Conhecimento e Raciocínio

2023/2024

Redes Neuronais – Dataset Stroke

Trabalho Prático

Realizado por:

- Tomás Ferreira – 2021130424
- Manuel Vicente - 2020151796

Índice

Introdução

Este relatório aborda a implementação e análise de um projeto prático centrado no uso de redes neuronais e raciocínio baseado em casos através da toolbox de Deep Learning do Matlab. O trabalho foca na aplicação destas tecnologias para resolver problemas de classificação de um dataset médico, especificamente o *Stroke*. O projeto, envolveu a seleção cuidadosa de datasets, preparação de dados, configuração e treino de redes neuronais, e a criação de uma aplicação gráfica para interação com os modelos. Este relatório documenta o processo e os resultados obtidos, assim como uma reflexão sobre os mesmos.

Preparação do Dataset

No âmbito da preparação inicial dos dados para o trabalho prático, procedemos à conversão dos atributos do ficheiro START para formatos numéricos, uma etapa crucial para garantir a compatibilidade e eficácia do processamento posterior por redes neuronais. Esta conversão foi realizada diretamente no Microsoft Excel, onde cada coluna de dados foi ajustada para refletir tipos de dados numéricos, conforme especificado nas diretrizes do projeto.

Para realizarmos estas alterações, abrimos o ficheiro START no Excel e revimos cada coluna para assegurar que todos os atributos estavam adequados aos requisitos do sistema de redes neuronais que seriam utilizados mais adiante. Através do uso de ferramentas de formatação do Excel, convertemos formatos de texto e outros tipos não numéricos em valores numéricos ou booleanos, garantindo uma uniformidade essencial para análises subsequentes. Este processo não só facilita a manipulação dos dados nas fases de treinamento e validação da rede neuronal, mas também contribui para a precisão dos resultados obtidos.

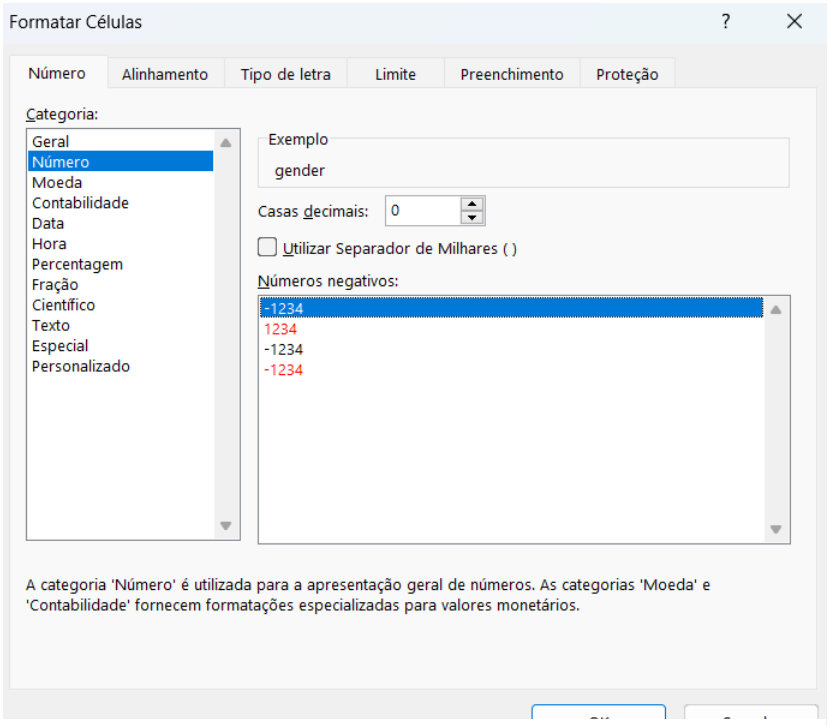

Figura 1- Conversão de Dados

Além disso, escolhemos a coluna do elemento Stroke, como target para a rede neuronal. Também reparámos que há valores em falta (identificados como ‘NA’) nessa coluna, e

resolvemos isso utilizando métodos para preencher esses dados na próxima etapa do trabalho.

Justificação dos pesos dos atributos

1. Género (gender) – Peso: 3
    - Importância moderada, influencia a incidência e tratamento de várias condições de saúde.

2. Idade (age) – Peso: 5
    - Fator crítico devido à forte associação com o risco de doenças crónicas e degenerativas.

3. Hipertensão (hypertension) – Peso: 3
    - Fator de risco significativo para doenças cardiovasculares, refletindo a sua relevância.

4. Doença Cardíaca (heart_disease) – Peso: 3
    - Crucial devido ao impacto substancial nas condições de saúde e mortalidade.

5. Estado Civil (ever_married) – Peso: 1
    - Impacto indireto, mais relacionado a aspetos psicossociais.

6. Tipo de residência (Residence_type) - Peso: 1
    - Influência menor em comparação com os outros fatores.

7. Nível médio de Glicose (avg_glucose_level) - Peso: 5
    - Indicador chave para condições metabólicas.

8. Índice de massa corporal (bmi) – Peso: 2
    - Indica obesidade e riscos relacionados, mas com impacto menos específico.

9. Hábito de fumar (smoking_status) – Peso: 3

- Fator de risco bem conhecido para doenças graves, justificando a sua importância moderada.

Justificação das funções de similaridade local e global

A fase de *Retrieve* é crucial para identificar casos anteriores de um dataset que se assemelham ao novo caso em análise. Para assegurar a eficácia deste processo, utilizamos funções de similaridade específicas para cada atributo (similaridade local) e uma função de similaridade global que integra as contribuições de todos os atributos individuais.

Funções de similaridade global

A função global compila as similaridades locais, ponderadas de acordo com peso de cada atributo. Isto é feito somando as similaridades locais e normalizando pelo total dos pesos, de forma a garantir que a similaridade local esteja sempre entre 0 e 1.
Esta abordagem ponderada permite que atributos considerados mais significativos para o resultado ou diagnóstico tenham uma influência proporcionalmente maior na avaliação da semelhança global entre casos. Por exemplo, atributos como idade e nível médio de glicose, com maior peso, têm um impacto mais substancial na semelhança global devido à sua relevância direta na determinação de condições de saúde.

```
function similarity = calculate_similarity(existing_case, new_case, attributes)

    similarity = 0;

   weighting_factors = [3, 5, 3, 3, 1, 1, 5, 2, 3]; % Correspondente aos atributos


    for i = 1:numel(attributes)
        attribute = attributes{i};
        attribute_weight = weighting_factors(i);

        % Calcular similaridade para cada tipo de atributo
        switch attribute
            case 'gender'
                attribute_similarity = calculate_categorical_similarity(existing_case.(attribute), new_case.(attribute));
            case 'age'
                attribute_similarity = calculate_age_similarity(existing_case.(attribute), new_case.(attribute));
            case 'avg_glucose_level'
                attribute_similarity = calculate_avg_glucose_level_similarity(existing_case.(attribute), new_case.(attribute));
            case 'bmi'
                attribute_similarity = calculate_bmi_similarity(existing_case.(attribute), new_case.(attribute));
            case {'hypertension', 'heart_disease', 'ever_married', 'Residence_type'}
                attribute_similarity = calculate_categorical_similarity(existing_case.(attribute), new_case.(attribute));
            case 'smoking_status'
                attribute_similarity = calculate_smoking_status_similarity(existing_case.(attribute), new_case.(attribute));
            otherwise
                error('Attribute %s not supported.', attribute);
        end

        % Aplicar o peso ao valor da similaridade do atributo
        similarity = similarity + attribute_similarity * attribute_weight;
    end

    % Normalizar pela soma dos pesos para manter a similaridade entre 0 e 1
    similarity = similarity / sum(weighting_factors);
end
```

Figura 2 - Função de similaridade global

Funções de similaridade local

Para os atributos binários, utilizamos uma função de similaridade binária, onde atributos idênticos recebem uma similaridade de 1 e atributos diferentes uma similaridade de 0. No caso do estado de fumador, adaptámos a função para considerar algumas semelhanças parciais, refletindo nuances como fumador ocasional versus regular.

```
function similarity = calculate_categorical_similarity(value1, value2)
    if value1 == value2
        similarity = 1; % Iguais
    else
        similarity = 0; % Diferentes
    end
end
```

Figura 3 -Função de Similaridade Local Binária

Esta função avalia a similaridade entre o *smoking_status* de dois indivíduos. Atribui uma similaridade de 1 quando os estados são iguais, refletindo uma correspondência exata. Uma similaridade de 0.75 é concedida entre ex-fumantes e fumantes atuais, reconhecendo

uma experiência de fumo comum, enquanto todas as outras combinações recebem uma similaridade de 0.

```
function similarity = calculate_smoking_status_similarity(value1, value2)

    if value1 == value2
        similarity = 1;
    elseif (value1 == 1 && value2 == 2) || (value1 == 2 && value2 == 1)
        similarity = 0.75;
    else
        similarity = 0;
    end
end
```

Figura 4 -Função de Similaridade Local adaptado ao smoking_status

As funções para atributos numéricos baseiam-se na diferença absoluta entre os valores, normalizada em relação a um intervalo de valores possível (0 a um máximo específico para cada atributo). Isso permite que a similaridade diminua linearmente à medida que a diferença entre os valores aumenta, refletindo uma abordagem mais precisa do que simples comparações binárias.

Lógica das funções de similaridade para atributos quantitativos:

- **Normalização de valores**: Define-se os valores máximo e mínimo esperados para cada atributo. Para *age*, é considerado um intervalo de 0 a 90 anos, para *avg_glucose_level* de 0 a 275 e para *bmi* de 0 a 60.

- **Cálculo da distância**: Mede-se a diferença absoluta entre os dois valores comparados.

- **Conversão da distância em similaridade**: A similaridade é determinada como, 1 menos a proporção da distância sobre o intervalo possível, ajustando assim a similaridade para que se encontre entre 0 (mínima similaridade) e 1 (máxima similaridade).

```
function similarity = calculate_age_similarity(value1, value2)

    min_value = 0; % Valor mínimo possível
    max_value = 90; % Valor máximo possível

    % Distância entre valores
    distance = abs(value1 - value2);

    % Calcular a similaridade como 1 menos a proporção da distância sobre o intervalo
    similarity = 1 - (distance / (max_value - min_value));

    % Garantir que a similaridade está entre 0 e 1
    similarity = max(0, min(1, similarity));
end
```

Figura 5 - Exemplo de função de similaridade local para atributos quantitativos

Estudo e análise de redes neuronais

Resultados obtidos utilizando o ficheiro START

Para iniciar a avaliação do desempenho da rede neuronal, utilizamos a configuração padrão.

Configuração						Média (50 iterações)		
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Total	Erro	Tempo de execução (segundos)
defaultStart	1	10	tansig, purelin	trainlm	sem segmentação	100.000000%	0.000000	0.007700

Figura 6 - Resultados obtidos usando a configuracão default

Após testarmos a configuração inicial, explorámos diversas variantes, ajustando as funções de treino e de ativação:

Configuração						Média (50 iterações)		
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Total	Erro	Tempo de execução (segundos)
defaultStart	1	10	tansig, purelin	trainlm	sem segmentação	100.000000%	0.000000	0.007700
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Total	Erro	Tempo de execução (segundos)
defaultStart2	1	10	tansig, purelin	traincgb	sem segmentação	100.000000%	0.000000	0.0227
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Total	Erro	Tempo de execução (segundos)
defaultStart3	1	10	tansig, purelin	traingdx	sem segmentação	100.000000%	0.000000	0.0668
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Total	Erro	Tempo de execução (segundos)
defaultStart4	1	10	logsig, purelin	trainlm	sem segmentação	100.000000%	0.000000	0.007360
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Total	Erro	Tempo de execução (segundos)
defaultStart5	1	10	logsig,compet	trainlm	sem segmentação	50.000000%	0.500000	0.001780
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Total	Erro	Tempo de execução (segundos)
defaultStart6	1	10	logsig, purelin	traingdx	sem segmentação	100.000000%	0.000000	0.118420

Figura 7 - Resultados médios obtidos variando funções de treino e de ativação

Com base nos resultados, podemos concluir que a escolha das funções de ativação e de treino tem um impacto significativo no desempenho da rede neuronal. Configurações que combinam 'tansig' e 'purelin' para ativação e 'traincgb', 'traingdx' e 'trainlm' para treino, obtiveram os melhores resultados em termos de precisão. A configuração defaultStart6, com 'logsig' e 'compet', apresentou um desempenho substancialmente inferior, indicando que essa combinação não é adequada para este problema. Além disso, observámos que o tempo de execução variou entre as configurações, sendo mais rápido na configuração defaultStart5 e mais lento na configuração defaultStart6.

É importante denotar, que a configuração defaultStart5, apesar de ter sido a mais rápida, registou uma precisão significativamente inferior, o que demonstra que nem sempre a velocidade computacional se correlaciona com os melhores resultados.

Portanto, concluímos assim que a melhor configuração de rede neuronal foi a defaultStart4, uma vez que alcançou 100% de precisão, sendo também a mais rápida entre aquelas que atingiram esse valor máximo de precisão.

Resultados obtidos utilizando o ficheiro TRAIN

Inicialmente, foram conduzidos testes utilizando a configuração default:

Configuração					
Nome	nero de camadas escondi	Número de Neurónios	Funções de Ativação	Funcão de Treino	Divisão dos Exemplos
default	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}

Média (50 iterações)	
Precisão Global	Precisão Teste
77.249180	73.347826

Melhor Rede (Global)		
Nome	Precisão Global	Precisão Teste
defaultG	80.655738	76.086957

Melhor Rede (Teste)		
Nome	Precisão Global	Precisão Teste
defaultT	79.508197	84.782609

Figura 8 - Configuração default para o ficheiro TRAIN

Reparámos que a configuração padrão, apresentou resultados razoáveis. De seguida, foram realizados testes, onde explorámos diferentes arquiteturas de redes feedforward, funções de treino, funções de ativação e segmentações de dados para train, validation e test.

O Número e dimensão das camadas escondidas influencia o desempenho?

O número e dimensão das camadas encondidas influencia o desempenho?					
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Funcão de Treino	Divisão dos Exemplos
conf1	2	5,5	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}
conf2	2	10,10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}
conf3	3	5,5,5	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}
conf4	3	10,10,10	tansig, tansig,tansig purelin	trainlm	dividerand = {0.7, 0.15, 0.15}
conf23	2	50,50	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}
conf24	2	100,100	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}

Média (50 iterações)	
Precisão Global	Precisão Teste
76.065574	72.456522
77.354098	71.695652
76.577049	73.347826
77.639344	71.913043
81.275410	70.252874
84.590164	69.264368

Melhor Rede (Global)		
Nome	Precisão Global	Precisão Teste
conf1G	79.344262	72.826087
conf2G	81.803279	69.565217
conf3G	80.000000	75.000000
conf4G	82.295082	68.478261
conf23G	87.213115	78.160920
conf24G	89.016393	72.413793

Melhor Rede (Teste)		
Nome	Precisão Global	Precisão Teste
conf1T	74.426230	83.695652
conf2T	78.196721	81.521739
conf3T	75.901639	82.608696
conf4T	75.901639	80.000000
conf23T	79.672131	81.609195
conf24T	85.245902	74.712644

Figura 9 - Resultados obtidos a variar o número e a dimensão das camadas escondidas

Com base nos resultados obtidos com diferentes configurações de redes neuronais, podemos tirar as seguintes conclusões em relação à influência do número e dimensão das camadas escondidas no desempenho do modelo:

- As configurações com um maior número de camadas escondidas e um maior número de neurónios tendem a apresentar maior precisão global.

- Redes com um maior número de camadas e neurónios tendem a apresentar uma queda no desempenho no conjunto de teste, o que sugere uma possível ocorrência de *overfitiing*. Por exemplo, a configuração conf24, com duas camadas escondidas e 100 neurónios por camada, obteve uma precisão global média, obteve uma precisão global média de 84.59%, mas uma precisão no conjunto de teste de apenas 69.26%.

A função de treino influencia o desempenho?

A função de treino influencia o desempenho?						Média (50 iterações)		Melhor Rede (Global)			Melhor Rede (Teste)		
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Funcão de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste
conf5	1	10	tansig,purelin	trainbfg	dividerand = {0.7, 0.15, 0.15}	76.039344	73.978261	conf5G	79.016393	78.260870	conf5T	71.803279	89.130435
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Funcão de Treino	Divisão dos Exemplos	Média (50 iterações)		Melhor Rede (Global)			Melhor Rede (Teste)		
						Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste
conf6	1	10	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}	68.600000	67.652174	conf6G	73.114754	77.173913	conf6T	73.114754	79.347826
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Funcão de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste
conf7	1	10	tansig, purelin	trainbr	dividerand = {0.7, 0.15, 0.15}	84.422951	71.652174	conf7G	86.885246	78.260870	conf7T	84.098361	78.260870
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Funcão de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste
conf8	1	10	tansig, purelin	traincgb	dividerand = {0.7, 0.15, 0.15}	75.990164	73.521739	conf8G	78.524590	75.000000	conf8T	76.885246	81.521739
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Funcão de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste
conf10	1	10	tansig, purelin	traincgp	dividerand = {0.7, 0.15, 0.15}	76.334426	73.217391	conf10G	78.524590	66.304348	conf10T	76.393443	84.782609

Figura 10 - Resultados ao variar a função de treino

Com base nos resultados obtidos com diferentes funções de treino, podemos retirar as seguintes conclusões em relação à influência dessas funções no desempenho do modelo:

- Verificaram-se variações significativas na precisão global e na precisão de teste ao longo das diferentes funções de treino utilizadas. Por exemplo, a precisão global variou de aproximadamente 68.6% a 84.4%, dependendo da função de treino.

- A função de treino "trainbfg" (conf7G) resultou numa das mais altas precisões globais, alcançando até 86,88%, enquanto "traingd" (conf6G) teve uma precisão global inferior, em torno de 73,11%.

- Notavelmente, a precisão de teste também variou consideravelmente entre as diferentes funções de treino. Por exemplo, a função “trainbfg” teve uma precisão de teste de cerca de 71.65%, enquanto “traincgp” teve uma precisão de teste de aproximadamente 73.22%.

- Algumas funções de treino, como “traincgp”, demonstram uma variação considerável na precisão de teste entre as execuções, o que sugere uma possível instabilidade no modelo.

- As melhores redes, tanto em termos de precisão global quanto de precisão de teste, variaram dependendo da função de treino. Por exemplo, enquanto a função “trainbr” produziu a melhor precisão global, a função “traincgp” resultou na melhor precisão de teste.

As funções de ativação influenciam o desempenho?

As funções de ativação influenciam o desempenho?					
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Funcão de Treino	Divisão dos Exemplos
conf13	1	10	logsig,purelin	trainlm	dividerand = {0.7, 0.15, 0.15}
conf14	1	10	tansig,logsig	trainlm	dividerand = {0.7, 0.15, 0.15}
conf15	1	10	logsig,compet	trainlm	dividerand = {0.7, 0.15, 0.15}
conf25	1	10	hardlim,tribas	trainlm	dividerand = {0.7, 0.15, 0.15}
conf26	1	10	poslin,radbas	trainlm	dividerand = {0.7, 0.15, 0.15}
conf27	1	10	netinv,softmax	trainlm	dividerand = {0.7, 0.15, 0.15}

Média (50 iterações)	
Precisão Global	Precisão Teste
76.977049	74.739130
76.518033	74.326087
49.878689	49.456522
68.763934	63.931034
66.701639	63.517241
55.126437	63.517241

Melhor Rede (Global)		
Nome	Precisão Global	Precisão Teste
conf13G	80.491803	76.086957
conf14G	79.180328	70.652174
conf15G	66.393443	69.565217
conf25G	72.295082	64.367816
conf26G	77.868852	78.160920
conf27G	68.852459	63.218391

Melhor Rede (Teste)		
Nome	Precisão Global	Precisão Teste
conf13T	77.540984	85.869565
conf14T	76.229508	86.956522
conf15T	66.393443	86.956522
conf25T	69.180328	77.011494
conf26T	77.868852	78.160920
conf27T	64.590164	67.816092

Figura 11 - Resultados obtidos a variar as funções de ativação

- Verificou-se que as diferentes combinações de funções de ativação tiveram um impacto significativo tanto na precisão global quanto na precisão de teste. Por exemplo, a precisão média global variou de aproximadamente 49.88% a 77.87%, enquanto a precisão média de teste variou de cerca de 49.46% a 86.96%.

- Nas configurações onde foram utilizadas funções de ativação convencionais, como “tansig” e “purelin”, observamos resultados consistentemente melhores em comparação com outros funções menos convencionais. Isso sugere que as funções de ativação convencionais tendem a ser mais adequadas para o problema em questão.

- Ao utilizar funções de ativação alternativas, como “logsig”, “compet”, “hardlim”, “tribas”, “poslin”, “radbas”, “netinv” e “softmax”, observamos uma ampla variação no desempenho. Estas funções quando não combinadas com as

funções de ativação mais convencionais, resultaram num desempenho abaixo do esperado.

A divisão de exemplos pelos conjuntos influencia o desempenho?

A divisão de exemplos pelos conjuntos influencia o desempenho?						Média (50 iterações)		Melhor Rede (Global)			Melhor Rede (Teste)		
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste
conf16	1	10	tansig,purelin	trainlm	dividerand = [0.9, 0.05, 0.05]	77.193443	74.645161	conf16G	82.950820	77.419355	conf16T	77.049180	90.322581
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste
conf18	1	10	tansig,purelin	trainlm	dividerand = [0.33, 0.33, 0.33]	74.849180	72.009852	conf18G	78.032787	72.906404	conf18T	77.377049	78.325123
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste
conf17	1	10	tansig,purelin	trainlm	dividerand = [0.8, 0.1, 0.1]	74.901639	71.270936	conf17G	78.196721	73.891626	conf17T	77.049180	77.832512
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste
conf19	1	10	tansig,purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	77.403279	72.934783	conf19G	82.131148	68.478261	conf19T	78.032787	81.521739
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste
conf20	1	10	tansig,purelin	trainlm	dividerand = [0.5, 0.25, 0.25]	76.504918	73.738562	conf20G	79.508197	73.202614	conf20T	76.557377	82.352941
Nome	Número de camadas escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste	Nome	Precisão Global	Precisão Teste
conf21	1	10	tansig,purelin	trainlm	dividerand = [0.85, 0.05, 0.1]	77.481967	72.645161	conf21G	82.459016	70.967742	conf21T	75.573770	90.322581

Figura 12 - Resultados a variar a divisão dos exemplos pelos conjuntos

- Observamos que diferentes proporções de divisão dos conjuntos de dados levaram a variações nas métricas de desempenho. Por exemplo, configurações como conf16 e conf21, que utilizaram proporções de divisão mais altas para o conjunto de treino, apresentaram uma precisão global geralmente superior, indicando que uma maior quantidade de dados de treino pode contribuir para um melhor desempenho da rede.

Resultados obtidos utilizando o ficheiro TEST

Resultados	
conf16T	70%
conf14T	70%
conf5T	70%
conf5G	60%
conf12G	80%
conf24G	60%

- A precisão das redes variou consideravelmente, com algumas redes a alcançar precisões de 60%, 70% e 80%. Isto sugere que a qualidade das redes neuronais treinadas pode depender de vários fatores, incluindo a arquitetura da rede, a função de treino e a divisão dos exemplos.

- Notavelmente, a rede conf12G alcançou a maior precisão, 80%. Isto sugere que essas configurações específicas foram capazes de apresentar um bom desempenho na classificação de exemplos desconhecidos.

- Por outro lado, algumas redes, como conf5G e conf24G, mostraram uma precisão mais baixa, 60%. Isto indica que essas configurações podem não ter sido tão eficazes na aprendizagem dos padrões nos dados de treino e, portanto, não conseguiram generalizar adequadamente para os exemplos de teste.

Com base nessas conclusões, podemos inferir que a escolha da arquitetura da rede, juntamente com a função de treino e a divisão dos exemplos, desempenha um papel crucial na determinação da capacidade de generalização e aprendizagem das redes neurais. Experimentar diferentes configurações e técnicas de treino pode ajudar a identificar a combinação mais eficaz para um determinado problema.

Conclusão

Com base nos resultados obtidos e nas análises realizadas, concluímos que a construção e o treino de redes neurais para a classificação de dados exigem uma abordagem cuidadosa na seleção dos parâmetros e configurações. A escolha adequada da arquitetura da rede, das funções de ativação, da função de treino e da divisão dos exemplos são cruciais para alcançar um desempenho ótimo. Além disso, os testes de generalização realizados com o conjunto de teste destacam a importância da capacidade da rede neural em lidar eficazmente com dados não vistos durante o treino. Em suma, a análise dos resultados sublinha a necessidade de uma abordagem experimental rigorosa e de ajustes iterativos para encontrar a melhor configuração de rede para um determinado problema de classificação.