ALGORITMOS INTEGRADOS PARA CLASSIFICAÇÃO DE DADOS COM ATRIBUTOS CATEGÓRICOS

Gabriel Matos Cardoso Leite

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Geraldo Bonorino Xexéo
Carlos Eduardo Pedreira
Carolina Gil Marcelino

Rio de Janeiro
Agosto de 2019

ALGORITMOS INTEGRADOS PARA CLASSIFICAÇÃO DE DADOS COM ATRIBUTOS CATEGÓRICOS

Gabriel Matos Cardoso Leite

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof. Felipe Maia Galvão França, Ph.D.

_____

Dra. Carolina Gil Marcelino, D. Sc.

_____

Profa. Elizabeth Fialho Wanner, D. Sc.

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2019

Leite, Gabriel Matos Cardoso

Algoritmos Integrados para Classificação de Dados com Atributos Categóricos/Gabriel Matos Cardoso Leite. – Rio de Janeiro: UFRJ/COPPE, 2019.

XII, 40p.: il.; 29,7 cm.

Orientador: Geraldo Bonorino Xexéo

Carlos Eduardo Pedreira

Carolina Gil Marcelino

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2019.

Referências Bibliográficas: p. 38-40.

1. Classificação Supervisionada. 2. Neighbourhood Component Analysis. 3. Dados Categóricos. I. Pedreira, Carlos Eduardo II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

DEDICATÓRIA

Aos meus pais.
À minha irmã.
À minha esposa.
A meu avô (in memoriam).
Com amor, admiração e gratidão.

AGRADECIMENTOS

Aos meus pais, minha irmã, minha esposa e meu avô que acompanharam e definiram os meios, tornando possível a minha formação de caráter e de conhecimento.

À COPPE e ao PESC por ter proporcionado as condições para que este trabalho pudesse ser concretizado.

Aos meus orientadores, professores Carlos Eduardo Pedreira, Carolina Gil Marcelino e Geraldo Bonorino Xexéo pela orientação, paciência, motivação, sugestões e por terem acreditado em mim e no meu trabalho. Agradeço também à banca examinadora por aceitar o convite e se dispor a participar da defesa desta dissertação.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ALGORITMOS INTEGRADOS PARA CLASSIFICAÇÃO DE DADOS COM ATRIBUTOS CATEGÓRICOS

Gabriel Matos Cardoso Leite

Agosto/2019

Orientadores: Geraldo Bonorino Xexéo
Carlos Eduardo Pedreira
Carolina Gil Marcelino

Programa: Engenharia de Sistemas e Computação

A classificação de padrões em dados categóricos e mistos apresenta um grande desafio a ser alcançado. O aumento na quantidade de dados gerados por diversas fontes requer classificadores que sejam capazes de lidar com diferentes tipos de dados. Este trabalho propõe a partir da integração de classificadores e formas de codificar atributos, uma nova abordagem para classificar dados. Por dados mistos entende-se conjunto de dados cujas observações são compostas por atributos contínuos e categóricos. O tratamento adequado de observações com atributos categóricos viabiliza, em classificações de padrões, a utilização de uma grande quantidade de bases. A abordagem proposta para tratar atributos categóricos e assim, viabilizar a aplicação de métodos de classificação de padrões, é resultante da integração em pares entre as codificações T*arget Encoding* (TE)*, One-hot*, N*aive* e os classificadores *Neighbourhood Componente Analysis* (NCA)*, Support Vector Machine* (SVM)*, k-Nearest Neighbors* (kNN)*.* Analisa-se o comportamento das codificações em bases de dados sintéticas e o desempenho dos algoritmos em bases de dados reais. A metodologia aplicada utilizou técnicas de validação cruzada, *k-fold* e um conjunto de teste com observações não vistas durante o treinamento. Técnicas de inferência estatística foram utilizadas a fim de identificar indícios de diferença entre os resultados da acurácia obtida pelos algoritmos integrados em cada conjunto de dados. O planejamento experimental realizado indicou que a integração formada pelo classificador NCA e a codificação TE (NCA+TE) se mostrou mais competitiva entre os demais.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

INTEGRATED CLASSIFICATION ALGORITHMS FOR DATA WITH CATEGORICAL FEATURES

Gabriel Matos Cardoso Leite

August/2019

Advisors: Geraldo Bonorino Xexéo
Carlos Eduardo Pedreira
Carolina Gil Marcelino

Department: Systems Engineering and Computer Science

Pattern classification on categorical and mixed data is a challenge to be surpassed. The increase in the amount of data being generated demands classifiers able to deal with different types of data. This work proposes algorithms for supervised classification on categorical and mixed data. Such algorithms are elaborated from integration between classifiers and ways of coding categorical features into continuous features. Mixed data is a set of observations with categorical features along with continuous features. Treating observations with categorical features properly allows the use of a huge number of databases containing categorical features. The approach proposed in order to handle categorical features and permit classification methods to be applied on such data, is a result of integration in pairs between the encodings T*arget Encoding* (TE)*, One-hot*, N*aive* and classifiers *Neighbourhood Componente Analysis* (NCA)*, Support Vector Machine* (SVM)*, k-Nearest Neighbors* (kNN). The behavior of the encodings chosen, and the performance of the presented algorithms are analyzed on synthetic databases and real databases, respectively. In order to evaluate the performance of the presented algorithms, an analysis was made on all results obtained. This analysis was made using cross-validation techniques, k-fold and a test set with unseen observations. Moreover, inferential statistics techniques were used to identify evidences of differences among integrated algorithm's accuracies on each dataset. The experimental planning proposed indicated that the integration built by NCA classifier and TE encoding (NCA+TE) turned up to be more competitive when compared to the other algorithms.

ÍNDICE

Lista de Figuras

Lista de Tabelas

Lista de Símbolos e abreviaturas

ABREVIATURAS

TE – *Target Encoding*

SVM – *Support vector machine*

kNN – *k-nearest neighbors*

NCA – N*eighbourhood componente analysis*

MDS – *Multidimensional scaling*

Capítulo 1

1 Introdução

Classificação de padrões é um campo em aprendizado de máquina que diz respeito a observações que tenham um rótulo de classe (conhecido ou não) associado [1]. Problemas como reconhecimento de caracteres [2], diagnósticos médicos [3] e reconhecimento de voz [4] são alguns exemplos comuns envolvendo classificação. Esta dissertação contempla exclusivamente problemas de classificações nos quais os rótulos de classe são conhecidos (aprendizado supervisionado). Dito isto, classificar consiste em mapear observações de um dado problema a um conjunto finito de possíveis rótulos para o problema [1].

Problemas envolvendo atributos representados por categorias (categóricos) são tão comuns quanto problemas envolvendo somente atributos numéricos ou quantitativos e problemas que possuem atributos numéricos e categóricos (mistos). Inúmeros classificadores para problemas com atributos numéricos foram propostos na literatura. Dentre estes classificadores destacam-se k-vizinhos mais próximos (*k-nearest neighbors*, kNN) [5], máquina de vetores de suporte (*support vector machine*, SVM) [6], e análise de componentes da vizinhança (*neighbourhood component analysis*, NCA) [7]. Para problemas puramente categóricos, muitos métodos podem ser extendidos de forma a viabilizar a sua aplicação em tais problemas [8]. A título de exemplo, o kNN pode ser modificado para utilizar uma métrica para distância capaz de quantificar a distância entre categorias [9]. Em contrapartida, alguns classificadores são inerentemente capazes de lidar tanto com problemas puramente quantitativos quanto com problemas puramente categóricos [8]. Árvores de decisão são um exemplo popular de tais tipos de classificadores [10].

Os métodos de classificação citados possuem muitas características em comum. Contudo é necessário destacar uma dentre estas características: a impossibilidade de lidar com problemas com dados mistos. Uma proposta de solução para viabilizar o uso desses classificadores em dados que possuem atributos categóricos é converter as categorias presentes nos dados para valores contínuos, permitindo assim, que estes algoritmos possam ser utilizados em sua forma canônica. Esta conversão, também conhecida como codificação, é feita de diversas formas.

A proposta deste trabalho é propor a construção de novos algoritmos de classificação integrados a codificadores e avaliar sua capacidade aplicados a bases reais (categóricas e mistas). Logo, o termo integrado representa a combinação entre uma codificação e um classificador. Dentre as codificações pesquisadas na literatura, este trabalho aborda três tipos: *naïve* [11] ou *dummy*, *one-hot* [12] e codificação por rótulo (*target encoding*, TE) [13]. Estas codificações diferem no grau de sofisticação com que os dados são tratados e na complexidade dos dados resultantes da aplicação da codificação. Dos classificadores listados, este trabalho utilizou o NCA, kNN e SVM. As árvores de decisão foram preteridas por conta de, durante a tarefa de classificação, os atributos dos dados serem tratados de forma independente (monovariada). Ao assumir independência entre atributos, a árvore descarta informações importantes acerca da distribuição dos atributos no problema [10]. Por outro lado, os classificadores NCA, kNN e SVM tratam atributos sem assumir alguma independência entre eles, isto é, de forma multivariada.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é propor a integração de técnicas para codificação de categorias em valores contínuos e métodos supervisionados de classificação de padrões. Tal proposta permite que problemas no campo de atributos categóricos e mistos possam ser abordados com uso de técnicas supervisionadas de aprendizado de máquina.

1.1.2 Objetivos Específicos

Além de obter uma integração entre métodos de classificação de padrões e técnicas de codificação para classificação de dados categóricos e mistos, este trabalho também abrange os objetivos específicos a seguir:

- Propor uma alteração na medida de dispersão utilizada pela codificação TE;
- Avaliar as codificações TE, *Naive* e *One-hot* com os classificadores NCA, kNN e SVM;
- Avaliar os algoritmos produzidos em bases de dados reais;
- Comparar por meio de inferência estatística os resultados obtidos pelos algoritmos.

1.2 Resumo das contribuições

As principais contribuições desta dissertação são:

- Alteração na medida de dispersão utilizada durante o cálculo feito pela codificação *Target Encoding* (Seção 3.2);
- Concepção e elaboração de algoritmos de classificação integrados a codificações com objetivo de classificar padrões sob bases de dados que possuam atributos categóricos (Seção 3.3);
- Implementação dos algoritmos propostos sob bases de dados reais com esquemas de teste fora da amostra (Seção 3.5);
- Comparação dos resultados obtidos pelos algoritmos integrados propostos com uso de inferência estatística (Capítulo 4);
- Discussão dos resultados obtidos (Capítulo 4).

1.3 Organização do documento

Esta dissertação está organizada da seguinte maneira. No capítulo 2 se faz uma revisão da literatura sobre os classificadores kNN, SVM, NCA, definindo-os em termos conceituais e mostrando como a classificação é feita em cada um. Igualmente, são apresentadas as codificações *dummy*, *one-hot*, *target encoding* e os efeitos de se aplicar essas codificações nos dados.

O Capítulo 3 descreve o fluxo para concepção dos distintos algoritmos integrados propostos neste trabalho. Neste capítulo, as bases de dados são apresentadas, seguido das metodologias utilizadas, das técnicas de validação cruzada e dos métodos estatísticos utilizados na avaliação da proposta.

A descrição dos experimentos realizados e discussão dos resultados dos mesmos é feita no Capítulo 4. No capítulo 5 é apresentada uma breve discussão além das conclusões obtidas. São também indicadas propostas de trabalhos futuros.

Capítulo 2

2 Revisão Bibliográfica

Neste capítulo se apresenta, com o objetivo de tornar o texto mais autocontido, a fundamentação teórica que foi utilizada neste trabalho. As Seções 2.1, 2.2 e 2.3 dão uma breve explicação sobre os algoritmos de classificação que serviram de base nesta dissertação. A Seção 2.4 discute sobre as três codificações abordadas tanto no método proposto como nos experimentos realizados.

2.1 K-Vizinhos mais próximos

O algoritmo k-vizinhos mais próximos (*k-nearest neighbors*, kNN) é um classificador não-paramétrico amplamente conhecido e utilizado [5] [14], no qual o rótulo de uma nova observação **x** é por votação, pela maioria das *K* observações mais próximas. Assim, o rótulo associado a uma nova observação é aquele com a maior probabilidade a posteriori [1], definida por,

$$p(y = c \,|\, x, K) = \frac{1}{K} \sum_{i \in N_K(x)} \mathbb{I}(y_i = c), \tag{2.1}$$

na qual $N_K(x)$ são os índices dos *K* pontos mais próximos a **x**, $c \in (1,2,\cdots,C)$ define o conjunto de rótulos possíveis. O termo $\mathbb{I}(\mathrm{v})$ é uma função indicadora definida como:

$$\mathbb{I}(\mathrm{v}) = \begin{cases} 1 \; se \; a \; condição \; v \; é \; verdadeira \\ 0 \; caso \; contrário. \end{cases} \tag{2.2}$$

Para medir a proximidade de um vizinho podem ser utilizadas métricas diversas. Neste trabalho utilizou-se a distância euclidiana. O Algoritmo 1 apresenta o pseudocódigo do classificador KNN.

Entrada: Conjunto de observações $X_{tr} = \{x_{tr_1}, x_{tr_2}, \cdots, x_{tr_n}\}$ e classes $Y_{tr} = \{y_{tr_1}, y_{tr_2}, \cdots, y_{tr_n}\}$ de treino; conjunto de observações $X_t = \{x_{t_1}, x_{t_2}, \cdots, x_{t_m}\}$
Saída: Conjunto de classes $Y_t = \{y_{t_1}, y_{t_2}, \cdots, y_{t_m}\}$
Definir o parâmetro **K**
Para cada observação $x_i \in X_t$ **faça:**

> Calcular a distância euclidiana $d_{ij} = \|x_i - x_j\|$ para todo $x_j \in X_{tr}$
> Guardar as distâncias em uma lista L e ordená-la
> A classe y_i é dada pela classe mais frequente nas **K** observações relativas às **K** primeiras entradas na lista L

Fim
Algoritmo 1: Pseudocódigo de *k-nearest neighbors,* kNN

2.2 Máquina de vetores de suporte

Máquina de vetores de suporte - *support vector machine* (SVM) é um algoritmo clássico e muito bem-sucedido em problemas de classificação [15]. A ideia central é encontrar o hiperplano que separe o conjunto de observações com a maior margem possível. Seja N o conjunto de dados de dimensão m, x_n a *n*-ésima observação e $y_n \in \{-1, 1\}$ o rótulo correspondente, este hiperplano é dado por,

$$w^T \phi(\boldsymbol{x}) + b = 0. \tag{2.3}$$

Na Equação 2.4, w^T é um vetor de dimensão m, b é o termo que representa o *bias* [16] e $\phi(\boldsymbol{x})$ é a transformação não-linear $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^{\tilde{d}}$ que resulta no conjunto de dados $\boldsymbol{z} = \phi(\boldsymbol{x})$ linearmente separável. O vetor w que define o hiperplano é obtido via solução de um problema de otimização modelado conforme a Equação 2.4,

$$\min_{b,\boldsymbol{w}} \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w}, \tag{2.4}$$

com a restrição $y_n(\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b) \geq 1$ para $n = 1, \cdots, N$. Este método é conhecido como *hard-margin SVM* e o hiperplano obtido particiona os dados como mostra a Figura 2.1.

Caso a condição de obter a maior margem seja relaxada para que alguns pontos possam ultrapassar a margem, é necessário que uma regularização seja empregada para evitar *overfitting*. Este método é conhecido como *soft-margin* SVM. Dito isto, a Equação 2.4 passa a ter a forma:

$$\min_{b,\boldsymbol{w},\boldsymbol{\xi}} \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} + C \sum_{n=1}^{N} \xi_n, \tag{2.5}$$

com respeito a

$$y_n(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) + b) \geq 1 \qquad n = 1, \cdots, N$$
$$\xi_n \geq 0 \qquad n = 1, \cdots, N.$$

A variável ξ_n quantifica quanto a observação $\boldsymbol{x}_n$ violou a margem encontrada e C é o parâmetro de regularização que define quanto a violação da margem é importante. Para valores grandes de C, o resultado se aproxima do método *hard-margin SVM*. A classificação de uma observação x_i em ambos os métodos é dada por,

$$y_i = sign(w^T\phi(\boldsymbol{x}) + b). \tag{2.6}$$

Ao utilizar a transformação não-linear $\phi(x)$, é necessário que o SVM consiga calcular de forma eficiente o produto interno $\phi(x)^T\phi(x')$ entre dois vetores $x, x' \in \mathbb{R}^{\tilde{d}}$, necessário para solucionar o *dual* do problema apresentado na Equação 2.5 [17]. O SVM executa o cálculo do produto interno por meio do uso de funções de *kernel* [16]. Uma função de *kernel* permite que o cálculo do produto interno seja feito sem levar em conta a dimensão $\tilde{d}$ do problema [17]. Tais funções são dadas por,

$$K_\phi(x, x') \equiv \phi(x)^T\phi(x'). \tag{2.7}$$

Diversas funções de *kernel* foram propostas na literatura, dentre as quais se destacam:

- Linear: $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T\boldsymbol{x}'$
- Polinomial: $K(\boldsymbol{x}, \boldsymbol{x}') = (\zeta + \gamma\boldsymbol{x}^T\boldsymbol{x}')^Q,\ \gamma > 0,\ \zeta > 0,\ Q \in \mathbb{N}$;
- *Radial Basis Function* (RBF): $K(\boldsymbol{x}, \boldsymbol{x}') = exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}'\|^2),\ \gamma > 0$.

No Algoritmo 2 é apresentado o pseudocódigo do classificador SVM.

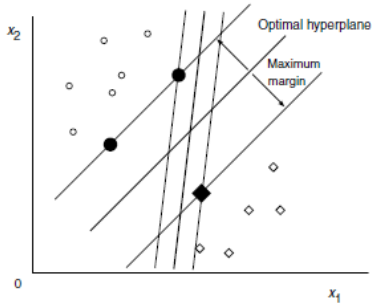

Figura 2.1: Ilustração do hiperplano e margens encontradas pelo *hard-margin* SVM. Extraído de **[16]**

Entrada: Conjunto de observações $X_{tr} = \{x_{tr_1}, x_{tr_2}, \cdots, x_{tr_n}\}$ e classes $Y_{tr} = \{y_{tr_1}, y_{tr_2}, \cdots, y_{tr_n}\}$ de treino; conjunto de observações $X_t = \{x_{t_1}, x_{t_2}, \cdots, x_{t_m}\}$
Saída: Conjunto de classes $Y_t = \{y_{t_1}, y_{t_2}, \cdots, y_{t_m}\}$
Definir o parâmetro **C**
Definir a função de *kernel*
Encontrar o hiperplano dado por $\boldsymbol{w}$ e b utilizando X_{tr}
Para cada observação $x_i \in X_t$ **faça:**
    A classe da observação x_i é calculada por $y_i = sign(w^T \phi(\boldsymbol{x}) + b)$
Fim

Algoritmo 2: Pseudocódigo de *support vector machine,* SVM

2.3 Neighbourhood component analysis

Neighbourhood Component Analysis (NCA) é um classificador cuja proposta é encontrar a partir de um conjunto de dados, uma transformação linear para um novo espaço e aplicar um kNN neste novo espaço. A transformação linear visa melhorar a performance do kNN no novo espaço em comparação à performance no espaço original [7]. A avaliação da performance do kNN é feita utilizando validação cruzada *leave-one-out* (LOO).

Devido a característica do kNN de particionar o espaço, o valor da função que calcula o erro da classificação LOO do kNN para um conjunto, $\{x_1, \cdots, x_n\} - \{x_i\}$ é diferente do valor obtido para o conjunto $\{x_1, \cdots, x_n\} - \{x_j\}$. Esta diferença evidencia uma descontinuidade na função do erro LOO. Para tratar o problema da descontinuidade do erro da classificação LOO, o NCA suaviza a atribuição de vizinhos no novo espaço. A suavização é feita de maneira que a seleção de qual observação é vizinha a cada observação deixa de ser determinística, como é feita originalmente no kNN, e passa a ser probabilística. Logo, seja p_{ij} a probabilidade do ponto i escolher o ponto j como seu vizinho, o valor de p_{ij} é dado por,

$$p_{ij} = \begin{cases} \dfrac{\kappa\left(D(x_i, x_j)\right)}{\sum_{k \neq i} \kappa\left(D(x_i, x_k)\right)}, & i \neq j \\ 0, & i = j. \end{cases} \qquad 2.8$$

Na equação 2.8, $D(x_i, x_j) = \left\| Ax_i - Ax_j \right\|^2$ é a distância euclidiana no espaço transformado por A e $\kappa(z) = e^{-\frac{z}{\sigma}}$. A função $\kappa(z)$ é uma função de *kernel* com largura definida por σ [18]. Seja $C_i = \{j \mid c_i = c_j\}$ o conjunto de pontos com a mesma classe que

o ponto i, a probabilidade deste ponto ser corretamente classificado pelo algoritmo é dada pela Equação 2.9,

$$p_i = \sum_{j \in C_i} p_{ij.} \tag{2.9}$$

A transformação A é obtida maximizando a probabilidade de cada ponto no conjunto ser classificado corretamente, dada pela Equação 2.10,

$$f(A) = \sum_i \sum_{j \in C_i} p_{ij} = \sum_i p_i. \tag{2.10}$$

Diferenciando esta função em relação a A, obtém-se a expressão a seguir:

$$\frac{\partial f}{\partial A} = 2A \sum_i \left(p_i \sum_k p_{ik} x_{ik} x_{ik}^T - \sum_{j \in C_i} p_{ij} x_{ij} x_{ij}^T \right). \tag{2.11}$$

As duas funções apresentadas podem ser modificadas para incluir um parâmetro de regularização [18]. A inclusão de um parâmetro de regularização é feita para minimizar o erro da classificação de novas observações não vistas, evitando *overfitting*[1] [17]. As novas funções com regularização são definidas pelas equações 2.12 e 2.13 nas quais N é o número de elementos de A, conforme,

$$f(A) = \sum_i p_i - \frac{\lambda}{N} \|A\|_F^2. \tag{2.12}$$

$$\frac{\partial f}{\partial A} = 2A \sum_i \left(p_i \sum_k p_{ik} x_{ik} x_{ik}^T - \sum_{j \in C_i} p_{ij} x_{ij} x_{ij}^T \right) - \frac{\lambda}{N} A \tag{2.13}$$

Sumarizando o funcionamento do NCA, ao maximizar a Equação 2.10 utilizando algum método de otimização de primeira ou segunda ordem, como os métodos baseados na classe de algoritmos que usam a informação do vetor gradiente, obtém-se A. Este trabalho utilizou o método de gradiente conjugado [19]. Por ser um método de segunda

[1] *Overfitting* é um fenômeno no qual a complexidade do modelo ajustado é maior do que o necessário. Desta forma o modelo obtém erro 0 dentro da amostra, ao custo da sua capacidade de generalização, levando a um erro alto quando avaliado fora da amostra.

ordem, a convergência é alcançada com um menor número de iterações quando comparado com métodos de otimização de primeira ordem [20]. Ao obter a transformação A, ela é aplicada aos dados [20]. A classificação de uma nova observação é feita transformando este novo ponto e atribuindo o rótulo de maior probabilidade na vizinhança neste novo espaço. Uma propriedade interessante do NCA surge a partir da transformação linear encontrada. Caso A seja de dimensão Dxd com $d < D$, é possível encontrar uma representação dos pontos em um espaço de dimensão menor e com $d = 2$ ou $d = 3$, pode-se visualizar esta nova representação. O Algoritmo 3 exibe o pseudocódigo do classificador NCA.

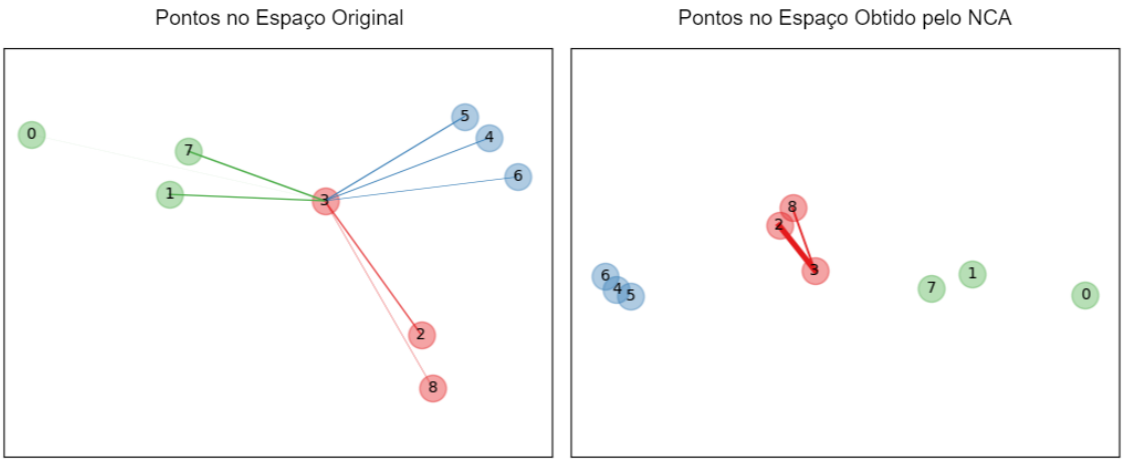

Figura 2.2: À esquerda pontos originalmente em $\mathbb{R}^2$ com linha interligando pontos cuja probabilidade $\boldsymbol{p_{3i} \neq 0}$ e espessura variando conforme o valor das probabilidades. À direita os mesmos pontos no espaço transformado pelo NCA. Nota-se que o algoritmo agrupou pontos com a mesma classe. Adaptado de **[21]**.

Entrada: Conjunto de observações $X_{tr} = \{x_{tr_1}, x_{tr_2}, \cdots, x_{tr_n}\}$ e classes $Y_{tr} = \{y_{tr_1}, y_{tr_2}, \cdots, y_{tr_n}\}$ de treino; conjunto de observações $X_t = \{x_{t_1}, x_{t_2}, \cdots, x_{t_m}\}$
Saída: Conjunto de classes $Y_t = \{y_{t_1}, y_{t_2}, \cdots, y_{t_m}\}$; Matriz A
Definir a dimensão d de A
Inicializar A com valores aleatórios
Calcular p_{ij} para todo $x_i \in X_{tr}$ e $x_j \in X_{tr}$
Minimizar $f(A)$
Para cada observação $x_i \in X_t$ **faça:**
    Calcular p_{ij} para todo $x_j \in X_{tr}$
    Calcular a probabilidade de cada classe do problema a partir dos valores de p_{ij}
    A classe y_i é dada pela classe com maior probabilidade
Fim

Algoritmo 3: Pseudocódigo de *neighbourhood componente analysis,* NCA

2.4 Codificações

Problemas com características que não podem ser representadas numericamente são comuns. Existem diversas maneiras de permitir o uso de classificadores inaptos a lidar com dados mistos em sua forma canônica. A maneira de tratar dados mistos utilizada neste trabalho é chamada de codificação ou *encoding* [22]. Nesta seção são descritas três formas de codificação de uso frequente na literatura que foram utilizadas no desenvolvimento deste trabalho.

2.4.1 Naïve

Pode ser considerado a mais simples dentre as codificações apresentadas. Consiste em, para cada variável categórica, atribuir um valor numérico a cada uma das categorias vistas, como mostrado na Figura 2.3. Um problema desta codificação é que ao atribuir uma sequência de valores às categorias, ela induz uma ordenação que não corresponde à realidade, resultando na perda de informações sobre a variável em questão [22].

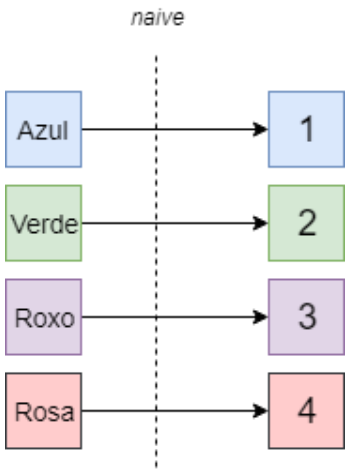

Figura 2.3: Exemplo da codificação *naive* para uma variável com 4 valores possíveis

2.4.2 One-hot

Nesta codificação, não é induzida uma ordenação entre as categorias como é feita pela codificação Naïve. A codificação *one-hot* deriva do estudo de máquinas de estados, com o propósito de codificar o estado atual de uma máquina. A codificação utiliza *bits* com valor 0 e um *bit* com valor 1 indicando qual o estado atual [23] [24]. Em aprendizado de máquina, esta codificação é utilizada para mapear categorias em vetores numéricos.

Seja C_i a variável que representa o atributo categórico i com o conjunto de possíveis valores dado por $C_i \in \{c_1, c_2, \cdots, c_k\}$ a codificação *one-hot* que consiste em criar um vetor de dimensão $d = |\{c_1, c_2, \cdots, c_k\}|$ com 1 na entrada j, correspondente à categoria

j e 0 nas outras entradas [12]. Assim, as categorias aparecem ortogonais entre si após o mapeamento, como mostra a Figura 2.4.

Na codificação *one-hot*, cada atributo categórico se transforma em um vetor de dimensão d e o conjunto de dados passa a ter dimensão $D - N_c + \sum_i |C_i|$, na qual D é a dimensão original das observações na base de dados. N_c é o número de atributos categóricos presentes em cada observação e $|C_i|$ é a cardinalidade do conjunto de possíveis valores que a variável representando a categoria i pode assumir. A título de exemplo, seja $X_i \in \mathbb{R}^2$ a observação i de uma base com 2 atributos categóricos C_1 e C_2 de cardinalidades $|C_1| = 10$ e $|C_2| = 12$, respectivamente. Ao aplicar a codificação *one-hot* a esta base, o atributo C_1 é mapeado para um novo atributo numérico $N_1 \in \mathbb{R}^{10}$ e o atributo C_2 é mapeado para um novo atributo numérico $N_2 \in \mathbb{R}^{12}$. Ao fim da codificação, as observações desta base pertencerão ao espaço $\mathbb{R}^{27}$, de modo que, é possível notar que quando houver muitas variáveis categóricas com alta cardinalidade, a dimensão do problema cresce bastante. Este crescimento da dimensão original do problema dificulta o processamento dos novos dados. O Algoritmo 4 apresenta o pseudocódigo da codificação o*ne-hot*.

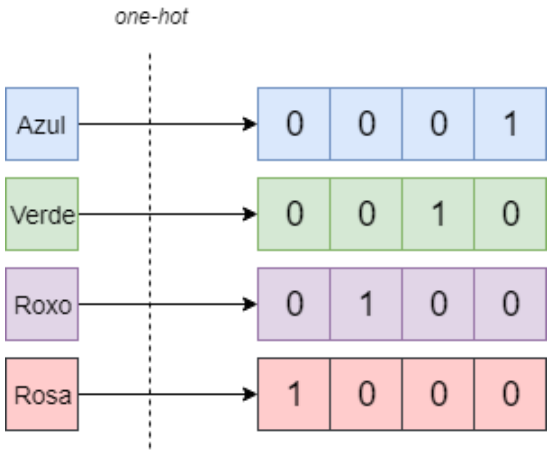

Figura 2.4: Exemplo da codificação *one-hot* para uma variável com 4 valores possíveis

Entrada: Conjunto de observações $X_{tr}^{c} = \{x_{tr_1}^{c}, x_{tr_2}^{c}, \cdots, x_{tr_n}^{c}\}$ e $X_{t}^{c} = \{x_{t_1}^{c}, x_{t_2}^{c}, \cdots, x_{t_m}^{c}\}$
Saída: Conjunto de observações $X_{tr} = \{x_{tr_1}, x_{tr_2}, \cdots, x_{tr_n}\}$ e $X_{t} = \{x_{t_1}, x_{t_2}, \cdots, x_{t_m}\}$
T = tabela de correspondências vazia
Para cada atributo A_i de X_{tr}^{c} **faça:**
    L = lista das categorias presentes no atributo a_i de todas as observações em X_{tr}^{c}
    T_i = tabela de correspondência vazia do atributo A_i
    Para cada categoria c_i em A_i **faça:**
        v = vetor de zeros de tamanho igual a cardinalidade $|A_i|$ com 1 na entrada correspondente a c_i
        $T_i[c_i] = v$
    Fim
    $T[A_i] = T_i$
Fim
Para cada $x_{tr_i}^{c} \in X_{tr}^{c}$ **faça:**
    Para cada atributo a_i de $x_{tr_i}^{c}$ **faça:**
        $x_{tr_i}[a_i] = T[a_i][x_{tr_i}^{c}]$
    Fim
Fim
Para cada $x_{t_i}^{c} \in X_{t}^{c}$ **faça:**
    Para cada atributo a_i de $x_{t_i}^{c}$ **faça:**
        Se $x_{t_i}^{c}$ existe em $T[a_i]$ **faça:**
            $x_{t_i}[a_i] = T[a_i][x_{t_i}^{c}]$
        Senão faça:
            v_z = vetor de zeros de tamanho igual a cardinalidade de $|a_i|$
            $x_{t_i}[a_i] = v_z$
        Fim
    Fim
Fim

Algoritmo 4: Pseudocódigo da codificação *One-hot*

2.4.3 Target Encoding

A codificação *Target Encoding* (TE) busca mitigar os problemas presentes nas codificações Naïve e *One*-hot, em dados com variáveis categóricas que possuem alta cardinalidade, de forma a tentar incorporar uma ordenação e posicionamento intrínsecos aos dados, evitando que a dimensão do problema aumente em excesso. Para alcançar este objetivo, *Target Encoding* faz uso dos rótulos dos dados, o que restringe sua aplicação a somente problemas supervisionados [13].

Seja X_i um valor que o atributo categórico X pode assumir e Y o conjunto de todos os possíveis rótulos no conjunto de dados, o mapeamento entre categorias e valores

contínuos é dado por um vetor de dimensão $d = |Y|$ em que cada entrada j é calculada de acordo com a Equação 2.14,

$$X_i \rightarrow S_i \cong P\big(Y = Y_j \big| X = X_i\big). \tag{2.14}$$

Como $P(Y = Y_j|X = X_i)$ é uma probabilidade, S_i é um escalar normalizado entre 0 e 1. Devido a codificação utilizar estatísticas calculadas sobre os rótulos das observações, é necessário um particionamento dos dados entre treino e teste. Este particionamento é necessário para que o classificador utilizado em conjunto com esta codificação tenha uma avaliação de seu desempenho fora da amostra.

Seja n_i o número de observações com $X = X_i$, n_{iY_j} o número de observações com $X = X_i$ e $Y = Y_j$, n_{Y_j} o número de observações com $Y = Y_j$ e n_{TR} o tamanho do conjunto de treino, quando n_{iY_j} for pequeno, a estimativa de $P(Y = Y_j|X = X_i)$ perde a confiança. Para contornar este problema, S_i passa a ser uma combinação convexa entre a probabilidade a posteriori e a probabilidade a priori, como mostra a equação 2.15,

$$\mathrm{S_i} = \lambda(\mathrm{n_i})\frac{\mathrm{n_{iY_j}}}{n_i} + \big(1 - \lambda(\mathrm{n_i})\big)\frac{\mathrm{n_Y}}{\mathrm{n_{TR}}}. \tag{2.15}$$

A função $\lambda(n_i)$ é uma função monotonicamente crescente com valores entre 0 e 1 dada por,

$$\lambda(n) = \frac{n}{m + n}, \tag{2.16}$$

com m sendo a razão entre a variância das classes para $X = X_i$, σ^2 e a variância das classes no conjunto de treino, τ^2 [13].

O T*arget Encoding* é uma codificação mais sofisticada que as codificações *One-hot* e *Naive* pois incorpora informações de ocorrências das categorias na base. Essa sofisticação é dada pelo uso de, tanto a probabilidade empírica de ocorrência de cada categoria condicionada no rótulo das observações das categorias em análise, quanto a probabilidade a priori de ocorrência de cada uma das classes da base. O pseudocódigo da codificação TE é apresentado no Algoritmo 5.

Entrada: Conjunto de observações $X_{tr}^{c} = \{x_{tr_1}^{c}, x_{tr_2}^{c}, \cdots, x_{tr_n}^{c}\}$, $Y_{tr} = \{y_{tr_1}, y_{tr_2}, \cdots, y_{tr_n}\}$ e $X_{t}^{c} = \{x_{t_1}^{c}, x_{t_2}^{c}, \cdots, x_{t_m}^{c}\}$

Saída: Conjunto de observações $X_{tr} = \{x_{tr_1}, x_{tr_2}, \cdots, x_{tr_n}\}$ e $X_t = \{x_{t_1}, x_{t_2}, \cdots, x_{t_m}\}$

T = tabela de correspondências vazia, τ^2 = entropia do conjunto Y_{tr}

Para cada atributo A_i de X_{tr}^{c} **faça:**

- L = lista das categorias presentes no atributo a_i de todas as observações em X_{tr}^{c}
- T_i = tabela de correspondência vazia do atributo A_i
- **Para cada** categoria c_i em A_i **faça:**
  - σ^2 = entropia do conjunto Y_{tr}^{ci} de classes das observações cujo atributo $A_i = c_i$
  - $m = \frac{\sigma^2}{\tau^2}$
  - v = vetor de zeros de tamanho igual ao número de classes distintas
  - n_{ci} = número de observações nas quais $A_i = c_i$
  - **Para cada** classe y_{tr_i} em Y_{tr} **faça:**
    - $n_{y_{tr_i}}^{ci}$ = número de observações nas quais $A_i = c_i$ e classe = y_{tr_i}
    - $n_{y_{tr_i}}$= número de observações nas quais a classe é y_{tr_i}
    - $v[y_{tr_i}] = \lambda(n_{ci}) \frac{n_{y_{tr_i}}^{ci}}{n_{ci}} + (1 - \lambda(n_{ci})) \frac{n_{y_{tr_i}}}{|X_{tr}^{c}|}$
  - **Fim**
  - $T_i[c_i] = v$
- **Fim**
- $T[A_i] = T_i$

Fim

Para cada $x_{tr_i}^{c} \in X_{tr}^{c}$ **faça:**

- **Para cada** atributo a_i de $x_{tr_i}^{c}$ **faça:**
  - $x_{tr_i}[a_i] = T[a_i][x_{tr_i}^{c}]$
- **Fim**

Fim

Para cada $x_{t_i}^{c} \in X_{t}^{c}$ **faça:**

- **Para cada** atributo a_i de $x_{t_i}^{c}$ **faça:**
  - **Se** $x_{t_i}^{c}$ existe em $T[a_i]$ **faça:**
    - $x_{t_i}[a_i] = T[a_i][x_{t_i}^{c}]$
  - **Senão faça:**
    - $x_{t_i}[a_i] = \left[\frac{n_{y_{tr_1}}}{|X_{tr}^{c}|}, \frac{n_{y_{tr_1}}}{|X_{tr}^{c}|}, \cdots, \frac{n_{y_{tr_r}}}{|X_{tr}^{c}|}\right]$, para r classes distintas
  - **Fim**
- **Fim**

Fim

Algoritmo 5: Pseudocódigo da codificação *Target Encoding,* TE.

Capítulo 3

3 Metodologia

As seções 3.1 e 3.2 apresentam conceitos metodológicos usados na seção 3.3 e no planejamento de experimentos proposto para avaliação dos resultados contidos no Capítulo 4. A seção 3.3 descreve a proposta de integração entre as codificações e classificadores apresentados no Capítulo 2.

3.1 Dados Categóricos e Mistos

Um conjunto de dados é composto por observações caracterizadas por meio de atributos quantitativos ou qualitativos. Uma variável é definida como categórica (ou qualitativa) caso a sua escala de medida seja feita a partir de um conjunto de categorias [25]. Medidas para variáveis categóricas que não possuem uma ordem entre si são chamadas de variáveis nominais. As variáveis categóricas que possuem uma ordenação natural são chamadas de variáveis ordinais. Evidentemente, este tipo de variável, ao ser utilizada, precisa que a ordenação seja levada em consideração.

Em contrapartida, variáveis quantitativas podem assumir qualquer valor real e são chamadas de variáveis contínuas. As variáveis discretas são aquelas que assumem somente valores inteiros dentro de um intervalo definido [26]. Variáveis discretas também podem ser consideradas como variáveis categóricas, caso cada número inteiro dentro do intervalo definido corresponda a uma categoria.

Define-se uma base de dados como categórica se os atributos de suas observações são representados somente por variáveis categóricas. Caso as observações de uma base possuam atributos representados por variáveis tanto qualitativas quanto quantitativas, essa base é definida como mista [27].

3.2 *Target Encoding* com Entropia

Na codificação *Target Encoding* é feita para cada atributo categórico uma combinação convexa entre a probabilidade a priori de ocorrência dos rótulos e a probabilidade a posteriori de ocorrência das categorias dados os rótulos. Esta combinação é feita por meio de uma função monotonicamente crescente $\lambda(n_i) = \frac{n_i}{m+ n_i}$, na qual o parâmetro n_i indica o número de ocorrências da categoria c_{ij} para o atributo categórico C_i na base de dados. O valor de m é dado pela razão entre a dispersão dos rótulos atribuídos as observações com $C_i = c_{ij}$ e a dispersão dos rótulos atribuídos as observações por toda a base. A medida de dispersão utilizada em [13] é a variância.

Em problemas de classificação supervisionada, os rótulos são dados por variáveis nominais, e.g., “bom”, “muito bom”, “neutro”. Tais rótulos podem ser mapeados de diversas formas para valores numéricos. A título de exemplo, seja a variável aleatória L que representa as possíveis classes de um problema. Os valores que a variável L pode assumir são $L_1 = "bom"$, $L_2 = "muito\ bom"$ e $L_3 = "neutro"$, com probabilidades $P(L_1) = 0.4, P(L_2) = 0.3, P(L_3) = 0.3$. Um possível mapeamento destes rótulos é $l_1 = 1,\ l_2 = 2,\ l_3 = 3$. A variância da variável L é dada por:

$$Var(L) = E[L^2] - E[L]^2\ , \tag{3.1}$$
$$Var(L) = \sum_{i=1}^{3} l_1^2 \cdot P(l_i) - \left(\sum_{i=1}^{3} l_1 \cdot P(l_i)\right)^2\ ,$$
$$Var(L) = 1 \cdot 0.4 + 4 \cdot 0.3 + 9 \cdot 0.3 - (1 \cdot 0.4 + 2 \cdot 0.3 + 3 \cdot 0.3)^2,$$
$$Var(L) = 0.69.$$

Uma outra forma de quantificar a dispersão de variáveis nominais é dada pela entropia proposta por Shannon em [28]. Para o cálculo da entropia não é necessário mapear os valores nominais para valores numéricos, pois somente são utilizadas as probabilidades de ocorrência de cada valor. A entropia $H(L)$ para o exemplo anterior é dada por,

$$H(L) = -\sum_{i=1}^{n} P(l_i) \log_2 P(l_i)\ , \tag{3.2}$$
$$\mathrm{H(L)} = -\sum_{\mathrm{i=1}}^{3} P(l_i) \log_2 P(l_i)\ ,$$
$$\mathrm{H(L)} = -\big(0.4 \cdot -1.32193 + 2 \cdot (0.3 \cdot -1.73697)\big),$$
$$\mathrm{H(L)} = 1{,}570954.$$

Supondo $R = 2 \cdot L$, as probabilidades associadas às ocorrências se mantêm. Em contrapartida, os mapeamentos são alterados para $r_1 = 2 \cdot l_1,\ r_2 = 2 \cdot l_2,\ r_3 = 2 \cdot l_3$. A variância de R é dada pela equação 2.7

$$Var(R) = E[R^2] - E[R]^2, \tag{3.3}$$
$$Var(R) = \sum_{i=1}^{3} r_1^2 \cdot P(r_i) - \left(\sum_{i=1}^{3} r_1 \cdot P(r_i)\right)^2\ ,$$
$$Var(R) = \sum_{i=1}^{3} (2l_i)^2\ \cdot P(r_i) - \left(\sum_{i=1}^{3} 2l_1 \cdot P(r_i)\right)^2\ ,$$
$$\mathrm{Var(R)} = 4 \cdot \mathrm{Var(L)}.$$

É possível notar que ao alterar o mapeamento para duas vezes o mapeamento inicial, a variância de L aumentou quatro vezes, enquanto a entropia não se altera devido às probabilidades das classes se manterem.

Ao utilizar a variância como medida de dispersão para as classes de um problema, é observada uma inconsistência na codificação visto que, o resultado é sensível ao mapeamento feito. Com o objetivo de evitar esta sensibilidade do valor de m em $\lambda(n_i)$ ao mapeamento utilizado nas bases para os possíveis rótulos das observações, foi proposta uma alteração na codificação TE. Tal alteração abrange a utilização da entropia no lugar da variância como medida de dispersão para o cálculo de m.

3.3 Algoritmo Integrado para Classificação de Dados com Atributos Categóricos

Tipicamente um conjunto de dados é separado em duas partições, uma para treinar (ajustar parâmetros) e outra para avaliar a qualidade do mesmo fora-da-amostra. Baseado nisto, uma das contribuições desta dissertação é propor uma combinação entre os classificadores e codificações discutidas na seção 2.4. Tal combinação é ilustrada pela Figura 3.1. Após o particionamento dos dados nos conjuntos de treino e teste, o conjunto de treino é utilizado para montar uma tabela chamada *lookup table*[2]. Esta tabela contém todos os mapeamentos de categorias para valores numéricos seguindo o cálculo de cada

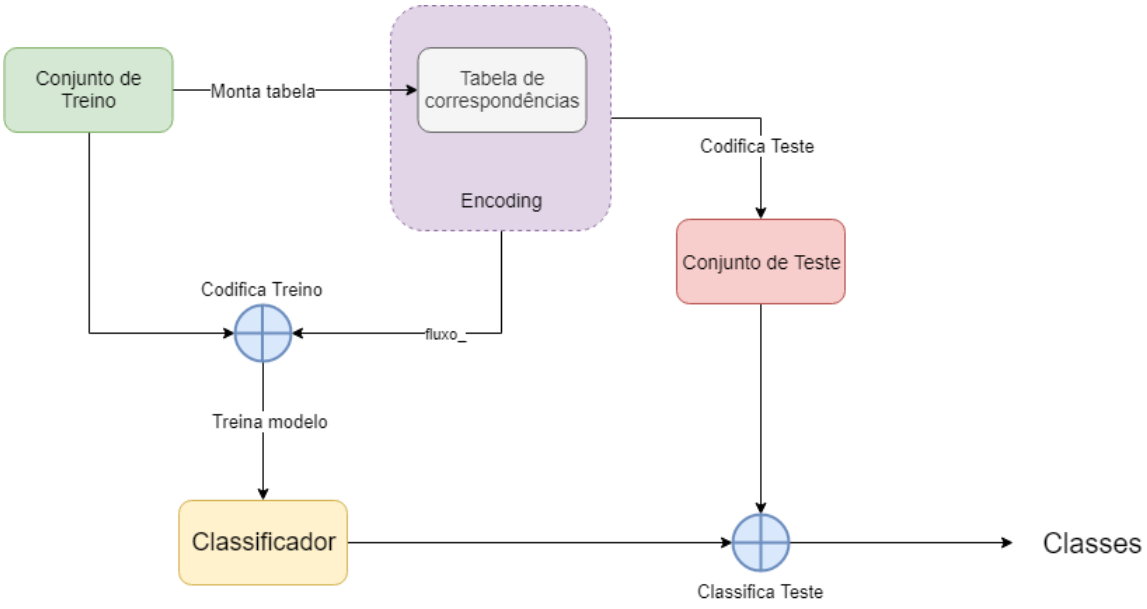

Figura 3.1: Ilustração do fluxo da abordagem proposta desde a etapa de treinamento do modelo até a etapa de avaliação fora da amostra

codificação. Uma vez montada a tabela, o conjunto de treino é codificado e dado como

[2] Entende-se por *lookup table* uma tabela responsável por mapeamentos na qual a entrada é a categoria e a saída é o valor contínuo atribuído a esta categoria

entrada para a etapa de ajuste de parâmetros (treino) do classificador. Depois de treinado o classificador, o conjunto de teste é codificado. Por fim, os rótulos das observações presentes no conjunto de teste são obtidos após o classificador treinado ser aplicado neste conjunto.

Ao fim foram geradas e avaliadas integrações entre codificadores e classificadores, possibilitando a concepção de nove algoritmos distintos. A abordagem proposta neste trabalho é capaz de realizar a classificação de padrões em dados mistos, com uso dos algoritmos dispostos na Tabela 3.1.

Tabela 3.1: Pares de codificação + classificador

Codificação + Classificador		
TE + NCA	TE + kNN	TE + SVM
ONE-HOT + NCA	ONE-HOT + kNN	ONE-HOT + SVM
NAIVE + NCA	NAIVE + kNN	NAIVE + SVM

Para cada algoritmo, os atributos foram normalizados entre 0 e 1. Um conjunto de hiperparâmetros foi utilizado de acordo com a Tabela 3.2.

Tabela 3.2: Valores dos hiperparâmetros utilizados em cada um dos classificadores nos experimentos realizados

Classificador	Hiperparâmetros
NCA	$epsilon = 10^{-7}$; $\lambda = \frac{0.007}{N}$ [3];
KNN	$K = 10$;
SVM	$C = 1.0$; $kernel = rbf$;

Além dos parâmetros mostrados, devido ao SVM ser idealizado para lidar com problemas de apenas duas classes (binários), foi necessário tratar situações nas quais as bases não eram binárias. Nestas situações o SVM é utilizado em conjunto com técnicas de classificação multiclasse a partir de classificadores binários. Dentre as diversas técnicas existentes, foi empregada uma técnica chamada *one-vs-one*. Em um problema com n rótulos, esta técnica consiste em montar $n(n-1)$ estimadores binários, um para cada possível par de rótulos. A classificação de uma observação é feita utilizando

[3] O valor de N é dado pelo número de elementos do problema e o valor de 0.007 foi adotado após ter sido feita uma busca em grid com valores entre 0 e 0.01. Em todos os casos o valor 0.007 obteve o melhor desempenho. A granularidade da busca de valores para λ foi definida para que o parâmetro não fosse tão pequeno, o que levaria a uma dimensão VC maior e consequentemente um pior desempenho fora da amostra [17].

um sistema de votação no qual cada estimador é aplicado e vota a favor de um rótulo. Ao fim a classe com mais votos é atribuída à observação [8].

3.4 Implementação da Abordagem de Integração

As implementações dos algoritmos foram feitas em MATLAB [29] utilizando, em geral, funções padrões do próprio programa com exceção à implementação do NCA, a qual utiliza, na etapa de otimização, o método de gradiente conjugado disponibilizado por [30]. Além disso, por conta da ausência de uma função padrão para o *target encoding*, foi realizada uma implementação própria. O pseudocódigo da implementação é apresentado no Algoritmo 6.

Entrada: Conjunto de observações $X_{tr}^{c} = \{x_{tr_1}^{c}, x_{tr_2}^{c}, \cdots, x_{tr_n}^{c}\}$ e classes $Y_{tr} = \{y_{tr_1}, y_{tr_2}, \cdots, y_{tr_n}\}$ de treino; conjunto de observações $X_{t}^{c} = \{x_{t_1}^{c}, x_{t_2}^{c}, \cdots, x_{t_m}^{c}\}$
Saída: Conjunto de classes $Y_t = \{y_{t_1}, y_{t_2}, \cdots, y_{t_m}\}$
Montar tabela de correspondências de categorias utilizando X_{tr}^{c} e Y_{tr}
Codificar $X_{tr} \xleftarrow{codificação} X_{tr}^{c}$ e $X_t \xleftarrow{codificação} X_t^{c}$
Treinar o classificador utilizando X_{tr} e Y_{tr}
Classificar X_t obtendo $Y_t \xleftarrow{classificação} X_t$
Algoritmo 6: Pseudocódigo da implementação do algoritmo para produzir as integrações entre codificações e classificadores.

3.5 Bases de Dados e Experimentos

Para avaliar o desempenho dos algoritmos apresentados, experimentos foram realizados utilizando bancos de dados artificiais e bancos reais obtidos dos repositórios UCI [31] e KEEL [32]. Os bancos sintéticos foram elaborados utilizando mistura de duas gaussianas com média e desvio padrão aleatórios conforme apresentado na seção 4.1. A Tabela 3.3 apresenta as bases utilizadas em conjunto com a quantidade de atributos categóricos e numéricos, o número de observações e a quantidade de classes distintas em cada conjunto de dados.

Tabela 3.3: Descrição e características dos atributos das bases dos repositórios UCI e KEEL

Bancos de dados UCI e KEEL

Código	Bancos	Atributos		Instâncias	Número de Classes
		Contínuos	Categóricos		
BD1	*Cylinder bands*	19	16	277	2
BD2	*Breast cancer*	-	9	958	2
BD3	*Car evaluation*	-	6	1728	2
BD4	*Contraceptive method choice*	2	7	1473	2
BD5	*Solar Flare*	-	11	1066	6
BD6	*German Credit*	7	13	1000	2
BD7	Mushroom	-	22	5644	2
BD8	Nursery	-	8	12960	2
BD9	*Tic-Tac-Toe Endgame*	-	9	958	2

As bases BD3, BD4 e BD8 apresentavam problema de desbalanceamento de classes. Este é um problema conhecido e um tema de pesquisa ativo com diversas soluções propostas [33], porém não é o alvo desta dissertação. Com isso, foi feita uma modificação nestas bases, transformando-as em problemas binários, sendo um rótulo correspondente à classe originalmente dominante e o outro correspondente à junção de todas as outras classes.

3.5.1 Estimativa Fora da Amostra

Para treinar e avaliar cada um dos modelos, é necessário que haja um conjunto de dados para cada tarefa. Entretanto, como este particionamento dos dados é feito de forma aleatória, o conjunto de teste gerado pode levar a um desempenho muito baixo ou muito alto. Para minimizar essa variabilidade, emprega-se uma técnica chamada de validação cruzada (*cross-validation*) [17]. Esta técnica é usualmente empregada de duas formas:

- **K-Folds:** Este método consiste em dividir o conjunto de dados em K subconjuntos de tamanho $\frac{N}{K}$ para um conjunto de N observações. Desta forma são gerados K modelos, dos quais cada um foi treinado em $K-1$ conjuntos e avaliado no conjunto restante. A estimativa final do erro fora da amostra é dada pela média do erro de cada um dos K modelos [17].
- **Leave-One-Out:** Consiste em executar o mesmo procedimento que o *K-Folds* com parâmetro K igual ao número de observações do conjunto, gerando N modelos. A estimativa final do erro fora da amostra é dada pela média do erro de cada um dos N modelos [17].

Nos experimentos realizados a validação cruzada foi empregada na forma de *K-Folds* com o parâmetro $K = 10$. O propósito dos experimentos foi avaliar os algoritmos NCA, kNN e SVM em conjunto com as três codificações apresentadas: Naive, One-hot e *Target Encoding*. O objetivo não foi estressar cada um dos classificadores, mas sim as codificações, contemplando desde bases categóricas com poucos atributos e poucas categorias, até bases mistas com muitos atributos de alta cardinalidade.

3.5.2 Avaliação dos Resultados

Para que fosse possível comparar estatisticamente o desempenho dos algoritmos nas bases de dados, cada algoritmo foi executado 10 vezes por base, armazenando as acurácias médias obtidas, utilizando validação 10-*Fold*, como realizado em [34]. Para comparar não apenas algoritmos, mas amostras de dados em geral utilizando testes de inferência estatística paramétricos é necessário garantir independência, normalidade e variância constante (homocedasticidade). Não é possível garantir que, ao executar n vezes o experimento de 10 execuções da validação cruzada por base para cada um dos algoritmos, as n amostras de tamanho 10 vão obter o mesmo resultado de variância. Por conta desta impossibilidade, não foi possível utilizar testes de inferência estatística paramétricos para comparação dos algoritmos [35]. Em contrapartida, a família de testes conhecidos como não paramétricos, não exige que as restrições para utilização de testes paramétricos sejam satisfeitas [36].

Ao utilizar estes testes, no escopo deste trabalho, foi necessário definir uma hipótese nula H_0, e uma hipótese alternativa H_1 a ser comparada para cada uma das 9 bases de dados. A hipótese nula aqui definida é a da igualdade das médias de acurácia, de todos os algoritmos integrados, aplicados a cada uma das bases de dados. A hipótese alternativa H_1 assume que ao menos um dos algoritmos integrados possui acurácia média diferente dos demais. Além disso, é necessário definir um parâmetro α responsável por controlar o grau de confiança do resultado. Ao longo dos experimentos, o valor utilizado foi $\alpha = 0.05$, o que garante uma confiança de 95% no resultado dos testes.

Devido à ausência de relacionamento entre as amostras com acurácias para cada integração entre codificação e classificador, foram escolhidos dois testes não paramétricos para comparação entre algoritmos: *Kruskal-Wallis H-Test* [37] e *Conover Test* com correção de *Bonferroni* [38]. Ao longo desta dissertação o teste *Kruskal-Wallis H-Test* será chamado apenas por *Kruskal*-Wallis e o teste *Conover Test* será chamado por *Conover*.

O teste *Kruskal*-Wallis é utilizado para identificar se ao menos um dos algoritmos difere dos outros sendo, porém, ele é incapaz de dizer qual ou quais dos algoritmos são diferentes [36]. Uma vez que este teste não identifica quais algoritmos possuem acurácias diferentes, o teste de *Conover* surge como uma abordagem *post-hoc* para analisar todos os possíveis pares de algoritmos e identificar quais diferem dos outros [38]. A escolha do teste de *Kruskal-Wallis* baseia-se no fato de que as amostras de cada integração entre codificação e classificador não se relacionam. A partir disto, o teste de *Conover* foi escolhido pelo fato de utilizar o mesmo ranqueamento feito pelo teste de *Kruskal-Wallis*.

Ao fazer uso de um teste como o teste de *Conover* para n algoritmos, são feitas $n(n-1)$ comparações. Quanto maior o número de algoritmos diferentes, maior é o número de comparações. Este aumento leva a um crescimento na taxa de erro de tipo I. Erros de tipo I ocorrem quando grupos são iguais e o teste diz que esses grupos são diferentes, isto é, a hipótese nula é rejeitada quando deveria ser aceita. Existem diversas soluções para mitigar este problema, dentre as quais se destaca a correção de *Bonferroni.* Esta correção consiste em recalcular o grau de confiança para analisar a possibilidade de rejeição da hipótese nula. O novo valor de α é dado pela equação 2.7,

$$\alpha = \frac{\alpha}{g}, \tag{3.4}$$

na qual $g = \frac{n\cdot(n-1)}{2}$ é o número de comparações feitas e $n = 9$ é o número de algoritmos distintos abordados.

Capítulo 4

4 Resultados e Discussões

Neste capítulo são apresentados os resultados obtidos a partir das execuções dos algoritmos propostos. Na seção 4.1 é apresentada uma avaliação do desempenho de cada codificação em um conjunto de dados gerados sinteticamente a partir de parâmetros controlados. A seção 4.2 mostra os resultados e discussões acerca da aplicação dos algoritmos apresentados nas bases de dados reais utilizadas.

4.1 Base de Dados Sintética

Para melhor entender e visualizar o desempenho das codificações *Naïve, One-hot* e *Target Encoding* apresentadas no capítulo 3, foram criadas seis bases de dados sintéticas. Estas bases foram geradas a partir de duas gaussianas multivariadas em $\mathbb{R}^2$ variando os valores das matrizes de covariância e médias de acordo com a Tabela 4.1. A partir de cada distribuição foram sorteados 500 pontos, totalizando 1000 observações. Aos pontos gerados pela gaussiana 1 foi atribuído o rótulo 0 e aos pontos gerados pela gaussiana 2, o rótulo 1.

Tabela 4.1: Tabela com os parâmetros utilizados para gerar as duas gaussianas presentes em cada uma das bases sintéticas

	Gaussiana 1	Gaussiana 2
Base 1	$\mu = [-9 \quad -7], \quad \Sigma = \begin{bmatrix} 3 & 3 \\ 3 & 1 \end{bmatrix}$	$\mu = [21.5 \quad 19.5], \quad \Sigma = \begin{bmatrix} 3 & 3 \\ 3 & 1 \end{bmatrix}$
Base 2	$\mu = [-9 \quad -7], \quad \Sigma = \begin{bmatrix} 15 & 15 \\ 3 & 1 \end{bmatrix}$	$\mu = [21.5 \quad 19.5], \quad \Sigma = \begin{bmatrix} 15 & 15 \\ 3 & 1 \end{bmatrix}$
Base 3	$\mu = [-9 \quad -7], \quad \Sigma = \begin{bmatrix} 18 & 18 \\ 3 & 1 \end{bmatrix}$	$\mu = [21.5 \quad 19.5], \quad \Sigma = \begin{bmatrix} 18 & 18 \\ 3 & 1 \end{bmatrix}$
Base 4	$\mu = [-9 \quad -7], \quad \Sigma = \begin{bmatrix} 33 & 33 \\ 3 & 1 \end{bmatrix}$	$\mu = [21.5 \quad 19.5], \quad \Sigma = \begin{bmatrix} 33 & 33 \\ 3 & 1 \end{bmatrix}$
Base 5	$\mu = [-9 \quad -7], \quad \Sigma = \begin{bmatrix} 63 & 63 \\ 3 & 1 \end{bmatrix}$	$\mu = [21.5 \quad 19.5], \quad \Sigma = \begin{bmatrix} 63 & 63 \\ 3 & 1 \end{bmatrix}$
Base 6	$\mu = [-9 \quad -7], \quad \Sigma = \begin{bmatrix} 123 & 123 \\ 3 & 1 \end{bmatrix}$	$\mu = [21.5 \quad 19.5], \quad \Sigma = \begin{bmatrix} 123 & 123 \\ 3 & 1 \end{bmatrix}$

Para que fosse possível aplicar as codificações aos conjuntos de dados foi necessário realizar uma discretização ou categorização dos dados. Por pertencer ao espaço $\mathbb{R}^2$, cada base possui 2 atributos, sendo o primeiro particionado em 5 categorias e o segundo

particionado em 4 categorias. Com a aplicação das codificações *One*-hot e TE, os dados resultantes pertencem a um espaço de dimensão maior que 3, o que não permite a visualização de tais dados. A viabilidade da visualização dos dados após as codificações mencionadas se deu pelo uso de uma técnica de redução de dimensionalidade chamada *Multidimensional Scaling* (MDS) [39]. Tal técnica foi utilizada para projetar os resultados das aplicações das codificações em $\mathbb{R}^2$ e sua escolha se deu pela característica própria de tentar preservar as distâncias entre todos os pares de pontos [40]. A Figura 4.1.mostra os dados gerados pelas gaussianas, as categorizações feitas e aplicação das codificações nos mesmos.

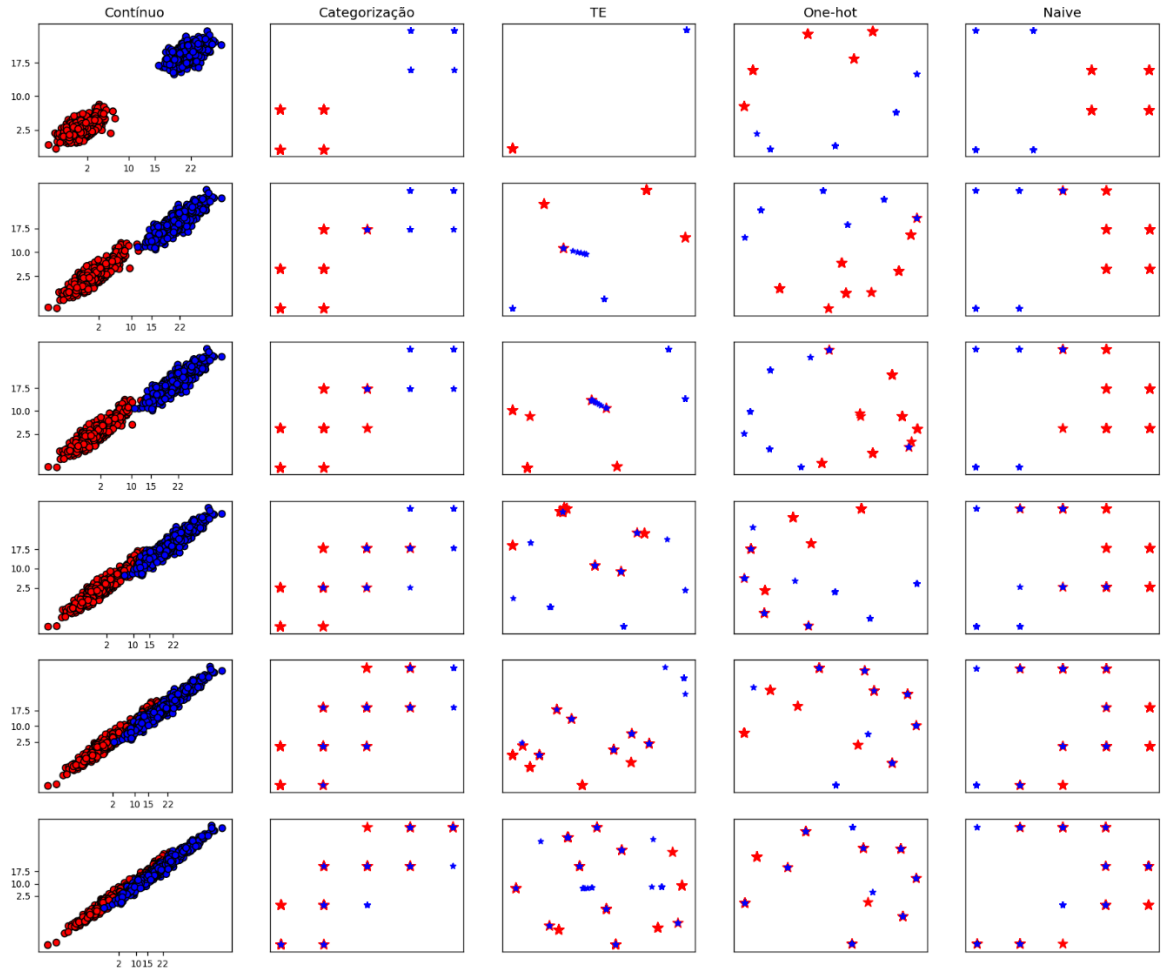

Figura 4.1: Da esquerda para a direita – bases de $\mathbf{1}$ a $\mathbf{6}$ geradas a partir das duas gaussianas com as marcações utilizadas para discretização, categorização das bases, dados após a codificação TE projetados em $\mathbb{R}^2$ utilizando MDS, dados após a codificação *One-Hot* projetados em $\mathbb{R}^2$ utilizando MDS e dados após a codificação *Naive* projetados em $\mathbb{R}^2$ utilizando MDS

Nota-se que nas primeiras 3 bases a codificação TE, quando comparado com as codificações restantes, visualmente se aproxima melhor dos dados categóricos iniciais presentes na segunda coluna da Figura 4.1. Da quarta base em diante, a mistura entre os pontos das gaussianas não permite uma visualização tão clara de alguma semelhança

entre a disposição espacial dos pontos categóricos originais e os pontos resultantes das codificações. Esta análise visual não garante que a codificação TE tenha uma performance superior as demais quando integrada à um classificador. Para validar ou refutar tal afirmação são necessários experimentos estressando as codificações e suas integrações com os classificadores apresentados no Capítulo 2 em bases de dados reais.

4.2 Bases de Dados Reais

O conjunto de algoritmos apresentados foi implementado utilizando os parâmetros informados no capítulo 3. Cada algoritmo foi executado 10 vezes por banco de dados e cada execução foi feita utilizando validação cruzada através de *K-Folds*, com $K = 10$. A Tabela 4.2 apresenta os valores de média e desvio padrão das 10 execuções de cada algoritmo para banco de dados.

Ao observar os valores das acurácias médias, nota-se qual algoritmo se destacou em cada base. Contudo, observando os valores dos desvios padrão e as diferenças entre o algoritmo que se destacou e os algoritmos restantes, não é possível afirmar superioridade no desempenho do destaque. Para avaliar se existem evidências que corroborem diferenças, foi utilizada, em cada base, uma abordagem aplicando os testes estatísticos apresentados no Capítulo 3. Cada base corresponde a uma amostra, sendo cada valor correspondente a acurácia obtida através da média do *K-Fold* em cada uma das 10 execuções dos algoritmos na base em questão.

Desta forma, inicialmente foi feita uma análise exploratória dos algoritmos através de um *boxplot* utilizando os valores das acurácias dos algoritmos aplicados as bases. O *boxplot* é um teste não paramétrico visual, no qual é possível identificar diferenças entre os algoritmos pela ausência de interseções entre caixas [36]. Como mostra a Figura 4.2, os resultados das integrações utilizando as codificações O*ne*-hot e *Naïve* nas bases BD3, BD5, BD7, BD8 e BD9 são visualmente diferentes em relação aos resultados das integrações utilizando a codificação TE. Entretanto, a análise visual não é suficiente para identificar possíveis diferenças entre todos os algoritmos.

A impossibilidade de verificar a existência de diferenças entre todos os algoritmos visualmente por meio do *boxplot* demanda uma análise mais robusta. Esta análise foi feita por meio da aplicação do teste de *Kruskal-Wallis* em cada base. Este teste utilizou como hipótese nula H_0 a igualdade das médias. O objetivo desta análise é verificar se existem evidências estatísticas suficientes com significância de $\alpha = 0.05$ que corroborem para aceitação ou rejeição de H_0. Considerando cada experimento realizado em cada base de dados, se o p-valor obtido for menor que 0.05, então é possível dizer que há evidência estatística suficiente para rejeitar H_0, significando que há uma diferença estatística entre as médias. Caso contrário, a diferença significativa não pode ser rejeitada. A Tabela 4.3 mostra os p-valores obtidos em cada aplicação do teste de *Kruskal-Wallis*.

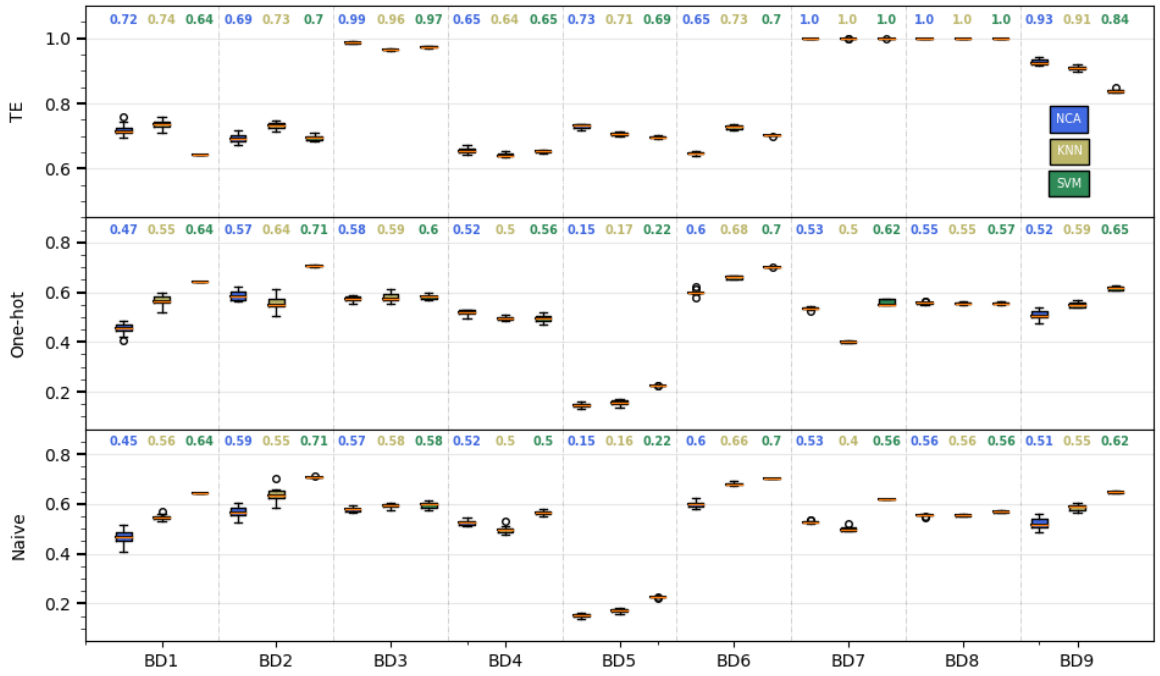

Figura 4.2: Boxplot dos algoritmos por base. Os valores exibidos no topo de cada codificação são as acurácias médias dos algoritmos em cada base de dados

Tabela 4.2: Comparativo entre as acurácias médias por base de dados de cada integração entre codificação e classificador (desvio padrão entre parênteses). Os valores de acurácia variam entre 0 e 1, no qual 1 representa 100% de acerto na base

Bases	NCA			kNN			SVM		
	TE	One-hot	Naïve	TE	One-hot	Naïve	TE	One-hot	Naïve
BD1	0.72 (0.018)	0.469 (0.031)	0.453 (0.022)	**0.735** (0.013)	0.546 (0.011)	0.565 (0.024)	0.643 (0.0)	0.643 (0.0)	0.643 (0.0)
BD2	0.693 (0.014)	0.566 (0.024)	0.588 (0.021)	**0.731** (0.01)	0.638 (0.029)	0.555 (0.028)	0.696 (0.009	0.708 (0.001)	0.705 (0.003)
BD3	**0.987** (0.002)	0.576 (0.009)	0.574 (0.012)	0.965 (0.001)	0.593 (0.008)	0.579 (0.016)	0.973 (0.002	0.596 (0.012)	0.58 (0.008)
BD4	**0.655** (0.009)	0.524 (0.009)	0.518 (0.011)	0.643 (0.005)	0.496 (0.015)	0.496 (0.007)	0.653 (0.004	0.564 (0.009)	0.495 (0.016)
BD5	**0.729** (0.007)	0.152 (0.008)	0.147 (0.009)	0.706 (0.005)	0.173 (0.009)	0.156 (0.01)	0.695 (0.003	0.224 (0.0)	0.224 (0.0)
BD6	0.648 (0.005)	0.599 (0.012)	0.599 (0.01)	**0.728** (0.007)	0.68 (0.006)	0.661 (0.007)	0.702 (0.001	0.7 (0.0)	0.7 (0.0)
BD7	**1.0** (0.0)	0.528 (0.005)	0.534 (0.005)	0.998 (0.0)	0.5 (0.008)	0.399 (0.003)	0.999 (0.0)	0.618 (0.0)	0.557 (0.011)
BD8	1.0 (0.0)	0.554 (0.004)	0.558 (0.004)	1.0 (0.0)	0.554 (0.003)	0.555 (0.005)	1.0 (0.0)	0.568 (0.003)	0.555 (0.004)
BD9	**0.927** (0.008)	0.52 (0.023)	0.508 (0.019)	0.908 (0.007)	0.586 (0.012)	0.55 (0.009)	0.838 (0.005	0.648 (0.004)	0.617 (0.007)

Uma vez rejeitada a hipótese nula, deseja-se encontrar quais algoritmos apresentam diferenças significativas. Para verificar tais diferenças, utilizam-se testes pertencentes a uma classe chamada de testes *post-hoc*. Dentre tais testes foi escolhido

o teste de *Conover.* Este teste executa $\frac{k(k-1)}{2}$ comparações, sendo $k = 9$ o número de algoritmos diferentes. A partir destas comparações constrói-se uma matriz de p-valores entre pares de algoritmos. As comparações em cada base estão dispostas nas tabelas Tabela 4.4, Tabela 4.5, Tabela 4.6, Tabela 4.7, Tabela 4.8, Tabela 4.9, Tabela 4.10, Tabela 4.11, Tabela 4.12.

Tabela 4.3: P-valores obtidos após aplicação do teste de *Kruskal-Wallis* no conjunto de valores dos algoritmos em cada uma das bases

Bases	p-valor
BD1	$2.480896e-15$
BD2	$5.698066e-14$
BD3	$1.839700e-12$
BD4	$5.926912e-14$
BD5	$6.682724e-15$
BD6	$2.011950e-15$
BD7	$1.354794e-15$
BD8	$7.381919e-13$
BD9	$2.294082e-15$

Tabela 4.4: P-valores obtidos pelo teste de *Conover* no conjunto de acurácias obtidas por cada par de algoritmos ao serem aplicados na base BD1. Estão destacados os valores maiores que $\mathbf{0.05}$, indicando os pares de comparações nos quais a hipótese nula $\boldsymbol{H_0}$ não pode ser rejeitada

	TE + kNN	TE + NCA	TE + SVM	Naïve + kNN	Naïve + NCA	Naïve + SVM	One-hot + kNN	One-hot + NCA	One-hot + SVM
TE + kNN	-	**3.62E-01**	6.51E-21	1.03E-37	3.76E-50	6.51E-21	1.42E-40	5.73E-49	6.51E-21
TE + NCA	**3.62E-01**	-	5.78E-16	3.08E-34	1.19E-47	5.78E-16	2.42E-37	2.19E-46	5.78E-16
TE + SVM	6.51E-21	5.78E-16	-	3.01E-16	3.62E-35	**1.00E+00**	1.20E-20	2.01E-33	**1.00E+00**
Naïve + kNN	1.03E-37	3.08E-34	3.01E-16	-	2.81E-17	3.01E-16	**7.78E-01**	7.96E-15	3.01E-16
Naïve + NCA	3.76E-50	1.19E-47	3.62E-35	2.81E-17	-	3.62E-35	1.03E-12	**1.00E+00**	3.62E-35
Naïve + SVM	6.51E-21	5.78E-16	**1.00E+00**	3.01E-16	3.62E-35	-	1.20E-20	2.01E-33	**1.00E+00**
One-hot + kNN	1.42E-40	2.42E-37	1.20E-20	**7.78E-01**	1.03E-12	1.20E-20	-	3.31E-10	1.20E-20
One-hot + NCA	5.73E-49	2.19E-46	2.01E-33	7.96E-15	**1.00E+00**	2.01E-33	3.31E-10	-	2.01E-33
One-hot + SVM	6.51E-21	5.78E-16	**1.00E+00**	3.01E-16	3.62E-35	**1.00E+00**	1.20E-20	2.01E-33	-

Na Tabela 4.4 pode-se ver que com exceção das entradas destacas, todos os p-valores são menores que 0.05 e, portanto, a hipótese nula de igualdade entre as médias de tais algoritmos pode ser rejeitada. Com base nas evidências apresentadas e verificando os resultados dispostos na Tabela 4.2, pode-se afirmar que os algoritmos TE+kNN e TE+NCA possuem médias iguais e superiores aos outros algoritmos.

Na Tabela 4.5, a hipótese nula de igualdade entre o algoritmo de maior média, TE+kNN, perante os outros, foi rejeitada em todas as comparações, comprovando sua superioridade.

Tabela 4.5: P-valores obtidos pelo teste de *Conover* no conjunto de acurácias obtidas por cada par de algoritmos ao serem aplicados na base BD2. Estão destacados os valores maiores que $\mathbf{0.05}$, indicando os pares de comparações nos quais a hipótese nula $\boldsymbol{H_0}$ não pode ser rejeitada

	TE + kNN	TE + NCA	TE + SVM	Naïve + kNN	Naïve + NCA	Naïve + SVM	One-hot + kNN	One-hot + NCA	One-hot + SVM
TE + kNN	-	2.09E-10	6.94E-10	8.71E-30	2.39E-25	1.96E-05	3.49E-19	3.86E-28	3.02E-03
TE + NCA	2.09E-10	-	**1.00E+00**	8.99E-16	2.63E-10	**3.84E-01**	7.48E-04	1.07E-13	6.75E-03
TE + SVM	6.94E-10	**1.00E+00**	-	2.76E-16	7.87E-11	**7.66E-01**	2.71E-04	3.21E-14	1.67E-02
Naïve + kNN	8.71E-30	8.99E-16	2.76E-16	-	**2.44E-01**	1.08E-20	6.14E-07	**1.00E+00**	5.15E-23
Naïve + NCA	2.39E-25	2.63E-10	7.87E-11	**2.44E-01**	-	1.87E-15	2.87E-02	**1.00E+00**	5.97E-18
Naïve + SVM	1.96E-05	**3.84E-01**	**7.66E-01**	1.08E-20	1.87E-15	-	1.33E-08	9.80E-19	**1.00E+00**
One-hot + kNN	3.49E-19	7.48E-04	2.71E-04	6.14E-07	2.87E-02	1.33E-08	-	5.21E-05	3.73E-11
One-hot + NCA	3.86E-28	1.07E-13	3.21E-14	**1.00E+00**	**1.00E+00**	9.80E-19	5.21E-05	-	3.96E-21
One-hot + SVM	3.02E-03	6.75E-03	1.67E-02	5.15E-23	5.97E-18	**1.00E+00**	3.73E-11	3.96E-21	-

A Tabela 4.6 mostra que não é possível rejeitar a hipótese nula de igualdade das médias entre os algoritmos TE+NCA e TE+SVM, TE+kNN e TE+SVM. Logicamente poderia ser dito que por conta destas igualdades, os algoritmos TE+NCA e TE+kNN também apresentam similaridade. Entretanto, o resultado do teste diz que não existem evidências suficientes para tal afirmação, resultando na rejeição de H_0. Em vista disto, ambos algoritmos TE+NCA e TE+SVM são vistos com desempenho significativo sob os demais.

Na Tabela 4.7 pode ser visto que dentre os algoritmos com as maiores médias, 4 deles não tiveram a hipótese de igualdade das médias H_0 rejeitada. São eles TE+NCA, TE+kNN, TE+SVM, One-hot+SVM.

Tabela 4.6: P-valores obtidos pelo teste de *Conover* no conjunto de acurácias obtidas por cada par de algoritmos ao serem aplicados na base BD3. Estão destacados os valores maiores que $\mathbf{0.05}$, indicando os pares de comparações nos quais a hipótese nula $\boldsymbol{H_0}$ não pode ser rejeitada

	TE + kNN	TE + NCA	TE + SVM	Naïve + kNN	Naïve + NCA	Naïve + SVM	One-hot + kNN	One-hot + NCA	One-hot + SVM
TE + kNN	-	1.18E-02	**1.00E+00**	1.00E-09	3.33E-11	6.42E-09	7.06E-03	6.58E-11	1.34E-02
TE + NCA	1.18E-02	-	**1.00E+00**	4.44E-17	1.68E-18	2.76E-16	1.29E-09	3.22E-18	3.01E-09
TE + SVM	**1.00E+00**	**1.00E+00**	-	1.98E-13	6.68E-15	1.29E-12	4.87E-06	1.31E-14	1.07E-05
Naïve + kNN	1.00E-09	4.44E-17	1.98E-13	-	**1.00E+00**	**1.00E+00**	9.75E-03	**1.00E+00**	5.08E-03
Naïve + NCA	3.33E-11	1.68E-18	6.68E-15	**1.00E+00**	-	**1.00E+00**	6.48E-04	**1.00E+00**	3.16E-04
Naïve + SVM	6.42E-09	2.76E-16	1.29E-12	**1.00E+00**	**1.00E+00**	-	3.83E-02	**1.00E+00**	2.08E-02
One-hot + kNN	7.06E-03	1.29E-09	4.87E-06	9.75E-03	6.48E-04	3.83E-02	-	1.14E-03	1.00E+00
One-hot + NCA	6.58E-11	3.22E-18	1.31E-14	1.00E+00	1.00E+00	1.00E+00	1.14E-03	-	5.62E-04
One-hot + SVM	1.34E-02	3.01E-09	1.07E-05	5.08E-03	3.16E-04	2.08E-02	1.00E+00	5.62E-04	-

Tabela 4.7: P-valores obtidos pelo teste de *Conover* no conjunto de acurácias obtidas por cada par de algoritmos ao serem aplicados na base BD4. Estão destacados os valores maiores que $\mathbf{0.05}$, indicando os pares de comparações nos quais a hipótese nula $\boldsymbol{H_0}$ não pode ser rejeitada

	TE + kNN	TE + NCA	TE + SVM	Naïve + kNN	Naïve + NCA	Naïve + SVM	One-hot + kNN	One-hot + NCA	One-hot + SVM
TE + kNN	-	**6.78E-02**	**1.15E-01**	2.05E-19	4.10E-10	2.82E-19	1.03E-18	6.48E-08	**1.55E-01**
TE + NCA	**6.78E-02**	-	**1.00E+00**	4.22E-25	2.01E-16	5.66E-25	1.83E-24	3.17E-14	9.99E-07
TE + SVM	**1.15E-01**	**1.00E+00**	-	8.36E-25	4.39E-16	1.12E-24	3.65E-24	7.02E-14	2.12E-06
Naïve + kNN	2.05E-19	4.22E-25	8.36E-25	-	2.62E-04	**1.00E+00**	**1.00E+00**	2.62E-06	8.82E-14
Naïve + NCA	4.10E-10	2.01E-16	4.39E-16	2.62E-04	-	3.51E-04	1.10E-03	**1.00E+00**	1.32E-04
Naïve + SVM	2.82E-19	5.66E-25	1.12E-24	**1.00E+00**	3.51E-04	-	**1.00E+00**	3.61E-06	1.24E-13
One-hot + kNN	1.03E-18	1.83E-24	3.65E-24	**1.00E+00**	1.10E-03	**1.00E+00**	-	1.27E-05	4.87E-13
One-hot + NCA	6.48E-08	3.17E-14	7.02E-14	2.62E-06	**1.00E+00**	3.61E-06	1.27E-05	-	8.72E-03
One-hot + SVM	**1.55E-01**	9.99E-07	2.12E-06	8.82E-14	1.32E-04	1.24E-13	4.87E-13	8.72E-03	-

Ao observar os p-valores da Tabela 4.8, a hipótese H_0 é rejeitada entre os 3 algoritmos que obtiveram maior média. Dito isto, pode-se afirmar que o algoritmo TE+NCA teve desempenho significativo frente aos demais.

Tabela 4.8: P-valores obtidos pelo teste de *Conover* no conjunto de acurácias obtidas por cada par de algoritmos ao serem aplicados na base BD5. Estão destacados os valores maiores que $\mathbf{0.05}$, indicando os pares de comparações nos quais a hipótese nula $\boldsymbol{H_0}$ não pode ser rejeitada

	TE + kNN	TE + NCA	TE + SVM	Naïve + kNN	Naïve + NCA	Naïve + SVM	One-hot + kNN	One-hot + NCA	One-hot + SVM
TE + kNN	-	1.79E-02	3.59E-02	7.95E-31	2.13E-35	6.40E-12	3.44E-23	8.54E-33	6.40E-12
TE + NCA	1.79E-02	-	1.99E-08	6.42E-36	5.47E-40	6.00E-19	3.75E-29	1.14E-37	6.00E-19
TE + SVM	3.59E-02	1.99E-08	-	1.95E-25	1.47E-30	2.27E-05	6.27E-17	1.21E-27	2.27E-05
Naïve + kNN	7.95E-31	6.42E-36	1.95E-25	-	**6.29E-02**	1.26E-15	3.35E-04	**1.00E+00**	1.26E-15
Naïve + NCA	2.13E-35	5.47E-40	1.47E-30	**6.29E-02**	-	1.11E-21	3.11E-10	**1.00E+00**	1.11E-21
Naïve + SVM	6.40E-12	6.00E-19	2.27E-05	1.26E-15	1.11E-21	-	2.10E-06	3.25E-18	**1.00E+00**
One-hot + kNN	3.44E-23	3.75E-29	6.27E-17	3.35E-04	3.11E-10	2.10E-06	-	1.33E-06	2.10E-06
One-hot + NCA	8.54E-33	1.14E-37	1.21E-27	**1.00E+00**	**1.00E+00**	3.25E-18	1.33E-06	-	3.25E-18
One-hot + SVM	6.40E-12	6.00E-19	2.27E-05	1.26E-15	1.11E-21	**1.00E+00**	2.10E-06	3.25E-18	-

Tabela 4.9: P-valores obtidos pelo teste de *Conover* no conjunto de acurácias obtidas por cada par de algoritmos ao serem aplicados na base BD6. Estão destacados os valores maiores que $\mathbf{0.05}$, indicando os pares de comparações nos quais a hipótese nula $\boldsymbol{H_0}$ não pode ser rejeitada

	TE + kNN	TE + NCA	TE + SVM	Naïve + kNN	Naïve + NCA	Naïve + SVM	One-hot + kNN	One-hot + NCA	One-hot + SVM
TE + kNN	-	1.77E-45	4.36E-06	1.11E-40	1.66E-53	1.78E-20	3.01E-33	3.40E-53	1.89E-19
TE + NCA	1.77E-45	-	6.27E-39	1.43E-03	1.66E-11	1.46E-28	1.74E-14	8.98E-11	2.00E-29
TE + SVM	4.36E-06	6.27E-39	-	3.28E-33	2.73E-48	2.91E-09	3.18E-24	6.25E-48	3.50E-08
Naïve + kNN	1.11E-40	1.43E-03	3.28E-33	-	6.83E-20	3.03E-21	4.87E-06	3.32E-19	3.07E-22
Naïve + NCA	1.66E-53	1.66E-11	2.73E-48	6.83E-20	-	2.64E-40	1.25E-29	**1.00E+00**	5.85E-41
Naïve + SVM	1.78E-20	1.46E-28	2.91E-09	3.03E-21	2.64E-40	-	3.80E-10	7.32E-40	**1.00E+00**
One-hot + kNN	3.01E-33	1.74E-14	3.18E-24	4.87E-06	1.25E-29	3.80E-10	-	4.66E-29	3.03E-11
One-hot + NCA	3.40E-53	8.98E-11	6.25E-48	3.32E-19	**1.00E+00**	7.32E-40	4.66E-29	-	1.59E-40
One-hot + SVM	1.89E-19	2.00E-29	3.50E-08	3.07E-22	5.85E-41	**1.00E+00**	3.03E-11	1.59E-40	-

Na Tabela 4.9, com exceção aos pares de comparações Naive+NCA, One-hot+NCA e Naive+SVM, One-hot+SVM, todas as hipóteses nulas de igualdade das médias podem ser rejeitadas. Logo, pode-se dizer que, assim como na base BD1, o algoritmo TE+kNN obteve desempenho superior aos demais.

Com base nos resultados da Tabela 4.10, a hipótese nula de igualdade é rejeitada para todos dos pares comparados. Contudo, não é possível dizer que um algoritmo foi superior aos demais algoritmos devido a média dos 3 algoritmos com melhor desempenho ser igual. Em consequência disto, afirma-se que na base BD7, dentre os 9 algoritmos apresentados, os algoritmos TE+NCA, TE+kNN e TE+SVM obtiveram os melhores resultados de acurácia frente aos demais.

Tabela 4.10: P-valores obtidos pelo teste de *Conover* no conjunto de acurácias obtidas por cada par de algoritmos ao serem aplicados na base BD7. Estão destacados os valores maiores que $\mathbf{0.05}$, indicando os pares de comparações nos quais a hipótese nula $\boldsymbol{H_0}$ não pode ser rejeitada

	TE + kNN	TE + NCA	TE + SVM	Naïve + kNN	Naïve + NCA	Naïve + SVM	One-hot + kNN	One-hot + NCA	One-hot + SVM
TE + kNN	-	4.87E-21	6.04E-08	3.43E-54	4.29E-34	4.87E-21	3.95E-48	3.31E-39	6.04E-08
TE + NCA	4.87E-21	-	6.04E-08	5.91E-64	1.74E-49	6.56E-41	2.13E-59	5.35E-53	4.47E-32
TE + SVM	6.04E-08	6.04E-08	-	2.13E-59	1.53E-42	4.47E-32	3.43E-54	1.01E-46	4.87E-21
Naïve + kNN	3.43E-54	5.91E-64	2.13E-59	-	5.83E-30	6.56E-41	6.04E-08	1.45E-23	3.95E-48
Naïve + NCA	4.29E-34	1.74E-49	1.53E-42	5.83E-30	-	9.90E-11	2.10E-18	6.17E-03	1.45E-23
Naïve + SVM	4.87E-21	6.56E-41	4.47E-32	6.56E-41	9.90E-11	-	4.47E-32	2.10E-18	6.04E-08
One-hot + kNN	3.95E-48	2.13E-59	3.43E-54	6.04E-08	2.10E-18	4.47E-32	-	9.90E-11	6.56E-41
One-hot + NCA	3.31E-39	5.35E-53	1.01E-46	1.45E-23	6.17E-03	2.10E-18	9.90E-11	-	5.83E-30
One-hot + SVM	6.04E-08	4.47E-32	4.87E-21	3.95E-48	1.45E-23	6.04E-08	6.56E-41	5.83E-30	-

A Tabela 4.11 mostra um comportamento similar ao da Tabela 4.10, todavia a hipótese nula de igualdade entre os algoritmos que obtiveram maior média, TE+NCA, TE+kNN, TE+SVM, não é refutada. Logo, pode-se afirmar que estes 3 algoritmos apresentaram resultados significativamente melhores que os demais.

Por fim, a Tabela 4.12 indica que em somente uma das comparações (One-hot+NCA e Naive+NCA) a hipótese H_o é rejeitada. Neste caso, o algoritmo TE+NCA mostra com significância estatística ser a melhor técnica para classificar padrões em BD9.

Tabela 4.11: P-valores obtidos pelo teste de *Conover* no conjunto de acurácias obtidas por cada par de algoritmos ao serem aplicados na base BD8. Estão destacados os valores maiores que $\mathbf{0.05}$, indicando os pares de comparações nos quais a hipótese nula $\boldsymbol{H_0}$ não pode ser rejeitada

	TE + kNN	TE + NCA	TE + SVM	Naïve + kNN	Naïve + NCA	Naïve + SVM	One-hot + kNN	One-hot + NCA	One-hot + SVM
TE + kNN	-	**1.00E+00**	**1.00E+00**	1.38E-14	2.45E-11	8.70E-15	1.45E-16	1.84E-15	2.92E-03
TE + NCA	**1.00E+00**	-	**1.00E+00**	1.38E-14	2.45E-11	8.70E-15	1.45E-16	1.84E-15	2.92E-03
TE + SVM	**1.00E+00**	**1.00E+00**	-	1.38E-14	2.45E-11	8.70E-15	1.45E-16	1.84E-15	2.92E-03
Naïve + kNN	1.38E-14	1.38E-14	1.38E-14	-	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	1.71E-06
Naïve + NCA	2.45E-11	2.45E-11	2.45E-11	**1.00E+00**	-	**1.00E+00**	**3.33E-01**	**1.00E+00**	1.29E-03
Naïve + SVM	8.70E-15	8.70E-15	8.70E-15	**1.00E+00**	**1.00E+00**	-	**1.00E+00**	**1.00E+00**	1.11E-06
One-hot + kNN	1.45E-16	1.45E-16	1.45E-16	**1.00E+00**	**3.33E-01**	**1.00E+00**	-	**1.00E+00**	2.01E-08
One-hot + NCA	1.84E-15	1.84E-15	1.84E-15	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	-	2.48E-07
One-hot + SVM	2.92E-03	2.92E-03	2.92E-03	1.71E-06	1.29E-03	1.11E-06	2.01E-08	2.48E-07	-

Tabela 4.12: P-valores obtidos pelo teste de *Conover* no conjunto de acurácias obtidas por cada par de algoritmos ao serem aplicados na base BD9. Estão destacados os valores maiores que $\mathbf{0.05}$, indicando os pares de comparações nos quais a hipótese nula $\boldsymbol{H_0}$ não pode ser rejeitada

	TE + kNN	TE + NCA	TE + SVM	Naïve + kNN	Naïve + NCA	Naïve + SVM	One-hot + kNN	One-hot + NCA	One-hot + SVM
TE + kNN	-	2.08E-04	1.16E-04	4.96E-39	4.09E-47	2.23E-23	2.18E-31	7.88E-45	3.74E-14
TE + NCA	2.08E-04	-	5.85E-14	1.44E-44	1.17E-51	3.62E-31	5.49E-38	1.22E-49	3.32E-23
TE + SVM	1.16E-04	5.85E-14	-	2.14E-32	8.76E-42	4.67E-14	2.23E-23	3.75E-39	1.41E-04
Naïve + kNN	4.96E-39	1.44E-44	2.14E-32	-	5.55E-09	1.32E-15	6.88E-06	7.83E-05	1.18E-24
Naïve + NCA	4.09E-47	1.17E-51	8.76E-42	5.55E-09	-	1.57E-28	5.17E-20	**1.00E+00**	1.11E-35
Naïve + SVM	2.23E-23	3.62E-31	4.67E-14	1.32E-15	1.57E-28	-	1.16E-04	6.63E-25	1.41E-04
One-hot + kNN	2.18E-31	5.49E-38	2.23E-23	6.88E-06	5.17E-20	1.16E-04	-	8.49E-16	3.74E-14
One-hot + NCA	7.88E-45	1.22E-49	3.75E-39	7.83E-05	**1.00E+00**	6.63E-25	8.49E-16	-	1.31E-32
One-hot + SVM	3.74E-14	3.32E-23	1.41E-04	1.18E-24	1.11E-35	1.41E-04	3.74E-14	1.31E-32	-

A partir da análise feita dos resultados contidos nas tabelas de comparações par a par, é possível listar quais algoritmos se mostraram significativamente diferentes frente aos demais em cada base. Para algumas bases de dados, não foram obtidas evidências estatísticas suficientes de diferença entre as médias de alguns algoritmos. Assim, a Tabela 4.13 exibe quais algoritmos obtiveram maior desempenho em classificar os padrões de cada base de dados. A codificação *Target Encoding* possui destaque nesta tabela. Em todos os problemas ela esteve presente nas integrações que se mostraram melhor que as demais. A solução utilizando a codificação *Naïve* é amplamente utilizada na literatura [41] [34]. Contudo, foi mostrado que apesar de a codificação *Naïve* ser de comum uso, a codificação TE é significativamente superior.

Tabela 4.13: Resumo da inferência estatística realizada, mostrando base a base, algoritmos que apresentaram resultados significativamente melhores que os demais em cada problema.

Bases de dados	Algoritmos
BD1	TE+kNN, TE+NCA
BD2	TE+kNN
BD3	TE+NCA, TE+SVM
BD4	TE+NCA, TE+kNN, TE+SVM, One-hot+SVM
BD5	TE+NCA
BD6	TE+kNN
BD7	TE+NCA, TE+kNN, TE+SVM
BD8	TE+NCA, TE+kNN, TE+SVM
BD9	TE+NCA

O classificador NCA possui uma característica que o diferencia dos demais, a possibilidade de reduzir a dimensão das observações utilizadas, como abordado no Capítulo 2. Dito isto, o NCA pode ser utilizado em problemas envolvendo tomada de decisão. Neste tipo de problema é necessário fornecer ao tomador de decisão, além do resultado da classificação, evidências que suportem os resultados obtidos pelo classificador. Problemas de tomada de decisão são comuns no meio médico [3]. Nestes problemas a classificação é utilizada como ferramenta de auxílio no diagnóstico de doenças e tratamentos adequados. Nos experimentos realizados tais evidências são obtidas a partir da utilização do NCA para encontrar uma representação aproximada dos resultados também em $\mathbb{R}^2$.

Das 9 bases utilizadas, em 7 o algoritmo TE+NCA esteve presente entre os que se destacaram. Destas 7 bases, em 5 delas o algoritmo TE+NCA teve desempenho equivalente a outros algoritmos. Como foi dito, apesar de outros algoritmos se mostrarem equivalente, o TE+NCA se destaca pelo fato de permitir a visualização de uma projeção dos dados. Esta visualização acrescenta ao resultado a informação, mesmo que aproximada, da disposição espacial das classes.

As 5 bases são BD1, BD3, BD4, BD7, BD8. Nestas bases foi aplicado o algoritmo TE+NCA, utilizando como resultado do NCA a transformação linear A de dimensão $Dx2$, sendo D a dimensão das observações das bases. As imagens obtidas para cada uma das bases mostram a classificação resultante do algoritmo nas observações pertencentes ao conjunto de teste. O conjunto de teste contém 20% do total de observações presentes na base não utilizadas durante o treinamento e escolhidas de forma aleatória. É importante destacar que como o resultado do NCA em $\mathbb{R}^2$ é uma projeção, a acurácia encontrada em $\mathbb{R}^2$ difere da acurácia encontrada em $\mathbb{R}^n$.

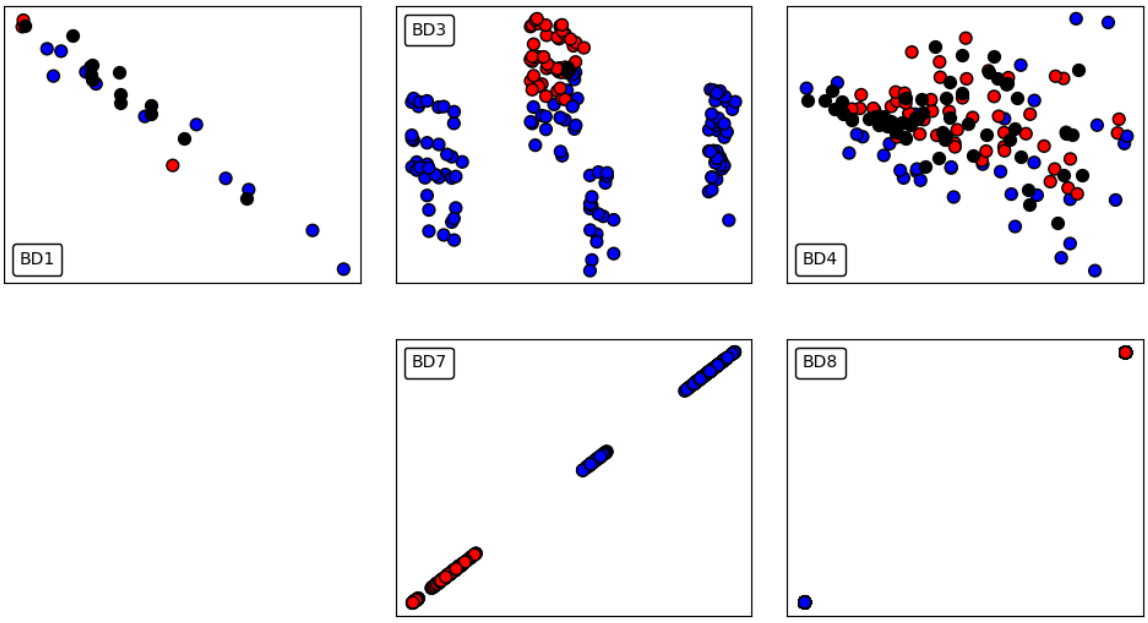

Figura 4.3: Visualização do resultado da projeção em $\mathbb{R}^2$ ao aplicar TE+NCA para as bases BD1, BD3, BD4, BD7, BD8. Os pontos em preto indicam erro na classificação.

Na Figura 4.3 a base BD1 apresenta uma pequena mistura entre as classes 1 e 2, refletindo a acurácia média obtida pelo algoritmo representada na Tabela 4.2. A base BD3 mostra uma separação quase completa entre as classes e também a presença de grupos dentro da mesma classe, refletindo a acurácia média de quase 100% obtida pelo algoritmo representada na Tabela 4.2. Na base BD4 é possível ver uma mistura muito grande entre as classes, refletindo a acurácia média de um pouco mais de 50% obtida pelo algoritmo. Uma acurácia média de 50% ou 0.5 equivale a uma mistura completa entre as classes e a incapacidade do classificador de conseguir distinguir a qual classe pertencem as observações.

Na base BD7 é vista uma separação total entre as classes, refletindo a acurácia média de 100%. A transformação aplicada pelo NCA dispôs os pontos ao longo de uma reta pois para separar 2 classes é necessário somente uma dimensão, i.e., uma reta. De um lado da reta ficam dispostos as observações de uma classe e do outro lado as observações da outra classe. Por fim, a base BD8 mostra uma separação total entre as

classes, refletindo a acurácia média de 100% obtida pelo. A transformação aplicada pelo TE+NCA dispôs os pontos da mesma forma que na base BD7. Nesta base, além das observações estarem dispostas ao longo de uma reta, houve também um agrupamento de todos os pontos em um único ponto para cada classe. Em contraste com o resultado obtido na base BD7, este resultado não é tão descritivo. Apesar de separar completamente as classes, ao agrupar todas as observações em uma mesma posição, a informação da disposição das observações intra-classe é perdida.

Nas bases BD5 e BD9 a integração TE+NCA se mostrou a melhor abordagem dentre os algoritmos apresentados. A Figura 4.4, mostra a projeção dos resultados em $\mathbb{R}^2$, reforçando a vantagem do algoritmo como ferramenta no auxílio à tomada de decisão. Na base BD5 é possível verificar a separação total entre duas classes e uma mistura entre as 6 demais classes. A base BD9 apresenta uma separação entre os dois grupos com um pequeno grau de mistura entre as observações próximas da fronteira que separa as 2 classes.

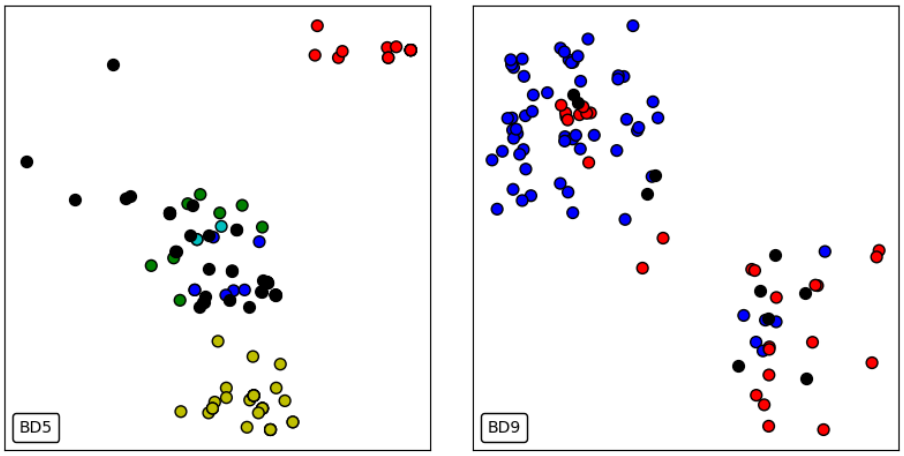

Figura 4.4: Visualização do resultado da projeção em $\mathbb{R}^2$ ao aplicar TE+NCA para as bases BD5, BD9. Os pontos em preto indicam erro na classificação

Capítulo 5

5 Discussões Finais

Este trabalho contribuiu com uma nova abordagem de avaliação de métodos que tem natureza estocástica. Foi proposta e implementada uma arquitetura que integra codificações para atributos categóricos e métodos para classificação de padrões. A partir desta proposta foram produzidos 9 algoritmos capazes de transformar dados categóricos e mistos em dados contínuos e classificá-los. O planejamento experimental realizado foi capaz de mostrar similaridade entre os algoritmos gerados. O uso de testes de inferência estatística garante uma análise mais sofisticada que simplesmente avaliar a média e o desvio padrão.

A partir dos experimentos realizados foi possível verificar que a codificação *Target Encoding,* independente do classificador utilizado, foi a que obteve o melhor desempenho. Em alguns casos como nas bases *Car Evaluation, Mushroom, Nursery* e *Tic-Tac-Toe Endgame* a acurácia média obtida foi quase o dobro da obtida ao utilizar as codificações One-hot e Naive. No caso da base *Solar Flare,* os algoritmos utilizando TE tiveram a acurácia média quase 4 vezes superior aos demais.

Ainda, em todos os casos que o TE foi utilizado em conjunto com NCA ou kNN, houve aumento na acurácia média. Este aumento é explicado pelo fato de ambos classificadores utilizarem a vizinhança dos pontos para a classificação. Em comparação com as codificações One-hot e Naive, o TE incorpora informações como a disposição espacial das categorias durante a codificação. Este fator ajuda classificadores locais como NCA e kNN, pois a estrutura da vizinhança de cada ponto tenta ser preservada ao máximo pela codificação.

Em contrapartida, em alguns casos, preservar a disposição original das categorias no novo espaço não ocasionou uma melhora no SVM, pois pela natureza global do classificador, não há melhora na separabilidade das classes mesmo em um espaço de dimensão infinita tal qual é utilizado para classificar as observações. Tais situações evidenciam uma limitação da integração do SVM com codificações frente as integrações com os classificadores NCA e kNN.

Logo, é possível concluir que dentre os 9 algoritmos apresentados neste trabalho, o algoritmo que se mostra mais competitivo em classificar as bases estudadas foi o TE+NCA, o qual esteve entre os melhores em 7 das 9 bases. Nas 2 bases restantes, o algoritmo com melhor resultado médio foi o TE+kNN. Estes resultados indicam que aplicar

uma codificação que procure preservar a estrutura espacial dos dados em conjunto com um classificador local, como kNN e NCA, leva a melhores resultados. Em comparação, codificações que impõe alguma ordenação não baseada em características dos próprios dados, tendem a obter pior desempenho independente do classificador utilizado conjuntamente. Um outro destaque do TE+NCA é a viabilidade do seu uso em problemas envolvendo tomada de decisão. Esta viabilidade dá-se pela capacidade de projetar os dados em espaços de dimensão menores que o espaço original. Tal característica permite que o tomador de decisão possa visualizar uma projeção dos dados em $\mathbb{R}^3$ e $\mathbb{R}^2$ como informação extra à informação dada pela classificação.

5.1 Trabalhos Futuros

Como investigação futura são propostos os seguintes pontos:

- Verificar se a superioridade dos algoritmos TE+NCA e TE+kNN se mantém em outras bases de dados;
- Avaliar a qualidade da projeção do TE+NCA de $\mathbb{R}^n$ para $\mathbb{R}^d$ com $n > d$;
- Avaliar o porquê da proximidade de desempenho das três codificações apresentadas em alguns casos;
- Procurar entender o comportamento do parâmetro σ que define a largura da função de *kernel* utilizada no NCA;
- Comparar os resultados obtidos com os resultados utilizando o teste *post-hoc* proposto em [42]

Referências Bibliográficas

[1] C. M. Bishop, Pattern Recognition and Machine Learning, New York: Springer, 2006.

[2] Y. LeCun, C. Cortes e C. J. Burges, “THE MNIST DATABASE of handwritten digits,” 1999. [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Acesso em 29 07 2019].

[3] C. Pedreira, E. S. d. C. Sobral, Q. Lecrevise, G. Grigore, R. Fluxa, J. Verde, J. Hernandez, J. van Dongen e A. Orfao, “From big flow cytometry datasets to smart diagnostic strategies: The EuroFlow approach,” *Journal of Immunological Methods,* 2019.

[4] S. Theodoridis e K. Koutroumbas, Pattern Recognition, Elsevier, 2009.

[5] T. Cover e P. Hart, “Nearest Neighbor Pattern Classification,” *IEEE Transactions on Information Theory,* vol. 13, nº 1, pp. 21-27, 1967.

[6] C. Cortes e V. Vapnik, “Support-Vector Networks,” *Machine Learning,* vol. 20, nº 3, pp. 273-297, 1995.

[7] J. Goldberger, S. Roweis, G. Hinton e R. Salakhutdinov, “Neighbourhood Components Analysis,” em *Advances in Neural Information Processing Systems*, 2005.

[8] C. C. Aggarwal, Data Classification: Algorithms and Applications, CRC Press, 2015.

[9] S. Boriah, V. Chandola e V. Kumar, “Similarity Measures for Categorical Data: A Comparative Evaluation,” em *Proceedings of the SIAM International Conference on Data Mining*, Atlanta, 2008.

[10] L. Breiman, J. Friedman, C. J. Stone e R. Olshen, Classification and Regression Trees, CRC Press, 1984.

[11] F. S. Crone, S. Lessmann e R. Stahlbock, “The impact of preprocessing on data mining: An evaluation,” *European Journal of Operational Research,* vol. 173, nº 3, pp. 781-800, 2006.

[12] P. Cohen, S. G. West e L. S. Aiken, Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences, Psychology Press, 1983.

[13] D. Micci-Barreca, “A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems,” em *ACM SIGKDD Explorations Newsletter*, 2001.

[14] K. P. Murphy, Machine Learning: A probabilistic Perspective, London: The MIT Press, 2012.

[15] R. O. Duda, P. E. Hart e D. G. Stork, Pattern Classification, New York: ACM, 2000.

[16] S. Abe, Support Vector Machines for Pattern Classification, 2ª ed., Springer, 2010.

[17] Y. S. Abu-Mostafa, M. Magdon-Ismail e H.-T. Lin, Learning From Data, AMLbook.com, 2012.

[18] W. Yang, K. Wang e W. Zuo, “Neighborhood Component Feature Selection for High-Dimensional Data,” *Journal of Computers,* vol. 7, nº 1, 2012.

[19] M. R. Hestenes e E. Stiefel, “Methods of Conjugate Gradients for Solving Linear Systems,” *Journal of Research of the National Bureau of Standards,* 1952.

[20] J. Solomon, Numerical Algorithms. Methods for Computer Vision, Machine Learning and Graphics, CRC Press, 2015.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay, “scikit-learn,” [Online]. Available: https://scikit-learn.org/stable/auto_examples/neighbors/plot_nca_illustration.html#sphx-glr-auto-examples-neighbors-plot-nca-illustration-py. [Acesso em 06 08 2019].

[22] D. Pyle, Data Preparation for Data Mining, Morgan Kaufmann, 1999.

[23] D. A. Huffman, “The Synthesis of Sequential Switching Circuits,” *Journal of the Franklin Institute,* vol. 257, nº 3, pp. 161-190, 1954.

[24] D. A. Huffman, “The Synthesis of Sequential Switching Circuits,” *Journal of the Franklin Institute,* vol. 257, nº 4, pp. 275-303, 1954.

[25] A. Agresti, An Introduction to Categorical Data Analysis, John Wiley & Sons, Inc., 1996.

[26] D. A. Powers e Y. Xie, Statistical Methods for Categorical Data Analysis, Academic Press, Inc., 1999.

[27] C. C. Aggarwal, Data Mining, Springer, 2015.

[28] C. E. Shannon, “A Mathematical Theory of Communication,” *Bell System Technical Journal,* vol. 27, nº 3, pp. 379-423, 1948.

[29] MATLAB, version 9.4.0.813654 (R2018a), The MathWorks Inc., 2018.

[30] G. E. Hinton, “Training a deep autoencoder or a classifier,” [Online]. Available: http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html. [Acesso em 22 07 2019].

[31] D. Dua e C. Graff, *UCI Machine Learning Repository,* 2017.

[32] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez e F. Herrera, “KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework,” *Journal of Multiple-Valued Logic and Soft Computing,* vol. 17, pp. 255-287, 2011.

[33] V. López, A. Fernández, S. García, V. Palade e F. Herrera, “An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics,” *Information Sciences,* vol. 250, pp. 113-141, 2013.

[34] W. Hu e Y. Tan, “Prototype Generation Using Multiobjective Particle Swarm Optimization for Nearest Neighbor Classification”.*IEEE Transactions on Cybernetic.*

[35] J. Derrac, N. Verbiest, S. García, C. Cornelis e F. Herrera, “On the use of evolutionary feature selection for improving,” *Soft Computing,* vol. 17, pp. 223-238, 2013.

[36] G. W. Corder e D. I. Foreman, Nonparametric Statistics for Non-Statisticians, John Wiley & Sons, Inc, 2009.

[37] W. H. Kruskal e W. A. Wallis, “Use of Ranks in One-Criterion Variance Analysis,” *Journal of the American Statistical Association,* vol. 47, nº 260, pp. 583-621, 1952.

[38] W. J. Conover e R. L. Iman, “On Multiple-Comparisons Procedures,” Technical report, Los Alamos Scientific Laboratory, 1979.

[39] J. B. Kruskal, “Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis,” *Psychometrika,* vol. 29, nº 1, pp. 1-27, 1964.

[40] I. Borg e P. J. F. Groenen, Modern Multidimensional Scaling, Springer-Verlag New York, 2005.

[41] S.-W. Lin, K.-C. Ying, S.-C. Chen e Z.-J. Lee, “Particle swarm optimization for parameter determination and feature selection of support vector machines,” *Expert Systems with Applications,* vol. 35, pp. 1817-1824, 2008.