UNIVERSIDADE FEDERAL DO PARÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DIRETORIA DE PESQUISA

PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO CIENTÍFICA –
PIBIC/UFPA, PIBIC/UFPA CAMPI DO INTERIOR, PIBIC/UFPA EBTT, PIBIC-AF/UFPA, PIBIC/CNPq, PIBIC-AF/CNPq, PIBITI/CNPq, PRODOUTOR e PRODOUTOR/RENOVAÇÃO.

RELATÓRIO TÉCNICO - CIENTÍFICO PARCIAL

Período: 01/08/2019 a 31/07/2020

IDENTIFICAÇÃO DO PROJETO

Título do Projeto de Pesquisa (ao qual está vinculado o Plano de Trabalho): Deep Learning aplicado a predição de séries temporais

Nome do Orientador: Lidio Mauro Lima de Campos

Titulação do Orientador: Doutor

Faculdade: Faculdade de Computação

Instituto/Núcleo: ICEN

Laboratório: Computação Bioinspirada

Título do Plano de Trabalho: Aplicações de técnicas de mineração de dados em problemas de reconhecimento de padrões

Nome do Bolsista: Yann Fabricio Cardoso de Figueiredo

Tipo de Bolsa:

(X) PIBIC/UFPA
( ) PIBIC/UFPA CAMPI DO INTERIOR,
( ) PIBIC/UFPA EBTT
( ) PIBIC-AF/UFPA
( ) PIBIC/CNPq
( ) PIBIC-AF/CNPq

( ) PIBITI/CNPq
( ) PIBIC PRODOUTOR
( ) PIBIC PRODOUTOR/RENOVAÇÃO

- **Atenção: No relatório, aborde diretamente os pontos essenciais, a partir dos quais será avaliado o desenvolvimento do projeto.**

- **O relatório não deverá ultrapassar 10 MB ou conter mais de vinte (20) páginas.**

1. INTRODUÇÃO

O projeto teve como ponto de partida o estudo da literatura sobre Redes Neurais Artificiais (RNAs), focando no algoritmo *backpropagation* com o uso de rede direta (3 e 4 camadas) e redes recorrentes. Após o estudo inicial foram feitos experimentos com o algoritmo *backpropagation* escrito em linguagem Python (rede direta) e C++ (rede recorrente), e posteriormente foram adaptados todos os algoritmos para Python. Antes dos experimentos foi necessário fazer um pré-processamento de dados utilizando algoritmos implementados em Python. Os experimentos consistiram em fazer o processo de classificação com base nas técnicas *hold-out* e *k-fold cross validation*, sendo utilizados como fonte de dados os *datasets hepatitis* e *german credit*.

2. COMPARAÇÃO ENTRE O PLANO ORIGINAL E O EXECUTADO

O projeto teve início no começo de agosto de 2019, iniciando com um estudo da literatura associada a Redes Neurais Artificiais. A etapa de estudo consistiu em conhecer o histórico, técnicas, função de ativação, tipos de aprendizado e algoritmos de RNA (Rede Neural Artificial). Após essa etapa inicial estudou-se o modelo matemático de aprendizado *backpropagation*, sendo disponibilizado, inicialmente, um algoritmo em linguagem C que trabalhava com uma arquitetura de 3 camadas da RNA. Para primeiros experimentos foram utilizadas bases de dados simples, como a tabela verdade XOR, para entender o funcionamento do algoritmo.

Já no mês de setembro foi utilizado o primeiro *dataset*, chamado de *hepatitis*, para experimentos mais completos, essa base de dados foi retirada do repositório público U*C Irvine Machine Learning Repository at the University of California* (http://archive.ics.uci.edu/ml/datasets.html). Com esse *dataset*, no primeiro experimento, utilizaram-se o algoritmo *backpropagation* 3 camadas juntamente do *backpropagation* 4 camadas também disponibilizado em C, sendo que ambos foram convertidos para a linguagem Python. As técnicas adotadas foram *hold-out* e *k-fold cross validation*, porém a primeira obteve resultados melhores e portanto, foi a técnica de validação adotada para os experimentos posteriores. Os experimentos com essa base de dados estenderam-se até outubro, utilizando, além dos algoritmos citados, o *backpropagation* com atraso no tempo (BPTT) para redes recorrentes, disponibilizado na linguagem C++.

O experimento com o *hepatitis* consistiu em configurar os parâmetros do algoritmo para aperfeiçoar o treinamento do *dataset*, onde foram configuradas variáveis referentes ao

número de neurônios da camada interna (variando dependendo do número de camadas), épocas e taxa de aprendizado. As mudanças desses parâmetros são para obter cálculos de pesos distintos com base nos neurônios, a cada camada da arquitetura em uso, que proporcionem uma acurácia melhor ao testar o algoritmo após o aprendizado do mesmo.

A partir do fim de outubro de 2019 até janeiro de 2020 foi utilizado o *dataset german credit* para novos experimentos, sendo que, assim como no *hepatitis*, este também conta com uma quantidade pequena de registros e portanto os testes tiveram melhores resultados com o uso da técnica *hold-out* para classificar a base de dados, onde a acurácia maior aparece sempre quando aumenta-se o conjunto de dados utilizado na etapa de treinamento do algoritmo utilizado.

3. OUTRAS ATIVIDADES

Desenvolvimento de algoritmos, escritos na linguagem Python, para tratamento de dados, com o intuito de adequar um *dataset* e utilizá-lo de forma correta nos códigos de *backpropagation* trabalhados durante o projeto. O tratamento consiste em normalizar dados, aplicar técnicas (*hold-out* e *k-fold*) para divisão de *dataset*, balancear a base de dados e eliminar *outliers*.

4. ATIVIDADES REALIZADAS

4.1. Hepatitis

4.1.1. *Obtenção e descrição da base de dados*

Este *dataset* contém no total 19 atributos relacionados a uma pessoa com hepatite, sendo que com a análise deles é definido se o indivíduo vai continuar vivo. É uma base de dados pequena e ficou ainda menor durante o projeto, pois devido a ausência de alguns dados em certos registros foi necessário excluir estes para não afetar a aprendizagem e classificação do conjunto de dados.

Nos experimentos de classificação com essa base de dados foram utilizados algoritmos de *backpropagation* direto com 3 e 4 camadas, e ainda foi utilizado também a versão recorrente do *backpropagation*. Esse *dataset*, utilizado durante a primeira fase do projeto, foi obtido do repositório: https://archive.ics.uci.edu/ml/datasets/hepatitis.

O conjunto de dados têm um total de 155 registros, sendo que após a eliminação daqueles com valores ausentes o total foi reduzido para 80. A tabela 1 mostra o que significa cada atributo do *dataset hepatitis*. No conjunto original haviam 123 pessoas definidas que iriam viver e 32 iriam morrer, após a eliminação de registros, o total de vivos e mortos ficou respectivamente 67 e 13.

Atributos	
1	Classe (vivo ou morto)
2	Idade
3	Sexo

4	Esteróide
5	Antivirais
6	Fadiga
7	Mal-estar
8	Anorexia
9	Fígado grande
10	Fígado empresarial
11	Baço palpável
12	Aranhas
13	Ascites
14	Varizes
15	Bilirrubina
16	Fosfato alcalino
17	SGOT (Soro Transaminase Glutamo-Oxalacético)
18	Albumina
19	Protime
20	Histologia

Tabela 1 - Descrição dos atributos de *hepatitis*

4.1.2. *Pré-processamento e normalização de dados*

Todos os algoritmos de *backpropagation* utilizados utilizam uma função de ativação chamada de sigmóide e esta trabalha com valores no intervalo entre 0 e 1, então, portanto foi necessário criar formas para normalizar os dados do *dataset* e adequar ao uso para que as simulações pudessem ser feitas. Os dados do *hepatitis* encontravam-se como descrito na amostra abaixo:

2,30,1,1,2,1,1,2,2,1,2,1,2,2,0.80,147,128,3.9,100,2
2,30,1,1,2,2,2,2,2,2,2,2,2,2,0.70,100,31,4.0,100,1
1,51,1,1,2,1,2,1,2,2,1,1,2,2,?,?,?,?,?,1
2,23,1,2,2,1,1,1,2,2,1,2,2,2,1.30,194,150,4.1,90,1
1,47,1,2,2,1,1,2,2,1,2,2,1,1,1.70,86,20,2.1,46,2

Após usar como entrada essa estrutura apresentada o algoritmo de tratamento faz a normalização para que os dados possam ser aceitos pelos códigos utilizados durante o projeto. O conjunto é adaptado para ser um arquivo python que será chamado por um dos dois outros algoritmos criados para dividir o *dataset* seguindo alguma das técnicas já mencionadas, além de balancear os valores da saída para que fique parecida a proporção entre treino e teste. O *dataset* normalizado ficou como descrito abaixo, onde x é referente aos valores de entrada e y é a saída da RNA (Rede Neural Artificial):

```
x[0][0] = 1.0;
x[0][1] = 0.2;
x[0][2] = 0.3;
x[0][3] = 0.1;
x[0][4] = 0.1;
x[0][5] = 0.2;
x[0][6] = 0.1;
x[0][7] = 0.1;
x[0][8] = 0.2;
x[0][9] = 0.2;
x[0][10] = 0.1;
x[0][11] = 0.2;
x[0][12] = 0.1;
x[0][13] = 0.2;
x[0][14] = 0.2;
x[0][15] = 0.8;
x[0][16] = 0.147;
x[0][17] = 0.128;
x[0][18] = 0.39;
x[0][19] = 0.1;

y[0][0] = 0.2;
```

4.1.3. *Ajustes dos parâmetros e treinamento da rede neural*

Os dados pré-processados e normalizados são utilizados agora para treinar a RNA com auxílio do algoritmo *backpropagation* direto e recorrente. Como já mencionado o *dataset* em uso possui uma quantidade bem pequena de registros que ficou ainda menor após a remoção daqueles cujo algum atributo possuía valor vazio. Tendo em vista esse tamanho do conjunto de dados, 80 registros após o pré-processamento, optou-se por utilizar a técnica *hold-out* para dividir o *dataset*. Para as primeiras simulações foi utilizada a proporção padrão da técnica em uso, que é dividir a base de dados em 2/3 para treino e 1/3 para teste, ficando então 53 registros para treinamento da rede e os outros 27 para testar. Na tabela 2 estão os resultados das simulações feitas na rede direta de 3 camadas, na tabela 3 os da rede direta de 4 camadas e na tabela 4 estão os resultados da rede recorrente.

Cada parâmetro das redes foi sendo alterado e fixado caso fosse o melhor dentre os testados, e essa técnica seguiu até o fim das simulações para encontrar o melhor resultado possível. Como exemplo do método de testes citado pode-se citar: começou-se com NINT (número de neurônios da camada interna) igual a 4, TAPR (taxa de aprendizagem) igual a 0.9 e 250000 épocas. Simulou-se com esses parâmetros iniciais e depois mudou-se o NINT, esse processo foi repetido "n" vezes até definir-se quais parâmetros da rede proporcionariam melhor acurácia no modelo de rede neural. Após encontrar o melhor NINT modificou-se a taxa de aprendizagem e posteriormente o número de épocas, esse método foi adotado em todas as redes, com seus devidos parâmetros, para encontrar os melhores parâmetros de cada RNA e consequentemente os melhores parâmetros para obter um resultado satisfatório de classificação do *dataset hepatitis*.

De forma geral, em cada rede, utilizou-se 20 neurônios na camada de entrada (19 atributos e 1 BIAS), uma para cada atributo apresentado, na Tabela 1. Além disso, 1 neurônio da camada de saída, correspondente ao atributo de classificação, vivo ou morto, e na(s) camada(s) oculta(s) o valor variou de acordo com cada rede e simulação.

Erros	Acertos	NINT	TAPR	Épocas	Acurácia	EMQ
5	22	4	0.9	250000	81.49%	1.08e-05
5	22	5	0.9	250000	81.49%	1.11e-05
5	22	6	0.9	250000	81.49%	1.18e-05
5	22	7	0.9	250000	81.49%	6.78e-06
5	22	8	0.9	250000	81.49%	8.08e-06
3	24	9	0.9	250000	88.89%	8.21e-06
5	22	10	0.9	250000	81.49%	8.23e-06
3	24	9	0.8	250000	88.89%	7.26e-06
5	22	9	0.7	250000	81.49%	6.63e-06
3	24	9	0.8	300000	88.89%	8.12e-06
4	23	9	0.8	350000	85.19%	1.12e-05
3	24	9	0.8	400000	88.89%	1.30e-05
4	23	9	0.8	450000	85.19%	1.23e-05
4	23	9	0.8	500000	85.19%	1.49e-05

Tabela 2 - Resultados da rede de 3 camadas (Valores obtidos com a base de testes, após o treinamento)

Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
4	23	9	8	0.3	0.5	300000	85.19%	0.000635
4	23	5	8	0.3	0.5	300000	85.19%	0.000635
4	23	6	8	0.3	0.5	300000	85.19%	0.000638
4	23	10	9	0.3	0.5	300000	85.19%	0.000635
4	23	11	10	0.3	0.5	300000	85.19%	0.000633
4	23	12	11	0.3	0.5	300000	85.19%	0.000631
4	23	12	11	0.4	0.5	300000	85.19%	0.000630
4	23	12	11	0.5	0.5	300000	85.19%	0.000630
4	23	12	11	0.6	0.5	300000	85.19%	0.000630
4	23	12	11	0.7	0.5	300000	85.19%	0.000630
4	23	12	11	0.9	0.5	300000	85.19%	0.000630
4	23	12	11	0.9	0.7	300000	85.19%	0.000628
4	23	12	11	0.9	0.9	300000	85.19%	0.000627
4	23	12	11	0.9	0.7	350000	85.19%	0.000626
4	23	12	11	0.9	0.7	400000	85.19%	0.000641
3	24	12	11	0.9	0.7	450000	88.89%	0.000592
5	22	12	11	0.9	0.7	500000	81.49%	0.000640

Tabela 3 - Resultados da rede de 4 camadas (Valores obtidos com a base de testes, após o treinamento)

Erros	Acertos	NINT	TAPR	Épocas	Acurácia	EMQ
5	22	5	0.9	200000	81.48%	0.000580
5	22	6	0.9	200000	81.48%	0.000600
6	21	7	0.9	200000	77.78%	0.000649
6	21	8	0.9	200000	77.78%	0.000734
5	22	9	0.9	200000	81.48%	0.000608
5	22	10	0.9	200000	81.48%	0.000625

5	22	5	0.8	200000	81.48%	0.000574
5	22	5	0.7	200000	81.48%	0.000568
5	22	5	0.7	250000	81.48%	0.000580
5	22	5	0.7	300000	81.48%	0.000583
5	22	5	0.7	350000	81.48%	0.000599
5	22	5	0.7	450000	81.48%	0.000611

Tabela 4 - Resultados da rede recorrente (Valores obtidos com a base de testes, após o treinamento)

Observando as tabelas 2, 3 e 4 é possível verificar que a rede direta de 3 camadas foi responsável por proporcionar resultados mais variados, pois as redes diretas de 4 camadas e recorrente ficaram praticamente estabilizadas, mesmo com inúmeras mudanças de parâmetros, em respectivamente 85.18% e 81.48% de acurácia nas simulações feitas. De forma geral, todas as redes variam pouco, devido o tamanho pequeno do *dataset* pós pré-processamento de dados, mas a rede direta de 3 camadas destacou-se mais por conta de uma variação maior e por ter alcançado em mais de uma oportunidade o melhor resultado observado nas simulações feitas, que foi de 88.89.

Para novos experimentos foi feito um novo tratamento no dataset quanto aos valores ausentes, onde somente os registros que tinham muitos atributos com valor vazio foram excluídos e no restante foi feita a adição de valores, baseados nos dos outros registros (usando como base os valores preenchidos no atributo de outros registros foi feita uma seleção aleatória para pegar um desses possíveis valores e colocar no atributo ausente de algum registro), para ocupar os espaços vazios. Após esse tratamento o *hepatitis* ficou com 146 registros, onde 117 eram referentes a pessoa que viveriam e 29 a pessoas que morreriam. Foram feitas simulações com a proporção padrão do *hold-out*, porém foram abaixo do esperado comparando com as tabelas de resultados mostradas anteriormente. Na tentativa de encontrar melhores resultados optou-se por alterar a proporção padrão de divisão da técnica *hold-out*, passou a ser 80% para treino e 20% para teste.

Na tabela 5 é possível ver um resumo dos resultados alcançados utilizando a rede recorrente, pois a mesma respondeu melhor ao novo dataset e conseguiu alcançar uma acurácia superior ao das simulações mostradas nas tabelas 2, 3 e 4. Foram feitas várias simulações, mas ficaram estáveis variando a acurácia entre 83% e 86.67%, portanto mostrou-se mais conveniente apresentar um resumo contendo a configuração que fez a rede chegar ao melhor desempenho durante as atividades feitas com o *dataset hepatitis*, que foi de 90% após usar um número bem elevado de épocas (1000000 e 1050000).

Erros	Acertos	NINT	TAPR	Épocas	Acurácia	EMQ
4	26	9	0.9	900000	86.67%	0.000519

4	26	9	0.9	950000	86.67%	0.000516
3	27	9	0.9	1000000	90.00%	0.000516
3	27	9	0.9	1050000	90.00%	0.000516
4	26	9	0.9	1100000	86.67%	0.000517

Tabela 5 - Resultados da rede recorrente pós segundo tratamento de dados (Valores obtidos com a base de testes, após o treinamento)

4.1.4. *Testes das RNAs*

A acurácia apontada nas tabelas mostradas anteriormente é baseada na proporção de acertos que a rede teve na classificação dos valores de saída (0.1 representando pessoa morta e 0.2 pessoa viva). Para definir que um valor calculado, com base nos pesos das camadas, no algoritmo utilizado podia ser dito como correto foi definido uma faixa de aceitação para cada valor de saída, onde qualquer valor acima ou igual a 0.05 e menor que 0.15 era dito como certo para o valor 0.1, e ainda qualquer valor acima ou igual a 0.15 e menor que 0.25 era dito como certo para o valor 0.2.

Após passar pela etapa de treinamento, com a devida parcela do dataset definida para esse fim, a base de dados para teste é usada para checar a confiabilidade de classificação do conjunto de dados e consequentemente resulta em um teste de generalização como mostrado no exemplo abaixo:

VERIFICACAO

Padrao Testes>>0
calculado>>0.19383 desejado>>0.2 Erro>>1.9034e-05
Padrao Testes>>1
calculado>>0.184682 desejado>>0.2 Erro>>0.000117318
Padrao Testes>>2
calculado>>0.19012 desejado>>0.2 Erro>>4.88096e-05
Padrao Testes>>3
calculado>>0.205044 desejado>>0.2 Erro>>1.27189e-05
Padrao Testes>>4
calculado>>0.208179 desejado>>0.2 Erro>>3.34471e-05
Padrao Testes>>5
calculado>>0.154893 desejado>>0.2 Erro>>0.00101732
Padrao Testes>>6
calculado>>0.188633 desejado>>0.2 Erro>>6.46006e-05
Padrao Testes>>7
calculado>>0.103038 desejado>>0.1 Erro>>4.6152e-06
Padrao Testes>>8
calculado>>0.191877 desejado>>0.2 Erro>>3.29888e-05

Padrao Testes>>9
calculado>>0.203332 desejado>>0.2 Erro>>5.55074e-06
Padrao Testes>>10
calculado>>0.195166 desejado>>0.2 Erro>>1.16817e-05
Padrao Testes>>11
calculado>>0.199794 desejado>>0.2 Erro>>2.12331e-08
Padrao Testes>>12
calculado>>0.191193 desejado>>0.2 Erro>>3.8781e-05
Padrao Testes>>13
calculado>>0.196951 desejado>>0.2 Erro>>4.64719e-06
Padrao Testes>>14
calculado>>0.200633 desejado>>0.2 Erro>>2.00267e-07
Padrao Testes>>15
calculado>>0.19316 desejado>>0.1 Erro>>0.00433937
Padrao Testes>>16
calculado>>0.149151 desejado>>0.1 Erro>>0.00120791
Padrao Testes>>17
calculado>>0.0994101 desejado>>0.1 Erro>>1.74001e-07
Padrao Testes>>18
calculado>>0.191804 desejado>>0.2 Erro>>3.35888e-05
Padrao Testes>>19
calculado>>0.208164 desejado>>0.2 Erro>>3.33243e-05
Padrao Testes>>20
calculado>>0.180046 desejado>>0.1 Erro>>0.00320366
Padrao Testes>>21
calculado>>0.207975 desejado>>0.2 Erro>>3.18035e-05
Padrao Testes>>22
calculado>>0.188327 desejado>>0.2 Erro>>6.81272e-05
Padrao Testes>>23
calculado>>0.186524 desejado>>0.2 Erro>>9.08044e-05
Padrao Testes>>24
calculado>>0.195815 desejado>>0.2 Erro>>8.75642e-06
Padrao Testes>>25
calculado>>0.113861 desejado>>0.2 Erro>>0.00370996
Padrao Testes>>26
calculado>>0.154368 desejado>>0.2 Erro>>0.00104114
Padrao Testes>>27
calculado>>0.197456 desejado>>0.2 Erro>>3.23685e-06
Padrao Testes>>28
calculado>>0.109839 desejado>>0.1 Erro>>4.84003e-05
Padrao Testes>>29
calculado>>0.177932 desejado>>0.2 Erro>>0.000243509

4.1.5. *Resumo dos resultados*

4.1.5.1. *Resultados de simulações do dataset balanceado e registros com valores ausentes eliminados*

Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
3	24	9		0.8		400000	88.89%	1.30e-05
3	24	9		0.8		250000	88.89%	7.26e-06
3	24	9		0.8		3000000	88.89%	8.12e-06
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
3	24	12	11	0.9	0.7	450000	88.89%	0.000592
4	23	12	11	0.9	0.7	350000	85.19%	0.000626
4	23	12	11	0.9	0.9	300000	85.19%	0.000627
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
5	22	5		0.9		200000	81.48%	0.000580
5	22	5		0.7		200000	81.48%	0.000568
5	22	5		0.8		200000	81.48%	0.000574

Legendas: RND 3 camadas (amarelo), RND 4 camadas (laranja) e RNR (verde)

Na RND de 3 camadas todos os melhores resultados chegaram a 88.89% de acurácia, e é possível ver que quanto maior o número de épocas menor o erro quadrático médio. A RND de 4 camadas teve pouca variação de parâmetros nos melhores resultados, possui o mesmo número de neurônios nas camadas internas, na taxa de aprendizagem varia apenas no resultado com maior EMQ (erro quadrático médio), onde a taxa de aprendizagem 2 é 0.9, e no número de épocas, assim como na RND de 3 camadas, a maior quantidade resultou no melhor desempenho (88.89%).

4.1.5.1. *Resultados de simulações do dataset balanceado e com novo tratamento quanto aos registros com valores ausentes*

Erros	Acertos	NINT	TAPR	Épocas	Acurácia	EMQ
4	26	9	0.9	950000	86.67%	0.000516
3	27	9	0.9	1000000	90.00%	0.000516
3	27	9	0.9	1050000	90.00%	0.000516

Nesta etapa do hepatitis foi exposto apenas os melhores resultados da rede recorrente, pois obteve melhor desempenho. Pelos resultados apresentados é possível ver a influência do aumento de épocas resultando na melhor acurácia (90%).

4.2. German Credit

4.2.1. *Obtenção e descrição da base de dados*

O segundo *dataset* selecionado foi obtido no mesmo repositório do primeiro, mais especificamente de: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data). O *german credit* possui 21 atributos e 1000 registros, porém ele contém valores categóricos e simbólicos. O fato de ele possuir esses tipos, diferentes, de atributos faz com os mesmos não possam ser utilizados nos algoritmos trabalhados no projeto, pois as redes neurais utilizadas trabalham com valores numéricos. Portanto foi utilizada a versão alternativa de *german credit* disponibilizada no mesmo repositório da versão original, onde há 25 atributos com a adição de variáveis indicadoras e codificações de atributos categóricos ou simbólicos para valores numéricos.

O *german credit* é um *dataset* para análise de risco de crédito com base em atributos de clientes como, por exemplo, o histórico de crédito e status da conta corrente. Abaixo encontra-se a descrição dos 21 atributos da versão original:

Atributos	
1	Status da conta corrente
2	Duração em meses (tempo de conta no banco)
3	Histórico de crédito
4	Motivo de crédito
5	Valor total de crédito
6	Status de conta poupança
7	Duração do emprego atual
8	Taxa de parcelamento
9	Sexo - Estado civil
10	Outros devedores/fiadores
11	Duração residência atual

12	Propriedade
13	Idade
14	Outros planos de parcelamento
15	Habitação
16	Número de créditos existentes neste banco
17	Emprego
18	Número de pessoas responsáveis por fornecer manutenção
19	Telefone
20	Trabalhador estrangeiro
21	Risco de crédito

Tabela 6 - Atributos do dataset german credit

4.2.2. *Pré-processamento e transformação dos dados*

Este dataset teve o mesmo tratamento do anterior, inicialmente, foram realizadas a normalização e o balanceamento, para darmos o prosseguimento correto dos experimentos com *backpropagation* utilizando RNA direta e recorrente. Como já mencionado o *dataset* original foi impossível de ser usado por conta dos atributos categóricos, portanto o pré-processamento e a transformação de dados foram feitas na versão alternativa da base de dados que continha apenas valores numéricos.

Outra etapa do pré-processamento de dados realizado no *dataset*, envolveu a eliminação estatística de *outliers* com o intuito de melhorar a classificação do conjunto de dados utilizado. Ainda foi necessário transformar os valores de saída, que antes eram 0.1 (risco bom) e 0.2 (risco ruim) e depois mudaram para 0.0 e 1.0 com o intuito de aumentar o intervalo de valores próximos e consequentemente impactar na acurácia durante os testes. Com os novos valores de saída foi definida, também, uma nova faixa de aceitação para os testes, onde o valor calculado é dito certo se for maior ou igual a zero e menor que 0.5 no caso da saída 0.0, e ainda se for igual ou maior que 0.5 e menor ou igual a 1 no caso da saída 1.0.

4.2.3. *Treinamento e testes das redes neurais*

Para este *dataset* foram feitas 4 etapas de simulações, sendo que foram descritas apenas as últimas 3 por conta do desempenho ruim da primeira etapa. Primeiro foi simulado o *dataset* sem balanceamento de dados, utilizando-se as técnicas *hold-out* e *k-fold cross validation* (k = 4), porém a acurácia ficou baixa, especialmente quando eram usadas as parcelas 1, 2 e 3 do *k-fold* para treinar a rede. Já a partir da segunda etapa, descrita nas tabela 7 e 8, foi usado o *dataset* balanceado, com as técnicas *hold-out* e *k-fold* (com a parcela 4

fixada como treino por conta do desempenho anterior). Na terceira etapa, descrita nas tabelas de 9 a 11, foram feitas simulações ignorando erros grandes na etapa de testes, ou seja, com remoção de outliers pelo método das discrepâncias, onde se o erro ao calcular a saída fosse obtido um valor maior que 0.25, o valor não seria considerado na definição de acurácia, com isso variou-se a quantidade de registros em cada simulação feita. Ainda na terceira etapa foram feitas simulações iniciais utilizando a plataforma weka, sendo que foi utilizada a versão original do german credit no formato arff (aceito pelo weka) e sem o tratamento de erros grandes mencionado. Na quarta etapa, descrita nas tabelas de 12 a 14, foi feita uma análise estatística para eliminar registros que tivessem acima ou abaixo dos limites calculados com base em cada atributo selecionado, ou seja, atributos com *outliers*. Para identificar e eliminar *outliers* foram selecionados os atributos com valores mais variáveis e portanto, mais prováveis de possuir algum fora dos limites, os selecionados foram os atributos 2, 4 e 10 seguindo a numeração exposta na tabela 6. Após a eliminação de *outliers* o *dataset* ficou com 862 registros.

Os testes feitos foram de forma semelhante ao que foi feito no *hepatis*, onde os parâmetros vão sendo testados enquanto o restante não se altera, ao achar o com melhor desempenho este é fixado e passa-se para o teste de outro parâmetro.

RND 3 Camadas									
K-Fold	**Erros**	**Acertos**	**NINT1**	**NINT2**	**TAPR**	**TAPR1**	**Épocas**	**Acurácia**	**EMQ**
D4-D1	69	181	5		0.2		300000	72.40%	0.000935
D4-D1	69	181	7		0.2		300000	72.40%	0.000954
D4-D1	66	184	9		0.2		200000	73.6%	0.000783
D4-D1	68	182	9		0.2		300000	72.8%	0.000670
D4-D1	75	175	2		0.1		300000	70.0%	0.000990
D4-D1	75	175	2		0.1		200000	70.0%	0.000990
D4-D1	75	175	2		0.1		400000	70.0%	0.000990
D4-D1	70	180	9		0.2		250000	72.0%	0.000906
D4-D2	62	188	9		0.2		800000	75.2%	0.001988
D4-D2	63	187	7		0.2		800000	74.8%	0.001255
D4-D2	64	186	9		0.2		700000	74.4%	0.001973
D4-D2	67	183	9		0.2		600000	73.2%	0.001936
D4-D2	68	182	8		0.2		800000	72.8%	3.60e-08
D4-D2	68	182	9		0.9		800000	72.8%	3.54e-07

D4-D2	64	186	9		0.2		650000	74.4%	0.001959
D4-D2	66	184	9		0.3		800000	73.6%	0.001999
D4-D3	69	181	9		0.9		750000	72.40%	0.002000
D4-D3	69	181	9		0.9		700000	72.40%	0.002000
D4-D3	63	187	9		0.7		700000	74.8%	0.002000
D4-D3	66	184	9		0.3		700000	73.6%	0.002000
D4-D3	73	177	9		0.9		500000	70.8%	0.002000
D4-D3	72	178	9		0.5		700000	71.2%	0.002000
D4-D3	71	179	9		0.2		700000	71.6%	0.002000
D4-D3	70	180	9		0.9		800000	72.0%	0.002000
RND 4 Camadas									
K-Fold	Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
D4-D1	69	181	5	7	0.9	0.9	450000	72.40%	0.001692
D4-D1	75	175	5	7	0.9	0.9	250000	70.0%	0.001651
D4-D1	69	181	5	7	0.9	0.9	350000	72.40%	0.001692
D4-D1	71	179	5	7	0.9	0.9	400000	71.6%	0.001623
D4-D1	75	175	9	5	0.5	0.7	300000	70.0%	0.002000
D4-D1	78	172	5	7	0.9	0.7	300000	68.8%	0.001680
D4-D1	75	175	5	7	0.9	0.9	200000	70.0%	0.001702
D4-D1	69	181	5	7	0.9	0.9	300000	72.40%	0.001696
D4-D2	62	188	5	5	0.9	0.9	400000	75.2%	0.001682
D4-D2	69	181	5	7	0.9	0.9	400000	72.40%	0.001981
D4-D2	70	180	5	7	0.9	0.9	350000	72.0%	0.001785
D4-D2	71	179	5	7	0.9	0.9	450000	71.6%	0.002000
D4-D2	75	175	5	8	0.9	0.9	400000	70.0%	0.000139
D4-D2	75	175	5	6	0.9	0.9	400000	70.0%	0.000773
D4-D2	70	180	5	7	0.9	0.9	300000	72.0%	0.001817

D4-D2	71	179	6	7	0.9	0.9	400000	71.6%	0.001602
D4-D3	65	185	3	4	0.3	0.9	700000	74.0%	0.001460
D4-D3	61	189	3	4	0.3	0.7	700000	75.6%	0.001743
D4-D3	65	185	3	4	0.3	0.3	700000	74.0%	0.001608
D4-D3	67	183	3	4	0.7	0.9	700000	73.2%	0.001601
D4-D3	69	181	3	8	0.9	0.9	700000	72.40%	0.001646
D4-D3	72	178	3	2	0.9	0.9	700000	71.2%	0.001699
D4-D3	68	182	3	4	0.5	0.9	700000	72.8%	0.001746
D4-D3	66	184	3	4	0.9	0.9	700000	73.6%	0.001658
RNR									
K-Fold	Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
D4-D1	62	188	14		0.2		800000	75,12%	0.113929
D4-D1	65	185	12		0.2		800000	74%	0.114039
D4-D1	61	189	14		0.2		750000	75.6%	0.112253
D4-D1	63	187	14		0.2		700000	74.8%	0.110895
D4-D1	63	187	13		0.2		700000	74.8%	0.110895
D4-D2	72	178	14		0.2		750000	71.2%	0.120025
D4-D2	72	178	14		0.2		700000	71.2%	0.11859
D4-D2	68	182	14		0.2		650000	72.8%	0.116419
D4-D2	66	184	11		0.2		650000	73.6%	0.109024
D4-D2	71	179	9		0.2		650000	71.6%	0.127646
D4-D3	76	174	9		0.8		800000	69.6%	0.142902
D4-D3	74	176	6		0.8		800000	70.4%	0.143929
D4-D3	76	174	3		0.8		800000	69.6%	0.139258
D4-D3	75	175	2		0.8		800000	70.0%	0.110225
D4-D3	76	174	2		0.8		700000	70.4%	0.109808

Tabela 7 - Resultados K-Fold com balanceamento (RND 3 camadas, RND 4 camadas e RNR)

Como pode-se observar, nos valores selecionados, a acurácia não ultrapassou 75.6%, valor este alcançado nas redes de 4 camadas e recorrente. Como no primeiro *dataset* este vai seguir com os experimentos utilizando a técnica *hold-out* por conta dos melhores resultados, o que pode ser conferido na tabela 8 (resumo das simulações com *hold-out*, contendo os 5 melhores resultados em cada rede).

RND 3 Camadas								
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
72	262	9		0.1		300000	78.44%	4.76e-05
73	261	6		0.1		300000	78.14%	5.76e-05
73	261	4		0.2		300000	78.14%	3.61e-05
70	264	4		0.1		300000	79.04%	4.72e-05
70	264	3		0.1		300000	79.04%	4.38e-05
RND 4 Camadas								
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
73	261	2	8	0.4	0.7	300000	78.14%	2.12e-05
74	260	3	8	0.2	0.4	300000	77.84%	2.91e-05
73	261	3	8	0.2	0.3	300000	78.14%	3.22e-05
72	262	3	8	0.2	0.2	300000	78.44%	2.74e-05
70	264	3	8	0.2	0.1	300000	79.04%	4.08e-05
RNR								
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
80	254	14		0.6		500000	76,05%	0.08413
74	268	14		0.1		500000	77,84%	0.07873
75	259	9		0.1		500000	77,54%	0.07893
73	261	8		0.1		500000	78,14%	0.07936
72	262	3		0.1		500000	78,44%	0.0788

Tabela 8 - Resultados hold-out com balanceamento (RND 3 camadas, RND 4 camadas e RNR)

A tabela 7 mostra as simulações feitas com o *dataset* balanceado e utilizando a técnica *k-fold cross validation*, enquanto na tabela 8 é usada a técnica *hold-out* no mesmo tipo de base de dados. Analisando as tabelas é possível concluir que a acurácia melhorou utilizando-se redes diretas com três e quatro camadas, onde a acurácia chegou a 79.04% para ambos as arquiteturas de RNAs.

Nas tabelas a seguir (9 a 11) foi feita a filtragem de registros que resultam em um erro grande (0.25), portanto foi ignorado esses casos durante o cálculo de acurácia na etapa de testes, fazendo com que variasse a quantidade de registros considerados para a acurácia a cada simulação feita.

Erros	Acertos	NINT	TAPR	Épocas	Acurácia	EMQ
45	258	4	0.8	300000	85.15%	2.03e-05
43	232	8	0.8	200000	84.36%	0.000178
40	233	7	0.8	200000	85.35%	0.000333
41	234	4	0.8	200000	85.09%	0.000182
46	249	4	0.8	250000	84.41%	5.94e-05
45	258	4	0.8	300000	85.15%	2.03e-05
39	244	4	0.8	350000	86.22%	8.50e-06
37	252	4	0.8	400000	87.20%	2.82e-06
46	257	4	0.9	300000	84.81%	1.88e-05
42	247	4	0.9	400000	85.47%	9.64e-06
36	254	4	0.7	400000	87.59%	2.63e-06
39	253	5	0.7	400000	86.64%	1.17e-05
40	255	6	0.7	400000	86.44%	6.36e-06
34	246	7	0.7	400000	87.86%	9.77e-06
31	249	8	0.7	400000	88.93%	7.98e-06
26	254	9	0.7	400000	90.71%	2.63e-06
27	251	10	0.7	400000	90.29%	7.81e-06

Tabela 9 - Resultados hold-out RND com 3 camadas (Remoção de outliers método das discrepâncias)

Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR 1	Épocas	Acurácia	EMQ

59	250	7	5	0.5	0.7	300000	80.91%	0.096
50	247	5	5	0.5	0.7	300000	83.16%	0.105
41	253	3	5	0.5	0.7	300000	86.05%	0.101
46	250	4	5	0.5	0.7	300000	84.46%	0.101
35	244	10	5	0.5	0.7	300000	87.46%	0.117
45	248	8	5	0.5	0.7	300000	84.64%	0.103
55	250	9	5	0.5	0.7	300000	81.97%	0.100
57	242	10	6	0.5	0.7	300000	80.94%	0.101
52	245	10	7	0.5	0.7	300000	82.49%	0.103
44	245	10	8	0.5	0.7	300000	84.78%	0.108
34	247	10	9	0.5	0.7	300000	87.90%	0.115
45	250	10	10	0.5	0.7	300000	84.75%	0.107
57	245	10	11	0.5	0.7	300000	81.13%	0.102
27	244	10	9	0.6	0.7	300000	90.04%	0.126
24	242	10	9	0.7	0.7	300000	90.98%	0.134
34	231	10	9	0.8	0.7	300000	87.17%	0.137
35	237	10	9	0.9	0.7	300000	87.13%	0.129
33	253	10	9	0.7	0.3	300000	88.46%	0.098
26	254	10	9	0.7	0.4	300000	90.71%	0.104
33	251	10	9	0.7	0.5	300000	88.38%	0.109
39	240	10	9	0.7	0.6	300000	86.02%	0.120
24	242	10	9	0.7	0.8	300000	90.98%	0.132
40	245	10	9	0.7	0.9	300000	85.96%	0.114
25	242	10	9	0.7	0.8	350000	90.64%	0.135
25	242	10	9	0.7	0.8	400000	90.64%	0.136
13	251	10	9	0.7	0.8	450000	95.08%	0.134
9	250	10	9	0.7	0.8	500000	96.53%	0.145
14	242	10	9	0.7	0.8	550000	94.53%	0.154

Tabela 10 - Resultados hold-out RND de 4 camadas (Remoção de outliers método das discrepâncias)

Erros	Acertos	NINT	TAPR	Épocas	Acurácia	EMQ
52	213	5	0.9	200000	80.38%	0.145
51	214	6	0.9	200000	80.75%	0.146
46	208	7	0.9	200000	81.89%	0.168
59	203	8	0.9	200000	77.48%	0.155
53	212	9	0.9	200000	80.0%	0.151
48	214	10	0.9	200000	81.68%	0.160
51	214	11	0.9	200000	80.75%	0.145
52	208	12	0.9	200000	80.0%	0.174
50	218	7	0.8	200000	81.34%	0.147
48	223	7	0.7	200000	82.29%	0.132
44	251	7	0.7	250000	85.08%	0.098
43	256	7	0.7	300000	85.62%	0.094
43	237	7	0.7	350000	84.64%	0.112
38	249	7	0.7	400000	86.76%	0.104
31	252	7	0.7	450000	89.05%	0.112
30	251	7	0.7	500000	89.32%	0.114
42	225	7	0.7	550000	84.27%	0.145
40	223	7	0.7	600000	84.79%	0.155

Tabela 11 - Resultados hold-out RNR (Remoção de outliers método das discrepâncias)

Analisando as tabelas de 9 a 11 é possível observar novamente uma melhor acurácia para as redes diretas de quatro camadas, onde a acurácia chegou a 96.52%.

Erros	Acertos	NINT	TAPR	Épocas	Acurácia	EMQ
62	226	4	0.8	400000	78.47%	2.40e-06
60	228	4	0.7	400000	79.17%	2.46e-06
64	224	7	0.7	400000	77.78%	1.57e-05

66	222	8	0.7	400000	77.083%	3.51e-06
76	212	9	0.7	400000	73.61%	8.40e-07
75	213	10	0.7	400000	73.96%	2.78e-06
57	231	4	0.8	350000	80.21%	1.66e-06
65	223	5	0.7	400000	77.43%	1.69e-05
60	228	6	0.7	400000	79.17%	2.74e-06
58	230	4	0.8	300000	79.86%	2.063e-06
60	228	8	0.8	200000	79.17%	1.49e-06
59	229	7	0.8	200000	79.51%	2.30e-05
61	227	4	0.8	200000	78.82%	1.57e-05
60	228	4	0.8	250000	79.17%	8.79e-06
58	230	4	0.8	300000	79.86%	2.06e-06
60	228	4	0.9	300000	79.17%	1.65e-06
64	224	4	0.9	400000	77.78%	2.65e-06
57	231	4	0.9	350000	80.21%	1.89e-06

Tabela 12 - Resultados hold-out RND de 3 camadas (Remoção de outliers método estatístico)

Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
86	202	7	5	0.5	0.7	300000	70.14%	0.098
78	210	5	5	0.5	0.7	300000	72.92%	0.088
68	220	3	5	0.5	0.7	300000	76.39%	0.086
83	205	4	5	0.5	0.7	300000	71.18%	0.095
71	217	10	5	0.5	0.7	300000	75.35%	0.090
77	211	8	5	0.5	0.7	300000	73.26%	0.088
75	213	9	5	0.5	0.7	300000	73.96%	0.097
73	215	10	6	0.5	0.7	300000	74.65%	0.097
76	212	10	7	0.5	0.7	300000	73.61%	0.102
70	218	10	8	0.5	0.7	300000	75.69%	0.101

75	213	10	9	0.5	0.7	300000	73.96%	0.102
70	218	10	10	0.5	0.7	300000	75.69%	0.101
76	212	10	9	0.6	0.7	300000	73.61%	0.101
78	210	10	9	0.7	0.7	300000	72.92%	0.105
74	214	10	9	0.8	0.7	300000	74.30%	0.102
82	206	10	9	0.9	0.7	300000	71.53%	0.106
61	227	10	9	0.7	0.3	300000	78.82%	0.080
74	214	10	9	0.7	0.4	300000	74.30%	0.084
67	221	10	9	0.7	0.5	300000	76.74%	0.080
75	213	10	9	0.7	0.6	300000	73.96%	0.099
69	219	10	9	0.7	0.8	300000	76.04%	0.103
70	218	10	9	0.7	0.9	300000	75.69%	0.101
62	226	10	9	0.7	0.3	350000	78.47%	0.078
65	223	10	9	0.7	0.3	400000	77.43%	0.079
70	218	10	9	0.7	0.3	450000	75.69%	0.085

Tabela 13 - Resultados hold-out RND de 4 camadas (Remoção de outliers método estatístico)

Erros	Acertos	NINT	TAPR	Épocas	Acurácia	EMQ
59	229	5	0.9	200000	79.51%	0.079
59	229	6	0.9	200000	79.51%	0.079
65	223	7	0.9	200000	77.43%	0.083
65	223	8	0.9	200000	77.43%	0.085
59	229	9	0.9	200000	79.51%	0.079
64	224	10	0.9	200000	77.78%	0.085
59	229	11	0.9	200000	79.51%	0.079
59	229	12	0.9	200000	79.51%	0.078
60	228	9	0.7	200000	79.17%	0.078
59	229	9	0.8	200000	79.51%	0.078

60	228	9	0.8	250000	79.17%	0.078
61	227	9	0.8	300000	78.82%	0.078
71	217	9	0.8	350000	75.35%	0.085
87	201	9	0.8	400000	69.79%	0.107
88	200	9	0.8	450000	69.44%	0.114
74	214	9	0.8	500000	74.31%	0.099
70	218	9	0.8	550000	75.69%	0.099
81	207	9	0.8	600000	71.87%	0.098

Tabela 14 - Resultados hold-out RNR (Remoção de outliers método estatístico)

Levando em conta todas as etapas a que teve o melhor desempenho foi a terceira, onde ocorreu a eliminação da ocorrência de erros grandes na fase de testes. A acurácia da melhor simulação foi de 96.52% na rede direta de 4 camadas, tendo a seguinte configuração de parâmetros: 10 neurônios na camada interna 1, 9 neurônios na camada interna 2, taxa de aprendizagem 1 de 0.7, taxa de aprendizagem 2 de 0.8 e 500000 épocas.

Nas tabelas abaixo são descritos os experimentos realizados com o weka utilizando o algoritmo KNN, onde na tabela 15 temos as simulações com o german credit original (21 atributos) e na 16 temos simulações com o mesmo *dataset,* mas sem os registros que contêm *outliers*. Nas duas etapas de simulações foram utilizadas as técnicas *k-fold* e *hold-out*, sendo que a segunda apresentou os melhores resultados quando usado 90% dos dados para treinamento. O melhor desempenho foi na primeira etapa do weka, sem eliminação de registros com outliers, tendo a seguinte configuração de parâmetros: 90% de dados para treino, 10% de dados para teste e 6 vizinhos próximos (atributo referente ao K do algoritmo KNN).

Técnica	Folds	Treino	Teste	K	Acurácia
K-Fold	2			1	71,40%
K-Fold	2			2	73,40%
K-Fold	2			3	74,20%
K-Fold	2			4	74,10%
K-Fold	2			5	74,00%
K-Fold	2			6	73,90%
K-Fold	2			7	73,30%
K-Fold	2			8	72,70%
K-Fold	2			9	73,10%
K-Fold	2			10	73,30%

K-Fold	3			1	69,30%
K-Fold	3			2	73,40%
K-Fold	3			3	72,70%
K-Fold	3			4	74,10%
K-Fold	3			5	73,60%
K-Fold	3			6	73,60%
K-Fold	3			7	72,70%
K-Fold	3			8	73,50%
K-Fold	3			9	74,80%
K-Fold	3			10	73,60%
K-Fold	4			1	72,40%
K-Fold	4			2	73,30%
K-Fold	4			3	73,40%
K-Fold	4			4	74,50%
K-Fold	4			5	75,00%
K-Fold	4			6	75,30%
K-Fold	4			7	74,40%
K-Fold	4			8	73,50%
K-Fold	4			9	73,90%
K-Fold	4			10	73,70%
Hold-Out		66%	34%	1	73%
Hold-Out		66%	34%	2	76,47%
Hold-Out		66%	34%	3	74,11%
Hold-Out		66%	34%	4	74,70%
Hold-Out		66%	34%	5	72,35%
Hold-Out		66%	34%	6	75%
Hold-Out		66%	34%	7	74,41%
Hold-Out		66%	34%	8	75,29%
Hold-Out		66%	34%	9	75,88%
Hold-Out		66%	34%	10	75,58%
Hold-Out		80%	20%	1	71%
Hold-Out		80%	20%	2	75,50%
Hold-Out		80%	20%	3	74%
Hold-Out		80%	20%	4	79%
Hold-Out		80%	20%	5	76,50%
Hold-Out		80%	20%	6	78%
Hold-Out		80%	20%	7	79%

Hold-Out		80%	20%	8	78%
Hold-Out		80%	20%	9	81%
Hold-Out		80%	20%	10	78%
Hold-Out		90%	10%	1	69%
Hold-Out		90%	10%	2	74%
Hold-Out		90%	10%	3	74%
Hold-Out		90%	10%	4	78%
Hold-Out		90%	10%	5	78%
Hold-Out		90%	10%	6	83%
Hold-Out		90%	10%	7	76%
Hold-Out		90%	10%	8	79%
Hold-Out		90%	10%	9	80%
Hold-Out		90%	10%	10	79%

Tabela 15 - Resultados KNN com weka

Técnica	Folds	Treino	Teste	K	Acurácia
K-Fold	2			1	71,00%
K-Fold	2			2	75,75%
K-Fold	2			3	74,48%
K-Fold	2			4	74,36%
K-Fold	2			5	74,24%
K-Fold	2			6	73,78%
K-Fold	2			7	74,59%
K-Fold	2			8	74,59%
K-Fold	2			9	75,40%
K-Fold	2			10	74,94%
K-Fold	3			1	70,53%
K-Fold	3			2	74,01%
K-Fold	3			3	73,66%
K-Fold	3			4	74,71%
K-Fold	3			5	75,29%
K-Fold	3			6	75,17%

K-Fold	3			7	75,98%
K-Fold	3			8	75,40%
K-Fold	3			9	75,98%
K-Fold	3			10	76,21%
K-Fold	4			1	75,85%
K-Fold	4			2	75,75%
K-Fold	4			3	75,17%
K-Fold	4			4	76,56%
K-Fold	4			5	74,59%
K-Fold	4			6	75,75%
K-Fold	4			7	75,29%
K-Fold	4			8	75,98%
K-Fold	4			9	75,40%
K-Fold	4			10	75,87%
Hold-Out		66,00%	34,00%	1	69,96%
Hold-Out		66,00%	34,00%	2	73,38%
Hold-Out		66,00%	34,00%	3	72,69%
Hold-Out		66,00%	34,00%	4	75,43%
Hold-Out		66,00%	34,00%	5	74,74%
Hold-Out		66,00%	34,00%	6	75,08%
Hold-Out		66,00%	34,00%	7	74,06%
Hold-Out		66,00%	34,00%	8	72,69%
Hold-Out		66,00%	34,00%	9	73,72%
Hold-Out		66,00%	34,00%	10	72,35%
Hold-Out		80,00%	20,00%	1	73,25%
Hold-Out		80,00%	20,00%	2	77,32%
Hold-Out		80,00%	20,00%	3	76,76%
Hold-Out		80,00%	20,00%	4	75,00%

Hold-Out		80,00%	20,00%	5	73,25%
Hold-Out		80,00%	20,00%	6	73,84%
Hold-Out		80,00%	20,00%	7	74,42%
Hold-Out		80,00%	20,00%	8	73,83%
Hold-Out		80,00%	20,00%	9	72,67%
Hold-Out		80,00%	20,00%	10	73,84%
Hold-Out		90,00%	10,00%	1	77,91%
Hold-Out		90,00%	10,00%	2	80,23%
Hold-Out		90,00%	10,00%	3	81,39%
Hold-Out		90,00%	10,00%	4	81,39%
Hold-Out		90,00%	10,00%	5	76,74%
Hold-Out		90,00%	10,00%	6	82,55%
Hold-Out		90,00%	10,00%	7	80,23%
Hold-Out		90,00%	10,00%	8	81,39%
Hold-Out		90,00%	10,00%	9	80,23%
Hold-Out		90,00%	10,00%	10	80,23%

Tabela 16 - Resultados KNN com weka (Remoção de outliers método estatístico)

Arquitetura da Rede	**Técnica**	**Pré-processamento**	**Acurácia**
3C-Direta	Hold-Out	Balanceamento	79.4%
4C-Direta	Hold-Out	Balanceamento	79.4%
Recorrente	Hold-Out	Balanceamento	78.44%
3C-Direta	Hold-Out	Remoção de outliers método estatístico	80.21%
4C-Direta	Hold-Out	Remoção de outliers método estatístico	78.82%
Recorrente	Hold-Out	Remoção de outliers método estatístico	79.51%

3C-Direta	Hold-Out	Remoção de outliers método das discrepâncias	90.71%
4C-Direta	Hold-Out	Remoção de outliers método das discrepâncias	96.53%
Recorrente	Hold-Out	Remoção de outliers método das discrepâncias	89.32%

Tabela 17 - Melhores resultados com as redes backpropagation

Algoritmo	**Técnica**	**Pré-processamento**	**Acurácia**
KNN	Hold-Out	Balanceamento	83%
	Hold-Out	Remoção de outliers método estatístico	82.55%

Tabela 18 - Melhores resultados com KNN no weka

Levando em conta os melhores resultados expostos nas tabelas 17 e 18, o algoritmo que teve o melhor desempenho foi o backpropagation com uma rede de 4 camadas e usando a técnica hold-out, onde ocorreu a Remoção de Outliers pelo método das discrepâncias na fase de testes. A acurácia da melhor simulação foi de 96.52% na rede direta de 4 camadas, tendo os seguintes parâmetros: 10 neurônios na camada interna 1, 9 neurônios na camada interna 2, taxa de aprendizagem 1 de 0.7, taxa de aprendizagem 2 de 0.8 e 500000 épocas.

Comparando-se os resultados das simulações do weka e backpropagation (tabelas 17 e 18), com pré-processamento semelhante, pode-se concluir que, se levando em conta o *dataset* balanceado (tabela 8 e tabela 15), a melhor acurácia obtida pelo KNN do weka, foi de 83%, sendo esse valor melhor do que o obtido com *backpropagation,* com arquiteturas diretas de 3 camadas e 4 camadas, cujo valor foi de 79.4%. Considerando as simulações utilizando o *dataset* após a remoção de outliers, o KNN obteve uma acurácia de 82.55%, enquanto a acurácia obtida com o backpropagation foi de 80.21%. Em todas as melhores simulações citadas foi utilizada a técnica *hold-out*, sendo que no weka foi adotada a proporção de 90% dos dados no treino e no *backpropagation* utilizou-se a divisão padrão na tabela 8 e nas tabelas 12 a 14.

Considerando as simulações com pré-processamento semelhante do backpropagation e KNN, a melhor configuração de parâmetros usados no weka foi nas simulações com o dataset sem remoção de outliers com a técnica hold-out, onde a proporção de treino foi 90%, teste 10% e o k do KNN foi 6, resultando na acurácia de 83%. No backpropagation a melhor configuração foi ao usar o dataset com remoção de outliers e a rede direta de 3 camadas com técnica hold-out e proporção padrão para treino e teste, onde o número de neurônios da

camada interna foi 4, taxa de aprendizagem foi 0.8 e com 350000 épocas, resultando na acurácia de 80.21%.

4.2.5. *Resumo dos resultados*

4.2.5.1. *Resultado das simulações com dataset balanceado*

4.2.5.1.1. *K-Fold*

RND 3 Camadas									
K-Fold	**Erros**	**Acertos**	**NINT1**	**NINT2**	**TAPR**	**TAPR1**	**Épocas**	**Acurácia**	**EMQ**
D4-D1	69	181	5		0.2		300000	72.40%	0.000935
D4-D1	66	184	9		0.2		200000	73.6%	0.000783
D4-D1	68	182	9		0.2		300000	72.8%	0.000670
D4-D2	62	188	9		0.2		800000	75.2%	0.001988
D4-D2	63	187	7		0.2		800000	74.8%	0.001255
D4-D2	64	186	9		0.2		700000	74.4%	0.001973
D4-D3	69	181	9		0.9		700000	72.40%	0.002000
D4-D3	63	187	9		0.7		700000	74.8%	0.002000
D4-D3	66	184	9		0.3		700000	73.6%	0.002000
RND 4 Camadas									
K-Fold	Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
D4-D1	69	181	5	7	0.9	0.9	450000	72.40%	0.001692
D4-D1	69	181	5	7	0.9	0.9	350000	72.40%	0.001692
D4-D1	69	181	5	7	0.9	0.9	300000	72.40%	0.001696
D4-D2	62	188	5	5	0.9	0.9	400000	75.2%	0.001682
D4-D2	69	181	5	7	0.9	0.9	400000	72.40%	0.001981
D4-D2	70	180	5	7	0.9	0.9	350000	72.0%	0.001785
D4-D3	65	185	3	4	0.3	0.9	700000	74.0%	0.001460
D4-D3	61	189	3	4	0.3	0.7	700000	75.6%	0.001743
D4-D3	65	185	3	4	0.3	0.3	700000	74.0%	0.001608
RNR									

K-Fold	Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
D4-D1	62	188	14		0.2		800000	75,12%	0.113929
D4-D1	61	189	14		0.2		750000	75.6%	0.112253
D4-D1	63	187	14		0.2		700000	74.8%	0.110895
D4-D2	68	182	14		0.2		650000	72.8%	0.116419
D4-D2	66	184	11		0.2		650000	73.6%	0.109024
D4-D2	71	179	9		0.2		650000	71.6%	0.127646
D4-D3	74	176	6		0.8		800000	70.4%	0.143929
D4-D3	75	175	2		0.8		800000	70.0%	0.110225
D4-D3	76	174	2		0.8		700000	70.4%	0.109808

Na RND de 3 camadas, usando as parcelas 4 e 1 do *dataset*, é possível ver uma acurácia melhor quando utilizado um número de épocas menor do que as demais e 9 neurônios na camada interna. Com as parcelas 4 e 2 na RND de 3 camadas praticamente os mesmo parâmetros das parcelas anteriores porém com um número de épocas bem mais elevado, o que acabou resultando em uma acurácia melhor. Usando as parcelas 4 e 3 a acurácia melhorou ao aumentar a taxa de aprendizagem em comparação com as outras simulações feitas com estas parcelas.

Na RND de 4 camadas, usando as parcelas 4 e 1, é possível ver uma estabilização de acurácia na faixa de épocas mostrada. Ao usar as parcelas 4 e 2 é possível ver uma variação, principalmente ao alterar a taxa de aprendizagem 2 e resultar em um desempenho melhor. Nas parcelas 4 e 3 é possível ver o melhor resultado, de novo por conta da variação da taxa de aprendizagem 2.

Na rede recorrente, ao usar as parcelas 4 e 1, é possível notar uma acurácia melhor ao aumentar o número de épocas, especialmente ao usar 750000. Ao usar as parcelas 4 e 2 houve variação positiva na acurácia ao aumentar o número de neurônios da camada interna, especialmente ao usar o valor 11. Nas parcelas 4 e 3 houve um resultado bom ao aumentar os neurônios da camada interna de 2 para 6 e ao diminuir as épocas de 800000 para 700000.

4.2.5.1.2. *Hold-Out*

RND 3 Camadas								
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
72	262	9		0.1		300000	78.44%	4.76e-05

70	264	4		0.1		300000	79.04%	4.72e-05
70	264	3		0.1		300000	79.04%	4.38e-05
RND 4 Camadas								
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
73	261	2	8	0.4	0.7	300000	78.14%	2.12e-05
72	262	3	8	0.2	0.2	300000	78.44%	2.74e-05
70	264	3	8	0.2	0.1	300000	79.04%	4.08e-05
RNR								
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
74	268	14		0.1		500000	77,84%	0.07873
73	261	8		0.1		500000	78,14%	0.07936
72	262	3		0.1		500000	78,44%	0.0788

Na RND de 3 camadas a acurácia melhorou e ficou com um menor erro quadrático médio quando foi reduzido o número de neurônios da camada interna para 3. Na RND de 4 camadas foi possível obter o melhor resultado ao estabelecer pequenas taxas de aprendizagem e menos neurônios na camada interna 1. A rede recorrente, assim como na RND de 3 camadas, alcançou uma acurácia melhor ao estabelecer 3 neurônios na camada interna.

4.2.5.2. *Resultado das simulações com eliminação de outliers pelo método das discrepâncias*

RND 3 Camadas								
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
26	254	9		0.7		400000	90.71%	2.63e-06
27	251	10		0.7		400000	90.29%	7.81e-06
31	249	8		0.7		400000	88.93%	7.98e-06
RND 4 Camadas								
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
9	250	10	9	0.7	0.8	500000	96.53%	0.145
13	251	10	9	0.7	0.8	450000	95.08%	0.134
14	242	10	9	0.7	0.8	550000	94.53%	0.154

RNR								
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
30	251	7		0.7		500000	89.32%	0.114
31	252	7		0.7		450000	89.05%	0.112
38	249	7		0.7		400000	86.76%	0.104

Na RND 3 camadas há variação positiva na acurácia quando o número de neurônios da camada interna varia, atingindo o melhor desempenho ao ser igual a 9. Na RND de 4 camadas e recorrente a acurácia melhora ao aumentar a quantidade de épocas, especialmente quando é igual a 500000.

4.2.5.3. *Resultado das simulações com eliminação de outliers pelo método estatístico*

RND 3 Camadas								
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
57	231	4		0.8		350000	80.21%	1.66e-06
57	231	4		0.9		350000	80.21%	1.89e-06
58	230	4		0.8		300000	79.86%	2.06e-06
RND 4 Camadas								
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
61	227	10	9	0.7	0.3	300000	78.82%	0.080
62	226	10	9	0.7	0.3	350000	78.47%	0.078
65	223	10	9	0.7	0.3	400000	77.43%	0.079
RNR								
Erros	Acertos	NINT1	NINT2	TAPR	TAPR1	Épocas	Acurácia	EMQ
59	229	9		0.8		200000	79.51%	0.078
59	229	12		0.9		200000	79.51%	0.078
59	229	11		0.9		200000	79.51%	0.079

Na RND de 3 camadas nota-se a influência da diminuição da taxa de aprendizagem para alcançar um melhor resultado. Na RND de 4 camadas o que melhora a acurácia é a variação do número de épocas, onde a quantidade de 300000 foi a ideal. Na rede recorrente

temos uma estabilização de acurácia nos melhores resultados, porém o EMQ menor foi alcançado ao alterar o número de neurônios da camada interna de 11 para 9 e por mudar a taxa de aprendizagem de 0.9 para 0.8.

4.2.5.3. *Resultados das simulações com KNN no weka utilizando dataset balanceado*

Técnica	Folds	Treino	Teste	K	Acurácia
K-Fold	3			9	74,80%
K-Fold	4			6	75,30%
K-Fold	2			3	74,20%
Hold-Out		66%	34%	2	76,47%
Hold-Out		80%	20%	9	81%
Hold-Out		90%	10%	6	83%

Ao usar K-Fold nota-se um melhor desempenho ao definir o k da técnica igual a 4 e o k do KNN igual a 6. No hold-out a acurácia aumenta ao definir uma parcela maior de dados para treino e o k do KNN igual a 6.

4.2.5.3. *Resultados das simulações com KNN no weka utilizando dataset após eliminação de outliers por método estístico*

Técnica	Folds	Treino	Teste	K	Acurácia
K-Fold	2			2	75,75%
K-Fold	3			10	76,21%
K-Fold	4			4	76,56%
Hold-Out		66,00%	34,00%	4	75,43%
Hold-Out		80,00%	20,00%	2	77,32%
Hold-Out		90,00%	10,00%	6	82,55%

Quando utilizado o k-fold nota-se uma acurácia melhor ao definir o k da técnica igual ao k do KNN, ou seja, igual a 4. Quanto ao hold-out, novamente uma parcela maior de dados para treino, junto com o k do KNN igual a 6, fez com que o resultado fosse melhor.

5. ATIVIDADE FUTURAS

Realizar tarefas de predições de séries temporais utilizando arquiteturas de redes recorrentes e com aprendizado profundo redes (LSTM), comparando o desempenho das mesmas com outras técnicas. Analisar e comparar diferentes algoritmos de aprendizado profundo e estatísticos para prever a velocidade do vento em vários passos no Brasil, utilizando os frameworks Keras/Tensorflow.

6. DIFICULDADES

Não houve grandes dificuldades por ter sido trabalhado *datasets* com poucos registros, porém se houvesse um aumento considerável nessa quantidade poderia ser um problema com relação ao tempo de processamento por conta do *hardware* bem limitado utilizado no projeto. A principal dificuldade encontrada foi com relação ao *dataset hepatitis*, pois o mesmo possui muito pouco dado para ser trabalhado e ainda por cima conta com muitos dados ausentes nos atributos do *dataset*.

7. REFERÊNCIAS

ZHAO, Z. et al (2014). "Investigation and improvement of multi-layer perceptron neural networks for credit scoring".

AMARAL, F. "Aprenda Mineração de Dados". Editora Alta Books. Rio de Janeiro, 2016.

HEMANTH, D. e KOSE, U. "Artificial Intelligence and Applied Mathematics in Engineering Problems". Editora Springer. Suíça, 2019.

PARECER DO ORIENTADOR: Manifestação do orientador sobre o desenvolvimento das atividades do aluno.

DATA: _____/_____/_____

_____

ASSINATURA DO ORIENTADOR

_____

ASSINATURA DO ALUNO