



FeMASS

FACULDADE PROF. MIGUEL ÂNGELO DA SILVA SANTOS

FACULDADE PROFESSOR MIGUEL ÂNGELO DA SILVA SANTOS – FeMASS
CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

APLICAÇÃO DE REDE NEURAL ARTIFICIAL PARA PREDIÇÃO DE
LITOLOGIA DE ROCHAS CARBONÁTICAS

POR:
WILLIAN DE SÁ CARVALHO

MACAÉ
2019

FACULDADE PROFESSOR MIGUEL ÂNGELO DA SILVA SANTOS – FeMASS
CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

Willian de Sá Carvalho

APLICAÇÃO DE REDE NEURAL ARTIFICIAL PARA PREDIÇÃO DE
LITOLOGIA DE ROCHAS CARBONÁTICAS

Trabalho Final apresentado ao curso de graduação em Sistemas de Informação, da Faculdade Professor Miguel Ângelo da Silva Santos (FeMASS), para obtenção do grau de BACHAREL em Sistemas de Informação.

Professor Orientador: Drº. Irineu de Azevedo Lima Neto

MACAÉ/RJ

2019

WILLIAN DE SÁ CARVALHO

**APLICAÇÃO DE REDE NEURAL ARTIFICIAL PARA PREDIÇÃO DE LITOLOGIA DE
ROCHAS CARBONÁTICAS**

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação, da Faculdade Professor Miguel Ângelo da Silva Santos (FeMASS), para obtenção do grau de BACHAREL em Sistemas de Informação.

Aprovada em ____ de _____ de 20____

BANCA EXAMINADORA

Drº. Irineu de Azevedo Lima Neto
Faculdade Professor Miguel Ângelo da Silva Santos (FeMASS)
1º Examinador

Me. Isac Mendes
Faculdade Professor Miguel Ângelo da Silva Santos (FeMASS)

ANEXO I

PARECER TÉCNICO: TRABALHO DE CONCLUSÃO DE CURSO (TCC)

CURSO: () ADMINISTRAÇÃO () ENGENHARIA DE PRODUÇÃO (X) SISTEMAS
DE INFORMAÇÃO () MATEMÁTICA

ALUNO (A): Willian de Sá Carvalho

MATR. 1501130007

DATA DA DEFESA: 10/12/2019

PROFESSOR ORIENTADOR:

Dr. Irineu de Azevedo Lima Neto

BANCA:

Me. Isac Mendes Lacerda

TÍTULO:

Aplicação de Rede Neural Artificial para Predição de Litologia de Rochas Carbonáticas

PARECER FINAL:

Em cumprimento ao Art. 9º, §5º da Deliberação nº 04/17, atesto que o (a) aluno (a) acima referido (a):

(X) atendeu às solicitações/ajustes encaminhados pela banca para validação da versão final do TCC.

() não atendeu às solicitações/ajustes encaminhados pela banca para validação da versão final do TCC.

Encaminhe-se à Secretaria Acadêmica da FeMASS, para os registros, considerando o (a) aluno(a):

(X) APROVADO () REPROVADO

Macaé-RJ, 18 de dezembro de 2019.



ASSINATURA DO PROFESSOR ORIENTADOR

AGRADECIMENTO

Agradeço a Deus, por me conduzir nesses caminhos cheios de desafios, me dando força, resiliência e persistência. Por me dar sabedoria de compreender que quando as coisas fogem do esperado não podemos mudar apenas aceitar e alterando aquilo que estão ao meu alcance.

Aos meu avós Carmem (*in memorian*) e Amaro, por serem sensacionais na minha criação, pelos carinhos, mimos e amores, por serem pais, amigos, apoiadores e incentivadores em todos os meus projetos, por nossas vidas serem tocadas e também por caminharmos lado a lado nessa existência, sendo papel fundamental em tudo que eu venha conquistar na vida.

Aos meu pais Izabel e Antônio Carlos, por auxiliarem na minha criação, por estarem por perto quando foi preciso, pelo carinho e amor.

À minha irmã Izabela, mãe, amiga, pelos abraços, choros, incentivos, por ter me dado dois sobrinhos que complementam minha felicidade, pelas risadas, furadas, por sempre ter estudado e sempre querer mais da vida e isso ter me inspirado a fazer o mesmo, por ser essa mulher inestimável, me apoioando em tudo que faço, tentando sempre auxiliar nas realizações dos meus sonhos e projetos, obrigado.

Aos meu sobrinhos Diogo e Rodrigo, pelas alegrias, carinhos, momentos únicos vivenciados, pelas gargalhadas em conjunto, por me fazerem perceber o sentido de família.

Ao Eliakim, por ser um grande amigo, incentivador, apoiador, pelas risadas, papos, furadas, trocas de ensinamento, por ter me dado o sábio conselho de transferir a faculdade pra FeMASS, por participar da construção e realização desse sonho, por muitas das vezes acredita em mim quando nem eu acredito, muitíssimo obrigado.

Aos meus amigos de infância, adquirido ao longo desses anos, do curso de graduação, da faculdade, em especial a Kamilly, Aline, Suahil, Mariana, Fabíola, Gabriel, Cintia, pelos papos, risadas, incentivo, auxílio, mimos e carinhos.

À faculdade por me permitir aquisição de conhecimentos, por me colocar diante a desafios que me fizeram crescer pessoalmente, intelectualmente e profissionalmente, pelo apoio, por me acolher e ter feito desse espaço nesses anos como se fosse uma segunda casa.

A todos os professores que contribuíram ao longo dessa jornada, em especial ao professor Geovane, por me direcionar ao caminho da área de dados e ter me feito descobrir uma paixão, pelas suas brilhantes aulas, conselhos de vida e profissionais, ao Afonso por sempre fazer boas análises, sempre solícito a ajudar aqueles que o procuram, e ao Isaac, pelas

aulas descontraídas, mas, sérias, por ser um grande professor, contribuído bastante para formação daqueles que possui a oportunidade de serem seus alunos.

Ao meu orientador Irineu que embarcou nessa viagem, aceitando um estranho e sem ele nada disso seria possível, também por ser um professor excepcional, abrindo realmente a minha cabeça para o conhecimento e inserindo-os, pelo apoio, por não ter desistido de mim, pelo incentivo e por me fazer concluir esse trabalho.

EPÍGRAFE

*“O conhecimento não é aquilo que você sabe, mas o que você
faz com aquilo que você sabe”*

Aldous Huxley

RESUMO

O reconhecimento de padrões tornou-se uma ferramenta altamente usada em diversas áreas, sendo um grande desafio para implementações de soluções. Há diversas técnicas que tentam solucionar tais questões, uma dessas abordagens são os algoritmos de Aprendizado de Máquina Supervisionado que vem ganhando destaque, alcançando resultados satisfatório em suas aplicações para previsões. Este trabalho estuda a aplicação de Redes Neurais artificiais *multilayer perceptron*, utilizando o algoritmo de *backpropagation* no problema de previsão de litologias de rochas carbonáticas. Para este objetivo, realizou-se uma série de testes, alterando alguns parâmetros da rede, verificando e comparando seus resultados, utilizando como métricas a acurácia, como também estabelecido um comparativo de qual era o modelo ideal que resolveria a problemática. Os resultados obtidos foram considerados satisfatórios, obtendo uma acurácia 100% no treinamento da rede e no teste em torno de 85,71 % ao prever em qual classe litológica a amostra pertencia.

Palavras-chave: Reconhecimento de Padrões, Aprendizado de Máquina, Redes Neurais Artificiais, *Backpropagation*, Litologias de Rochas Carbonáticas.

ABSTRACT

Pattern recognition has become a highly used tool in many areas and is a major challenge for implementing solutions. There are several techniques in charge of solving problems, one of these approaches are related to Supervised Machine Learning algorithms that have gained prominence, reaching satisfactory results in their prediction applications. This work studies the application of artificial multilayer perceptron of Neural Networks, using Backpropagation algorithm without prediction problems of carbonate rock lithologies. For this purpose, perform a series of tests, changing some network parameters, checking and comparing results, using a metric measurement accuracy, as well as using a comparative of which was the ideal model capable of solving the problem. The results were considered satisfactory, obtaining a 100% accuracy on training of the network and in test around 85.71% to predict the relevant sample lithological class.

Key words: Pattern Recognition, Machine Learning, Artificial Neural Networks, Backpropagation, Carbornate Rock Lithologies.

LISTA DE FIGURAS

Figura 1 - DATA WAREHOUSE	20
Figura 2 - CICLO DO KDD	23
Figura 3 - CICLO DO CRISP-FM	24
Figura 4 - FLUXOGRAMA DE APRENDIZADO DE MÁQUINA.....	27
Figura 5 - ALGORITMO DE APRENDIZADO POR REFORÇO	28
Figura 6 - CICLO DO ALGORITMO GENÉTICO	30
Figura 7 - ALGORITMO DE ÁRVORE DE DECISÃO TABULAR.....	31
Figura 8 - FLUXO DO ALGORITMO DE ÁRVORE DE DECISÃO.....	32
Figura 9 - GRÁFICO DE CLASSIFICAÇÃO DO KNN	33
Figura 10 - GRÁFICOS DAS REGRESSÕES LINEAR E LOGÍSTICA	34
Figura 11 - NEURÔNIO BIOLÓGICO	36
Figura 12 - NEURÔNIO ARTIFICIAL	37
Figura 13 - REDE NEURAL ACÍCLICA.....	39
Figura 14 - REDE NEURAL CÍCLICA.....	40
Figura 15 - MODELO MATEMÁTICO DO NEURÔNIO ARTIFICIAL	41
Figura 16 - OS PESOS SINÁPTICOS ARTIFICIAIS.....	42
Figura 17 - A FUNÇÃO DE ATIVAÇÃO.....	43
Figura 18 - GRÁFICO DA FUNÇÃO LINEAR.....	44
Figura 19 - GRÁFICO DA FUNÇÃO SIGMOIDE	45
Figura 20 - GRÁFICO DA FUNÇÃO TANGENTE	46
Figura 21 - GRÁFICO DA FUNÇÃO SOFTMAX	46
Figura 22 – GRÁFICO DA FUNÇÃO RELU	47
Figura 23 - LOCALIZAÇÃO DOS POÇOS LA CIOTAT-1 E LA CIOTAT -2.....	50
Figura 24 – CARTA ESTRATIGRAFICA	51
Figura 25 - DISTRIBUIÇÃO ANACONDA NAVIGATOR	56
Figura 26 - ANACONDA AMBIENTES VIRTUAIS E BIBLIOTECAS	57
Figura 27 - IMPORTAÇÕES DE BIBLIOTECAS DO PYTHON.....	58
Figura 28 - FUNÇÃO CARREGAR DADOS	58
Figura 29 - FUNÇÃO DE TRANSFORMAÇÃO DO CLASSIFICADOR	59
Figura 30 - FUNÇÃO DE TRANSFORMAÇÕES DAS VARIÁVEIS X E Y	59

Figura 31 - FUNÇÃO DE TRANSFORMAÇÃO PARA ARRAYS NUMPY	60
Figura 32 - FUNÇÃO PRÉ-PROCESSAMENTO.....	60
Figura 33 - RNA COM OS DADOS PRÉ-PROCESSADOS	61
Figura 34 - GRÁFICO DA PERDA DE TREINO E TESTE	65
Figura 35 - GRÁFICO DA ACURÁCIA DE TREINO E TESTE	66
Figura 36 - EFICIÊNCIA DO TREINAMENTO NA SIMULAÇÃO.....	67
Figura 37 – ERRO RELATIVO DA SIMULAÇÃO DO TREINAMENTO.....	68
Figura 38 -VERIFICAÇÃO DA EFICÁCIA DO TESTE CEGO	69
Figura 39 - ERRO RELATIVO DA SIMULAÇÃO DO TESTE CEGO	70
Figura 40 - ARRAYS NUMPY	83
Figura 41 - DATA FRAME DO PANDAS	84
Figura 42 - GRÁFICOS DO MATPLOTLIB	86
Figura 43 - ESTRUTURA DO TENSORFLOW.....	87

LISTA DE TABELAS

Tabela 1 – ANÁLISE ELEMENTAR DO ÓLEO CRU (% EM PESO)	49
Tabela 2 - LITOLOGIA DE ROCHAS.....	52
Tabela 3 - QUANTIDADE DE AMOSTRAS POR LITOLOGIA	55
Tabela 4 - VALORES BINÁRIOS DAS LITOLOGIAS.....	62
Tabela 5 – PARÂMETROS DE CONFIGURAÇÕES INICIAIS	63
Tabela 6 – QUANTIDADE DE NEURÔNIOS DA CAMADA OCULTA	64
Tabela 7 - FUNÇÕES DE ATIVAÇÕES DA CAMADA OCULTA.....	64
Tabela 8 - FUNÇÕES DE ATIVAÇÃO DA CAMADA DE SAÍDA	65
Tabela 9 – PARÂMETROS DE CONFIGURAÇÕES DA REDE NEURAL.....	66

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina
AMS	Aprendizado de Máquina Supervisionado
API	Interface de Programação de Aplicativos
BD	Banco de Dados
CPU	Unidade Central de Processamento
CRISP-DM	<i>CRoss Industry Standard Process for Data Mining</i>
CSV	<i>Comma Separated Value</i>
DM	<i>Data Mining</i>
DF	<i>Data Frame</i>
DW	<i>Data Warehouse</i>
GPU	Unidade Gráfica de Processamento
IA	Inteligência Artificial
IDEs	<i>Integrated Development Environment</i>
KDD	<i>Knowledge Discovery in Database</i>
MD	Mineração de Dados
MSE	<i>Mean Square Error</i>
OLAP	<i>Online Analytical Processing</i>
OLTP	<i>Online Transaction Processing</i>
RNA	Rede Neural Artificial
SQL	<i>Structured Query Language</i>

SUMÁRIO

1 INTRODUÇÃO	14
2 FUNDAMENTAÇÃO TEÓRICA	19
2.1 MINERAÇÃO DE DADOS.....	19
2.1.1 <i>Banco de Dados</i>	20
2.1.2 <i>Metodologias para Exploração de Dados</i>	22
2.2 APRENDIZADO DE MÁQUINA.....	25
2.2.1 <i>Definição</i>	26
2.2.2 <i>Aprendizado Supervisionado</i>	26
2.2.3 <i>Aprendizado não Supervisionado</i>	27
2.2.4 <i>Aprendizado por Reforço</i>	28
2.2.5 <i>Algoritmos de Aprendizado de Máquina Supervisionado</i>	29
2.2.6 <i>Overfitting</i>	35
2.3 TEORIAS E APLICAÇÃO DO ALGORITMO DE REDE NEURAL ARTIFICIAL.....	35
2.3.1 <i>Bioinspiração</i>	36
2.3.2 <i>Neurônio Artificial</i>	37
2.3.3 <i>Arquitetura da Rede Neural Artificial</i>	39
2.3.4 <i>Perceptron</i>	40
2.3.5 <i>Multi Layer Perceptron</i>	41
2.3.6 <i>Função de Ativação</i>	43
2.3.7 <i>Função de Perda ou Custo</i>	47
2.3.8 <i>Algoritmo de Backpropagation</i>	48
3 ESTUDO DE CASO	49
3.1 PETRÓLEO	49
3.2 BASE DE DADOS.....	50
3.3 METODOLOGIA E PROPOSIÇÃO DE REDE NEURAL ARTIFICIAL	52
3.3.1 <i>O processo de Aprendizagem</i>	53
3.3.2 <i>Ferramentas utilizadas</i>	54
3.4 DESENVOLVIMENTO DA RNA	55
4 TESTES E RESULTADOS	63
5 CONSIDERAÇÕES FINAIS	71
4.1 TRABALHOS FUTUROS	71
REFERÊNCIAS BIBLIOGRÁFICAS	73

ANEXO A – CONJUNTO DE DADOS OBTIDOS DE FOURNIER & BORGOMANO (2009)	76
ANEXO B – CONJUNTO DE DADOS DE TREINAMENTO E TESTE DA REDE NEURAL ARTIFICIAL.....	78
B1 – CONJUNTO DE TREINAMENTO - DADOS ORDENADOS POR LITOLOGIA	78
B2 – CONJUNTO DE TESTE “CEGO” - DADOS ORDENADOS POR LITOLOGIA.....	80
APÊNDICE A - FERRAMENTAS, LINGUAGEM DE PROGRAMAÇÃO E BIBLIOTECAS USADAS	81
APÊNDICE B – CÓDIGOS DO PRÉ-PROCESSAMENTO E CRIAÇÃO DA RNA EM PYTHON	89

1 INTRODUÇÃO

Seja nas ciências teórica, aplicada ou nas engenharias, o reconhecimento de padrões tem se mostrado uma estratégia recorrente, como na validação de hipóteses, por exemplo. Uma ferramenta poderosa nesse segmento são as técnicas da área de Inteligência Artificial (IA), podendo ser aplicadas em alguns exemplos como: problemas de regressão, classificação e agrupamento de dados, controle de sistemas dinâmicos, reconhecimento de imagens e áudios, e até na construção de modelos preditivos (SILVA, SPATTI e FLAUZINO, 2010).

Estima-se que o mercado de IA, como um todo, movimentou cerca de 700 milhões de euros em 2013, como também era esperado que esse valor crescesse em 2015 de forma exponencial e ultrapassassem o valor de 27 bilhões de euros. E as previsões para os próximos anos tenderão a acompanhar esse crescimento, sendo que no ano de 2021 produzirão cerca de 2,9 trilhões em valor de negócio, bem como 2,3 milhões de empregos. Ou seja, um novo paradigma na tecnologia que proporcionam bons resultados econômico (OBSERVATORY, 2013; GARTNER, 2018).

A execução de ações de forma programada e sequencial não caracteriza a existência de IA, cuja sua definição é caracterizada por métodos computacionais capazes de realizar operações baseadas no raciocínio humano, tais como: pensar, aprender, tomar decisões, resolver problemas, ou seja, ser inteligente (MENDES, 2009; BRINK; RICHARDS, 2014; LÉVY, 1993).

A área de Aprendizado de Máquina (AM) é um subcampo da IA que visa a criação de algoritmos e sistemas capazes de aprender baseando-se em dados (BRINK; RICHARDS, 2014). Uma das técnicas oriundas da área de AM são as Redes Neurais Artificiais (RNAs) que são modelos estatísticos bioinspirados no funcionamento do sistema nervoso humano, principalmente do cérebro. Geralmente, sendo implementadas por métodos computacionais através dos algoritmos.

Segundo HAYKIN (2001), as RNAs são denominadas algoritmos de aprendizagem, cuja função é modificar os pesos sinápticos da rede de uma forma ordenada para alcançar o objetivo de aprendizado desejado pelo projeto.

Baseando-se em Levada, Fieri e Pivesso (1996), em um organismo complexo, os neurônios são células que compõem o sistema nervoso e têm como principal função a condução de impulsos nervosos. Para a realização de um pensamento, é necessária a ativação de diversos neurônios. Um neurônio artificial funciona de forma similar, ele recebe estímulos iniciais que são ponderados como peso as quais são aplicadas por uma função de ativação,

gerando assim uma saída. As interconexões desses diversos neurônios são chamadas de Rede Neural Artificial (RNA).

A Geologia estuda as crostas terrestre, suas formações e composições, possuindo como um subcampo a Litologia que tem como base os estudos das rochas, tamanho do seu grão, tamanho das partículas, suas características físicas e químicas, os processos pelos quais os sedimentos rochosos não consolidados são transformados em rochas sedimentares consolidados (TEIXEIRA; TOLEDO; FAIRCHILD; TAIOLI, 2000).

Visando a criação de um modelo de Aprendizado de Máquina Supervisionado (AMS), este projeto propõe a implementação e implantação de uma RNA, baseando-se nas melhores metodologias e técnicas encontradas que satisfaça o conjunto de dados a qual será elaborada. Sendo assim, utilizando-se da linguagem de programação Python como ferramenta, visa criar as condições necessárias em conjunto com o ambiente de desenvolvimento Jupyter Notebook, que permite a confecção dos chamados *notebook*, que são documentos nos quais podem conter elementos de código, texto, imagens, fórmulas matemáticas, gráfico, entre outros, facilitando o desenvolvimento, teste e visualização dos dados, pois permite a execução dos códigos em tempo real.

Este trabalho tem como objetivo geral realizar um estudo sobre Rede Neural Artificial (RNA) e aplicar um modelo de Aprendizado de Máquina Supervisionado. E, como os objetivos específicos, pretendem-se:

- Estudar a linguagem de programação Python em conjunto com suas bibliotecas, as Interfaces de Programação de Aplicativos (API) referente aos conceitos abordados no trabalho e também documentar os recursos que serão utilizados para implantação do algoritmo de RNA aplicada a litologia de rochas carbonáticas;
- Preparar a base de dados para o processo de criação da rede, o que inclui o pré-processamento dos dados para treinamento e testes de performance da RNA;
- Validar a acurácia do modelo proposto em um estudo de caso.

A escolha desse tema justifica-se dada as mudanças da atual revolução industrial que difere das outras na qual possuíam uma matriz que a impulsionaram, como os exemplos, da máquina a vapor, a eletricidade e a microeletrônica, isto é, ondas tecnológicas se disseminaram de maneira centralizada, nesta quarta revolução trouxe como características várias tecnologias que convergiram. Assim, os domínios físicos, digitais e biológicos estão

inter-relacionados e se beneficiam uns dos outros, gerando a relevância e a possibilidade para as aplicações de técnicas de IA (Schwab, 2016).

O amplo emprego de RNAs deve-se à capacidade de ser programada para operar em tempo real, alterando-se rapidamente em ambientes onde as estatísticas mudam o tempo todo, demonstrando capacidades de aprendizagem inspiradas no cérebro humano (HAYKIN, 2001; SILVA, SPATTI e FLAUZINO, 2010).

Visto que a produção científica tem como objetivo a concepção do conhecimento e sendo as RNAs um campo da Ciência da Computação a qual possui raízes e aplicações multidisciplinares, referenciando os aspectos teóricos que constitui a oportunidade de aquisição de uma variabilidade de conceitos, levando ainda em consideração as práticas das quais realizam do ato de fazer inferência nos dados passados para a descoberta de informações ou padrões, ou seja, tendo da extração e do processamento dos dados para a elaboração da informação, justifica a construção desse trabalho do ponto de vista científico-acadêmico.

E, para os alunos do curso de Sistemas de Informação mostra a evolução da tecnologia e como ela se readapta no contexto na qual está inserida, relacionando aspectos de disciplinas aprendidas durante a graduação, reforçando suas teorias e evidenciando suas aplicações práticas neste trabalho.

A metodologia adotada para o desenvolvimento deste estudo consiste em estudar, compreender, e estabelecer os métodos utilizados, tendo como base o nível de aplicação, examinar, descrever e avaliar métodos e técnicas de pesquisa que possibilitam a coleta e o processamento de informações, visando ao encaminhamento e a resolução de problemas e/ou questões de investigação.

Assim, será estudado sobre as melhores técnicas de RNAs e desenvolvimento em linguagem Python, para que haja a implementação e a implantação dela visando o treinamento da rede e o reconhecimento de padrões.

O método científico adotado neste estudo é do tipo dedutivo. Esse método é considerado racionalista, pois a partir de princípios, leis ou teorias consideradas verdadeiras e indiscutíveis, prediz a ocorrência de casos particulares com base na lógica, encontrando assim a ampla aplicação nas ciências exatas. Visto que a aplicação deste trabalho aborda a multidisciplinaridade das ciências exatas, como estatística, matemática, computação científica e física, reforçando os princípios e teorias existentes os quais fundamentam a base dessa pesquisa, criando conceitos e técnicas necessárias.

A pesquisa bibliográfica desenvolvida consiste no estudo sobre os conceitos de

mineração de dados, aprendizados de máquina, redes neurais e também a linguagem de programação Python, garantindo o conhecimento necessário para utilização de suas metodologias.

A abordagem deste trabalho é quantitativa, o que significa traduzir em números as opiniões e informações para classificá-las e analisá-las, utilizando-se das técnicas da ciência estatística, que serão fundamentais para discussões de resultados sobre o aprendizado de máquina AM e algoritmos inerentes.

Dito isso, utilizando-se de um ambiente propício para o desenvolvimento do algoritmo da rede (RNA), conhecido como Jupyter Notebook, auxiliando no controle e manipulação de seus estudos, como também foi utilizado de ferramentas apropriadas para realização dos testes, observações e visualização dos resultados. O estudo de caso proposto visa a aplicação de RNA para previsão de litologias geológicas em poços de interesse da Engenharia de Petróleo da literatura.

O trabalho estrutura-se da seguinte forma:

O primeiro (presente) capítulo se inicia contextualizando sobre o AM, apresentando brevemente também os conceitos de RNAs, como também descrever os principais pontos abordados nesse trabalho, seus objetivos, justificativa e metodologia, dando a introdução ao leitor sobre o assunto.

O segundo capítulo é dedicado para descrever sobre Mineração de Dados (MD), contendo os principais pontos, técnicas que fazem parte da área em questão e a relação com o AM, descrevendo sobre aprendizado supervisionado e não supervisionado, pré-processamento de dados, apresentando brevemente sobre as principais técnicas dessa área. Também são discutidas teorias e aplicações Sobre RNA, detalhando a bioinspiração, o contexto histórico, a topologia da rede, e o processo de aprendizado.

No terceiro capítulo apresenta-se a proposta da definição do estudo de caso para previsão litológica em dados petrofísicos alinhados a perfis de poços, contextualizando sobre os conceitos inerentes para criação do minimundo, assim definindo requisitos necessários para o desenvolvimento da RNA, construídos diagramas de caráter necessário para o processo de desenvolvimento do projeto, finalizando com a proposta do modelo de RNA que satisfaça o estudo de caso detalhado.

No quarto capítulo serão apresentados os resultados obtidos do desenvolvimento da RNA e a coleta de dados referente à aplicação desta para análise verificando a acurácia do modelo.

O quinto capítulo traz as considerações finais, onde se podem observar as conclusões

do trabalho e indicações de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir, serão apresentadas as abordagens teóricas referentes aos conceitos de Mineração de Dados (MD), Aprendizado de Máquina (AM) e teoria de aplicação e configuração de uma rede neural artificial (RNA).

2.1 Mineração de Dados

As evoluções tecnológicas e uso cotidiano dos meios de informação, e como elas conectam-se a internet, propiciam a produção de uma grande escala de dados diariamente, como também sua variabilidade. Alguns exemplos são: transações comerciais, comunicação em redes sociais, compartilhamento de fotos, vídeos, áudios e arquivos no geral, entre outros. Logo, a MD auxilia a buscar padrões que proporcionem inferir conhecimento, como tratados a seguir.

Segundo Carvalho (2005), a tecnologia, mais do que nunca, faz parte do cotidiano da vida das pessoas, a quantidade de dados gerados só tende a aumentar. Com a evolução da tecnologia, estes dados podem ser armazenados em grandes repositórios de dados, como os Banco de Dados (BD), *Data Warehouses* (DW), entre outros. Contudo, apesar da grande quantidade de dados armazenados, muitas instituições apenas os guardam para fins operacionais e após o uso são descartados, ou deixados de lado, sem a consideração de que podem representar (e representam) fonte de informações valiosa para elas.

Pode existir uma mina de ouro em dados brutos, segundo Witten e Frank (2002), vivemos em uma situação “rica em dados, mas pobre em informação”. Mas, sem as ferramentas certas e necessárias, não é possível obter essas pedras preciosas, ou seja, da mesma forma que os mineradores utilizam técnicas e ferramentas necessárias para obtenção e extração de pedras preciosas no meio de tanto minério que não possui tanto valor a ser considerado, tem-se a mesma necessidade para exploração dos dados brutos, produzindo informações valiosas, tornando-se o diferencial da realidade em que vivemos, onde a informação é considerada a chave para o conhecimento.

2.1.1 Banco de Dados

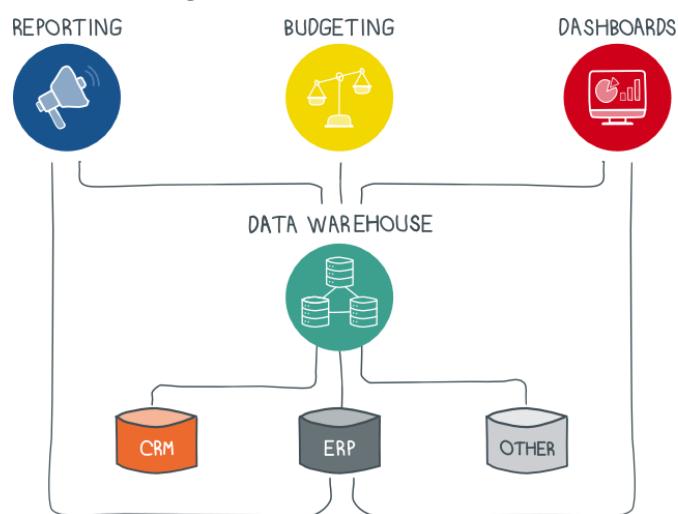
Existe uma variabilidade de tipos de bancos de dados, como também diversas aplicações diferentes no nosso cotidiano. Um exemplo básico disso é a lista de telefone a qual guarda informações pertinentes de um contato. Apesar do conceito-base ser diversificados por muitos autores, o significado principal de um BD é um agrupamento de informações que estabelecem significados, ou uma relação entre si (DATE, 2004; SILBERSCHATZ, 1999).

Segundo Date (2004), os bancos de dados se beneficiam com a evolução dos computadores e processamento em paralelo, oportunizando guardar arquivos digitais, garantido armazenamento e consulta de forma quase instantânea nos dias atuais.

2.1.1.1 Data Warehouse

Segundo Han e Kamber (2012), DW pode ser considerado a junção de vários repositórios de dados unificados em um repositório central a qual armazena todas as informações agrupadas nele, facilitando o gerenciamento dos mesmos. Um exemplo poderia ser dado por uma rede de livraria a qual possui várias unidades espalhadas por países, estados e cidades, cada uma dessas unidades possui um BD respectivamente, contendo suas informações. Nesse contexto, o DW seria o repositório central de informações dessa livraria.

Figura 1 - DATA WAREHOUSE



Fonte: <https://blog.adverity.com/data-warehouse-marketing-better-faster-insights>

(Acessado em 16/04/2019)

Observa-se na Figura 1 que o DW encontra-se no centro da imagem, fazendo uma analogia ao fato de ser o repositório central de dados que contém as informações dos demais repositórios, vindo de diferentes formas, ou seja, através dele pode consultar as demais fontes de dados, como também produzir relatórios resumidos desses dados.

Há eficientes métodos de OLTP (*Online Transaction Processing* - Processamento de Transações Online), no entanto, para difundir o acesso ao DW para seus múltiplos clientes, ferramentas como o OLAP (*Online Analytical Processing* - Processamento Analítico Online) tornou-se necessária, ou seja, técnicas de análise que permitem a visualização dos dados de maneira multidimensional.

Segundo Stair (2010), a principal diferença entre OLTP e OLAP é, que enquanto o primeiro método trabalha com cenários pré-definidos, como por exemplo, consultas de transações em bancos de dados relacionais, o segundo pode trabalhar com uma quantidade muito maior de dados em caminhos que não precisam ser pré-definidos.

Os sistemas OLAP utilizam-se da extensa indexação dos DW para possibilitar o acesso e a apresentação gráfica de pedaços dos dados combinados praticamente de qualquer modo desejado pelo usuário, permitindo rápida generalização, gerando totais a partir de registros individuais, ou o contrário.

Normalmente, os dados são estruturados em forma de um cubo ou de uma tabela tridimensional, na qual as variáveis se relacionam por meio de seus significados comerciais. Assim, o usuário pode percorrer essa tabela da forma que desejar, e realizar com esse dado o que consideram importantes, emitindo relatórios parciais ou totais (CARVALHO, 2005).

De acordo com Thomsen (2002), o OLAP é um sistema flexível e rápido de consultas, muitos fazem confusão com mineração. No entanto, sistemas OLAP não são considerados técnicas de mineração e sim uma ferramenta que em conjunto com técnicas de análise de dados usadas em um DW, por meio de consultas *Structured Query Language* (SQL) para visualização e combinação de dados permitem a análise de eventos.

Vale ressaltar, que vivemos na era do *Big Data* em que grande quantidade de dados produzidos diariamente, tornando as tarefas de análise de dados em um DW árdua, fazendo em que o usuário dessa técnica necessite de conhecimento específico, lembrando também que alguns padrões são escondidos e não são encontrados com facilidades devido ao volume e variabilidade desses dados. Portanto, a necessidade de realizar técnicas de Mineração de Dados (*Data Mining*) é cada vez mais pertinente para descobrir padrões ocultos (REZENDE, 2003).

Existem diversas definições para o que é conhecido como Mineração de Dados que

podem ser vistas como a evolução natural da tecnologia da informação, que possui um potencial de transformar grandes quantidades de dados em conhecimento. Baseando-se em Han e Kamber (2012), seria “o processo de descoberta de padrões interessantes em grandes blocos de dados, armazenados em BD, DW, ou outros repositórios de informação”. No entanto para Hand, Smyth e Mannila (2001), seria “a análise de grandes conjuntos de dados observacionais para encontrar relações não suspeitas e resumir tais dados em estados que possam ser entendidos e utilizados pelo dono dos dados”.

Alguns autores também consideram a mineração como um processo do *Knowledge Discovery in Database* (KDD), uma possível tradução seria a descoberta de conhecimento através dos dados, outros consideram o KDD como um sinônimo para o processo de mineração. Ainda há uma confusão entre os dois termos, todavia eles estão relacionados (FAYYAD, PIATETSKY-SHAPIRO E SMYTH, 1996).

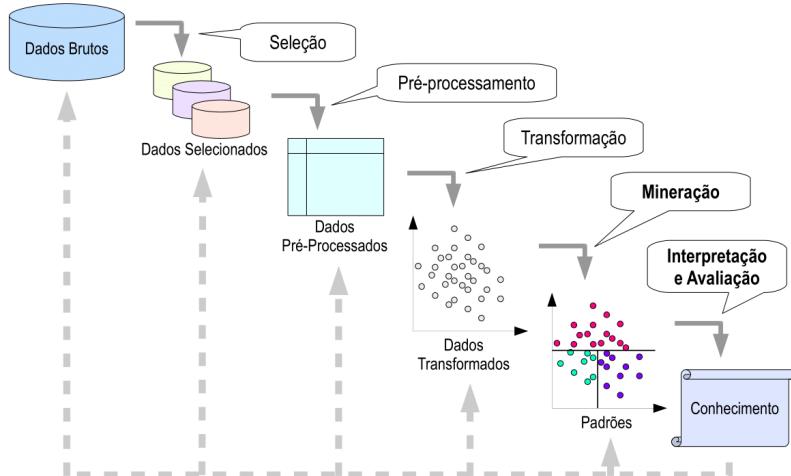
2.1.2 Metodologias para Exploração de Dados

A análise de dados é uma área considerada complexa e nem sempre é fácil conseguir *insights* visualizando um conjunto de dados, esperando que alguma informação apareça. Nesse contexto, foram desenvolvidas técnicas de análise exploratória de dados.

2.1.2.1 KDD

O KDD é um dos métodos mais antigos existentes, seu surgimento foi ao final da década de 1980, focando na questão de descoberta do conhecimento a partir dos dados, aplicando algoritmos específicos para extrair padrões de dados (FAYYAD, 1996).

Figura 2 - CICLO DO KDD



Fonte: Adaptado de Fayyad (1996)

Pode-se observar na Figura 2 que o processo do KDD apresenta um ciclo dividido em seis etapas, que podem ser organizadas em três grandes processos: Pré-processamento de Dados, Mineração de Dados e Pós-processamento de Dados.

Baseando-se na Figura 2, os passos 1 a 4, são as formas de se preparar os dados, eliminados possíveis divergências. Ou seja, nesses passos estão o processo de pré-processamento de dados que consiste de: Extração e Integração, Limpeza, Seleção e Transformação.

O processo de Extração e Integração tem por finalidade a união dos dados em uma única fonte, então os dados brutos são organizados em um mesmo repositório. Algumas vezes os dados a serem processados estão armazenados em diversos repositórios, tais como vários BD ou até mesmo DW, arquivos de textos, arquivos *Comma Separated Value* (CSV). Essa ocorrência é frequente, existem organizações que possuem uma matriz e diversas outras filiais. Essa etapa fica responsável pela padronização desses dados.

A limpeza de dados é responsável pela remoção de possíveis inconsistências nos dados que possam atrapalhar o processo de descoberta de conhecimento. Ao obter um conjunto de dados podem existir dados que vieram como nulos ou com algum padrão não normal perante aos demais. Esse processo tem o objetivo de identificar e eliminar esses possíveis erros, normalmente é recomendado ser executado por alguém que conhece o negócio (*Business*) e o dado desejado.

Algumas organizações possuem base de dados muito grande e parte desse conjunto de dados acabam não fazendo parte do processo da Figura 2, visto que não são coerentes para

descoberta de algum padrão. Sendo assim, a etapa Seleção visa analisar os dados que podem ser importantes para o processo de mineração ou aprendizado de máquina, retirando os que não possuem relevância para tais atividades.

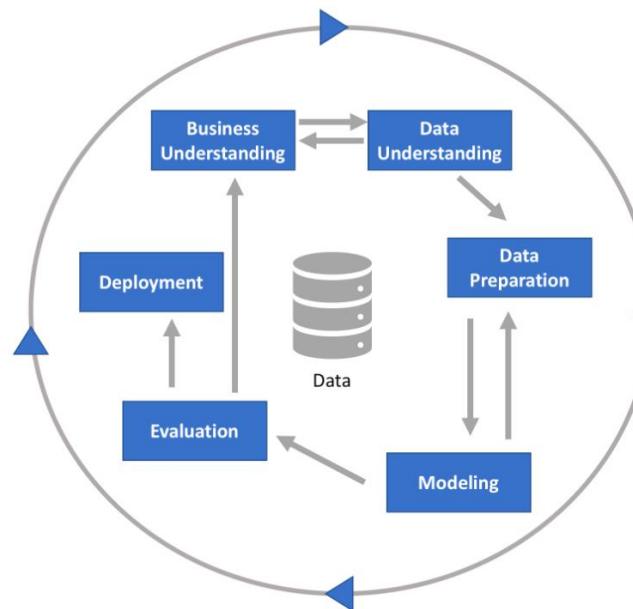
No entanto, segundo Brink e Richads (2014), ao realizar a seleção de algumas características, o modelo será prejudicado, existindo a possibilidade de não saber identificar o que realmente é um padrão ou um conjunto de dados que destoam dos padrões que estão presentes na base de dados, como também, omite informação pertinente, o que afeta a precisão da análise.

Há ainda o último passo na Figura 2, que é a Transformação de Dados, visando transformar e consolidar os dados apropriadamente para mineração via operações de redução e/ou agregação. Após todos esses passos, estarão aptos para a realização do processo de mineração (HAN; KAMBER, 2012).

2.1.2.2 CRISP-DM

O *Cross Industry Standard Process for Data Mining* (CRISP-DM) que possui uma possível tradução livre “Processo Padrão para Indústria Cruzada de mineração de dados”, é uma técnica concebida no final de 1996 que consistem em auxiliar as pessoas envolvidas no processo de análise de dados.

Figura 3 - CICLO DO CRISP-FM



Fonte: <http://crisp-dm.eu/> (Acessado em 20/04/2019)

Observa-se na Figura 3 que a metodologia CRISP-DM apresenta um ciclo dividido em seis etapas, a qual as setas identificam a dependência entre as fases correlacionadas.

A primeira etapa da Figura 3, Compreensão do Negócio (*Business Understanding*), está relacionada com a identificação do problema, definindo os objetivos e requisitos do projeto.

Na segunda etapa da Figura 3, Compreensão dos Dados (*Data Understanding*), consistem em coletar, organizar, documentar os dados para análise exploratória, também está vinculada a essa etapa tratar os dados, podendo identificar padrões, insights relacionados ao projeto nessa etapa.

O terceiro passo é a Preparação dos Dados (*Data Preparation*), preparação dos dados para a modelagem (Figura 3). Essa etapa consiste em selecionar os dados para a modelagem, pré-processamento dos dados e integração dos dados.

A etapa quatro da Figura 3 é a Modelagem dos Dados (*Modeling*), onde será construído o modelo com base nos objetivos do projeto, não necessariamente criará apenas um modelo, poderá ter a criação de vários e realizar a comparação no próximo estágio, verificando com isso qual é o melhor modelo para a abordagem do projeto.

No estágio cinco da Figura 3 define-se a Avaliação (*Evaluation*), do modelo proposto, ou seja, avaliar os resultados, verificando se os critérios definidos na primeira fase foram atingidos. Se não foi, entender o que deu errado para depois determinar um novo escopo e repetir o processo.

Por último, tem-se a etapa Desenvolvimento (Deployment) (Figura 3), colocando o modelo em produção para que possa ser usado, avaliando o que deu certo, o que deu errado e o que pode ser melhorado para projetos futuros.

2.2 Aprendizado de máquina

Segundo Russel e Norving (2013), computadores solucionam problemas por meios de instruções de programas, que define uma sequência de passos ou ações que serão realizadas para haver o processamento de dados. Ações dinâmicas como reconhecimento de padrões são realizadas facilmente por seres humanos por processo de aprendizagem, identificação de padrões e associação dessas características a uma pessoa ou objeto, mas, analisar grandes volumes de informações tornam essas tarefas difíceis ou impossíveis de serem realizadas pelos mesmos. E nesse contexto que entram as técnicas de AM,

solucionando problemas complexos como o reconhecimento de padrões.

2.2.1 Definição

Baseando-se em Brink e Richards (2014), AM é considerada um subcampo da área de IA em que é “uma abordagem guiada a dados para a resolução de problemas, na qual padrões ocultos em um conjunto de dados são identificados e utilizados para ajudar na tomada de decisões”. Ou seja, utiliza-se dos dados do conjunto de treinamento para generalizar as instâncias, criando uma regra de definição do problema e ao ser inserido um conjunto de outros problemas seja capaz de informar uma solução a qual foi apreendida no treinamento e isso gera a tomada de decisões baseadas em dados.

2.2.2 Aprendizado Supervisionado

Segundo Wintten e Frank (2005), os algoritmos de AMS consistem em encontrar uma função a partir dos dados de treinamento que possa ser utilizada para prever um rótulo ou valor que caracterize um novo exemplo, com base nos valores de seus atributos de entrada. Para isso, cada objeto do conjunto de treinamento deve possuir atributos de entrada e saída.

Esses algoritmos seguem o paradigma de aprendizado supervisionado que o termo vem da simulação da presença de um supervisor externo na qual conhece a saída (rótulo) desejada para cada exemplo. Com isso o supervisor externo pode avaliar a capacidade de hipótese induzida de predizer o valor de saída para novos modelos, podendo avaliar o erro entre o que o modelo avaliou e o classificador da avaliação.

Figura 4 - FLUXOGRAMA DE APRENDIZADO DE MÁQUINA



Fonte: <https://luisfred.com.br/machine-learning/machine-learning-a-matematica-da-aprendizagem-supervisionada> (Acessado em 30/04/2019)

Pode-se observar na Figura 4, um fluxo de criação de um algoritmo de AM, sendo as entradas os conjuntos de dados de treinamento. A partir daí o modelo generaliza e cria o modelo preditivo, onde pode receber um novo conjunto de dados para realizar previsões com base no que foi aprendido. Ou seja, aprende um padrão e determina uma regra para solução do problema.

2.2.3 Aprendizado não Supervisionado

Como o Aprendizado não Supervisionado é uma abordagem utilizada na descoberta de padrões não categóricos, sendo assim, também está relacionado a exploração de dados. O conjunto de dados abordados nesse tipo de aprendizagem não possui classificação, ou seja, uma saída esperada para cada uma de suas instâncias. O objetivo é encontrar dados de alguma forma ou descrever suas estruturas, não existe uma construção de modelos preditivos nessa abordagem, e sim um modelo cuja função seja encontrar regularidades nos dados a fim de serem úteis.

Vale ressaltar que não podemos prever em que tipo de características ou agrupação irá determinar o modelo para predizer a ocorrência de algo, ou seja, não existi a possibilidade de classificação que são pertencentes aos algoritmos supervisionados.

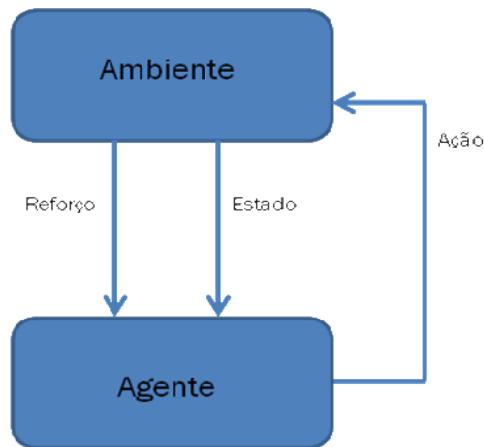
Apesar de frequentemente ser utilizada para atividades de exploração de dados (FURNKANZ; GAMBERGER; LAVRAC, 2012), em conjunto com algoritmos supervisionados para criar conjuntos de dados de teste ou encontrar novos padrões em

determinado conjunto de dados, o Aprendizado Não-Supervisionado também pode ser utilizado para fins preditivos.

2.2.4 Aprendizado por Reforço

Os algoritmos deste tipo de aprendizagem consistem em reforçar ou recompensar os estímulos que são considerados positivos, ou seja, as ações que são desejadas para o aprendizado em questão são compensadas para que o algoritmo possa reproduzi-las, pois são consideradas como resposta para uma saída ideal.

Figura 5 - ALGORITMO DE APRENDIZADO POR REFORÇO



Fonte: <https://medium.com/@avinicius.adorno/introdu%C3%A7%C3%A3o-a-aprendizado-de-m%C3%A1quina-e39ec5ef459b>
 (Acessado 02/05/2019)

Podemos observar na Figura 5 um exemplo de diagrama proposto para Aprendizagem por Reforço, onde o agente é que faz a ação, a ação é o movimento executado pelo o agente, o ambiente é onde a ação foi executada, o estado são as possibilidades daquela ação e a recompensa é determinada com base no estado em que se almeja.

Um exemplo de problema proposto para esse algoritmo seria jogar um jogo de corrida, verificamos as ações com o decorrer do jogo, por exemplo: o agente seria o carro, as ações possibilidades de movimentação que o carro pode executar, o ambiente seria o jogo e o

estado e o que esperamos em cada momento determinado, ou seja, aonde existir uma curva o estado desejado seria que o carro, o agente, fizesse a curva, avaliamos a ação que o agente tomou nesse momento com o ambiente e se o estado desejado foi atingido, estimulamos uma recompensa para que o agente entenda que aquele estado é o almejado.

Como o foco deste trabalho é a respeito dos algoritmos de AMS, os algoritmos de Aprendizado Não-Supervisionado e Aprendizado por Reforço não serão tratados neste trabalho.

2.2.5 Algoritmos de Aprendizado de Máquina Supervisionado

Há diversos algoritmos na área de AM, cada qual melhor indicado a uma abordagem específica, finalidade do problema, quantidade de dados, linearidade dos dados ou não, números de parâmetros, números de recursos, complexidade do problema, acurácia, tempo, entre outros parâmetros são algumas das abordagens. Esses algoritmos, normalmente são subdivididos em grupos, sendo os principais: classificação, clusterização, redução de dimensões e regressão.

Segundo Carvalho (2005), os algoritmos de AMS são definidos pelos grupos de classificação ou regressão, no entanto alguns modelos se beneficiam dos dois paradigmas, sendo os algoritmos mais usuais e que serão citados nesse trabalho, os descritos abaixo:

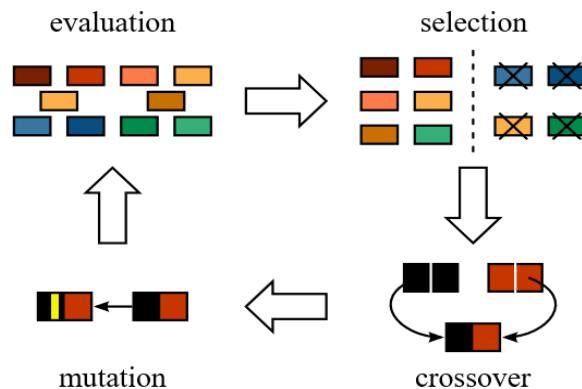
- Algoritmos Genéticos
- Árvores de Decisão
- *K-Nearest Neighbor*
- Regressão Linear e Logística
- Redes Neurais Artificiais

2.2.5.1 Algoritmos Genéticos

De acordo com Carvalho (2005), os algoritmos genéticos assim como as redes neurais têm sua bioinspiração na biologia humana, são algoritmos inspirados na teoria da evolução de Darwin e nas teorias de genética de Mendel, incorporando conceitos da evolução, seleção, reprodução e mutação da genética. Durante cada iteração, esses princípios citados

acima são aplicados a uma população de candidatos que existe uma variação, através da seleção é determinado a evolução dos algoritmos que irão se reproduzir, gerando n descendentes para a próxima evolução, até se ter o melhor algoritmo para solução viável.

Figura 6 - CICLO DO ALGORITMO GENÉTICO



Fonte: <http://www.electicalelibrary.com/2018/04/13/o-que-e-algoritmo-genetico/>
 (Acessado em 05/05/2019)

Observa-se na Figura 6 um exemplo de ciclo de vida do algoritmo genético, as setas definem o fluxo dessas sequências com base nas iterações definidas para alcançar a resolução do problema proposto, onde temos a evolução, seleção, validação cruzada e mutação que irão se repetir por um número selecionado de iteração e no final do processo garantir a diversidade da população, pois se não houver essa diversidade os algoritmos irão ficar na seleção imatura que são as seleções locais.

2.2.5.2 Árvore de Decisão

As árvores de decisão são algoritmos que têm a representação de uma tabela de decisões baseadas em árvores, expressando os mesmos dados da tabela, por meio dessa abordagem. Existem diversos tipos de algoritmos de árvores de decisões tendo suas aplicações diversificadas, sendo que cada um desses tipos de algoritmo são indicados para determinadas aplicações, como também por suas características de funcionamento. Ou seja, baseando-se na probabilidade, somado e ponderado as ocorrências de eventos e assim determinado seu resultado com base em estimativas (FURNKRANZ; GAMBERGER; LAVRAC, 2012).

Figura 7 - ALGORITMO DE ÁRVORE DE DECISÃO TABULAR

Exemplos de Treino

Dia	Aspecto	Temp.	Humidade	Vento	Jogar Ténis
D1	Sol	Quente	Elevada	Fraco	Não
D2	Sol	Quente	Elevada	Forte	Não
D3	Nuvens	Quente	Elevada	Fraco	Sim
D4	Chuva	Amenos	Elevada	Fraco	Sim
D5	Chuva	Fresco	Normal	Fraco	Sim
D6	Chuva	Fresco	Normal	Forte	Não
D7	Nuvens	Fresco	Normal	Fraco	Sim
D8	Sol	Amenos	Elevada	Fraco	Não
D9	Sol	Fresco	Normal	Fraco	Sim
D10	Chuva	Amenos	Normal	Forte	Sim
D11	Sol	Amenos	Normal	Forte	Sim
D12	Nuvens	Amenos	Elevada	Forte	Sim
D13	Nuvens	Quente	Normal	Fraco	Sim
D14	Chuva	Amenos	Elevada	Forte	Não

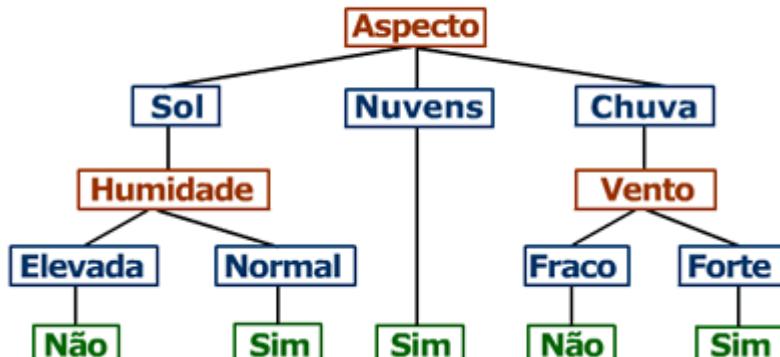
Fonte:

<http://web.tecnico.ulisboa.pt/ana.freitas/bioinformatics.ath.cx/bioinformatics.ath.cx/indexf23d.html?id=199> (Acessado 05/05/2019)

Observa-se na Figura 7 um exemplo de uma tabela de decisão genérico em que mapeia o dia, aspectos climatológicos daquele dia, a temperatura, umidade e assim a decisão de ir ou não jogar tênis.

Figura 8 - FLUXO DO ALGORITMO DE ÁRVORE DE DECISÃO

Árvore de Decisão para Jogar Ténis



Fonte:

<http://web.tecnico.ulisboa.pt/ana.freitas/bioinformatics.ath.cx/bioinformatics.ath.cx/indexf23d.html?id=199> (Acessado 05/05/2019)

A Figura 8 mostra um exemplo de diagrama de algoritmo de árvore de decisão, baseado na Figura 7 que é a representação da tabela, em que o nó raiz é o problema em questão, cada nó filho dessa raiz, denominado nó de decisão testa um atributo e que são atreladas as folhas que determina uma classificação desse nó em questão.

O problema em questão é jogar ténis, o nó raiz é determinado pelos aspectos que influenciam na decisão de jogar ténis ou não. Sendo assim, os nós filhos deste nó principal determinam variáveis a serem analisadas para a tomada de decisão, e as folhas as classificações da ocorrência dessas variáveis com a decisão de ir jogar ténis ou não. No final é calculada a probabilidade da ocorrência de cada um desses eventos e assim determinada a regra do modelo com bases no conjunto de entrada supervisionado pelo classificador de saída.

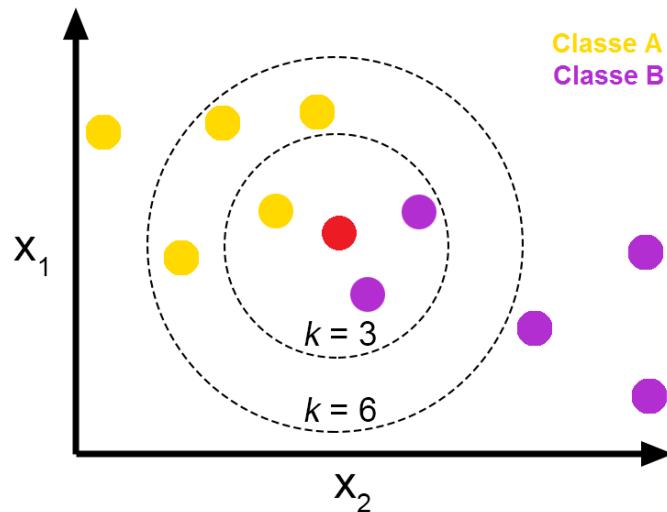
2.2.5.3 K-Neatest Neighbor

O *K-Neatest Neighbor (KNN)* é um método de classificação que calcula os vizinhos mais próximos da instância atual (entrada) e o rotula a classe corresponde a ela (classificador). Ou seja, determinar um classificador de uma amostra baseado nas amostras vizinhas (*K-Vizinhos*) do conjunto de treinamento.

Segundo Brink e Richards (2014), esse algoritmo é simples, facilmente compreendido e eficiente, todavia, deve-se projetar com cautela a solução que utilizará este método para que esta não fique excessivamente lenta e custosa e caso o problema possua

grande número de amostras o algoritmo consome muito tempo computacional. Vale lembrar que o método é totalmente dependente da escolha do K.

Figura 9 - GRÁFICO DE CLASSIFICAÇÃO DO KNN



Fonte: <https://medium.com/brasil-ai/knn-k-nearest-neighbors-1-e140c82e9c4e>
(Acessado em 07/05/2019)

A Figura 9 representa graficamente um modelo de KNN simples, tal que o círculo vermelho é considerado uma nova amostra das possíveis outras amostras já evidenciadas pelas classes A (círculo amarelo) e a classe B (círculo roxo).

O gráfico de duas coordenadas x_1 e x_2 apresenta também dois KNN possíveis, $K = 3$ e $K = 6$ que serão usados para averiguar em qual classe a nova instância pertencente. Ou seja, o K escolhido é o mais próximo possível daquela instância, nesse caso o $K = 3$, sendo que 1 amostra é da classe A e 2 amostras da classe B, a nova instância é classificada como classe B. As definições de métricas de distâncias e valores de K baseiam em fórmulas estatísticas para sua escolha.

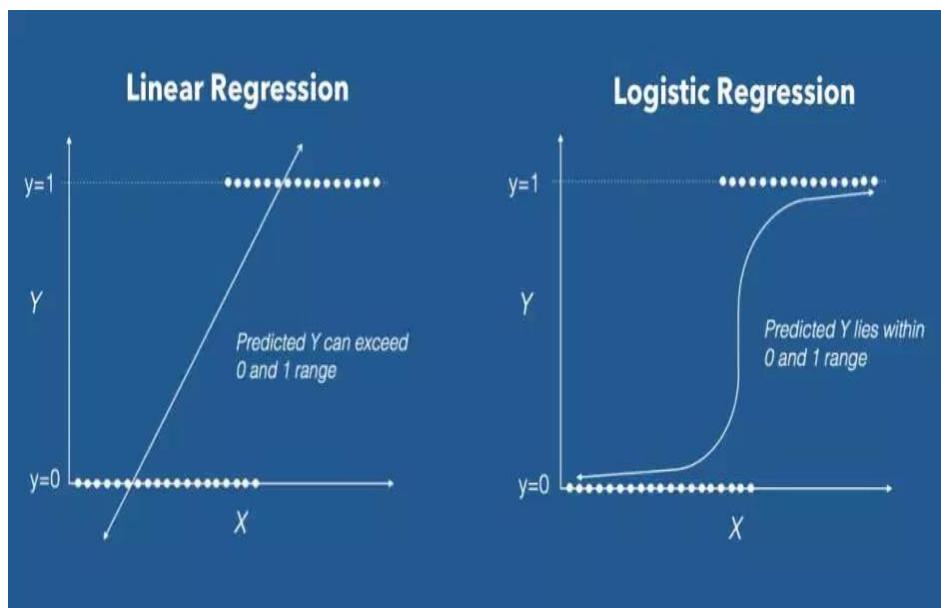
2.2.5.4 Regressão Linear e Logística

A regressão linear é um algoritmo onde a saída prevista é contínua e tem um declive constante. São modelos matemáticos que relacionam o comportamento de uma variável **X** que é a variável independente, a entrada do modelo, com outra variável **Y** que é a variável dependente, a saída do modelo. O modelo é chamado de simples quando envolve apenas duas

variáveis, e múltiplo quando possui mais de duas variáveis.

A regressão logística é um algoritmo de classificação que prever a probabilidade de uma variável categórica. Normalmente é utilizada em problemas de classificações binárias ou multiclasse, onde a variável dependente **Y** contém dados codificados com 1 (sim, ou sucesso, ou outros sinônimos correlacionados) e 0 (não, fracasso, ou outros sinônimos correlacionados). Ou seja, prevê Y em função de X.

Figura 10 - GRÁFICOS DAS REGRESSÕES LINEAR E LOGÍSTICA



Fonte: [\(Acessado em 10/05/2019\)](https://medium.com/greyatom/logistic-regression-89e496433063)

Na Figura 10, há dois gráficos distintos para correspondência X e Y. O primeiro representa um gráfico de regressão linear e o segundo de regressão logística.

2.2.5.5 Rede Neural Artificial

Sendo a RNA o tema principal deste trabalho, merece uma explicação detalhada sobre esse algoritmo, assim, a seção 2.3 (Teorias e Aplicações da Rede Neural Artificial), abordará os assuntos principais dessa abordagem.

2.2.6 Overfitting

Segundo Domingos (2012), um dos principais erros cometidos ao avaliar o modelo de AM é achar que, pelo seu modelo por ter sido eficiente no treinamento ele é um excelente modelo, ou seja, por ele ter alcançado boa acurácia durante a fase de treino, a regra gerada desse modelo satisfaça uma condição eficiente em qualquer conjunto de dados. Por isso é necessário separar o conjunto de dados entre treinamento e teste, para averiguar e evitar um sério problema conhecido como *overfitting* que é considerado o grande problema dos modelos de AM.

Mitchell (1997) fornece uma definição sobre o problema: “dado um modelo H , é dito que H causa *overfitting* no conjunto de dados de treinamento se existe uma segunda hipótese H' , em que a taxa de erros de $H < H'$, em relação aos dados de treinamento, mas a taxa de erros de $H' < H$ em relação ao conjunto total de dados”.

Os modelos que realizam boas previsões no conjunto de treinamento, não significam que a regra gerada por ele satisfaça quaisquer outros conjuntos de dados, ou seja, ele apenas pode ter se ajustado ao conjunto de dados de treinamento, tornando-se um especialista nele. Um exemplo disso seria um classificador que previu 99% no conjunto de dados de treinamento, mas no conjunto de teste atingiu menos de 50% das previsões, isso quer dizer que o modelo se ajustou ao conjunto de dados de treino, em outras palavras o modelo teve *overfitting*.

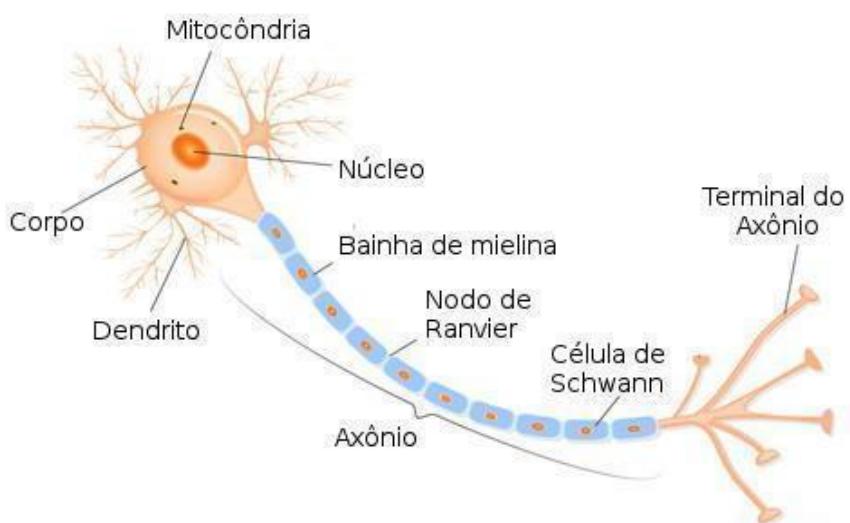
2.3 Teorias e Aplicação do Algoritmo de Rede Neural Artificial

O cérebro humano é um órgão altamente poderoso e complexo capaz de processar uma grande quantidade de informações em tempo mínimo. As unidades principais, ou células, do cérebro são os neurônios e são nelas que ocorrem o processamento e as transmissões das informações. Assim, diversos pesquisadores se inspiraram nesse modelo para tentar simular o funcionamento dele, principalmente o processo de aprendizagem. Os resultados dessas pesquisas levaram ao surgimento do modelo de neurônio artificial e posteriormente com as interconexões destes neurônios surgiu o modelo de RNA.

2.3.1 Bioinspiração

O cérebro humano tem a característica estrutural de processar informações de forma distribuída e paralela, tendo como elemento básico para esse processamento o neurônio. Como também tem uma propriedade chamada de plasticidade que permitem os neurônios mudar a natureza de suas conexões com outros neurônios em resposta a eventos que ocorram, permitindo assim o aprendizado (COPPIN 2013).

Figura 11 - NEURÔNIO BIOLÓGICO



Fonte: <https://infoescola.com/sistema-nervoso/neuronios/>

(Acessado em 10/05/2019)

Pode-se observar na Figura 11 a representação da célula elementar do sistema nervoso, o neurônio, cujo seu papel é conduzir impulsos sob determinadas condições de operação. Assim, o neurônio pode ser resumidamente dividido em três partes principais: dendritos, corpo celular (soma) e o axônio.

As informações provenientes de outros neurônios chegam ao neurônio em questão pelos dendritos, ou seja, sua função está relacionada em captar, de forma contínua, os estímulos vindos de diversos outros neurônios ou do próprio meio externo onde podem estar em contato.

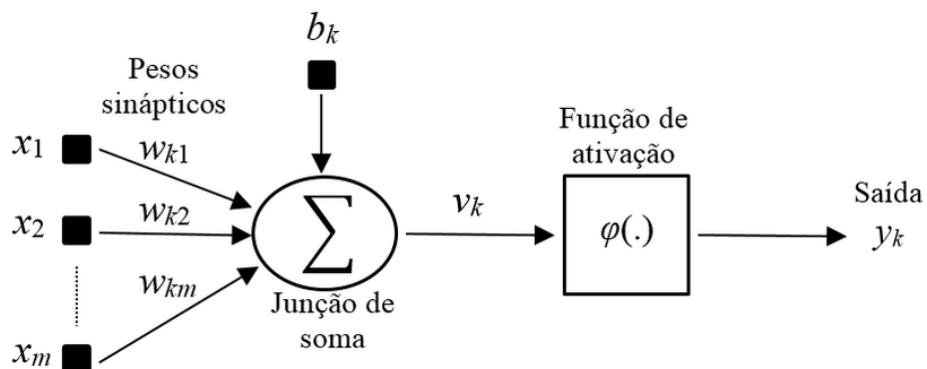
O corpo celular é incumbido de processar todas as informações trazidas pelos dendritos a fim de produzir um potencial de ativação que poderá disparar um impulso elétrico ao longo de seu axônio, que é constituído de um único prolongamento condutor de impulsos e

tem a responsabilidade de conduzir os impulsos nervosos. Ao final deste prolongamento estão situadas as terminações sinápticas, conectando o neurônio em questão a outro, passando a informação adiante. No entanto essas conexões não são diretas, necessitando de substâncias neurotransmissoras para sua comunicação.

2.3.2 Neurônio Artificial

O neurônio artificial é um modelo simplificado do neurônio biológico que são modelos inspirados a partir da análise da geração e propagação de impulsos elétricos pela membrana celular dos modelos biológicos. Esses modelos são não-lineares, fornecem saídas tipicamente contínuas, e realizam funções simples, como coletar sinais existentes em suas entradas, agregá-los de acordo com sua função de ativação (HAYKIN, 2001).

Figura 12 - NEURÔNIO ARTIFICIAL



Fonte: Haykin (2011)

A Figura 12 representa um neurônio artificial por métodos matemáticos e são constituídos pelas seguintes estruturas: sinais de entrada, pesos sinápticos, combinador linear, limiar de ativação, potencial de ativação, função de ativação e sinal de saída, podemos descrever esses elementos básicos do neurônio artificial da seguinte forma:

- Os sinais de entradas $x = \{x_1, x_2, x_3, \dots, x_n\}$ que são sinais ou medidas vindas do meio externo e representam os valores assumidos pelas variáveis de entrada, são multiplicados pelos pesos sinápticos $w = \{w_1, w_2, w_3, \dots, w_n\}$ que são valores usados para ponderar as entradas, assim o limiar de ativação $\{\theta\}$ é a

variável que especifica qual será o patamar apropriado para que o resultado produzido pelo combinador linear (HAYKIN, 2011).

- O combinador linear $\{\Sigma\}$ que agraga todos os sinais de entrada que foram ponderados pelos respectivos pesos sinápticos a fim de produzir um potencial de ativação $\{u\}$ que é o resultado final do somatório do combinador linear.
- A função de ativação $\{g(u)\}$ tem como objetivo limitar a saída do neurônio dentro de um intervalo de valores razoáveis a serem assumida pela própria imagem funcional.
- O sinal de saída $\{y\}$ é o valor resultante final, podendo ser usado como entrada de outros neurônios que estão sequencialmente interligados.

Segundo Haykin (2001), o resultado final do neurônio proposto por McCulloch e Pitts (1943), em termos matemáticos pode ser resumido nas seguintes expressões:

$$u_k = \sum_{j=1}^m w_{kj}x_j \quad (1)$$

$$y_k = \varphi(u_k + b_k) \quad (2)$$

Observa-se que na equação (1) que as variáveis w_i são os pesos da conexão do neurônio e x representa as entradas do neurônio, estas variáveis são vetores. Assim, o resultado da variável u é um produto escalar vetorial entre x e w . A equação (2) y é a saída do neurônio, ressaltando que φ é a função de ativação, b_k são os pesos sinápticos dele. A função de saída deste neurônio é a *Heaviside*.

Baseando-se em Azevedo, Brasil e Oliveira (2000), uma forma de expressar matematicamente a função *Heaviside* é dada por:

$$yi = 1 \text{ se } net_i > 0 \quad (3)$$

$$yi = 0 \text{ se } net_i \leq 0 \quad (4)$$

Observando-se a equação (3), se o resultado for maior que 1 o neurônio é ativado, complementando com a equação (4) se o resultado for igual ou menor que 0 o neurônio em questão não é ativado.

2.3.3 Arquitetura da Rede Neural Artificial

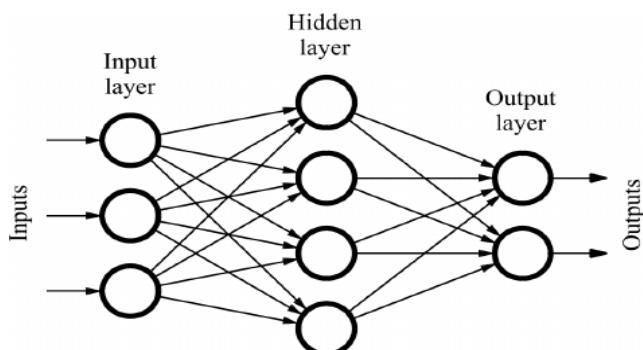
A maneira como as RNAs estão estruturadas influencia a maneira que o algoritmo aprende, ou seja, o número de camadas, número de neurônio em cada camada, tipo de conexão entre os neurônios e sua topologia.

Podemos classificar as RNAs quanto ao tipo de conexão em acíclicas ou redes de alimentação direta e redes cíclicas ou redes recorrentes.

2.3.3.1 Redes Acíclicas

As redes acíclicas (*feedforward*) são definidas por seus gráficos acíclicos dirigidos onde seus sinais só propagam em uma direção, na direção da camada de saída, ou seja, alimentada adiante.

Figura 13 - REDE NEURAL ACÍCLICA



Fonte: <https://msatechnosoft.in/blog/tech-blogs/artificial-neural-network-types-feed-forward-feedback-structure-perceptron-machine-learning-applications>
 (Acessado em 20/05/2019)

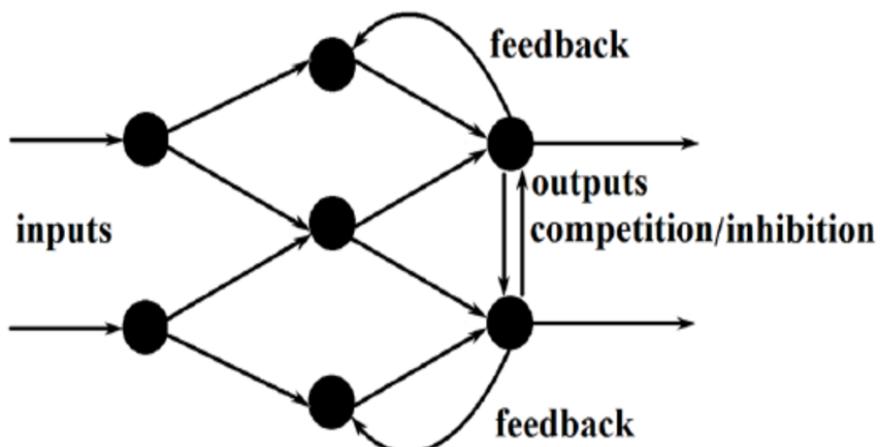
A Figura 13 representa uma estrutura de rede alimentada acíclica, observa-se que a rede contém três camadas, uma de entrada, uma oculta e a camada de saída. As conexões

entre essas camadas são todas propagando em um único sentido, para frente em direção ao sinal de saída.

2.3.3.2 Redes Cíclicas

As redes cíclicas (*feedback*) são definidas por conterem ciclos que podem representar um estado interno da rede, fazendo com que ela mude em função de um determinado tempo, baseando-se em suas entradas. O estado da rede continua mudando até atingir um ponto de equilíbrio que precisa ser encontrado (HAYKIN; 2001).

Figura 14 - REDE NEURAL CÍCLICA



Fonte: <https://msatechnosoft.in/blog/tech-blogs/artificial-neural-network-types-feed-forward-feedback-structure-perceptron-machine-learning-applications>

(Acessado em 20/05/2019)

Na Figura 14, uma representação de uma RNA do tipo cíclica, possuindo três camadas, uma de entrada, uma oculta e outra de saída. Na camada de saída observar-se que ela realimenta a camada oculta e também dois neurônios da camada de saída se realimentam.

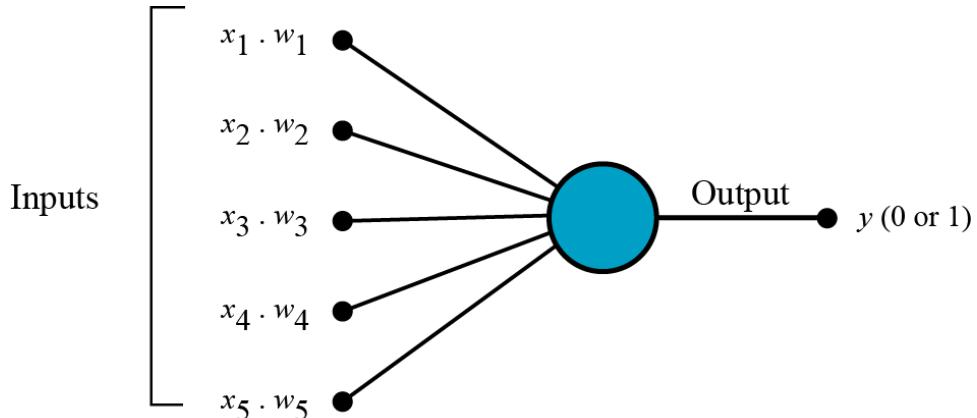
2.3.4 Perceptron

Segundo Haykin (2001), o *perceptron* constitui-se de um modelo de camada única, considerado um classificador binário que é ativado pela função de ativação, normalmente ativado por uma função de *Heaviside*, como já foi mencionado (Seção 2.3.2).

De acordo com Russel e Norving (2013), esse modelo é indicado para problemas linearmente separáveis, ou seja, ao se traçar um gráfico desses dados seja possível encontrar valores que distingue essas duas classes no plano cartesiano.

A inicialização do vetor pesos é realizada de forma randômica, normalmente muito utilizada uma distribuição normal, também chamada de Gaussiana, onde seus valores se encontram na variância de 0 e 1 (HAYKIN, 2001).

Figura 15 - MODELO MATEMÁTICO DO NEURÔNIO ARTIFICIAL



Fonte: O autor

Observa-se na Figura 15, um modelo de perceptron, onde suas entradas são multiplicadas pelos pesos que por sua vez são multiplicados pela função de ativação, gerando uma saída de classificação binária.

De acordo com Coppin (2013), o processo de aprendizado do perceptron é calculado pelos produtos da sua entrada pelos seus pesos, dando a entrada para a função de ativação, o algoritmo não irá terminar enquanto a entrada não for linearmente separável. Estimulando uma saída, se a classificação estiver correta os pesos não serão alterados, mas se os pesos estiverem incorretos serão atualizados, há um *loop* de ocorrência dessas atividades a fim de garantir o aprendizado, denominado de épocas, com o conceito de melhorar a classificação dos algoritmos em cada interação.

2.3.5 Multi Layer Perceptron

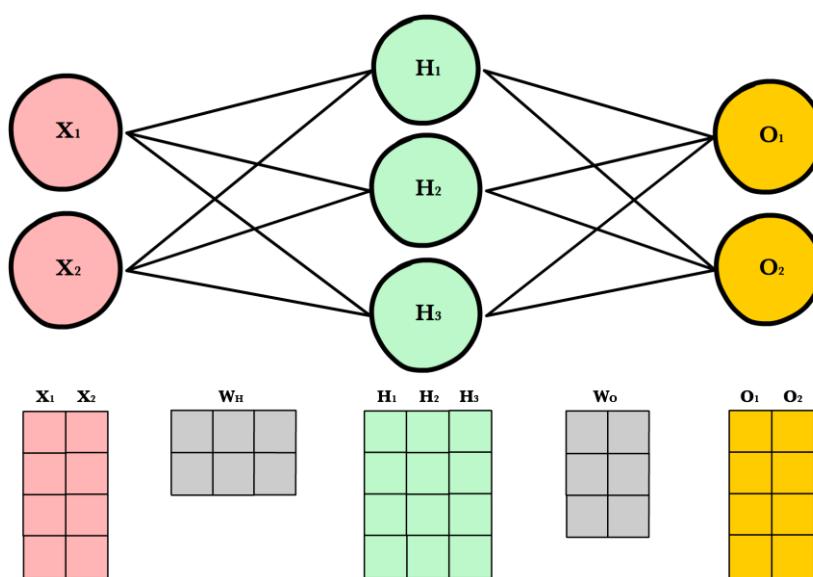
Segundo Azevedo, Brasil e Oliveira (2000), uma MLP é uma RNA construída por

composições de perceptrons, surgindo da ideia que os perceptron só conseguem resolver problemas linearmente separados, ou seja, as redes MLP derivam das redes perceptron, no entanto consegue lidar com problemas não lineares, questões mais complexas.

Uma RNA do tipo MLP é constituído por uma camada de entrada (*input layer*), uma ou mais camadas ocultas (*hidden layer*) e uma camada de saída (*output layer*).

- **Camada de Entrada:** responsável pelas características que sua rede irá aprender e assim fazer as devidas classificações, ou seja, são as entradas das redes, cada neurônio nessa camada representa um atributo único do seu conjunto de **dados e com isso a quantidade de neurônio nela representa o total de atributos**.
- **Camada Oculta:** é a camada que interliga as características (inputs) com o classificador (outputs), aplicando uma função de ativação, podendo variar pela quantidade de camadas, são totalmente conectadas, isso quer dizer que cada neurônio desta camada recebe todos os neurônios da camada anterior enviando sua saída para a camada seguinte.
- **Camada de Saída:** é a camada final da rede, recebendo a saída da camada anterior podendo aplicar o não uma função de ativação e gerando uma saída que é a classificação da rede.
- **Pesos:** são os pesos entre as camadas irão ajustar ao longo do treinamento, existindo um determinado peso para cada entrada.

Figura 16 - OS PESOS SINÁPTICOS ARTIFICIAIS



Fonte: <https://ml-cheatsheet.readthedocs.io/en/latest/forwardpropagation.html>

(Acessado em 29/05/2019)

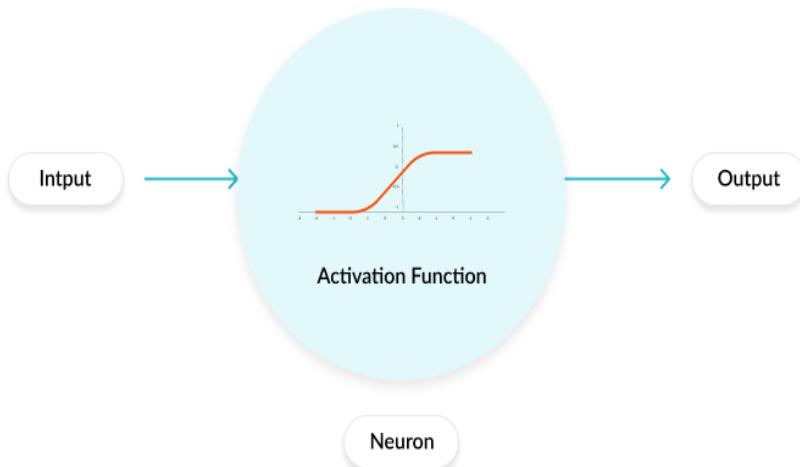
Observa-se na Figura 16 um exemplo de rede MLP com três camadas, a camada de entrada contendo dois neurônios, X_1 e X_2 , a oculta contendo 3 neurônios, H_1, H_2, H_3 , e por último a camada de saída contendo 2 neurônios O_1 e O_2 , todos os neurônios de todas as camadas estão conectados. Observa-se também os pesos entre as camadas de entrada e a oculta W_H e entre a oculta e a saída W_O .

2.3.6 Função de Ativação

Segundo Silva, Spatti e Flauzino (2010), a função de ativação é a transformação não linear que são feitas entre as variáveis de entradas (x) e as saídas (y). Vale ressaltar, que as RNA são consideradas aproximadores de funções universais, ou seja, podem calcular e aprender qualquer função.

As ativações são responsáveis por determinar se um determinado neurônio será ativado ou não, se neurônio em questão naquela ativação irá transferir os dados pertinentes para a próxima camada, ou a saída. As funções também ajudam a normalizar a saída de cada neurônio para um intervalo de 0 ou 1 ou -1 e 1, dependendo da função utilizada (RUSSEL, NORVIG, 2013).

Figura 17 - A FUNÇÃO DE ATIVAÇÃO



Fonte: <https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/> (Acessado em 30/05/2019)

Na Figura 17 tem-se os dados da entrada (*input*) que é passado para o neurônio, realizando todos os produtos entre entrada, pesos e depois é ativado pela função de ativação, assim produzindo uma saída.

- **Função Linear**

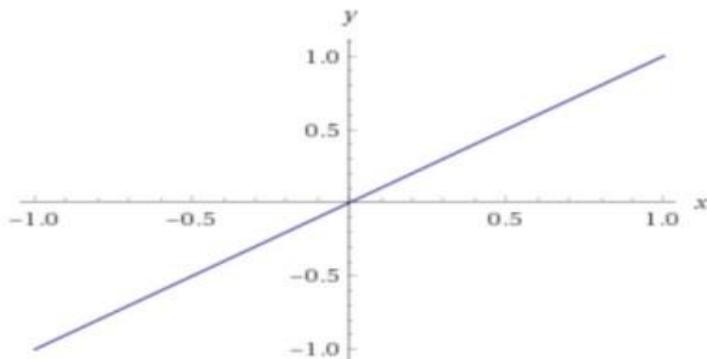
A função de ativação Linear leva as entradas multiplicadas pelos pesos para cada neurônio, e cria um sinal de saída proporcional a entrada.

A equação da função linear pode ser dada por:

$$f(x) = cx \quad (5)$$

Figura 18 - GRÁFICO DA FUNÇÃO LINEAR

Linear Activation Function



Fonte: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6> (Acessado em 30/05/2019)

- **Função Sigmoide**

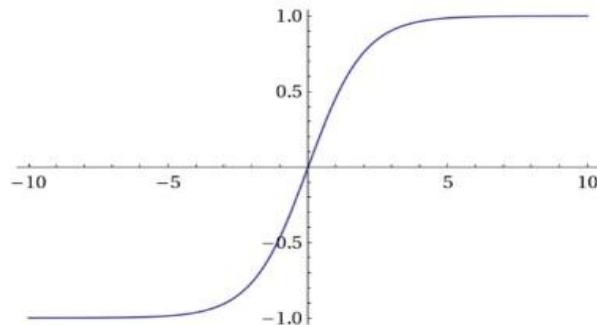
A função de ativação Sigmoide, também chamada de Logística, é diferenciável, ou seja, significa que se pode encontrar a inclinação da curva da função em dois pontos. Diferente da função linear, a saída sempre estará no intervalo (0, 1), pegando um valor qualquer de entrada e convertendo a esse intervalo. São muito utilizadas para classificação binária, onde as saídas são apenas 0 ou 1.

A equação da função sigmoide pode ser dada pela seguinte equação:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (6)$$

Figura 19 - GRÁFICO DA FUNÇÃO SIGMOIDE

Sigmoid Activation Function



Fonte: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6> (Acessado em 30/05/2019)

- **Tangente**

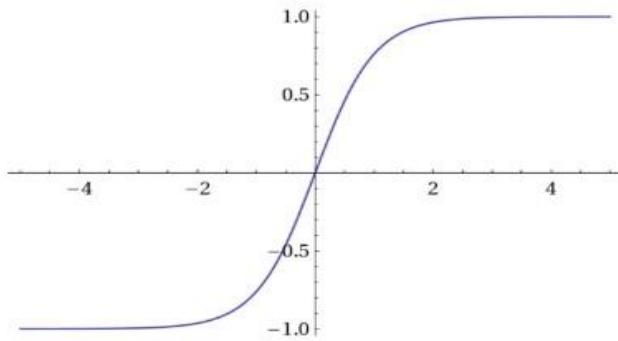
A função de ativação Tangente Hiperbólica (Figura 20) possui o mesmo formato da função sigmoide, porém atua nos intervalos de (-1, 1).

A equação da função tangente pode ser dada por:

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (7)$$

Figura 20 - GRÁFICO DA FUNÇÃO TANGENTE

Hyperbolic Tangent Activation Function



Fonte: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>
(Acessado em 30/05/2019)

- **Softmax**

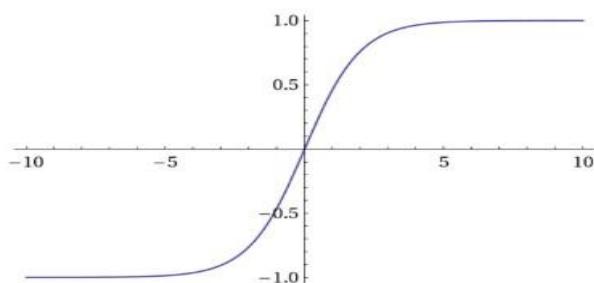
A função de ativação softmax (Figura 21) também é um tipo de função sigmoide, como mencionado anteriormente (seção 2.3.6) que a sigmoide consegue tratar bem problemas de classificação binárias, a softmax lida com problemas de multilassses, transformando as saídas para cada classe entre 0 e 1, ou seja, a probabilidade daquelas entradas estarem na determinada classe.

A equação da função softmax pode ser dada por:

$$f(x) = \frac{e^{x_i}}{\sum_{j \in group} e^{x_j}} \quad (8)$$

Figura 21 - GRÁFICO DA FUNÇÃO SOFTMAX

Softmax Activation Function



Fonte: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>
(Acessado em 30/05/2019)

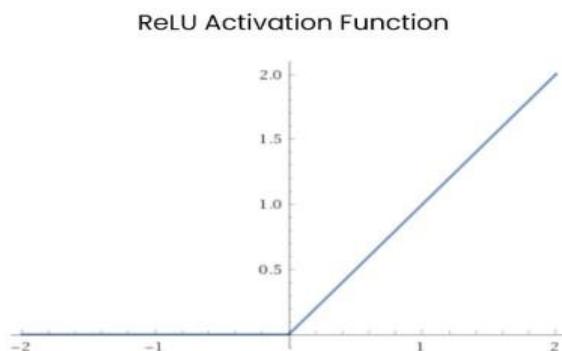
- **ReLU**

A função de ativação ReLU (Figura 22), Unidade Linear Retificada, ativa todos os neurônios ao mesmo tempo, se as entradas forem negativas serão convertidas em 0 e os neurônios não serão ativados, ou seja, apesar de ativar todas as entradas, os neurônios não negativos serão ativados pela função e os negativos desativados.

A equação da função ReLU pode ser dada por:

$$f(x) = \max(0, x) \quad (9)$$

Figura 22 – GRÁFICO DA FUNÇÃO RELU



Fonte: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6> (Acessado em 30/05/2019)

2.3.7 Função de Perda ou Custo

A função perda (*Loss Function*) tem como objetivo realizar a medição do erro entre o valor previsto do modelo e o valor real que era pra ser previsto pelo modelo. Existe uma variação de funções custo, mas este trabalho fará abordagem a *Mean Square Error* (MSE).

- **Mean Square Error**

O erro quadrático médio (MSE) mede a média dos quadrados dos erros, ou seja, a diferença quadrática média entre os valores estimados e o estimado. Pode ser definida pela

seguinte equação:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (10)$$

2.3.8 Algoritmo de Backpropagation

Segundo Russell e Norving (2013), o algoritmo de *Backpropagation* ou retro propagação de erro é o método de ajustes dos pesos da rede com base na taxa de erro na época (iteração) anterior, permitindo que o modelo aumente a generalização. Ou seja, ajustando os pesos com base no erro geral, referenciando o quanto cada peso de cada camada contribui para erro.

Vale ressaltar que esse algoritmo utiliza-se da descida de gradiente para minimizar o erro quadrático entre a saída dos erros de cada iteração no treinamento e o valor do classificado que era esperado pela saída.

Este algoritmo consiste em dois passos de computação: o processamento direto (*forward*) e o processamento reverso (*backward*).

O *Forward* (processamento direto) consiste em os dados de entrada a serem propagados para frente, passando por cada camada e realizando os devidos cálculos para no final gerar uma previsão do modelo. Neste processamento os pesos sinápticos mantêm-se inalterados, ou seja, a rede é calculada diretamente por etapa sem alteração de suas variáveis (HAYKIN, 2001).

O *Backward* (processamento reverso) consiste em ajuste dos pesos da rede proporcionalmente ao quanto ele contribuiu para erro geral. Ou seja, enquanto no *forward* a propagação é feita para frente, calculando a transferência do neurônio em cada camada até chegar à saída, no *backward* ocorre à retro propagação, que com base no erro geral, calcula o erro que cada camada contribui para ele, assim atualizando os pesos dessas camadas, reajustando os parâmetros para uma nova iteração (RUSSELL; NORVIG, 2013).

3 Estudo de Caso

No presente capítulo serão descritos a situação problema, as ferramentas, métodos e materiais utilizados no trabalho, ou seja, a criação e validação de uma RNA, utilizando-se o algoritmo *Backpropagation*.

3.1 Petróleo

Fazendo uma análise etimológica, o petróleo vem de duas palavras do latim petra (pedra) e oleum (óleo), ou seja, “o óleo que sai da pedra”. “É um composto que quando em estado líquido, é uma substância oleosa, inflamável, menos densa que a água, que possui cor variando do negro ao castanho claro” (THOMAS, 2001). O petróleo é constituído basicamente, por uma mistura de compostos químicos orgânicos (hidrocarbonetos) e alguns contaminantes conforme tabela 1.

Tabela 1 – ANÁLISE ELEMENTAR DO ÓLEO CRU (% EM PESO)

Compostos Químicos	Percentual
Carbono	83 – 87%
Enxofre	0,06 – 8%
Hidrogênio	11 – 14%
Metais	Até 0,3%
Oxigênio	0,1 – 2%
Nitrogênio	0,11 – 1,7%

Fonte: Adaptado de Thomas (2001)

De acordo com Barros et al (2009), as matérias orgânicas como algas planctônicas e *fitoplâncton* depositadas em sedimentos juntamente com a interação da temperatura, pressão ao longo do tempo geram processos geoquímicos com o produto final hidrocarbonetos.

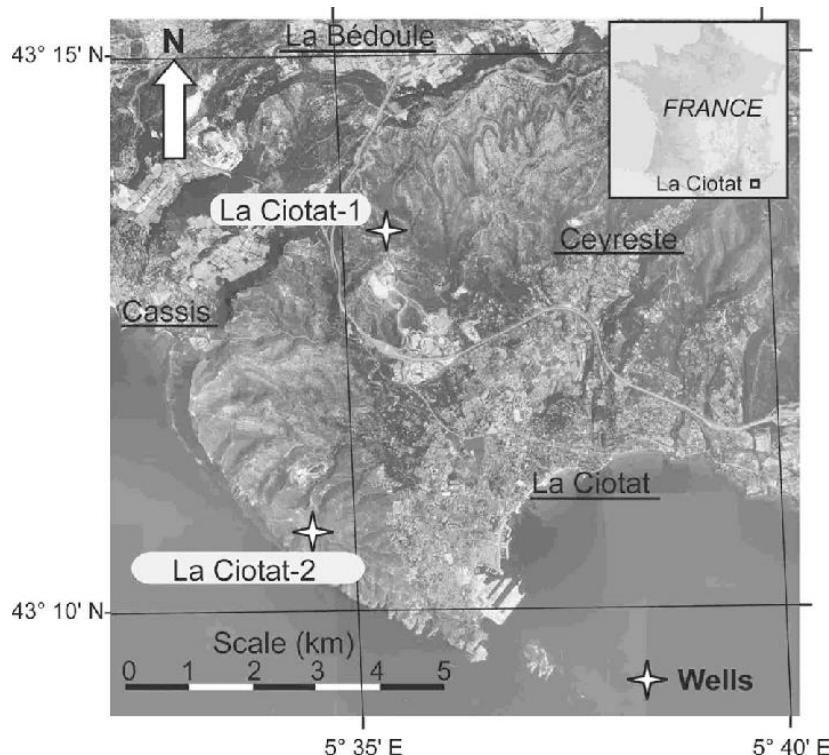
O tipo de hidrocarboneto gerado (óleo ou gás) depende da matéria orgânica original e pela intensidade do processo térmico que nela for empregado. *Fitoplânctons* marinhos originam hidrocarbonetos líquidos, já matéria orgânica lenhosa tem a propensão de originar hidrocarbonetos gasosos.

O petróleo é formado por uma rocha geradora que migra para a rocha reservatório, que deve ser permeável o suficiente para acomodar o óleo em seu interior. Tendo o sistema condições de gerar e migrar o óleo, faz-se necessário a existência de uma rocha selante (folhelhos e evaporitos) normalmente de baixíssima permeabilidade, para aprisionar o óleo evitando que o mesmo exsude para a superfície.

3.2 Base de Dados

Para desenvolvimento do projeto a base de dados utilizada neste trabalho foi retirada da literatura Fournier & Borgomano (2009) e Silva et al. (2015). São dois poços de petróleo La Ciota-1 e La Ciotat-2, localizadas na bacia do sul da Provença na França, a sua formação geológica é datada durante o Cretáceo Superior. A Figura 23 resume a localização dos poços mencionados.

Figura 23 - LOCALIZAÇÃO DOS POÇOS LA CIOTAT-1 E LA CIOTAT -2

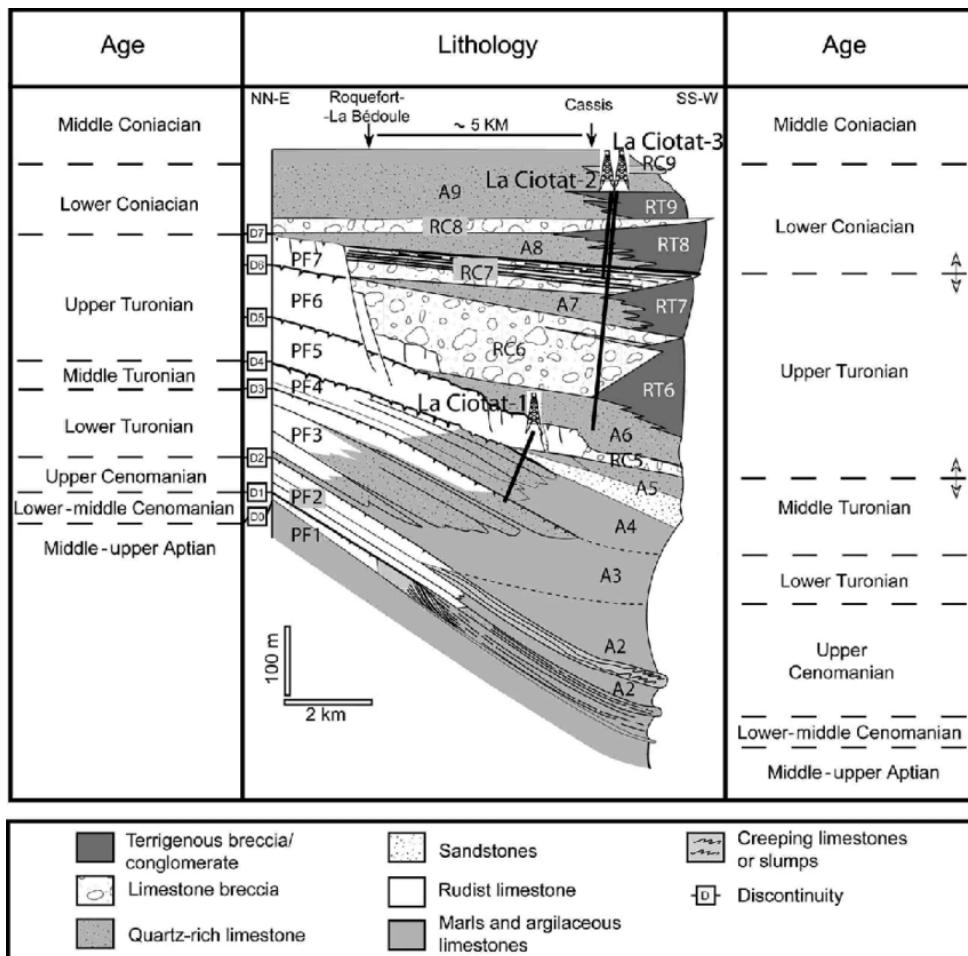


Fonte: Fournier & Borgomano (2009).

O conjunto de dados contém 53 instâncias de rochas coletadas de dois poços de

petróleo, como mencionado, contendo características de dados geológicos referentes a essas rochas, subdivididos em 22 características, com 7 classificações de litologia como mostrado na carta estratigráfica da Figura 24 e na Tabela 2.

Figura 24 – CARTA ESTRATIGRAFICA



Fonte: Fournier & Borgomano (2009).

Observa-se na Figura 24 a carta estratigráfica das principais litologias encontradas baseando-se nos estudos e análise dos poços por Fournier & Borgomano (2009).

Tabela 2 - LITOLOGIA DE ROCHAS

Litologia	Nome
1	<i>Limestone</i> com textura <i>grainstone</i> (quartzo < 5%)
2	<i>Limestone</i> com textura <i>wackestone-packstone</i> (quartzo < 5%)
3	<i>Limestone</i> rico em quartzo com textura <i>grainstone</i> (quartzo 5 – 50%)
4	<i>Limestone</i> rico em quartzo com textura <i>wackestone-packstone</i> (quartzo 5 – 50%)
5	<i>Limestone</i> rico em quartzo com leve ocorrência de argilas (quartzo 5 – 50% e argilas 2 – 5%)
6	<i>Sandstone</i> limpo (quartzo 50%)
7	<i>Sandstone</i> com matriz micrítica carbonática (quartzo > 50 %)

Fonte: Adaptado de Silva et al. (2015)

Segundo Fournier & Borgomano (2009), os *datasets* registrados das análises dos poços inclui dados de raios gama, ultrassônicos de onda P e S, densidade e resistividade. Ressaltando que a densidade e a porosidade foram determinadas por métodos de saturação de salmora e a análise quantitativa da mineralogia usando Difração de Raios X e a abordagem de Rietveld. Esses dados possuem propriedades elásticas, mineralógicas e petrográficas que são correlacionadas à litologia.

3.3 Metodologia e Proposição de Rede Neural Aartificial

Para concepção do estudo de caso, predizer litologias de rochas carbonáticas utilizando-se RNA, aplicou-se a metodologia do *KDD*, cujos conceitos foram introduzidos no Capítulo 2, como também o conjunto de dados oriundo de Fournier & Borgomano (2009).

A RNA criada é denominada MLP, sua conexão é dada por todos os neurônios estarem totalmente conectados entre todas as camadas, denominada rede acíclica, reforçando a abordagem de arquitetura de RNA visto no Capítulo 2, mencionada a estrutura de suas camadas e também o tipo das conexões realizadas entre elas. Como também, adotou-se o algoritmo de *backpropagation* junto com as funções de ativações.

3.3.1 O processo de Aprendizagem

O aprendizado da RNA está vinculado ao processo de treinamento que se inicia após o pré-processamento dos dados, levando em consideração que nesta etapa os conjuntos de dados de treino e teste foram separados. Ou seja, nenhum dos conjuntos de dados sabe da existência do outro desde o pré-processamento, isso é fator fundamental para etapas posteriores.

Posteriormente, na etapa de processamento dos dados, realiza a criação da rede, onde a camada de entrada é o número de características do conjunto de dados para a classificação, a camada oculta é estipulada empiricamente e a camada de saída é o classificador ou os classificadores do *dataset*.

Deste modo, é determinado um número de épocas que o algoritmo irá treinar que determina a quantidade de iteração que o modelo terá, como também o lote (*batch size*) que define a quantidade de amostra a trabalhar antes de atualizar os parâmetros internos. Lembrando que uma amostra significa uma linha no *dataset* em questão.

Utilizando o algoritmo de *backpropagation* que se divide em duas etapas o processamento de informações pela rede. Ou seja, na primeira etapa os sinais de entrada são passados pelos neurônios, camada a camada, realizando produto vetorial entre os dados de entrada, os pesos e a função de ativação pertencente a camada em questão fluindo com esses dados para a saída, normalmente na camada de saída são usadas as funções sigmóide ou softmax, esse é o processo de *forward*.

Vale ressaltar que depois de terminado o processo *forward*, o modelo gera uma saída de classificação que ele determina que seja correta, contudo, como se trata de um algoritmo supervisionado, há um classificador esperado pelo modelo. Caso esse desempenho não seja suficiente, entra a segunda etapa do *backpropagation*, a fase *backward* que visa minimizar o erro da estimativa, comparando os valores de saída esperado com o de saída produzido pelo modelo, utilizando a equação (8), verificando quanto cada camada contribuiu para o erro geral. Atualizando os pesos de cada camada.

Esse processo do algoritmo *Backpropagation* e a atualização dos pesos são decorridos enquanto a iteração estiver no processo de treinamento definido pelo número de épocas, produzindo um desempenho não significativo, sendo necessário o ajuste para atualização dos parâmetros e verificação de sua saída, reduzindo a função custo e aumentando a acurácia (tratado na Seção 2.3).

3.3.2 Ferramentas utilizadas

As ferramentas utilizadas neste trabalho para manipulação dos dados e construção do estudo de caso são abordadas resumidamente a seguir. O Apêndice A pode ser consultado para maiores informações.

- **Anaconda:** é uma distribuição de código aberto (*open-source*) disponível para as linguagens de programação Python e R, usada para ciência de dados e aprendizado de máquina, com uma variabilidade de bibliotecas, algumas IDE e ferramentas específicas, facilitando o desenvolvimento dos projetos correlacionados.
- **Python:** foi criada em 1991, orientada a objeto, sendo multiplataforma como: desenvolvimento web, automatização, Internet das Coisas do inglês *Internet of Things* (IoT), Big Data e áreas correlacionadas. A linguagem faz uso de baixa quantidade de palavras reservadas no código, identação obrigatória, autogerenciamento de memória, entre demais funcionalidades.
- **Jupyter Notebook:** é um projeto sem fins lucrativos de código aberto criado em 2014 fruto da extensão do projeto *IPython* e cujo objeto é criar um suporte em diversas linguagens de programação para o desenvolvimento iterativo da ciência dos dados e da computação científica, permitindo a confecção do chamado *notebook*, documentos que podem conter código, imagem, gráficos, texto e afins.
- **Numpy:** é uma biblioteca do Python que suporta *arrays* e matrizes multidimensionais, possibilitando variedade de funções, como multiplicações de matrizes ou da transposta, facilitando o trabalho com essas estruturas.
- **Pandas:** é uma biblioteca do Python para manipulação e análise de dados, oferecendo funcionalidade para trabalho com estruturas de tabela, permitindo rápidas visualizações, indexações, inclusões e exclusões, como também a visualização das medidas estáticas desses dados de maneira simplificada.
- **Scikit-Learn:** é uma biblioteca do Python para aplicação dos algoritmos de mineração de dados e aprendizado de máquina, integrando diversos algoritmos famosos de maneira eficiente e simples de aplicar.

- **Matplotlib:** é uma biblioteca de plotagem de gráficos do Python, fornecendo uma API orientada a dados e incorporadas em suas plotagens, permitindo a criação de uma variabilidade de gráficos.
- **Tensorflow:** é uma biblioteca de código aberto para aprendizado de máquina, aplicado a uma variabilidade de problemas, constitui-se de um sistema de criação desenvolvido pela *Google*, utilizando para implementação de redes neurais artificiais.
- **Keras:** é uma biblioteca voltada para implementação de redes neurais artificiais e redes neurais profundas (*Deep Learning*).

3.4 Desenvolvimento da RNA

Para auxiliar no desenvolvimento do projeto, foi utilizado a ferramenta de controle de versão GitHub que é uma plataforma de serviço web que fornece diversas funcionalidades para o projeto vinculado ao Git que por sua vez é um sistema de controle de versão de arquivos. O repositório do projeto está disponível pelo seguinte link: <[>](#).

Após definir a base é necessário começar o processo visto no capítulo 2 na metodologia do *KDD* chamado de pré-processamento, sendo assim, os dados são carregados para *Python*, transformando-o em um *data frames* do *Pandas*, permitindo assim a utilização de seus métodos e funcionalidades proveniente da biblioteca mencionada na seção anterior.

O Anexo B contém as informações do conjunto de dados referente ao trabalho, e como mencionado na Seção 3.1, são dados correspondentes das análises de Fournier & Borgomano (2009) em dois poços de petróleo situados na França. Esse *dataset* possui 53 amostras (linhas), 22 características (colunas), sendo uma dessas referentes ao classificador, litologia, com variância de 1 a 7. A Tabela 3 mostra a quantidade de amostras por litologia.

Tabela 3 - QUANTIDADE DE AMOSTRAS POR LITOLOGIA

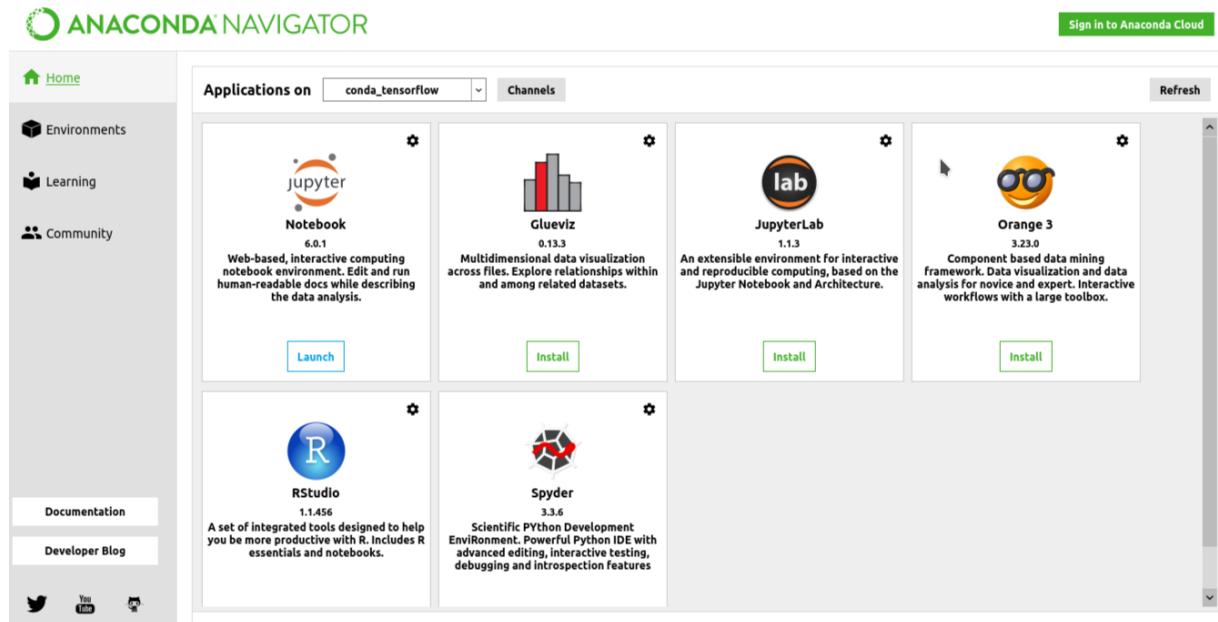
Amostras	Litologia
7	1
10	2
8	3
11	4
2	5
5	6
10	7

Fonte: O autor

O Anexo B1 contém o conjunto de dados de treinamento e o B2 o conjunto de dados de testes, ressaltando que esses dados não possuem conhecimento uns dos outros após essa divisão.

Utilizou-se a distribuição Anaconda que integrou o Jupyter-Notebook de forma nativa, tratado na Seção 3.2.3, tanto nos processos de pré-processamento como nos de processamento dos dados. O Apêndice B pode ser consultado para maiores detalhes dos códigos criados.

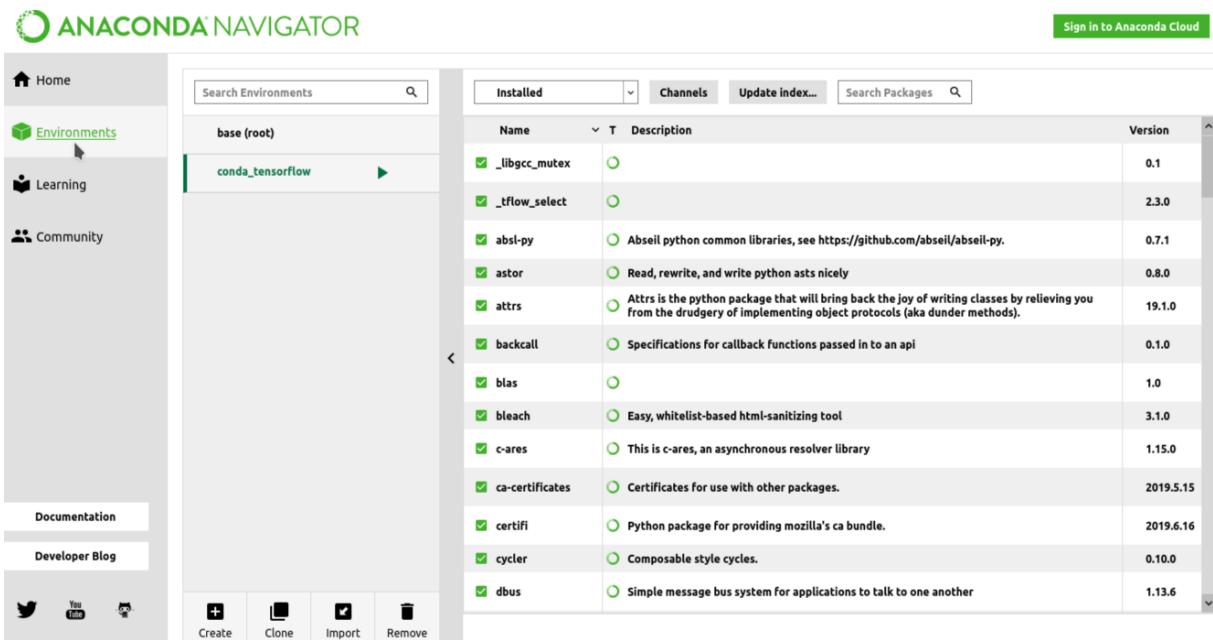
Figura 25 - DISTRIBUIÇÃO ANACONDA NAVIGATOR



Fonte: O autor

Observa-se na Figura 25 a tela inicial da distribuição Anaconda que possui diversas outras ferramentas integradas.

Figura 26 - ANACONDA AMBIENTES VIRTUAIS E BIBLIOTECAS



Fonte: O autor

A Figura 26 mostra o *setup* para criação de ambientes de desenvolvimentos virtuais do Anaconda para Python, como por exemplo: o *conda_tensorflow* que possui as bibliotecas utilizadas durante o desenvolvimento do projeto. Ao lado direito, podemos realizar o gerenciamento das bibliotecas desse ambiente.

Figura 27 - IMPORTAÇÕES DE BIBLIOTECAS DO PYTHON

The screenshot shows a Jupyter Notebook interface with the title "Importações de Bibliotecas". In the code cell (In [1]), the following Python code is displayed:

```

In [1]: import IPython
         from IPython import display
         import matplotlib
         %matplotlib inline
         import matplotlib.pyplot as plt
         import numpy as np
         import os
         import os.path
         import pandas as pd
         import random
         from random import seed
         import seaborn as sns
         import sklearn
         from sklearn.base import BaseEstimator
         from sklearn.preprocessing import StandardScaler, MinMaxScaler
         from subprocess import check_output
         import tensorflow as tf
         from tensorflow import keras
         import warnings
         warnings.filterwarnings('ignore')

         print("IPython version: {}".format(IPython.__version__))
         print("matplotlib version: {}".format(matplotlib.__version__))
         print("NumPy version: {}".format(np.__version__))
         print("pandas version: {}".format(pd.__version__))
         print("Keras version: {}".format(keras.__version__))
         print("seaborn version: {}".format(sns.__version__))
         print("Sklearn version: {}".format(sklearn.__version__))
         print("Tensorflow version: {}".format(tf.__version__))

IPython version: 7.9.0
matplotlib version: 3.1.1
NumPy version: 1.17.4
pandas version: 0.25.3

```

Fonte: O autor

Observam-se na Figura 27 as importações das bibliotecas utilizadas para desenvolvimento do projeto no Jupyter Notebook.

Figura 28 - FUNÇÃO CARREGAR DADOS

```

def load_data():
    """
    Função para carregar os datasets pelo pandas.
    Retun: a variável passa a ser um dataframe do pandas, podendo ter acesso aos métodos da lib.
    dftrain = treinamento
    dftest = teste
    """
    dftrain = pd.read_csv('/home/willian/Keras_Litologia/arquivos_csv/train.csv')
    dftest = pd.read_csv('/home/willian/Keras_Litologia/arquivos_csv/test.csv')
    return dftrain, dftest

```

Fonte: O autor

Observa-se na Figura 28 a função *load_data*, que carrega os arquivos CSV em um DF da biblioteca Pandas, permitindo a manipulação desses dados com o conjunto de métodos que a biblioteca possui. O *pd* é a chamada da simplificação da biblioteca Pandas, nesse caso o *pd.read_csv* é o método que permitiu a leitura e gravação desses dados na variável que o recebe, sendo o retorno dessa função os *dataset* de treino e teste como variável da biblioteca

Pandas, ou seja, variáveis que podem acessar todos os métodos de manipulações de *dataset* disponível na biblioteca.

Figura 29 - FUNÇÃO DE TRANSFORMAÇÃO DO CLASSIFICADOR

```
def data_encode(data_frame, classificador):
    """
    Função para transformar a coluna do classificado em multiclassificadores *litologia de 1 a 7.
    data_frame: dicionário a ser recebido.
    classificador: nome da coluna do classificador.
    Return: Um novo data frame com as linhas de classificação transformada em colunas.
    """
    data_frame = pd.get_dummies(data_frame, columns=[classificador])
    return data_frame
```

Fonte: O autor

Na Figura 29 tem-se a função *data_enconde*, que recebe como parâmetros o *data_frame* e o nome da coluna que serão transformados nas codificações necessárias. Neste caso, a transformação é do classificador em colunas com as saídas 0 ou 1, ausência ou presença desse atributo. Ao chamar essa função passando o *dataset* em questão e o classificador 'Litologia', a variável *data_frame* receberá um novo DF, com as transformações dos dados de litologia gerando 7 colunas de classificadores binárias (Litologia1 a Litologia7).

Figura 30 - FUNÇÃO DE TRANSFORMAÇÕES DAS VARIÁVEIS X E Y

```
def variables_x_y(data_frame, cabecalho, n):
    """
    Função para transforma as variáveis de características e classificadores.
    n = as colunas que fazem parte do classificador.
    y = classificador.
    x = características.
    return: um data_frame de características e outro de classificador do dataset.
    """
    x = data_frame[cabecalho[:n]]
    y = data_frame[cabecalho[n:]]
    return x, y
```

Fonte: O autor

A Figura 30 refere-se à função *variables_x_y* que recebe como parâmetro o *dataset* a qual deseja ser transformado, a variável *cabecalho* que é uma *array list* com os nomes de cabeçalho do *dataset* (primeira linha do conjunto de dados) a ser transformado e o *n* que é o parâmetro o qual representa as colunas de classificação do *dataset*. Essa função define as variáveis e o classificador em questão, *x* são as variáveis de entrada referente ao *dataset*, as quais são os atributos pertinentes para a classificação da litologia, e *y* é a variável classificadora, ou seja, a saída esperada de cada linha desses atributos combinados. Apesar de *data_frame* ser uma variável DF do Pandas, ainda é uma estrutura de dados compostas do *Python*, possibilitando a utilização das funções básicas das demais estruturas de dados. Neste

caso, ao usar o [:] está copiando os dados de uma variável *data_frame* para outra, o primeiro espaço vazio dos dois pontos refere-se ao início da lista a qual são desejados os dados, o outro ao fim, e passando *n* como parâmetro diz-se que x irá receber um *data frame* do início deste até a coluna *n*, e na coluna anterior do classificador, o mesmo procedimento acontece com a variável *y*.

Figura 31 - FUNÇÃO DE TRANSFORMAÇÃO PARA ARRAYS NUMPY

```
def transfer_array_np(data_frame, vtype):
    """
    Função para transformar o data_frame em array numpy [ matriz numpy]
    df = data_frame a ser transformado
    vtype = nome da variável do tipo a ser transformada
    Return: matriz numpy do dataset.
    """
    np_data_frame = np.array(data_frame, dtype=vtype)
    return np_data_frame
```

Fonte: O autor

Na Figura 31 tem-se a função *transf_array_np* que tem como parâmetros o *data_frame* e o *vtype* que é nome do tipo de *array numpy* a qual se deseja a transformação. A variável *np_data_frame* é o resultado da transformação do *dataset*, resultando em um *array* da biblioteca *numpy*. Essa função apenas irá converter a estrutura do *dataset* em uma matriz *numpy*, possibilitando assim acessar os seus métodos.

Figura 32 - FUNÇÃO PRÉ-PROCESSAMENTO

```
def preprocessamento():
    dftrain, dftest = load_data()
    train = data_encode(dftrain, 'Litologia')
    test = data_encode(dftest, 'Litologia')
    train = train.drop(['Poço'], axis = 1)
    test = test.drop(['Poço'], axis = 1)
    cabecalho_list = list(train.columns.values)

    # n = -7 representa primeira coluna do classificado (decrescente).
    n = -7

    x_train, y_train = variables_x_y(train, cabecalho_list, n)
    x_test, y_test = variables_x_y(test, cabecalho_list, n)
    X_train = transfer_array_np(x_train, 'float32')
    y_train = transfer_array_np(y_train, 'int8')
    X_test = transfer_array_np(x_test, 'float32')
    y_test = transfer_array_np(y_test, 'int8')
    scaler = MinMaxScaler()
    x_train = scaler.fit_transform(X_train)
    x_test = scaler.fit_transform(X_test)
    return x_train, y_train, x_test, y_test, dftrain, dftest
```

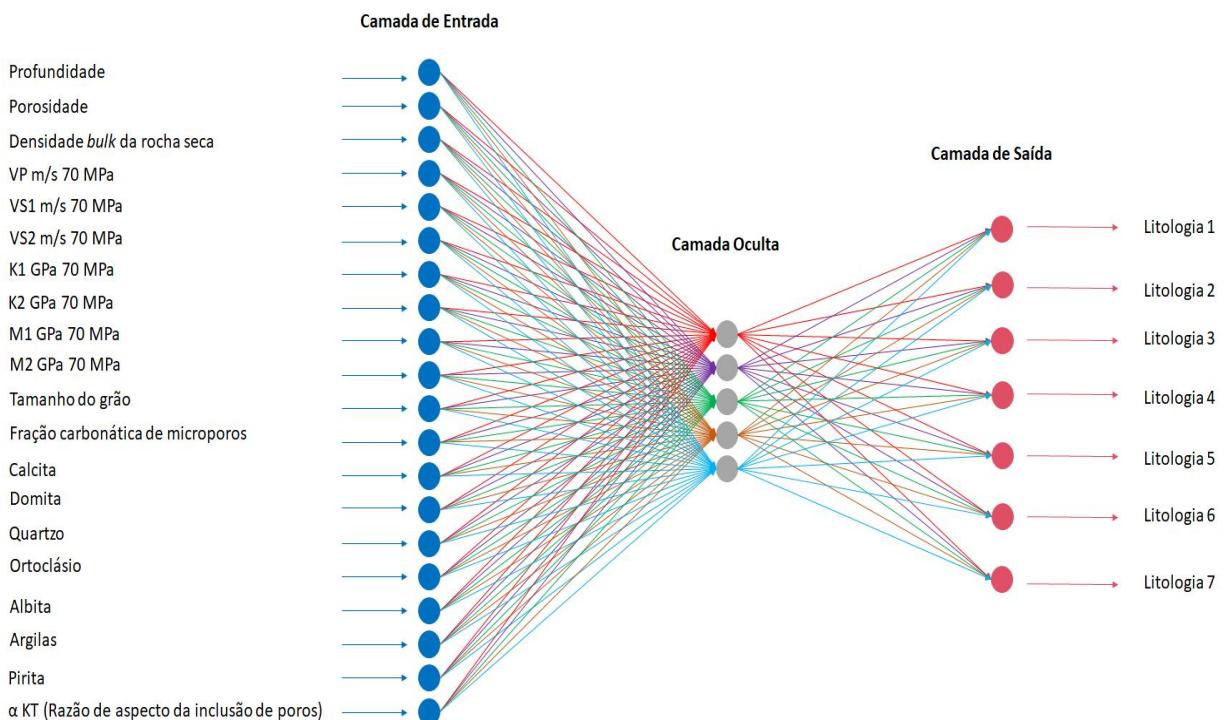
Fonte: O autor

Observa-se na Figura 32 que a função *preprocessamento* é responsável por chamar todas as outras funções mencionadas anteriormente (Figuras 28 a 31), ressaltando a exclusão do atributo poço dos conjuntos de dados, sendo irrelevante para o aprendizado do algoritmo.

Realizou-se as transformações desses dados em escalas favoráveis, utilizando o método matemático de Mínimo-Máximo, cujo sua equação é dada por:

$$\frac{X_i - \min(x)}{\max(x) - \min(x)} \quad (11)$$

Figura 33 - RNA COM OS DADOS PRÉ-PROCESSADOS



Fonte: O autor

A Equação 11 reduz o intervalo dos números da amostragem entre 0 ou 1, e caso haja números negativos, então o intervalo fica entre -1 a 1, buscando o mínimo valor daquela amostra de dados e o máximo valor, realizando subtração e divisão entre eles. E, como resultado de todas essas etapas, obteve as seguintes características visto na Figura 33 como *input* e *output* da RNA e complementado os *outputs* a Tabela 3.

A Figura 33 representa uma RNA criada com o conjunto de dados do Fournier & Borgomano (2009) após as etapas de pré-processamento. Os *input* são as 20 características oriundas do conjunto de dados contido no Anexo A, fazendo parte da camada de entrada (círculo azul), a camada oculta (círculo cinza), contém 5 neurônios apenas para ilustração da

rede, a camada de saída (círculo rosa), contém uma das características do conjunto de dados que virou o classificador, são os *outputs* do conjunto de dados após passar pelas etapas de transformações, tornando-se litologias com variância de 1 a 7 e com dados de presença ou ausência de características (binários) conforme exibido na Tabela 4.

Tabela 4 - VALORES BINÁRIOS DAS LITOLOGIAS

Litologia	Binário
1	1000000
2	0100000
3	0010000
4	0001000
5	0000100
6	0000010
7	0000001

Fonte: O autor

4 Testes e Resultados

Foram realizados testes para validação da base de dados proposta para o estudo de caso do trabalho, subdividindo-os em três tipos chaves com os demais parâmetros fixos.

Vale ressaltar que as estruturas das RNAs são definidas de maneira hipotética, ou seja, tentativas e erros. Sendo assim, ficou definido 20 neurônios na camada de entrada, a escolha dessa quantidade é referente à quantidade de características que o conjunto de dados possui, como exibido no Capítulo 3 (Figura 33), a quantidade de neurônios da camada de saída ficou definidos como 7 que são as possibilidades de classificação do problema, ou seja, litologia 1 até a litologia 7. A quantidade de épocas que o modelo treinou, foi definido como 2.500, escolhido hipoteticamente e julgando que nesse intervalo o modelo conseguiria aprender e se manter estabilizado. O tamanho do lote foi definido como 30 julgando que esse valor seria razoável para as mudanças dos parâmetros internos e atualizações dos parâmetros da rede.

A configuração da máquina utilizada para os testes tratou-se de um processador Intel Core i5 2.20 GHz e 8 GB de memória RAM e Sistema Operacional Linux com a Distribuição Manjaro 18.1.3 com Interface KDE Plasma.

Tabela 5 – PARÂMETROS DE CONFIGURAÇÕES INICIAIS

N. Entrada	Função Entrada	Função Oculta	N. Saída	Função Saída	Época	Lote
20	Relu	Relu	7	Softmax	2500	25

Fonte: O autor

A Tabela 5 mostra os parâmetros fixados escolhido de maneira hipotética para realizações do primeiro grupo de testes de criação da RNA.

Os primeiros testes foram relacionados à quantidade de neurônios da camada oculta (*Hidden-Layer*), ressaltando que os valores de variação mencionados para teste foram escolhidos hipoteticamente e os demais parâmetros fixos como mencionados na Tabela 5.

Tabela 6 – QUANTIDADE DE NEURÔNIOS DA CAMADA OCULTA

Teste	Neurônios	Treino Acurácia	Teste Acurácia	Época
1.1	5	54,34 %	53,10 %	2483
1.2	10	100 %	83,15 %	1911
1.3	15	100 %	83,15 %	944
1.4	20	97,83 %	82,54 %	2313
1.5	27	71,74 %	71,43 %	2424
1.6	30	71,74 %	57,14 %	2061

Fonte: O autor

Observando-se os detalhes e resultados dos testes da Tabela 6, os modelos de Teste 1.2 e 1.3 obtiveram um resultado excelente utilizando-se como métrica apenas a acurácia (de treino), sendo o modelo que apresentou o melhor desempenho aquele que possui 15 neurônios ocultos (teste 1.3), julgando que demorou menos épocas para o aprendizado.

O teste 1.1 (Tabela 6) demorou 2483 épocas quase atingindo o limite definido (2500) e mostrou ser o pior resultado com acurácia de 54,34 %, realizando um comparativo com os demais casos, verifica-se que uma rede com poucos neurônios na camada oculta interfere na taxa da avaliação de seu aprendizado, como também se submete a análise que modelos com grandes quantidades de neurônios também tem essas taxas, ou seja, os testes 1.5 e 1.6 também são afetados.

E, analisando a partir desse detalhe a escolha do melhor modelo foi o 1.3, com 15 neurônios na camada oculta (teste 1.3), definido como parâmetro do modelo. Após essa definição, fixou esse parâmetro na rede e realizou outros testes alterando a função de ativação da camada oculta com os demais parâmetros fixos, podemos observar na tabela a seguir:

Tabela 7 - FUNÇÕES DE ATIVAÇÕES DA CAMADA OCULTA

Teste	Função de Ativação	Treino Acurácia	Teste Acurácia	Época
1.3	Relu	100 %	83,15 %	944
2.1	Softmax	54,34 %	42,86 %	2483
2.2	Tangente	54,34 %	42,86 %	2473
2.3	Sigmoid	63,89 %	42,86 %	2375

Fonte: O autor

Observando-se os detalhes dos testes mencionados na Tabela 7, a função de ativação que obteve o melhor resultado na camada oculta foi a ReLU (teste 1.3), ou seja, o fato desta

função ativar apenas neurônios positivos, fazendo com que os neurônios negativos não sejam ativados mostrou ser uma características que trouxe bons resultados. Ressaltando também que as outras três funções são mais indicadas para camada de saída.

Sendo assim, a função Relu foi definida como a função de saída da camada oculta da RNA do modelo. Após isso, realizou-se o terceiro teste, alterando a função de ativação da camada de saída e os demais parâmetros fixos, cujos resultados são apresentados na Tabela 8.

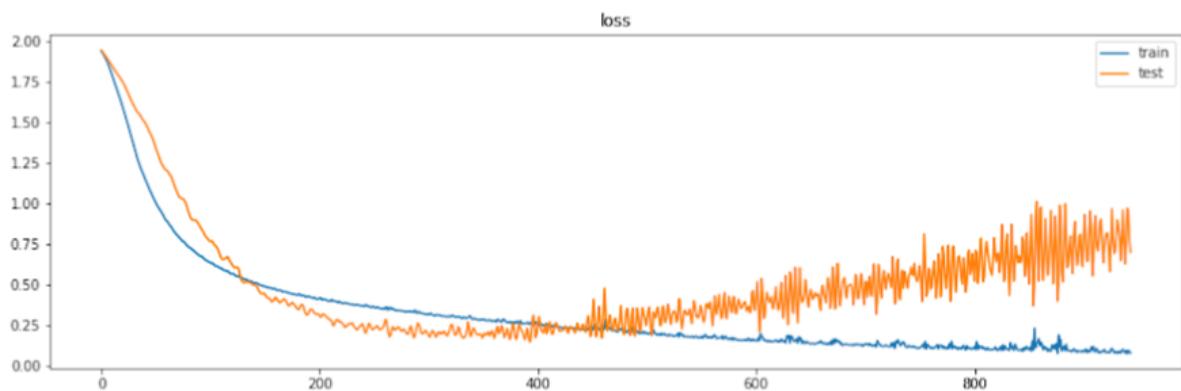
Tabela 8 - FUNÇÕES DE ATIVAÇÃO DA CAMADA DE SAÍDA

Teste	Função de Ativação	Treino	Teste Acurácia	Época
1.3	Softmax	100 %	83,15 %	944
3.1	Sigmoide	100 %	82.54 %	1285
3.2	Tangente	100 %	82.54 %	1312

Fonte: O autor

A Tabela 8 mostra os resultados obtidos após os últimos testes com alterações das funções de ativações da camada de saída, ressaltando que as três funções são derivações da função Sigmoide. A função Softmax, obteve o melhor desempenho, ou seja, lidando bem com problemas de multiclassos de previsões, essa função consegue entender e submeter boas previsões realizando a probabilidade de cada ocorrência de classificação, correlacionado as 7 classes possíveis de litologias e conseguindo diferenciá-las.

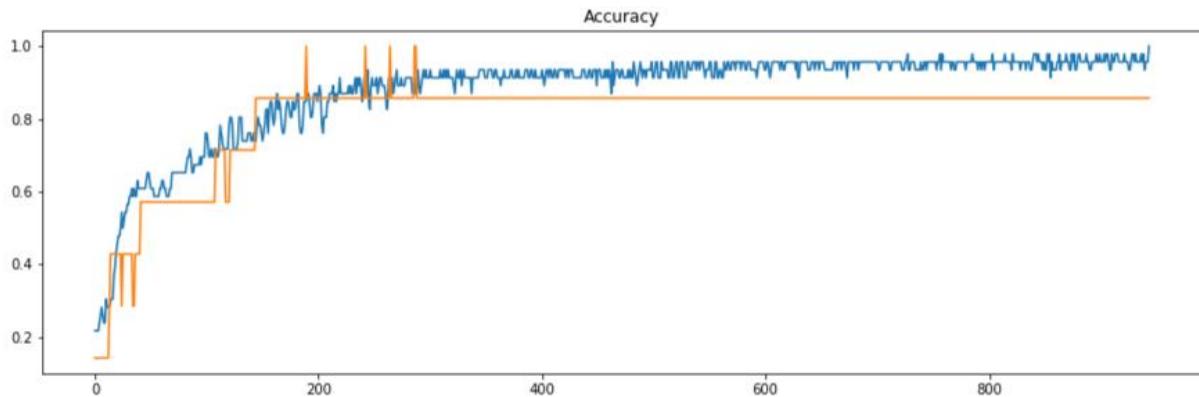
Figura 34 - GRÁFICO DA PERDA DE TREINO E TESTE



Fonte: O autor

A Figura 34 representa a perda (*loss function*) ao longo das épocas nos conjuntos de treinamento e de teste.

Figura 35 - GRÁFICO DA ACURÁCIA DE TREINO E TESTE



Fonte: O autor

A Figura 35 demonstra a acurácia (*accuracy*) ao longo das épocas nos conjuntos de treinamento e de teste.

A definição do melhor modelo de RNA obtido após a etapa de treinamento é evidenciado na Tabela 9 que exibe suas características gerais.

Tabela 9 – PARÂMETROS DE CONFIGURAÇÕES DA REDE NEURAL

N. Entrada	Função Entrada	N. Ocultos	Função Oculta	N. Saída	Função Saída	Época	Lote
20	Relu	15	Relu	7	Softmax	2500	25

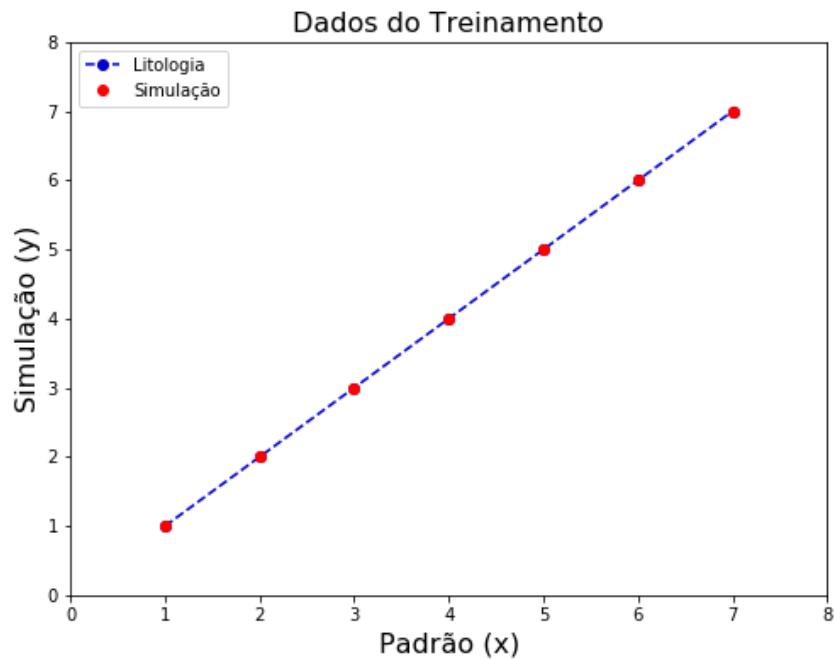
Fonte: O autor

Após o treinamento da RNA e definição do modelo, como mencionado anteriormente (Capítulo 4), realizou-se uma simulação, sendo os dados de entrada o conjunto (treino), após ser avaliado pelo modelo resultou no conjunto (y) que foi comparado com o conjunto (x), ou seja, o conjunto com os padrões das respectivas classes litológicas as quais eles pertencem.

A avaliação seguiu a fórmula do Erro Relativo que é a diferença existente entre o valor da classe que o modelo pertence pelo valor predito e o resultado dividido pelo valor da classe do modelo, podendo ser observado na equação a seguir:

$$Er = \frac{valor_{classe} - valor_{predito}}{valor_{classe}} \vee \quad (12)$$

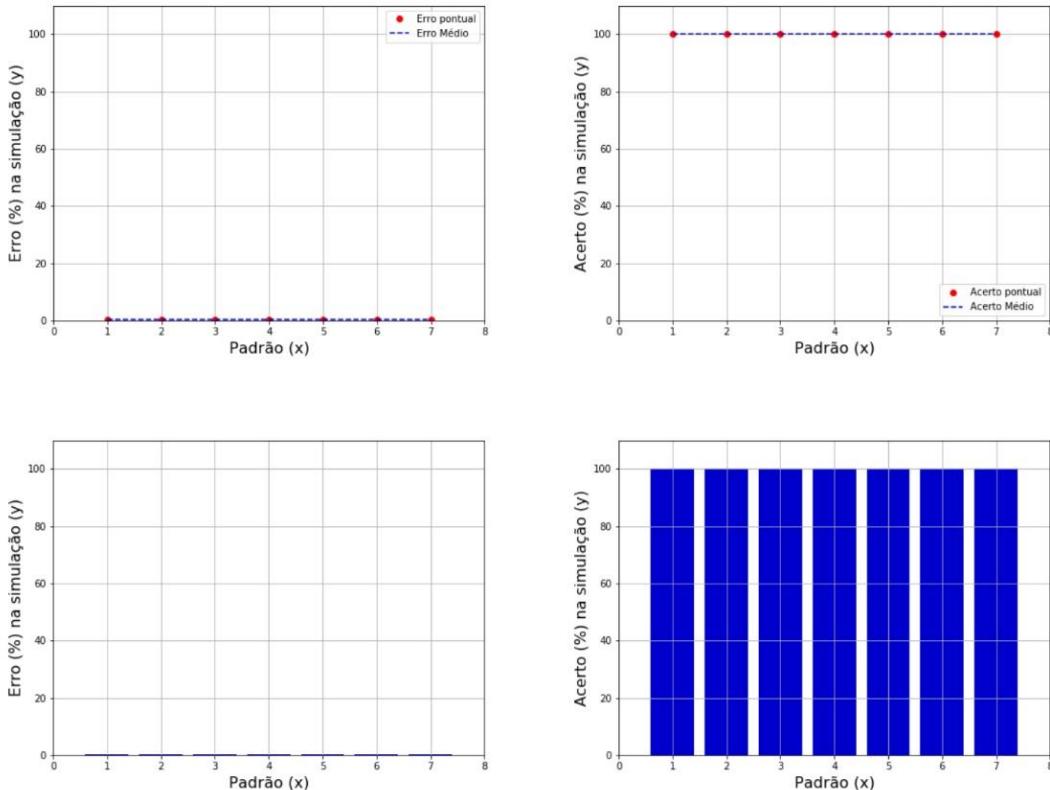
Figura 36 - EFICIÊNCIA DO TREINAMENTO NA SIMULAÇÃO



Fonte: O autor

Observa-se na Figura 36 a eficiência do modelo realizado por uma simulação. O conjunto de treinamento (treino) após a submissão no modelo, resultando no conjunto (y) com sua correspondência litológica o conjunto (x) conforme Fournier & Borgomano (2009).

Figura 37 – ERRO RELATIVO DA SIMULAÇÃO DO TREINAMENTO

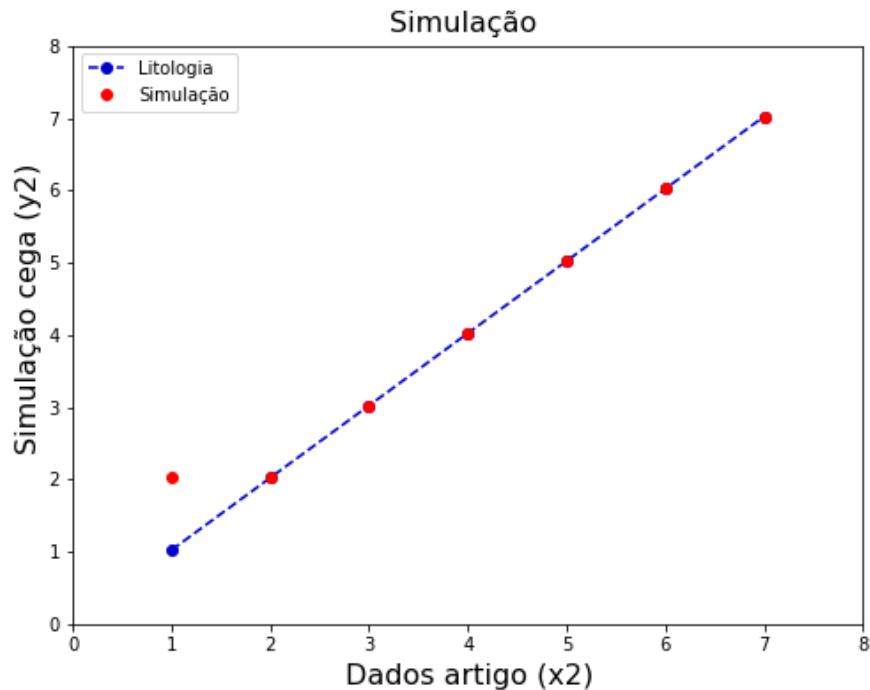


Fonte: O autor

Observamos na Figura 37 uma verificação da simulação para o conjunto de dados de treinamento, os erros (coluna à esquerda) e os acertos (coluna à direita) obtido pela simulação. O percentual de acerto foi de 100%, logo o erro 0%, valores considerados excelentes, ou seja, as classes litológicas foram previstas corretamente.

Posterior à verificação da eficiência do treinamento eficaz do modelo da RNA, foi realizado uma simulação usando como entrada o conjunto de teste “cego” (teste), observados no Anexo B, tendo suas respectivas classes litológicas (x_2) prevista por Fournier & Borgomano (2009), produzindo como resultado o padrão litológico previsto pela rede (y_2). Foi também utilizando a equação (12) para calcular o erro relativo.

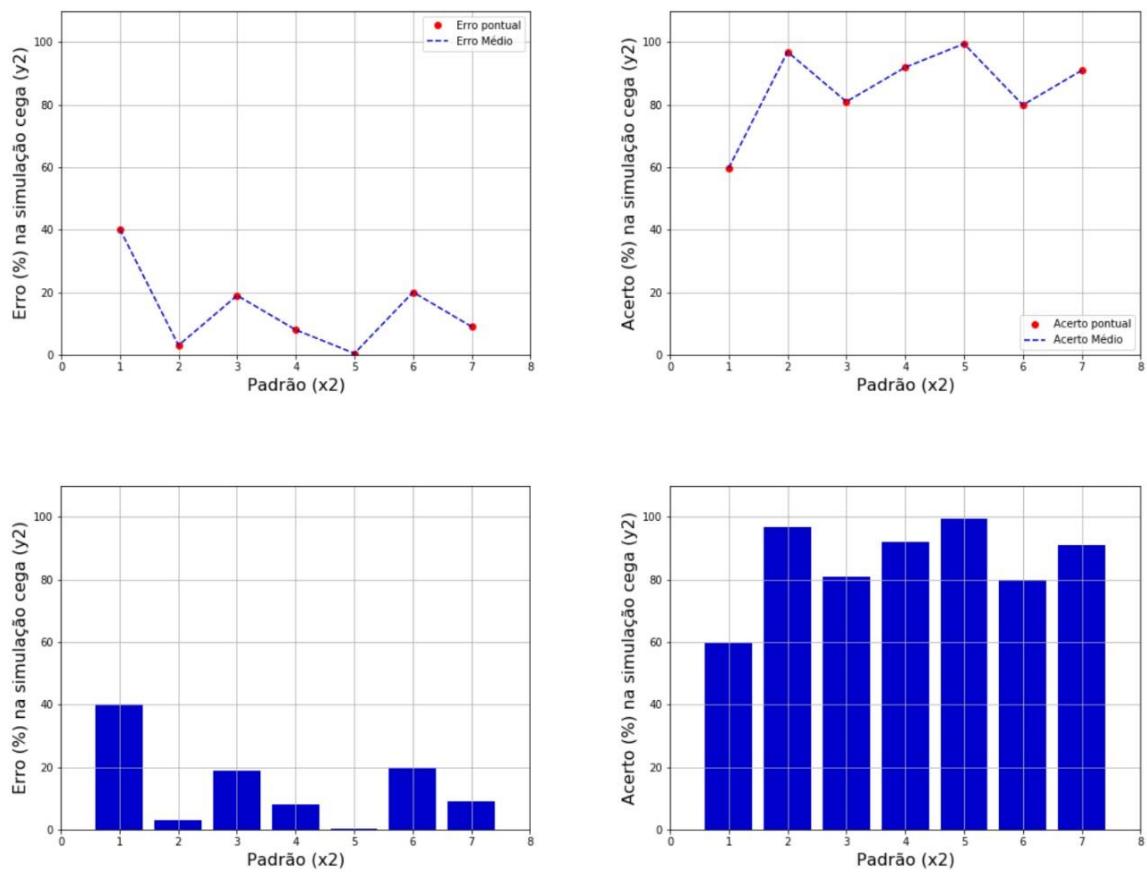
Figura 38 -VERIFICAÇÃO DA EFICÁCIA DO TESTE CEGO



Fonte: O autor

Observa-se na Figura 38 a relação do modelo que foi treinando predizendo no conjunto de dados do teste a ‘cego’ e comparando com a classificação prevista. Ou seja, a linha tracejada em azul faz referência às classes litológicas (x_2) contidas no conjunto de dados ‘cegos’ e os círculos vermelhos são as previsões realizadas pelo modelo (y_2) conforme Fournier & Borgomano (2009).

Figura 39 - ERRO RELATIVO DA SIMULAÇÃO DO TESTE CEGO



Fonte: O autor

Observamos na Figura 39 uma verificação da simulação para o conjunto de dados de teste “cego”, os erros (coluna à esquerda) e os acertos (coluna à direita) obtidos pela simulação. O conjunto da simulação de litologia (y_2) após ser predito pelo modelo de RNA e suas respectivas litologias (x_2). O percentual de acerto foi de 85,71%, logo o erro 14,29%.

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou um modelo de Redes Neurais Artificiais utilizando o algoritmo de *backpropagation* com seu desenvolvimento na linguagem e bibliotecas do *Python*. O algoritmo desenvolvido foi capaz de apreender e predizer classes litológicas de rochas carbonáticas.

A ideia é que um algoritmo de rede neural multicamadas poderia resultar em melhores previsões ao fazer uso dos tipos de conexões existentes nos neurônios, pesos sinápticos, funções de ativações e no processo de ida e vinda (*forward* e *backward*), corrigindo erros e atualizando seus pesos. Por meio do treinamento realizado, foi perceptível o quanto isso acontecia, ou seja, a rede aprendia em cada época que se passava e, com isso, ela ia melhorando seu desempenho, realizando melhores classificações, aumentando sua acurácia.

Foi observado que o modelo proposto após os testes nesse trabalho respondeu de maneira satisfatória na simulação, alcançando uma acurácia próxima de 100%, ou seja, um modelo ideal, apresentando um valor satisfatório no conjunto de teste “cego” (acerto 85.71%), neste caso o modelo proposto foi bem validado utilizando a acurácia como métrica.

Ressaltando que o conjunto de dados de treinamento e teste possui poucas instâncias por ser composto de medições laboratoriais. Contudo, é possível observar que o algoritmo criado possui um bom desempenho sendo uma ferramenta útil para previsões de litologias e auxiliando no processo de tomada de decisões.

4.1 Trabalhos Futuros

Esse tópico abordará possíveis trabalhos a serem realizados, podendo complementar o atual:

- **Projeto de previsões de litologias em conjunto de dados com características faltantes do modelo criado:** Em virtude da possibilidade de existir diferentes poços de petróleo e também para testar a capacidade das Redes Neurais Artificiais em realizar previsões com características faltantes, assim, analisando o desempenho do modelo, buscando interpretar e analisar os resultados obtidos. Propondo assim uma análise de como as Redes Neurais Artificiais lidam com características ausentes, observando se isso é prejudicial para seu desempenho.

- **Projeto de utilizações de outros algoritmos de Redes Neurais Artificiais, analisando os desempenhos e comparando os seus resultados:** Partindo da premissa da existência de outros modelos de RNA, na tentativa de entender como cada um deles sobressai a esse conjunto de dados e qual seria o melhor modelo em questão. Esta proposta visa analisar os resultados obtidos de cada uma dessas RNAs na tentativa de compreender como cada rede estruturou o aprendizado e assim determina qual seria o melhor algoritmo de RNA para esse conjunto de dados.
- **Projeto de utilizações de outros algoritmos de Aprendizado de Máquina Supervisionado, realizando o comparativo entre eles:** Partindo da premissa da existência de vários algoritmos de Aprendizado de Máquina Supervisionado, na tentativa de analisar qual possui melhores resultados sobre essa base de dados. Esta proposta de trabalho visa estimar qual é o melhor algoritmo de Aprendizado de Máquina Supervisionado a ser aplicado para predizer litologias de rochas carbonáticas.
- **Utilizar outras técnicas para dimensionar os dados:** Em virtude da existência de diferentes métodos, cabe a discussão de qual método pode apresentar o melhor desempenho e tenta buscar os porquês desses resultados.

REFERÊNCIAS BIBLIOGRÁFICAS

- AZEVEDO, F. M.; BRASIL, L. M.; OLIVEIRA, R. C. L. Redes neurais com aplicações em controle e em sistemas especialistas. 1 ed. Florianópolis: Visual Books, 2000.
- BARROS Maria; GUIMARÃES Marcus; SAYD Alexandre. Noções de Reservatório. 2^a Ed. Apostila Curso de Especialização Completação, Avaliação, Flexitubo e Arame – CAFA, 2009.
- BRINK, H.; RICHARDS, J. *Real World Machine Learning*. [S.l.]: Manning Publications C.O, 2014.
- CARVALHO, L. A. Data mining: a mineração de dados no marketing, medicina, economia, engenharia e administração. São Paulo: Ciência Moderna, 2005.
- COPPIN, B. Inteligência Artificial. Rio de Janeiro: LTC. ISBN: 978-8521617297, 2013.
- DATE, J. C. Introdução a Sistemas de Bancos de Dados, 8a. Edição, Rio de Janeiro: Elsevier, 2004.
- DOMINGOS, P. A Few Useful Things to Know About Machine Learning. Commun. ACM. 55. 78–87, 2012. Doi: 10.1145/2347736.2347755
- FAYYAD, U. M., PIATETSKY-SHAPIRO, G. & SMYTH, P. Advances in knowledge discovery and data mining. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth & R. Uthurusamy (ed.), (pp. 1–34). American Association for Artificial Intelligence. ISBN: 0-262-56097-6, 1996.
- FOURNIER, F., BORGOMANO, J., Critical porosity and elastic properties of microporous mixed carbonate-siliciclastic rocks. Geophysics 74, E93–E109, 2009.
- FÜRNKRANZ, J., GAMBERGER, D., LAVRAC, N., Foundations of Rule Learning. Springer. ISBN: 978-3-540-75196-0, 2012.
- GARTNER, Top 10 Strategic Technology Trends for 2019. ID: G00374252, 2018.
- HAN, J.W., KAMBER, M. AND PEI, J., Data Mining Concepts and Techniques. 3rd Edition, Morgan Kaufmann Publishers, Waltham, 2012.
- HAND, D. J., SMITH, P., MANNILA, H., Principles of Data Mining. Cambridge, MA, USA: MIT Press. ISBN 0-262-08290-X, 9780262082907, 2001.
- HAYKIN, S., Redes Neurais: princípios e prática. 2 ed. São Paulo: Bookman, 2001.
- LEVADA, M. M. O., FIERI, W. J., PIVESSO, M. S. G., Apontamentos Teóricos de Citologia, Histologia e Embriologia. 5 ed. São Paulo, Catálise Editora, 1996.

- LÉVY, P., As Tecnologias da Inteligência. O Futuro do pensamento na era da Informática. 1. ed. Rio de Janeiro: Editora 34, 1993.
- MCCULLOCH, W.S. & PITTS, W., Bulletin of Mathematical Biophysics. . 5: 115, 1943.
- MENDES, M. A., Sistematização da assistência de enfermagem usando raciocínio baseado em casos implementado em JAVA. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade de São Paulo, São Paulo, 2009. Disponível em: <>. Acesso em: 15 mar. 2019.
- MITCHELL, T., Machine Learning. New York, NY: McGraw-Hill. ISBN 0-07-042807-7. 1997.
- OBSERVATORY. B. I., Big data: Artificial Intelligence. Netherlands, 2013.
- REZENDE, D. A., Planejamento de Sistemas de Informação e Informática. São Paulo: Atlas, 2003.
- RUSSEL, S.; NORVIG, P., Inteligência Artificial. Tradução de Regina Célia Simille de Macedo. 3 ed. Rio de Janeiro: Elsevier, 2013.
- SCHWAB, K., A quarta revolução industrial. São Paulo: Edipro, 2016.
- SILBERSCHATZ, A., Sistemas de Bancos de Dados, Makron Books, 1999.
- SILVA, A., LIMA, I. A., MISSAGIA, R., CEIA, M., CARRASQUILLA, A. & LOPES, A. N., Artificial neural networks to support petrographic classification of carbonate-siliciclastic rocks using well logs and textural information. Journal of Applied Geophysics. 117, 2015. Doi: 10.1016/j.jappgeo.2015.03.027
- SILVA, I. N., SPATTI, D. H., FLAUZINO, R. A., Redes Neurais Artificiais: para engenharia e ciências aplicadas. São Paulo: Artliber, 2010.
- STAIR, R. M., Princípios de Sistemas de Informação. 9^a ed. Editora Pioneira: São Paulo, 2010.
- STRAUSS, A. L., & CORBIN, J. M., Basics Of Qualitative Research: Grounded Theory Procedures And Techniques. The Modern Language Journal, 77, 1990. Doi: 10.2307/328955
- TEIXEIRA, W.; TOLEDO, C.; FAIRCHILD, T.; TAIOLI, F. Decifrando a Terra. São Paulo: Oficina de Textos, 2000.
- THOMAS, José E. Fundamentos de engenharia de petróleo. 2.ed Rio de Janeiro Editora Interciênciac, 2001.

THOMSEN, E., OLAP: construindo sistemas de informações multidimensionais. Rio de Janeiro: Campus, 2002.

WITTEN, I. & FRANK, I.H., Data Mining - Practical Machine Learning Tools and Techniques with JAVA Implementations. Morgan Kaufmann, 31, 2002.

ZACCONE, G., Getting Started with TensorFlow. Packt Publishing, 2016.

ANEXO A – Conjunto de dados obtidos de Fournier & Borgomano (2009)

Poço	Profundidade (m)	Porosidade (%)	Densidade bulk rocha seca g/cm ³	VP m/s 70 MPa	VSI m/s 70 MPa	VS2 m/s 70 MPa	K1 GPa 70 MPa	K2 GPa 70 MPa	M1 GPa 70 MPa	M2 GPa 70 MPa	Tamanho do grão	Fração carbonática de microporos	Calcita	Dolomita	Quartzo	Ortoclásio	Albita	Argilas	Pirita	α_{KT} (Razão de aspecto da inclusão de poros)	Litologia
1	0.5	0.82	2.7	6238	3202	3188	68.15	68.48	27.68	27.44	3	64.4	92.1	0	7.9	0	0	0	0	0.7	1
1	2.7	0.39	2.69	6494	3498	3400	69.47	71.89	32.87	31.06	3	70	98.9	0	1.1	0	0	0	0	0.7	1
1	9.85	0.62	2.68	6344	3231	3258	70.54	69.91	27.97	28.44	3	64.27	98.9	0	1.1	0	0	0	0	0.65	1
1	15.7	0.18	2.69	6172	3242	3268	64.87	64.27	28.32	28.77	3	69.43	99.2	0	0.8	0	0	0	0	0.7	1
1	21.1	0.44	2.69	6375	3196	3209	72.58	72.28	27.44	27.66	3	69.86	99.8	0	0.2	0	0	0	0	0.7	1
1	22.1	0.66	2.68	6368	3217	3230	71.81	71.51	27.78	28	2	69.5	99.3	0	0.7	0	0	0	0	0.7	1
1	25.55	0.24	2.69	6207	3221	3206	66.44	66.78	27.91	27.65	2	67.9	100	0	0	0	0	0	0	0.7	1
1	28.45	0.65	2.68	6243	3230	3243	67.11	66.81	27.93	28.16	4	61.87	99.8	0	0.2	0	0	0	0	0.62	2
1	30.4	1.05	2.67	6341	3260	3244	69.62	69.99	28.41	28.14	4	69.72	99.6	0	0.5	0	0	0	0	0.7	2
1	32.35	1.73	2.65	6110	3261	3269	61.36	61.18	28.18	28.32	2	71.35	99.1	0	0.9	0	0	0	0	0.72	2
1	34.4	0.93	2.67	6404	3446	3332	67.27	70.03	31.73	29.66	3	68.72	98.2	0	1.8	0	0	0	0	0.7	2
1	36.45	2.65	2.62	5702	3043	3030	52.89	53.16	24.28	24.08	3	58.75	97.9	0	2.1	0	0	0	0	0.6	2
1	42.4	4.44	2.58	5738	3091	3159	52.02	50.56	24.62	25.72	4	84.17	99	0	1	0	0	0	0	0.85	2
1	56.5	3.39	2.61	5846	3338	3296	50.34	51.31	29.03	28.31	4	34.38	98.2	0	1.8	0	0	0	0	0.35	2
1	64.4	6.4	2.49	4644	2741	2810	28.78	27.5	18.72	19.67	5	43.96	56.4	0	35.1	1.5	2.5	4.5	0	0.75	2
1	66.4	15.2	2.24	3773	2368	2431	15.13	14.23	12.55	13.23	5	52.5	75.7	0	23.4	0	0	0.9	0	0.7	2
1	72.5	15.2	2.26	3701	2242	2321	15.79	14.7	11.34	12.16	4	47.2	75.7	0	23.4	0	0	0.9	0	0.8	2
1	74.4	5.62	2.53	4623	2507	2547	32.89	32.2	15.91	16.42	4	36.5	75.7	0	23.4	0	0	0.9	0	0.5	3
1	76.45	8.61	2.46	4334	2471	2479	26.17	26.04	15.02	15.11	5	67.2	75.7	0	23.4	0	0	0.9	0	0.8	3
1	78.45	16.2	2.21	3730	2084	2248	17.93	15.84	9.59	11.16	5	46.9	75.7	0	23.4	0	0	0.9	0	0.7	3
1	80.4	20.8	2.06	3307	2032	2060	11.21	10.9	8.53	8.76	5	46	11.5	0	85.5	0	0	3	0	0.92	3
1	86.45	6.94	2.48	4890	3065	3029	28.25	28.98	23.31	22.76	4	16.9	23.2	0	76.8	0	0	0	0	0.6	3
1	92.4	11.7	2.33	4036	2585	2523	17.22	18.2	15.59	14.85	4	20.58	11.5	0	85.5	0	0	3	0	0.95	3
1	92.4	11.7	2.33	4036	2585	2523	17.22	18.2	15.59	14.85	4	20.58	11.5	0	85.5	0	0	3	0	0.95	3
1	106.3	9.31	2.4	4049	2390	2448	21.06	20.17	13.71	14.38	4	32.84	11.5	0	85.5	0	0	3	0	0.8	3

1	110.4	6.7	2.49	4745	2585	2626	33.88	33.17	16.64	17.17	5	53.84	75.7	0	23.4	0	0	0.9	0	0.7	4
1	112.4	3.07	2.59	5543	3495	3286	37.41	42.3	31.65	27.98	4	23.6	92.1	0	7.9	0	0	0	0	0.4	4
1	114.35	2.01	2.64	6240	3332	3230	63.74	66.09	29.32	27.55	3	66.15	97.9	0	2.1	0	0	0	0	0.7	4
1	116.35	6.71	2.5	5058	2909	2853	35.78	36.86	21.17	20.37	4	20.2	65.5	0	34	0	0	0.55	0	0.3	4
1	118.4	7.39	2.44	4662	2705	2755	29.21	28.32	17.84	18.51	4	28.91	40	0	55	0	0	3.5	1.5	0.65	4
1	124.4	16.6	2.2	4162	2666	2582	17.27	18.57	15.65	14.68	5	34.55	29	0	61	1.7	2	3.9	2.5	0.85	4
1	126.4	6.84	2.48	5274	2940	2899	40.34	41.13	21.4	20.81	4	79.83	77.17	0	19.37	0.89	0.59	0.89	1.19	0.9	4
1	132.45	7.35	2.48	4599	2567	2611	30.67	29.91	16.34	16.91	5	49.64	89.6	1.2	8.3	0	0	0.9	0	0.55	4
1	134.4	20.2	2.1	3620	1980	1967	16.56	16.7	8.24	8.13	4	51.6	23.9	0	75.6	0	0	0.5	0	0.6	4
1	136.4	14.5	2.26	3771	2344	2306	15.57	16.1	12.41	12.01	5	31.15	23.9	0	75.6	0	0	0.5	0	0.9	4
1	140.4	8.61	2.43	3810	2302	2378	18.14	16.98	12.9	13.77	4	47.49	77.17	0	19.37	0.89	0.59	0.89	1.19	0.6	4
1	146.4	5.43	2.55	5097	2903	2892	37.62	37.83	21.5	21.34	3	45.64	94.9	0	5.1	0	0	0	0	0.48	5
1	148.4	4.23	2.59	5586	2989	3000	49.9	49.67	23.11	23.28	2	81.59	95.5	0	4.1	0	0	0.5	0	0.85	5
2	64.4	1.68	2.65	5463	3105	3066	45.07	45.92	25.57	24.94	2	13.33	66.6	0	33.4	0	0	0	0	0.2	6
2	114.4	3.18	2.62	5226	2820	2995	43.69	40.14	20.8	23.46	4	21.28	84.9	0	15.1	0	0	0	0	0.25	6
2	136.4	2.4	2.63	5640	3142	3102	49.07	49.94	25.98	25.32	3	17.77	88.8	0	11.2	0	0	0	0	0.2	6
2	84.4	5.14	2.54	5123	2942	2943	37.41	37.39	22.02	22.04	3	52.8	75.7	0	23.4	0	0	0.9	0	0.8	6
2	6.3	23.9	2.02	3496	2157	2139	12.15	12.35	9.39	9.23	2	38.03	12.4	0	87.6	0	0	0	0	0.95	6
2	20.45	21.2	2.1	3535	2204	2140	12.62	13.39	10.18	9.6	1	29.24	12.4	0	87.6	0	0	0	0	0.95	7
2	34.4	7.39	2.48	4544	2829	2815	24.73	25	19.84	19.65	2	28.98	12.4	0	87.6	0	0	0	0	0.6	7
2	78.4	5.22	2.54	5026	3046	3068	32.78	32.32	23.59	23.94	2	17.69	41.7	0	58.3	0	0	0	0	0.4	7
1	28.45	0.65	2.68	6243	3230	3243	67.11	66.81	27.93	28.16	4	61.87	99.8	0	0.2	0	0	0	0	0.62	7
1	148.4	4.23	2.59	5586	2989	3000	49.9	49.67	23.11	23.28	2	81.59	95.5	0	4.1	0	0	0.5	0	0.85	7
2	114.4	3.18	2.62	5226	2820	2995	43.69	40.14	20.8	23.46	4	21.28	84.9	0	15.1	0	0	0	0	0.25	7
1	72.5	15.2	2.26	3701	2242	2321	15.79	14.7	11.34	12.16	4	47.2	75.7	0	23.4	0	0	0.9	0	0.8	7
1	64.4	6.4	2.49	4644	2741	2810	28.78	27.5	18.72	19.67	5	43.96	56.4	0	35.1	1.5	2.5	4.5	0	0.75	7
1	86.45	6.94	2.48	4890	3065	3029	28.25	28.98	23.31	22.76	4	16.9	23.2	0	76.8	0	0	0	0	0.6	7
1	106.3	9.31	2.4	4049	2390	2448	21.06	20.17	13.71	14.38	4	32.84	11.5	0	85.5	0	0	3	0	0.8	7

Observações:

Código de litologia: (1) limestone (grainstone), (2) limestone (wacke-packstone), (3) quartz-rich limestone (grainstone), (4) quartz-rich limestone (wacke-packstone), (5) slightly argillaceous quartz-rich limestone, (6) clean sandstone, (7) sandstone with carbonate matrix

Código do tamanho do grão: (1) VC, very coarse; (2) C, coarse; (3) M, medium; (4) F, fine; (5) VF, very fine.

ANEXO B – Conjunto de dados de treinamento e teste da Rede Neural Artificial

B1 – Conjunto de treinamento - dados ordenados por litologia

Poço	Profundidade (m)	Porosidade (%)	Densidade bulk rocha seca g/cm ³	VP m/s 70 MPa	VS1 m/s 70 MPa	VS2 m/s 70 MPa	K1 GPa 70 MPa	K2 GPa 70 MPa	M1 GPa 70 MPa	M2 GPa 70 MPa	Tamanho do grão	Fração carbonática de microporos	Calcita	Dolomita	Quartzo	Ortoclásio	Albita	Argilas	Pirita	α_{KT} (Razão de aspecto da inclusão de poros)	Litologia	
1	21.1	0.44	2.69	6375	3196	3209	72.58	72.28	27.44	27.66	3	69.86	99.8	0	0.2	0	0	0	0	0.7	1	
1	22.1	0.66	2.68	6368	3217	3230	71.81	71.51	27.78	28	2	69.5	99.3	0	0.7	0	0	0	0	0.7	1	
1	28.45	0.65	2.68	6243	3230	3243	67.11	66.81	27.93	28.16	4	61.87	99.8	0	0.2	0	0	0	0	0.62	1	
1	30.4	1.05	2.67	6341	3260	3244	69.62	69.99	28.41	28.14	4	69.72	99.6	0	0.5	0	0	0	0	0.7	1	
1	36.45	2.65	2.62	5702	3043	3030	52.89	53.16	24.28	24.08	3	58.75	97.9	0	2.1	0	0	0	0	0.6	1	
1	114.35	2.01	2.64	6240	3332	3230	63.74	66.09	29.32	27.55	3	66.15	97.9	0	2.1	0	0	0	0	0.7	1	
1	2.7	0.39	2.69	6494	3498	3400	69.47	71.89	32.87	31.06	3	70	98.9	0	1.1	0	0	0	0	0.7	2	
1	9.85	0.62	2.68	6344	3231	3258	70.54	69.91	27.97	28.44	3	64.27	98.9	0	1.1	0	0	0	0	0.65	2	
1	15.7	0.18	2.69	6172	3242	3268	64.87	64.27	28.32	28.77	3	69.43	99.2	0	0.8	0	0	0	0	0	0.7	2
1	25.55	0.24	2.69	6207	3221	3206	66.44	66.78	27.91	27.65	2	67.9	100	0	0	0	0	0	0	0	0.7	2
1	32.35	1.73	2.65	6110	3261	3269	61.36	61.18	28.18	28.32	2	71.35	99.1	0	0.9	0	0	0	0	0	0.72	2
1	34.4	0.93	2.67	6404	3446	3332	67.27	70.03	31.73	29.66	3	68.72	98.2	0	1.8	0	0	0	0	0	0.7	2
1	42.4	4.44	2.58	5738	3091	3159	52.02	50.56	24.62	25.72	4	84.17	99	0	1	0	0	0	0	0	0.85	2
1	56.5	3.39	2.61	5846	3338	3296	50.34	51.31	29.03	28.31	4	34.38	98.2	0	1.8	0	0	0	0	0	0.35	2
1	148.4	4.23	2.59	5586	2989	3000	49.9	49.67	23.11	23.28	2	81.59	95.5	0	4.1	0	0	0.5	0	0	0.85	2
1	0.5	0.82	2.7	6238	3202	3188	68.15	68.48	27.68	27.44	3	64.4	92.1	0	7.9	0	0	0	0	0	0.7	3
1	112.4	3.07	2.59	5543	3495	3286	37.41	42.3	31.65	27.98	4	23.6	92.1	0	7.9	0	0	0	0	0	0.4	3
1	116.35	6.71	2.5	5058	2909	2853	35.78	36.86	21.17	20.37	4	20.2	65.5	0	34	0	0	0.55	0	0	0.3	3
1	132.45	7.35	2.48	4599	2567	2611	30.67	29.91	16.34	16.91	5	49.64	89.6	1.2	8.3	0	0	0.9	0	0	0.55	3
2	64.4	1.68	2.65	5463	3105	3066	45.07	45.92	25.57	24.94	2	13.33	66.6	0	33.4	0	0	0	0	0	0.2	3
2	114.4	3.18	2.62	5226	2820	2995	43.69	40.14	20.8	23.46	4	21.28	84.9	0	15.1	0	0	0	0	0	0.25	3
2	136.4	2.4	2.63	5640	3142	3102	49.07	49.94	25.98	25.32	3	17.77	88.8	0	11.2	0	0	0	0	0	0.2	3
1	66.4	15.2	2.24	3773	2368	2431	15.13	14.23	12.55	13.23	5	52.5	75.7	0	23.4	0	0	0.9	0	0	0.7	4
1	72.5	15.2	2.26	3701	2242	2321	15.79	14.7	11.34	12.16	4	47.2	75.7	0	23.4	0	0	0.9	0	0	0.8	4

1	74.4	5.62	2.53	4623	2507	2547	32.89	32.2	15.91	16.42	4	36.5	75.7	0	23.4	0	0	0.9	0	0.5	4
1	76.45	8.61	2.46	4334	2471	2479	26.17	26.04	15.02	15.11	5	67.2	75.7	0	23.4	0	0	0.9	0	0.8	4
1	78.45	16.2	2.21	3730	2084	2248	17.93	15.84	9.59	11.16	5	46.9	75.7	0	23.4	0	0	0.9	0	0.7	4
1	110.4	6.7	2.49	4745	2585	2626	33.88	33.17	16.64	17.17	5	53.84	75.7	0	23.4	0	0	0.9	0	0.7	4
1	126.4	6.84	2.48	5274	2940	2899	40.34	41.13	21.4	20.81	4	79.83	77.17	0	19.37	0.89	0.59	0.89	1.19	0.9	4
1	140.4	8.61	2.43	3810	2302	2378	18.14	16.98	12.9	13.77	4	47.49	77.17	0	19.37	0.89	0.59	0.89	1.19	0.6	4
1	146.4	5.43	2.55	5097	2903	2892	37.62	37.83	21.5	21.34	3	45.64	94.9	0	5.1	0	0	0	0	0.48	4
2	84.4	5.14	2.54	5123	2942	2943	37.41	37.39	22.02	22.04	3	52.8	75.7	0	23.4	0	0	0.9	0	0.8	4
1	64.4	6.4	2.49	4644	2741	2810	28.78	27.5	18.72	19.67	5	43.96	56.4	0	35.1	1.5	2.5	4.5	0	0.75	5
1	86.45	6.94	2.48	4890	3065	3029	28.25	28.98	23.31	22.76	4	16.9	23.2	0	76.8	0	0	0	0	0.6	6
2	6.3	23.9	2.02	3496	2157	2139	12.15	12.35	9.39	9.23	2	38.03	12.4	0	87.6	0	0	0	0	0.95	6
2	20.45	21.2	2.1	3535	2204	2140	12.62	13.39	10.18	9.6	1	29.24	12.4	0	87.6	0	0	0	0	0.95	6
2	34.4	7.39	2.48	4544	2829	2815	24.73	25	19.84	19.65	2	28.98	12.4	0	87.6	0	0	0	0	0.6	6
1	80.4	20.8	2.06	3307	2032	2060	11.21	10.9	8.53	8.76	5	46	11.5	0	85.5	0	0	3	0	0.92	7
1	92.4	11.7	2.33	4036	2585	2523	17.22	18.2	15.59	14.85	4	20.58	11.5	0	85.5	0	0	3	0	0.95	7
1	92.4	11.7	2.33	4036	2585	2523	17.22	18.2	15.59	14.85	4	20.58	11.5	0	85.5	0	0	3	0	0.95	7
1	106.3	9.31	2.4	4049	2390	2448	21.06	20.17	13.71	14.38	4	32.84	11.5	0	85.5	0	0	3	0	0.8	7
1	118.4	7.39	2.44	4662	2705	2755	29.21	28.32	17.84	18.51	4	28.91	40	0	55	0	0	3.5	1.5	0.65	7
1	124.4	16.6	2.2	4162	2666	2582	17.27	18.57	15.65	14.68	5	34.55	29	0	61	1.7	2	3.9	2.5	0.85	7
1	134.4	20.2	2.1	3620	1980	1967	16.56	16.7	8.24	8.13	4	51.6	23.9	0	75.6	0	0	0.5	0	0.6	7
1	136.4	14.5	2.26	3771	2344	2306	15.57	16.1	12.41	12.01	5	31.15	23.9	0	75.6	0	0	0.5	0	0.9	7
2	78.4	5.22	2.54	5026	3046	3068	32.78	32.32	23.59	23.94	2	17.69	41.7	0	58.3	0	0	0	0	0.4	7

B2 – Conjunto de teste “cego” - dados ordenados por litologia

Poço	Profundidad e (m)	Porosidade (%)	Densidade bulk rocha seca g/cm ³	VP m/s 70 MPa	VS1 m/s 70 MPa	VS2 m/s 70 MPa	K1 GPa 70 MPa	K2 GPa 70 MPa	M1 GPa 70 MPa	M2 GPa 70 MPa	Tamanho do grão	Fração carbonática de microporos	Calcita	Dolomita	Quartzo	Ortoclássio	Albita	Argilas	Pirita	α KT (Razão de aspecto da inclusão de poros)	Litologia
1	28.45	0.65	2.68	6243	3230	3243	67.11	66.81	27.93	28.16	4	61.87	99.8	0	0.2	0	0	0	0	0.62	1
1	148.4	4.23	2.59	5586	2989	3000	49.9	49.67	23.11	23.28	2	81.59	95.5	0	4.1	0	0	0.5	0	0.85	2
2	114.4	3.18	2.62	5226	2820	2995	43.69	40.14	20.8	23.46	4	21.28	84.9	0	15.1	0	0	0	0	0.25	3
1	72.5	15.2	2.26	3701	2242	2321	15.79	14.7	11.34	12.16	4	47.2	75.7	0	23.4	0	0	0.9	0	0.8	4
1	64.4	6.4	2.49	4644	2741	2810	28.78	27.5	18.72	19.67	5	43.96	56.4	0	35.1	1.5	2.5	4.5	0	0.75	5
1	86.45	6.94	2.48	4890	3065	3029	28.25	28.98	23.31	22.76	4	16.9	23.2	0	76.8	0	0	0	0	0.6	6
1	106.3	9.31	2.4	4049	2390	2448	21.06	20.17	13.71	14.38	4	32.84	11.5	0	85.5	0	0	3	0	0.8	7

APÊNDICE A - Ferramentas, Linguagem de Programação e Bibliotecas usadas

- **Anaconda**

O Anaconda é uma distribuição livre (*open source*) que possibilita a utilização da linguagem Python, ferramenta bastante utilizada para ciência de dados (*Data Science*) e AM, permitindo gerenciar pacotes de instalações pelo ambiente conda, ou seja, permite o gerenciamento de ambientes virtuais denominado conda que possui uma série de pacotes instaláveis, como também *Integrated Development Environment* - Ambientes de Desenvolvimento Integrados - (IDE's), e outros projeto correlacionados à área, permitindo a utilização das linguagens de programação Python, R e Julia.

Vale ressaltar que a distribuição integrou o Jupyter notebook que é uma aplicação web desenvolvida em cima do projeto IPython, permitindo a criação, manutenção e execução dos chamados notebooks que são documentos *JavaScript Object Notation* (JSON), em tradução Notação de Objetos JavaScript, em listas ordenadas de células. Essas células podem conter textos, códigos, fórmulas matemáticas, imagens, plotagem de gráficos, entre outras coisas, esses documentos podem ser convertidos em formatos *Hypertext Markup Language* - Linguagem de Marcação de Hipertexto - (HTML), *Portable Document Format* – Formato de Documento Portátil - (PDF), Python, Slide, entre demais outros.

- **Python**

A linguagem de programação *Python* foi concebida no final dos anos 80 e sua implementação começou em dezembro de 1989 por Guido van Rossum. *Python* é atualmente uma das mais populares linguagens dinamicamente tipificadas de propósito geral, junto com *Perl*, *PHP*, *Ruby*. Ela pode ser utilizada em diversas aplicações desde *web servers* que proveem serviços 24 horas por dia, sistemas com interface gráfica, testes automatizados, entre outros. É utilizado tanto por cientistas nos computadores mais rápidos do mundo quanto por crianças aprendendo a programar (ROSSUM, 2009).

Python é uma linguagem de programação criada em 1991 com o objetivo de produtividade e legibilidade. Ou seja, é uma linguagem que foi criada para produzir código

bom e fácil de manter de maneira rápida, suportando múltiplos paradigmas de programação. É uma linguagem interpretada e orientada a objetos, que suporta sobrecarga de operadores, oferece interfaces multiplataformas e tem seu código aberto (*Open-source*).

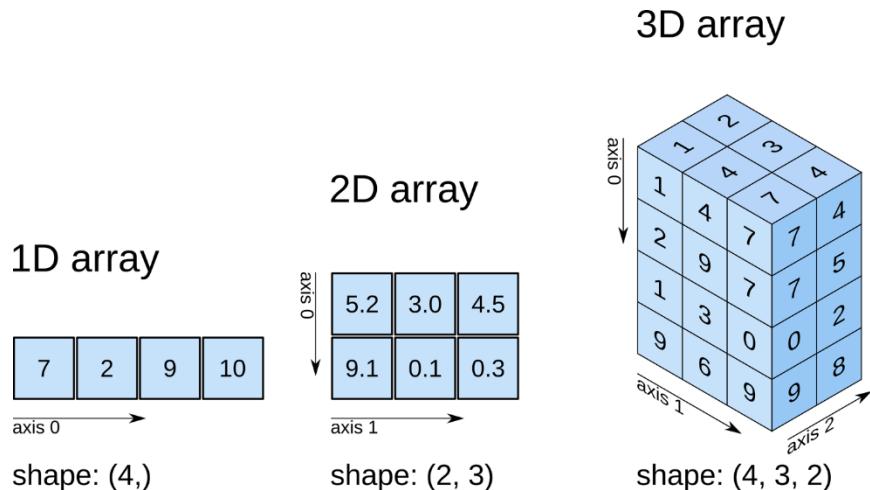
Uma das áreas que tem ganhado destaque é em aplicações que utilizam AM, MD, Data Science, ecossistema de Big Data, pois, apesar de ser uma linguagem interpretada possui um desempenho considerável quando trabalha com um grande número de dados, ademais é de fácil compreensão/aprendizado e existem inúmeras bibliotecas que facilitam a utilização de alguns algoritmos consagrados da área.

- ***Numpy***

É uma biblioteca para computação científica na qual implementa *arrays* multidimensionais e permite a fácil execução de operações matemáticas e lógicas como ordenação, seleção, transformação, operações estatísticas básicas etc. Essa biblioteca oferece as seguintes possibilidades:

- *Ndarray*, um *array* multidimensional rápido e eficiente em termos de espaço, proporcionando operações aritméticas vetorizadas e sofisticados recursos de transmissões;
- Funções matemáticas padrão para operações rápidas em matrizes inteiras de dados sem ter que gravar loops.
- Álgebra linear, geração de números aleatórios e capacidade de transformada de *Fourier*;
- Ferramentas para integração de códigos escritos em linguagens *C*, *C++* e *Fortran*.

Embora NumPy por si só não ofereça muitas funcionalidades analíticas de dados de alto nível, podem ser trabalhados em conjuntos com outras bibliotecas como pandas, explorando o máximo das funcionalidades de cada uma gerando mais eficiência de suas aplicações. Isto é importante para o condicionamento de dados que serão ofertados à RNA.

Figura 40 - ARRAYS NUMPY

Fonte: (Acessado em 31/05/2019)

Baseando-se na Figura 40, três tipos de *arrays* diferentes, criados com a numpy, *array* de 1 dimensão, 2 dimensões e 3 dimensões e suas respectivas notações de atribuição de código para a criação, referenciados por parênteses contendo primeiramente o número de linhas, a vírgula para separar as dimensões e a quantidade de dados daquela dimensão e o parêntese de finalização do *array* em questão.

- **Pandas**

Uma biblioteca *Open Source*, licenciada pela *Berkeley Software Distribution* (BSD) que fornece ferramentas para manipulação de estruturas de dados de forma extremamente simples. Operações complexas que trabalham com matrizes e vetores podem ser facilmente realizadas com uma ótima performance. Também lida bem com dados incompletos e não estruturas, provenientes de diferentes fontes, e fornece ferramentas para moldá-los, fundi-los, remodelá-los e organizá-los em grupos de dados afins.

Python tem sido ótimo para dos e preparação, mas menos para análise e modelagem de dados. A biblioteca panda ajuda a preencher essa lacuna, permitindo que seja executado todo o fluxo do trabalho de análise de dados apenas com a linguagem, alguns pontos de destaque da biblioteca podem ser vistos a seguir:

- Um objeto *Data Frame* (DF) que assemelha a uma matriz, mas suas colunas os índices são nomeadas e também contém variabilidade de dados.
- Ferramentas para ler e gravar dados entre estruturas de dados na memória e em diferentes formatos,
- Ferramentas para ler e gravar dados entre estruturas de dados na memória e em diferentes formatos, CSV , textos, SQL, entre outros;
- Remodelagem e giro flexíveis de conjuntos de dados;
- Colunas podem ser inseridas e excluídas de estrutura de dados para mutação de tamanhos;
- Fusão de alto desempenho e junção de conjuntos de dados.

Figura 41 - DATA FRAME DO PANDAS

INDEX	País	Capital	População
1	Bélgica	Bruxelas	123465
2	Índia	Nova Delí	456789
3	Brasil	Brasília	987654

Fonte: (Acessado em 31/05/2019)

A Figura 41 é um exemplo de um DF do Pandas, trazendo o conceito de tabelas ou matriz com a exemplificação dos índices que são as linhas e as colunas contendo a nomeação para elas.

- ***Scikit-Learn***

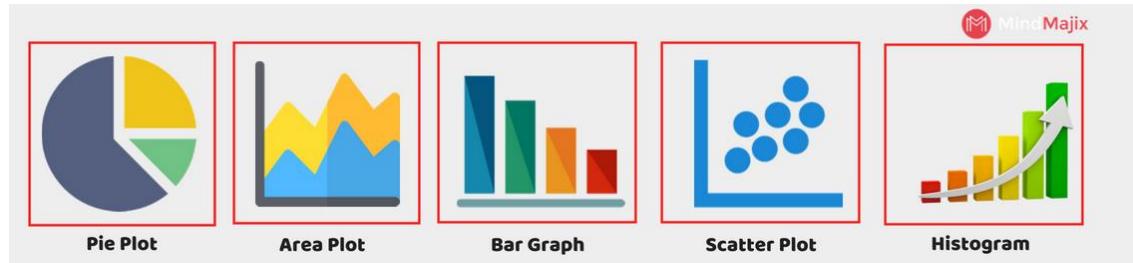
Existem várias bibliotecas *Python* que fornecem implementações sólidas de uma variedade de algoritmos de *Machine Learn* (Aprendizado de Máquina). Esse pacote fornece versões eficientes de grandes números de algoritmos comuns. O *Skicit-Learn* é caracterizado por uma API limpa, uniforme e simplificada, bem como por uma documentação on-line muito útil e completa. O benefício dessa uniformidade é que uma vez que se entende o uso básico e a sintaxe de um modelo com esse pacote, mudar para outro novo modelo ou algoritmo é bem simples, possuindo os seguintes princípios orientadores:

- Consistência, todos os objetos compartilham uma interface comum extraída de um conjunto limitado de métodos, como documentação consistente;
 - Inspeção, todos os valores de parâmetros especificados são expostos como atributos públicos;
 - Hierarquia de objetos limitados, somente algoritmos são representados por classes Python, conjuntos de dados são representados por formato padrão (Matrizes *Numpy*, Pandas DF, Matrizes esparsas *SciPy*);
 - Composição, muitas tarefas de AM podem ser expressas como sequências de algoritmos mais fundamentais, e o *Scikit-Learn* faz uso disso sempre que possível;
 - Padrões sensíveis, quando os modelos exigem parâmetros especificados pelo usuário, a biblioteca define um valor padrão apropriado.
- ***Matplotlib***

É uma biblioteca com recursos para geração de gráficos 2D a partir de *arrays*. Gráficos comuns podem ser criados com alta qualidade a partir de comandos simples, inspirados nos comandos dos gráficos do *MATLAB*, utilizada para geração de visualização e plotagem simples e poderosas de gráficos. Pode ser utilizados para gerar diversos tipos de gráficos como histogramas, gráficos de barras, gráficos de pizza, tudo de forma fácil e rápida, podendo ser conceitualmente divididos em três partes:

- *Pylab*, conjunto de funções em que permite a geração de códigos similares ao *MATLAB*;
- Front-end ou API, conjunto de classes que realizam o trabalho pesado, criando as figuras, textos, linhas entre outros. Essa é uma interface abstrata que independe da saída.
- Back-end, conjunto de funções que dependem do dispositivo de saída (*display*), podendo gerar gráficos em vários formatos.

Figura 42 - GRÁFICOS DO MATPLOTLIB



Fonte: (Acessado em 31/05/2019)

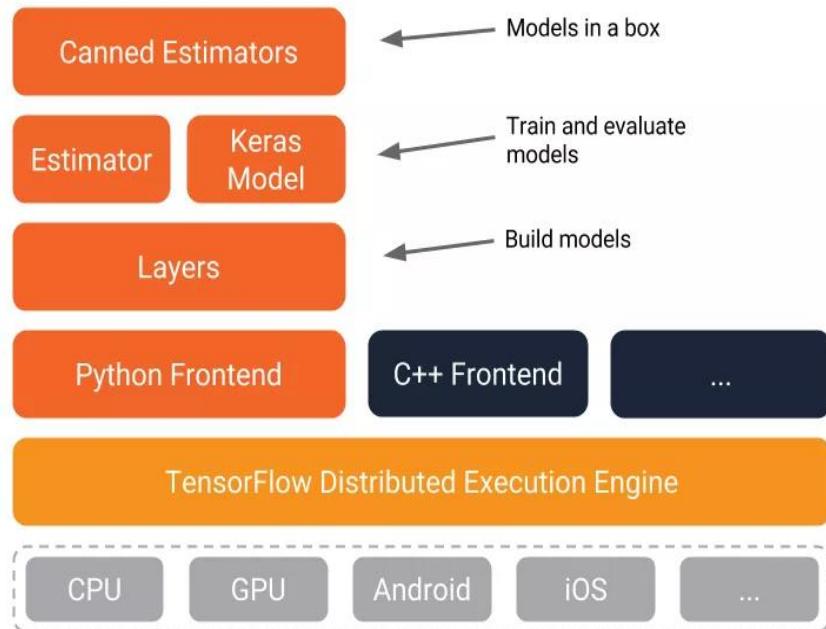
A Figura 42 exibe diferentes gráficos, podendo ser elaborados pela biblioteca *Matplotlib*, como histogramas, gráfico de pizza, entre outros, possibilitando a utilização de placa de vídeo gráfica para processamento.

- ***Tensorflow***

O *Tensorflow* é uma biblioteca open source, desenvolvido e mantido pela *Google* para AM e Aprendizado de Máquina Profundo, tendo como unidade básica os tensores que são matrizes n dimensional, estrutura que executam os cálculos. A base de fluxo de trabalho do *tensorfflow* são os gráficos computacionais é se dividem em operações e tensores.

Segundo Zaccone (2016), os gráficos de operações descrevem os cálculos que consomem os tensores e os gráficos de tensores representam valores que percorre o fluxo e como eles vão transferir esses dados. Eles ficam alocados a uma sessão que precisa ser iniciada para que o conjunto de dados e operações possa ser executado pelos gráficos computacionais.

Figura 43 - ESTRUTURA DO TENSORFLOW



Fonte: (Acessado em 31/05/2019)

A Figura 43 exibe a estrutura que compõem a biblioteca do *tensorflow*, podemos observar que a base do modelo é onde a biblioteca pode ser executada, tanto em unidade central de processamento (CPU), unidade gráfica de processamento (GPU) com utilização de placas de vídeo dedicadas, como também em sistemas operacionais mobile, entre outros.

O *tensorflow* foi desenvolvido utilizando a linguagem de programação *C++*, permitindo performance nos processamentos de dados executados, permitindo a utilização para seu fluxo de trabalho, ou seja o front-end, por diversas outras linguagens de programação, incluindo *c++*, *JavaScript*, *R*, *Python*, entre outras.

Vale ressaltar que permite a utilização de outras bibliotecas integradas a sua para que desempenhe o front-end da aplicação, como por exemplo, o *Keras*.

- ***Keras***

O *Keras* é uma API de RNA desenvolvida em Python, capaz de rodar como o *front-end* das principais bibliotecas de Aprendizado de Máquina, como por exemplo o *tensorflow*, permitindo neste caso a execução dos gráficos e tensores com uma abordagem mais simples e intuitiva.

O *Keras* permitiu executar os algoritmos para RNA combinando as camadas com

diferentes dimensões, função de ativação, função de custo, otimizadores de modelo, entre demais possibilidades, trazendo o ganho de performance desses modelos. A partir da versão 2.0 do *tensorflow* o *keras* passou a ser nativo dessa biblioteca, ressaltando que a *Google* auxiliou no desenvolvimento dessa API. Há duas maneiras de criar um modelo no *keras*, o sequencial ou o modelo da API funcional.

O modelo sequencial permite criar um fluxo de trabalho mais intuitivo, passando uma lista de instâncias e valores pertinentes a cada atributo, gerando assim o modelo *keras* para a compilação.

A API funcional permite trabalhar com o conceito de Programação Orientada a Objeto, fazendo que a classe que desenvolvida para RNA herde da classe do *keras* seu conjunto de funcionalidade, ou seja, seus métodos e atributos para implementação.

APÊNDICE B – CÓDIGOS DO PRÉ-PROCESSAMENTO E CRIAÇÃO DA RNA EM PYTHON

```
import IPython
from IPython import display
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import os
import os.path
import pandas as pd
import random
from random import seed
import seaborn as sns
import sklearn
from sklearn.base import BaseEstimator
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from subprocess import check_output
import tensorflow as tf
from tensorflow import keras
import yellowbrick as yw
from yellowbrick.datasets import load_concrete
from yellowbrick.regressor import PredictionError
import warnings
warnings.filterwarnings('ignore')

print("IPython version: {}".format(IPython.__version__))
print("matplotlib version: {}".format(matplotlib.__version__))
print("NumPy version: {}".format(np.__version__))
print("pandas version: {}".format(pd.__version__))
print("Keras version: {}".format(keras.__version__))
print("seaborn version: {}".format(sns.__version__))
print("Sklearn version: {}".format(sklearn.__version__))
```

```

print("Tensorfflow version: {}".format(tf.__version__))
print("Yellowbrick version: {}".format(yw.__version__))
print('Arquivos CSV:')
print(check_output(["ls", "/home/willian/Keras_Litologia/arquivos_csv"]).decode("utf8"))

def load_data():
    dftrain = pd.read_csv('/home/willian/Keras_Litologia/arquivos_csv/train.csv')
    dftest = pd.read_csv('/home/willian/Keras_Litologia/arquivos_csv/test.csv')
    return dftrain, dftest

def data_encode(data_frame, classificador):
    data_frame = pd.get_dummies(data_frame, columns=[classificador])
    return data_frame

def variables_x_y(data_frame, cabecalho, n):
    x = data_frame[cabecalho[:n]]
    y = data_frame[cabecalho[n:]]
    return x, y

def transfer_array_np(data_frame, vtype):
    np_data_frame = np.array(data_frame, dtype=vtype)
    return np_data_frame

def pre_processamento():
    dftrain, dftest = load_data()
    train = data_encode(dftrain, 'Litologia')
    test = data_encode(dftest, 'Litologia')
    train = train.drop(['Poço'], axis = 1)
    test = test.drop(['Poço'], axis = 1)
    cabecalho_list = list(train.columns.values)
    # n = - 7 representa primeira coluna do classificado (decrescente).
    n = -7
    x_train, y_train = variables_x_y(train, cabecalho_list, n)
    x_test, y_test = variables_x_y(test, cabecalho_list, n)

```

```

X_train = transfer_array_np(x_train, 'float32')
y_train = transfer_array_np(y_train, 'int8')
X_test = transfer_array_np(x_test, 'float32')
y_test = transfer_array_np(y_test, 'int8')
scaler = MinMaxScaler()
x_train = scaler.fit_transform(X_train)
x_test = scaler.fit_transform(X_test)
return x_train, y_train, x_test, y_test, dftrain, dftest

x_train, y_train, x_test, y_test, dftrain, dftest = preprocessamento()

epoch = 2500
batch_s = 15
path_file_models = '/home/willian/Keras_Litologia/model_save/'

class TerminateOnBaseline(tf.keras.callbacks.Callback):
    def __init__(self, monitor='acc', baseline=0.99, patience=1, mode='max'):
        super(TerminateOnBaseline, self).__init__()
        self.monitor = monitor
        self.baseline = baseline
        self.patience = patience
        self.mode = mode
    def on_epoch_end(self, epoch, logs=None):
        logs = logs or {}
        acc = logs.get(self.monitor)
        if acc is not None:
            if acc >= self.baseline:
                print('Epoch %d: Reached baseline, terminating training' %
( epoch))
                self.model.stop_training = True

def create_model(n_input, n_hidden, n_output, func_input, func_hidden, func_output):
    model = tf.keras.models.Sequential()

```

```

model.add(tf.keras.layers.Dense(n_input, input_dim = n_input, activation =
func_input, kernel_initializer = 'uniform', bias_initializer = 'RandomNormal'))
model.add(tf.keras.layers.Dense(n_hidden, activation = func_hidden))
model.add(tf.keras.layers.Dense(n_output, activation = func_output))
model.compile(loss = 'categorical_crossentropy', optimizer = keras.optimizers.SGD
(lr= 0.01, momentum = 0.9), metrics = ['acc', 'mse'])

return model

def load_model(path_file, num_model):
    json_file = open(path_file + 'model_' + num_model + '.json', 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    model = model_from_json(loaded_model_json)
    print('Loaded model from disk')
    return model

def verific_load_or_create(n_input, n_hidden, n_output, func_input, func_hidden,
                           func_output, num_model):
    arquivo = True
    f_path = path_file_models + num_model + '/'
    file = f_path + 'model_' + num_model + '.json'
    if os.path.isfile(file) == True:
        model = load_model(path_file_models, num_model)
        return model, arquivo
    else:
        model = create_model(n_input, n_hidden, n_output, 'relu', 'relu', 'softmax')
        model.summary()
        arquivo = False
        return model, arquivo

def history_model(model, arquivo, x_train, y_train, x_test, y_test, epoch, batch_s,
                  path_file_models, num_model):
    if arquivo == False:
        checkpoint = [TerminateOnBaseline(monitor='acc', baseline=0.99, patience=1,
```
```

```

 mode='max'),
tf.keras.callbacks.ModelCheckpoint (path_file_models +
'Teste' + num_model + '/' +
'epoch{epoch:02d}_acc{acc:.2f}' + '.json", monitor = 'acc',
verbose = 1,
save_best_only = True,
save_weights_only = False,
mode = 'max', save_freq = 'epoch')

]

history = model.fit(x_train, y_train, validation_data = (x_test, y_test),
callbacks = checkpoint, epochs = epoch, verbose = 1,
batch_size=batch_s, shuffle=True)

score_train, score_test = score_model(model, x_train, y_train, x_test, y_test)
return history, score_train, score_test

else:

 opt = SGD(lr = 0.01, momentum = 0.9)
 history = model.compile(loss = 'categorical_crossentropy', optimizer = opt,
 metrics = ['acc', 'mse'])
 score_tropath_file_modelsin, score_test = score_model(model, x_train, y_train,
 x_test, y_test)

 return history, score_train, score_test

def display_score(score_train, score_test):

 print(" Baseline Error Train: %.2f%%" % (100-score_train[1]*100))
 print(' Train Accuracy:', score_train[1])
 print(" Baseline Error Test: %.2f%%" % (100-score_test[1]*100))
 print(' Test Accuracy:', score_test[1])

def score_model(model, x_train, y_train, x_test, y_test):

 score_train = model.evaluate(x_train, y_train, verbose = 0)
 score_test = model.evaluate(x_test, y_test, verbose = 0)
 return score_train, score_test

```



```
model21 = model13
```

```
model22, arquivo22 = verific_load_or_create(20, 15, 7, 'relu', 'softmax', 'softmax', '22')
history22, score_train22, score_test22 = history_model(model22, arquivo22, x_train, y_train,
 x_test, y_test, epoch, batch_s,
 path_file_models, '22')
```

```
model23, arquivo23 = verific_load_or_create(20, 15, 7, 'relu', 'tanh', 'softmax', '23')
history23, score_train23, score_test23 = history_model(model23, arquivo23, x_train, y_train,
 x_test, y_test, epoch, batch_s,
 path_file_models, '23')
```

```
model24, arquivo24 = verific_load_or_create(20, 15, 7, 'relu', 'sigmoid', 'softmax', '24')
history24, score_train24, score_test24 = history_model(model24, arquivo24, x_train, y_train,
 x_test, y_test, epoch, batch_s,
 path_file_models, '24')
```

```
model31 = model21
```

```
model32, arquivo32 = verific_load_or_create(20, 15, 7, 'relu', 'relu', 'sigmoid', '32')
history32, score_train32, score_test32 = history_model(model32, arquivo32, x_train, y_train,
 x_test, y_test, epoch, batch_s,
 path_file_models, '32')
```

```
model33, arquivo33 = verific_load_or_create(20, 15, 7, 'relu', 'relu', 'tanh', '33')
history33, score_train33, score_test33 = history_model(model33, arquivo33, x_train, y_train,
 x_test, y_test, epoch, batch_s,
 path_file_models, '33')
```

```
predict = predict_model(model13) # (Or 2.1), (or 31)
```