Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e de Computação

IA048 – Aprendizado de Máquina 13 de abril de 2024

Docentes: Levy Boccato & Romis Attux

Discente:

– Gabriel Toffanetto França da Rocha – 289320

Atividade 2 – Classificação

Sumário

1 Apresentação dos dados

O problema de classificação com o reconhecimento de atividades humanas utiliza como base de dados amostras tomadas do acelerômetro e do giroscópio do *smartphone* preso à cintura do candidato. Dessa forma, com base na leitura desses sensores, pode-se identificar se a pessoa está caminhando, subindo escadas, descendo escadas, sentada, de pé ou deitada, que representam as seis classes do problema.

Além dos dados brutos, é fornecido também os dados processados, com extração sobre os dados no tempo, na frequência, e também características estatísticas dos mesmos.

1.1 Dados tratados

Os dados tratados são formados por amostras de 561 atributos derivados da análise no tempo e na frequência dos dados provenientes do acelerômetro e do giroscópio do *smartphone*. São um total de 7352 amostras para treinamento e validação, e 2947 amostras para teste.

O balanceamento das classes nos conjuntos de dados foi realizado por meio do cálculo da taxa de ocorrência dos mesmos, dada de acordo com (1). A Figura 1 mostra a distribuição das classes, e pode-se ver que não existe um balanceamento homogêneo, onde a classe 3 é a que menos ocorre, enquanto a classe 6 é a que mais ocorre.

Devido a esse desbalanceamento, a métrica que será utilizada para a avaliação do desempenho de cada classificador será a acurácia balanceada, dada por (2).

$$Rate_i = \frac{N_i}{N} \tag{1}$$

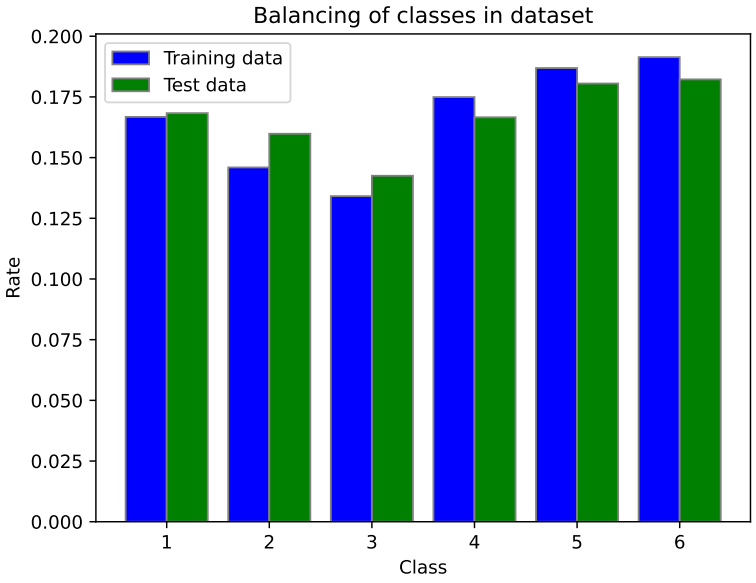


Figura 1: Gráfico da ocorrência das classes nos conjuntos de dados de treinamento e teste.

$$BA = \frac{\sum_{i=1}^{Q} Recall_i}{Q} = \frac{\sum_{i=1}^{Q} \frac{\mathrm{TP}_i}{N_i}}{Q} = \frac{\sum_{i=1}^{Q} \frac{\mathrm{TP}_i}{N \cdot Rate_i}}{Q} \tag{2}$$

2 Classificação via Regressão Logística

A classificação multi-classe é feita de forma elegante ao ter um modelo, que dadas Q classes à serem reconhecidas, apresente Q saídas, onde cada saída é a probabilidade da amostra pertencer à classe em questão. Tal implementação se dá por meio da função *softmax*, enunciada em (3), e a saída é dada pela notação *one-hot encoding*.

$$\hat{y}_k(\mathbf{x}(i)) = \frac{e^{\left(\mathbf{\Phi}(\mathbf{x}(i))^T \mathbf{w}_k\right)}}{\sum_j e^{(\mathbf{\Phi}(\mathbf{x}(i))^T \mathbf{w}_j)}} \tag{3}$$

Onde o $\mathbf{w}_k$ é o vetor de pesos para a classe k, e a matriz de pesos $\mathbf{W}$ é dada por (4).

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_Q \end{bmatrix} \tag{4}$$

Não existe forma fechada para a obtenção dos pesos de $\mathbf{W}$, logo, o mesmo precisa ser feito de forma iterativa. A métrica utilizada como função de custo para o problema é a entropia cruzada, dada por (5). O otimização dos pesos se dá pela técnica do gradiente descendente, dado por (6).

$$J_{CE}(\mathbf{W}) = -\sum_{i=0}^{N-1} \sum_{k=1}^{Q} y_{i,k} \log\left[\hat{y}_k(\mathbf{x}(i)\right] \tag{5}$$

$$\nabla \mathbf{W} = \frac{\partial J_{CE}(\mathbf{W})}{\partial \mathbf{w}_k} = \sum_{i=0}^{N-1} \left(y_{i,k} - \hat{y}_k\left(\mathbf{x}(i)\right)\right) \mathbf{\Phi}(\mathbf{x}(i))^T \tag{6}$$

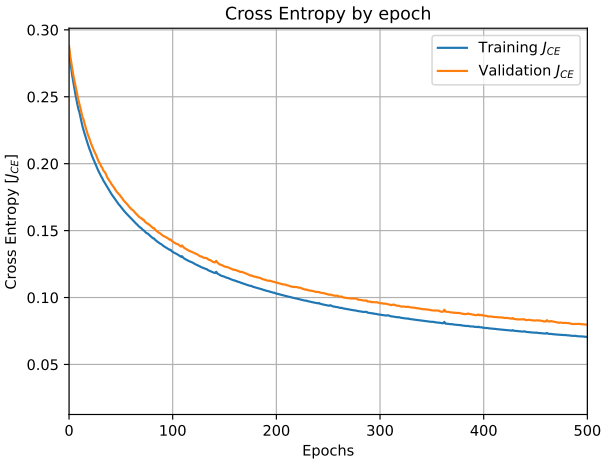
A atualização dos pesos é dada por (7), onde l é a iteração dos pesos.

$$\mathbf{W}[l+1] = \mathbf{W}[l] - \eta \nabla \mathbf{W} \tag{7}$$

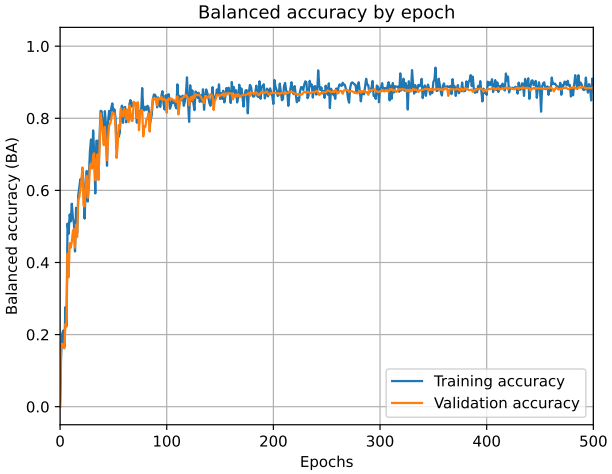
Devido ao tamanho do conjunto de dados, foi proposto o treinamento por *mini-batch*, testadas com tamanhos de 500, 1000 e 2000 amostras, e um passo (η) de 0,01, que apresentou boa convergência nos testes realizados *a priori*. O processo de validação cruzada para o treinamento do modelo foi feito na forma *holdout*, considerando o conjunto de validação 30% do *dataset* de treinamento.

2.1 Dados tratados

Mini-batch 500 amostras



(a) Evolução da entropia cruzada.



(b) Evolução da acurácia balanceada.

Figura 2: Evolução da entropia cruzada e da acurácia balanceada durante o treinamento para *mini-batch* de 500 amostras e $\eta = 0,01$.

$$BA = 0,9162 \tag{8}$$

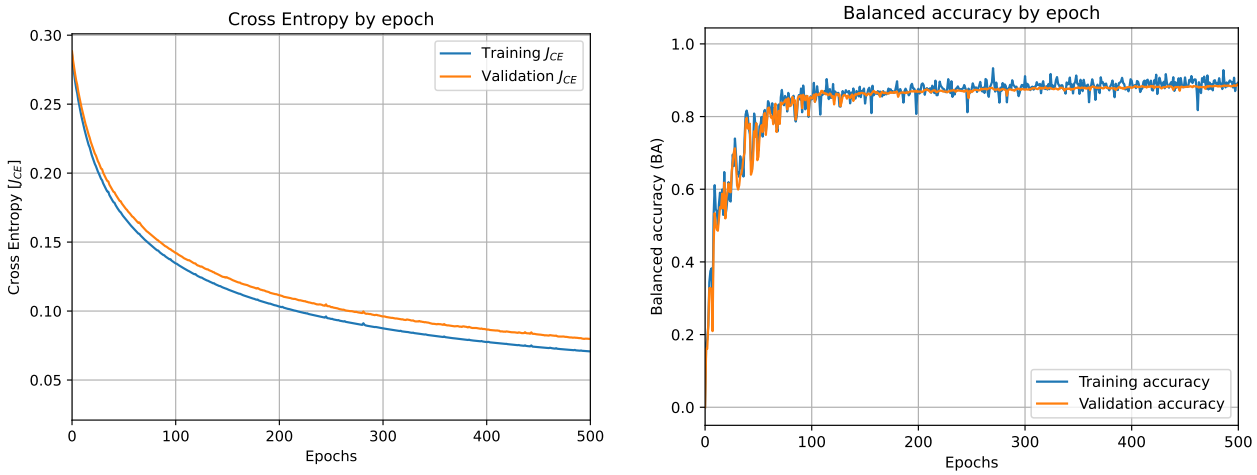
	1	**2**	**3**	**4**	**5**	**6**
1	486	0	10	0	0	0
2	30	440	1	0	0	0
3	39	52	329	0	0	0
4	0	3	0	424	64	0
5	1	0	0	33	498	0
6	0	0	0	0	0	537

Tabela 1: Matriz de confusão do classificador com *mini-batch* de 500 amostras.

Classe	**Precisão**	***Recall***
1	0.9798	0.8741
2	0.9342	0.8889
3	0.7833	0.9676
4	0.8635	0.9278
5	0.9361	0.8861
6	1.0000	1.0000

Tabela 2: Precisão e *Recall* do classificador com *mini-batch* de 500 amostras por classe.

Mini-batch 1000 amostras



(a) Evolução da entropia cruzada.

(b) Evolução da acurácia balanceada.

Figura 3: Evolução da entropia cruzada e da acurácia balanceada durante o treinamento para *mini-batch* de 1000 amostras e $\eta = 0,01$.

$$BA = 0,9042 \tag{9}$$

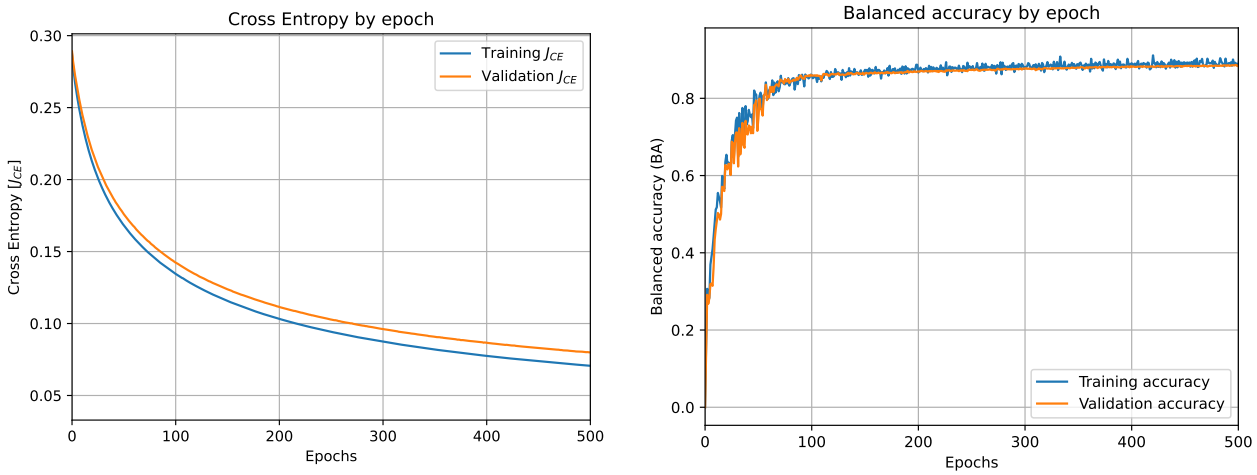
	1	**2**	**3**	**4**	**5**	**6**
1	486	0	10	0	0	0
2	28	443	0	0	0	0
3	56	53	311	0	0	0
4	0	3	0	426	61	1
5	0	1	0	54	477	0
6	0	0	0	0	0	537

Tabela 3: Matriz de confusão do classificador com *mini-batch* de 1000 amostras.

Classe	**Precisão**	***Recall***
1	0.9798	0.8526
2	0.9406	0.8860
3	0.7405	0.9688
4	0.8676	0.8875
5	0.8966	0.8866
6	1.0000	0.9981

Tabela 4: Precisão e *Recall* do classificador com *mini-batch* de 1000 amostras por classe.

Mini-batch 2000 amostras



(a) Evolução da entropia cruzada.

(b) Evolução da acurácia balanceada.

Figura 4: Evolução da entropia cruzada e da acurácia balanceada durante o treinamento para *mini-batch* de 2000 amostras e $\eta = 0,01$.

$$BA = 0,8898 \tag{10}$$

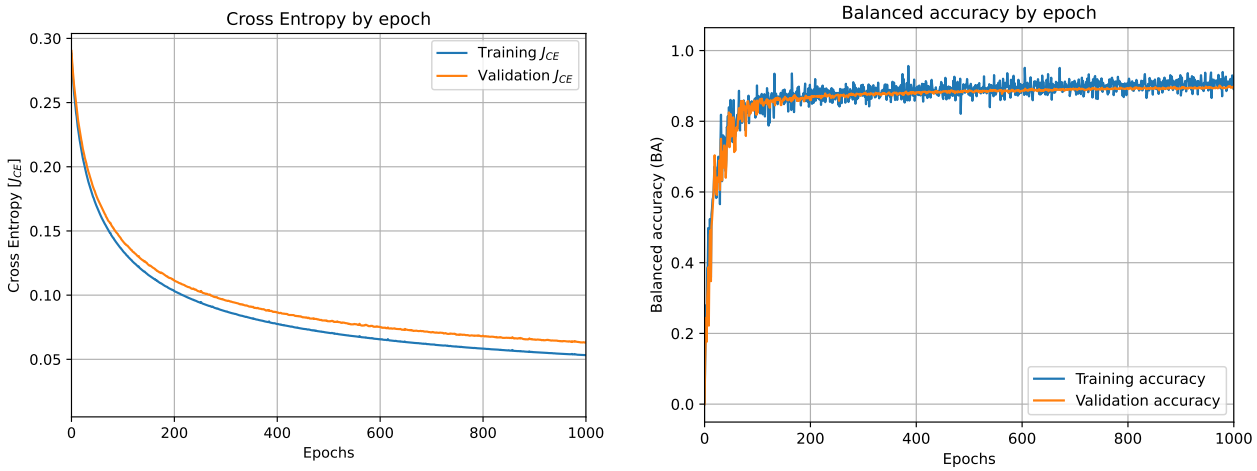
	1	**2**	**3**	**4**	**5**	**6**
1	489	0	7	0	0	0
2	27	444	0	0	0	0
3	70	49	301	0	0	0
4	0	3	0	396	91	1
5	1	1	0	58	472	0
6	0	0	0	0	0	537

Tabela 5: Matriz de confusão do classificador com *mini-batch* de 2000 amostras.

Classe	**Precisão**	***Recall***
1	0.9859	0.8330
2	0.9427	0.8934
3	0.7167	0.9773
4	0.8065	0.8722
5	0.8872	0.8384
6	1.0000	0.9981

Tabela 6: Precisão e *Recall* do classificador com *mini-batch* de 2000 amostras por classe.

Análise



(a) Evolução da entropia cruzada.

(b) Evolução da acurácia balanceada.

Figura 5: Entropia cruzada e acurácia balanceada para o treinamento do modelo com *mini-batch* de 500 amostras por 1000 épocas.

2.2 Dados brutos

3 Classificação via *k neareast neighbours*

A classificação pelo método *k neareast neighbours* é baseada em inferir a classe do dado a ser classificado com base nos k dados mais próximos à ele. Como hiper-parâmetros para esse problema, têm-se principalmente o valor de k, a ordem p da distância de Minkowski entre os dados e o critério de classificação.

O critério de classificação pode se basear puramente na classe majoritária entre os k vizinhos, ou levar em consideração a distância como um peso, que normalmente é inversamente proporcional a distância, evidenciando o rótulo dos pontos mais próximos do dado teste.

3.1 Dados tratados

Para implementação do algoritmo de kNN, foi escolhido a utilização da distância euclidiana no espaço dos atributos, e a decisão do rótulo vencedor por meio do voto majoritário dos k vizinhos mais próximos.

Para obtenção do valor de k, foi executada uma busca em *grid* do hiper-parâmetro, variando seu valor entre 1 e 29. Utilizando da técnica de validação cruzada *k-fold*, com quatro pastas, foi realizada a inferência das classes dos dados da pasta de validação com base nos vizinhos mais próximos encontrados nas pastas de treinamento, para cada valor de k testado. A Figura 6 exibe à evolução da acurácia balanceada para os valores de k, obtendo um conjunto de valores ótimos em (11).

$$k = [17\ 28\ 12\ 18] \tag{11}$$

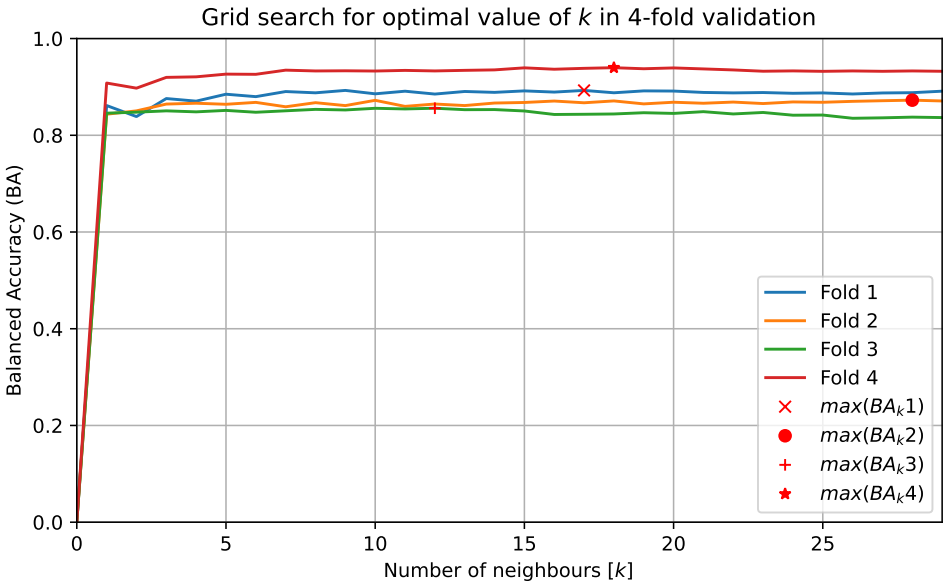


Figura 6: Busca em *grid* do valor de k ótimo utilizando *4-fold validation*.

A heurística escolhida para avaliar o melhor valor de k com base no conjunto obtido por meio da busca em *grid* com validação cruzada se dá em obter a acurácia balanceada média das

pastas e obter o número de vizinhos que maximiza essa combinação das pastas. A Figura 7 mostra a progressão da acurácia balanceada média de acordo com k, e assim se obtém o valor de k ótimo em $k = 15$.

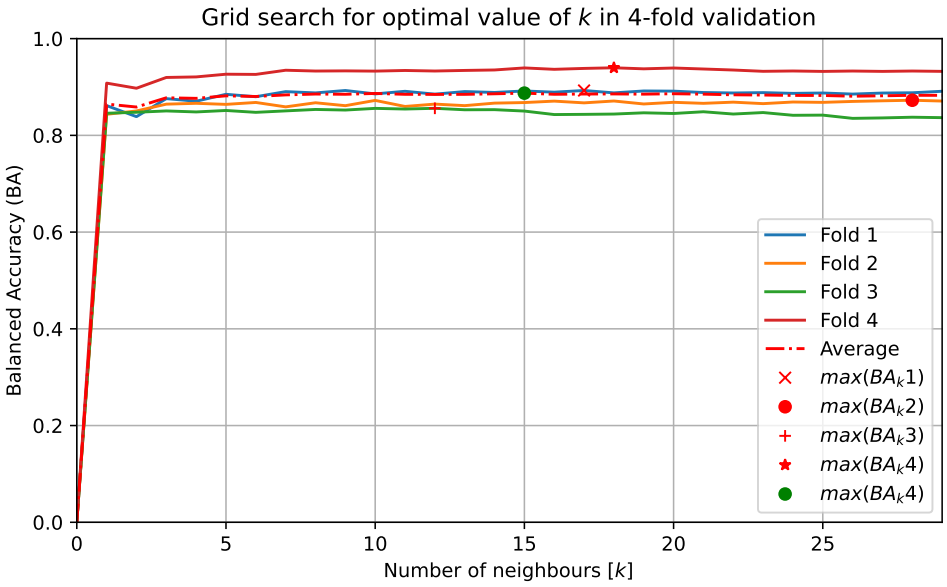


Figura 7: Busca em *grid* do valor de k ótimo utilizando *4-fold validation*.

Uma vez definido o classificador ótimo, obtém-se os indicadores de performance do classificador com base nos dados de teste. A acurácia balanceada encontrada foi de 0,8991 e a matriz de confusão do classificador por ser vista na Tabela 7.

$$BA = 0,8991 \tag{12}$$

	1	**2**	**3**	**4**	**5**	**6**
1	488	0	8	0	0	0
2	39	427	5	0	0	0
3	51	44	325	0	0	0
4	0	4	0	389	98	0
5	0	0	0	31	501	0
6	0	0	0	1	1	535

Tabela 7: Matriz de confusão do classificador k-NN com k = 15.

Extraindo da matriz de confusão as métricas de precisão e *recall*, obtém-se a Tabela 8. Pode-se observar que a classe 3 foi a que apresentou menor precisão, sendo muito confundida com a classe 1 e 2. Já a classe 1 possuí o pior *recall*, uma vez que as classes 2 e 3 se confundem com a 1. A classe 6 foi a que apresentou o melhor desempenho, apresentando *recall* unitário, logo, nenhuma classe se confunde com ela, e a maior precisão, muito próxima de 1.

Classe	Precisão	*Recall*
1	0.9839	0.8443
2	0.9066	0.8989
3	0.7738	0.9615
4	0.7923	0.9240
5	0.9417	0.8350
6	0.9963	1.0000

Tabela 8: Precisão e *Recall* do classificador por classe.

Comparação com a regressão logística

Ao comparar os classificadores utilizando a regressão logística e o k-NN, observa-se que a acurácia balanceada para a regressão logística de melhor desempenho foi superior, mas para ter uma comparação mais minuciosa entre os dois classificadores, deve-se analisar também suas matrizes de confusão. Comparando a precisão e o *recall*, o desempenho das classes em *ranking* se apresenta da mesma forma, porém, para a regressão logística a classe 6 conseguiu obter tanto precisão quanto *recall* máximo.

A regressão logística apresentou um desempenho 1,87% maior que o k-NN para a acurácia balanceada, porém possuí uma complexidade muito maior. Para obter o modelo de regressão logística multi-classe com *softmax*, é necessário realizar o treinamento do modelo, buscando variar os hiper-parâmetros para se conseguir o melhor desempenho. Essa etapa demanda de muito processamento, porém garante uma utilização do classificador mais leve, uma vez que é necessária apenas o vetor de pesos. Já o k-NN, não demanda treinamento, sendo necessária apenas a inferência da classe com base nos vizinhos mais próximos, o que apesar de consumir armazenamento para manter o *dataset*, não precisa de tanto processamento prévio, sendo uma solução simples e rápida. Porém, devido a busca em *grid* utilizando a validação *k-fold* para obter o melhor k para os dados de treinamento, tal etapa demandou bastante processamento, o que elevou consideravelmente a complexidade do k-NN.

3.2 Dados brutos

4 Comparação entre os classificadores

4.1 Dados tratados

Anexos

Códigos fonte

Todos os códigos fonte e arquivos de dados utilizados para a elaboração deste documento podem ser encontrados no repositório do GitHub no link: github.com/toffanetto/ia048.