Universidade Federal de Pernambuco

Programa de Pós-Graduação em Informática (CIn-UFPE)

Relatório do Projeto da Disciplina de Aprendizagem de Máquina

Discentes: Cynthia Moreira Maia e Lucas Benevides Viana de Amorim

Docente: Dr. Francisco de Assis Tenorio de Carvalho

Recife, Março / 2021

RESUMO

Este relatório versa sobre dois experimentos realizados sobre um conjunto de dados sobre imagens de cédulas genuínas e forjadas do UCI. O primeiro experimento envolve a execução do algoritmo de agrupamento *Variable-wise kernel fuzzy c-means* com kernelização da métrica, e o segundo experimento compara os resultados de 6 classificadores: bayesiano gaussiano, bayesiano baseados em k-vizinhos, bayesiano baseado em janela de Parzen, regressão logística, regressão logística com o acréscimo de novos atributos e um ensemble com voto majoritário combinando os 5 classificadores anteriores. Como principais resultados do experimento 1, destacamos: (1) a importância do parâmetro de *fuzzificação* m para o valor dos graus de pertinência, em que para valores próximos a 1, o algoritmo tende a realizar uma partição quase *crisp*, (2) a baixa performance de classificação quando consideradas partições *crisp* versus os rótulos originais dos dados. Já no experimento 2, foi possível verificar significância estatística para as diferenças entre os classificadores, sendo o modelo bayesiano baseado em k-vizinhos aquele com melhor performance apesar da simplicidade da abordagem.

Palavras-chave: Kernel Fuzzy c-means, Agrupamento, Bayesiano, Classificação, Não-Supervisionado, Supervisionado.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O Aprendizado de Máquina (AM) tem como objetivo desenvolver e aplicar técnicas computacionais na construção de sistemas que sejam capazes de adquirir conhecimento de forma automática, aprendendo a induzir uma função ou hipótese, sendo assim capaz de resolver um determinado problema [2]. Há diferentes tipos de aprendizado, como supervisionado, não supervisionado e por reforço. No aprendizado supervisionado, os dados são rotulados, ou seja, no conjunto de treinamento, cada instância de entrada está associada ao seu respectivo valor para o atributo de saída. Já no aprendizado não-supervisionado, os dados não são rotulados, ou seja, o conjunto de treinamento recebe apenas os dados de entrada sem nenhuma informação sobre como deve ser a saída.

O aprendizado não-supervisionado envolve, por exemplo, a tarefa de agrupamento, que objetiva formar grupos de acordo com uma determinada estrutura e que esses grupos sejam homogêneos e bem separados entre si, ou seja, que os objetos pertencentes a um mesmo grupo sejam semelhantes e os objetos de grupos diferentes sejam dissimilares. As estruturas mais populares de agrupamento são: estruturas hierárquicas e estruturas de partição. A hierárquica fornece uma saída conhecida como um dendograma, em que para cada nível tem-se um diferente número de grupos, já a estrutura de partição visa obter um único particionamento dos dados com número fixo de grupos. Há dois principais métodos de agrupamento por particionamento: *hard* e *fuzzy*. No *hard* (ou *crisp*), os agrupamentos são realizados de forma que um objeto pertence ou a um grupo ou a outro, ou seja, a apenas um grupo. Já o método *fuzzy*, um objeto pode pertencer a vários grupos, de acordo com um certo grau de pertinência [1].

Com relação à tarefa de agrupamento, este relatório visa apresentar e discutir os resultados do Experimento 1 que consiste na aplicação do aprendizado não-supervisionado com uso de um algoritmo de agrupamento fuzzy com aplicação de kernel à métrica de dissimilaridade e com ponderação automática das variáveis. A aplicação do kernel à métrica de distância ao invés da transformação do espaço de dados, que é convencionalmente realizada, tem a vantagem de permitir o uso de distâncias adaptativas que mudam a cada iteração do algoritmo, algo que contribui para melhor definição dos pesos das variáveis e melhor performance geral do algoritmo. Vale ressaltar que, neste algoritmo, é levada em consideração a importância das variáveis ao aplicar pesos diferentes a elas, ao contrário

dos algoritmos convencionais de agrupamento que consideram que todas as variáveis são igualmente importantes [1].

No aprendizado supervisionado, tem-se por exemplo as tarefas de classificação e regressão. A tarefa de classificação é uma tarefa importante da AM, com aplicação em diferentes domínios [3]. Este relatório também visa apresentar e discutir os resultados do Experimento 2, que envolve a aplicação dos seguintes algoritmos de classificação: Classificador Bayesiano Gaussiano, Classificador Bayesiano baseados em k-vizinhos, Classificador Bayesiano baseado em janela de Parzen, Classificador por Regressão Logística, Classificador por Regressão Logística com o acréscimo de novos atributos e um Ensemble de todos os modelos anteriores com decisão combinada por voto majoritário.

Para ambos os experimentos, o mesmo conjunto de dados foi utilizado: *Banknote Authentication Data Set* presente no repositório da UCI [4]. Para este conjunto de dados, os autores extraíram atributos de imagens de cédulas genuínas e forjadas com o objetivo de usar os dados para a tarefa de classificação de cédulas entre genuínas ou forjadas [5].

1.1 Objetivos Gerais

1. Avaliar o desempenho do algoritmo de agrupamento *Variable-wise kernel fuzzy c-means* com kernelização da métrica, para o agrupamento de cédulas entre forjadas e genuínas.

2. Avaliar o desempenho de 6 diferentes modelos de classificação para o reconhecimento de cédulas forjadas e genuínas.

1.1.1 Objetivos Específicos

Para atingir os objetivos gerais, foram definidos os seguintes objetivos específicos:

- Estudo teórico dos algoritmos envolvidos nos experimentos;
- Implementação dos algoritmos na linguagem Python;
- Execução dos experimentos com base nos requisitos do exercício;
- Análise e discussão dos resultados.

1.2 Organização do Relatório

Este relatório encontra-se organizado como segue: na **Seção 2** é apresentado a metodologia experimental, na **Seção 3** os resultados e discussões e na **Seção 4** as considerações finais.

2 METODOLOGIA

Nessa seção são apresentados os métodos utilizados para realização dos experimentos. São descritos o ambiente de implementação, conjunto de dados, algoritmos desenvolvidos e as métricas de avaliação. Como para ambos os experimentos do relatório, foi utilizado o mesmo conjunto de dados e o mesmo ambiente de implementação, a seguir, são descritos de maneira geral. Em seguida, serão apresentados, para cada experimento, os métodos individuais que foram utilizados para o desenvolvimento dos algoritmos.

2.1 Ambiente de Implementação

Foi utilizada a linguagem Python, versão 3, para implementação dos algoritmos, utilizando o Google Colab[1] como ambiente de programação, utilizou-se de diferentes bibliotecas, como: Pandas[2], Numpy[3], Scipy[4] e Scikit-learn[5].

2.2 Conjunto de Dados

Foi utilizado o conjunto de dados: *Banknote Authentication Data Set*[6], composto de 1372 instâncias e 5 atributos. Com os 4 atributos de entrada (que denominaremos x_1, x_2, x_3, x_4) do tipo contínuo e o atributo de saída (y) do tipo inteiro.

Conforme descrito no repositório supracitado, os dados foram obtidos a partir da extração automática de atributos de imagens de cédulas genuínas e forjadas. Foi utilizada uma câmera industrial para digitalização das cédulas resultando em imagens de 400x400 pixels em escala de cinza e com uma resolução de cerca de 660 pontos por polegada. A extração de atributos foi feita por meio de uma ferramenta de transformada de Wavelet. Os atributos do conjunto de dados são:

1. x_1 - Variância da imagem após Transformada Wavelet (contínuo);

2. x_2 - Assimetria (skewness) da imagem após Transformada Wavelet (contínuo);

[1]Disponível em: https://colab.research.google.com
[2]Disponível em: https://pandas.pydata.org/
[3]Disponível em: https://numpy.org/
[4]Disponível em: https://www.scipy.org/
[5]Disponível em: https://scikit-learn.org/stable/
[6]Disponível em: https://archive.ics.uci.edu/ml/datasets/banknote+authentication#

3. x_3 - Curtose da imagem após Transformada Wavelet (contínuo);

4. x_4 - Entropia da imagem (contínuo);

5. y - Classe (inteiro: 0 - genuína, 1 - forjada).

2.3 Análise Exploratória dos Dados

Realizou-se uma análise exploratória dos dados, com objetivo de conhecer as características do conjunto de dados e subsidiar decisões nas etapas seguintes. Verificamos se existiam atributos com dados ausentes entre as linhas conjunto de dados e não havia. Por meio de um mapa de calor, apresentado na Figura 1, estudamos a correlação entre os atributos, em que foi constatado que em nenhuma das colunas há alto grau de correlação, não prejudicando a importância dos atributos. Atributos com alto grau de correlação não agregam valor e podem prejudicar a interpretação da importância dos atributos [6]. O mapa apresenta mais cores frias do que cores quentes tendo em vista as baixas correlações entre atributos distintos. Portanto, infere-se que todos os atributos são relevantes para o aprendizado do modelo.

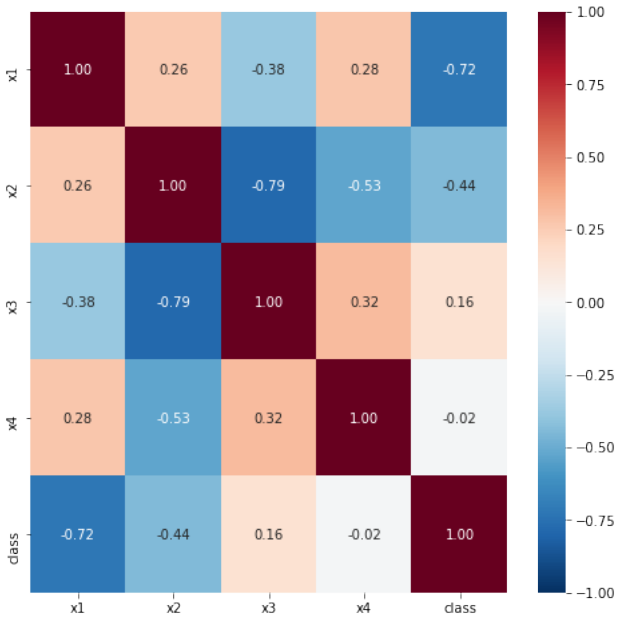


Figura 1: Correlação entre cada variável em um mapa de calor. Fonte: Autores (2021).

2.4 Pré-Processamento dos Dados

Os dados foram normalizados usando a técnica Min-Max, que altera a escala de cada atributo para o intervalo escolhido. Neste caso, para evitar divisão por zero e logaritmo argumento nulo na etapa de geração de novos atributos no Experimento 2, definimos o intervalo com valores mínimo (min = 0,0001) e máximo (max = 1,0001).

2.5 Experimento 1

Este experimento corresponde ao desenvolvimento e execução do algoritmo de agrupamento *Variable-wise kernel fuzzy c-means* com kenelização da métrica [1] com a distância adaptativa local e com a restrição de o produtório dos pesos ser unitário para cada grupo, portanto, trata-se do algoritmo VKFCM-K-LP do artigo citado, a ser detalhado a seguir.

2.5.1 Descrição do algoritmo VKFCM-K-LP

De maneira geral, o algoritmo *Variable-wise kernel fuzzy c-means* com kenelização da métrica funciona de acordo com o pseudo-código mostrado no Algoritmo 1:

Algorithm 1: VKFCM-K-LP

```
Inicialize os parâmetros c, m, T e ϵ;
Inicialize aleatoriamente v, λ, u;
J_{t-1} ← 0;
for t ← 0 to T do
    Atualize v (Eq.2.5);
    Atualize λ (Eq.2.6);
    Atualize u (Eq.2.4);
    Calcule J (Eq.2.1);
    if |J − J_{t-1}| < ϵ then
        Break;
    end
    J_{t-1} ← J ;
end
```

Onde c é o número de grupos, m é o parâmetro de fuzzificação a ser detalhado adiante, T é o número máximo de iterações e ϵ é o limiar que se considera na diferença entre os valores de J, a função objetivo, para a convergência do algoritmo. Para este experimento os parâmetros foram fixados com os seguintes valores: $c = 2$, $T = 150$,

$\epsilon = 10^{-10}$, enquanto que a m foram atribuídos três diferentes valores $\{1.1, 1.6, 2.0\}$, um para cada execução para posterior comparação. O critério de parada do algoritmo é alcançar o valor máximo para T ou atingir diferença entre J e J_{t-1} (de uma iteração para outra) menor que ϵ, quando dizemos que o algoritmo convergiu para um mínimo local. A solução final, depende do ponto de partida, especialmente da inicialização dos centroides. As variáveis λ, u e v serão detalhadas a seguir.

Para a função objetivo, utiliza-se para este algoritmo a expressão da equação 2.1, a qual visa-se minimizar para que o algoritmo produza grupos homogêneos e bem separados:

$$J = \sum_{i=1}^{c} \sum_{k=1}^{n} (u_{ik})^m \varphi^2(x_k, v_i) \tag{2.1}$$

O algoritmo busca uma partição *fuzzy* em dois grupos com um protótipo para cada grupo que minimize a função levando em consideração os pesos de relevância das variáveis. Na equação 2.1, u_{ik} é o grau de pertinência *fuzzy*, com: $(i = 1, ..., c, k = 1, ..., n)$, onde c é o número de clusters, n os exemplos e m é um parâmetro que controla a fuzzificação. Considera-se as seguintes restrições na construção do algoritmo: $u_{ik} \geq 0$ e $\sum_{i=1}^{c} u_{ik} = 1$.

Inicialmente, foi construída uma matriz em que contém o grau de pertinência *fuzzy* u_{ik}, no qual, as linhas correspondem aos exemplos e as colunas correspondem aos grupos. A matriz, foi inicializada de maneira aleatória, gerando os graus de pertinência entre 0 e 1, considerando as restrições destacadas acima. Posteriormente, esta matriz será atualizada conforme equação 2.4 a ser detalhada mais a frente.

Ainda de acordo com a função objetivo na equação 2.1, tem-se o uso da distância adaptativa local $\varphi^2(x_k, v_i)$, com a restrição de que o produto dos pesos das variáveis sejam iguais a um para cada grupo. A distância adaptativa utilizada é definida pela equação 2.2, na qual faz-se uso do kernel gaussiano.

$$\varphi^2(x_k, v_i) = \sum_{j=1}^{p} \lambda_{ij} 2(1 - K(x_{kj}, v_{ij})) \tag{2.2}$$

Na equação 2.2, λ_{ij} corresponde à componente j dos pesos das variáveis dado o centroide do grupo i, de tal forma que para inicializar os pesos de maneira aleatória, criou-se uma matriz com valores unitários, com número de linhas igual ao número de grupos e número de colunas igual ao número de variáveis (atributos do conjunto de dados). Sendo assim, tem-se um peso para cada variável a depender do grupo, em que o produto dos

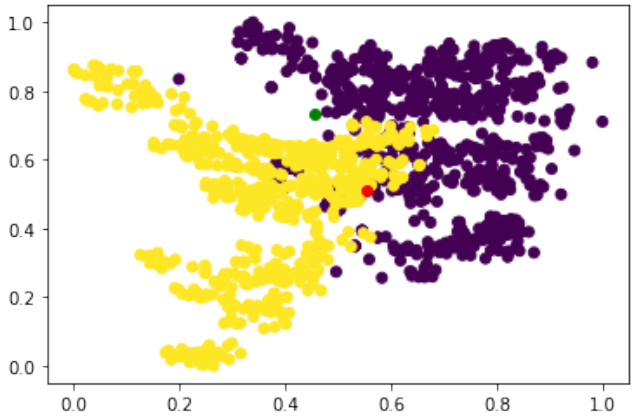
pesos entre as variáveis de um determinado grupo é igual a um, atendendo a restrição. Ao longo das iterações do algoritmo, os pesos são atualizados, de acordo com a função de atualização da equação 2.6. A equação 2.2 apresenta o uso do kernel gaussiano $K_j(x_{kj}, v_{ij})$, definido na equação 2.3.

$$K_j(x_{kj}, v_{ij}) = e^{-\frac{(x_{kj}-v_{ij})^2}{2\sigma_j^2}} \tag{2.3}$$

Onde o termo $2\sigma_j^2$ pode ser estimado como a média dos quantis 0.1 e 0.9 de $(x_{ij} - x_{kj})^2, i \neq k.$, conforme sugerido em [1].

Depois de estimados os valores de σ, foi possível calcular o kernel que aplica a transformação não linear sobre a distância entre os objetos e os protótipos. Os protótipos v_{ij} foram inicializados de maneira aleatória, escolhendo-se instâncias do conjunto de dados e aplicando-se um pequeno *offset* a cada uma de suas componentes. Posteriormente, foram atualizados de acordo com a função de atualização da equação 2.5. Um exemplo de inicialização aleatória dos protótipos é apresentada na Figura 2, considerando as variáveis ($x1$ e $x2$) e as cores definidas pelos rótulos das classes.

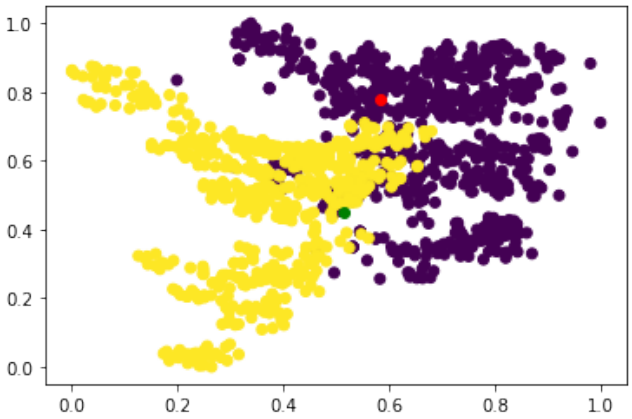
Figura 2: Inicialização aleatória dos protótipos.



Fonte: Autores (2021).

Observa-se que os protótipos estão bem dispersos quando inicializados de forma aleatória e, ao longo das atualizações, nas iterações vão sendo alocados mais próximos ao centros das regiões definidas pelas classes, como ilustrado na Figura 3 que apresenta o posicionamento dos mesmos após algumas iterações.

Figura 3: Protótipos após algumas iterações.



Fonte: Autores (2021).

Após as inicializações aleatórias do grau de pertinência, protótipos e pesos, tem início o laço principal do algoritmo e esses passam a ser atualizados de acordo com suas respectivas funções de atualização. Na equação 2.4 apresenta-se a atualização do grau de pertinência u_{ik} $(i = 1, ..., c, k = 1, ..., n)$, de tal forma que ao ao longo das atualizações da função a seguinte restrição é satisfeita: $\sum_{i=1}^{c} u_{ik} = 1$.

$$u_{ik} = \left[\sum_{h=1}^{c} \left(\frac{\varphi^2(x_k, v_i)}{\varphi^2(x_k, v_h)} \right)^{\frac{1}{m-1}} \right]^{-1} \tag{2.4}$$

Já na equação 2.5 é apresentada a função que foi utilizada para atualização dos protótipos, em que i $(i = 1, ..., c)$ corresponde aos grupos, j $(j = 1, ..., p)$ as variáveis e k $(k = 1, ..., n)$ aos exemplos.

$$v_{ij} = \frac{\sum_{k=1}^{n} (u_{ik})^m K(x_{kj}, v_{ij}) x_{kj}}{\sum_{k=1}^{n} (u_{ik})^m K(x_{kj}, v_{ij})} \tag{2.5}$$

E a equação 2.6 apresenta a função utilizada para atualização dos pesos λ_{ij}, onde i $(i = 1, ..., c)$ são os grupos, j $(j = 1, ..., p)$ corresponde às variáveis. Foi considerada a seguinte restrição $\prod_{j=1}^{p} \lambda_{ij} = 1$, ou seja, o produto dos pesos entre as variáveis de um determinado grupo fosse igual a 1.

$$\lambda_{ij} = \frac{\{\prod_{l=1}^{p} (\sum_{k=1}^{n} (u_{ik})^m 2(1 - K(x_{kl}, v_{il})))\}^{1/p}}{\sum_{k=1}^{n} (u_{ik})^m 2(1 - K(x_{kj}, v_{ij}))} \tag{2.6}$$

2.5.2 Métricas de Avaliação

Como a tarefa é de agrupamento, foram utilizadas algumas métricas para avaliar a qualidade da partição obtida. Foram calculados o Modified Partition Coefficient (MPC), Partition Entropy (PE), Indice de Rand Corrigido, F-measure e o erro de classificação. O MPC foi calculado por meio da equação 2.7, apresentada a seguir:

$$MPC = 1 - \frac{c}{c-1}(1 - PC) \tag{2.7}$$

No MPC, c corresponde ao número de grupos, tem uma faixa de valores entre [0, 1] e é necessário do calculo do coeficiente PC, que foi obtido de acordo com a equação 2.8.

$$PC = \frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{n}(u_{ij})^2 \tag{2.8}$$

Onde, no PC n é o número de exemplos, com faixa de : $[\frac{1}{c}, 1]$. O PE foi obtido a partir da equação 2.9, com a faixa entre $[0, loga(c)]$.

$$PE = -\frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{n}(u_{ij})log_a(u_{ij}) \tag{2.9}$$

A F-Measure foi calculada considerando-se a matriz de contingência da Tabela 1 e segundo a equação Tabela 11. Na Tabela Tabela 1, cada elemento n_{ij} da matriz indica a quantidade de instâncias que foram alocadas ao grupo i e que estão rotuladas no conjunto de dados com a classe j.

Tabela 1: Matriz de contingência para o problema com duas classes e dois grupos.Fonte: Adaptada de Ferreira et. al. 2014 [1].

	Classe 0	Classe 1	Σ
Grupo 0	n_{00}	n_{01}	$n_{0\bullet}$
Grupo 1	n_{10}	n_{11}	$n_{1\bullet}$
Σ	$n_{\bullet 0}$	$n_{\bullet 1}$	

$$\text{F-measure} = 2 \cdot \sum_{j}\frac{n_{\bullet j}}{n}\max_{i}\left[\frac{\frac{n_{ij}}{n_{i\bullet}}\frac{n_{ij}}{n_{\bullet j}}}{\frac{n_{ij}}{n_{i\bullet}} + \frac{n_{ij}}{n_{\bullet j}}}\right] \tag{2.10}$$

Vale ressaltar que, para esta equação, não importa se os grupos estiverem invertidos com relação às classes anotadas nos dados. O erro de classificação é calculado como a razão entre as instâncias em que os grupos não coincidem com as classes e o número total

de instâncias. Já o Índice de Rand Corrigido (ARI) foi calculado com base no pacote de métricas de agrupamento do *sklearn* e, também com base na matriz de contingência da Tabela 1 é definido da seguinte maneira:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{n_{i\bullet}}{2}\sum_j \binom{n_{\bullet j}}{2}\right] \Big/ \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i\bullet}}{2} + \sum_j \binom{n_{\bullet j}}{2}\right] - \left[\sum_i \binom{n_{i\bullet}}{2}\sum_j \binom{n_{\bullet j}}{2}\right] \Big/ \binom{n}{2}} \tag{2.11}$$

A métrica ARI assume valores no intervalo [-1,1], o valor 1 indica concordância perfeita entre as partições que estão sendo comparadas. Frequentemente, o índice é usado para comparar uma partição dada por um processo de agrupamento e outra definida por algum critério externo [7].

2.6 Experimento 2

Este experimento, corresponde ao uso de 6 diferentes modelos de classificação, que são: Classificador Bayesiano Gaussiano, Classificador Bayesiano baseados em k-vizinhos, Classificador Bayesiano baseado em janela de Parzen, Classificador por Regressão Logística, Classificador por Regressão Logística com o acréscimo de novos atributos e um Ensemble de todos os modelos anteriores com decisão combinada por voto majoritário.

2.6.1 Descrição dos modelos utilizados

O primeiro modelo implementado foi o Classificador Bayesiano Gaussiano. Para este modelo, fixou-se a forma paramétrica como uma normal multivariada e usou-se a estimativa de máxima verossimilhança para estimar os parâmetros correspondentes: o vetor de médias e a matriz de variância e covariância, com matriz sigma diagonal. Buscou-se estimá-los em relação a cada classe. Para cada classe, computou-se a probabilidade a *posteriori* que é composta da função de densidade, uma normal multivariada, e a probabilidade *a priori*. Ao final, a decisão é classificar a instância para com a classe cuja probabilidade *a posteriori* é máxima. A probabilidade *a priori* de uma classe foi estimada pela razão entre o número de exemplos da classe e número total de exemplos.

O segundo modelo é o Classificador Bayesiano baseado em k-vizinhos, para o qual, a probabilidade *a posteriori* de uma classe para uma determinada instância é estimada

pela razão entre o número de vizinhos da mesma classe e o número total vizinhos (k). Como métrica de dissimilaridade, utilizamos a distância euclidiana, e o número de vizinhos k foi escolhido como o melhor numa escala variando entre 1 e 30. Para tanto, realizou-se o *tuning* do hiperparâmetro k por meio da técnica de busca em grade (GridSearch) com validação cruzada, implementada no *SciKit-learn* pelo pacote *GridSearchCV*. Os dados de treinamento foram divididos de modo que 20% das instâncias foram utilizados para validação e ajuste do hiperparâmetro. Após definido o parâmetro k ótimo, o modelo foi treinado com a totalidade do conjunto de treinamento. Após estudo minucioso do algoritmo Bayesiano baseado em k-Vizinhos, verificamos que a versão do KNN presente no *Scikit-learn* implementa o mesmo algoritmo bastando configurar os pesos como sendo as distâncias: *KNeighborsClassifier(weights='distance')*. Sendo assim, utilizamos esta implementação.

O terceiro modelo é o Classificador Bayesiano baseado em janela de Parzen, que foi implementado usando a função kernel multivariada produto. Na função kernel multivariada produto, fixou-se o parâmetro h (largura da janela) para todas as dimensões e selecionamos a função kernel gaussiana unidimensional. Assim, estima-se a função de densidade pela função kernel multivariada produto, que quando combinada com a probabilidade *a priori* nos permite obter a probabilidade *a posteriori* para cada classe. Da mesma forma que para o classificador anterior, usamos 20% do conjunto de treino para ajuste do hiperparâmetro, com uma busca em grade para h variando entre os valores do conjunto $\{0.1, 0.2, 0.3, 0.6, 10, 20, 30\}$.

O quarto modelo é o Classificador por Regressão Logística, que foi utilizado para estimar a probabilidade de uma instância pertencer a uma determinada classe, para o qual fizemos uso do modelo presente na biblioteca do *Scikit-learn* utilizando uma regularização L2 para reduzir o sobreajuste.

Já para o quinto classificador, utilizamos o mesmo algoritmo do quarto, entretanto, fizemos o acréscimo de mais quatro atributos: $x_5 = x_1^2$, $x_6 = \sqrt{x_2}$, $x_7 = log(x_3)$, $x_8 = \frac{1}{x_4}$ aos quais também foi aplicada a normalização Min-Max para que a escala ajuste-se ao mesmo intervalo dos demais atributos. Um ponto a ressaltar, é que também consideramos 20% do conjunto de treino para ajuste do parâmetro de regularização C, fazendo novamente uso de uma busca em grade com validação cruzada no seguinte conjunto de valores: $\{0.001, 0.008, 0.1, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5\}$.

Finalmente, o último modelo consiste em um ensemble formado pelos cinco classificadores anteriores, cujas decisões são combinadas por voto majoritário.

2.6.2 Métricas de Avaliação

Para avaliar o desempenho dos classificadores, foi realizada uma validação cruzada estratificada com 10-folds, em seguida foi obtida uma estimativa pontual e um intervalo de confiança para cada métrica de avaliação da performance de classificação: Taxa de Erro, Precisão, Cobertura e F-measure. A estimativa pontual foi obtida considerando a média, e fez-se o cálculo do intervalo de confiança com nível de confiança de 95%. Todas as métricas foram calculadas por meio do pacote *metrics* do *Scikit-learn*. A taxa de erro foi calculada a partir da acurácia, sendo $erro = 1 - \text{acurácia}$. Ao final, os classificadores foram comparados por meio do uso do teste de hipótese não-paramétrico de Friedman.

3 RESULTADOS E DISCUSSÕES

Nessa seção serão apresentados e discutidos os resultados dos experimentos 1 e 2, de acordo com os métodos descritos no capítulo anterior.

3.1 Experimento 1

O algoritmo VKFCM-K-LP foi executado 100 vezes para cada um dos três valores de $m \in \{1.1, 1.6, 2.0\}$ a fim de buscar o modelo com melhor resultado (menor valor para J), entre as 100 execuções, para cada valor de m, e com este modelo calcular as métricas de performance.

Com relação ao valor da função objetivo, a Tabela 2 traz os valores de J obtidos pelo melhor modelo de cada centena de execuções (para cada valor de m). Nota-se que o menor valor de J é obtido para o maior valor de m, o que é consequência principalmente do termo $(u_{ik})^m$ na equação 2.1, pois, já que os graus de pertinência u_{ik} são sempre valores no intervalo $(0, 1)$, o termo será menor com m maior.

A partir deste ponto, dividiremos os resultados e a análise desse experimento em duas subseções, uma que analisará o agrupamento do ponto de vista *fuzzy* e outra do ponto de vista *hard*.

3.1.1 Análise do ponto de vista de agrupamento *fuzzy*

A Figura 4 traz histogramas para os graus de pertinência para os melhores modelos encontrados com cada valor de m para o grupo 0 (omitimos os gráficos para o grupo 1 já que são simétricos a estes).

Nota-se que para $m = 1.1$ os graus de pertinência tendem a se concentrar nos extremos, como seria para uma partição crisp (em que os graus de pertinência são binários,

Tabela 2: Melhores valores da função objetivo J entre as 100 execuções com cada valor de m.

m	J
1.1	1548.297973916541
1.6	1374.3780335872975
2.0	1127.7134873957498

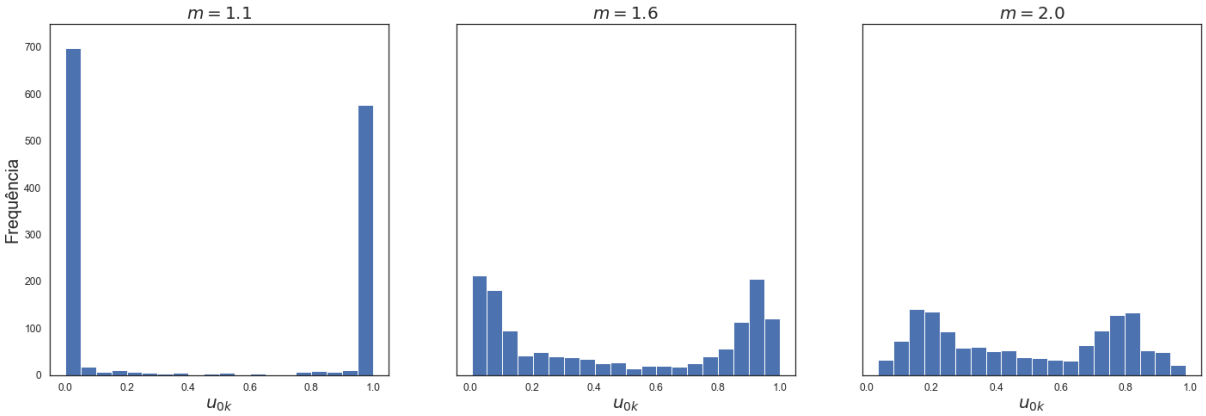


Figura 4: Histogramas para os graus de pertinência dos objetos do grupo 0 para os diferentes valores de m.

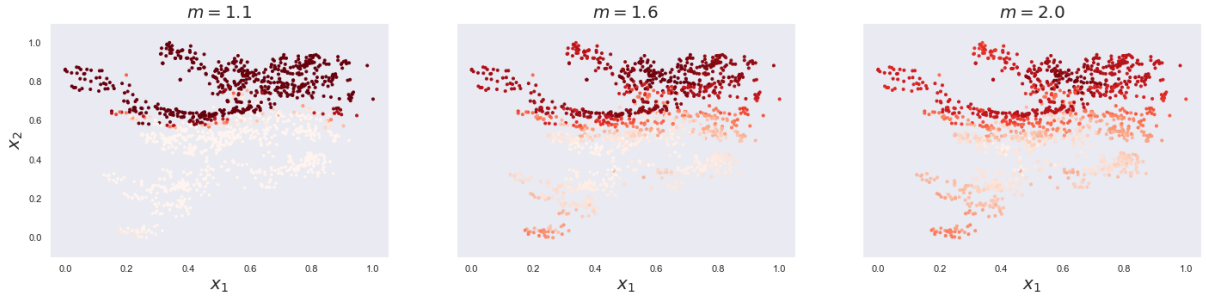


Figura 5: Gráficos de dispersão das instâncias do conjunto de dados, para os diferentes valores de m, com o grau de pertinência ao grupo 0 representado pela escala de cores.

0 ou 1), à medida que aumentamos o valor de m, a frequência é diluída em direção ao centro, ou seja, crescem as barras mais próximas ao centro a medida quem decrescem as barras dos extremos, indicando um agrupamento mais difuso, com graus de pertinência mais próximos da fronteira de decisão entre os grupos. A Figura 5 mostra com clareza esse efeito. Esta figura traz um gráfico de dispersão das instâncias em que os pontos estão numa escala de cores representativa do grau de pertinência ao grupo 0, onde cores mais escuras indicam grau maior. Nota-se que para m=1.1 a partição é quase *crisp* tendo objetos muito escuros e muito claros, ficando mais difusa para os valores seguintes de m, com cada vez mais objetos de cores intermediárias e menos objetos das cores extremas.

Isso ocorre porque o parâmetro de *fuzzificação* m tem influência direta sobre os graus de pertinência dos objetos, uma vez que controla o quão difuso será o agrupamento. Valores de m mais próximos a 1 ocasionam partições mais próximas de um agrupamento *hard* enquanto que valores maiores de m deixam as partições mais difusas. No limite, com $m \to \infty$, os graus de pertinência se concentrariam em 0.5, onde não seria possível

particionar os dados.

A tabela Tabela 3 apresenta os valores das métricas que avaliam a performance do agrupamento *fuzzy*: MPC e PE. Nesta tabela, estão destacados em negrito os melhores valores de cada métrica.

Tabela 3: Valores das métricas de performance para agrupamento *fuzzy* obtidos para cada valor do parâmetro m.

m	**MPC**	**PE**
1.1	**0.9553**	**0.0379**
1.6	0.5737	0.3503
2.0	0.3145	0.5198

Observamos que a métrica MPC é maior quando m é menor, já que, como vemos pelas equações 2.7 e 2.8, graus de pertinência mais extremos tendem a ocasionar um Coeficiente de Partição (PC), e consequentemente um MPC, mais alto, enquanto que gruas de pertinência mais próximos de 0.5 geram PC e MPC mais baixos. Como vimos pelos histogramas da Figura 4 que os graus de pertinência são mais extremos para $m = 1.1$, é para esse valor que se obtém o maior MPC.

Já a métrica PE, que está relacionada com a entropia das partições, notamos que aumenta com m pois as partições mais difusas são as que têm maior desordem, e portanto entropia, na alocação dos objetos aos grupos. Quando se quer grupos com maior separabilidade, busca-se minimizar esta métrica.

3.1.2 Análise do ponto de vista de agrupamento *hard*

Conforme descrito na Seção 2, após a execução do algoritmo para obter o melhor modelo (aquele que minimiza J) para cada valor de m, geramos, em um última etapa, partições *crisp* alocando cada objeto ao grupo para o qual apresenta maior grau de pertinência. Com isso, convertemos o processo de agrupamento *fuzzy* em um agrupamento *hard*, sendo assim, analisaremos os resultados desta etapa com base em métricas próprias para este tipo de agrupamento.

A Tabela 6 traz os resultados das métricas F-measure, Erro de classificação e o Índice de Rand Corrigido. Nesta tabela, estão destacados em negrito os melhores valores de cada métrica. Vê-se que embora a melhor performance, para as três métricas, tenha sido obtida com $m = 1.1$, a diferença entre os valores das métricas considerando-se os

Tabela 4: Protótipos dos dois grupos para os diferentes valores de m.

m	Grupo	x_1	x_2	x_3	x_4
1.1	0	0.5828148	0.7767222	0.1575695	0.5476241
	1	0.5149677	0.4500529	0.3588664	0.7892637
1.6	0	0.5905118	0.7698198	0.1589473	0.5563664
	1	0.5068148	0.4444541	0.3662078	0.7894565
2.0	0	0.5961270	0.7627088	0.1632641	0.5748662
	1	0.5013098	0.4497731	0.3610325	0.7869952

Tabela 5: Número de objetos de cada grupo para os diferentes valores de m.

m	**Grupo**	**Núm. de objetos**
1.1	**0**	618
	1	754
1.6	**0**	633
	1	739
2.0	**0**	648
	1	724

diferentes valores para m é praticamente irrisória. Esta insensibilidade das métricas à mudança de m explica-se pelo fato de que, ao realizar a partição *crisp* atribuindo um objeto k a um grupo i, não faz diferença se o grau de pertinência u_{ik} daquele objeto é muito próximo da fronteira de decisão (ex.: 0.51) ou do extremo (ex.: 0.99), o objeto será atribuído ao grupo da mesma forma.

A Tabela 4 apresenta os protótipos de cada grupo e a Tabela 5 traz o número de objetos em cada grupo. Estes mesmos protótipos e os objetos em cada partição *crisp* (em que atribui-se um objeto a dado grupo quando o grau de pertinência do objeto àquele grupo é maior do que para os demais grupos) estão representados graficamente nos plots par-a-par da Figura 6 à Figura 8. Os objetos de cada grupo estão também representados em tabelas nos arquivos CSV disponíveis junto com o código fonte do experimento.

Notamos que há pouca diferença entre os agrupamentos gerados para os diferentes valores de m, o que ratifica a discussão que fizemos sobre as métricas da Tabela 6.

Entretanto, é importante observar também que as métricas da Tabela 6 indicam uma baixa performance de classificação. Isto pode ser um indicativo de que os atributos extraídos das imagens da cédula podem não ser suficientemente informativos para que as classes sejam reconhecidas por um modelo não supervisionado. Uma outra interpretação é que as classes indicadas nos rótulos presentes no conjunto de dados não coincidem com

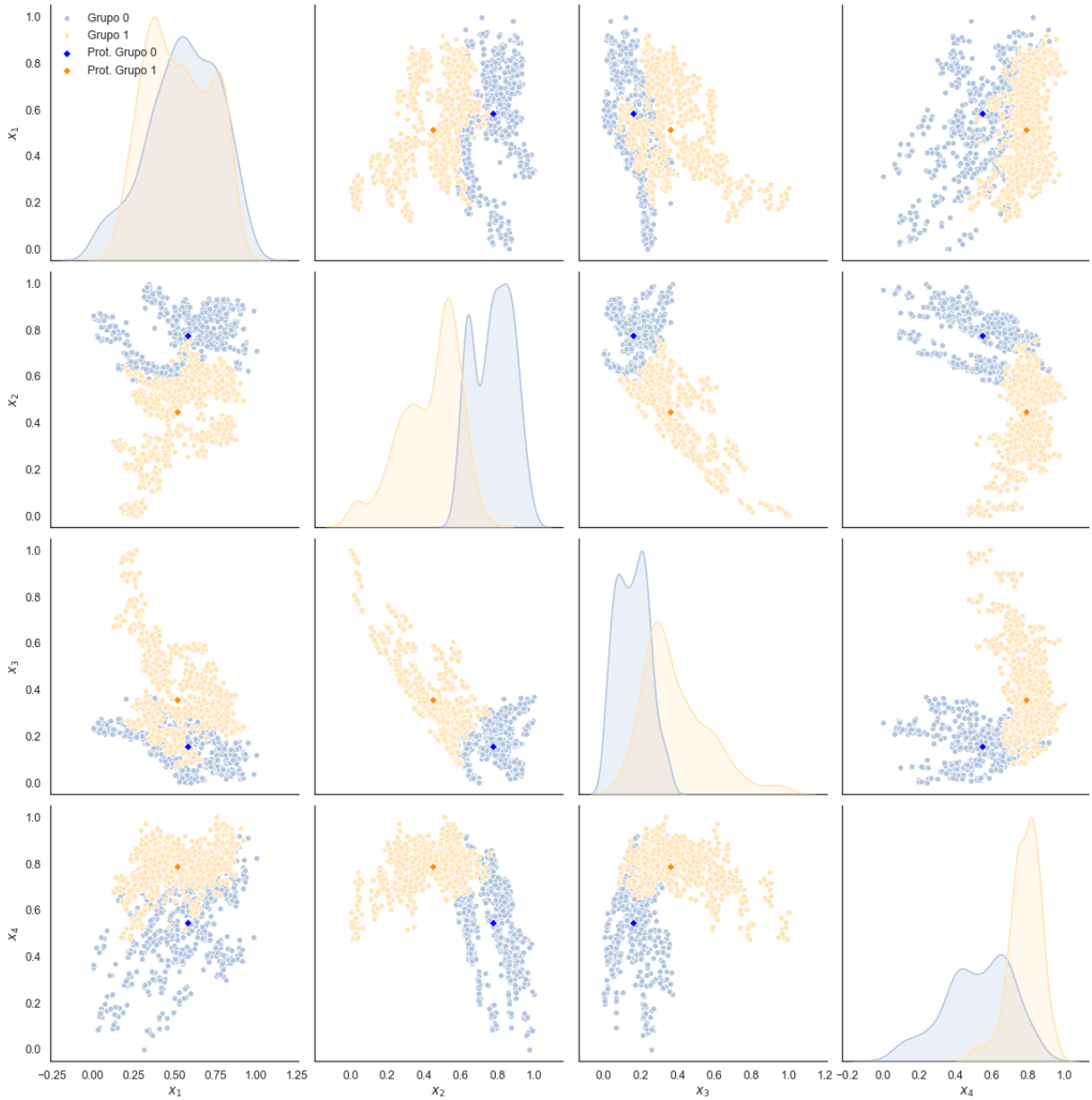


Figura 6: Plot par-a-par para as quatro variáveis do conjunto de dados mostrando em diferentes cores os objetos de cada grupo e seus protótipos, conforme resultado final para $m = 1.1$.

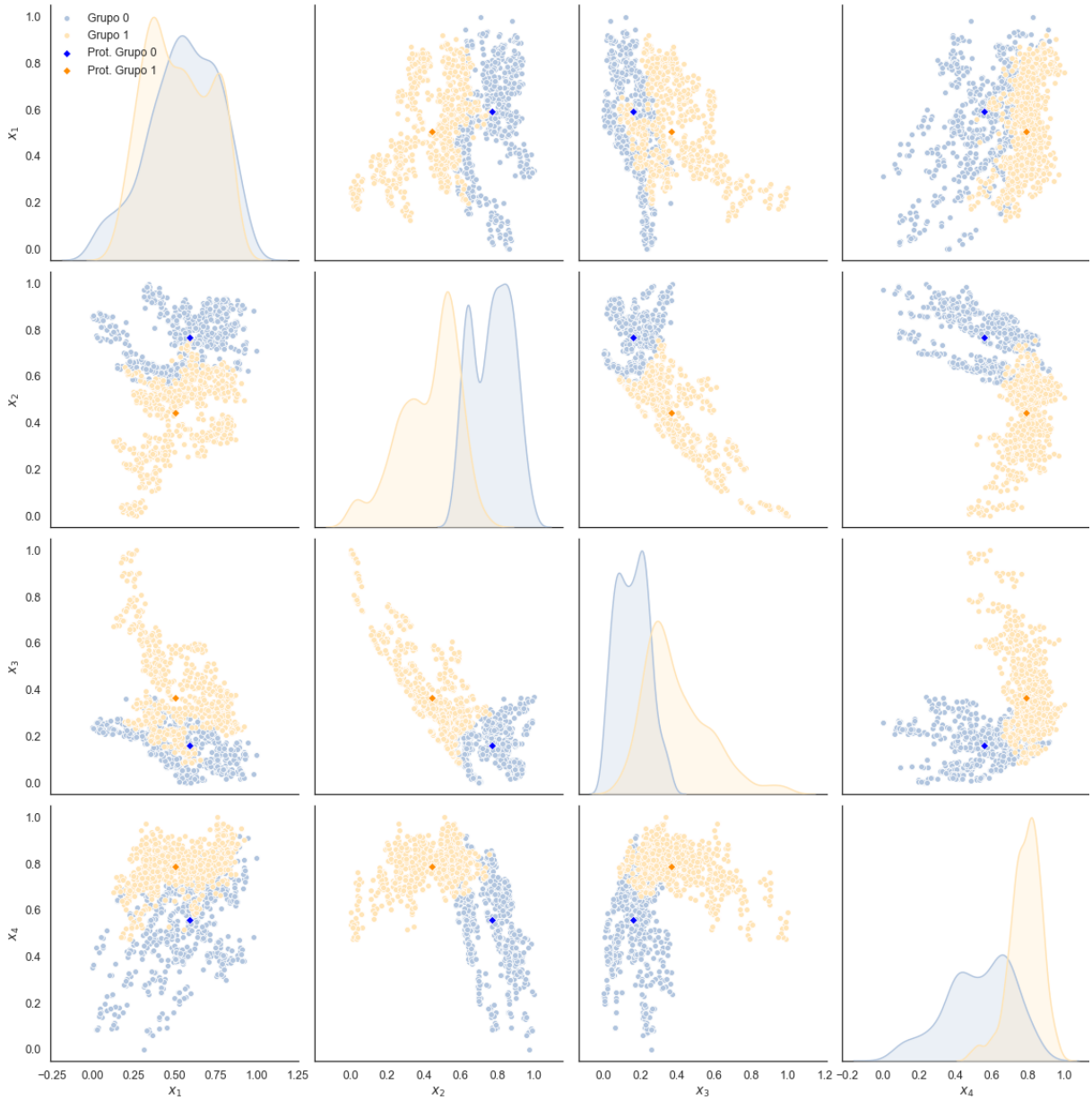


Figura 7: Plot par-a-par para as quatro variáveis do conjunto de dados mostrando em diferentes cores os objetos de cada grupo e seus protótipos, conforme resultado final para $m = 1.6$.

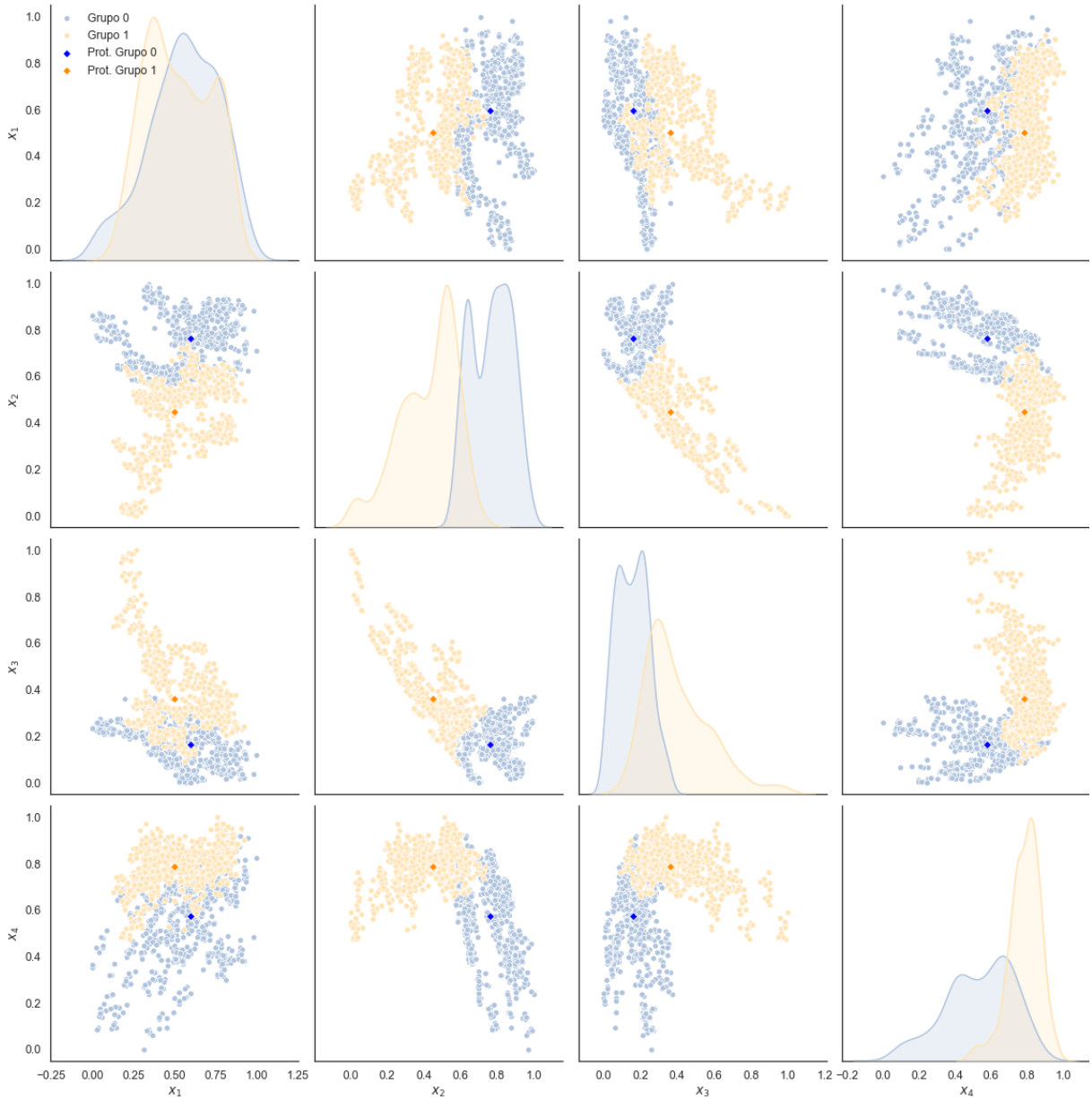
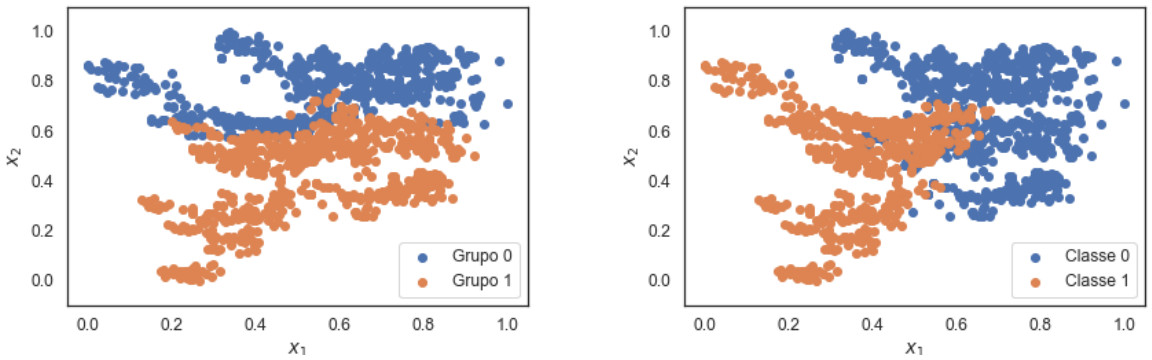


Figura 8: Plot par-a-par para as quatro variáveis do conjunto de dados mostrando em diferentes cores os objetos de cada grupo e seus protótipos, conforme resultado final para $m = 2.0$.



(a) Grupos (b) Classes

Figura 9: Gráfico de dispersão das instâncias em que as cores representam em (a): os grupos resultantes (com $m = 1.1$) e em (b): os rótulos das classes originais dos dados.

o agrupamento mais natural, ou seja, a estrutura latente, dos exemplos no espaço de atributos. Ao comparar a distribuição das classes originais no conjunto de dados com os grupos resultantes do algoritmo, a Figura 9 nos dá uma evidência de que esta última interpretação pode estar correta.

Embora o algoritmo não tenha se demonstrado muito útil para realizar o reconhecimento das classes de maneira não supervisionada, os novos rótulos, oriundos dos grupos resultantes, podem servir como um novo atributo a ser adicionado ao conjunto de dados para posterior análise por um algoritmo de classificação. Este novo atributo pode ter grande contribuição para performance de classificação uma vez que retrata a estrutura através da qual os dados estão organizados no espaço.

Tabela 6: Valores das métricas de performance para agrupamento *hard* obtidos para cada valor do parâmetro m.

m	**F-measure**	**Erro de classificação**	**Índice de Rand Corrigido**
1.1	**0.5833**	**0.4169**	**0.0268**
1.6	0.5816	0.4191	0.0254
2.0	0.5797	0.4213	0.0240

3.2 Experimento 2

Neste experimento, todos os classificadores foram executados com validação cruzada estratificada 10-fold, que mantém em cada *fold* a proporção original entre as classes. Utilizou-se busca em grade com validação cruzada, através do módulo *GridSearchCV* do

Scikit-learn para o ajuste de hiperparâmetros do classificador bayesiano baseado em k vizinhos, janela de Parzen e regressão logística com acréscimo de novos atributos.

Por meio de curvas de aprendizagem, analisamos se os modelos sofreram de sobre ou subajustamento. A curva de aprendizagem é um gráfico do desempenho do classificador no conjunto de treinamento e no teste ou validação, em relação ao tamanho do conjunto de treinamento. Na Figura 10 são apresentadas as curvas de aprendizado dos 5 classificadores: Classificador Bayesiano Gaussiano, Classificador Bayesiano baseado em K-vizinhos, Classificador Bayesiano baseado em Janela de Parzen, Regressão Logística e Regressão Logística com acréscimo de novos atributos. No classificador Bayesiano Gaussiano, Figura 10a, o modelo consegue se ajustar bem aos dados de treino, mas não generaliza bem para dados não vistos, sofrendo de sobreajustamento. À medida que novas instâncias são inseridas, o modelo tende a uma convergência a partir de, aproximadamente, 1000 instâncias. Observa-se que, para o classificador Bayesiano baseado em Janela de Parzen, Figura 10c, o modelo tem dificuldades para se ajustar aos dados de treino, a medida que novas instancias são inseridas e nota-se também uma dificuldade para atingir um bom ajuste em dados não vistos.

Já em relação ao Bayesiano baseado em k-vizinhos, Figura 10b, os desempenhos do modelo nos dados tendem a uma convergência a partir de um número de instâncias aproximadamente igual a 700, indicando, a partir de então um bom ajuste. E nos classificadores, de regressão logística, Figura 10d, e regressão logística com acréscimo de novos atributos, Figura 10e, as curvas caminham juntas desde o início e indicam um bom ajuste.

Na Tabela 7 são apresentados os resultados dos classificadores com a estimativa pontual e o intervalo de confiança para cada métrica de avaliação. Em negrito, são destacados os melhores resultados das métricas de avaliação e o classificador bayesiano baseado em k-vizinhos, em princípio, é o melhor.

Tabela 7: Resultados da estimativa pontual e intervalo de confiança de cada métrica de avaliação dos classificadores

Classificador	**Taxa de Erro (est.pont)**	**Taxa de Erro (int.conf)**	**Precisão (est.pont)**	**Precisão (int.conf)**	**Cobertura (est.pont)**	**Cobertura (int.conf)**	**F-measure (est.pont)**	**F-measure (int.conf)**
Bayesiano Gaussiano	0.1618	(0.1607, 0.1628)	0.8205	(0.8194, 0.8217)	0.8147	(0.8129, 0.8165)	0.8171	(0.8129, 0.8165)
Bayesiano baseado em k-vizinhos	**0.0014**	(0.0013, 0.0015)	**0.9967**	(0.9964, 0.9970)	**1.0**	(1.0, 1.0)	**0.9983**	(0.9982, 0.9985)
Bayesiano baseado em Janela de parzen	0.1589	(0.1572, 0.1606)	0.8408	(0.8394, 0.8422)	0.7918	(0.7886, 0.7949)	0.8144	(0.8122, 0.8166)
Regressão Logística	0.03135	(0.0306 0.0320)	0.9526	(0.9515, 0.9536)	0.9786	(0.9778, 0.9795)	0.9652	(0.9645, 0.9660)
Regressão Logística com acréscimo de atributos	0.0255	(0.0248, 0.0261)	0.9546	(0.9536, 0.9556)	0.9901	(0.9894, 0.9909)	0.9719	(0.9712, 0.9726)
Ensemble com voto majoritário	0.02478	(0.0241, 0.0254)	0.9669	(0.9657, 0.9682)	0.9786	(0.9778, 0.9795)	0.9724	(0.9717, 0.9731)

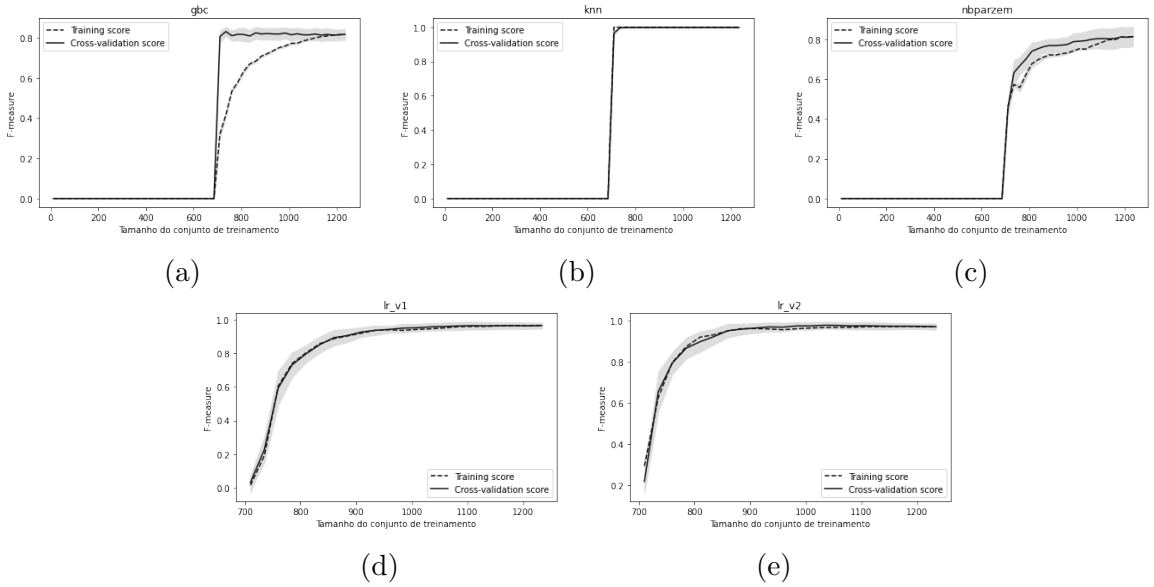


Figura 10: Curvas de aprendizagem com relação à métrica F-measure dos modelos (a): Classificador Bayesiano Gaussiano, (b): Classificador Bayesiano baseado em k-vizinhos, (c): Classificador Bayesiano baseado em janela de Parzen, (d): Regressão logística, (e): Regressão logística com atributos adicionais. Fonte: Autores (2021).

Na Figura 11, um gráfico de barras representa o desempenho dos classificadores em relação às métricas avaliadas do ponto de vista da estimativa pontual. Observa-se que o classificador bayesiano baseado em k-vizinhos, em todas métricas, apresentou resultados superiores quando comparado aos demais classificadores.

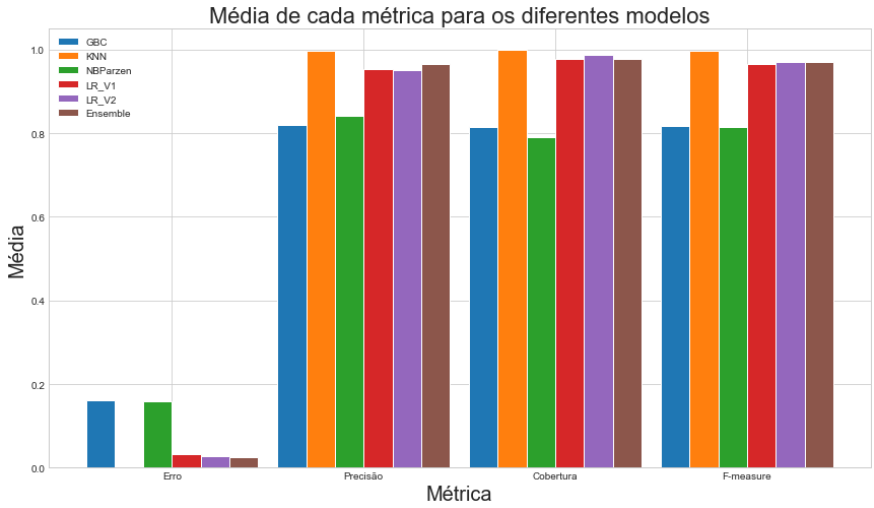


Figura 11: Resultados dos classificadores em relação a cada métrica de avaliação, do ponto de vista da estimativa pontual. Fonte: Autores (2021).

Foi aplicado o teste de Friedman [8] para verificar aferir a significância estatística das diferenças nos desempenhos dos classificadores, para tanto, considerou-se o nível de

significância $\alpha = 0.05$, com as seguintes hipóteses:

H_0 - Não há diferença significativa no desempenho dos classificadores.

H_1 - Há diferença significativa no desempenho dos classificadores.

De acordo com o teste, há uma diferença significativa entre os classificadores, sendo possível a rejeição da hipótese nula H_0. Como rejeitou-se H_0, foi possível aplicar um teste post-hoc de Nemenyi, que permitiu verificar melhor quais são os classificadores com diferença significativa. Este teste foi realizado para cada métrica, e em todos os casos pôde-se concluir que a diferença de performance é significante. Na Tabela 8, é possível visualizar essas diferenças entre os classificadores considerando a métrica da precisão. A partir

Tabela 8: Aplicação do teste Nemenyi em relação a métrica de Precisão.

	Bayesiano Gaussiano	Bayesiano baseado em k-vizinhos	Bayesiano baseado em Janela de parzen	Regressão Logística	Regressão Logística com acréscimo de atributos	Ensemble com voto majoritário
Bayesiano Gaussiano	-	**0.000002**	0.999088	**0.082416**	**0.045757**	**0.007197**
Bayesiano baseado em k-vizinhos	**0.000002**	-	**0.000021**	0.188514	0.285497	0.623670
Bayesiano baseado em Janela de parzen	0.999088	**0.000021**	-	0.209802	0.131828	**0.028587**
Regressão Logística	0.082416	0.188514	0.209802	-	0.999960	0.980599
Regressão Logística com acréscimo de atributos	**0.045757**	0.285497	0.131828	0.999960	-	0.995636
Ensemble com voto majoritário	**0.007197**	0.623670	0.131828	0.980599	0.995636	-

dos resultados do teste, pode-se observar alguns classificadores que são estatisticamente diferentes, no qual, os classificadores são comparados uns com os outros, destacados em negrito considerando o p-value de 0.05. Por exemplo, com p-value de 0.05 o classificador bayesiano baseado em k-vizinhos tem uma diferença significativa com o algoritmo bayesiano gaussiano e com bayesiano baseado em janela de parzen. Já na Tabela 9 é apresentado essas diferenças entre os classificadores em relação a métrica de Taxa de Erro, que continua por exemplo tendo diferenças significativas entre o classificador bayesiano baseado em k-vizinhos e o classificador bayesiano gaussiano e com o baseado em janela de parzen. Na

Tabela 9: Aplicação do teste Nemenyi em relação a métrica de Taxa de Erro.

	Bayesiano Gaussiano	Bayesiano baseado em k-vizinhos	Bayesiano baseado em Janela de parzen	Regressão Logística	Regressão Logística com acréscimo de atributo	Ensemble com voto majoritário
Bayesiano Gaussiano	-	**0.000003**	1.000000	0.146943	**0.027690**	**0.043669**
Bayesiano baseado em k-vizinhos	**0.000003**	-	**0.000004**	0.120177	0.398669	0.313228
Bayesiano baseado em Janela de parzen	1.000000	**0.000004**	-	0.175565	**0.035529**	**0.055075**
Regressão Logística	0.146943	0.120177	0.175565	-	0.993094	0.998157
Regressão Logística com acréscimo de atributo	**0.027690**	0.398669	**0.035529**	0.993094	-	0.999993
Ensemble com voto majoritário	**0.043669**	0.313228	**0.055075**	0.998157	0.999993	-

Tabela 10 é apresentada as diferenças em relação a métrica de Cobertura, no qual, tem-se diferenças entre os classificadores ditos anteriormente e por exemplo, tem-se também entre bayesiano baseado em janela de parzen e ensemble com voto majoritário. E já na Tabela

11 considera-se as comparações entre os classificadores em relação a métrica F-Measure.

Tabela 10: Aplicação do teste Nemenyi em relação a métrica de Cobertura.

	Bayesiano Gaussiano	Bayesiano baseado em k-vizinhos	Bayesiano baseado em Janela de parzen	Regressão Logística	Regressão Logística com acréscimo de atributo	Ensemble com voto majoritário
Bayesiano Gaussiano	-	**0.000097**	0.999999	0.064036	**0.005996**	0.064036
Bayesiano baseado em k-vizinhos	**0.000097**	-	**0.000056**	0.634212	0.955511	0.634212
Bayesiano baseado em Janela de parzen	0.999999	**0.000056**	-	**0.047452**	**0.003983**	**0.047452**
Regressão Logística	**0.064036**	0.634212	**0.047452**	-	0.985249	1.000000
Regressão Logística com acréscimo de atributo	**0.005996**	0.955511	**0.003983**	0.985249	**-**	0.985249
Ensemble com voto majoritário	**0.064036**	0.634212	**0.047452**	1.000000	0.985249	-

Tabela 11: Aplicação do teste Nemenyi em relação a métrica de F-Measure.

	Bayesiano Gaussiano	Bayesiano baseado em k-vizinhos	Bayesiano baseado em Janela de parzen	Regressão Logística	Regressão Logística com acréscimo de atributo	Ensemble com voto majoritário
Bayesiano Gaussiano	-	**0.000003**	1.00000	0.124060	**0.043205**	**0.045446**
Bayesiano baseado em k-vizinhos	**0.000003**	-	**0.000003**	0.147540	0.323422	0.314050
Bayesiano baseado em Janela de parzen	1.000000	**0.000003**	-	0.13452	**0.047782**	0.050218
Regressão Logística	0.124060	0.147540	0.134528	-	0.999154	0.999318
Regressão Logística com acréscimo de atributo	**0.043205**	0.323422	**0.047782**	0.999154	**-**	1.000000
Ensemble com voto majoritário	**0.045446**	0.314050	**0.050218**	0.999318	1.000000	-

4 CONCLUSÃO

Neste relatório foi apresentado um estudo que aplica aprendizagem de máquina para análise de um conjunto de dados relativo a imagens de cédulas categorizadas entre genuínas e forjadas. O estudo foi dividido em dois experimentos: No experimento 1, aplicamos aprendizado não-supervisionado, por meio de um algoritmo de agrupamento *fuzzy* com kernelização da métrica de distância, ponderação automática das variáveis e distância adaptativa, VKFCM-LP, que integra o estado da arte na área. Já no experimento 2, aplicamos aprendizado supervisionado, efetuando uma tarefa de classificação com a comparação de 6 diferentes de algoritmos de classificação.

No experimento 1, ficou patente a importância do parâmetro m para definir o quão difuso será o particionamento dos dados. Mostramos que para o valor $m = 1.1$ o algoritmo aproxima-se de uma agrupamento *hard*, enquanto que para $m = 1.6$ ou mesmo 2.0, aumenta-se a entropia e fronteira de decisão entre os grupos passa a ser mais difusa.

Ainda no experimento 1, constatamos que o parâmetro m tem pouca influência na forma dos grupos quando tratamos o problema pela abordagem *hard*, a qual demonstrou um baixo desempenho no que diz respeito ao reconhecimento das classes originais nos dados, conforme registrado pelas métricas F-measure, Erro de classificação e Índice de Rand Corrigido. Ressaltamos que, apesar do desempenho ruim para classificar os dados quando compara-se os grupos com os rótulos originais dos dados, a saída do algoritmo de agrupamento pode ser de importante valia como um atributo extra para servir de entrada a um algoritmo de classificação uma vez que tende a representar os agrupamentos que são mais naturais para os dados apresentados.

Já em relação ao experimento 2, o classificador bayesiano baseado em k-vizinhos apresentou um melhor desempenho comparado aos demais classificadores, considerando as quatro métricas (Taxa de Erro, Precisao, Cobertura e F-measure). A partir do teste de hipótese de Friedman, foi possível rejeitar H_0, com nível de significância: $\alpha = 0.05$, prevalecendo então a hipótese alternativa de que existe diferença significativa entre os classificadores. Por fim, o teste de Nemenyi permitiu comparar os classificadores uns com os outros, em que destacamos uma diferença mais significativa entre os classificadores bayesiano gaussiano e bayesiano baseado em k-vizinhos, e também entre a regressão logística e o gaussiano.

REFERÊNCIAS

[1] FERREIRA, M. R.; CARVALHO, F. D. A. D. Kernel fuzzy c-means with automatic variable weighting. *Fuzzy Sets and Systems*, Elsevier, v. 237, p. 1–46, 2014.

[2] FACELI, K. et al. Inteligência artificial: Uma abordagem de aprendizado de máquina. *Rio de Janeiro: LTC*, v. 2, p. 192, 2011.

[3] FREITAS, A.; CARVALHO, A. A tutorial on hierarchical classification with applications in bioinformatics. *Research and trends in data mining technologies and applications*, IGI Global, p. 175–208, 2007.

[4] DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2017. Available at: <http://archive.ics.uci.edu/ml>.

[5] LOHWEG, V. et al. Banknote authentication with mobile devices. In: ALATTAR, A. M.; MEMON, N. D.; HEITZENRATER, C. D. (Ed.). *Media Watermarking, Security, and Forensics 2013*. SPIE, 2013. v. 8665, p. 47 – 60. Available at: <https://doi.org/10.1117/12.2001444>.

[6] HARRISON, M. Machine learning – guia de referência rápida: Trabalhando com dados estruturados em python. *Novatec Editora*, v. 1, p. 272, 2019.

[7] SANTOS, J. M.; EMBRECHTS, M. On the use of the adjusted rand index as a metric for evaluating supervised classification. In: ALIPPI, C. et al. (Ed.). *Artificial Neural Networks – ICANN 2009*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 175–184. ISBN 978-3-642-04277-5.

[8] DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, JMLR. org, v. 7, p. 1–30, 2006.