DOI: 10.48011/asba.v2i1.1504

Uma Proposta de Correção de Dados Ruidosos via K-Nearest Neighbours

Victor M. Magalhães Pinto * **Cristiano L. de Castro** **

* *Programa de Pós-Graduação em Engenharia Elétrica - Universidade Federal de Minas Gerais - Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brasil (e-mail: victormmp@ufmg.br).*
** *Programa de Pós-Graduação em Engenharia Elétrica - Universidade Federal de Minas Gerais - Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brasil (e-mail: crislcastro@ufmg.br)*

Abstract: Noisy data sets contribute to the increase of classification model complexity leading to decision boundaries with larger complexity than the one required by the problem. This situation implies bad performance when applied to the real distribution (data generator function). This paper proposes a noise filtering method through the use of a weighted k-nearest neighbor algorithm, whose weights are proportional to the distance between the noisy sample and its nearest neighbors. Experiments conducted with synthetic and real data show the effectiveness of the proposed method in binary classification problems.

Resumo: Conjuntos de dados ruidosos contribuem para o aumento da complexidade de modelos de classificação resultando em superfícies de decisão com complexidade superior à requerida pelo problema. Tal situação leva a um pior desempenho ao ser aplicado à distribuição real (função geradora) dos dados. Este artigo propõem um método de filtragem de ruído a partir da utilização do algoritmo *k-nearest neighbours* ponderado, cujos pesos são proporcionais à distância da amostra ruidosa sob análise e seus vizinhos mais próximos. Experimentos conduzidos com dados sintéticos e reais mostram a eficácia do método proposto em problemas binários de classificação.

Keywords: knn, machine learning; label errors; filter; classifier; noise.

Palavras-chaves: knn; machine learning; erros de rotulação; filtro; classficadores; ruído.

1. INTRODUÇÃO

A presença de ruído de rotulação em conjuntos de dados representa um desafio no desenvolvimento de modelos, especialmente em problemas de classificação. Segundo Li and Abu-Mustafa (2006), conjuntos de dados ruidosos contribuem para o aumento da complexidade de modelos resultando geralmente em superfícies de decisão com complexidade superior à requerida pelo problema. Tal situação leva a um pior desempenho ao ser aplicado à distribuição real (função geradora) dos dados.

Durante o processo de obtenção da amostra de treinamento, a definição dos rótulos nem sempre é perfeita, e erros podem inserir ruído no conjunto de treinamento. No artigo de Garcia et al. (2015), os autores demonstram a relação entre diferentes níveis de ruído de rotulação existentes em conjuntos de dados e a complexidade resultante dos mesmos, observada através de diferentes indicadores de complexidade.

Existem na literatura algumas soluções para tratamento de ruído no conjunto de dados, como aquelas propostas em Wilson and Martinez (2000) e Tomek (1976), baseadas na distância das amostras no espaço de características. O trabalho de Garcia et al. (2015) faz uma análise comparativa desses métodos, avaliando-os em termos do *F-score* de classificação, juntamente com o método proposto em Wilson (1972). Este último utiliza uma técnica chamada de *Edited Nearest Neighbours - ENN*, removendo amostras se os rótulos de seus k vizinhos são diferentes. Neste comparativo, são empregadas variações do ENN, como *Repeated ENN*, onde repetidas iterações do ENN são aplicadas, e *All-kNN*, onde valores crescentes de k são utilizados. O filtro proposto por Garcia et al. (2015) utiliza duas métricas para medir a complexidade de conjuntos de dados, que são usadas para descobrir as amostras possivelmente ruidosas. Alguns outros trabalhos também utilizam a técnica ENN para melhoria de desempenho de modelos em diversas condições, como Alejo et al. (2010), ou com objetivos diversos, como o pacote desenvolvido na biblioteca *imbalanced-learn* (Lemaître et al., 2017) para subamostragem.

Neste artigo, é proposta uma variação do algoritmo ENN, a partir da utilização de um algoritmo *k-nearest neighbours* que usa pesos relacionados à distância da amostra sob análise com seus vizinhos, numa estratégia de *leave-one-out*. O objetivo é corrigir eventuais erros de rotulação, diminuindo a variância dos dados tanto para a melhora do desempenho do classificador quanto da complexidade da superfície de separação.

Para avaliar o método proposto dois experimentos foram realizados. O primeiro utiliza uma Rede Neural de Múltiplas Camadas (MLP) superdimensionada (Haykin, 2001), cujo

espaço de hipóteses varia desde complexidades baixas a mais altas. Neste experimento controlado, dados sintéticos são amostrados de uma função geradora, de forma que rótulos ruidosos são adicionados apenas no conjunto de treinamento, mas não no conjunto de teste. Como será visto adiante, a remoção de ruído proporcionada pelo método aqui proposto tem um efeito de redução de complexidade da rede, fazendo com que o treinamento selecione uma hipótese com complexidade mais adequada ao problema em questão.

O segundo experimento representa um estudo de caso sobre uma base de dados real, um problema de classificação de pontos de reversão em séries temporais do mercado de energia elétrica. Neste problema, o procedimento de rotulação automática para assinalar pontos de reversão (classe positiva), a serem aprendidos pelo modelo, é naturalmente ruidoso. Nesse caso, o método proposto no artigo é aplicado como estratégia de preprocessamento para redução do ruído e melhoria da capacidade de aprendizado de modelo XGBoost (Chen and Guestrin, 2016). Para ambos os experimentos, testes de significância estatística foram aplicados para analisar a eficácia da metodologia.

2. MÉTODO PROPOSTO

2.1 Ilustração do Problema de Rótulos Ruidosos

Considere um conjunto de amostras reais como as da figura 1. Assumindo que no processo de rotulação desses dados possa ter havido erros na atribuição dos mesmos - e portanto a inserção de ruído no *dataset* - um conjunto coletado como o da figura 2 pode ser o obtido ao final deste processo.

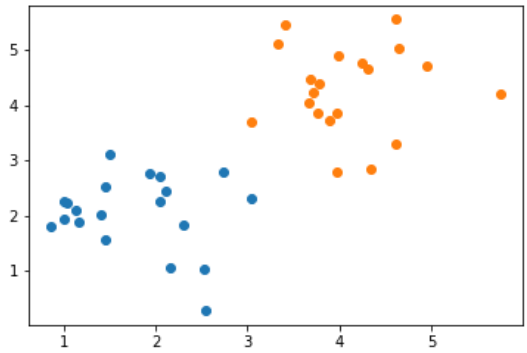

Figura 1. Exemplo de distribuição real.

Este resultado é especialmente problemático em regiões de separação entre classes com alta proximidade em suas componentes, conforme ilustrado pela figura 3, que devido a erros na atribuição dos rótulos, gera um conjunto de treinamento como o da figura 4, onde a fronteira de separação se torna mais complexa e pouco definida.

Para avaliar a consequência deste efeito para um classificador, utilizamos uma rede neural MLP como base. Neste contexto, é proposta a aplicação do algoritmo de *K-Nearest Neighbours* com pesos calculados à partir das distâncias entre as amostras, como uma etapa de pré-processamento

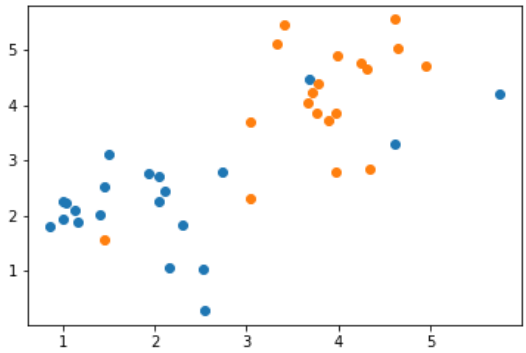

Figura 2. Exemplo de erros na distribuição real por rotularização.

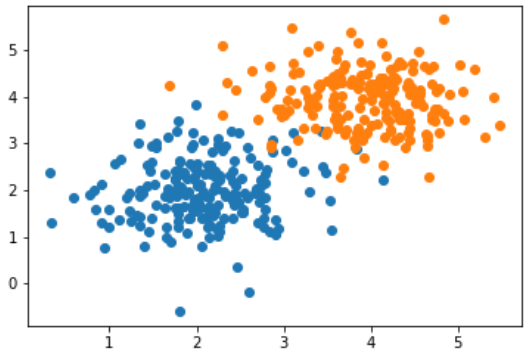

Figura 3. Exemplo de distribuição real de dados.

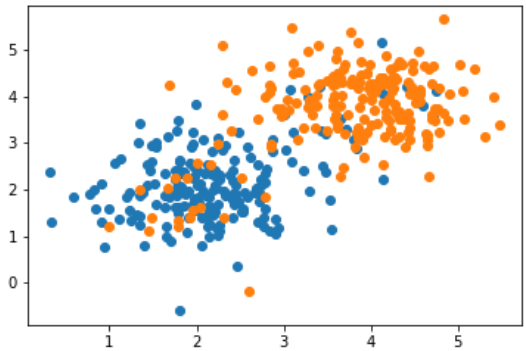

Figura 4. Distribuição com fronteiras pouco definidas por erros de rotulação.

dos dados, a fim de reatribuir rótulos a cada observação a partir da avaliação dos rótulos de amostras vizinhas, como na figura 5. Caso o ruído inserido não seja grande a ponto de tornar indistinguível as distribuições, espera-se, além de diminuir a variância das classes, aprimorar o desempenho da rede mesmo em situação de sobreajuste.

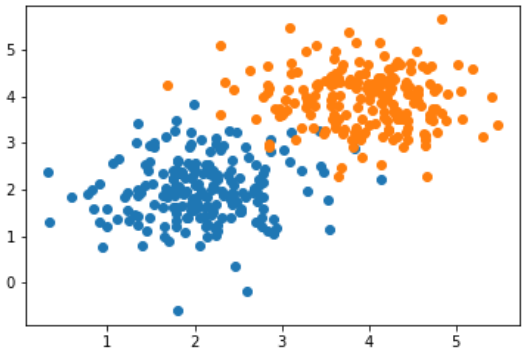

Figura 5. Rótulos corrigidos após aplicação de algoritmo com K-Nearest Neighbours.

2.2 Descrição do Método

Considere uma amostra x_i de classe desconhecida. A decisão do sobre seu rótulo é feita observando-se as k amostras mais próximas. Atribuindo-se um peso para cada uma delas, realiza-se a soma das amostras pertencentes a cada classe, e uma votação com base na maior soma. Então, sendo p_j a soma para a classe j, e Γ_j o conjunto de amostras dentre as k vizinhas que pertencem à classe j:

$$p_j = \sum_{t \in \Gamma_j} w_t \tag{1}$$

Caso seja atribuído o mesmo peso para cada amostra, então esse somatório pode ser compreendido como uma contagem de quantas amostras de cada classe existem dentre as k selecionadas. O peso atribuído será o inverso ao quadrado da distância normalizada entre cada amostra e x_i, ou seja, considerando-se o vetor de distâncias entre cada amostra do conjunto utilizado como base para o KNN, e a amostra desconhecida x_i, $D = \{d_1, d_2, ..., d_N\}$:

$$w_t = 1 - \left(\frac{d_t}{||D||}\right)^2 \tag{2}$$

Desta forma, a proximidade de cada amostra vizinha passa a ser importante na decisão do rótulo de x_i. Ao normalizar os dados, a relação entre as distâncias permanece, porém seu valor absoluto não importa, sendo este padronizado $[0, 1]$. Ao tomarmos o quadrado desta distância normalizada, o efeito é como se penalizássemos classes que possuem amostras muito próximas e amostras muito distantes, dentre os vizinhos considerados, em detrimento de amostras com certa "regularidade"nas distâncias. Por exemplo, considerando a análise para $k = 4$ vizinhos, cujas distâncias euclidianas são:

- Classe 1: 50, 75
- Classe 2: 25, 100

Aplicando o cálculo da equação 2 sem considerarmos o quadrado das distâncias, a soma dos pesos para as classes, de acordo com a equação 1 é:

- Classe 1: 1.087
- Classe 2: 1.087

E aplicando considerando a equação 2 como escrita:

- Classe 1: 1.567
- Classe 2: 1.433

Portanto, para cada amostra considerada, o seguinte conjunto de passos é executado:

(1) Cria-se um conjunto de treinamento auxiliar composto por todas as amostras do conjunto de treinamento original menos a amostra sob questão.
(2) Treina-se o modelo KNN com essa amostra auxiliar.
(3) Obtém-se o rótulo desta amostra com base neste modelo.

O pseudo-código pode ser visto no algoritmo 1.

Algoritmo 1 Ajuste de Rótulo usando KNN

Seja $\mathbf{X}$ o conjunto de N amostras
Seja $\mathbf{y}$ o vetor de rótulos iniciais, atribuídos
$\mathbf{y_{corrigido}} \leftarrow \mathbf{y}$

function CalculaDistancias(x_i, $\mathbf{X}$)
  distancias $\leftarrow$ [ ]
  for x_j em $\mathbf{X}$ **do**
    distancias.append($|x_j - x_i|$)
  end for
  Retorna $1 - (\mathbf{distancias}/||\mathbf{distancias}||)^2$
end function

for $i \in \{1, 2, ..., N\}$ **do**
  $\mathbf{x_{auxiliar}} \leftarrow \mathbf{X} - \{\mathbf{X}[i]\}$
  $\mathbf{y_{auxiliar}} \leftarrow \mathbf{y} - \{\mathbf{y}[i]\}$
  $D \leftarrow$ CalculaDistancias($\mathbf{X}[i], \mathbf{x_{auxiliar}}$)
  KNN.$ajusta(\mathbf{x_{auxiliar}}, \mathbf{y_{auxiliar}})$
  $\mathbf{y_{corrigido}}[i] \leftarrow$ KNN.$prediz(\mathbf{X}[i], pesos = D)$
end for

Ao final, tem-se um conjunto de rótulos para todas as amostras de treinamento, baseado nos vizinhos mais próximas, que é utilizado para o treinamento do modelo, ao invés do conjunto de rótulos original.

Para os testes, foram utilizados os algoritmos *KNeighboursClassifier* e *MLPClassifier* do módulo scikit-learn (Pedregosa et al., 2011). Uma rodada de *tunning* para os hiperparâmetros do MLP será executada, a fim de se obter o melhor modelo para os dados em questão.

3. PROVA DE CONCEITO COM CONJUNTO DE DADOS SINTÉTICO

O desempenho do algoritmo proposto será avaliado através da análise dos resultados sobre um conjunto de dados em forma de meia lua. Uma série de 30 execuções do ciclo completo - obtenção do conjunto de dados, treinamento e teste sem a utilização de KNN, treinamento e teste com a utilização de KNN - será executada e seus resultados comparados através de testes estatísticos. Ao final, espera-se obter uma conclusão se a aplicação do método proposto no artigo melhorou ou não o desempenho do modelo MLP.

Para a geração dos conjuntos de treinamento e teste, a seguintes hipótese serão considerada:

- O conjunto de treinamento possui variância nos rótulos associada à erros no processo de amostragem, não presente no conjunto de teste.
- A rede foi treinada de forma que seu espaço de hipóteses contempla modelos mais suaves até mais complexos, com possibilidade de sobreajuste (overfitting).

3.1 Teste de hipóteses

Para avaliar se houve melhora com a aplicação do método de filtragem de ruído, são realizados testes estatísticos sobre os resultados obtidos para rotinas sem a aplicação do KNN e com a aplicação. 30 amostras são obtidas de cada algoritmo, e a hipótese nula é de que a utilização de um modelo auxiliar não contribua para a melhora do desempenho da MLP. A hipótese alternativa, sob teste, é a de que a aplicação do KNN apresenta sim uma melhora no desempenho. Estes testes serão aplicados para os valores de F1 score, e espera-se que este parâmetro apresente uma melhora. Portanto, avaliando-se a média μ dos valores de desempenho:

$$\begin{cases} H_0 : \mu_{\text{f1 sem KNN}} = \mu_{\text{f1 com KNN}}, \\ H_1 : \mu_{\text{f1 sem KNN}} < \mu_{\text{f1 com KNN}}. \end{cases} \tag{3}$$

3.2 Resultados obtidos

Para ilustrar o efeito do método de filtragem de ruído proposto na complexidade da superfície de decisão escolhida pela rede, selecionou-se uma das 30 rodadas para fins de ilustração. A curva de separação obtida pode ser vista na figura 6. Seu desempenho se comparado a um conjunto de amostras de teste pode ser observado na figura 7.

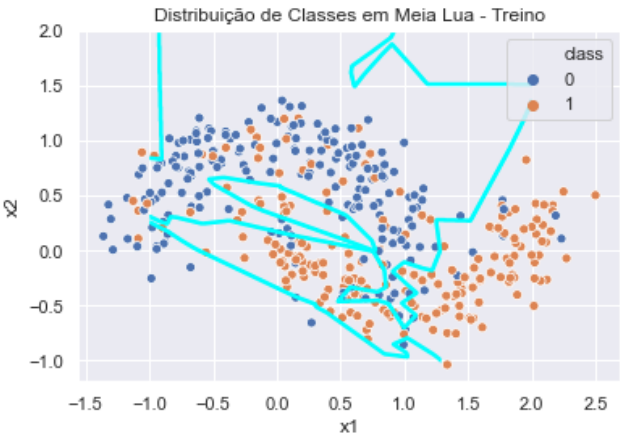

Figura 6. Superfície obtida em comparação com o conjunto de treinamento, sem KNN.

Após correção do conjunto de treinamento com o filtro de ruído proposto, a superfície de separação obtida em comparação com os conjuntos de treinamento e teste pode ser observada nas figuras 8 e 9, respectivamente. É possível observar uma diminuição da complexidade da mesma. O F1 score para os casos mencionados acima podem ser vistos na tabela 1. Houve uma melhora no F1 score, para esta execução.

A fim de verificar estatisticamente o desempenho dos dois algoritmos, 30 execuções foram realizadas, conforme

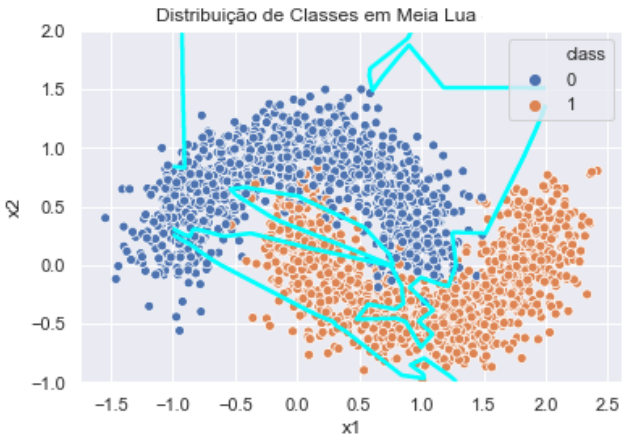

Figura 7. Superfície obtida em comparação com o conjunto de teste, sem KNN.

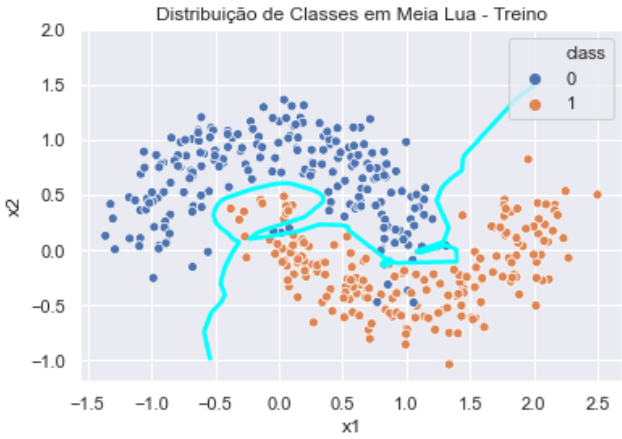

Figura 8. Conjunto de treinamento.

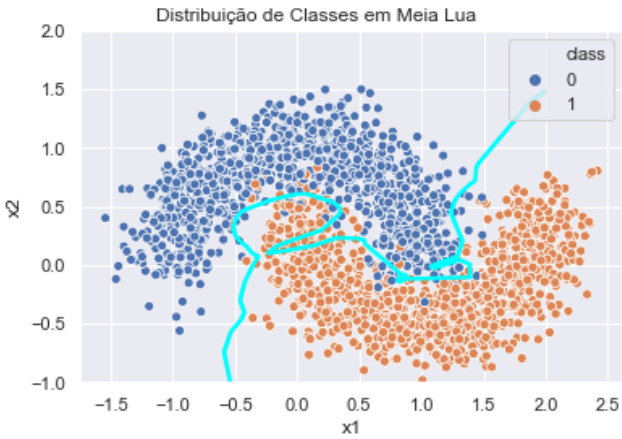

Figura 9. Conjunto de teste.

Tabela 1. F1 Score para dados de teste

Execução	F1
Sem KNN	0.881
Com KNN	0.954

descrito na seção 3.1. Os testes foram realizados em uma rede neural de duas camadas, com 100 neurônios na camada escondida. Os resultados dos testes, para um nível de

confiabilidade de 95%, está sumarizado na tabela 2. Foi testado o *f1-score*, medida de desempenho de classificação.

Tabela 2. Resultado dos testes de hipótese.

Métrica Avaliada	p-valor	Média sem KNN	Média com KNN
F1-score	2.413e-14	0.8713499	0.9351324

Pelo p-valor dos testes realizados o algoritmo proposto foi melhor do que realizar o treinamento com a rede neural diretamente com os dados ruidosos.

4. ESTUDO DE CASO SOBRE UMA BASE DE DADOS REAL

Este problema consiste em distinguir pontos de reversão de pontos de continuidade em séries temporais do mercado de Energia Elétrica. O procedimento de rotulação automática para assinalar pontos de reversão (classe positiva), a serem aprendidos pelo modelo, é naturalmente ruidoso. Tal processo de rotulação assume que amostras presentes nos mínimos locais da série temporal correspondem a pontos causadores da reversão, o que pode não ser verdade. A reversão pode ter sido causada por qualquer amostra (ou sequencia de amostras) na sua vizinhança, ou mesmo, por um evento externo. Dessa forma, o conjunto de dados deve conter ruídos de rotulação, tornando-se um bom laboratório para a aplicação do método de filtragem proposto no artigo.

4.1 Metodologia Experimental

Para simular um conjunto de treinamento ruidoso em relação ao conjunto de teste, taxas controladas de ruído foram adicionadas ao conjunto de treinamento. Para cada amostra uma certa probabilidade de ter seu rótulo trocado foi atribuído, e dessa forma foi gerado o ruído no conjunto. Foram realizados experimentos para 20% de chance de ter seu rótulo trocado e para 40% de chance. Para ambos, foi verificado o treinamento no conjunto de dados original, o treinamento para o conjunto de dados após a inserção de ruído e o treinamento após a aplicação do método de limpeza dos rótulos. As métricas avaliadas foram:

- F1-score da classe *continuation*;
- F1-score da classe *reversion*;
- Precisão da classe *continuation*;
- Precisão da classe *reversion*;
- Recall da classe *continuation*;
- Recall da classe *reversion*;
- Acurácia.

Como as métricas de *f1-score*, *precisão* e *recall* são relativas a cada classe separadamente, são avaliadas para cada classe também. A *acurácia* leva em conta todo o conjunto de teste, portanto não é avaliada por classe.

Foram executadas 100 rodadas de treino e teste seguindo os procedimentos descritos acima, e foram feitos três testes estatísticos com os resultados, para cada métrica:

- A hipótese alternativa de que o desempenho com o conjunto ruidoso é pior do que com o conjunto original;
- A hipótese alternativa de que o desempenho com o conjunto tratado é melhor do que com o conjunto ruidoso;
- A hipótese nula de que o desempenho com o conjunto tratado é próximo do desempenho com o conjunto original.

Os resultados para os testes realizados estão sumarizados nas tabelas 3, 4 e 5.

Tabela 3. Comparação da performance média do dado original e do dado após inserção de ruído.

Métrica Avaliada	p-valor	Média Original	Média Ruidosa
Ruído a 20%			
F1-score classe Reversion	<2.2e-16	0.882	0.815
F1- score classe Continuation	<2.2e-16	0.839	0.756
Precisão classe Reversion	<2.2e-16	0.869	0.815
Precisão classe Continuation	<2.2e-16	0.858	0.757
Recall classe Reversion	<2.2e-16	0.897	0.816
Recall classe Continuation	<2.2e-16	0.821	0.755
Acurácia	<2.2e-16	0.864	0.790
Ruído a 40%			
F1-score classe Reversion	<2.2e-16	0.882	0.650
F1- score classe Continuation	<2.2e-16	0.839	0.574
Precisão classe Reversion	<2.2e-16	0.869	0.674
Precisão classe Continuation	<2.2e-16	0.856	0.55
Recall classe Reversion	<2.2e-16	0.895	0.627
Recall classe Continuation	<2.2e-16	0.823	0.600
Acurácia	<2.2e-16	0.864	0.616

Tabela 4. Comparação da performance média do dado ruidoso e após tratamento.

Métrica Avaliada	p-valor	Média Ruidosa	Média Tratada
Ruído a 20%			
F1-score classe Reversion	<2.2e-16	0.815	0.827
F1- score classe Continuation	0.019	0.756	0.759
Precisão classe Reversion	1	0.815	0.808
Precisão classe Continuation	<2.2e-16	0.757	0.784
Recall classe Reversion	<2.2e-16	0.816	0.846
Recall classe Continuation	1	0.755	0.735
Acurácia	9.477e-16	0.790	0.798
Ruído a 40%			
F1-score classe Reversion	<2.2e-16	0.650	0.690
F1- score classe Continuation	<2.2e-16	0.574	0.606
Precisão classe Reversion	<2.2e-16	0.674	0.701
Precisão classe Continuation	<2.2e-16	0.550	0.595
Recall classe Reversion	<2.2e-16	0.627	0.679
Recall classe Continuation	4.667e-06	0.600	0.619
Acurácia	<2.2e-16	0.616	0.653

4.2 Discussão dos resultados

Como visto na tabela 3, a aplicação de ruído de label no conjunto de treinamento degrada a performance do modelo no conjunto de dados de treinamento, representando a atuação do modelo em uma situação real. Um nível maior de ruído resulta em uma queda maior de performance.

Na tabela 4 é possível observar que a aplicação de uma rodada da rotina proposta surte um efeito sobre o ruído dos dados, e uma melhora no desempenho pode ser observada

Tabela 5. Comparação da performance média do dado original e após tratamento.

Métrica Avaliada	**p-valor**	**Média Original**	**Média Tratada**
Ruído a 20%			
F1-score classe Reversion	<2.2e-16	0.882	0.827
F1- score classe Continuation	<2.2e-16	0.839	0.759
Precisão classe Reversion	<2.2e-16	0.869	0.808
Precisão classe Continuation	<2.2e-16	0.858	0.784
Recall classe Reversion	<2.2e-16	0.897	0.846
Recall classe Continuation	<2.2e-16	0.821	0.735
Acurácia	<2.2e-16	0.864	0.798
Ruído a 40%			
F1-score classe Reversion	<2.2e-16	0.882	0.690
F1- score classe Continuation	<2.2e-16	0.840	0.606
Precisão classe Reversion	<2.2e-16	0.870	0.701
Precisão classe Continuation	<2.2e-16	0.857	0.595
Recall classe Reversion	<2.2e-16	0.895	0.679
Recall classe Continuation	<2.2e-16	0.823	0.619
Acurácia	<2.2e-16	0.864	0.653

para todas as métricas, para um nível de ruído de 40%. Para 20%, porém, houve uma queda na performance de precisão para uma classe, e recall para a outra, e uma melhora nas outras métricas que se reflete em um aumento no valor do *f1-score*. E de acordo com os resultados da tabela 5, o valor de desempenho obtido após a rotina de correção não se aproxima do valor que seria obtido caso nenhum ruído tivesse sido inserido anteriormente no conjunto de dados.

Estes resultados levantam algumas observações sobre o método proposto. O conjunto de dados original ainda possuía um certo nível de ruído intrínseco, que não era decorrente dos erros de rotulação inseridos, apenas. Como o algoritmo tenta ajustar a classe atribuída a uma amostra com base em seus vizinhos, é possível que o mesmo tenha corrigido rótulos decorrentes deste ruído intrínseco, e com isso o conjunto de treinamento tenha se tornado diferente da distribuição do conjunto de teste. Sendo assim, no caso de ruído a 20%, a influência da correção nestes erros inseridos foi próxima da influência da correção do erro intrínseco do conjunto de dados, e por isso para algumas métricas as melhoras na correção foram baixas, se comparadas ao caso de ruído a 40%, e para outras não houve melhoras. O ganho de performance na correção dos erros foi compensado pela perda de performance com a correção do ruído intrínseco, que tornou as distribuições de treinamento e teste diferentes. Para o caso de ruído a 40%, porém, a influência do ruído novo na queda de performance é maior, e portanto a correção dos rótulos apresenta uma melhora nas métricas mais considerável. Porém, como visto, ainda havendo a correção deste ruído natural, ao final a métrica do conjunto tratado ainda era inferior à métrica do conjunto original.

5. CONCLUSÃO

O método proposto procura observar os vizinhos de cada uma das amostras, e tenta inferir se o rótulo atribuído à mesma é plausível dada sua vizinhança, ou se pode ter sido atribuído de maneira errônea. Para conjuntos bem distribuídos, como o conjunto de dados sintéticos utilizado neste trabalho, esta melhora é intuitiva e bem definida. Para um conjunto de dados reais, a aplicação deste algoritmo leva em consideração que o ruído presente no mesmo é majoritariamente devido a erros de rotulação. Nestes casos, é possível observar uma melhora de desempenho de um modelo, e por isso a rotina pode ser aplicada como um método de pré-processamento de dados onde o processo de atribuição das classes apresenta um ruído, ou gera uma taxa de erros. Caso exista uma alta presença de ruído natural dos dados, comum ao conjunto coletado para treinamento e para os conjuntos observados em um cenário real, o mesmo pode tentar corrigir rótulos que são ruidosos tanto pela natureza das amostras, ou por outros vieses, e consequentemente gerar um conjunto de dados diferente do conjunto real.

AGRADECIMENTOS

O presente trabalho foi realizado com o apoio financeiro da CAPES - Brasil.

REFERÊNCIAS

Alejo, R., Sotoca, J.M., Valdovinos, R.M., and Toribio, P. (2010). Edited nearest neighbor rule for improving neural networks classifications. In L. Zhang, B.L. Lu, and J. Kwok (eds.), *Advances in Neural Networks - ISNN 2010*, 303–310. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 785–794. ACM, New York, NY, USA. doi:10.1145/2939672.2939785. URL `http://doi.acm.org/10.1145/2939672.2939785`.

Garcia, L.P.F., Carvalho, A.C.P.L.F., and Lorena, A.C. (2015). Effect of label noise in the complexity of classification problems. *Neurocomputing*. doi:10.1016/j.neucom.2014.10.085.

Haykin, S. (2001). *Redes Neurais: princípios e prática*. Bookman, Porto Alegre, 2nd edition. Trad. Paulo Martins Engel.

Lemaître, G., Nogueira, F., and Aridas, C.K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17), 1–5. URL `http://jmlr.org/papers/v18/16-365.html`.

Li, L. and Abu-Mustafa, Y.S. (2006). Data complexity in machine learning. Technical report, Learning Systems Group, California Institute of Technology.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Tomek, I. (1976). An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*.

Wilson, D.R. and Martinez, T.R. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning 38*.

Wilson, D.L. (1972). Asymtoptic properties of nearest neighbor rules using edited data. *IEEE Transactions On Systems, Man, And Cybernetics, Vol. Smc-2, No.3*.