Instituto Superior de Engenharia

Politécnico de Coimbra

Licenciatura em Engenharia Informática

Curso Engenharia Informática

Ramo de Desenvolvimento de Aplicações

Unidade Curricular de Conhecimento e Raciocínio

Ano Letivo de 2023/2024

11 de maio de 2024

Relatório do trabalho prático de Conhecimento e Raciocínio

João Alves Pereira de Carvalho - 2019131769

Pedro Romão Raimundo Vaz de Paiva - 202113462

Trabalho de avaliação de natureza académica

Coimbra, 11 de maio de 2024

Índice

1. INTRODUÇÃO

Neste relatório será feita a descrição de todo o processo de realização deste trabalho da Unidade Curricular de Conhecimento e Raciocínio, em que se pretendia que os alunos explorassem e aprofundassem os conceitos de raciocínio baseado em casos e de redes neuronais (RNs).
O trabalho foi implementado usando o Matlab, nomeadamente a toolbox Deep Learning (redes neuronais).

2. Introdução ao Dataset

Optamos por explorar um conjunto de dados que se concentra nos casos de AVC (Acidente Vascular Cerebral), uma condição médica de grande relevância e impacto na saúde pública. Este dataset oferece uma visão abrangente desses casos, fornecendo nove atributos de entrada que abordam diversos aspetos relacionados à saúde e estilo de vida dos indivíduos.

Os atributos incluem informações como género, idade, presença de hipertensão e doença cardíaca, estado civil, tipo de residência, nível médio de glucose, índice de massa corporal (IMC) e histórico de tabagismo. Cada um desses atributos desempenha um papel significativo na análise e compreensão dos fatores de risco associados ao AVC.

A coluna alvo, representada pelo atributo "stroke", desempenha um papel fundamental na tarefa de classificação, indicando se o indivíduo sofreu um AVC ou não. Esta distinção entre casos positivos e negativos de AVC é crucial para a aplicação de técnicas de aprendizado de máquina e classificação, visando identificar padrões e características que possam estar associados a esta condição médica de classificação para identificar fatores de risco e contribuir para uma melhor compreensão dessa condição médica.

3. Preparação do Dataset

Para preparar o dataset em conformidade com as informações dadas no enunciado, foi necessário seguir alguma etapas:

Conversão de Atributos para Valores Numéricos:

Primeiro foi necessário examinar os tipos de dados dos atributos no dataset do arquivo START.

De seguida, foi necessário fazer alterações no dataset do arquivo TRAIN para que ele possua as mesmas características do dataset START.

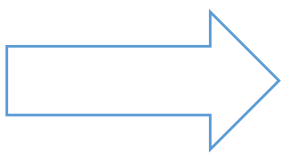
A	B	C	D	E	F	G	H	I	J
id	gender	age	hypertensi	heart_dise	ever_marri	Residence	avg_gluco	bmi	smoking_status
1	Male	67	0	1	Yes	Urban	228.69	36.6	formerly smoked
2	Female	61	0	0	Yes	Rural	202.21	48.2	never smoked
3	Male	80	0	1	Yes	Rural	105.92	32.5	never smoked
4	Female	49	0	0	Yes	Urban	171.23	34.4	smokes
6	Male	81	0	0	Yes	Urban	186.21	29	formerly smoked
7	Male	74	1	1	Yes	Rural	70.09	27.4	never smoked

id	gender	age	hyperte	heart_d	ever_ma	Reside	avg_glu	bmi	smokin
1	0	67	0	1	1	1	228.69	36.6	1
2	1	61	0	0	1	0	202.21	48.2	0
3	0	80	0	1	1	0	105.92	32.5	0
4	1	49	0	0	1	1	171.23	34.4	2
6	0	81	0	0	1	1	186.21	29	1
7	0	74	1	1	1	0	70.09	27.4	0

Identificação do Atributo de Saída (Target):

Na segunda etapa, foi necessário identificar qual a coluna do conjunto de dados do ficheiro de TRAIN correspondia à saída desejada da rede neuronal. Esta coluna teria valores identificados como NA, indicando os valores em falta. No entanto, por questões de logística, optamos por alterar o valor para NaN (not a number), evitando assim a conversão de string para número..

stroke
1
1
1
1
1
1
NA



stroke
1
1
1
1
1
1
NaN

4. Implementação do Sistema de Raciocínio Baseado em Casos

Implementámos um sistema de raciocínio baseado em casos para preencher os valores em falta (NA) nos atributos do conjunto de dados.

O raciocínio baseado em casos (CBR de Case-Based Reasoning) procura resolver novos problemas adaptando soluções de problemas anteriores, utilizando experiências armazenadas no próprio sistema.

Resumidamente, o sistema CBR armazena experiências anteriores (casos) na memória. Quando surge um novo problema, extrai (Retrieve) da memória casos semelhantes, utilizando uma medida de semelhança para estimar a diferença entre o novo caso e os casos armazenados.

Na implementação, focámo-nos na fase de RETRIEVE do sistema de CBR. Durante esta fase, percorremos todos os casos armazenados na memória e calculamos a similaridade entre cada um desses casos e o novo caso que necessita de preenchimento.

Para calcular a similaridade, consideramos diferentes tipos de atributos, como binários, numéricos e categóricos.

Além disso, atribuímos pesos aos diferentes atributos para refletir a sua importância na determinação do risco de AVC.

Este processo envolve a comparação das características do novo caso com os casos armazenados, utilizando medidas de similaridade para determinar a proximidade entre eles. O caso mais semelhante é então utilizado para preencher os valores em falta no novo caso.

Este sistema de raciocínio baseado em casos oferece uma abordagem eficaz para lidar com valores em falta nos atributos do conjunto de dados, contribuindo assim para uma análise mais completa e precisa.

5. JUSTIFICAÇÃO DOS PESOS ATRIBUIDOS

Para calcularmos corretamente a similaridade entre casos, é essencial atribuirmos pesos aos diferentes atributos de entrada. Esses pesos refletem a importância relativa de cada atributo na determinação do risco de AVC, permitindo uma análise mais precisa e eficaz dos dados. Neste contexto, a atribuição adequada de pesos é crucial para garantir que a análise dos atributos leve em consideração sua relevância clínica e impacto potencial na ocorrência de AVC.

'Gender' (Género): Peso 1

Consideramos este atributo como tendo um peso menor devido à sua influência potencialmente menor na ocorrência de AVC em comparação com outros fatores de saúde mais diretos.

'Age' (Idade): Peso 4

A idade é um fator significativo no risco de AVC, com estudos mostrando uma correlação forte entre idade avançada e maior probabilidade de ocorrência de AVC.

'Hypertension' (Hipertensão): Peso 5

A hipertensão arterial é um dos principais fatores de risco para AVC, sendo atribuído um peso mais alto devido à sua forte associação com a condição.

'Heart_disease' (Doença Cardíaca): Peso 4

Assim como a hipertensão, a presença de doença cardíaca é um fator de risco significativo para AVC, justificando o peso mais elevado atribuído a este atributo.

'Ever_married' (Estado Civil): Peso 1

Embora o estado civil possa ter alguma influência no estilo de vida e saúde, atribuímos um peso menor a este atributo devido à sua relação potencialmente menos direta com a ocorrência de AVC.

'Residence_type' (Tipo de Residência): Peso 1

O tipo de residência pode ter alguma influência no estilo de vida, mas consideramos que sua contribuição para o risco de AVC é menos significativa em comparação com outros fatores de saúde.

'Avg_glucose_level' (Nível Médio de Glucose): Peso 3

Níveis elevados de glucose no sangue estão associados a um maior risco de AVC, justificando o peso atribuído a este atributo, embora seja menor do que para hipertensão e doença cardíaca.

'BMI' (Índice de Massa Corporal): Peso 2

O índice de massa corporal é um indicador importante de saúde e peso corporal, que pode influenciar indiretamente o risco de AVC..

6. JUSTIFICAÇÃO DAS FUNÇÕES DE SIMILARIDADE LOCAL E GLOBAL

As funções de similaridade local e global desempenham papéis fundamentais no sistema de raciocínio baseado em casos (CBR), contribuindo para a determinação da proximidade entre casos armazenados e novos casos a serem avaliados. A escolha e justificação dessas funções foram cuidadosamente consideradas para se adequarem ao contexto do problema em questão.

7. FUNÇÕES DE SIMILARIDADE LOCAL

As funções de similaridade local são responsáveis por calcular a similaridade entre os atributos individuais de dois casos. Como demonstrado a seguir, essas funções foram implementadas para diferentes tipos de atributos:

7.1 Atributos Binários - calculate_binary_distance

Para atributos como género, hipertensão, doença cardíaca, casado(a) anteriormente e tipo de residência, a função calculate_binary_distance é utilizada, atribuindo uma similaridade de 1 se os valores forem iguais e 0 caso contrário. Essa abordagem é apropriada, pois permite uma comparação direta entre valores binários.

```
function similarity = calculate_binary_distance(val1, val2)
    % 1 se iguais, 0 cc
    similarity = double(val1 == val2);

    % fprintf('calculate_binary_distance\n');
    % fprintf('Valores de entrada: val1 = %d, val2 = %d\n', val1, val2);
    % fprintf('Similaridade calculada: %.2f\n', similarity);
end
```

7.2 Atributos Intervalares - calculate_intervalo_similarity

A função ***calculate_intervalo_similarity*** foi feita para atributos como idade, nível médio de glicose e IMC. Esta função calcula a similaridade com base na distância entre os valores e na amplitude do intervalo permitido para cada atributo. Isso proporciona uma medida de similaridade que leva em consideração a proximidade dos valores dentro de um intervalo definido.

```
function similarity = calculate_intervalo_similarity(value1, value2,min_value,max_value)
    % Distância entre valores
    distance = abs(value1 - value2);

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

7.3 Atributo Categórico - calculate_smoking_status_similarity

Esta função é específica para o atributo de status de tabagismo, representada por números de 0 a 3, cada um representando um estado diferente. A similaridade é definida com base nas diferentes combinações possíveis de valores para o status de tabagismo.

```
function similarity = calculate_smoking_status_similarity(val1, val2)
    if val1 == val2
        similarity = 1;
    elseif (val1 == 1 && val2 == 2) || (val1 == 2 && val2 == 1)
        similarity = 0.60;
    elseif (val1 == 1 && val2 == 0) || (val1 == 0 && val2 == 1)
        similarity = 0.25;
    else
        similarity = 0; % Diferentes
    end

    % fprintf('calculate_smoking_status_distance\n');
    % fprintf('Valores de entrada: val1 = %d, val2 = %d\n', val1, val2);
    % fprintf('Similaridade calculada: %.2f\n', similarity);
end
```

8. FUNÇÕES DE SIMILARIDADE GLOBAL

A função de similaridade global é responsável por determinar se um caso armazenado na memória é considerado suficientemente semelhante ao novo caso para ser recuperado durante a fase de Recuperação. No código fornecido, isso é realizado pelo cálculo de uma similaridade total com base nas similaridades locais de todos os atributos relevantes. O limiar de similaridade threshold é estabelecido como critério para essa decisão.

Essa função de similaridade global influencia diretamente a decisão final do sistema, permitindo que ele recupere casos que são considerados suficientemente semelhantes ao novo caso para preencher valores em falta. Ajustar o limiar de similaridade permite controlar a sensibilidade do sistema à semelhança entre casos.

Em resumo, as funções de similaridade local e global foram escolhidas e justificadas com base na natureza dos atributos envolvidos e na necessidade de encontrar casos semelhantes para preenchimento de valores em falta de forma eficaz e precisa, contribuindo assim para a qualidade do sistema de raciocínio baseado em casos.

9. ESTUDO E ANÁLISE DE REDES NEURONAIS

9.1 Análise do START

Analisando os resultados dos testes de configuração da rede neuronal, podemos observar várias conclusões importantes:

Impacto das Funções de Ativação: As diferentes combinações de funções de ativação nas camadas da rede neuronal têm um impacto significativo no desempenho do modelo. Por exemplo, a configuração com a função de ativação 'tansig' na camada escondida e 'purelin' na camada de saída alcançou uma precisão de 100%, enquanto outras combinações apresentaram resultados variados, com precisões entre 48,20% e 99,40%.

Variação da Precisão: A precisão do modelo varia consideravelmente entre as diferentes configurações testadas, indo de 48,20% a 100%. Isso destaca a importância de escolher cuidadosamente as configurações da rede neuronal para obter os melhores resultados possíveis.

Tempo de Execução: O tempo de execução também varia entre as diferentes configurações, com valores que vão de 0,01 segundos a 5,29 segundos. Configurações que alcançam maior precisão podem exigir mais tempo de treino, o que pode ser um fator a ser considerado dependendo dos requisitos de tempo do sistema.

Função de Treino: As diferentes funções de treino utilizadas também têm um impacto significativo no desempenho da rede neuronal. Por exemplo, configurações que utilizam a função de treino 'trainlm' tendem a ter tempos de execução mais curtos e, em alguns casos, maior precisão em comparação com outras funções de treino.

Efeito da Segmentação: Os testes foram realizados sem segmentação dos exemplos. Introduzir segmentação pode ser benéfico para melhorar o desempenho do modelo em conjuntos de dados complexos, dividindo-os em subconjuntos mais geríveis durante o treino.

Configuração					Média - 50 Iterações		
Número de Camadas Escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão	Tempo de Execução (s)	Erro
1	10	tansig, purelin	trainlm	S/ Segmentação	100.00 %	0.01	0.00
1	10	logsig, purelin	traingd	S/ Segmentação	85.60 %	0.20	0.12
1	10	tansig, softmax	trainlm	S/ Segmentação	50.00 %	0.01	0.13
1	10	poslin, radbas	trainr	S/ Segmentação	50.00%	5.29	0.17
1	10	satlin, tribas	trainscg	S/ Segmentação	50.00 %	0.13	0.15
1	10	hardlim, hardlims	traingdm	S/ Segmentação	49.40 %	0.01	0.51
1	10	satlins, softmax	trainbfg	S/ Segmentação	50.00 %	0.05	0.14
1	10	logsig, poslin	trainbr	S/ Segmentação	50.00 %	0.01	0.25
1	10	purelin, radbas	traincgf	S/ Segmentação	50.00 %	0.05	0.18
1	10	hardlim, satlin	traingd	S/ Segmentação	88.80 %	0.74	0.10
1	10	hardlims, tribas	traincgp	S/ Segmentação	50.00 %	0.01	0.22
1	10	radbasn, satlins	trains	S/ Segmentação	96.80 %	0.87	0.09
1	10	poslin, tansig	trainrp	S/ Segmentação	99.40 %	0.01	0.01
1	10	softmax, netinv	traincgb	S/ Segmentação	48.20 %	0.02	0.20

9.2 Análise do TRAIN

Para este estudo, foram realizados testes utilizando o ficheiro TRAIN, levando em consideração diversos parâmetros para configurar a rede neural. Esses parâmetros incluíram o nome da configuração, o número de camadas escondidas, o número de neurónios em cada camada, as funções de ativação utilizadas, a função de treino e a divisão dos exemplos (segmentação). Durante os testes, cada configuração foi submetida a 50 iterações, e os resultados foram registrados, incluindo a média das precisões global e de teste, o tempo de execução e o erro. Além disso, foram identificadas a melhor rede global e a melhor rede de teste, com base em critérios como precisão, tempo de execução e erro.

Configuração						Média - 50 Iterações			
Nome	Número de Camadas Escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro
Default	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	**77.57%**	**73.28%**	**0.04**	**0.18**

Melhor Rede - Global				
Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro	Iteração
81.15%	**71.74%**	**0.05**	**0.19**	**12**

Melhor Rede - Teste				
Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro	Iteração
76.89%	**81.52%**	**0.04**	**0.14**	**1**

9.2.1 O número e dimensão das camadas escondidas influencia o desempenho?

Após analisar os resultados dos testes com diferentes configurações de rede neuronal, concluímos que o número e dimensão das camadas escondidas influenciam significativamente o desempenho do modelo. Aqui estão as principais conclusões:

Precisão Global e de Teste: Houve um aumento geral na precisão global à medida que aumentamos o número de neurónios e camadas escondidas. No entanto, o aumento não foi linear, e observou-se uma diminuição na precisão nos dados de teste à medida que esses valores aumentavam.

Tempo de Execução: O tempo de execução aumentou consideravelmente conforme aumentamos o número de neurónios e camadas. Isso se deve ao aumento na complexidade computacional do treino das redes mais densas. Por exemplo, as configurações com 50 neurónios em duas camadas (cfg5) e três camadas (cfg10) apresentaram os maiores tempos de execução.

Erro: O erro do modelo também aumentou com o número de neurónios e camadas. Isso sugere que, embora a precisão global possa aumentar, o modelo pode ter dificuldade em generalizar para novos dados, resultando em um aumento no erro.

Melhores Redes Globais: Duas das três melhores redes globais totais (cfg5 e cfg10) apresentaram uma configuração com duas camadas escondidas e um número relativamente grande de neurónios (50 em cada camada). Isso sugere que, para este conjunto de dados e configuração de rede, uma arquitetura mais profunda e complexa pode ser mais eficaz em melhorar a precisão global do modelo.

Em suma, com base nos resultados obtidos, podemos concluir que o número e dimensão das camadas escondidas têm um impacto significativo no desempenho da rede neuronal, com um aumento na precisão global acompanhado por um aumento no tempo de execução e erro do modelo. Uma arquitetura mais profunda e complexa pode levar a uma melhoria na precisão, mas também pode exigir mais recursos computacionais.

O número e dimensão das camadas encondidas influencia o desempenho?						Média - 50 Iterações			
Nome	Número de Camadas Escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro
cfg1	2	5, 5	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	77.07%	74.57%	0.04	0.17
cfg2	2	10,10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	78.04%	72.91%	0.06	0.19
cfg3	2	15,15	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	78.55%	72.48%	0.12	0.20
cfg4	2	25,25	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	79.05%	70.63%	0.43	0.21
cfg5	2	50,50	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	81.43%	68.93%	10.24	0.25
cfg6	3	5, 5	tansig, tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	77.22%	72.63%	0.06	0.19
cfg7	3	10,10	tansig, tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	77.71%	72.24%	0.10	0.19
cfg8	3	15,15	tansig, tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	79.29%	71.50%	0.28	0.20
cfg9	3	25,25	tansig, tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	79.16%	69.59%	1.97	0.22
cfg10	3	50,50	tansig, tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	81.90%	71.33%	32.76	0.23

Melhor Rede - Global					Melhor Rede - Teste				
Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro	Iteração	Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro	Iteração
81.15%	70.65%	0.07	0.19	47	76.89%	81.52%	0.05	0.14	5
82.30%	72.83%	0.08	0.21	6	80.16%	82.61%	0.06	0.14	39
83.93%	73.91%	0.17	0.27	35	80.49%	79.35%	0.13	0.14	12
85.08%	77.17%	0.55	0.21	31	79.84%	79.35%	0.43	0.18	4
88.03%	73.91%	11.96	0.27	18	88.03%	79.35%	14.11	0.18	50
81.97%	70.65%	0.08	0.24	27	76.89%	81.52%	0.05	0.16	44
81.80%	76.09%	0.11	0.15	20	80.49%	81.52%	0.13	0.13	4
85.74%	77.17%	0.39	0.20	31	80.16%	79.35%	0.28	0.17	12
86.07%	70.65%	2.48	0.22	35	83.44%	81.52%	2.12	0.12	16
88.36%	72.83%	43.62	0.25	41	83.61%	84.78%	29.38	0.17	42

9.2.2 A função de treino influencia o desempenho?

Após analisar os resultados dos testes com diferentes funções de treino em uma rede neural com uma única camada escondida, observamos algumas tendências gerais que podem influenciar o desempenho do modelo:

Impacto na Precisão Global e de Teste: As diferentes funções de treino tiveram um impacto variado na precisão global e de teste do modelo. Enquanto algumas configurações alcançaram precisões relativamente altas, outras apresentaram resultados mais modestos. A precisão global variou entre aproximadamente 69% e 84%, e a precisão de teste variou entre cerca de 68% e 85%. Isso sugere que a escolha da função de treino pode ser crucial para obter um bom desempenho do modelo.

Variação no Tempo de Execução: Houve uma considerável variação no tempo de execução entre as diferentes funções de treino. Algumas configurações foram mais eficientes em termos de tempo, com tempos de execução na faixa de milissegundos, enquanto outras exigiram mais recursos computacionais, com tempos de execução na faixa de segundos. Isso pode ser importante considerando as restrições de tempo em diferentes aplicações.

Impacto no Erro do Modelo: O erro do modelo também variou entre as diferentes funções de treino. Configurações com menor precisão tendiam a ter um erro ligeiramente maior, indicando uma relação inversa entre precisão e erro. No entanto, algumas configurações conseguiram manter uma precisão relativamente alta com um erro moderado, enquanto outras tiveram dificuldade em generalizar para novos dados.

Destaque para Resultados Surpreendentes: Vale ressaltar a configuração cfg15, que se destacou como a terceira melhor rede global. Esta configuração obteve uma precisão global de 84.34%, superando significativamente outras configurações em termos de precisão. Isso sugere que a função de treino 'trainbr' pode ser especialmente eficaz para este conjunto de dados e arquitetura de rede, merecendo uma investigação mais aprofundada.

Em resumo, os resultados indicam que a escolha da função de treino pode ter um impacto significativo no desempenho do modelo de rede neural, influenciando sua precisão, tempo de execução e capacidade de generalização. Experimentar diferentes funções de treino é essencial para encontrar a configuração mais adequada para cada aplicação específica.

A função de treino influencia o desempenho?						Média - 50 Iterações			
Nome	Número de Camadas Escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro
cfg11	1	10	tansig, purelin	trains	dividerand = {0.7, 0.15, 0.15}	71.83%	71.83%	1.49	0.19
cfg12	1	10	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}	69.36%	68.46%	0.57	0.22
cfg13	1	10	tansig, purelin	trainbfg	dividerand = {0.7, 0.15, 0.15}	76.88%	74.74%	0.10	0.17
cfg14	1	10	tansig, purelin	traingdm	dividerand = {0.7, 0.15, 0.15}	69.73%	69.11%	0.59	0.21
cfg15	1	10	tansig, purelin	trainbr	dividerand = {0.7, 0.15, 0.15}	84.34%	70.98%	1.08	0.22
cfg16	1	10	tansig, purelin	traincgf	dividerand = {0.7, 0.15, 0.15}	76.66%	74.41%	0.05	0.17
cfg17	1	10	tansig, purelin	traingdx	dividerand = {0.7, 0.15, 0.15}	76.99%	74.93%	0.10	0.17
cfg18	1	10	tansig, purelin	trainoss	dividerand = {0.7, 0.15, 0.15}	76.47%	75.65%	0.06	0.17
cfg19	1	10	tansig, purelin	trainr	dividerand = {0.7, 0.15, 0.15}	76.68%	75.04%	5.14	0.17
cfg20	1	10	tansig, purelin	trainrp	dividerand = {0.7, 0.15, 0.15}	76.62%	74.54%	0.04	0.17

Melhor Rede - Global					Melhor Rede - Teste				
Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro	Iteração	Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro	Iteração
77.05%	80.43%	1.44	0.14	8	75.74%	82.61%	1.46	0.15	12
76.56%	75.00%	0.54	0.17	4	70.98%	78.26%	0.58	0.20	21
80.00%	77.17%	0.10	0.17	35	76.39%	84.78%	0.10	0.12	18
74.92%	79.35%	0.67	0.17	9	71.97%	80.43%	0.56	0.19	37
87.05%	72.83%	0.79	0.23	31	83.93%	81.52%	0.97	0.15	20
79.51%	69.57%	0.07	0.22	2	78.03%	84.78%	0.06	0.14	20
80.16%	72.83%	0.12	0.18	50	79.18%	84.78%	0.12	0.13	49
79.34%	76.09%	0.08	0.19	11	76.23%	85.87%	0.06	0.12	2
80.33%	71.74%	9.54	0.21	24	79.02%	82.61%	10.25	0.16	20
79.34%	69.57%	0.05	0.19	45	78.36%	83.70%	0.04	0.13	15

9.2.3 As funções de ativação influenciam o desempenho?

Com base nos resultados dos testes com diferentes configurações de funções de ativação em uma única camada escondida, podemos tirar algumas conclusões sobre o impacto dessas funções no desempenho do modelo:

Precisão Global e de Teste: A precisão global e de teste variou consideravelmente entre as diferentes configurações de funções de ativação. Observamos que algumas combinações, como cfg29 (purelin, netinv) e cfg25 (tansig, hardlim), apresentaram resultados bastante baixos em termos de precisão global e de teste, indicando que essas funções de ativação podem não ser adequadas para o conjunto de dados em questão.

Tempo de Execução: O tempo de execução para todas as configurações foi bastante baixo, variando entre 0,02 segundos e 0,07 segundos. Isso sugere que a escolha da função de ativação não teve um impacto significativo no tempo de treino da rede neuronal.

Erro: O erro do modelo também variou entre as diferentes configurações, mas não houve uma tendência clara de desempenho com base nas funções de ativação utilizadas. Algumas configurações apresentaram erros relativamente altos, como cfg29 (purelin, netinv) e cfg25 (tansig, hardlim), enquanto outras tiveram erros mais moderados.

Destaques: Notavelmente, as configurações cfg29 (purelin, netinv) e cfg25 (tansig, hardlim) destacaram-se negativamente, mostrando precisões globais e de teste significativamente mais baixas em comparação com outras configurações. No entanto, não houve configurações que se destacaram particularmente pela positiva em termos de precisão global ou de teste.

Em suma, os resultados indicam que a escolha da função de ativação pode ter um impacto significativo no desempenho da rede neuronal, com algumas funções sendo mais adequadas do que outras para o conjunto de dados em análise. No entanto, é importante realizar mais análises para determinar com mais precisão quais funções de ativação são as mais adequadas para o problema em questão.

As funções de ativação influenciam o desempenho?						Média - 50 Iterações			
Nome	Número de Camadas Escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro
cfg21	1	10	logsig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	77.24%	74.30%	0.04	0.17
cfg22	1	10	tansig, logsig	trainlm	dividerand = {0.7, 0.15, 0.15}	76.79%	73.91%	0.05	0.22
cfg23	1	10	satlin,tansig	trainlm	dividerand = {0.7, 0.15, 0.15}	77.39%	73.15%	0.05	0.18
cfg24	1	10	softmax, tansig	trainlm	dividerand = {0.7, 0.15, 0.15}	76.02%	72.15%	0.04	0.19
cfg25	1	10	tansig, hardlim	trainlm	dividerand = {0.7, 0.15, 0.15}	54.16%	54.33%	0.02	0.39
cfg26	1	10	losig, poslin	trainlm	dividerand = {0.7, 0.15, 0.15}	72.87%	69.15%	0.05	0.22
cfg27	1	10	tribas, radbas	trainlm	dividerand = {0.7, 0.15, 0.15}	73.52%	70.28%	0.05	0.22
cfg28	1	10	satlins, satlin	trainlm	dividerand = {0.7, 0.15, 0.15}	76.53%	72.22%	0.04	0.21
cfg29	1	10	purelin, netinv	trainlm	dividerand = {0.7, 0.15, 0.15}	51.34%	52.02%	0.07	0.84
cfg30	1	10	hardlims, softmax	trainlm	dividerand = {0.7, 0.15, 0.15}	68.25%	65.96%	0.04	0.28

Melhor Rede - Global					Melhor Rede - Teste				
Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro	Iteração	Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro	Iteração
79.67%	75.00%	0.05	0.19	4	77.54%	83.70%	0.04	0.14	22
80.16%	68.48%	0.06	0.23	44	78.20%	83.70%	0.05	0.21	9
82.62%	68.48%	0.07	0.20	15	78.20%	83.70%	0.04	0.13	5
82.79%	72.83%	0.08	0.19	1	76.56%	83.70%	0.04	0.13	36
62.13%	63.04%	0.02	0.28	47	58.20%	70.65%	0.02	0.44	20
81.31%	71.74%	0.05	0.22	30	77.87%	80.43%	0.04	0.19	18
80.00%	75.00%	0.05	0.21	3	77.87%	79.35%	0.05	0.21	1
79.67%	78.26%	0.05	0.20	10	77.38%	79.35%	0.04	0.20	30
63.44%	63.04%	0.04	0.33	47	60.66%	65.22%	0.04	0.25	21
73.11%	69.57%	0.04	0.26	37	71.48%	78.26%	0.03	0.24	40

9.2.4 A divisão de exemplos pelos conjuntos influencia o desempenho?

Após analisar os resultados dos testes com diferentes divisões dos exemplos pelos conjuntos de treino, validação e teste, podemos tirar algumas conclusões sobre o impacto dessa divisão no desempenho do modelo:

Precisão Global e de Teste: No geral, observamos que as diferentes divisões dos exemplos não tiveram um impacto significativo na precisão global do modelo. No entanto, duas configurações (cfg31 e cfg32) destacaram-se pela positiva, apresentando as duas melhores precisões no conjunto de teste. Isso sugere que uma divisão eficaz dos exemplos pode contribuir para um melhor desempenho do modelo em dados não vistos durante o treino.

Equilíbrio entre Precisão Global e de Teste: Notamos que, em todas as configurações, a diferença entre a precisão global e a precisão no conjunto de teste é relativamente pequena, o que indica um bom equilíbrio na capacidade do modelo de generalizar para novos dados.

Desempenho Consistente: Não houve configurações que se destacaram negativamente, sugerindo que todas as divisões dos exemplos testadas forneceram resultados consistentes e razoáveis em termos de precisão e erro.

Tempo de Execução: O tempo de execução permaneceu estável em todas as configurações, indicando que a divisão dos exemplos não teve um impacto significativo nesse aspeto.

Em resumo, embora algumas divisões dos exemplos tenham apresentado resultados ligeiramente melhores no conjunto de teste, todas as configurações foram bastante equilibradas em termos de desempenho, destacando a importância de escolher uma divisão adequada dos dados para o treino de redes neuronais.

A divisão de exemplos pelos conjuntos influencia o desempenho?						Média - 50 Iterações			
Nome	Número de Camadas Escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro
cfg31	1	10	tansig, purelin	trainlm	dividerand = {0.6, 0.3, 0.1}	77.24%	74.43%	0.04	0.17
cfg32	1	10	tansig, purelin	trainlm	dividerand = {0.9, 0.05, 0.05}	78.12%	72.65%	0.04	0.19
cfg33	1	10	tansig, purelin	trainlm	dividerand = {0.8, 0.05, 0.15}	77.88%	74.41%	0.04	0.18
cfg34	1	10	tansig, purelin	trainlm	dividerand = {0.5, 0.35, 0.15}	76.93%	73.83%	0.04	0.18
cfg35	1	10	tansig, purelin	trainlm	dividerand = {0.4, 0.3, 0.3}	76.33%	72.91%	0.04	0.19
cfg36	1	10	tansig, purelin	trainlm	dividerand = {0.6, 0.2, 0.2}	77.02%	73.48%	0.04	0.18
cfg37	1	10	tansig, purelin	trainlm	dividerand = {0.4, 0.2, 0.4}	75.56%	72.22%	0.04	0.19
cfg38	1	10	tansig, purelin	trainlm	dividerand = {0.5, 0.3, 0.2}	76.42%	73.16%	0.04	0.18
cfg39	1	10	tansig, purelin	trainlm	dividerand = {0.5, 0.2, 0.3}	76.86%	72.86%	0.04	0.18
cfg40	1	10	tansig, purelin	trainlm	dividerand = {0.5, 0.15, 0.35}	76.51%	72.93%	0.04	0.18

Melhor Rede - Global					Melhor Rede - Teste				
Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro	Iteração	Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro	Iteração
79.67%	72.13%	0.04	0.16	50	77.05%	86.89%	0.03	0.13	4
85.41%	70.97%	0.06	0.22	17	78.85%	87.10%	0.03	0.11	28
82.79%	64.13%	0.06	0.25	14	80.33%	84.78%	0.04	0.14	30
79.51%	81.52%	0.04	0.16	46	79.51%	81.52%	0.04	0.16	46
78.69%	76.50%	0.04	0.16	27	76.72%	81.42%	0.03	0.15	15
79.34%	72.13%	0.03	0.18	8	77.70%	83.61%	0.03	0.13	45
78.36%	73.77%	0.04	0.19	29	77.05%	78.28%	0.04	0.18	4
78.85%	66.39%	0.04	0.22	4	75.74%	81.15%	0.04	0.13	31
79.67%	73.77%	0.05	0.21	37	78.36%	79.78%	0.04	0.16	8
79.51%	74.30%	0.05	0.22	38	78.20%	78.04%	0.04	0.15	11

9.3 Análise do TEST

Depois de aplicar as melhores redes previamente treinadas no conjunto de dados de teste, os resultados são os seguintes:

Melhores Redes para Generalização: As melhores redes globais (cfg5, cfg10, cfg15) apresentaram uma precisão global média de 87,48%, com precisão média nos testes de 73,19%. Elas demonstraram uma capacidade sólida de generalização e aprendizagem.

Melhores Desempenhos Individuais: A melhor rede global foi a cfg10, com uma precisão global de 88,36% e precisão nos testes de 72,83%. No entanto, o seu tempo de execução foi significativamente longo, 43,62 segundos. A melhor rede nos testes foi a cfg32, com uma precisão global de 78,85% e uma precisão nos testes de 87,10%, com um tempo de execução de 0,03 segundos.

Comparação com a Base: Comparando com a precisão média das configurações de base (53,33% para globais e 70% para testes), todas as melhores redes mostraram melhorias substanciais na precisão, evidenciando a sua eficácia na aprendizagem e generalização. Notavelmente, as melhores redes nos testes superaram as globais em precisão média, com 70% para as redes de teste e 53,33% para as globais.

Precisão no Conjunto de Dados de Teste: A precisão média das redes nos testes no conjunto de dados de teste foi de 86,28%, destacando a sua capacidade de generalização para novos dados.

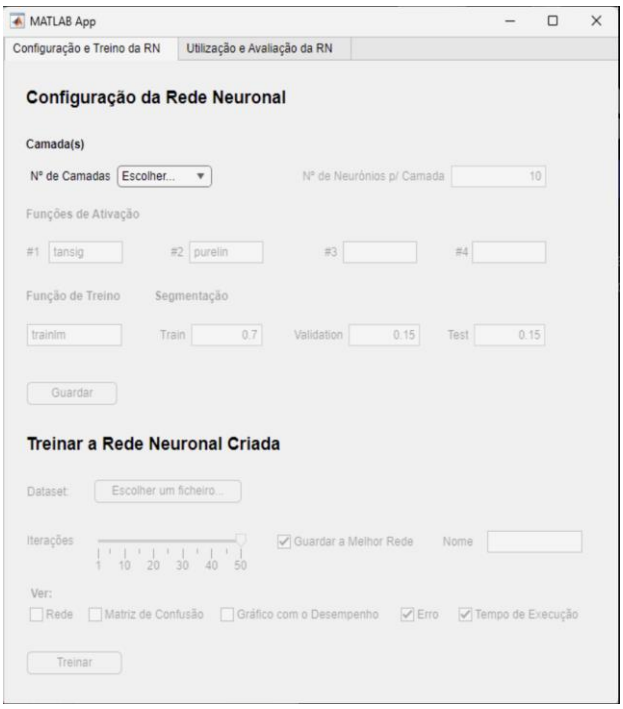
Em suma, as melhores redes treinadas mostraram uma capacidade robusta de generalização, com melhorias notáveis na precisão em comparação com as configurações de base. Isso destaca a sua eficácia em aprender e representar os padrões nos dados de teste.

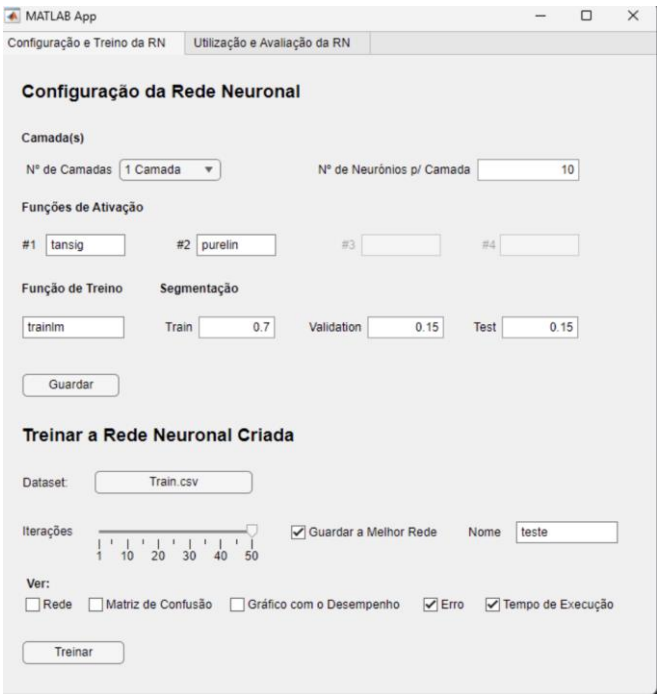
Configuração						Melhor Rede - Global					Melhor Rede - Teste					Start.csv	
Nome	Número de Camadas Escondidas	Número de Neurónios	Funções de Ativação	Função de Treino	Divisão dos Exemplos	Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro	Iteração	Precisão Global	Precisão Teste	Tempo de Execução (s)	Erro	Iteração	Precisão	Médias
cfg5	2	50,50	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	88.03%	73.91%	11.96	0.27	18	---	---	---	---	---	50.00%	
cfg10	3	50,50	tansig, tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	88.36%	72.83%	43.62	0.25	41	---	---	---	---	---	60.00%	53.33 %
cfg15	1	10	tansig, purelin	trainbr	dividerand = {0.7, 0.15, 0.15}	87.05%	72.83%	0.79	0.23	31	---	---	---	---	---	60.00%	
cfg18	1	10	tansig, purelin	trainoss	dividerand = {0.7, 0.15, 0.15}	---	---	---	---	---	76.23%	85.87%	0.06	0.12	2	60.00%	
cfg31	1	10	tansig, purelin	trainlm	dividerand = {0.6, 0.3, 0.1}	---	---	---	---	---	77.05%	86.89%	0.03	0.13	4	80.00%	70%
cfg32	1	10	tansig, purelin	trainlm	dividerand = {0.9, 0.05,	---	---	---	---	---	78.85%	87.10%	0.03	0.11	28	70.00%	

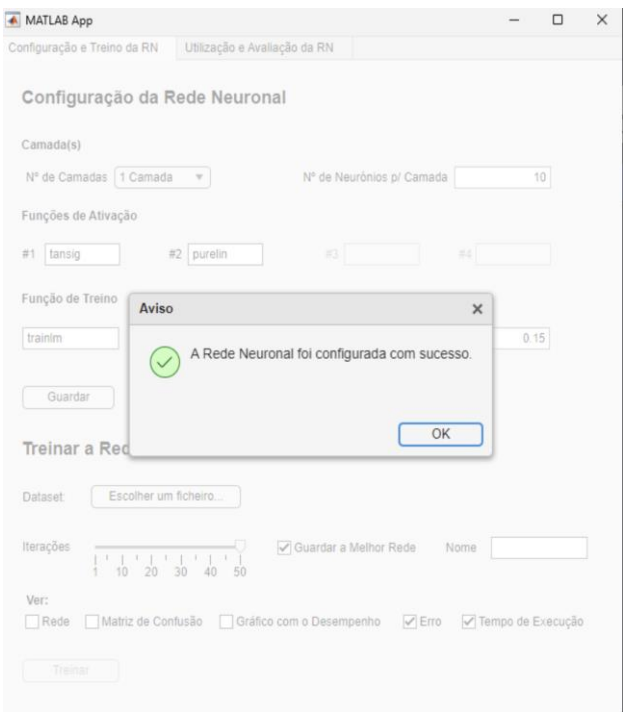
10. APLICAÇÃO GRÁFICA

Nesta seção do relatório, descreveremos o desenvolvimento de uma aplicação gráfica simples em Matlab, destinada a oferecer funcionalidades essenciais para a criação, treino e utilização de redes neuronais. A aplicação foi concebida para proporcionar uma interface intuitiva e interativa, permitindo aos utilizadores configurar uma rede neuronal com flexibilidade, ajustando o número de camadas, neurónios por camada, funções de ativação, métodos de treinamento e rácios de segmentação.

Além disso, a aplicação permite carregar redes neuronais previamente treinadas e armazenadas em disco, obtidas durante o treino com um conjunto de dados específico. Essas redes podem ser utilizadas para realizar previsões e avaliar o desempenho da rede neuronal em novos datasets, mostrando os resultados obtidos.







11. CONCLUSÕES

Neste trabalho, além de explorar a relevância do dataset escolhido para o estudo de Acidente Vascular Cerebral (AVC), foram obtidas conclusões importantes sobre a preparação e aplicação dos dados. A implementação do Sistema de Raciocínio Baseado em Casos revelou-se uma abordagem eficaz para lidar com valores ausentes, destacando a importância da fase de RETRIEVE na identificação de casos similares para inferência.

A justificação dos pesos atribuídos aos atributos evidenciou a necessidade de considerar a relevância de cada um para uma análise precisa. Além disso, as funções de similaridade local e global desempenharam papéis cruciais na determinação da similaridade entre casos, contribuindo para a qualidade do preenchimento de valores em falta e para a eficácia do sistema como um todo.

Ao avaliar os resultados obtidos com o uso de redes neuronais, constatou-se a importância da preparação adequada dos dados, visto que os desempenhos diferiram entre os conjuntos START e TRAIN. Essa análise permitiu extrair insights valiosos sobre a eficácia das redes neuronais feedforward na classificação de casos de AVC, ressaltando a necessidade de uma abordagem cuidadosa na seleção e preparação dos dados.

Em suma, este trabalho não apenas abordou a importância do dataset e as etapas de preparação, mas também proporcionou insights significativos sobre a aplicação prática de técnicas de análise de dados, destacando a relevância do Sistema de Raciocínio Baseado em Casos e das escolhas realizadas durante o processo de modelagem e análise.