961	**0,977**
yeast-0-5-6-7-9vs4	0,500	0,500	0,500	0,500	0,511	0,515	0,528	0,560	0,777	0,724	**0,787**
yeast-1-4-5-8vs7	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,502	**0,561**	**0,561**
yeast-2vs4	0,500	0,500	0,500	0,500	0,503	0,591	0,721	0,815	0,891	0,830	**0,893**
Média	0,505	0,524	0,545	0,559	0,574	0,609	0,632	0,710	0,748	0,768	0,816

Tabela 6 – Número médio de *clusters* utilizados na etapa de treinamento para cada valor de τ avaliado, assim como a quantidade média de *clusters* removidos (entre parênteses).

Dataset	$\tau = 10\%$	$\tau = 20\%$	$\tau = 30\%$	$\tau = 40\%$	$\tau = 50\%$	$\tau = 60\%$	$\tau = 70\%$	$\tau = 80\%$	$\tau = 90\%$	$\tau = 100\%$	$\tau =$ "Auto"
balance	01,24 (14,18)	02,64 (13,94)	04,07 (12,89)	05,53 (11,98)	06,92 (11,27)	09,01 (10,37)	12,00 (09,20)	14,42 (07,38)	17,32 (04,92)	16,76 (00,00)	16,88 (00,05)
banana	01,00 (05,02)	01,00 (05,02)	02,02 (04,70)	03,00 (04,11)	03,95 (04,88)	04,09 (02,13)	05,09 (02,95)	05,01 (01,02)	08,20 (02,19)	14,71 (00,00)	14,71 (00,00)
banknote	01,27 (10,30)	03,73 (25,56)	05,77 (24,51)	07,28 (19,67)	09,98 (19,10)	12,56 (16,43)	13,71 (11,50)	14,45 (07,36)	06,29 (00,62)	26,86 (00,00)	26,43 (00,33)
bupa	01,00 (03,00)	01,55 (08,68)	02,27 (10,52)	03,17 (11,01)	03,99 (09,96)	04,63 (08,04)	06,17 (07,76)	07,75 (06,52)	09,98 (04,50)	13,84 (00,00)	13,89 (00,77)
cleveland	01,00 (22,95)	02,00 (21,51)	03,46 (19,73)	04,86 (15,82)	06,54 (13,95)	08,53 (11,90)	10,90 (09,79)	12,69 (06,73)	15,00 (03,77)	17,17 (00,00)	14,17 (04,61)
cleveland-0vs4	01,00 (10,50)	01,02 (10,48)	02,00 (09,50)	02,78 (08,72)	03,19 (08,31)	04,24 (07,26)	05,52 (05,98)	06,74 (04,76)	08,74 (02,76)	08,28 (00,00)	08,46 (00,38)
ecoli-0-1-4-7vs2-3-5-6	01,00 (03,79)	01,00 (03,79)	01,01 (03,78)	01,01 (03,78)	01,09 (03,70)	01,10 (03,69)	02,69 (03,50)	07,32 (06,23)	09,57 (04,56)	06,75 (00,00)	09,56 (04,29)
ecoli1	01,00 (05,60)	01,15 (06,39)	01,57 (06,90)	02,39 (07,34)	03,82 (06,91)	05,04 (06,69)	05,31 (05,04)	06,80 (04,69)	07,62 (02,79)	06,94 (00,00)	06,57 (01,62)
ecoli2	01,00 (05,76)	01,00 (05,76)	01,00 (05,76)	01,03 (05,81)	01,96 (04,99)	02,59 (05,13)	03,90 (05,57)	06,48 (05,38)	08,52 (03,83)	07,04 (00,00)	08,55 (03,67)
ecoli3	01,00 (05,00)	01,00 (05,00)	01,00 (05,00)	01,12 (05,19)	02,25 (04,41)	03,01 (05,18)	04,72 (05,07)	06,38 (05,26)	07,60 (04,07)	06,00 (00,00)	06,11 (05,56)
glass-0-1-5vs2	01,00 (08,51)	01,00 (08,51)	01,00 (08,51)	01,00 (08,51)	01,00 (08,51)	01,00 (08,51)	01,98 (07,53)	03,66 (05,96)	05,67 (03,90)	09,29 (00,00)	05,49 (04,00)
hayes-roth	01,14 (10,79)	01,63 (10,70)	02,45 (07,38)	03,13 (07,49)	04,17 (07,90)	05,47 (06,36)	06,74 (05,07)	08,33 (03,47)	09,56 (01,69)	10,16 (00,00)	09,67 (00,88)
indian-liver-patient	01,00 (03,00)	01,00 (03,00)	01,00 (03,00)	01,20 (03,66)	02,58 (06,34)	06,15 (09,94)	08,81 (09,94)	11,04 (07,84)	13,89 (04,99)	18,54 (00,00)	14,43 (04,59)
led7digit-0-2-4-5-6-7-8-9vs1	01,02 (17,88)	02,04 (16,86)	03,08 (15,82)	04,19 (14,71)	05,40 (13,50)	06,80 (12,10)	08,07 (10,75)	09,59 (08,19)	11,59 (05,28)	16,40 (00,00)	10,52 (05,50)
new-thyroid1	01,00 (03,88)	01,00 (03,88)	01,00 (03,88)	01,00 (03,88)	01,00 (03,88)	01,00 (03,88)	01,00 (03,88)	04,16 (04,71)	05,61 (02,94)	08,42 (00,00)	06,26 (02,51)
newthyroid2	01,00 (03,90)	01,00 (03,90)	01,00 (03,90)	01,00 (03,90)	01,00 (03,90)	01,00 (03,90)	01,00 (03,90)	04,06 (04,75)	05,46 (02,95)	08,34 (00,00)	06,45 (02,27)
page-blocks0	01,00 (04,81)	01,00 (04,81)	01,00 (04,81)	01,09 (05,25)	03,90 (14,81)	14,13 (36,62)	21,13 (37,97)	27,19 (31,91)	35,94 (23,16)	14,41 (00,00)	35,14 (16,06)
poker-8-9vs5	01,00 (03,00)	01,00 (03,00)	01,00 (03,00)	01,00 (03,00)	01,00 (03,00)	02,00 (02,00)	02,00 (02,00)	02,00 (02,00)	02,00 (02,00)	14,68 (00,00)	12,82 (01,86)
poker-8-9vs6	01,00 (03,00)	01,00 (03,00)	01,00 (03,00)	01,00 (03,00)	01,02 (02,98)	02,00 (02,00)	02,00 (02,00)	02,00 (02,00)	02,00 (02,00)	24,17 (00,00)	24,17 (00,00)
shuttle-6vs2-3	01,00 (08,30)	01,00 (08,30)	02,00 (07,30)	02,12 (07,18)	03,00 (06,30)	03,91 (05,39)	04,31 (04,99)	05,30 (04,00)	07,20 (02,10)	09,31 (00,00)	09,31 (00,00)
transfusion	01,00 (03,00)	01,07 (03,73)	01,71 (07,02)	03,44 (11,92)	05,15 (13,16)	07,27 (13,08)	09,33 (11,64)	08,28 (06,07)	11,94 (05,20)	14,57 (00,00)	11,66 (06,31)
vertebral2c	01,00 (03,00)	01,03 (03,24)	01,90 (05,97)	02,29 (05,58)	03,25 (05,08)	03,93 (04,76)	04,89 (03,68)	06,00 (02,90)	06,98 (01,64)	04,74 (00,00)	06,51 (02,35)
vowel	04,08 (53,12)	08,26 (48,97)	12,67 (46,83)	16,92 (39,92)	21,17 (33,80)	25,57 (29,19)	30,76 (24,07)	37,72 (19,08)	46,09 (12,32)	53,53 (00,00)	53,32 (00,27)
winequality-red-4	01,00 (03,01)	01,00 (03,01)	01,00 (03,01)	01,00 (03,01)	01,00 (03,01)	02,00 (02,01)	02,00 (02,01)	02,00 (02,01)	07,36 (04,24)	07,64 (00,00)	10,73 (03,10)
winequality-red-8vs6-7	01,00 (03,00)	01,00 (03,00)	01,00 (03,00)	01,00 (03,00)	01,25 (02,75)	02,00 (02,00)	02,00 (02,00)	02,00 (02,00)	02,00 (02,00)	19,54 (00,00)	18,94 (00,09)
winequality-white-3-9vs5	01,00 (04,05)	01,00 (04,05)	01,00 (04,05)	01,00 (04,05)	01,19 (03,86)	02,00 (03,05)	02,00 (03,05)	02,00 (03,05)	02,00 (03,05)	15,67 (00,00)	15,67 (00,00)
wisconsin	01,00 (03,89)	01,00 (03,89)	01,01 (03,88)	01,63 (03,26)	02,21 (05,92)	06,02 (16,78)	04,48 (06,36)	04,26 (04,09)	08,58 (04,35)	08,54 (00,00)	09,15 (07,07)
yeast-0-5-6-7-9vs4	01,00 (04,92)	01,00 (04,92)	01,04 (04,88)	02,00 (03,92)	02,13 (04,22)	02,29 (04,15)	03,11 (03,72)	04,30 (03,81)	07,97 (04,86)	07,37 (00,00)	08,30 (03,88)
yeast-1-4-5-8vs7	01,00 (07,87)	01,00 (07,87)	01,00 (07,87)	01,11 (07,76)	01,79 (07,08)	01,90 (06,97)	02,78 (06,09)	02,86 (06,01)	04,29 (05,27)	17,80 (00,00)	16,68 (00,47)
yeast-2vs4	01,00 (06,77)	01,00 (06,77)	01,00 (06,77)	01,94 (05,83)	02,03 (05,84)	02,94 (07,10)	04,89 (07,66)	06,35 (06,93)	08,39 (05,69)	07,72 (00,00)	08,37 (05,62)
Média	01,12 (08,19)	01,50 (08,72)	02,07 (08,57)	02,71 (08,08)	03,63 (08,11)	05,14 (08,55)	06,44 (07,49)	08,04 (06,20)	10,11 (04,45)	13,84 (00,00)	13,96 (02,94)

5.3 COMPARAÇÃO ENTRE AS REGRAS DE AGREGAÇÃO

Nessa seção será analisado o impacto da escolha da regra de agregação, definidas na Seção 3.2.1, utilizada para combinar as perturbações provocadas pelas inserção de uma instâncias de teste. Nesse experimento, a estratégia de Combinação dos *clustering validation* índices é empregada, assim como o procedimento para determinar o valor do parâmetro τ ($\tau =$ "Auto") automaticamente. A Tabela 7 exibe os resultados para a métrica AUC, sobre o conjunto de validacão e o número médio de *clusters* utilizados no treinamento do classificador PerC. A última linha apresenta a pontuação *Win/Tie/Loss*, isto é, a quantidade de vezes que a regra de agregação Mínimo apresenta resultados superiores quando comparada as regras de agregação Média, Mediana e *Cluster*, separadamente.

Ao averiguar os resultados expostos na Tabela 7, nota-se que a regra de agregação Mínimo (AUC $=$ 0,816) encontra, em média, desempenho inferior quando comparada a regra de agregação *Cluster* (AUC $=$ 0,844). Por outro lado, a agregação Mínimo supera os resultados alcançados pelas regras de agregação Mediana (AUC $=$ 0,762) e Média (AUC $=$ 0,732).

Analisando a pontuação *Win/Tie/Loss*, as regras de agregação Média e Mediana obtém desempenhos superiores em 26,60% (8/30) das bases de dados, e desempenho inferiores em 73,40% (22/30) dos *datasets*, ao serem comparadas à regra de agregação Mínimo. Já em relação à regra de agregação *Cluster*, a regra de agregação Mínimo obtém desempenho superior em apenas 1 dos conjunto de dados (*wisconsin*). Contudo, em 90% (27/30) dos *datasets* a regra de agregação *Cluster* alcançou desempenho superior comparada a regra de agregação Mínimo. Para as bases *banana* e *banknote* (7% das bases de dados avaliadas), ambos os desempenhos são equivalentes. Em relação ao número médio de *clusters* utilizados para treinar o classificador PerC, as regras de agregação Mediana (10,44) e Média (8,27) apresentam valores inferiores quando comparadas as regras de agregação Mínimo (13,96) e *Cluster* (17,60), destacando a influência da remoção dos *small clusters*, como demonstrado na Seção 5.2, e da utilização da estratégia de Combinação, como avaliado na Seção 5.1.

Dada as explanações mencionadas acima, nenhum conclusão a cerca dos resultados pode ser realizada sem a aplicação de um teste estatístico. Para tal, o teste de hipótese não paramétrico de Friedman (com nível de significância $\alpha = 5\%$) foi aplicado com o objetivo de verificar se existe dominância entre os resultados encontrados para a métrica AUC. Como resultado do teste de Friedman (*p-value* $=$ 7,288 $\times 10^{-9}$), é possível concluir que existem evidências de dominância nas amostras analisadas. Desse modo, o *post-hoc* teste de

Bonferroni-Dunn foi empregado.

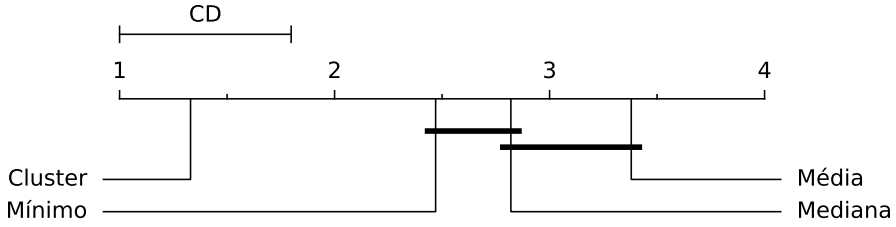
Tabela 7 – Valores da métrica AUC no conjunto de treinamento para as regras de agregação Mínimo, Média, Media e *Cluster*, assim como o número médio de partições utilizadas durante a etapa de treinamento. Os melhores resultados encontram-se em destaque.

Dataset	Mínimo	Média	Mediana	*Cluster*
balance	0,839 (16,88)	0,806 (12,41)	0,788 (12,55)	**0,840 (16,77)**
banana	**0,893 (14,71)**	0,597 (06,65)	0,593 (06,93)	**0,893 (14,91)**
banknote	**1,000 (26,43)**	0,930 (03,00)	0,955 (14,38)	**1,000 (26,98)**
bupa	0,768 (13,89)	0,630 (02,83)	0,629 (02,74)	**0,789 (14,88)**
cleveland	0,741 (14,17)	0,674 (06,83)	0,693 (07,94)	**0,812 (15,18)**
cleveland-0vs4	0,861 (08,46)	0,811 (04,80)	0,772 (05,27)	**0,896 (09,02)**
ecoli-0-1-4-7vs2-3-5-6	0,895 (09,56)	0,659 (08,47)	0,778 (06,18)	**0,896 (10,41)**
ecoli1	0,902 (06,57)	0,855 (03,79)	0,863 (06,37)	**0,914 (08,60)**
ecoli2	0,903 (08,55)	0,863 (05,18)	0,891 (06,69)	**0,911 (10,12)**
ecoli3	0,820 (06,11)	0,893 (04,04)	**0,916 (04,17)**	0,821 (08,44)
glass-0-1-5vs2	0,669 (05,49)	0,782 (03,86)	**0,882 (07,23)**	0,708 (07,46)
hayes-roth	0,765 (09,67)	0,660 (06,67)	0,668 (07,62)	**0,823 (11,41)**
indian-liver-patient	0,543 (14,43)	**0,636 (14,77)**	0,592 (08,50)	0,602 (17,74)
led7digit-0-2-4-5-6-7-8-9vs1	0,847 (10,52)	0,848 (10,66)	0,872 (10,89)	**0,879 (11,44)**
new-thyroid1	0,994 (06,26)	0,987 (04,95)	0,987 (04,68)	**0,995 (06,58)**
newthyroid2	0,993 (06,45)	0,987 (04,77)	0,986 (04,64)	**0,994 (06,67)**
page-blocks0	0,929 (35,14)	0,779 (21,82)	0,884 (37,31)	**0,952 (59,10)**
poker-8-9vs5	0,512 (12,82)	0,514 (09,94)	0,518 (11,89)	**0,584 (31,17)**
poker-8-9vs6	0,984 (24,17)	0,507 (06,73)	0,538 (11,91)	**0,988 (24,07)**
shuttle-6vs2-3	0,986 (09,31)	0,500 (01,00)	0,530 (06,56)	**0,998 (09,33)**
transfusion	0,655 (11,66)	0,635 (06,22)	**0,661** (11,17)	0,659 (13,69)
vertebral2c	0,827 (06,51)	0,820 (04,08)	0,823 (04,47)	**0,833 (07,18)**
vowel	0,997 (53,32)	0,603 (12,56)	0,609 (21,06)	**1,000 (53,51)**
winequality-red-4	0,558 (10,73)	0,590 (10,84)	**0,716 (09,73)**	0,676 (28,39)
winequality-red-8vs6-7	0,615 (18,94)	0,678 (18,09)	**0,852 (20,06)**	0,687 (20,14)
winequality-white-3-9vs5	0,756 (15,67)	0,551 (19,56)	0,685 (23,37)	**0,827 (26,85)**
wisconsin	**0,977 (09,15)**	0,968 (08,89)	0,971 (13,55)	0,976 (08,43)
yeast-0-5-6-7-9vs4	0,787 (08,30)	0,767 (05,79)	0,786 (07,15)	**0,833 (15,42)**
yeast-1-4-5-8vs7	0,561 (16,68)	0,575 (13,63)	0,542 (08,17)	**0,596 (18,04)**
yeast-2vs4	0,893 (08,37)	0,871 (05,42)	0,886 (09,96)	**0,932 (15,96)**
Média	0,816 (13,96)	0,733 (8,27)	0,762 (10,44)	0,844 (17,60)
Win/Tie/Loss	n/a	22/0/8	22/0/8	1/2/27

A Figura 14 exibe o diagrama CD para o *post-hoc* teste de Bonferroni-Dunn, cujo menor *ranking* apresenta um melhor desempenho entre as agregações avaliadas, e regras de agregação conectados por um segmento de reta não possuem diferenças estatísticas significativas.

Conforme exibido na Figura 14, a regra de agregação *Cluster* apresenta diferenças estatísticas significativas em relação as demais agregações. Além disso, a regra de agregação *Cluster* (1,330) apresenta menor *ranking*, seguida das regras de agregação Mínimo (2,470), Mediana (2,820) e Média (3,380). Por apresentar menor *ranking*, e desempenho médio superior, a regra de agregação *Cluster* será utilizada pela arquitetura proposta para combinar as perturbações provocadas pelas amostras de teste.

Figura 14 – Diagrama CD do *post-hoc* teste de Bonferroni-Dunn para as regras de agregação avaliadas, sobre as 30 bases de dados, sendo CD = 0,798.



5.4 COMPARAÇÃO COM ALGORITMOS DO ESTADO-DA-ARTE

Nessa seção o objetivo é comparar o desempenho da arquitetura proposta *subconcept PerC* (sPerC) com os algoritmos do estado-da-arte (vide Seção 4.4). Em relação ao método proposto, o número de *clusters* é estimado a partir da combinação dos *clustering validation* índices (vide Seção 5.1), o valor do parâmetro τ é determinado automaticamente, como definido na Seção 5.2, e as perturbações provocadas pelas amostras de testes são agregadas através da regra *Cluster* (como apresentado na Seção 5.3). A Tabela 8 apresenta os resultados alcançados para a métrica AUC. O Apêndice A apresenta os resultados encontrados para as métricas Acurácia (Tabela 11) e *F-Measure* (Tabela 12). A linha *Wilcoxon-p*, nas Tabelas 8, 11 e 12, apresenta o resultado do teste de hipótese de *Wilcoxon*, enquanto a última linha das tabelas apresenta a pontuação *Win/Tie/Loss*, isto é, o número de vezes que o método proposto sPerC alcança resultados superiores as demais técnicas comparadas.

Observando os resultados apresentados na Tabela 8 (AUC), em média, sPerC (AUC = 0,749) alcança desempenho superior as técnicas PerC (AUC = 0,663), *Random Forest* (AUC = 0,741), k-NN (AUC = 0,726), MLP (AUC = 0,686), *Naïve Bayes* (AUC = 0,702) e SVM (AUC = 0,608). Já em relação a técnica Árvore de Decisão (AUC = 0,753), sPerC apresenta desempenho competitivo, entretando obtém valor médio inferir para a métrica empregada. Analisando a pontuação *Win/Tie/Loss*, o método sPerC alcança desempenho superior, no

mínimo, em 21 dos 30 conjuntos de dados quando comparada as técnicas PerC, *Random Forest*, k-NN, MLP e SVM. Em relação aos métodos *Naïve Bayes* e Árvore de Decisão esse número diminiu, com a arquitetura proposta obtendo desempenho superior em 16 e 18 conjuntos de dados, respectivamente. Analisando os resultados apresentados nas Tabelas 11 (Acurácia) e 12 (*F-Measure*), a técnica sPerC obtém resultados superiores aos algoritmos *Naïve Bayes*, SVM e PerC, alcançado o mesmo comportamento quando analisado os resultados para a métrica AUC, e apresentando comportamento oposto em relação as técnicas *Random Forest* e k-NN, quando analisado os resultados para as métricas Acurácia e *F-Measure*.

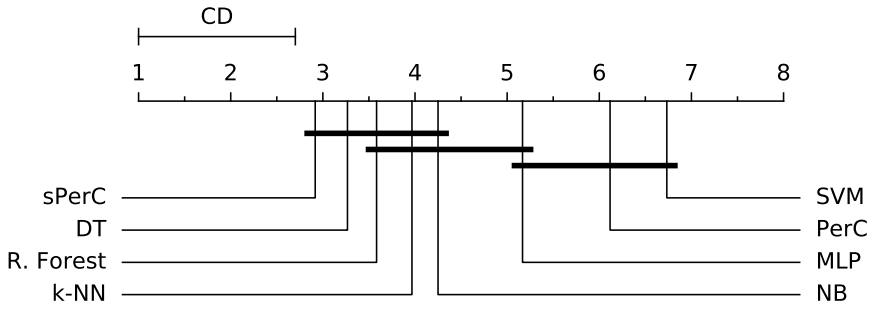
Tabela 8 – Resultados da métrica AUC no conjunto de teste para a arquitetura proposta (sPerC), e os algoritmos do estado-da-arte PerC, Random Forest (R. Forest), k-NN, MLP, Naïve Bayes (NB), SVM e Árvore de Decisão (DT), sobre as 30 bases de dados avaliadas. Os melhores resultados encontram-se em destaque.

Dataset	R. Forest	k-NN	MLP	NB	SVM	DT	PerC	sPerC
balance	0,746	0,748	**0,799**	**0,799**	0,795	0,726	0,798	0,727
banana	0,887	0,887	0,817	0,585	0,584	0,869	0,597	**0,889**
banknote	0,991	**0,999**	0,991	0,834	0,980	0,984	0,990	0,998
bupa	**0,676**	0,596	0,661	0,587	0,500	0,625	0,572	0,601
cleveland	0,577	0,570	**0,596**	0,590	0,561	0,577	0,540	0,585
cleveland-0vs4	0,639	0,682	0,673	**0,845**	0,500	0,681	0,542	0,767
ecoli-0-1-4-7vs2-3-5-6	0,788	**0,891**	0,735	0,643	0,590	0,788	0,669	0,804
ecoli1	0,840	**0,857**	0,807	0,763	0,808	0,838	0,809	0,843
ecoli2	0,837	**0,935**	0,782	0,575	0,515	0,854	0,725	0,855
ecoli3	0,720	0,766	0,530	**0,833**	0,500	0,763	0,537	0,744
glass-0-1-5vs2	0,543	0,543	0,500	0,545	0,500	**0,617**	0,500	0,546
hayes-roth	0,866	0,575	0,678	0,772	0,650	**0,876**	0,658	0,632
indian-liver-patient	0,579	0,532	0,538	**0,677**	0,500	0,574	0,500	0,533
led7digit-0-2-4-5-6-7-8-9vs1	0,892	0,650	0,881	0,856	**0,903**	0,901	0,554	0,857
new-thyroid1	0,956	0,923	0,733	0,985	0,734	0,948	0,896	**0,964**
newthyroid2	0,952	0,928	0,736	**0,985**	0,740	0,949	0,899	0,961
page-blocks0	0,918	0,850	0,852	0,704	0,704	0,911	0,906	**0,929**
poker-8-9vs5	0,500	0,500	0,500	0,500	0,500	**0,580**	0,500	0,518
poker-8-9vs6	0,526	**0,742**	0,570	0,500	0,500	0,553	0,500	0,695
shuttle-6vs2-3	0,975	0,800	0,650	0,999	0,500	**1,000**	0,950	0,991
transfusion	0,583	**0,612**	0,560	0,558	0,500	0,578	0,571	0,604
vertebral2c	0,786	0,751	0,749	**0,804**	0,499	0,776	0,559	0,771
vowel	0,959	0,966	0,857	0,815	0,691	0,897	0,952	**0,971**
winequality-red-4	0,505	0,508	0,500	0,532	0,500	**0,537**	0,500	0,520
winequality-red-8vs6-7	**0,549**	0,500	0,500	0,525	0,500	0,548	0,500	**0,549**
winequality-white-3-9vs5	0,526	0,520	0,500	**0,661**	0,500	0,612	0,500	0,556
wisconsin	0,963	**0,971**	0,964	0,966	0,969	0,942	0,946	0,954
yeast-0-5-6-7-9vs4	0,625	0,642	0,600	0,498	0,500	0,682	0,534	**0,743**
yeast-1-4-5-8vs7	0,503	0,498	0,500	0,542	0,500	**0,556**	0,490	0,500
yeast-2vs4	0,827	0,835	0,811	0,581	0,515	0,845	0,691	**0,850**
Média	0,741	0,726	0,686	0,702	0,608	0,753	0,663	0,749
Wilcoxon-p	0,107	0,053	0,001	0,209	$3{,}617 \times 10^{-5}$	0,926	$2{,}594 \times 10^{-5}$	n/a
Win/Tie/Loss	21/1/8	21/0/9	22/1/7	16/0/14	25/1/4	18/0/12	28/0/2	n/a

Após as explanações mencionadas anteriormente, nenhuma conclusão a cerca dos resultados encontrados pode ser realizada sem empregar um teste de hipótese não paramétrico. Para tal, os testes de Friedman e Wilcoxon (ambos com nível de significância $\alpha = 5\%$) foram empregados para verificar se existe diferença significativa para os resultados alcançados. Para o teste de Wilcoxon (penúltima linha da Tabela 8), os resultados encontrados demonstram que a arquitetura proposta alcança resultados estatisticamente superiores em relação aos algoritmos PerC, MLP e SVM (*p-values* $<$ 0,05). Já em relação as técnicas *Random Forest*, k-NN, *Naïve Bayes* e Árvore de Decisão, não existem evidências estatisticas para recusar a hipótese nula, ou seja, não podemos afirmar que a arquitetura proposta possui desempenho médio superior em relação as técnicas comparadas. Para o teste de Friedman (*p-value* $= 3{,}239 \times 10^{-12}$), o resultado indica que existe diferença significativa para as amostras analisadas. Com base nesse indício, o *post-hoc* teste de Bonferroni-Dunn foi empregado.

A Figura 15 apresenta o resultado do teste de Bonferroni-Dunn, para a métrica AUC, cujo menor *ranking* apresenta um melhor desempenho entre os algoritmos, e técnicas conectadas (CD $=$ 1,701) não possuem diferença estatísticas significativas. O Apêndice B exibe os resultados do teste de Bonferroni-Dunn para as métricas Acurácia (Figura 17) e *F-Measure* (Figura 18).

Figura 15 – Diagrama CD do *post-hoc* de Bonferroni-Dunn para os algoritmos do estado da arte sobre as 30 bases de dados, sendo CD $=$ 1,701.



Como ilustrado na Figura 15, a técnica sPerC exibe menor *ranking* (2,917) entre os algoritmos avaliados, apresentando desempenho estatisticamente semelhantes as técnicas Árvore de Decisão (3,267), Random Forest (3,583), k-NN (3,967) e Naïve Bayes (4,250). Já em relação aos métodos MLP (5,167), PerC (6,117) e SVM (6,733) a arquitetura proposta apresenta resultados estatisticamente superior, com a técnica SVM apresentando maior *ranking*.

Analisando as Figuras 17 e 18, nota-se que a técnica proposta sPerC obtém desempenho estatisticamente semelhantes, para ambas as métricas, em relação as técnicas *Random Forest*, k-NN, Árvore de Decisão e MLP, mesmo apresentando desempenho inferiores as técnicas Random Forest e k-NN. Para a métrica Acurácia (Figura 17), a técnica sPerC obtém resultados estatisticamente superiores apenas em relação ao algoritmo *Naïve Bayes*, e apresentando *ranking* inferior as técnicas PerC, MLP, k-NN e *Random Forest*. Já para a medida *F-Measure*, a técnica sPerC alcança desempenhos estatisticamente superiores em relação aos algoritmos SVM e PerC (versão original), e *ranking* inferior as técnicas k-NN, Árvore de Decisão e *Random Forest*.

Assim, podemos concluir que a aplicação da arquitetura proposta *subconcept PerC* (sPerC) apresenta desempenho superior em uma quantidade maior de cenários (conjuntos de dados) avaliados para a métria AUC, além de alcançar um resultado médio competitivo, quando comparada aos algoritmos do estado-da-arte. Esse comportamento não é refletido para as métricas Acurácia e *F-Measure*, com a técncia sPerC obtendo desempenho inferiores em um número maior de conjuntos de dados quando comparado as técnicas *Random Forest* e *k-NN*, mesmo apresentando desempenho estatisticamente competitivos.

5.5 ANÁLISE DE EFICIÊNCIA - SPERC *VS* PERC

Nessa seção avaliou-se as características presentes nos bancos de dados (definidos na Seção 4.1), extraídas a partir das medidas de complexidade (Seção 4.5), analisando quais propriedades justificam a eficiência dos algoritmos sPerC e PerC (versão original). O propósito desta análise é responder a seguinte pergunta: **Em quais circunstâncias devemos utilizar a técnica sPerC em relação à técnica PerC?**. O objetivo é descrever, a partir de regras, quais os cenários vantajosos para a utilização das técnicas sPerC e PerC.

Para responder ao questionamento levantado, um *meta-learning dataset* T foi construído. A base de dados T é composta por 30 exemplos, onde cada amostra representa um banco de dados definido na Seção 4.1. Os exemplos do *dataset* T são formados a partir das 22 medidas de complexidade definidas (Seção 4.5), onde cada métrica representa uma *meta-feature* do *dataset* T produzido.

Para rotular as amostras da base de dados T, os resultados para as técnicas sPerC e PerC, apresentados na Seção 5.4 (Tabela 8), foram utilizados. Nesse contexto, a técnica que obteve desempenho superior para cada base de dados avaliada nos experimentos foi definida como

rótulo da amostra em questão. A Tabela 9 resume os resultados encontrados na Tabela 8 para as técnicas PerC e sPerC (método proposto).

Tabela 9 – Resultados da métrica AUC no conjunto de teste para a técnica proposta sPerC e o algoritmo do estado-da-arte PerC, sobre as 30 bases de dados avaliadas. Os melhores resultados encontram-se em destaque.

Dataset	PerC	sPerC	Dataset	PerC	sPerC
balance	**0,798**	0,727	newthyroid2	0,899	**0,961**
banana	0,597	**0,889**	page-blocks0	0,906	**0,929**
banknote	0,990	**0,998**	poker-8-9vs5	0,500	**0,518**
bupa	0,572	**0,601**	poker-8-9vs6	0,500	**0,695**
cleveland	0,540	**0,585**	shuttle-6vs2-3	0,950	**0,991**
cleveland-0vs4	0,542	**0,767**	transfusion	0,571	**0,604**
ecoli-0-1-4-7vs2-3-5-6	0,669	**0,804**	vertebral2c	0,559	**0,771**
ecoli1	0,809	**0,843**	vowel	0,952	**0,971**
ecoli2	0,725	**0,855**	winequality-red-4	0,500	**0,520**
ecoli3	0,537	**0,744**	winequality-red-8vs6-7	0,500	**0,549**
glass-0-1-5vs2	0,500	**0,546**	winequality-white-3-9vs5	0,500	**0,556**
hayes-roth	**0,658**	0,632	wisconsin	0,946	**0,954**
indian-liver-patient	0,500	**0,533**	yeast-0-5-6-7-9vs4	0,534	**0,743**
led7digit-0-2-4-5-6-7-8-9vs1	0,554	**0,857**	yeast-1-4-5-8vs7	0,490	**0,500**
new-thyroid1	0,896	**0,964**	yeast-2vs4	0,691	**0,850**

Por exemplo, para a base de dados *balance*, a técnica PerC obteve desempenho superior em comparação à técnica sPerC. Portanto, a amostra que representa o *dataset balance* será rotulada com a classe "PerC". Por outro lado, a técnica sPerC alcançou desempenho superior à técnica PerC para a base de dados *banknote*. Dessa forma, a amostra correspondente a base *banknote* será rotulada com a classe "sPerC".

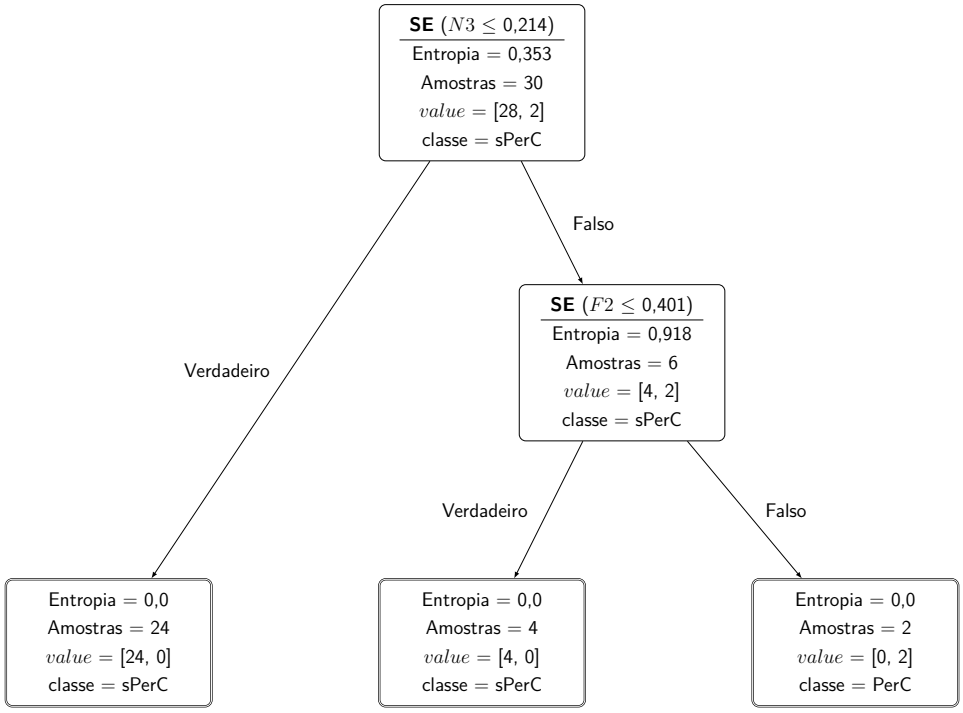
Uma vez rotulada todas as amostras do *meta-learning dataset* T, o mesmo será composto por 28 exemplos pertencentes à classe "sPerC", e 2 exemplos pertencentes à classe "PerC".

Após a construção do *dataset* T, uma árvore de decisão (BREIMAN, 2017) foi utilizada como um meta-classificador (VILALTA; DRISSI, 2002), sendo treinada sobre a base de dados T produzida. O objetivo do modelo é escolher qual técnica, entre sPerC e PerC, deve ser utilizada quando uma base de dados for avaliada. A Figura 16 apresenta as regras extraídas a partir da árvore de decisão utilizada. Os nós folhas da estrutura apresentam a saída do modelo utilizado.

Observando a Figura 16, nota-se que das 22 medidas de complexidades adotadas para descrever cada banco de dados avaliados nos experimentos, a árvore de decisão empregada encontrou uma correlação entre as métricas $N3$ (*The leave-one-out error rate of the one-nearest neighbor classifier*) e $F2$ (*The overlap of the per-class bouding boxes*). A métrica

IR (definida pela medida de complexidade C2), assim como as demais medidas de complexidade adotadas, não tiveram correlação com o problema.

Figura 16 – Regras definidas por uma árvore de decisão, após o treinamento sobre o *meta-learning dataset* T produzido.



A partir do nó raiz da estrutura apresentada na Figura 16, as regras definidas pelo modelo podem ser descritas em equivalências lógicas, no formato *SE ENTÃO*, conforme exibido a seguir:

1. **SE** (*N3* $\leq$ 0,214) **ENTÃO** "sPerC"
2. **SE** (*N3* $>$ 0,214) $\wedge$ (*F2* $\leq$ 0,401) **ENTÃO** "sPerC"
3. **SE** (*N3* $>$ 0,214) $\wedge$ (*F2* $>$ 0,401) **ENTÃO** "PerC"

De acordo com as equivalências lógicas 1 e 2, o método sPerC obtém desempenhos superiores à técnica PerC em base de dados que possuem como característica poucos pontos que estão próximos de exemplos de outras classes, conforme definido pela medida de complexidade N3, apresentando baixa complexidade (equivalência lógica 1). Ou, quando as bases de dados contém diversas amostras próximas de pontos de outras classes, apresentam como característica baixa sopreposição entre as classes do problema, como definido pela medida de complexidade F2 (equivalência lógica 2).

Já a técnica PerC, segundo a equivalência lógica 3, obtém desempenho superiores à técnica sPerC em bases de dados que possuem como características amostras que estão próximas de exemplos de outras classes (medida de complexidade N3), e em cenários onde a sobreposição das classes é elevado (de acordo com a medida de complexidade F2).

Utilizando o conjunto de treinamento (*meta-learning dataset* T) como um conjunto de validação, o meta-classificador pode ser avaliado, alcançando uma taxa de acerto igual à 100%. Ou seja, o meta-classificador determina corretamente qual técnica, entre sPerC e PerC, obteve desempenho superior para cada base de dados (Seção 4.1) avaliada nos experimentos.

Diante dos resultados apresentados na Tabela 9, o valor da métrica AUC, correspondente à técnica (PerC ou sPerC) determinada pelo meta-classificador, pode ser obtido para cada um dos conjutos de dados, representando o desempenho do meta-classificador sobre os *datasets* avalidados. A Tabela 10 apresenta os resultados encontrados para o meta-classificador (Meta), além dos resultados para os algoritmos PerC (versão original) e sPerC (método proposto). Por alcançar uma taxa de acerto igual à 100% sobre o *meta-learning dataset* T (conjunto de treinamento), o meta-classificador obtém desempenho médio (AUC $=$ 0,752) superior às técnicas sPerC (AUC $=$ 0,749) e PerC (AUC $=$ 0,663).

Visando simular a predição de uma nova base de dados a partir da árvore de decisão empregada, foi realizado um experimento utilizando a abordagem *leave-one-dataset-out*, no qual 29 das 30 bases de dados definidos nos experimentos são utilizados para construir o *meta-learning dataset* T e treinar o meta-classificador, enquanto o conjunto de dados restante é utilizado para testar o modelo. O experimento foi repetido 30 vezes, de modo que cada base de dados seja utilizada para testar o meta-classificador. Média da acurácia foi computada. Como resultado, a árvore de decisão obteve uma taxa de acerto média de 86,67%, ou seja, o modelo aplicado escolheu a técnica mais apropriada (entre PerC e sPerC) em 26 das 30 bases de dados avaliadas.

Tabela 10 – Resultados da métrica AUC no conjunto de teste para a arquitetura proposta (sPerC), o algoritmo do estado-da-arte PerC, e para o meta-classificador (Meta) gerado. Os melhores resultados encontram-se em destaque.

Dataset	PerC	sPerC	Meta
balance	**0,798**	0,727	**0,798**
banana	0,597	**0,889**	**0,889**
banknote	0,990	**0,998**	**0,998**
bupa	0,572	**0,601**	**0,601**
cleveland	0,540	**0,585**	**0,585**
cleveland-0vs4	0,542	**0,767**	**0,767**
ecoli-0-1-4-7vs2-3-5-6	0,669	**0,804**	**0,804**
ecoli1	0,809	**0,843**	**0,843**
ecoli2	0,725	**0,855**	**0,855**
ecoli3	0,537	**0,744**	**0,744**
glass-0-1-5vs2	0,500	**0,546**	**0,546**
hayes-roth	**0,658**	0,632	**0,658**
indian-liver-patient	0,500	**0,533**	**0,533**
led7digit-0-2-4-5-6-7-8-9vs1	0,554	**0,857**	**0,857**
new-thyroid1	0,896	**0,964**	**0,964**
newthyroid2	0,899	**0,961**	**0,961**
page-blocks0	0,906	**0,929**	**0,929**
poker-8-9vs5	0,500	**0,518**	**0,518**
poker-8-9vs6	0,500	**0,695**	**0,695**
shuttle-6vs2-3	0,950	**0,991**	**0,991**
transfusion	0,571	**0,604**	**0,604**
vertebral2c	0,559	**0,771**	**0,771**
vowel	0,952	**0,971**	**0,971**
winequality-red-4	0,500	**0,520**	**0,520**
winequality-red-8vs6-7	0,500	**0,549**	**0,549**
winequality-white-3-9vs5	0,500	**0,556**	**0,556**
wisconsin	0,946	**0,954**	**0,954**
yeast-0-5-6-7-9vs4	0,534	**0,743**	**0,743**
yeast-1-4-5-8vs7	0,490	**0,500**	**0,500**
yeast-2vs4	0,691	**0,850**	**0,850**
Média	0,663	0,749	0,752

6 CONSIDERAÇÕES FINAIS

A multimodalidade é uma característica que aumenta o grau de complexidade dos conjuntos de dados, devido à presença de diferentes grupos (subconceitos), dispersos no domínio do problema, para uma mesma classe. Essa propriedade afeta a generalização e consequentemente, o desempenho do classificador PerC (do inglês, *Perturbation-based Classifier*), uma vez que os vetores médios, $\hat{\mu}$, e matrizes de covariâncias, $\hat{\Sigma}$, das classes existentes, utilizados para rotular novas amostras, possuem um baixo nível de discernimento. Os subconceitos existentes em cada uma das classes representam particularidades do problema, sendo formados por aglomerados de dados que compartilham características em comum.

Neste trabalho apresentou-se uma nova abordagem de classificação para o modelo PerC, denominada sPerC (do inglês, ***s**ubconcept* PerC), no qual foi utilizado a técnica *K-Means* para estimar os subconceitos (*clusters*) existentes nas classes, visando aumentar o grau de discernimento dos dados. Fornecendo os *clusters* como entrada para o classificador PerC, o objetivo é incrementar o poder de generalização do modelo, assim como seu desempenho. A arquitetura proposta atua sobre as etapas de treinamento e teste do classificador PerC. Ao todo, 30 conjuntos de dados, obtidos a partir dos repositórios *Knowledge Extraction based on Evolutionary Learning* (KEEL) e *UCI Machine Learning*, foram utilizados nos experimentos através de uma validação cruzada estratificada. Como métrica de avaliação foi utilizado a medida AUC (do inglês, *Area Under Curve*).

Durante a etapa de treinamento, a arquitetura proposta utiliza o algoritmo *K-Means* como uma técnica de pré-processamento de dados, buscando particionar os dados de acordo com suas respectivas distribuições. Para estimar o valor do parâmetro K, da técnica *K-Means*, os *clustering validation* índices *Calinski–Harabasz*, *Davies–Bouldin*, *Gap Statistic* e *Silhouette* foram avaliados. Além disso, a presente dissertação propôs uma estratégia, chamada de Combinação, onde a cada iteração seleciona o *clustering validation* índice com melhor desempenho sobre o conjunto de validação. Os resultados alcançados mostraram uma eficiência da estratégia definida, em comparação aos índices do estado-da-arte. Os resultados possibilitaram observar uma maior influência dos índices *Gap Statistic* e *Silhouette* sobre a estratégia proposta.

Ainda na etapa de treinamento, outro ponto investigado foi a influência dos *small clusters* durante o treinamento do algoritmo PerC. Neste ponto, foi avaliada a remoção de tais partições

durante o treinamento do modelo. Sendo necessário definir a *priori* o valor do parâmetro τ, o trabalho de (HE; XU; DENG, 2003) foi considerado para categorizar as partições geradas em *small* (*large*) *clusters*. O parâmetro τ foi variado no intervalo 10%, 20%, ..., 100%. Os resultados apresentados demonstram que a remoção total dos *small clusters* degrada a performance do classificador PerC. Dessa forma, para a presente proposta, foi construída uma abordagem para determinar, iterativamente, o valor do parâmetro τ para cada base de dados. Neste cenário, a remoção de alguns *small clusters* incrementou o desempenho da técnica PerC, em 21 dos 30 conjuntos de dados avaliados.

Na etapa de teste, a arquitetura proposta combina as perturbações geradas pelo classificador PerC para rotular novas amostras. Neste trabalho, além da regra do Mínimo, proposta na versão original do modelo PerC, foram definidas as regras de Média, Mediana e *Cluster* Mais Próximo. Os resultados apresentados comprovam que a aplicabiliade da regra *Cluster* Mais Próximo supera estatisticamente as demais regras avaliadas, sugerindo sua utilização como modo para agregar as perturbações computadas pela técnicas PerC.

O desempenho da proposta sPerC foi comparada com técnicas do estado-da-arte. Além da versão original do PerC, foram utilizados nos experimentos as técnicas Árvore de Decisão, k-NN, MLP, *Naïve Bayes*, *Random Forest* e SVM. Os resultados demonstraram desempenho competitivos entre o método sPerC e as técnicas *Random Forest* e Árvore de Decisão, com o sPerC obtendo um desempenho médio inferior em relação a técnica Árvore de Decisão. Por meio dos testes estatísticos de *Wilcoxon* e *Friedman*, o método proposto teve seu desempenho comprovado, sendo significativamente superior aos algoritmos PerC, MLP e SVM, tendo o método sPerC menor *ranking* entre os métodos avaliados.

Por fim, foi analisado quais características existentes nos conjuntos de dados, avaliados durante os experimentos, justificam a eficiência da técnica sPerC em relação à técnica PerC. Tais características foram extraídas a partir de 22 medidas de complexidades e utilizadas para construir um *meta-learning dataset*, onde cada medida de complexidade representa uma *meta-feature* da base produzida. Ao final, uma árvore de decisão foi treinada sobre o *dataset* criado. Os resultados demonstram uma maior correlação entre as medidas de complexidade $N3$ e $F2$, com a técnica sPerC alcançando desempenho superiores em cenários que apresentam uma quantidade reduzidas de pontos próximos de outras classes ou baixa sobreposição entre os dados das classes existentes.

6.1 LIMITAÇÕES

Durante o desenvolvimento do método sPerC, algumas limitações foram observadas:

- A não utilização de técnicas de agrupamento de dados mais recentes (ESTER et al., 1996; FREY; DUECK, 2007) que, por sua vez, determinam automaticamente o número de *clusters*.
- A não definição de um parâmetro para limitar o número máximo de *clusters* gerados pelo sPerC, impedindo fornecer um conhecimento prévio do problema durante sua execução.
- Avaliação muito abrangente do parâmetro τ, desconsiderando intervalos mais restritos, e possivelmente, mais relevantes para a execução do método proposto.
- Não realizar um processo de *Grid Search* para otimizar os hiperparâmetros dos algoritmos do estado-da-arte.

6.2 TRABALHOS FUTUROS

- Nesta proposta o método *K-Means* foi empregado para realizar o agrupamentos dos dados de treinamento. Esse modelo é sensível a configuração inicial dos *clusters*, sendo passivo de deficiências. Uma alternativa é verificar a aplicabilidade de *Ensemble Clustering* (EC) (GODER; FILKOV, 2008), para determinar melhores resultados em termos de robustez e qualidade;
- Explorar alternativas para tratar a presença de *small clusters*, ao invés de removê-los do treinamento;
- As bases de dados utilizadas nos experimentos possuem como característica o desbalanceamento dos dados. Essa assimetria na distribuição acaba tornando os algoritmos de classificação menos efetivos (ZHANG et al., 2017). A partir desse cenário, pode-se investigar a aplicabilidade das técnicas de *resampling* para atacar o problema do desbalanceamento de dados, precedendo a execução da técnica de agrupamento.

REFERÊNCIAS

ADE, R.; DESHMUKH, P. Methods for incremental learning: a survey. *International Journal of Data Mining & Knowledge Management Process*, Academy & Industry Research Collaboration Center (AIRCC), v. 3, n. 4, p. 119, 2013.

AHA, D. W.; KIBLER, D.; ALBERT, M. K. Instance-based learning algorithms. *Machine Learning*, Springer, v. 6, n. 1, p. 37–66, 1991.

ALCALÁ-FDEZ, J.; FERNÁNDEZ, A.; LUENGO, J.; DERRAC, J.; GARCÍA, S.; SÁNCHEZ, L.; HERRERA, F. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, Citeseer, v. 17, 2011.

ALI, S.; SMITH, K. A. On learning algorithm selection for classification. *Applied Soft Computing*, Elsevier, v. 6, n. 2, p. 119–138, 2006.

ALPAYDIN, E. *Introduction to machine learning*. [S.l.]: MIT press, 2014.

ARAÚJO, E. L. *Um classificador baseado em perturbações*. Tese (Doutorado) — Universidade Federal de Pernambuco, 4 2017.

ARAÚJO, E. L.; CAVALCANTI, G. D.; REN, T. I. Perturbation-based classifier. Soft Computing, 2020.

ARBELAITZ, O.; GURRUTXAGA, I.; MUGUERZA, J.; PÉREZ, J. M.; PERONA, I. An extensive comparative study of cluster validity indices. *Pattern Recognition*, Elsevier, v. 46, n. 1, p. 243–256, 2013.

ARTHUR, D.; VASSILVITSKII, S. k-means++: The advantages of careful seeding. In: SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. [S.l.], 2007. p. 1027–1035.

AWOYEMI, J. O.; ADETUNMBI, A. O.; OLUWADARE, S. A. Credit card fraud detection using machine learning techniques: A comparative analysis. In: *2017 International Conference on Computing Networking and Informatics (ICCNI)*. [S.l.: s.n.], 2017. p. 1–9.

BASU, M.; HO, T. K. *Data complexity in pattern recognition*. [S.l.]: Springer Science & Business Media, 2006.

BEALE, R.; JACKSON, T. *Neural Computing-an introduction*. [S.l.]: CRC Press, 1990.

BERARD, A.; PIETQUIN, O.; SERVAN, C.; BESACIER, L. *Listen and Translate: A Proof of Concept for End-to-End Speech-to-Text Translation*. 2016.

BERRY, M. W.; MOHAMED, A.; YAP, B. W. *Supervised and Unsupervised Learning for Data Science*. [S.l.]: Springer, 2019.

BISHOP, C. M. *Pattern recognition and machine learning*. [S.l.]: springer, 2006.

BRADLEY, A. P. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, Elsevier, v. 30, n. 7, p. 1145–1159, 1997.

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

BREIMAN, L. *Classification and regression trees*. [S.l.]: Routledge, 2017.

BRUN, M.; SIMA, C.; HUA, J.; LOWEY, J.; CARROLL, B.; SUH, E.; DOUGHERTY, E. R. Model-based evaluation of clustering validation measures. *Pattern Recognition*, Elsevier, v. 40, n. 3, p. 807–824, 2007.

BZDOK, D.; KRZYWINSKI, M.; ALTMAN, N. *Points of significance: Machine learning: supervised methods*. [S.l.]: Nature Publishing Group, 2018.

CALIŃSKI, T.; HARABASZ, J. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, Taylor & Francis, v. 3, n. 1, p. 1–27, 1974.

CARLINI, N.; WAGNER, D. Audio adversarial examples: Targeted attacks on speech-to-text. In: *2018 IEEE Security and Privacy Workshops (SPW)*. [S.l.: s.n.], 2018. p. 1–7.

CHENG, B.; TITTERINGTON, D. M. Neural networks: A review from a statistical perspective. *Statistical science*, JSTOR, p. 2–30, 1994.

DAVIES, D. L.; BOULDIN, D. W. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, n. 2, p. 224–227, 1979.

DOMINGOS, P.; PAZZANI, M. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, Springer, v. 29, n. 2-3, p. 103–130, 1997.

DRINEAS, P.; FRIEZE, A.; KANNAN, R.; VEMPALA, S.; VINAY, V. Clustering large graphs via the singular value decomposition. *Machine Learning*, Springer, v. 56, n. 1-3, p. 9–33, 2004.

DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml>.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern classification*. [S.l.]: John Wiley & Sons, 2012.

DUNN, J. C. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Taylor & Francis, 1973.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. [S.l.]: AAAI Press, 1996. (KDD'96), p. 226–231.

FLORES, M. J.; GÁMEZ, J. A.; MARTÍNEZ, A. M. Domains of competence of the semi-naive bayesian network classifiers. *Information Sciences*, Elsevier, v. 260, p. 120–148, 2014.

FONSÊCA, C.; MACIEL, A. M. Anomaly detection in the registry of the secondary energy distribution network (s). In: *SEKE*. [S.l.: s.n.], 2019. p. 766–777.

FORGY, E. W. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics*, v. 21, p. 768–769, 1965.

FREY, B.; DUECK, D. Clustering by passing messages between data points. *Science*, v. 315, p. 972 – 976, 2007.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937.

FUKUNAGA, K. *Introduction to statistical pattern recognition*. [S.l.]: Elsevier, 2013.

GENTLEMAN, R.; CAREY, V. J. Unsupervised machine learning. In: *Bioconductor case studies*. [S.l.]: Springer, 2008. p. 137–157.

GODER, A.; FILKOV, V. Consensus clustering algorithms: Comparison and refinement. In: SIAM. *2008 Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX)*. [S.l.], 2008. p. 109–117.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. On clustering validation techniques. *Journal of Intelligent Information Systems*, Springer, v. 17, n. 2-3, p. 107–145, 2001.

HÄMÄLÄINEN, J.; JAUHIAINEN, S.; KÄRKKÄINEN, T. Comparison of internal clustering validation indices for prototype-based clustering. *Algorithms*, Multidisciplinary Digital Publishing Institute, v. 10, n. 3, p. 105, 2017.

HAMERLY, G.; ELKAN, C. Learning the k in k-means. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2004. p. 281–288.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The elements of statistical learning: data mining, inference, and prediction*. [S.l.]: Springer Science & Business Media, 2009.

HE, Z.; XU, X.; DENG, S. Discovering cluster-based local outliers. *Pattern Recognition Letters*, Elsevier, v. 24, n. 9-10, p. 1641–1650, 2003.

HO, T. K.; BASU, M. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 24, n. 3, p. 289–300, 2002.

HOLTE, R. C.; ACKER, L. E.; PORTER, B. W. Concept learning and the problem of small disjuncts. In: *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989. (IJCAI'89), p. 813–818.

HUBERT, L.; ARABIE, P. Comparing partitions. *Journal of Classification*, Springer, v. 2, n. 1, p. 193–218, 1985.

JACCARD, P. Nouvelles recherches sur la distribution florale. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, v. 44, p. 223–70, 01 1908.

JAIN, A.; DUIN, R.; MAO, J. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 22, p. 4–37, 01 2000.

JAIN, A. K. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, Elsevier, v. 31, n. 8, p. 651–666, 2010.

JAIN, A. K.; DUBES, R. C. *Algorithms for clustering data*. [S.l.]: Prentice-Hall, Inc., 1988.

JOHNSON, R. A.; WICHERN, D. W. et al. *Applied multivariate statistical analysis*. [S.l.]: Prentice hall Upper Saddle River, NJ, 2002. v. 5.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. *Science*, American Association for the Advancement of Science, v. 349, n. 6245, p. 255–260, 2015.

JR, E. R. S.; CAVALCANTI, G. D.; REN, T. I. Class-wise feature extraction technique for multimodal data. *Neurocomputing*, Elsevier, v. 214, p. 1001–1010, 2016.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, v. 4, p. 237–285, 1996.

KENNEDY, J. Swarm intelligence. In: *Handbook of nature-inspired and innovative computing*. [S.l.]: Springer, 2006. p. 187–219.

KIVINEN, J.; SMOLA, A. J.; WILLIAMSON, R. C. Online learning with kernels. *IEEE Transactions on Signal Processing*, IEEE, v. 52, n. 8, p. 2165–2176, 2004.

KODINARIYA, T.; MAKWANA, P. Review on determining of cluster in k-means clustering. *International Journal of Advance Research in Computer Science and Management Studies*, v. 1, p. 90–95, 01 2013.

KOTSIANTIS, S. B.; ZAHARAKIS, I.; PINTELAS, P. Supervised machine learning: A review of classification techniques. *Emerging Artificial Intelligence Applications in Computer Engineering*, Amsterdam, v. 160, p. 3–24, 2007.

KUMAR, P. M.; GANDHI, U. D. A novel three-tier internet of things architecture with machine learning algorithm for early detection of heart diseases. *Computers Electrical Engineering*, v. 65, p. 222 – 235, 2018. ISSN 0045-7906. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0045790617328410>.

LÁZARO, M.; HERRERA, F.; FIGUEIRAS-VIDAL, A. R. Ensembles of cost-diverse bayesian neural learners for imbalanced binary classification. *Information Sciences*, Elsevier, v. 520, p. 31–45, 2020.

LIU, Y.; LI, Z.; XIONG, H.; GAO, X.; WU, J. Understanding of internal clustering validation measures. In: . [S.l.: s.n.], 2010. p. 911–916.

LIU, Z.; JAPKOWICZ, N.; WANG, R. Subconcept based one class classification method with cluster updating. In: *Proceedings of the 2020 12th International Conference on Machine Learning and Computing*. New York, NY, USA: Association for Computing Machinery, 2020. (ICMLC 2020), p. 23–28. ISBN 9781450376426. Disponível em: <https://doi.org/10.1145/3383972.3384016>.

LIU, Z.; JAPKOWICZ, N.; WANG, R.; LIU, L. A sub-concept-based feature selection method for one-class classification. *Soft Computing*, Springer, p. 1–16, 2020.

LLOYD, S. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, IEEE, v. 28, n. 2, p. 129–137, 1982.

LÓPEZ, V.; FERNáNDEZ, A.; GARCíA, S.; PALADE, V.; HERRERA, F. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, v. 250, p. 113 – 141, 2013. ISSN 0020-0255. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025513005124>.

LORENA, A. C.; GARCIA, L. P.; LEHMANN, J.; SOUTO, M. C.; HO, T. K. How complex is your classification problem? a survey on measuring classification complexity. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 52, n. 5, p. 1–34, 2019.

LORENA, A. C.; MACIEL, A. I.; MIRANDA, P. B. de; COSTA, I. G.; PRUDÊNCIO, R. B. Data complexity meta-features for regression problems. *Machine Learning*, Springer, v. 107, n. 1, p. 209–246, 2018.

LUENGO, J.; HERRERA, F. An automatic extraction method of the domains of competence for learning classifiers using data complexity measures. *Knowledge and Information Systems*, Springer, v. 42, n. 1, p. 147–180, 2015.

LÜTZ, A.; RODNER, E.; DENZLER, J. Efficient multi-class incremental learning using gaussian processes. In: CITESEER. *Open German-Russian Workshop on Pattern Recognition and Image Understanding*. [S.l.], 2011. p. 182–185.

LÜTZ, A.; RODNER, E.; DENZLER, J. I want to know more—efficient multi-class incremental learning using gaussian processes. *Pattern Recognition and Image Analysis*, Springer, v. 23, n. 3, p. 402–407, 2013.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*. [S.l.: s.n.], 1967. p. 281–297.

MAO, J.; JAIN, A. K. A self-organizing network for hyperellipsoidal clustering (hec). *IEEE Transactions on Neural Networks*, IEEE, v. 7, n. 1, p. 16–29, 1996.

MITCHELL, T. M. et al. *Machine learning*. [S.l.]: McGraw-Hill, Inc., New York, NY, USA, 1997.

MUÑOZ, M. A.; VILLANOVA, L.; BAATAR, D.; SMITH-MILES, K. Instance spaces for machine learning classification. *Machine Learning*, Springer, v. 107, n. 1, p. 109–147, 2018.

NETO, E. F. da S.; FEITOSA, A. R.; CAVALCANTI, G. D.; SILVA-FILHO, A. G. Detecting anomalies in the engine coolant sensor using one-class classifiers. In: IEEE. *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. [S.l.], 2019. p. 1–5.

NG, A. Clustering with the k-means algorithm. *Machine Learning*, 2012.

NILASHI, M.; IBRAHIM, O. bin; AHMADI, H.; SHAHMORADI, L. An analytical method for diseases prediction using machine learning techniques. *Computers Chemical Engineering*, v. 106, p. 212 – 223, 2017. ISSN 0098-1354. ESCAPE-26. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0098135417302570>.

OLIVEIRA, D. V.; CAVALCANTI, G. D.; SABOURIN, R. Online pruning of base classifiers for dynamic ensemble selection. *Pattern Recognition*, Elsevier, v. 72, p. 44–58, 2017.

PATIL, C.; BAIDARI, I. Estimating the optimal number of clusters k in a dataset using data depth. *Data Science and Engineering*, Springer, v. 4, n. 2, p. 132–140, 2019.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V. et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, v. 12, n. Oct, p. 2825–2830, 2011.

QUINLAN, J. R. Induction of decision trees. *Machine Learning*, Springer, v. 1, n. 1, p. 81–106, 1986.

RANDHAWA, K.; LOO, C. K.; SEERA, M.; LIM, C. P.; NANDI, A. K. Credit card fraud detection using adaboost and majority voting. *IEEE Access*, v. 6, p. 14277–14284, 2018.

RENDÓN, E.; ABUNDEZ, I.; ARIZMENDI, A.; QUIROZ, E. M. Internal versus external cluster validation indexes. *International Journal of Computers and Communications*, v. 5, n. 1, p. 27–34, 2011.

ROKACH, L.; MAIMON, O. Clustering methods. In: *Data mining and knowledge discovery handbook*. [S.l.]: Springer, 2005. p. 321–352.

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, Elsevier, v. 20, p. 53–65, 1987.

SAITTA, S.; RAPHAEL, B.; SMITH, I. F. A bounded index for cluster validity. In: SPRINGER. *International Workshop on Machine Learning and Data Mining in Pattern ecognition*. [S.l.], 2007. p. 174–187.

SANTOS, T.; FERREIRA, V.; FORTES, M.; GONZALES, V.; ROSA, A. Machine learning applied to shutdown risk estimation in transmission system maintenance interventions. In: IEEE. *2019 IEEE PES Innovative Smart Grid Technologies Conference-Latin America (ISGT Latin America)*. [S.l.], 2019. p. 1–6.

SCHOLKOPF, B.; SMOLA, A. J. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. [S.l.]: MIT press, 2001.

SHARMA, S.; SOMAYAJI, A.; JAPKOWICZ, N. Learning over subconcepts: Strategies for 1-class classification. *Computational Intelligence*, Wiley Online Library, v. 34, n. 2, p. 440–467, 2018.

SOWA, J. F. Semantic networks. Citeseer, 1987.

SUGIYAMA, M. Local fisher discriminant analysis for supervised dimensionality reduction. In: *Proceedings of the International Conference on Machine Learning*. [S.l.: s.n.], 2006. p. 905–912.

SUGIYAMA, M. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *Journal of Machine Learning Research*, v. 8, n. May, p. 1027–1061, 2007.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN 0262039249.

TAHERI, M.; MOSLEHI, Z.; MIRZAEI, A.; SAFAYANI, M. A self-adaptive local metric learning method for classification. *Pattern Recognition*, Elsevier, v. 96, p. 106994, 2019.

TAIGMAN, Y.; YANG, M.; RANZATO, M.; WOLF, L. Deepface: Closing the gap to human-level performance in face verification. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2014.

TIBSHIRANI, R.; WALTHER, G.; HASTIE, T. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Wiley Online Library, v. 63, n. 2, p. 411–423, 2001.

TRYON, R. C. Cluster analysis. edwards brothers. *Ann Arbor, Michigan*, p. 122, 1939.

ÜNLÜ, R.; XANTHOPOULOS, P. Estimating the number of clusters in a dataset via consensus clustering. *Expert Systems with Applications*, Elsevier, v. 125, p. 33–39, 2019.

VALENTINI, G. An experimental bias-variance analysis of svm ensembles based on resampling techniques. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 35, n. 6, p. 1252–1271, 2005.

VILALTA, R.; DRISSI, Y. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, Springer, v. 18, n. 2, p. 77–95, 2002.

WANG, H. Nearest neighbors by neighborhood counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 28, n. 6, p. 942–953, 2006.

WANG, J.; NESKOVIC, P.; COOPER, L. N. Improving nearest neighbor rule with a simple adaptive distance measure. *Pattern Recognition Letters*, Elsevier, v. 28, n. 2, p. 207–213, 2007.

WANG, T.; TIAN, S.; HUANG, H.; DENG, D. Learning by local kernel polarization. *Neurocomputing*, Elsevier, v. 72, n. 13-15, p. 3077–3084, 2009.

WEISS, G. M. Learning with rare cases and small disjuncts. In: *Machine Learning Proceedings 1995*. [S.l.]: Elsevier, 1995. p. 558–565.

WEISS, G. M. The impact of small disjuncts on classifier learning. In: SPRINGER. *Data Mining*. [S.l.], 2010. p. 193–226.

WEISS, G. M.; HIRSH, H. A quantitative study of small disjuncts. In: *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. [S.l.]: AAAI Press, 2000. p. 665–670. ISBN 0262511126.

WHITLEY, D. A genetic algorithm tutorial. *Statistics and computing*, Springer, v. 4, n. 2, p. 65–85, 1994.

WILCOXON, F. Individual comparisons by ranking methods. In: *Breakthroughs in statistics*. [S.l.]: Springer, 1992. p. 196–202.

ZHANG, C.; LIU, C.; ZHANG, X.; ALMPANIDIS, G. An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, v. 82, p. 128 – 150, 2017. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417417302397>.

APÊNDICE A – RESULTADO PARA AS MÉTRICAS ACURÁCIA E *F-MEASURE*

Tabela 11 – Resultados da métrica Acurácia no conjunto de teste, para a arquitetura proposta (sPerC), e os algoritmos do estado-da-arte PerC (versão original), Random Forest (R. Forest), k-NN, MLP, Naïve Bayes (NB), SVM e Árvore de Decisão (DT), sobre as 30 bases de dados avaliadas. Os melhores resultados encontram-se em destaque

Dataset	R. Forest	k-NN	MLP	NB	SVM	DT	PerC	sPerC
balance	81,20	81,57	**90,69**	90,61	90,05	77,40	90,54	86,85
banana	89,03	88,89	82,70	61,35	62,15	86,98	62,57	**89,36**
banknote	99,16	**99,85**	99,00	83,92	97,75	98,38	98,89	99,78
bupa	68,20	61,29	**69,26**	55,89	57,98	63,25	63,19	61,96
cleveland	56,34	55,69	**58,52**	53,07	57,90	49,02	55,98	49,33
cleveland-0vs4	93,52	**94,71**	94,31	89,98	92,55	90,20	93,12	92,08
ecoli-0-1-4-7vs2-3-5-6	95,67	**97,38**	94,84	93,26	92,37	93,31	93,73	91,44
ecoli1	89,20	**91,16**	88,15	65,20	87,94	88,63	89,26	87,86
ecoli2	93,92	**96,27**	91,53	28,61	84,99	91,69	89,44	91,90
ecoli3	92,41	**92,45**	89,76	75,65	89,61	91,73	90,29	91,21
glass-0-1-5vs2	**90,27**	89,82	90,22	43,49	90,22	86,49	90,22	86,70
hayes-roth	80,93	47,29	56,53	67,25	53,29	**82,14**	51,25	52,31
indian-liver-patient	70,75	64,37	71,11	55,71	**71,51**	64,82	71,47	71,22
led7digit-0-2-4-5-6-7-8-9vs1	96,46	93,40	96,55	89,49	**96,61**	96,57	91,82	92,03
new-thyroid1	**98,21**	96,89	91,33	97,48	91,33	96,71	96,62	98,11
newthyroid2	**98,28**	97,13	91,38	97,43	91,52	96,66	96,71	**98,28**
page-blocks0	**97,47**	95,95	96,12	90,23	93,15	96,67	92,50	94,57
poker-8-9vs5	98,78	**98,80**	**98,80**	**98,80**	**98,80**	97,61	**98,80**	98,68
poker-8-9vs6	98,41	**99,13**	98,55	98,32	98,32	97,36	98,32	96,72
shuttle-6vs2-3	99,78	98,26	96,96	99,74	95,65	**100,00**	99,56	98,35
transfusion	73,98	76,19	77,61	75,00	76,20	71,95	**77,89**	77,09
vertebral2c	**82,87**	78,61	79,29	77,97	67,45	80,64	71,48	81,58
vowel	92,59	93,89	74,00	66,39	43,81	81,30	91,25	**94,64**
winequality-red-4	96,52	96,64	96,62	93,63	**96,69**	93,99	**96,69**	96,08
winequality-red-8vs6-7	**97,98**	97,90	97,90	96,59	97,90	96,29	97,90	97,90
winequality-white-3-9vs5	98,19	98,26	**98,31**	96,80	**98,31**	97,64	**98,31**	97,77
wisconsin	96,57	**97,28**	96,79	96,24	97,04	94,94	92,96	95,18
yeast-0-5-6-7-9vs4	91,50	91,70	**91,90**	12,12	90,34	87,61	90,08	90,08
yeast-1-4-5-8vs7	95,60	95,38	**95,67**	12,28	**95,67**	92,06	93,84	94,65
yeast-2vs4	95,57	**96,29**	95,86	28,59	90,37	94,34	93,37	94,89
Média	90,31	88,75	88,34	73,04	84,92	87,88	87,27	88,29
Wilcoxon-p	0,004	0,183	0,198	0,001	0,319	0,075	0,632	n/a
Win/Tie/Loss	6/1/23	11/1/18	10/1/19	22/0/8	13/1/16	21/0/9	14/2/14	n/a

Tabela 12 – Resultados da métrica *F-Measure* no conjunto de teste, para a arquitetura proposta (sPerC), e os algoritmos do estado-da-arte PerC (versão original), Random Forest (R. Forest), k-NN, MLP, Naïve Bayes (NB), SVM e Árvore de Decisão (DT), sobre as 30 bases de dados avaliadas. Os melhores resultados encontram-se em destaque

Dataset	R. Forest	k-NN	MLP	NB	SVM	DT	PerC	sPerC
balance	0,588	0,589	**0,630**	0,629	0,625	0,573	0,628	0,601
banana	0,889	0,887	0,820	0,565	0,537	0,868	0,576	**0,892**
banknote	0,992	**0,999**	0,990	0,836	0,977	0,984	0,989	0,998
bupa	**0,673**	0,593	0,660	0,551	0,367	0,620	0,524	0,584
cleveland	0,280	0,263	**0,296**	0,292	0,241	0,282	0,218	0,279
cleveland-0vs4	0,631	0,668	0,662	**0,770**	0,481	0,638	0,527	0,512
ecoli-0-1-4-7vs2-3-5-6	0,813	**0,904**	0,766	0,671	0,607	0,780	0,702	0,683
ecoli1	0,843	**0,869**	0,819	0,633	0,819	0,837	0,830	0,830
ecoli2	0,865	**0,930**	0,811	0,277	0,485	0,843	0,753	0,839
ecoli3	0,735	**0,777**	0,519	0,649	0,473	0,761	0,527	0,721
glass-0-1-5vs2	0,526	0,524	0,475	0,371	0,475	**0,598**	0,475	0,511
hayes-roth	0,838	0,448	0,583	0,706	0,555	**0,850**	0,492	0,505
indian-liver-patient	**0,579**	0,529	0,513	0,556	0,417	0,571	0,417	0,501
led7digit-0-2-4-5-6-7-8-9vs1	0,885	0,683	0,883	0,757	**0,892**	0,890	0,536	0,748
new-thyroid1	0,965	0,933	0,768	0,959	0,773	0,940	0,923	**0,966**
newthyroid2	**0,964**	0,941	0,775	0,958	0,778	0,940	0,923	**0,964**
page-blocks0	**0,929**	0,880	0,884	0,717	0,758	0,909	0,832	0,860
poker-8-9vs5	0,497	0,497	0,497	0,497	0,497	**0,564**	0,497	0,519
poker-8-9vs6	0,531	**0,794**	0,589	0,496	0,496	0,547	0,496	0,592
shuttle-6vs2-3	0,974	0,796	0,642	0,989	0,489	**1,000**	0,949	0,944
transfusion	0,588	**0,621**	0,550	0,556	0,433	0,579	0,567	0,612
vertebral2c	**0,792**	0,751	0,753	0,768	0,406	0,774	0,511	0,775
vowel	0,925	0,938	0,735	0,661	0,419	0,812	0,912	**0,946**
winequality-red-4	0,500	0,505	0,492	0,528	0,492	**0,531**	0,492	0,512
winequality-red-8vs6-7	0,553	0,495	0,495	0,520	0,495	**0,541**	0,495	0,498
winequality-white-3-9vs5	0,530	0,523	0,496	**0,616**	0,496	0,608	0,496	0,530
wisconsin	0,962	**0,970**	0,965	0,959	0,968	0,944	0,926	0,948
yeast-0-5-6-7-9vs4	0,651	0,670	0,625	0,117	0,475	0,662	0,531	**0,714**
yeast-1-4-5-8vs7	0,494	0,488	0,489	0,121	0,489	**0,545**	0,484	0,497
yeast-2vs4	0,853	**0,872**	0,852	0,276	0,499	0,839	0,736	0,842
Média	0,728	0,711	0,668	0,600	0,564	0,728	0,632	0,697
Wilcoxon-p	0,045	0,726	0,191	0,046	10^{-4}	0.049	$3{,}458 \times 10^{-5}$	n/a
Win/Tie/Loss	11/2/17	16/0/14	19/0/11	19/0/11	26/0/4	12/0/18	25/1/4	n/a

APÊNDICE B – DIAGRAMA CD PARA AS MÉTRICAS ACURÁCIA E *F-MEASURE*

Figura 17 – Diagrama CD do *post-hoc* de Bonferroni-Dunn, em relação aos resultados da métrica Acurácia, para os algoritmos do estado da arte sobre as 30 bases de dados, sendo CD = 1,701.

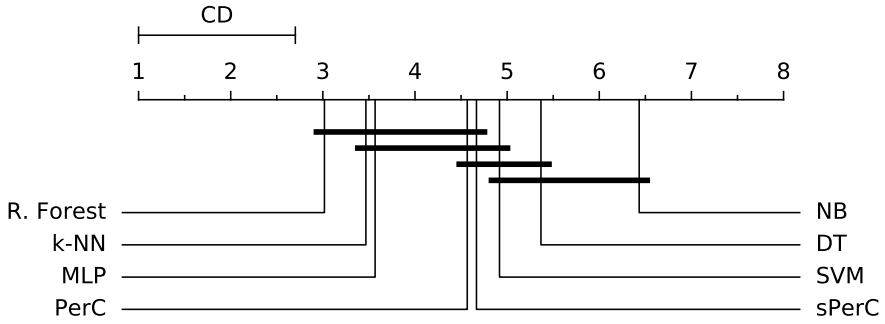


Figura 18 – Diagrama CD do *post-hoc* de Bonferroni-Dunn, em relação aos resultados da métrica *F-Measure*, para os algoritmos do estado da arte sobre as 30 bases de dados, sendo CD = 1,701.