UNIVERSIDADE FEDERAL DE OURO PRETO
ESCOLA DE MINAS
COLEGIADO DO CURSO DE ENGENHARIA DE CONTROLE
E AUTOMAÇÃO - CECAU

SANTINO MARTINS BITARÃES

PREVISÃO DE SÉRIES TEMPORAIS EM PROCESSOS DE MINERAÇÃO UTILIZANDO REDES NEURAIS RECORRENTES E SÉRIES TEMPORAIS NEBULOSAS

MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Ouro Preto, 2019

SANTINO MARTINS BITARÃES

PREVISÃO DE SÉRIES TEMPORAIS EM PROCESSOS DE MINERAÇÃO UTILIZANDO REDES NEURAIS RECORRENTES E SÉRIES TEMPORAIS NEBULOSAS

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: Prof. Dr. Thiago Antonio Melo Euzébio

Coorientadora: Profª . MSc. Adrielle de Carvalho Santana

Ouro Preto
Escola de Minas – UFOP
2019

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

B624p Bitaraes, Santino Martins .
Previsão de séries temporais em processos de mineração utilizando redes neurais recorrentes e séries temporais nebulosas. [manuscrito] / Santino Martins Bitaraes. - 2019.
64 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Thiago Antonio Euzébio.
Coorientadora: Profa. Ma. Adrielle de Carvalho Santana.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Escola de Minas.

1. Probabilidades. 2. Computação em nuvem - Predição. 3. Redes neurais recorrentes (Computação). 4. Análise de séries temporais nebulosas. I. Euzébio, Thiago Antonio. II. Santana, Adrielle de Carvalho. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 681.5

Bibliotecário(a) Responsável: Maristela Sanches Lima Mesquita - CRB:1716

Monografia defendida e aprovada, em 13 de Dezembro de 2019, pela comissão avaliadora constituída pelos professores:

Prof. Dr. Thiago Antônio Melo Euzébio – Orientador

Profª. M.Sc. Adrielle de Carvalho Santana – Coorientadora

Prof. Dr. Agnaldo José da Rocha Reis – Professor Convidado

Prof. Dr. Paulo Marcos de Barros Monteiro – Professor Convidado

Este trabalho é dedicado primeiramente a Deus, em especial a minha mãe (in memoriam), meu pai e a minha amada esposa. Dedico também ao meu orientador e minha coorientadora.

AGRADECIMENTOS

Nesta seção quero prestar meus agradecimentos a todos aqueles, que de alguma maneira contribuíram para que esse trabalho fosse realizado.

Primeiramente agradeço a Deus, porque Dele, e por Ele, e para Ele são todas as coisas (Romanos 11.6). Dedico, em especial, essa monografia a minha mãe (*in memoriam*) e meu pai, exemplos de caráter e amor, que sem medirem esforços, deixando por vezes de pensar neles mesmos, trabalharam duro para garantir que essa caminhada fosse possível.

Aos meus amigos Clebis Taborda, Leonardo Taborda e Vanessa Taborda, amigos/irmãos, que sempre estiveram ao meu lado durante toda essa jornada. Ao Gabriel e Higor Garibalde pelo conselho na escolha desse curso.

À minha amada esposa Priscila que com paciência, amor e compreensão sempre esteve ao meu lado, aos meus sogros Darci e Claudiano o carinho e acolhimento, ao Seu Ivan e Sandra por estarem sempre presentes e os demais familiares da minha esposa pelo acolhimento, à Assembleia de Deus Mariana pelos conselhos e orações.

Aos meus amigos de curso Caio Andrade, Anna Cristyna, Cleyson, Kássia, Dauberson, Irã, Guilherme Silva, Wellington, Sofia, André Cid, Maurício, Rafael Braga, Guilherme Horta, Alexandre, Wagner, Paulo Santos, Vinícius Mattos entre outros pela convivência e por tudo que passamos juntos.

Aos professores dos quais tenho orgulho de ter sido aluno: Marcos Marcial pela motivação; Monique Rafaella e Alan Kardec pela excelente didática; Paulo Monteiro por me apresentar a ciência mais fascinante de todas, a Teoria de Controle; Cocota pela oportunidade de aplicar na prática a teoria.

Ao meu orientador Thiago Euzébio e minha coorientadora profª Adrielle pelo acompanhamento, pelas correções enriquecedoras e pela atenção prestada durante todo trabalho.

Ao Vinícius Gomes pela disponibilidade dos dados para execução desse trabalho, pelos esclarecimentos e atenção dada.

Agradeço também professor Petrônio Cândido (IFNM) pelo suporte sobre sua biblioteca *pyFTS*, usada nesse trabalho e da mesma forma a ele e o professor Frederico Gadelha (UFMG) pelo excelente curso ministrado durante o SBAI 2019 que motivou o uso da biblioteca *pyFTS*.

Como disse Guimarães Rosa “é junto dos bão que a gente fica mió”.

"Todo que ama a disciplina ama o conhecimento, mas aquele odeia a repreensão é tolo."
(Provérbios 12.1)

RESUMO

Na indústria mineral, a previsão de variáveis de interesse ajudam na tomada de decisões para o ajuste contínuo de processos. O percentual de ferro e sílica afetam diretamente o desempenho do circuito de separação magnética de finos. Em mina de Fábrica, em Minas Gerais, a análise laboratorial do percentual de ferro e sílica possuem atraso médio de 2 horas para o resultado. Isso afeta rápidas tomadas de decisão em relação ao processo que gera a polpa de alimentação do separador magnético. Com base nesse problema, são propostos dois modelos de previsor de um passo a frente baseados nas últimas análises de laboratório fornecidas pela mina, um baseado em redes neurais recorrentes e outro em série temporal nebulosa. O modelo com rede neural recorrente é baseado no ajuste de pesos em células de memória interligadas com outras células e com ela mesma. Já o modelo de séries temporais nebulosas é baseado na extração do conhecimento da série temporal por meio de lógica *fuzzy*. Os métodos foram avaliados pelas métricas de erro quadrático médio quadrático, erro médio ponderado e a estatística U de *Theil*. O melhor resultado foi apresentado pela série temporal nebulosa com percentual de erro de apenas 1,70%. O uso de um modelo como esse possibilitaria uma rápida tomada de decisão, otimizando o processo produtivo

Palavras-chaves: Série Temporal . Predição. Redes Neurais Recorrentes. Séries Temporais Nebulosas.

ABSTRACT

In the mineral industry, the forecast of variables of interest helps in the decision making for the continuous adjustment of processes. The percentage of iron and silica directly affects the performance of the magnetic separation circuit of fines. At the Minas Gerais mill, the laboratory analysis of the iron and silica percentage is, on average, 2 hours late for the result. This affects rapid decision making regarding the process that generates the magnetic separator feed pulp. Based on this problem, two predictive one-step forward models are proposed based on the latest laboratory analysis provided by the mine, one based on recurring neural networks and the other on a cloudy time series. The model with recurrent neural nets is based on the adjustment of weights in memory cells interconnected with other cells and with itself. On the other hand, the nebulous time series model is based on the extraction of the knowledge of the time series by means of fuzzy logic. The methods were evaluated by metrics of quadratic mean square error, weighted mean error, and Theil's U statistics. The best result was presented by the cloudy time series with an error percentage of only 1.70%. The use of such a model would enable quick decision making, optimizing the production process.

Key-words: Time Series. Prediction. Recurrent Neural Networks. Fuzzy Time Series

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RNA	Redes Neurais Artificiais
RNN	Recurrent Neural Network
MARS	Multivariate Adaptative Regression Splines
GRU	Neurais Gated Recurrent Unit
LSMT	Long Short Term Memory
$\%Fe$	Percentual de Ferro
WHIMS	Wet High Intensity Magnetic Separators
COFI	Concentração de Finos
$\%SiO_2$	Percentual de Sílica
EEG	Eletroencefalograma
ACF	Função de Autocorrelação
RNR	Redes Neurais Recorrentes
FTS	Fuzzy Time Series
RMSE	Raiz Média do Erro ao Quadrado
MAPE	Erro Ponderado Médio
U	Coeficiente U de Theil
MG	Minas Gerais

LISTA DE SÍMBOLOS

b_k	Bias
w_k	Pesos
$\sum$	Somatório
u_k	Saída do combinador linear
$\varphi(\cdot)$	Função de ativação

SUMÁRIO

1 INTRODUÇÃO

Na indústria mineral, previsões de variáveis de interesse ajudam na tomada de decisões para o ajuste contínuo dos processos. No circuito de separação magnética de finos da concentração de finos (**COFI**), em Mina de Fábrica, não há dados em tempo real da composição mineral da polpa alimentada no circuito. Isso afeta diretamente a eficiência do circuito no geral. Analisadores em linha como *Blue Cube MQi* e *Outotec Courier 8 SL* são caros e ainda não estão bem estabelecidos na literatura. Atualmente, o desempenho real de muitas instalações minerais somente é conhecido após a conclusão dos ensaios feitos em amostras nos laboratórios.

A densidade, a porcentagem de ferro ($\%Fe$) e de sílica ($\%SiO_2$) da polpa de alimentação do separador magnético é obtida a partir de amostras da polpa, logo na entrada do circuito. A densidade da polpa é medida em tempo real por meio de um densímetro, mas a $\%Fe$ e a $\%SiO_2$ são provenientes de análises laboratorial química com um atraso médio de 2 horas. Durante o intervalo de espera do resultado laboratorial, o último valor obtido em laboratório é usado no ajuste do circuito. Uma previsão confiável das medidas de %Fe e $\%SiO_2$ ao longo do tempo possibilitaria a tomada de decisões otimizadas sobre o processo.

A previsão de séries temporais envolve a adaptação de um modelo a um conjunto de dados e a utilização desse modelo para a predição de dados não utilizados previamente no treinamento. Uma série temporal é um conjunto de observações feitas sequencialmente ao longo do tempo. Um modelo de séries temporais fornece previsões de observações futuras por meio da relação que possui com valores do passado. As redes neurais e a série temporal nebulosa são exemplos de técnicas que podem ser aplicadas para a realização da previsão de séries temporais. (CHATFIELD, 2000).

Redes Neurais Artificiais (RNA) são modelos matemáticos inspirados no funcionamento de neurônios biológicos. Basicamente uma rede neural é composta por neurônios interconectados e essas interconexões são quantificadas com os pesos da conexões. Uma rede neural é um processador distribuído massivamente paralelo, composto por unidades de processamento simples, que tem uma propensão natural para armazenar conhecimento experiencial e torná-lo disponível para uso (HAYKIN, 2007).

Por serem não lineares por natureza, as redes neurais se tornam atrativas para tarefas de previsão em relação a métodos tradicionais de series temporais (HAYKIN, 2007). Redes neurais são métodos autoadaptáveis baseados em dados. Elas aprendem a partir de exemplos e incorporam relações funcionais sutis entre os dados, sem a necessidade de especificar a forma funcional que os relacionam (ZHANG; PATUWO; HU, 1998).

A série temporal nebulosa é um método de modelagem de séries temporais. O modelo é baseado em regras *fuzzy* e, por isso, é mais interpretável que as RNA. As regras *fuzzy* descrevem

a base de conhecimento do modelo. Assim a série temporal nebulosa se apresenta como outra ferramenta para previsões (SILVA, 2019).

1.1 Estado da Arte

É crescente a aplicação de técnicas de aprendizagem de máquina na indústria mínero-metalúrgica. O trabalho de McCoy e Auret (2019) visa dotar os investigadores e profissionais da indústria de conhecimentos estruturados sobre o estado da aplicação de tais técnicas no processamento de minerais. Seu trabalho fornece um material suplementar com todas as técnicas revistas, com domínios que incluem a natureza dos dados de estudos de casos (sintéticos/laboratórios/industriais), nível de sucesso, área de aplicação (por exemplo: moagem, flotação, etc.) e categoria do problema principal (modelização baseada em dados, detecção e diagnóstico de falhas e visão mecânica). A Tabela 1 mostra um resumo da quantidade de trabalhos revistos pelo autor.

Tabela 1 – Aplicações de aprendizado de máquina na mineração

Processos Aplicados	Modelagem baseada em dados	Detecção de Falhas e Diagnósticos	Machine Vision	Total
Flotação	16	8	71	95
Granulometria	2	0	26	28
Moagem	15	7	3	25
Concentração	2	6	3	11
Fornos / Fundição	12	8	0	20
Hidrometalurgia	1	3	0	4
Outros	5	1	4	10

Fonte: (MCCOY; AURET, 2019)

Na literatura é possível encontrar trabalhos que aplicam as redes neurais e a série temporal nebulosa em diferentes problemas de predição de séries temporais com sucesso, mostrando o potencial dessas técnicas para aplicações industriais.

Coulibaly e Baldwin (2005) apresentam um estudo da eficácia das redes neurais recorrentes (RNN) para a modelagem de sistemas complexos de recursos hídricos variáveis no tempo. Seu trabalho é mostrado que o modelo RNN dinamicamente conduzido pode ser uma boa alternativa para a modelagem de dinâmicas complexas de um sistema hidrológico, com melhor desempenho do que o modelo MARS (*Multivariate Adaptative Regression Splines*) nas três séries temporais hidrológicas selecionadas, ou seja, os volumes históricos de armazenamento do Grande Lago Salgado, das vazões do Rio Saint-Lawrence e das vazões do Rio Nilo.

Já Gomes et al. (2017) apresentam em seu trabalho várias arquiteturas de redes neurais recorrentes, alternando topologias com estruturas GRU e LSTM. As redes são avaliadas na tarefa de prever o nível de atividade de abelhas com base nos valores dos níveis passados. Também é

mostrado como as RNNs podem melhorar sua precisão ao avaliar como diferentes janelas de tempo de entrada afetam os resultados.

Em relação às séries temporais nebulosas, Silva (2019) expõe métodos de séries temporais nebulosas com crescente expansão nos últimos anos. Seu trabalho enfatiza a facilidade de implementação dos métodos, o seu baixo custo computacional e a interpretabilidade de seus modelos. Afirma, ainda, que os métodos de séries temporais nebulosas têm sido utilizados em áreas como previsão de demanda energética, indicadores e ativos de mercado, turismo e outras.

Uma aplicação de um desses métodos pode ser vista no trabalho de Chen (1996) em que as séries temporais nebulosas são utilizadas na predição de matrículas em universidades. O autor dados de matrículas usa de dados históricos da Universidade do Alabama. Finalmente conclui que o método proposto não só pode fazer boas previsões das matrículas nas universidades, como também pode fazer previsões robustas quando os dados históricos não são precisos. O resultado do trabalho é apresentado na Figura 1

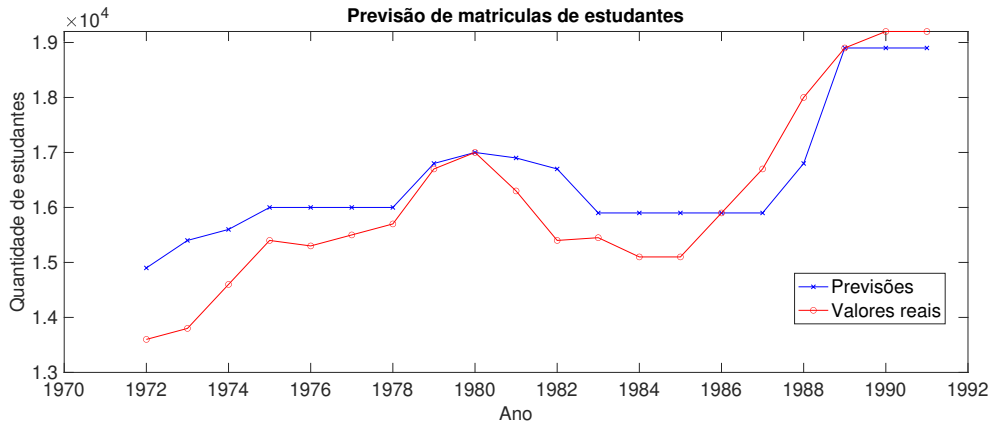

Figura 1 – Previsão de matriculas

Fonte: (CHEN, 1996)

Em um trabalho mais recente, Sun et al. (2015) fazem a previsão dos preços futuros do índice chinês de ações usando conjuntos *fuzzy* e o método de séries temporais multivariadas *fuzzy*. Eles descrevem em seu trabalho que o modelo de séries temporais pode ser utilizado para resolver os dados difusos nos preços dos futuros do índice de ações. Em seu trabalho estabelecem um modelo multivariado, com melhoras na precisão do cálculo.

Além dessas aplicações baseadas no conjunto nebuloso, a lógica *fuzzy* é comumente aplicada em projeto de controladores como apresentados por Júnior et al. (2018), onde um controlador Fuzzy multivariável foi projetado para uma planta de desaguamento de minério de ferro da VALE S.A em Serra Norte de Carajás e no trabalho de Júnior, Bitarães e Euzébio (2019) onde é apresentado um projeto de um controle *fuzzy* para secagem rotativa na unidade de produção de fertilizantes da empresa Yara em Paulínea/SP, com objetivo de reduzir o consumo de energia do processo sem perder a qualidade do produto final.

1.2 Justificativa

Neste trabalho propõe-se investigar a previsibilidade da porcentagem de ferro na polpa de alimentação do separador magnético aplicando dois métodos de previsão, um baseado em redes neurais e outro baseado lógica nebulosa. Espera-se que, com base nas últimas amostras da alimentação fornecidas pelo laboratório, a rede modelada possa contribuir com uma faixa em que esses teores irão apresentar-se no instante da análise atual, possibilitando tomadas de decisão otimizadas sobre o processo.

1.3 Objetivos geral e específicos

1.3.1 Objetivo geral

Projetar um preditor capaz de realizar prever a porcentagem de ferro da polpa de alimentação de um separador magnético a fim de influenciar em tempo real tomadas de decisão sobre o processo.

1.3.2 Objetivos específicos

- Obter o modelo de rede neural recorrente e da série temporal nebulosa utilizando dados históricos fornecidos pela mineradora.
- Identificar as características do melhor previsor para aplicação nesse tipo de problema.
- Realizar previsões dos teores de ferro e sílica da polpa e comparar as predições obtidas com as mais recentes fornecidas pelo laboratório.

1.4 Estrutura do trabalho

O trabalho esta organizado em 5 Capítulos e 2 Anexos. Após a introdução feita Capítulo 1, é realizado uma revisão dos conceitos teóricos no Capítulo 2. No Capítulo 3 são apresentados a base de dados usada nesse trabalho e o desenvolvimento das aplicações da rede neural recorrente e da série temporal nebulosa na previsão de um passo a frente. No Capítulo 4, os resultados obtidos no Capítulo 3 são discutidos. No Capítulo 5 apresenta um resumo com os resultados com cada técnica e a comparação entre elas. Por fim são apresentados dois Anexos. No Anexo A a base de dados é apresentada em uma tabela e no Anexo B, os códigos fonte dos métodos são apresentados.

2 CONTEXTO TEÓRICO

Nesse capítulo será apresentada uma revisão dos conceitos teóricos que serão utilizados como base para o desenvolvimento desse trabalho. Na seção 2.1 será apresentado o processo de separação magnética com suas respectivas características e aplicações, na seção 2.2 será apresentado conceitos sobre séries temporais e por último, na seção 2.3 conceitos sobre redes neurais artificiais são expostos.

2.1 Circuito de separação magnética

A maioria dos minérios é encontrada na natureza com teores minerais economicamente inviáveis para a metalurgia. Um minério pode conter vários minerais em sua composição e, na maioria das vezes, um mineral dentre todos é o de maior interesse econômico o mineral útil.

A redução do tamanho das partículas, separação em classes de tamanhos e a elevação de teores de elementos úteis (concentração) são etapas necessárias para a metalurgia. A fragmentação do material até as partículas alcancem um tamanho específico permite a liberação do mineral útil. A elevação do teor de concentração de um minério de ferro é por muitas vezes feita por um separador magnético (CHAVES, 2002).

O princípio básico da separação é a diferença das susceptibilidades magnéticas das partículas. A separação magnética é uma separação física de partículas baseada nas iterações entre forças magnéticas de tração, gravitacional, (de atrito ou inercial) e forças interpartículas atrativas ou repulsivas. Estas forças combinam-se para atuar diferentemente sobre partículas de diferentes propriedades magnéticas no material de alimentação (OBERTEUFFER, 1974). A Figura 2 representa os principais elementos do funcionamento de um separador magnético.

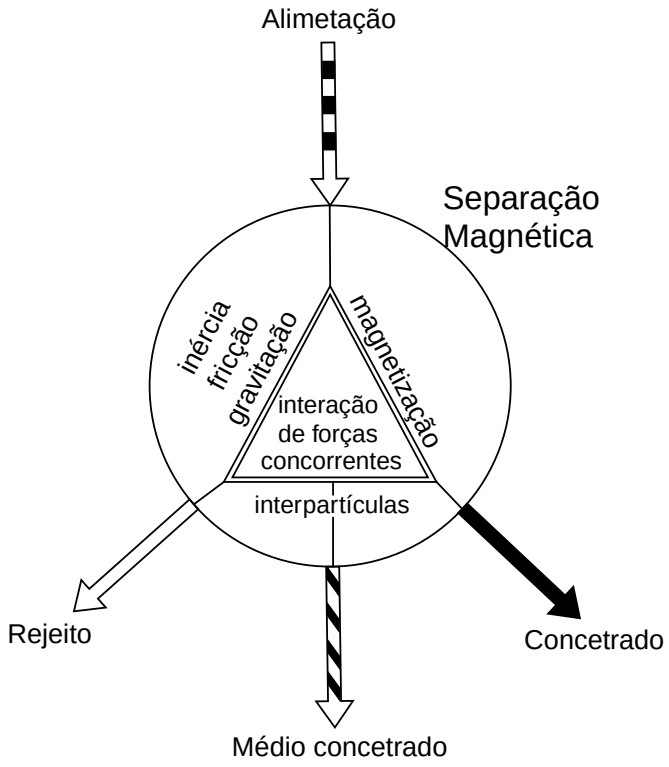

Figura 2 – Representação esquemática de um separador magnético

Fonte: (OBERTEUFFER, 1974)

O separador magnético WHIMS é composto por um sistema de tração acoplado a um eixo com discos que possuem matrizes magnéticas nas extremidades. O processo se inicia com a alimentação da polpa nas matrizes onde atravessa uma região de campo magnético de alta intensidade. As partículas com características magnéticas ficam retidas entre as ranhuras das matrizes e as não magnéticas atravessam diretamente as matrizes, sendo enviadas para o rejeito. Com a rotação do disco, as matrizes chegam a uma região com jatos fixos ajustados para remoção de partículas de baixa característica magnética. Na sequência, essa matrizes passam por uma região com jatos de spray com pressão superior para remoção do material com maior característica magnética (concentrado) (GOMES, 2019), (WASMUTH; UNKELBACH, 1991). A Figura 3 apresenta as etapas descritas.

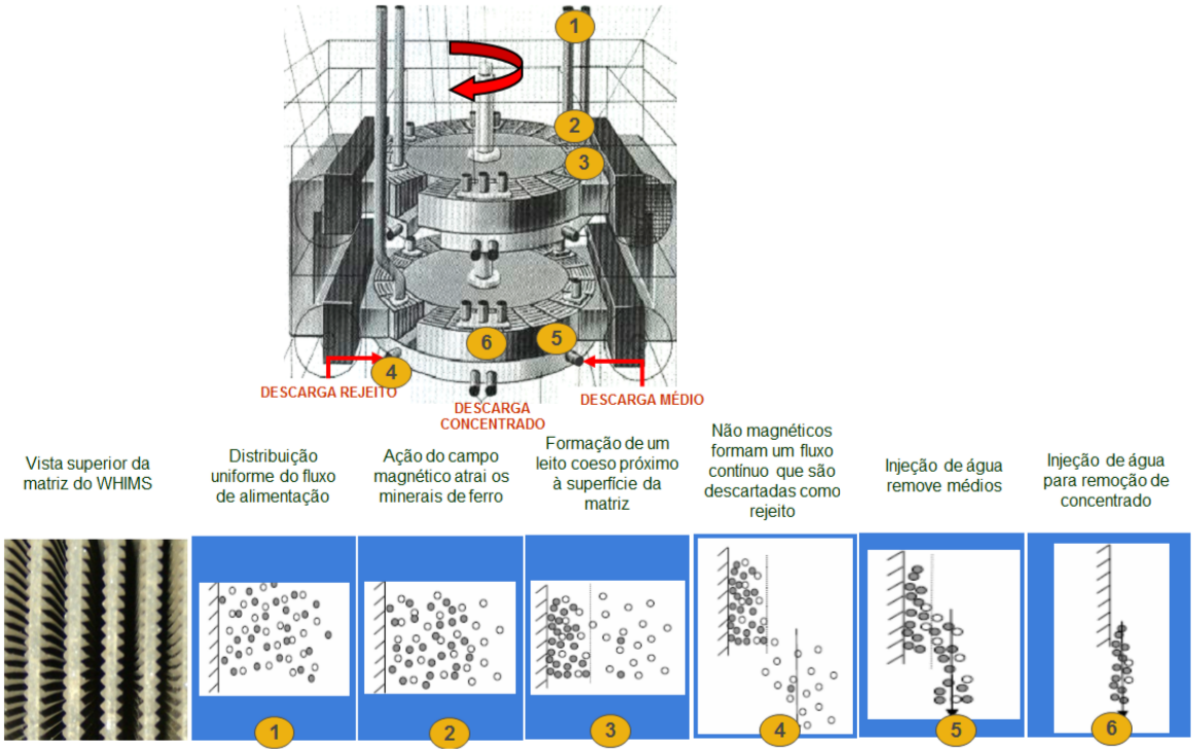

Figura 3 – Esquema sequencial de funcionamento do separador WHIMS

Fonte: (GOMES, 2019)

A eficiência do circuito de separação magnética é afetada diretamente por diversas variáveis. No caso apresentado por Gomes (2019), o circuito de separação magnética de finos do COFI, em Mina Fábrica, as variáveis que podem ser manipuladas são: taxa de alimentação, intensidade do campo magnético, velocidade de rotação dos discos, pressão da água na lavagem de médio concentrado. O número de estágios do circuito, o destino de fluxo dos médios e a abertura entre as placas ranhuradas das matrizes magnética são variáveis que não são possíveis de serem alteradas. A densidade da alimentação, a $\%Fe$ e a $\%SiO_2$ na alimentação são consideradas como distúrbio no sistema.

2.2 Séries Temporais

Para Brockwell e Davis (2016) e Chatfield (2000) uma série temporal é uma coleção de observações, cada uma das quais registradas sequencialmente ao longo do tempo. As medições podem ser feitas continuamente ao longo do tempo ou num conjunto discreto de pontos temporais. Por convenção, estes dois tipos de séries são chamados de séries temporais contínuas e discretas, respectivamente, embora a variável medida possa ser discreta ou contínua em ambos os casos. Em outras palavras, para séries temporais discretas, por exemplo, é o eixo temporal que é discreto.

Um série temporal discreta é aquela em que as observações são feitas a intervalos de

tempo fixos. Exemplos incluem:“ *(i)* vendas de um determinado produto em meses sucessivos, *(ii)* a temperatura num determinado local ao meio-dia em dias sucessivos e *(iii)* o consumo de eletricidade numa determinada área durante períodos sucessivos de uma hora” (CHATFIELD, 2000).

Em uma série temporal contínua, a variável observada é tipicamente uma variável contínua registrada continuamente em um traço, como uma medida da atividade cerebral registrada de uma máquina EEG. O método usual de analisar tal série é amostrar-la (ou digitalizar) a série em intervalos de tempo iguais para dar uma série cronológica discreta. Pouca ou nenhuma informação é perdida por este processo, desde que o intervalo de amostragem seja suficientemente pequeno, pelo que não há necessidade de dizer mais nada sobre as séries contínuas. (CHATFIELD, 2000)

2.2.1 Análise de dados de uma série temporal

A característica especial dos dados de séries temporais é que as observações sucessivas normalmente não são independentes, pelo que a análise deve ter em conta a ordem pela qual as observações são recolhidas.

Para Chatfield (2000), a análise de séries temporais objetiva principalmente:“ *(i)* descrever os dados utilizando estatísticas resumidas e/ou métodos gráficos, *(ii)* encontrar um modelo estatístico adequado para descrever o processo de geração de dados, no caso de um modelo monovariável para uma determinada variável baseia-se em apenas os valores do passado dessa variável, *(iii)* estimar valores futuros da série, em que se espera que o futuro seja semelhante ao passado, *(iiii)* previsões para um analista tomar medidas para controlar um determinado processo, quer seja um processo industrial ou de economia. ”

Antes de tentar modelar e prever uma determinada série cronológica, é desejável ter uma visão preliminar dos dados para ter uma ideia das suas principais propriedades. O gráfico de tempo é a ferramenta mais importante, mas outros gráficos e estatísticas resumidas também podem ajudar. No processo, o analista também será capaz de ‘limpar’ os dados removendo ou ajustando quaisquer erros óbvios. Toda esta abordagem é por vezes descrita como Análise Inicial de Dados (ou IDA).

Para Brockwell e Davis (2016), antes de introduzir o conceito de dependência é necessário traçar a série em um gráfico, verificando em particular se há uma tendência, uma componente sazonal, mudanças aparentes e acentuadas no comportamento e quaisquer observações exteriores.

Para avaliação do grau de dependência dos dados, uma ferramenta importante é a função de autocorrelação da amostra. Esta estimativa pode sugerir qual dos muitos modelos de séries temporais estacionárias possíveis é um candidato adequado para representar a dependência nos dados. Por exemplo, uma amostra ACF que está próxima de zero para todos os atrasos não nulos sugere que um modelo apropriado para os dados pode ser o ruído.

2.2.2 Previsão de séries temporais

A previsão de séries temporais é uma área importante de previsão em que as observações passadas da mesma variável são coletadas e analisadas para desenvolver um modelo que descreva a relação subjacente. O modelo é então utilizado para extrapolar a série cronológica para o futuro. Esta abordagem de modelação é particularmente útil quando há pouco conhecimento disponível sobre o processo de geração de dados subjacentes ou quando não existe um modelo explicativo satisfatório que relacione a variável de previsão com outras variáveis explicativas (ZHANG, 2003).

2.3 Redes Neurais Artificiais

De acordo com Haykin (2007), as redes neurais são modelos matemáticos inspirados no funcionamento dos neurônios biológicos. O trabalho de Cajal e Azoulay (1952) introduziu a ideia dos neurônios como os constituintes estruturais do cérebro humano. Um neurônio continuamente recebe informações, entende isso e cria decisões apropriadas conforme representado na Figura 4.

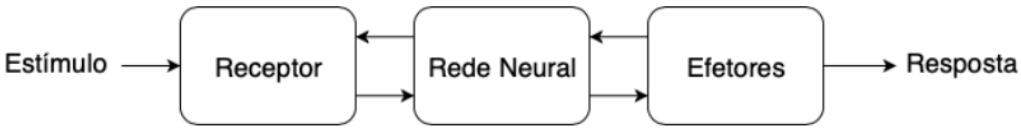

Figura 4 – Representação em diagrama de blocos do sistema nervoso

Fonte: (HAYKIN, 2007)

Um neurônio biológico é composto pelo(s):“ *(i)* axônio que consta de um tubo longo e fino que se divide em bulbos que quase tocam os dendritos dos outros neurônios, *(ii)* dendritos que são ramificações arbóreas que formam uma malha de filamentos finíssimos ao redor do neurônio, *(iii)* corpo celular (ou soma) é o centro metabólico, em conjunto com os dendritos é o local de recepção dos estímulos” (HAYKIN, 2007).

A Figura 5 apresenta uma representação da interação entre dois neurônios. O pequeno espaço entre o fim do bulbo e o dendrito é conhecido como sinapse, que tem como papel fundamental a memorização da informação. Os sinapses são estruturas elementares e unidades funcionais que mediam a interação entre os neurônios. Os synapses mais comuns são os synapse químicos, que operam da seguinte forma: o processo de synapse converte um sinal elétrico pré sináptico em um sinal químico e volta para um sinal elétrico pós sináptico. De acordo com o resultado dessa reação química o sinal elétrico pós sináptico pode aumentar ou diminuir o potencial elétrico do corpo celular assim excitando ou inibindo o neurônio receptor.

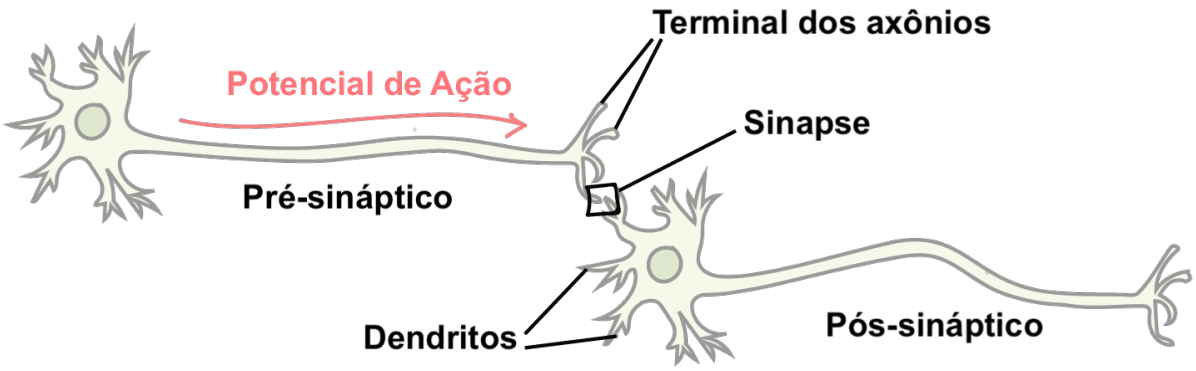

Figura 5 – Neurônio Biológico

Fonte: (HAYKIN, 2007) adaptado

2.3.1 Neurônio Artificial

O modelo artificial de neurônio, ilustrado na Figura 6, é uma alternativa que tenta imitar a operação do neurônio biológico. As entradas são os sinais elétricos recebidos pelos axônios de outros neurônios. Os pesos equivalem ao processo de synapse que pode ampliar ou atenuar o sinal antes de ser somado aos outros no corpo celular. O resultado dessa soma de sinais faz com que exista ou não energia suficiente para o neurônio ser ativado e emitir um sinal de saída pelo seu axônio. A função de ativação modela a forma como o neurônio responde ao nível de excitação, limitando e definindo a saída da rede neural (HAYKIN, 2007).

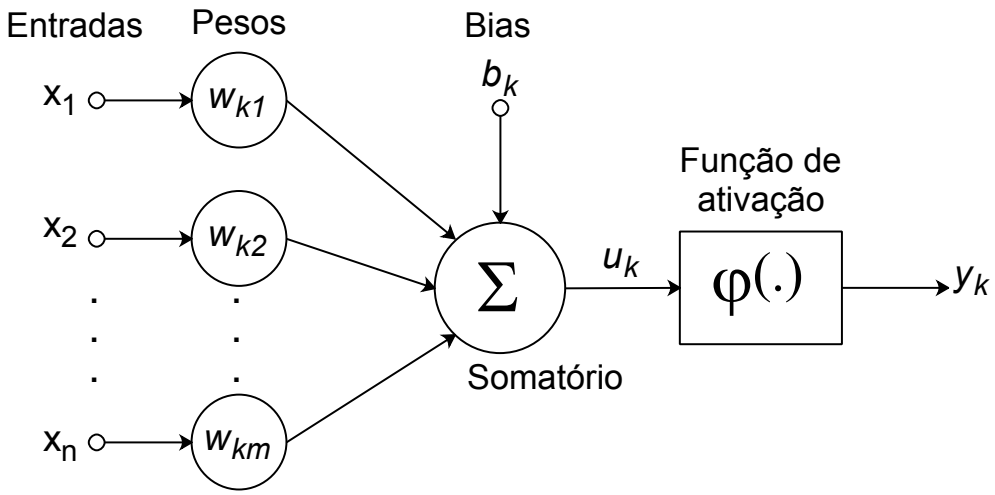

Figura 6 – Neurônio Artificial

Fonte: (HAYKIN, 2007) adaptado

O modelo de neurônio artificial da Figura 6 apresenta também um *bias* aplicado externamente, denotado por b_k. O *bias* b_k tem o efeito de aumentar ou diminuir a entrada da função de

ativação, dependendo se ele é positivo ou negativo.

Em termos matemáticos, um neurônio k pode ser descrito com as seguintes equações

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

e

$$y_k = \varphi(u_k + b_k) \tag{2.2}$$

onde x_1, x_2, ..., x_m são os valores de entrada; w_{k1}, w_{k2}, ..., w_{km} são os pesos do neurônio k; u_k é a saída do combinador linear devido aos sinais de entrada; b_k é o *bias*; $\varphi(\cdot)$ é a função de ativação; e y_k é o valor de saída do neurônio.

Uma rede neural é organizada em camadas. Cada camada contem um ou mais neurônios com saídas combinadas e conectadas com neurônios da próxima camada. A arquitetura mais comum é a *feedforward*, Figura 7, onde as redes possuem fluxo de informação unidirecional, não havendo nenhum ciclo. Uma rede neural deste tipo é uma função de sua entrada, não possuindo nenhum tipo de estado interno além dos pesos em si.

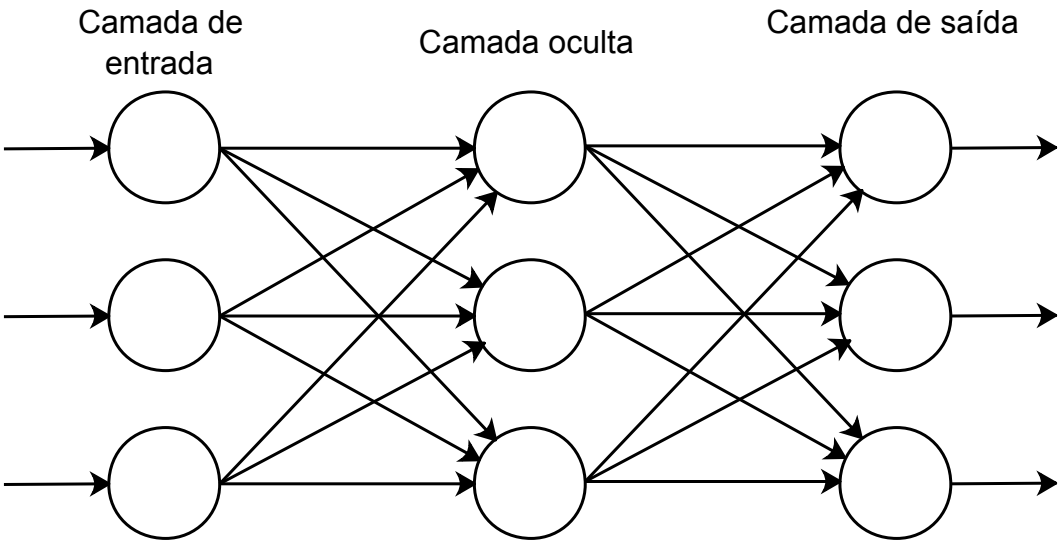

Figura 7 – Arquitetura *feedforward*

Fonte: próprio autor

O algoritmo de treinamento *back-propagation* é o mais popular, seu objetivo é otimizar os pesos da rede de forma a encontrar uma ligação entre as entradas e saídas fornecidos. A Figura 8 apresenta o funcionamento desse algoritmo. Nele as entradas são distribuídas como sinais para os neurônios que inicialmente são amplificados ou atenuados de acordo com pesos aleatórios. A saída da rede é comparada com as saídas conhecidas da rede gerando um erro. Esse erro é

enviado para um algoritmo de aprendizagem que por meio de outro algoritmo de otimização realiza os ajustes dos pesos. Esse processo se repete até que o erro seja o mínimo possível.

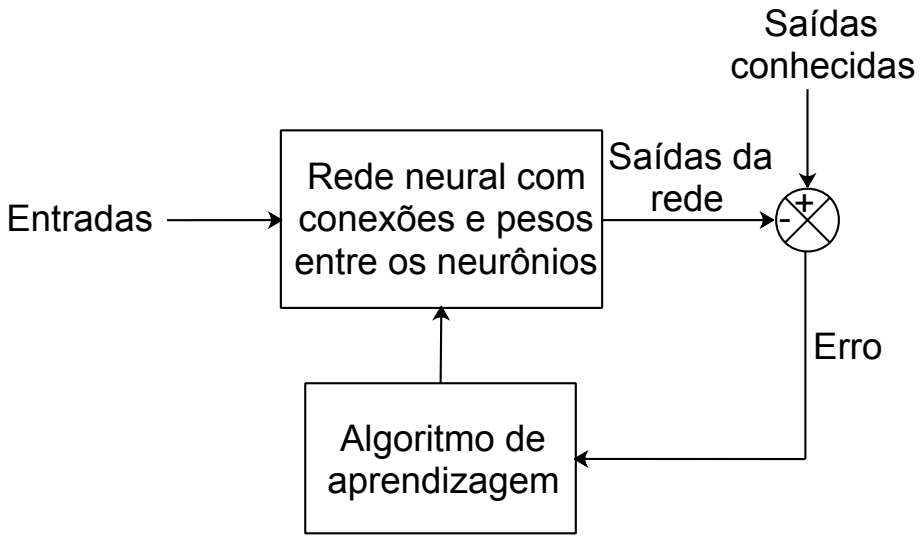

Figura 8 – Processo de treinamento de uma rede neural

Fonte: próprio autor

Haykin (2007) apresenta as vantagens e capacidades de uma rede neural:“ *(i)* os neurônios podem ser não lineares ou lineares dado uma característica especial de não linearidade distribuída a rede, *(ii)* mapeamento da entrada-saída consiste na rede ao receber um exemplos de entradas, que possuem únicas saídas, realizar ajustes nos pesos até que sejam mínimos ajustes fazendo com que a rede consiga fazer uma relação genérica entre as entradas e saídas. ”

2.4 Redes Neurais Recorrentes

As redes neurais recorrentes (RNRs) são redes neurais *feedforward* aumentadas pela inclusão de ciclos entre seus neurônios. Os neurônios podem ter conexões com neurônios de camadas anteriores, ou da mesma camada, ou com eles mesmos. Assim, as informações não fluem em um único sentido. Dessa forma, a saída não depende apenas da entrada atual, mas também das anteriores, portanto introduzindo uma noção de memória para o modelo (LIPTON; BERKOWITZ; ELKAN, 2015). Uma rede neural recorrente simples é apresentada na Figura 9

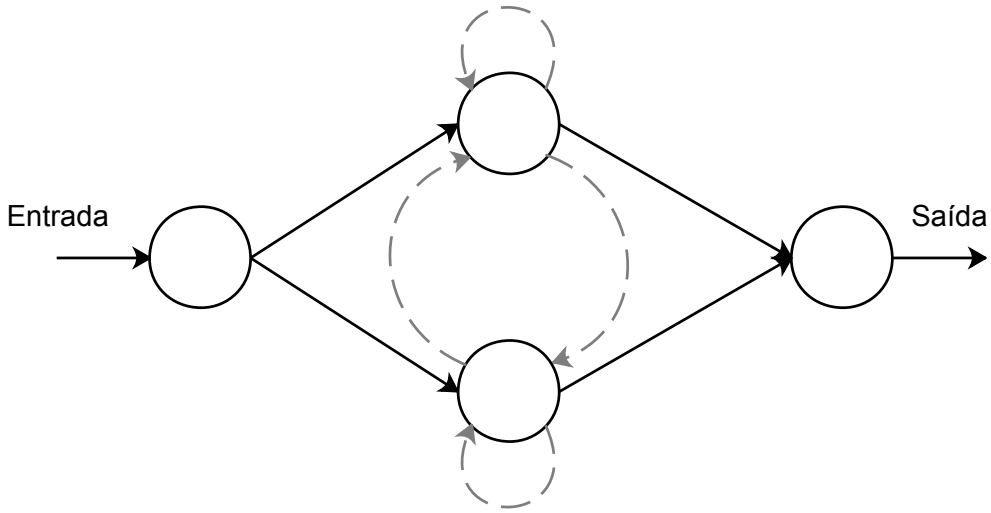

Figura 9 – Rede neural recorrente simples

Fonte: próprio autor

Na rede recorrente mostrada na Figura 9, a cada passo de tempo os sinais são passados ao longo dos caminhos sólidos como em uma rede *feedforward*. Os caminhos tracejados conectam o neurônio de origem em cada passo com o neurônio de destino no próximo passo. Os neurônios com conexões recorrentes recebem entrada de dados atual e também de valores de neurônios do estado anterior da rede. A entrada anterior pode influenciar a saída e posteriormente por meio das conexões recorrentes.

2.4.1 Redes *Long-Short Term Memory* (LSTM)

As redes LSTM é um modelo recorrente e profundo de redes neurais que foi introduzida por Schmidhuber e Hochreiter (1997). O modelo assemelha-se a uma rede neural recorrente padrão com uma camada oculta, mas cada neurônio comum na camada oculta é substituído por uma célula de memória (Figura 10). Cada célula de memória contém um neurônio autoconectado recorrente de peso fixo um, e mecanismos de liberar memória, adicionar memória e ler memória.

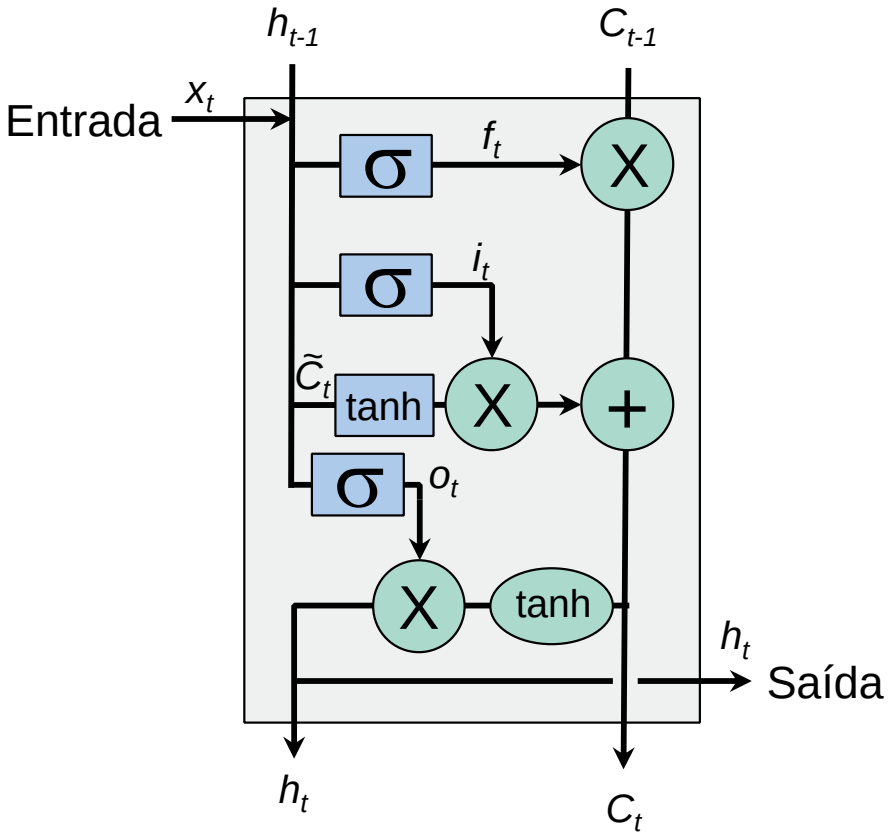

Figura 10 – Célula de memória de uma rede LSTM

Fonte: (LIPTON; BERKOWITZ; ELKAN, 2015) adaptado

O primeiro passo decide qual dado vai ser apagado, os sinais de x_t e h_{t-1} passam pela função sigmoide que se retornar com valor 0 o dado será apagado e, com 1, não será apagado da memória. O próximo passo é decidir quais novas informações vão ser armazenadas no memoria. Isso tem duas partes: Primeiro, uma camada sigmóide decide quais valores vai ser atualizado. Em seguida, uma camada $tanh$ cria um vetor de novos valores candidatos, $\tilde{C}_t$ que poderia ser adicionado ao estado. No próximo passo, combina-se estes dois para criar uma atualização para o estado.

Nesse passo, o estado da célula C_{t-1} é atualizado para C_t. Os passos anteriores já decidiram o que apagar e o que armazenar, esse passo somente efetiva a ação. Finalmente um sigmoide e a tangente hiperbólica obtém o que realmente é importante para propagar para os próximos neurônios. Isso funciona com uma espécie de filtro.

2.5 Séries Temporais Nebulosas

A séries temporais nebulosas (FTS) foram baseadas nos trabalhos de Zadeh (1975), por sua vez é um método de modelagem de séries temporais. Trata-se de um processo dinâmico

especial de valores linguísticos à medida que suas observações são definidas e estudadas. Para descrever séries temporais nebulosas, equações relacionais nebulosas são empregadas como modelos. (SONG; CHISSOM, 1993b).

A primeira metodologia FTS foi proposta por Song e Chissom (1993a). Genericamente a metodologia de um série temporal nebulosa pode ser divida em duas etapas: o treinamento e a previsão. O método de treinamento tem como objetivo criar a variável linguística A e fazer uma representação do conhecimento da dinâmica da série temporal. A previsão envolve as etapas de: *fuzzificação* do valor numérico de entrada no instante t, encontrar regras nebulosas associadas a esse valor *fuzzy* realiza a *defuzificação* e obter o valor numérico da saída para $t + 1$.

2.5.1 Treinamento e previsão

Silva (2019) descreve o processo de treinamento de uma série temporal nebulosa. Primeiro os dados devem passar por um pré-processamento para reduzir ruídos, dessazonalizar ou normalizar os dados. O processo mais importante do treinamento é a partição dos dados. Esse processo é responsável por dividir o intervalo dos dados de entrada em n conjuntos *fuzzy*. Conforme pode ser observado na Figura 11, cada conjunto *fuzzy* A possui sua própria função de pertinência. Comumente as funções de pertinência são funções triangulares, trapezoidais, sigmóides ou gaussianas. Assim criando variáveis linguísticas para descrever y.

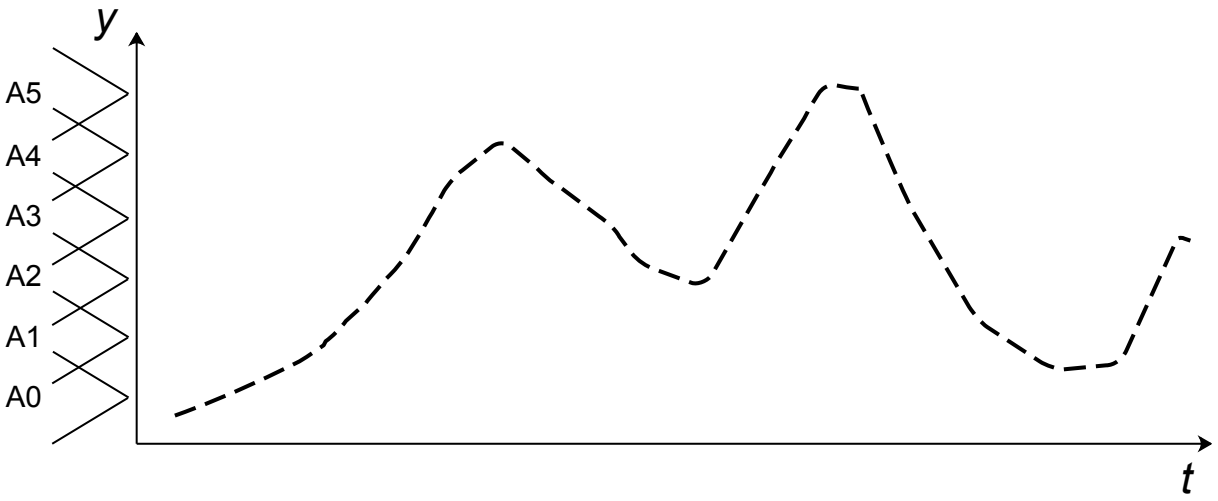

Figura 11 – Particionamento dos dados de entrada

Fonte: (SILVA, 2019) adaptado

O processo de transformar as amostras dos valores numéricos da série em seus conjuntos *fuzzy* correspondentes é chamado de *fuzzyficação* da série. Este processo visa transformar a série temporal numérica nítida y numa série temporale linguística F, também conhecida como série temporal $fuzzy$. No caso mais simples da literatura mostrado na Figura 12, cada valor numérico amostrado é associado ao conjunto *fuzzy* de maior pertinência. Em outros casos, como abordado

por Silva (2019), as amostras numéricas são associadas usando um vetor de pertinências a cada conjunto *fuzzy*.

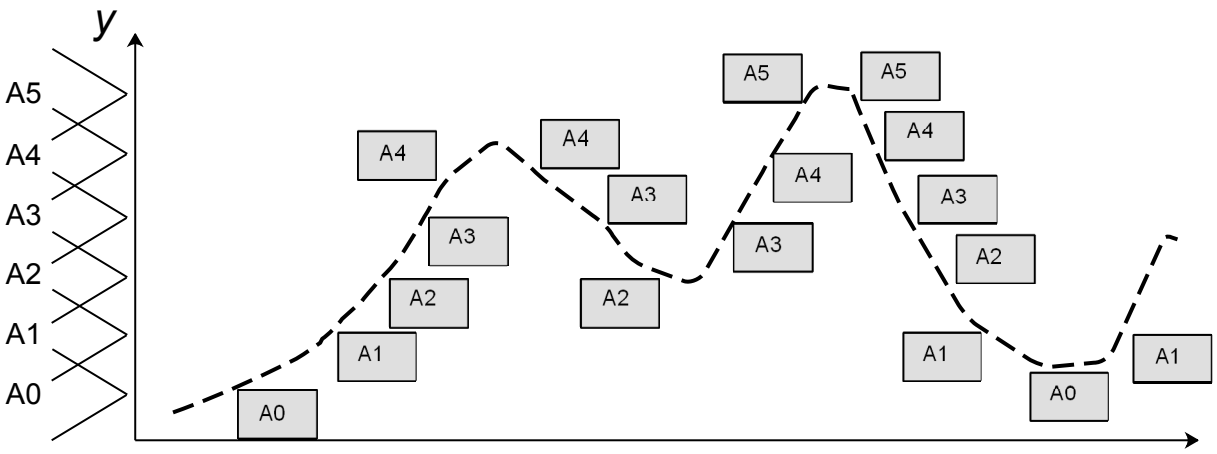

Figura 12 – Fuzzificação

Fonte: (SILVA, 2019) adaptado

O segundo passo mais importante é a extração e representação do conhecimento. A partir da sequência de conjuntos *fuzzy* o método vai tentar identificar regras que modelam a série temporal. Esse processo cria um modelo de conhecimento realizando um reconhecimento de padrões na serie temporal $fuzzy$. Na primeira parte observa-se nos dados históricos da série F o padrão em que ocorre as sequencias de valores de F. Algumas sequências observadas na Figura 12 são apresentadas na Figura 13.

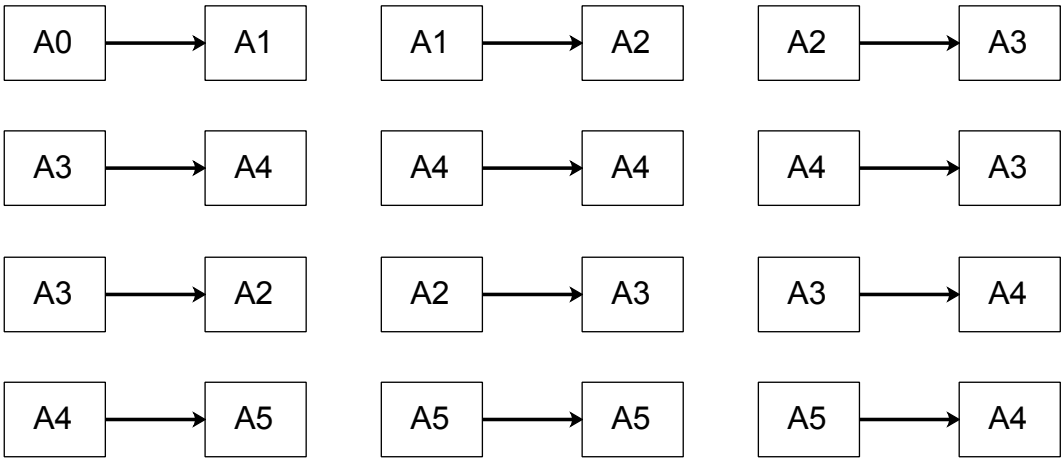

Figura 13 – Relação lógica *fuzzy*

Fonte: (SILVA, 2019) adaptado

Na segunda etapa da extração do conhecimento os padrões que contêm o mesmo antecedente são agrupados. A agrupamento com os padrões são usados para definir regras de *fuzzy*. Os

modelos variam de acordo com a maneira que se organiza esse padrões nas regras de previsão. Os padrões da Figura 13 podem ser agrupados da seguinte forma na Figura 14.

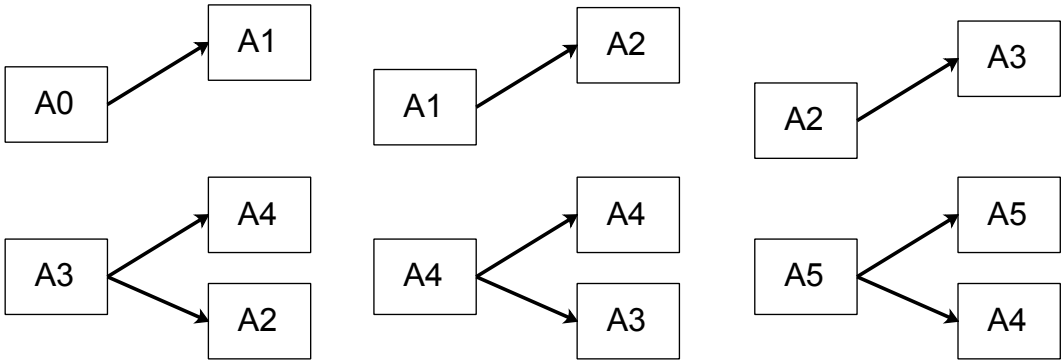

Figura 14 – Grupos com relações lógicas *fuzzy*

Fonte: (SILVA, 2019) adaptado

A base de conhecimento é criada a partir das regras *fuzzy*, que por sua vez são modelos interpretáveis, caixa branca. A partir dessas regras para realizar a previsão do próximo valor *fuzzy*. Para encontrar o valor numérico é realizada a *defuzzificação*. Os resumos dos processos de treinamento e previsão são apresentados nas Figuras 15 e 16

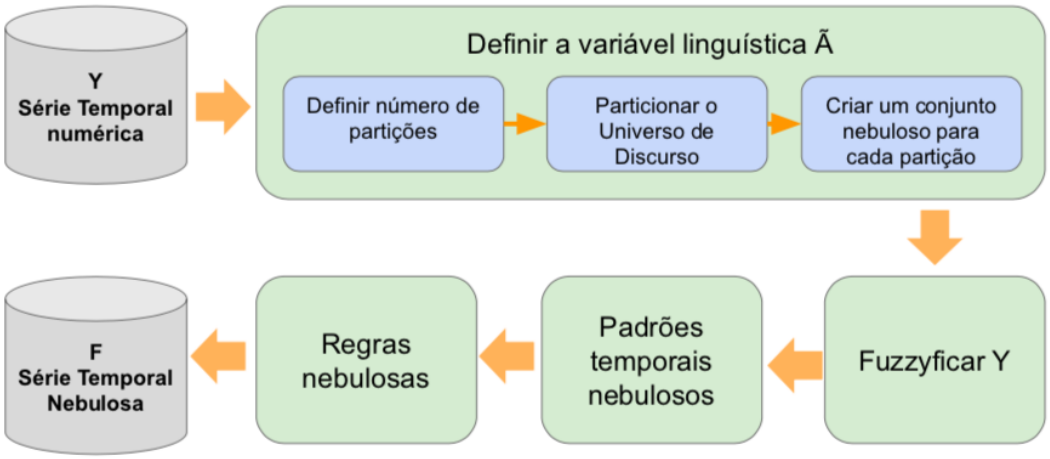

Figura 15 – Processo de treinamento

Fonte: (SILVA, 2019)

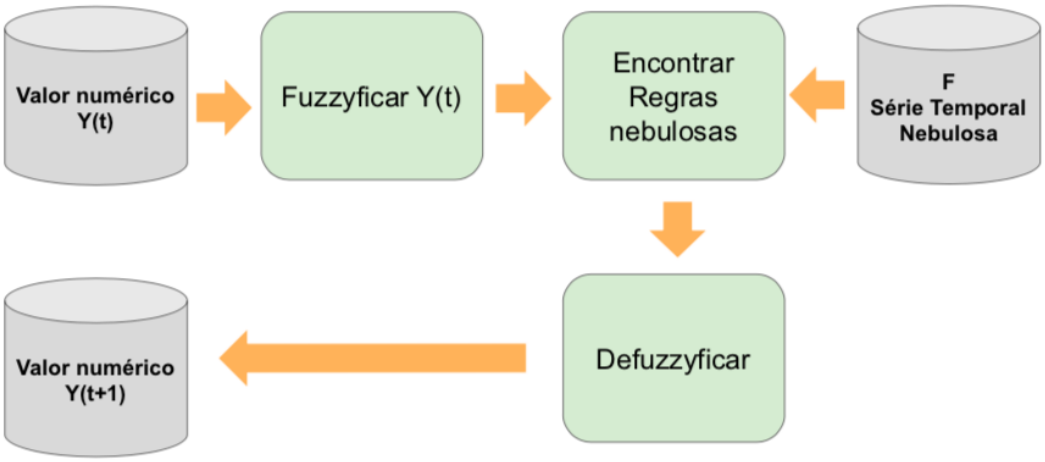

Figura 16 – Processo de predição

Fonte: (SILVA, 2019)

Na etapa de particionamento podem ser ajustados três hiperparâmetros: o número de partições, o método de particionamento e a função de pertinência. Já para a modelagem pode ser ajustado a ordem do modelo que é o número de atrasos necessários para prever um próximo valor e também o modelo em si que define metodologia com que as regras serão organizadas.

2.6 Métricas adotadas

As métricas de precisão normalmente empregadas para avaliar os modelos de previsão pontual são o erro quadrático médio quadrático (RMSE), o erro médio ponderado (MAPE) e a estatística U de *Theil*. Em todos as equações abaixo considera-se y os dados reais, $\hat{y}$ os valores previstos e n número de amostras.

2.6.1 RMSE

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{2.3}$$

O RMSE é a raiz do erro médio quadrático da diferença entre a predição e o valor real. É uma medida análoga ao desvio padrão. Interpreta-se seu valor como uma medida do desvio médio entre observado e predito.

2.6.2 MAPE

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|y_t - \hat{y}_t|}{|y_t|} \times 100 \tag{2.4}$$

O MAPE (Erro Absoluto Médio Percentual) mede o erro em porcentagem. Este é calculado como a média do erro percentual. Expressa a acurácia do erro em porcentagem. Por exemplo, se temos um MAPE de 10% quer dizer o erro médio do estimador é de 10%. Quanto seja menor o MAPE melhor é o ajuste.

2.6.3 U de *Theil*

$$U = \sqrt{\frac{\sum_{t=1}^{n-1}\left(\frac{\hat{y}_{t+1}-y_{t-1}}{y_t}\right)^2}{\sum_{t=1}^{n-1}\left(\frac{y_{t+1}-y_{t-1}}{y_t}\right)^2}} \tag{2.5}$$

O coeficiente U de *Theil* avalia o desempenho da previsão em relação à previsão pelo método Naïve. A previsão Naïve significa que a estimativa do valor futuro é igual ao valor atual. A estatística U mede o quanto a previsão é melhor que o método Naïve, com $U = 1$ significando que ambos os métodos são iguais, $U > 1$ que o método proposto é pior que Naïve e $U < 1$ que ele é melhor.

3 DESENVOLVIMENTO E RESULTADOS

Neste capítulo são apresentados os dados utilizados neste trabalho, a aplicação das técnicas baseadas em redes neurais recorrentes e séries temporais nebulosas e os resultados obtidos. Na seção 3.1 são apresentados os dados de forma gráfica e suas características. Na seção 3.2 o desenvolvimento da aplicação da rede neural recorrente é apresentado e por último na seção 3.3 o desenvolvimento do previsor é feito por meio da série temporal nebulosa. O serviço na nuvem *Colab*, hospedada pela Google, que é disponibilizada online e gratuitamente foi usada como plataforma de programação em *Python*.

3.1 Dados

Os dados utilizados nesse trabalho, Figura 17, foram obtidos do banco de dados da planta de concentração da Mina de Fábrica situada em Congonhas-MG. A cada intervalo de 2 horas uma amostra da polpa que alimenta um separador magnético é tomada e enviada para análise de porcentagem de Ferro e Sílica em laboratório. Cada medida demora cerca de 2 horas para ficar pronta.

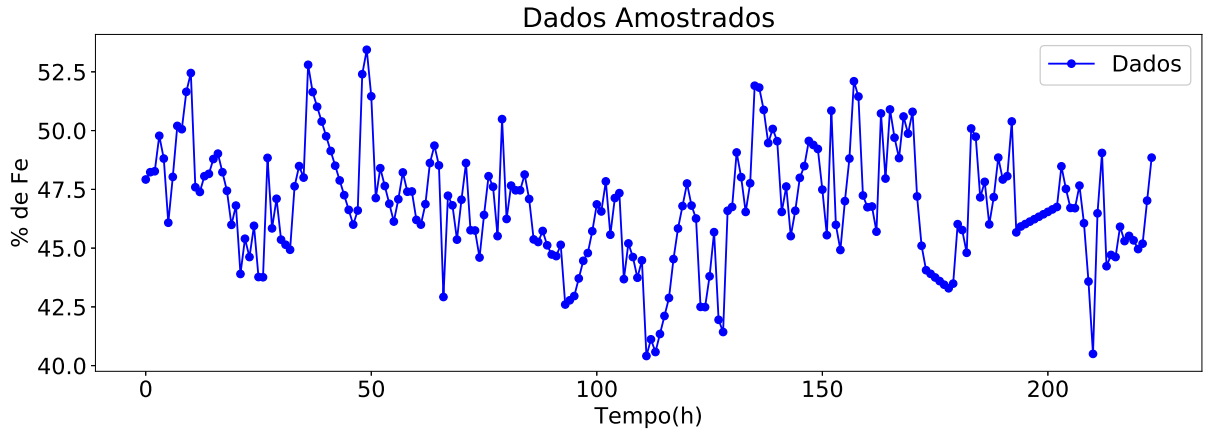

Figura 17 – Medidas da % de Fe

Fonte: banco de dados da Mina Fábrica

O intervalo dos dados utilizados nesse trabalho está entre 01 de janeiro de 2018 às 0h até 19 de janeiro às 14h de 2018 totalizando 224 amostras apresentadas no Anexo A e disponíveis no link `http://bit.ly/dadosMonografiaSantino`. Nesse trabalho será analisado somente o preditor para percentual de ferro. Os dados foram divididos em 200 amostras iniciais para treinamento da rede neural e os 24 valores restantes foram usados para teste da rede neural.

3.1.1 Análise dos dados

Uma abordagem inicial visa analisar a auto correlação entre as observações da série temporal com a função de autocorrelação (ACF). O objetivo é verificar a relação temporal entre as amostras de percentual de ferro amostradas no decorrer do tempo. No gráfico da Figura 18 apresenta-se a aplicação da função AFC nos 100 primeiros dados amostrados.

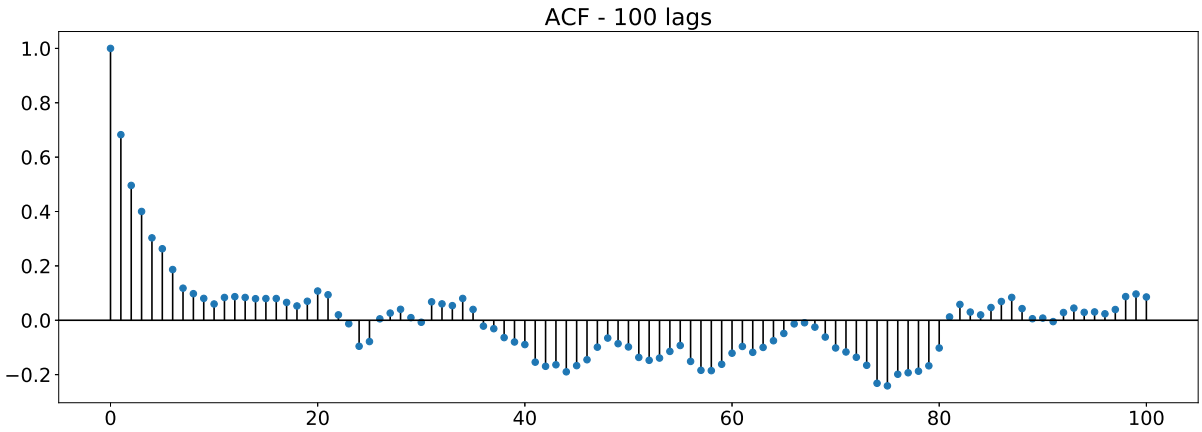

Figura 18 – Auto correlação entre as observações

Fonte: próprio autor

3.2 Redes neurais recorrentes LSTM

3.2.1 Obtenção dos parâmetros da rede neural

O primeiro parâmetro obtido foi a quantidade de valores do passado que deveriam ser usados para realizar a previsão de um passo a frente. A cada treinamento de uma rede neural há uma atribuição aleatória dos pesos iniciais e que depois são ajustados. Por isso, para cada número de valores do passados em um intervalo de 1 a 16, a rede foi treinada 20 vezes e os valores de desempenho salvos. A rede foi iniciada com 2 camadas ocultas possuindo 20 células de memória cada. O melhor desempenho é obtido analisando-se a Figura 19, que apresenta um *boxplot* do desempenho dos treinamentos.

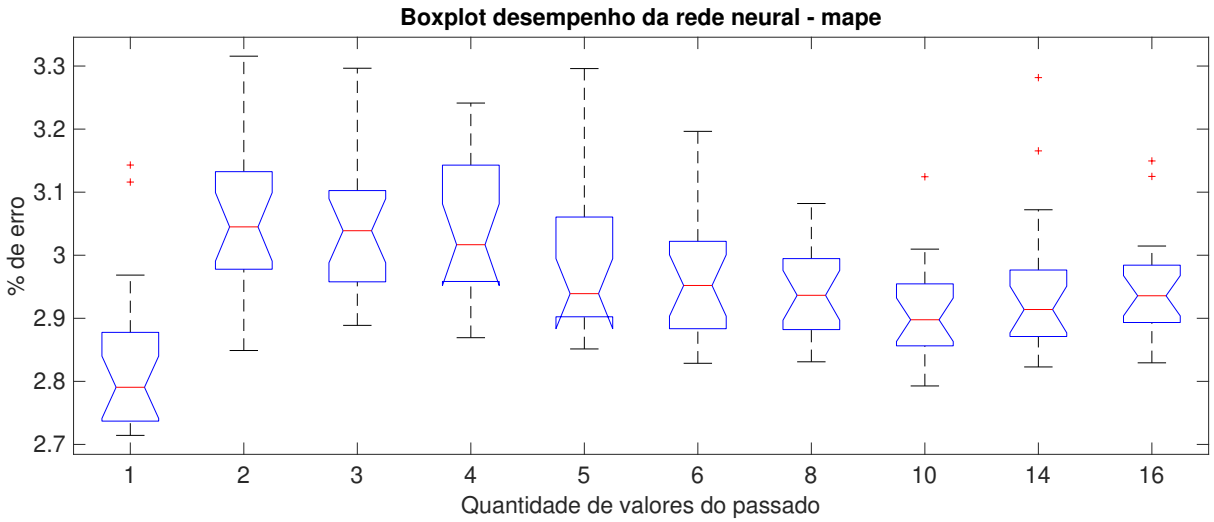

Figura 19 – Percentual de erro da rede neural para quantidade de valores do passado diferentes

Fonte: próprio autor

De acordo com o *boxplot* da Figura 19, tem-se que a rede apresentou uma mediana de 2,7906% de erro com apenas 1 valor do passado sendo usado para prever o próximo valor. A medida que foi-se aumentando a quantidade de atrasos no tempo, a rede apresentou pior desempenho. No entanto, após 14 atrasos o desempenho volta a piorar. Assim a base de dados para teste da rede terá que levar em consideração o último valor real para prever o próximo valor. A base completa é mostrada na Figura 20.

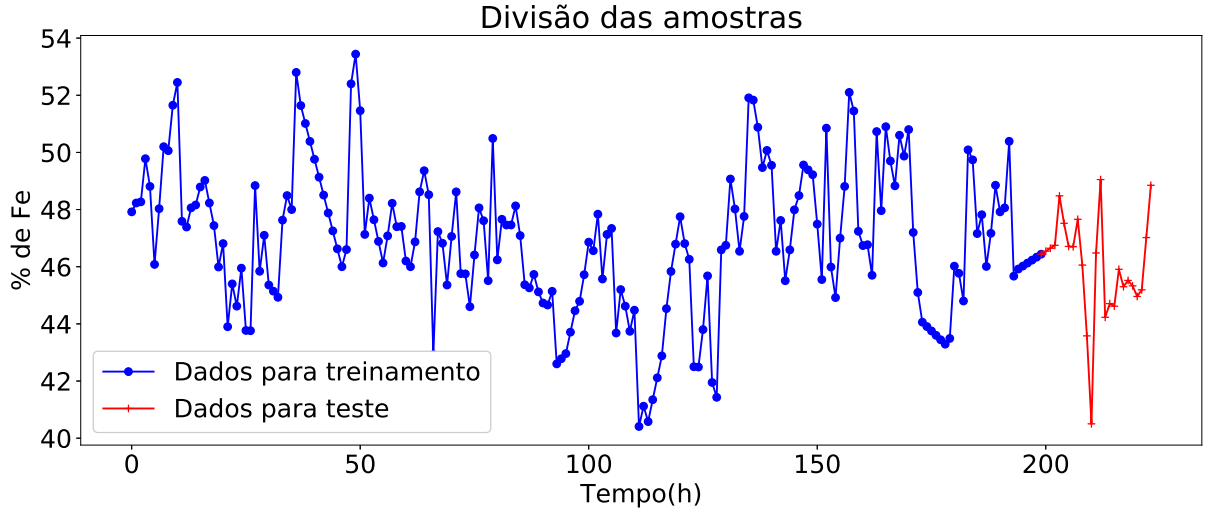

Figura 20 – Divisão dos dados

Fonte: próprio autor

Partindo-se dessa etapa variou-se o o número de células de memória das camadas ocultas

entre 2 e 30 para avaliar qual parâmetro apresenta um melhor desempenho da rede. O resultado é apresentado no *boxplot* da Figura 21.

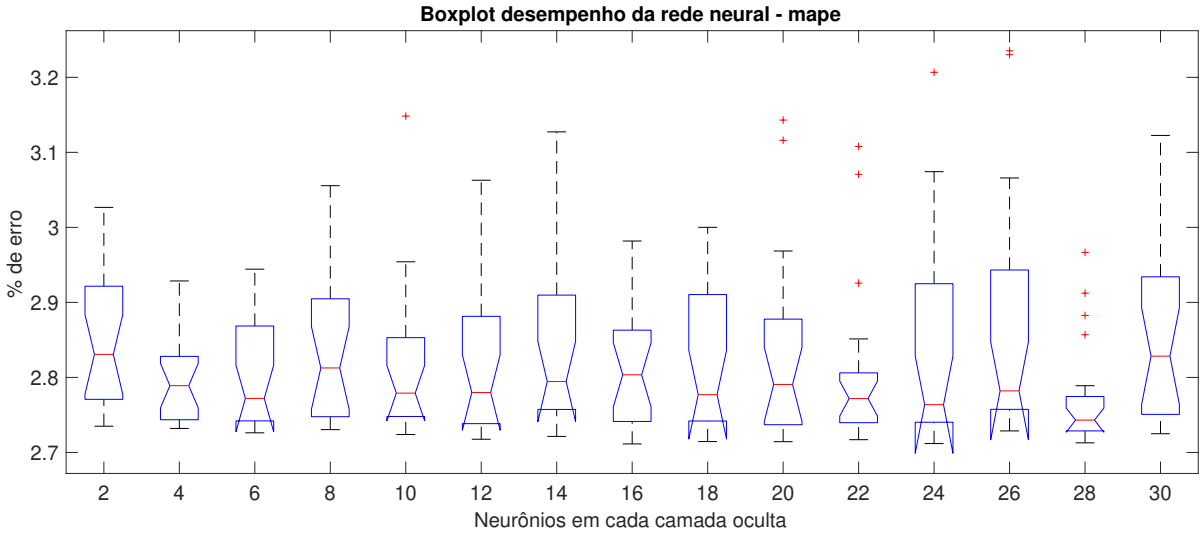

Figura 21 – Desempenho da rede neural para quantidade de neurônios diferentes

Fonte: próprio autor

Finalmente, foi realizado teste com dois otimizadores diferentes disponibilizados pela biblioteca Keras, o *rmsprop*, *adam* e outros.

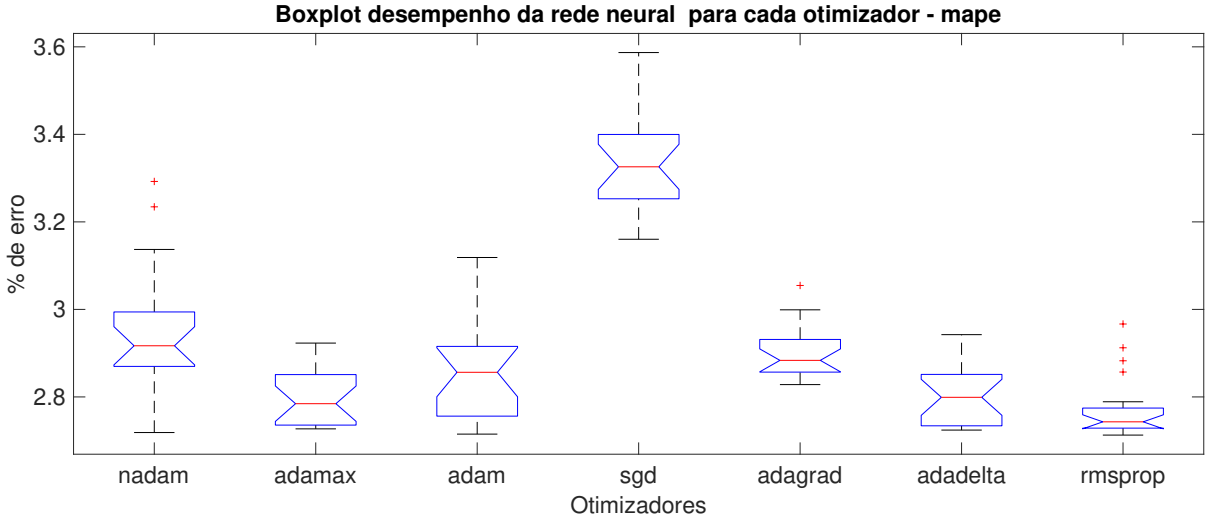

Figura 22 – Desempenho da rede neural otimizadores diferentes

Fonte: próprio autor

A rede neural com melhor desempenho é ilustrada na Figura 23. Ela possui 2 camadas ocultas com 28 células de memória cada, usa 1 valor de atraso no tempo para prever o próximo

valor e usa o otimizador *rmsprop*. O resultado das previsões é apresentado na Figura 24 onde é possível visualizar os dados do conjunto de teste e sua previsão pela rede.

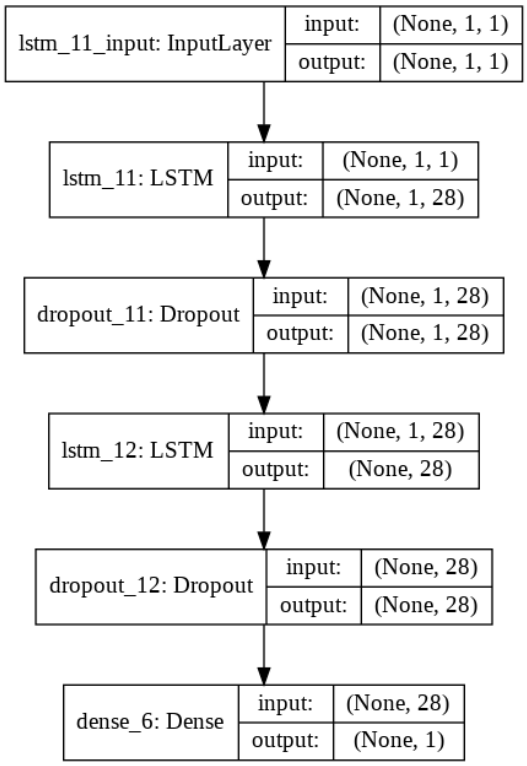

Figura 23 – Rede neural

Fonte: próprio autor

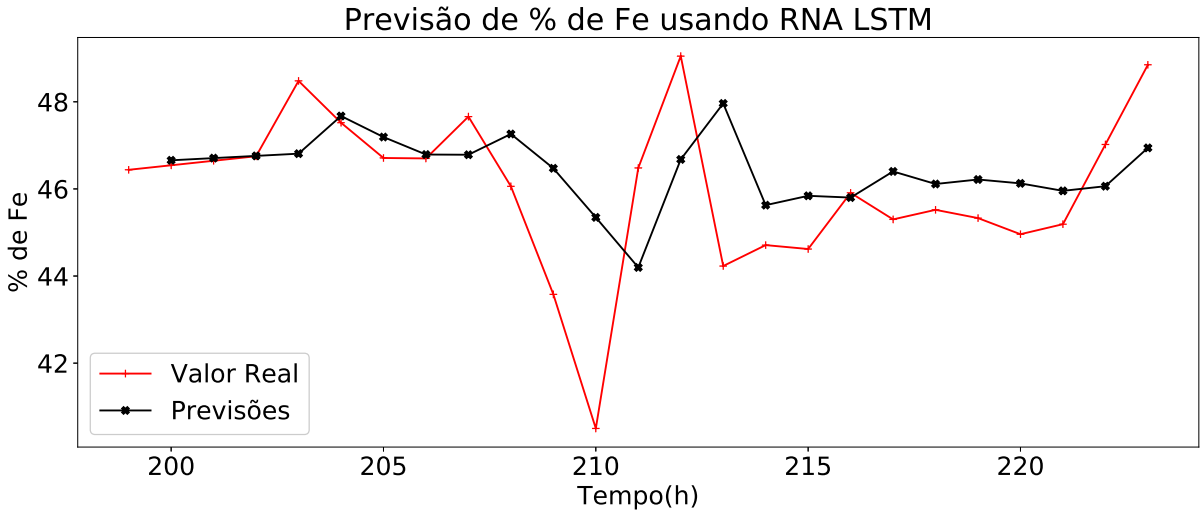

Figura 24 – Previsão

Fonte: próprio autor

O melhor desempenho da rede neural apresentou U = 0.84, como apresenta o *boxplot* da Figura 25 sendo assim 16% melhor que o modelo trivial *naive*. Ainda assim apresentou apenas 2.82% de erro. Os índices de avaliação são apresentados na Tabela 2.

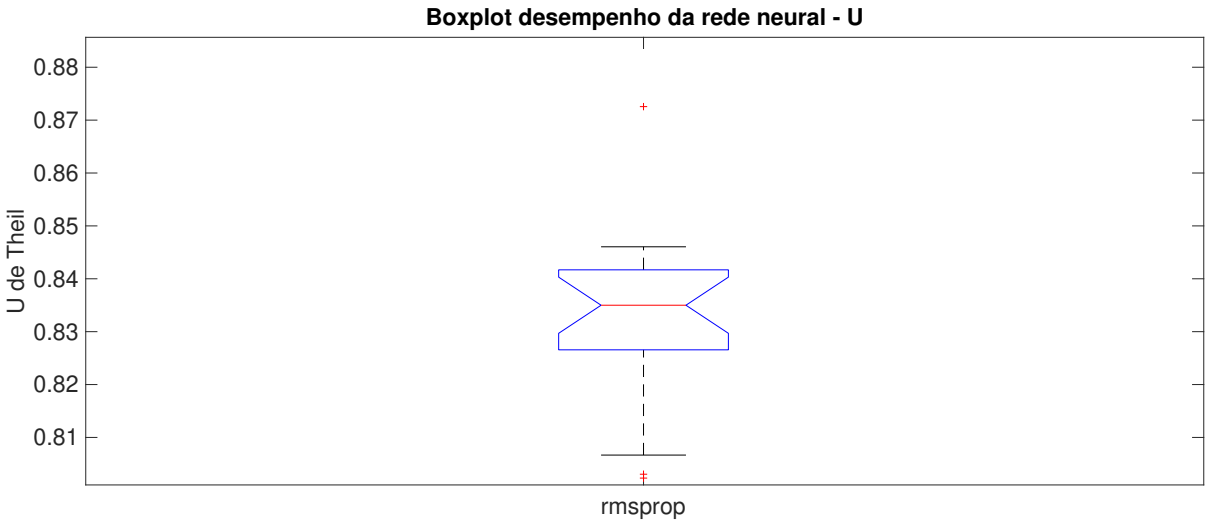

Figura 25 – Índice de desempenho U

Fonte: próprio autor

Tabela 2 – Índices de avaliação para o modelo de rede neural

Índices de Avaliação		
RMSE	MAPE	U
1,75	2,82	0,84

Todo o código utilizado nessa parte do trabalho pode ser acessado e executado online no seguinte link `http://bit.ly/codeLSTM`.

3.3 Séries temporais nebulosas (FTS)

3.3.1 Obtenção dos parâmetros da série temporal nebulosa

O primeiro parâmetro a ser obtido é a ordem do modelo da série temporal, para isso foi realizado 8 testes no intervalo de 1 a 8 tempos de atraso usados para criar o modelo. Um fato importante é que a série temporal nebulosa não há necessidade de vários treinamentos, pois para cada configuração há a criação de um modelo fixo sem nenhum ajuste aleatório inicial como nas redes neurais. Os resultados são apresentados em um gráfico na Figura 26. O melhor valor foi um modelo com ordem 1, ou seja, apenas o 1 valor de atraso no tempo será usado para prever o próximo valor no tempo.

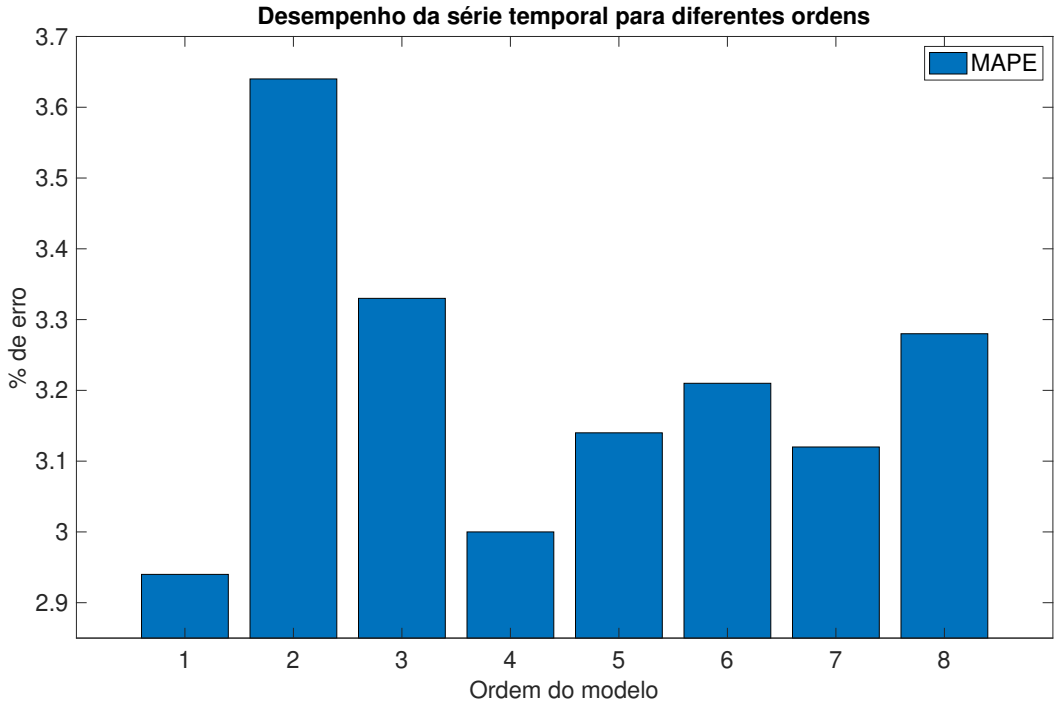

Figura 26 – Desempenho da série temporal para ordens diferentes

Fonte: próprio autor

O segundo parâmetro é a escolha do melhor particionador para o conjunto de dados. O primeiro teste foi analisado o intervalo de 0 a 50 partições aplicadas aos dados, uma breve ilustração é apresentada na Figura 27 com 2, 4, 6, 8, 10, 12 partições.

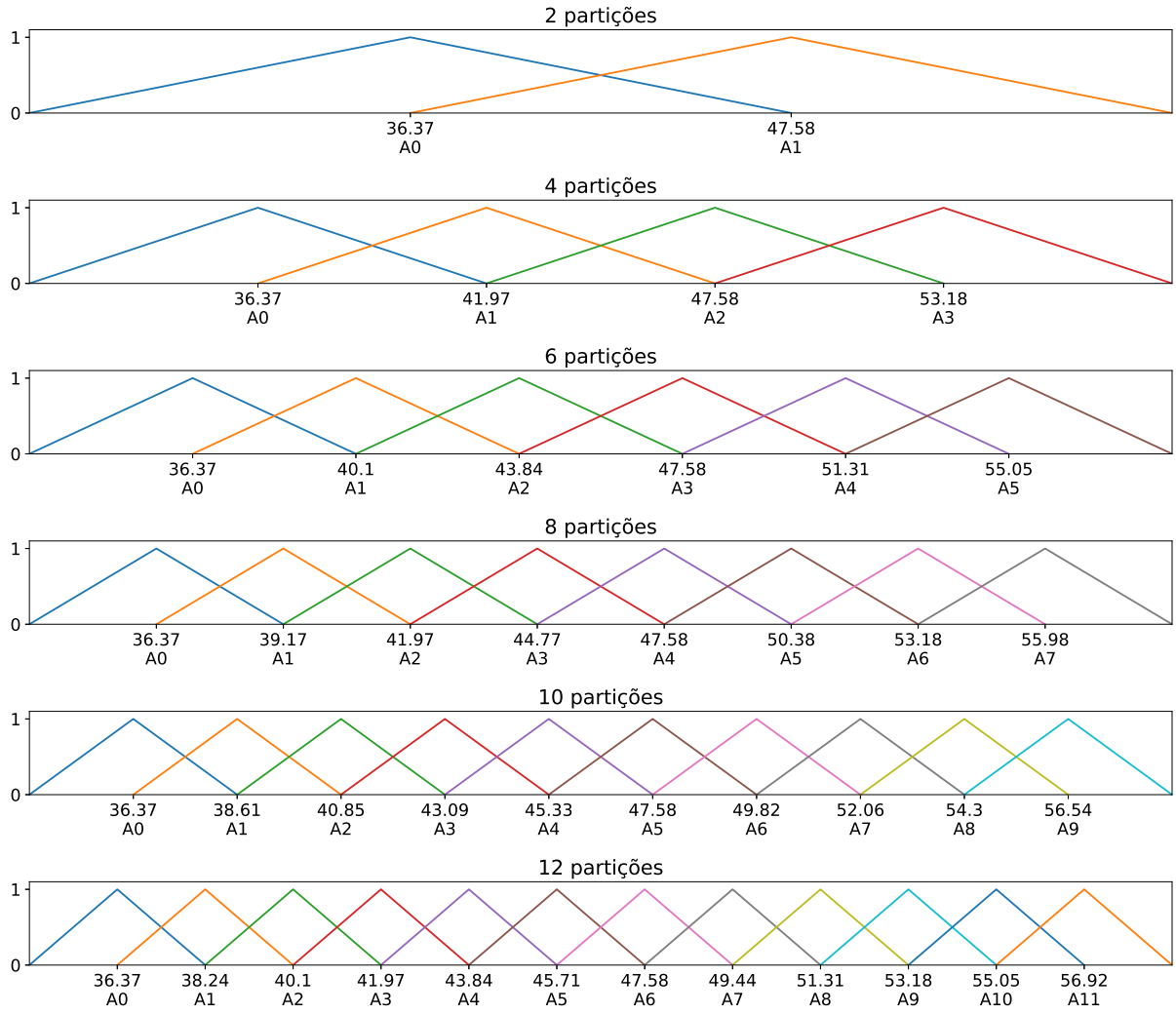

Figura 27 – Quantidade de partições

Fonte: próprio autor

O resultado comparando a quantidade de partições e o percentual de erro é mostrado na Figura 28. Logo no início, com no intervalo de 1 a 4 partições, o modelo apresentou um erro muito grande, e dessa forma, não foi possível mensurar, nos próximos, intervalos o melhor resultado. Sendo assim, a Figura 29 em que a partir de 4 partições tem-se uma menor porcentagem de erro.

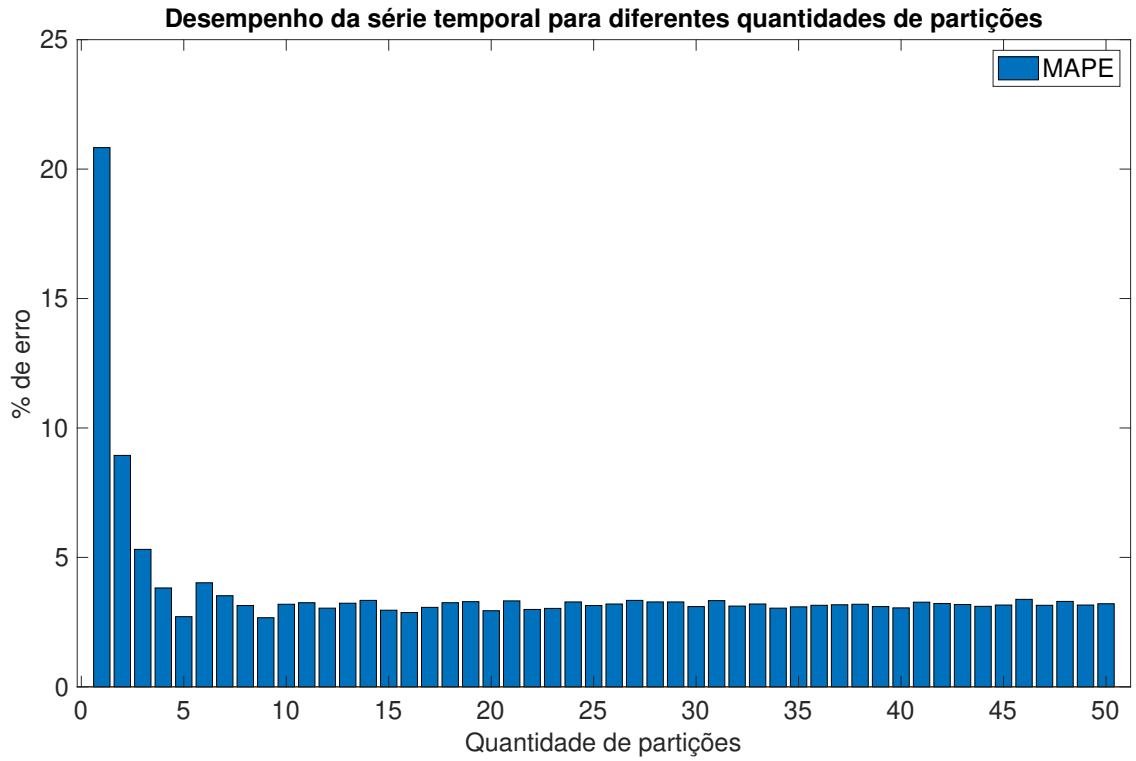

Figura 28 – Desempenho da série temporal para quantidades de partições diferentes

Fonte: próprio autor

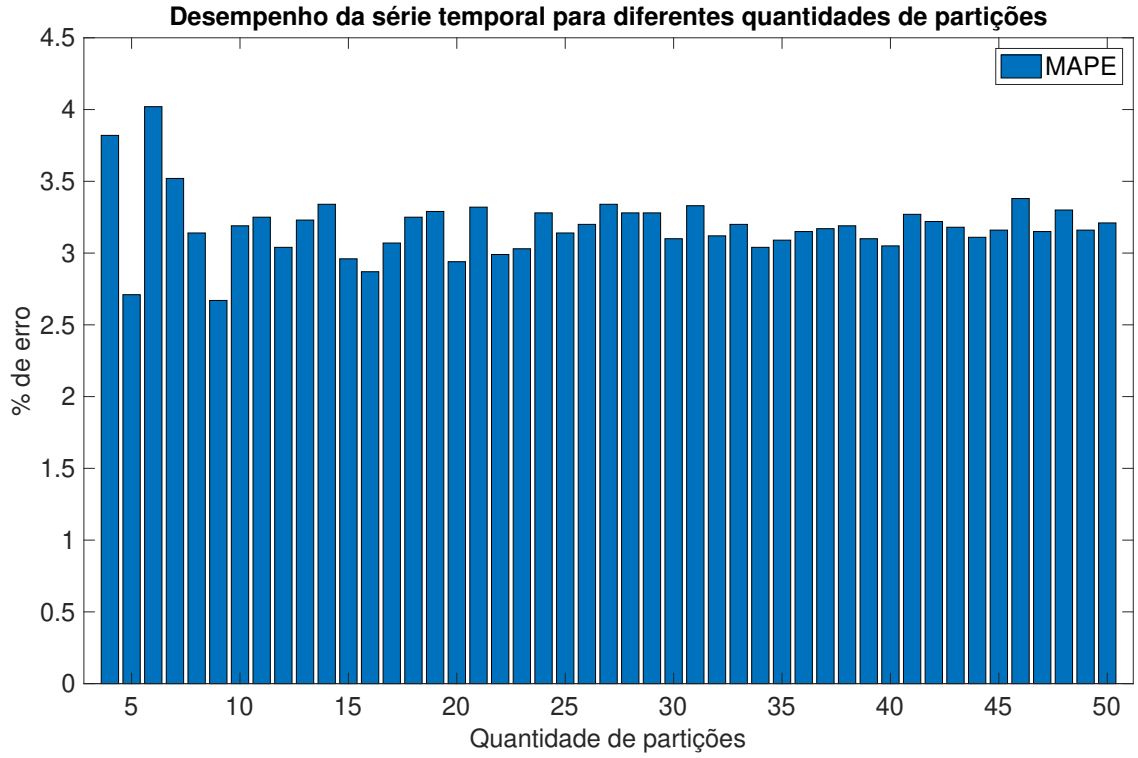

Figura 29 – Desempenho da série temporal para quantidades de particionadores diferentes

Fonte: próprio autor

Um terceiro teste apresenta a forma como os intervalos das partições serão divididos. O particionamento em *Grid* dá uma distribuição uniforme dos conjuntos *fuzzy*. No particionador *Entropy* a entropia de dados que define os melhores pontos médios para os conjuntos *fuzzy*, de forma semelhante o FCM e Cmeans, separa os dados com incremento de estatística.

As ilustrações dos particionadores são apresentados na Figura 30 e os resultados dos desempenhos de cada método são apresentados na Figura 31, onde se mostra um menor percentual de erro, com 2,67% para o método mais simples, particionamento em *Grid*.

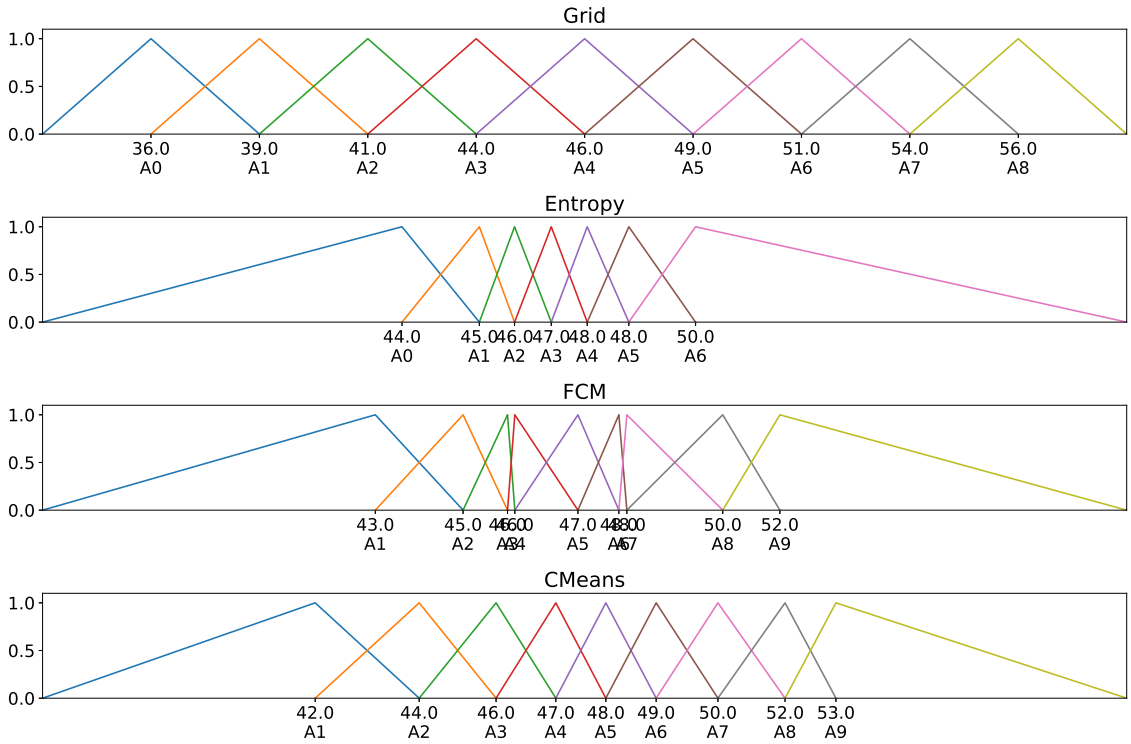

Figura 30 – Métodos de particionadores

Fonte: próprio autor

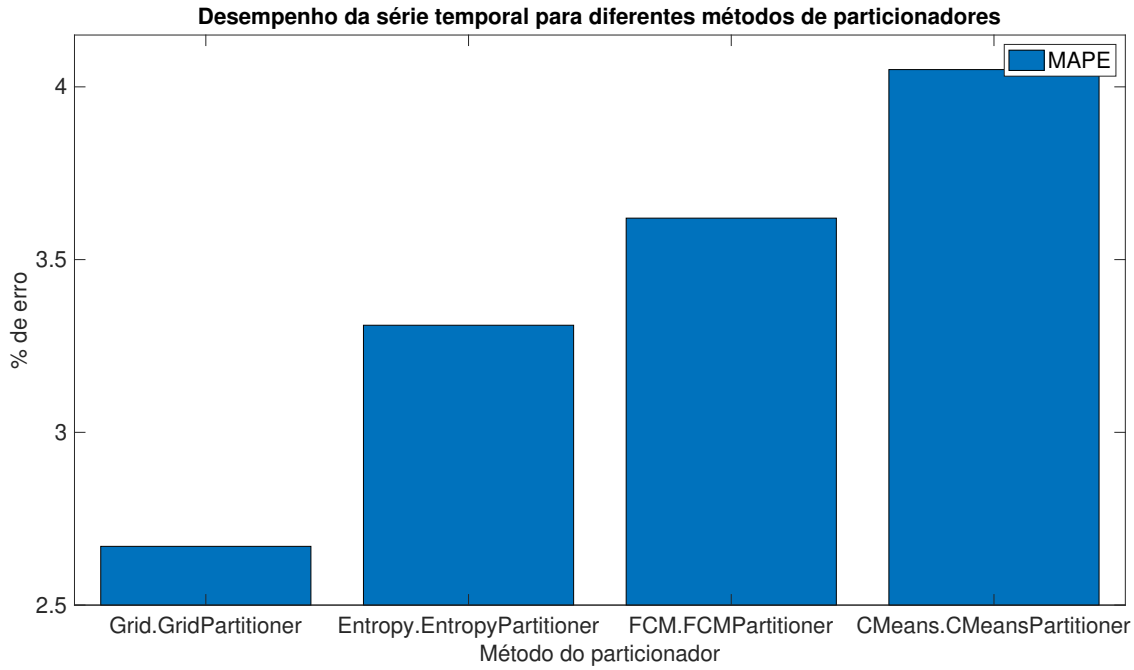

Figura 31 – Desempenho da série temporal para métodos de particionadores diferentes

Fonte: próprio autor

No quarto teste foram avaliadas diferentes funções de pertinência: a *membership.trimf* são funções triangulares; a *membership.trapmf* são funções trapezoidais e a *membership.gaussm* são funções gaussianas. Uma ilustração das funções de pertinência são apresentados na Figura 32.

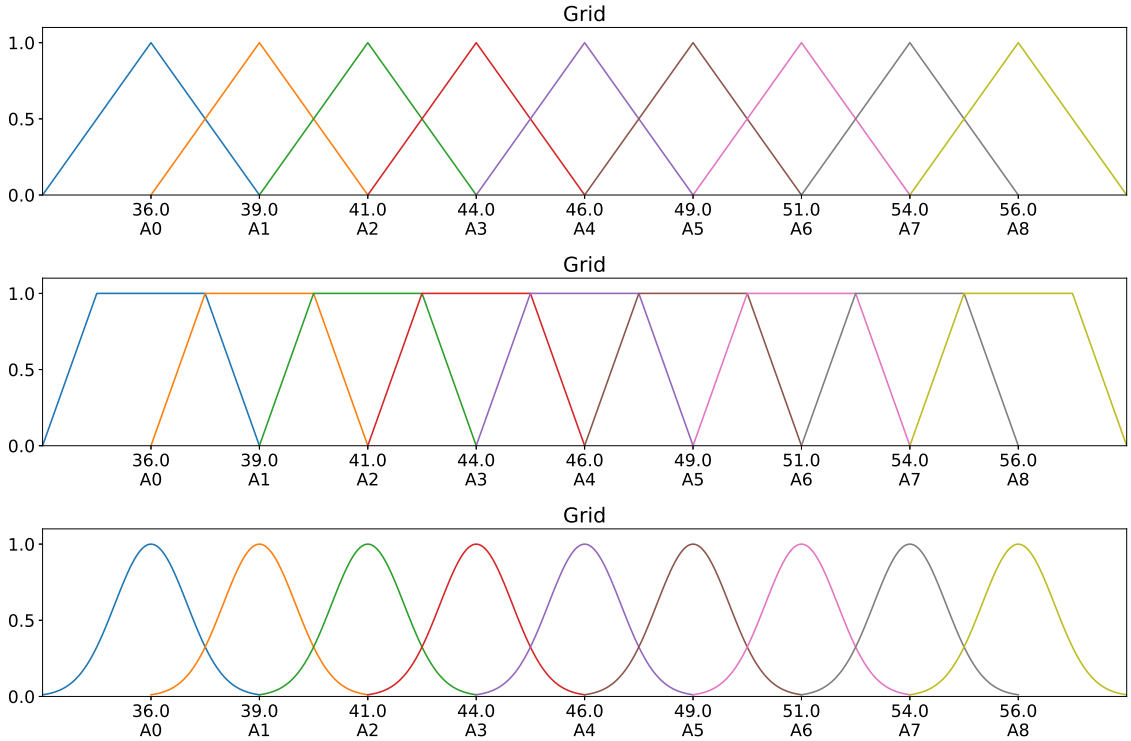

Figura 32 – Funções de pertinência

Fonte: próprio autor

O desempenho do modelo para cada função de pertinência é apresentado em um gráfico da Figura 33. O percentual de erro foi igual tanto para a *membership.trimf* quanto para *membership.trapmf*. A função *membership.trimf* por ser mais simples foi adotada.

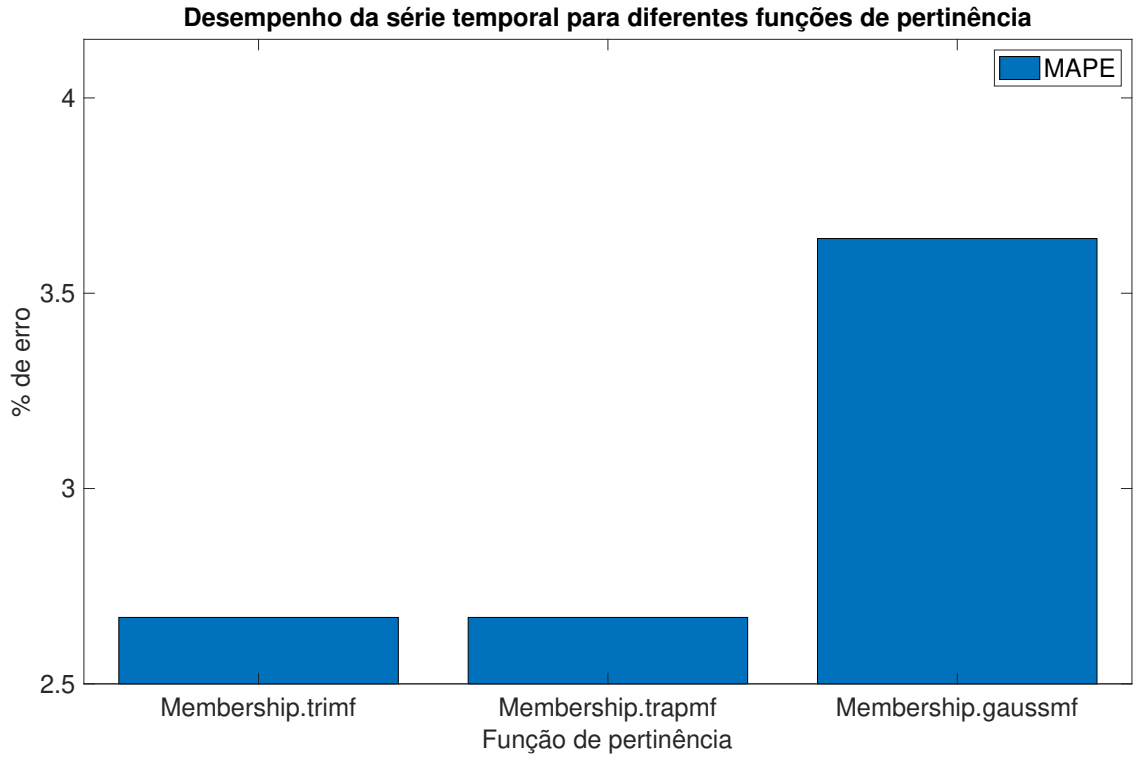

Figura 33 – Desempenho da série temporal para funções de pertinência diferentes

Fonte: próprio autor

Uma vista parcial do modelo final das séries temporais é apresentado na Figura 34.

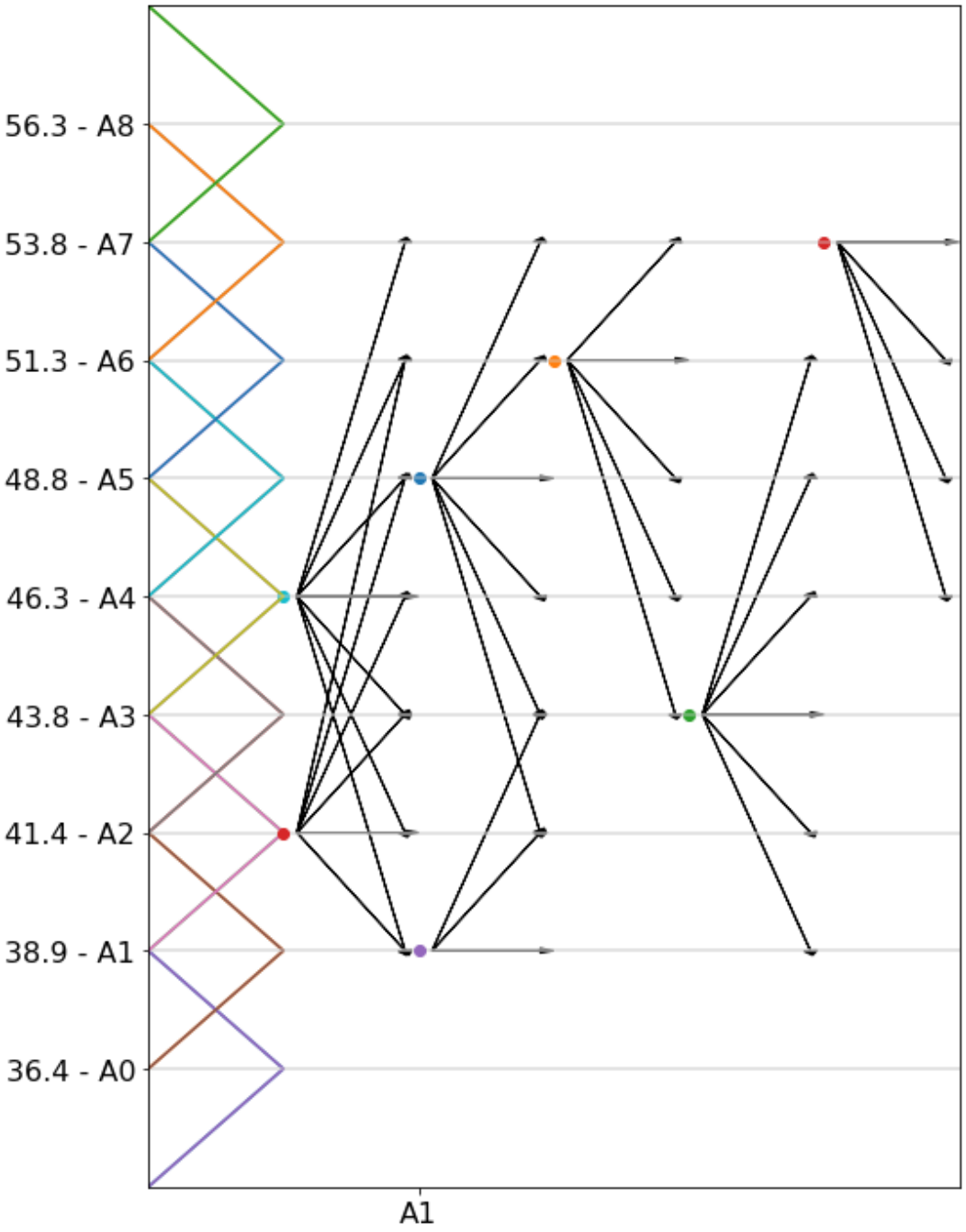

Figura 34 – Modelo da série temporal nebulosa

Fonte: próprio autor

Portanto, a série temporal nebulosa com melhor desempenho é de ordem 1, com particionador com 9 funções de pertinência triangulares de distribuição uniforme. O resultado das previsões para o conjunto de testes é apresentado na Figura 35.

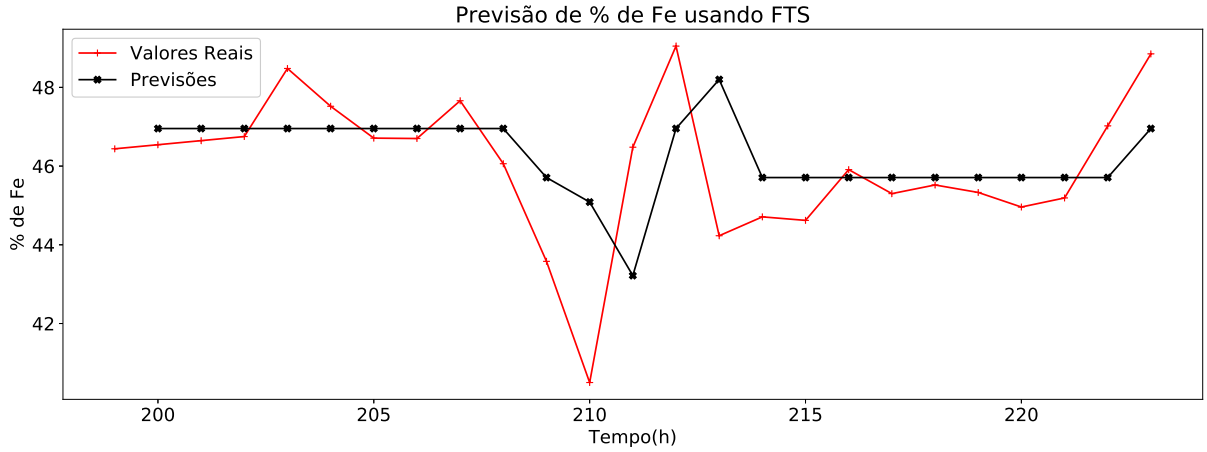

Figura 35 – Resultado

Fonte: próprio autor

O desempenho do modelo de série temporal *fuzzy* proposto é apresentado na Tabela 3.

Tabela 3 – Desempenho da série temporal nebulosa

Índices de Avaliação		
RMSE	MAPE	U
1,70	2,67	0,82

Todo o código utilizado nessa parte do trabalho pode ser acessado e executado online no seguinte link `http://bit.ly/codeFTS`

4 DISCUSSÃO

Os dados dos índices de avaliação da metodologia por redes neurais recorrentes LSTM e séries temporais *fuzzy* são apresentados na Tabela 4.

Tabela 4 – Índices de Avaliação

Índices de Avaliação			
Método	RMSE	MAPE	U
Rede neural LSTM	1,75	2,81	0,84
Série temporal nebulosa	1,70	2,67	0,82

Como passo inicial tentou-se usar redes neurais mais simples porém não apresentaram resultados significantes para serem abordados nesse trabalho.

O método que aplica rede neural recorrente se demostrou computacionalmente mais complexo, demandando um consumo maior de memória *ram* da plataforma *colab*. Um fator que contribui para essa complexidade é que a rede neural, a cada treinamento, inicia-se com pesos aleatórios. No decorrer do treinamento, o algoritmo de aprendizagem busca um ajuste final dos pesos. Devido a essa inicialização aleatória, o algoritmo encontrará um ajuste de pesos diferentemente para cada término do treinamento.

Essa aleatoriedade também implica que para cada modelo final, após um treinamento, a rede não apresentará o mesmo resultado e, portanto, são necessários vários treinamentos para os mesmos parâmetros para se obter uma estatística do índice de desempenho. Uma desvantagem da rede neural é que ela apresenta um modelo caixa preta, ou seja, mesmo tendo acesso aos pesos ajustados ao final do treinamento eles, em si, não dizem nada, não sendo possível interpretar o modelo.

O método por série temporal nebulosa foi computacionalmente simples com execuções na plataforma *colab* em alguns segundos. Isso foi possível pelo uso de um modelo de ordem 1, que representa um modelo simples com baixa complexidade nos agrupamentos. Nesse modelo, não há a inicialização aleatória de qualquer parâmetro, contribuindo para sua simplicidade. Assim, não são necessárias várias execuções para obter uma aproximação do desempenho. Uma outra vantagem é que esse método pode ser interpretável, ou seja, a relação entre as regras *fuzzy* que constituem o modelo podem ser explicadas.

5 CONCLUSÃO

A eficiência do circuito de separação magnética é afetada pelo percentual de ferro na composição da polpa de alimentação. Em mina de Fábrica, as amostras no percentual de ferro são obtidas de análise laboratorial com atraso de duas horas para se obter o resultado real. Atualmente o circuito opera repetindo o último ajuste de acordo com o resultado do laboratório não implicando em um ajuste de acordo com a característica da polpa atual. Portanto, é importante poder prever de forma confiável as medidas de porcentagem de ferro e sílica. Duas técnicas de inteligência computacional são propostas como soluções para o problema proposto.

O índice de desempenho U de *Theil* propõe avaliar o quão melhor foi cada método em relação a metodologia atual de ajuste do circuito.

O modelo previsor com rede neural recorrente apresentou previsões próximas e até iguais aos valores reais. Porém, com oscilações bruscas, apresentou erros grandes. Entretanto em um intervalo de 24 amostras seu desempenho médio foi 16% melhor do que a metodologia atual de ajuste.

Já o modelo por série temporal nebulosa apresentou previsões com valores fixos em intervalos com pouca variabilidade dos dados reais, enquanto nas variações bruscas o previsor apresentou melhor desempenho que a rede neural. No mesmo intervalo, seu índice de desempenho foi melhor que o modelo por rede neural recorrente, sendo 18% melhor que a metodologia atual.

Como trabalhos futuros recomenda-se: *i)* aplicação desses métodos com percentual de sílica; *ii)* analisar a viabilidade da implementação da densidade do material como variável exógena para prever a porcentagem de ferro e sílica; *iii)* testar outra técnicas de precisão de séries temporais como: estimadores de mínimos quadrados e NARMAX e ARMAX; *iiii* avaliar técnicas de seleção de dados do passado intervalados entre si para entradas dos modelos; *iiiii* avaliar previsões de mais passos a frente com outros modelos.

REFERÊNCIAS

BROCKWELL, P. J.; DAVIS, R. A. *Introduction to time series and forecasting*. [S.l.]: springer, 2016. Citado 2 vezes nas páginas 7 e 8.

CAJAL, S. R. y; AZOULAY, L. *Histologie du systeme nerveux de l'homme & des vertebres*. [S.l.]: Consejo superior de investigaciones cientificas, Instituto Ramon Y Cajal, 1952. Citado na página 9.

CHATFIELD, C. *Time-series forecasting*. [S.l.]: Chapman and Hall/CRC, 2000. Citado 3 vezes nas páginas 1, 7 e 8.

CHAVES, A. P. *Teoria e prática do tratamento de minérios*. 2. ed. São Paulo: Signus Editora, 2002. Citado na página 5.

CHEN, S.-M. Forecasting enrollments based on fuzzy time series. *Fuzzy sets and systems*, 1996. Elsevier, v. 81, n. 3, p. 311–319, 1996. Citado na página 3.

COULIBALY, P.; BALDWIN, C. K. Nonstationary hydrological time series forecasting using nonlinear dynamic methods. *Journal of Hydrology*, 2005. Elsevier, v. 307, n. 1-4, p. 164–174, 2005. Citado na página 2.

GOMES, P. A. et al. Exploiting recurrent neural networks in the forecasting of bees' level of activity. In: SPRINGER. *International Conference on Artificial Neural Networks*. [S.l.], 2017. p. 254–261. Citado na página 2.

GOMES, V. G. R. *Controle Multivariável de Separação Magnética Baseado em predição Temporal Qualitativa do Rom*. 36 p. Dissertação (Mestrado) — Universidade Federal de Ouro Preto / Instituto Tecnológico Vale, Ouro Preto, 2019. Citado 2 vezes nas páginas 6 e 7.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2007. Citado 4 vezes nas páginas 1, 9, 10 e 12.

JÚNIOR, Ê. L. et al. Projeto de um controlador fuzzy multivariavel para uma planta de desaguamento de minério de ferro. 2018. 2018. Citado na página 3.

JÚNIOR, M. P.; BITARãES, S. M.; EUZÉBIO, T. A. Projeto de controle fuzzy para aprimorar a eficiência energética de secadores rotativos. 2019. 2019. Citado na página 3.

LIPTON, Z. C.; BERKOWITZ, J.; ELKAN, C. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015. 2015. Citado 2 vezes nas páginas 12 e 14.

MCCOY, J.; AURET, L. Machine learning applications in minerals processing: A review. *Minerals Engineering*, 2019. Elsevier, v. 132, p. 95–109, 2019. Citado na página 2.

OBERTEUFFER, J. Magnetic separation: A review of principles, devices, and applications. *IEEE Transactions on Magnetics*, 1974. v. 10, n. 2, p. 223–238, June 1974. Citado 2 vezes nas páginas 5 e 6.

SCHMIDHUBER, J.; HOCHREITER, S. Long short-term memory. *Neural Comput*, 1997. v. 9, n. 8, p. 1735–1780, 1997. Citado na página 13.

SILVA, P. C. de Lima e. *Scalable Models for Probabilistic Forecasting with Fuzzy Time Series*. Tese (Doutorado) — Machine Intelligence and Data Science Laboratory, Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, 2019. Disponível em: <http://hdl.handle.net/1843/30040>. Acesso em: 08 out. 2019. Citado 6 vezes nas páginas 2, 3, 15, 16, 17 e 18.

SONG, Q.; CHISSOM, B. S. Forecasting enrollments with fuzzy time series—part i. *Fuzzy sets and systems*, 1993. Elsevier, v. 54, n. 1, p. 1–9, 1993. Citado na página 15.

SONG, Q.; CHISSOM, B. S. Fuzzy time series and its models. *Fuzzy sets and systems*, 1993. Elsevier, v. 54, n. 3, p. 269–277, 1993. Citado na página 15.

SUN, B. et al. Prediction of stock index futures prices based on fuzzy sets and multivariate fuzzy time series. *Neurocomputing*, 2015. Elsevier, v. 151, p. 1528–1536, 2015. Citado na página 3.

WASMUTH, H.-D.; UNKELBACH, K.-H. Recent developments in magnetic separation of feebly magnetic minerals. *Minerals Engineering*, 1991. Elsevier, v. 4, n. 7-11, p. 825–837, 1991. Citado na página 6.

ZADEH, L. A. The concept of a linguistic variable and its application to approximate reasoning—i. *Information sciences*, 1975. Elsevier, v. 8, n. 3, p. 199–249, 1975. Citado na página 14.

ZHANG, G.; PATUWO, B. E.; HU, M. Y. Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 1998. Elsevier, v. 14, n. 1, p. 35–62, 1998. Citado na página 1.

ZHANG, G. P. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 2003. Elsevier, v. 50, p. 159–175, 2003. Citado na página 9.

ANEXO A – DADOS

Tabela 5 – Dados

Janeiro/2018								
N	Dia - hora	%Fe	N	Dia - hora	%Fe	N	Dia - hora	%Fe
0	01 - 00h	47,9200	41	04 - 10h	49,1332	82	07 - 20h	47,4600
1	01 - 02h	48,2300	42	04 - 12h	48,5066	83	07 - 22h	47,4601
2	01 - 04h	48,2702	43	04 - 14h	47,8799	84	08 - 00h	48,1299
3	01 - 06h	49,7799	44	04 - 16h	47,2532	85	08 - 02h	47,0898
4	01 - 08h	48,8096	45	04 - 18h	46,6266	86	08 - 04h	45,3700
5	01 - 10h	46,0803	46	04 - 20h	46,0001	87	08 - 06h	45,2601
6	01 - 12h	48,0303	47	04 - 22h	46,6008	88	08 - 08h	45,7299
7	01 - 14h	50,2000	48	05 - 00h	52,4001	89	08 - 10h	45,1199
8	01 - 16h	50,0602	49	05 - 02h	53,4397	90	08 - 12h	44,7300
9	01 - 18h	51,6501	50	05 - 04h	51,4594	91	08 - 14h	44,6601
10	01 - 20h	52,4493	51	05 - 06h	47,1302	92	08 - 16h	45,1396
11	01 - 22h	47,5900	52	05 - 08h	48,3999	93	08 - 18h	42,6000
12	02 - 00h	47,3901	53	05 - 10h	47,6432	94	08 - 20h	42,7800
13	02 - 02h	48,0600	54	05 - 12h	46,8866	95	08 - 22h	42,9601
14	02 - 04h	48,1601	55	05 - 14h	46,1301	96	09 - 00h	43,7101
15	02 - 06h	48,7900	56	05 - 16h	47,0802	97	09 - 02h	44,4600
16	02 - 08h	49,0199	57	05 - 18h	48,2199	98	09 - 04h	44,7901
17	02 - 10h	48,2299	58	05 - 20h	47,4000	99	09 - 06h	45,7202
18	02 - 12h	47,4398	59	05 - 22h	47,4098	100	09 - 08h	46,8600
19	02 - 14h	45,9901	60	06 - 00h	46,2000	101	09 - 10h	46,5602
20	02 - 16h	46,8096	61	06 - 02h	46,0001	102	09 - 12h	47,8397
21	02 - 18h	43,9002	62	06 - 04h	46,8702	103	09 - 14h	45,5702
22	02 - 20h	45,3999	63	06 - 06h	48,6201	104	09 - 16h	47,1300
23	02 - 22h	44,6202	64	06 - 08h	49,3599	105	09 - 18h	47,3395
24	03 - 00h	45,9497	65	06 - 10h	48,5192	106	09 - 20h	43,6802
25	03 - 02h	43,7700	66	06 - 12h	42,9206	107	09 - 22h	45,1999
26	03 - 04h	43,7607	67	06 - 14h	47,2299	108	10 - 00h	44,6199
27	03 - 06h	48,8396	68	06 - 16h	46,8198	109	10 - 02h	43,7401
28	03 - 08h	45,8402	69	06 - 18h	45,3602	110	10 - 04h	44,4794
29	03 - 10h	47,0998	70	06 - 20h	47,0602	111	10 - 06h	40,4101
30	03 - 12h	45,3600	71	06 - 22h	48,6196	112	10 - 08h	41,1199
31	03 - 14h	45,1450	72	07 - 00h	45,7600	113	10 - 10h	40,5801
32	03 - 16h	44,9304	73	07 - 02h	45,7498	114	10 - 12h	41,3468
33	03 - 18h	47,6301	74	07 - 04h	44,6003	115	10 - 14h	42,1134
34	03 - 20h	48,4899	75	07 - 06h	46,4102	116	10 - 16h	42,8802
35	03 - 22h	48,0007	76	07 - 08h	48,0599	117	10 - 18h	44,5336
36	04 - 00h	52,7998	77	07 - 10h	47,6097	118	10 - 20h	45,8351
37	04 - 02h	51,6399	78	07 - 12h	45,5107	119	10 - 22h	46,7901
38	04 - 04h	51,0132	79	07 - 14h	50,4894	120	11 - 00h	47,7499
39	04 - 06h	50,3866	80	07 - 16h	46,2402	121	11 - 02h	46,8099
40	04 - 08h	49,7599	81	07 - 18h	47,6600	122	11 - 04h	46,2595

Tabela 6 – Dados continuação

Janeiro/2018								
N	Dia - hora	%Fe	N	Dia - hora	%Fe	N	Dia - hora	%Fe
123	11 - 06h	42,5000	164	14 - 16h	47,9604	205	18 - 02h	46,7100
124	11 - 08h	42,4902	165	14 - 18h	50,8998	206	18 - 04h	46,7001
125	11 - 10h	43,8003	166	14 - 20h	49,6999	207	18 - 06h	47,6598
126	11 - 12h	45,6795	167	14 - 22h	48,8302	208	18 - 08h	46,0597
127	11 - 14h	41,9499	168	15 - 00h	50,5999	209	18 - 10h	43,5796
128	11 - 16h	41,4307	169	15 - 02h	49,8701	210	18 - 12h	40,5008
129	11 - 18h	46,5900	170	15 - 04h	50,7995	211	18 - 14h	46,4804
130	11 - 20h	46,7503	171	15 - 06h	47,1997	212	18 - 16h	49,0493
131	11 - 22h	49,0699	172	15 - 08h	45,0999	213	18 - 18h	44,2301
132	12 - 00h	48,0198	173	15 - 10h	44,0600	214	18 - 20h	44,7100
133	12 - 02h	46,5402	174	15 - 12h	43,9060	215	18 - 22h	44,6202
134	12 - 04h	47,7606	175	15 - 14h	43,7520	216	19 - 00h	45,9099
135	12 - 06h	51,9100	176	15 - 16h	43,5980	217	19 - 02h	45,3000
136	12 - 08h	51,8299	177	15 - 18h	43,4440	218	19 - 04h	45,5200
137	12 - 10h	50,8798	178	15 - 20h	43,2900	219	19 - 06h	45,3300
138	12 - 12h	49,4701	179	15 - 22h	43,4904	220	19 - 08h	44,9600
139	12 - 14h	50,0699	180	16 - 00h	46,0200	221	19 - 10h	45,1903
140	12 - 16h	49,5496	181	16 - 02h	45,7699	222	19 - 12h	47,0203
141	12 - 18h	46,5401	182	16 - 04h	44,8007	223	19 - 14h	48,8499
142	12 - 20h	47,6197	183	16 - 06h	50,0900			
143	12 - 22h	45,5101	184	16 - 08h	49,7396			
144	13 - 00h	46,5902	185	16 - 10h	47,1601			
145	13 - 02h	47,9901	186	16 - 12h	47,8197			
146	13 - 04h	48,4902	187	16 - 14h	46,0102			
147	13 - 06h	49,5600	188	16 - 16h	47,1702			
148	13 - 08h	49,3900	189	16 - 18h	48,8499			
149	13 - 10h	49,2198	190	16 - 20h	47,9200			
150	13 - 12h	47,4897	191	16 - 22h	48,0603			
151	13 - 14h	45,5507	192	17 - 00h	50,3893			
152	13 - 16h	50,8493	193	17 - 02h	45,6700			
153	13 - 18h	45,9899	194	17 - 04h	45,9200			
154	13 - 20h	44,9203	195	17 - 06h	46,0238			
155	13 - 22h	47,0003	196	17 - 08h	46,1275			
156	14 - 00h	48,8105	197	17 - 10h	46,2313			
157	14 - 02h	52,0999	198	17 - 12h	46,3350			
158	14 - 04h	51,4494	199	17 - 14h	46,4388			
159	14 - 06h	47,2399	200	17 - 16h	46,5425			
160	14 - 08h	46,7400	201	17 - 18h	46,6463			
161	14 - 10h	46,7699	202	17 - 20h	46,7502			
162	14 - 12h	45,7007	203	17 - 22h	48,4799			
163	14 - 14h	50,7296	204	18 - 00h	47,5199			

ANEXO B – CÓDIGOS

B.1 Código para rede neural LSTM para predição de porcentagem de Ferro

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Criado em Domingo 1 dezembro 16:43:44 2019

@author: santinobitaraes
"""


from keras.models import Sequential # modelo da rede neural
from keras.layers import Dense, Dropout, LSTM # tipos de camadas que serao
    usadas
from sklearn.preprocessing import MinMaxScaler # normalizados de dados
from keras.utils import plot_model # usado para plotar modelo da rede
    neural
import numpy as np # permite trabalhar com vetores e matrizes
import pandas as pd # permite manipulacao e analise de dados
import matplotlib.pyplot as plt # ferramenta para plotagem
import warnings

warnings.filterwarnings('ignore') # ignora avisos desnecessarios

# funcoes para calcular as metricas
def rmse(targets, forecasts):
    """
    Root Mean Squared Error
    :param targets:
    :param forecasts:
    :return:
    """
    if isinstance(targets, list):
        targets = np.array(targets)
    if isinstance(forecasts, list):
        forecasts = np.array(forecasts)
    return np.sqrt(np.nanmean((targets - forecasts) ** 2))

def mape(targets, forecasts):
    """
    Mean Average Percentual Error
    :param targets:
    :param forecasts:
```

```
40      :return:
41      """
42      if isinstance(targets, list):
43          targets = np.array(targets)
44      if isinstance(forecasts, list):
45          forecasts = np.array(forecasts)
46      return np.nanmean(np.abs(np.divide(np.subtract(targets, forecasts),
        targets))) * 100
47
48  def UStatistic(targets, forecasts):
49      """
50      Theil's U Statistic
51      :param targets:
52      :param forecasts:
53      :return:
54      """
55
56      if not isinstance(forecasts, (list, np.ndarray)):
57          forecasts = np.array([forecasts])
58      else:
59          forecasts = np.array(forecasts)
60
61      if not isinstance(targets, (list, np.ndarray)):
62          targets = np.array([targets])
63      else:
64          targets = np.array(targets)
65
66      l = forecasts.size
67      l = 2 if l == 1 else l
68
69      naive = []
70      y = []
71      for k in np.arange(0, l - 1):
72          y.append(np.subtract(forecasts[k], targets[k]) ** 2)
73          naive.append(np.subtract(targets[k + 1], targets[k]) ** 2)
74      return np.sqrt(np.divide(np.nansum(y), np.nansum(naive)))
75
76
77  # importa os dados
78  df = pd.read_excel('https://query.data.world/s/
        ubbxamr7q7tyi3fyg7wu664ggwq3ju')
79
80  # dados treinamento
81  qtde_valores_modelo = 200 # quantidade de valores da serie que serao usados
         para criar o modelo
82  base_treinamento = df.iloc[0:qtde_valores_modelo, 1:2].values # intervalo
        de dados para treinamento
```

```
83
84 # carrega os dados para testar a rede
85 base_teste = df.iloc[qtde_valores_modelo:224, 1:2].values # intervalo de
       dados para teste da rede neural
86 base_completa = df.iloc[0:224, 1:2].values # carrega a base completa
87
88
89 normalizador = MinMaxScaler(feature_range=(0, 1)) # configura o
       normalizador entre 0 e 1
90 base_treinamento_normalizada = normalizador.fit_transform(base_treinamento)
        # faz a normalizacao dos dados
91
92 data_rmse = [] # inicia a variavel erro rmse como nula
93 data_mape = [] # inicia a variavel erro mape como nula
94 data_u = [] # inicia a variavel de avaliacao u como nula
95 previsores = [] # variavel que recebe a entrada da rede neural
96 valor_real = [] # variavel que recebe o valor real de cada previsao que
       sera feita no treinamento
97
98 qtde_valores_passado = 1 # quantidade de valores do passado que sera usado
       para prever um passo a frente
99 qtde_neuronios = 28 # neuronios nas camadas ocutas
100
101 # cria uma base de dados com entradas (previsores) e tags (valor_real) para
        o treinar a rede.
102 for i in range(qtde_valores_passado, len(base_treinamento)):
103     previsores.append(base_treinamento_normalizada[i - qtde_valores_passado
       :i, 0])
104     valor_real.append(base_treinamento_normalizada[i, 0])
105
106 previsores, valor_real = np.array(previsores), np.array(valor_real) # passa
        os valores para array somente
107
108 # transforma os valores para um formato que keras nescessita
109 # previsores -> sao os valores que irao ser transformados
110 # previsores.shape[0] -> tamanho total da base de dados
111 # previsores.shape[1] -> qtde de valores que serao usados para prever
112 # 1 -> 1 entrada de dados
113 previsores = np.reshape(previsores, (previsores.shape[0], previsores.shape
       [1], 1)) # prepara os valores
114
115 regressor = Sequential() # cria a rede neural
116
117 # camada de entrada  e  camada oculta da rede neural
118 # units -> numero de neuronios ou celulas de memoria na camada
119 # return_sequences = True, -> indica que deve passar informacao pra frente
       para outras camadas
```

```
120 # input_shape -->  valores na primeira camada
121 regressor.add(LSTM(units = qtde_neuronios, return_sequences = True,
        input_shape = (previsores.shape[1], 1)))
122 regressor.add(Dropout(0.3)) # zera 30% dos valores, isso ajuda evitar
        overfiting
123
124 # segunda camada oculta
125 regressor.add(LSTM(units = qtde_neuronios))
126 regressor.add(Dropout(0.3))
127
128 # camada saida
129 # units -> uma saida
130 # activation -> funcao de ativacao linear
131 regressor.add(Dense(units = 1, activation = 'linear' ))
132
133 # configura o otimizador da rede neural
134 # optimizer -> otimizador usado pela rede
135 # loss --> usado para ajuste dos pesos
136 # metrics --> para entender os resultados
137 regressor.compile(optimizer = 'rmsprop', loss = 'mean_squared_error',
        metrics = ['mean_absolute_error'])
138
139 # plota a rede neural
140 plot_model(regressor, to_file = 'model.png', show_shapes = 'true',
        expand_nested = 'true')
141
142 # treinamento da rede
143 # bath_size -> batch indica quantos registros passamos para o algoritmo
        fazer a atualizacao dos pesos.
144 # Nesse caso, a base completa e dividida em batches com 2 registros e apos
        calcular o erro para os 2
145 # e que os pesos sao atualizados
146 regressor.fit(previsores, valor_real, epochs = 100, batch_size = 2)
147
148 # prepara as novas entradas para testar na rede neural
149 entradas = base_completa[len(base_completa) - len(base_teste) -
        qtde_valores_passado:]
150 entradas = entradas.reshape(-1, 1) # somente para ficar no formato do numpy
151 entradas = normalizador.transform(entradas) # normaliza entre 0 e 1
152
153 X_teste = [] # variavel que vai receber as novas entradas para rede neural
154
155 # prepara os valores para entrar na rede
156 for i in range(qtde_valores_passado, len(entradas)):
157     X_teste.append(entradas[i - qtde_valores_passado:i, 0])
158
159 X_teste = np.array(X_teste)
```

```
160
161 # prepara os valores para entrada de acordo com o pacote keras
162 X_teste = np.reshape(X_teste, (X_teste.shape[0], X_teste.shape[1], 1))
163
164 # realiza previsoes
165 previsoes = regressor.predict(X_teste)
166 previsoes = normalizador.inverse_transform(previsoes) # desnormaliza os
    valores
167
168 # calcula as metricas
169 rmse_v = rmse(base_teste, previsoes)
170 mape_v = mape(base_teste, previsoes)
171 u = UStatistic(base_teste, previsoes)
172
173
174 data_rmse.append(rmse_v)
175 data_mape.append(mape_v)
176 data_u.append(u)
177
178 # plota os resultados
179
180 # vetor para o eixo x do grafico
181 ttr = list(range(len(base_treinamento)))
182 tts = list(range(len(base_treinamento) - qtde_valores_passado, len(
    base_completa)))
183 tts1 = tts[qtde_valores_passado:]
184
185 # dados para o teste
186 data_test = base_completa[len(base_completa) - len(base_teste) -
    qtde_valores_passado:]
187 data_test = data_test.transpose().tolist()
188 data_test = data_test[0]
189
190 plt.rcParams.update({'font.size': 20}) # configura a fonte do texto do
    grafico
191 fig, ax = plt.subplots(nrows = 1, ncols = 1, figsize = [16, 6]) # tamanho
    da figura
192 plt.plot(tts, data_test, color = 'red', label = 'Valor Real', marker = "+")
     # plota os dados reais
193 plt.plot(tts1, previsoes, color = 'black', label = 'Previsoes', marker = "X
    ") # plota os dados preditos
194 plt.title('Previsao de % de Fe usando RNA LSTM')
195 plt.xlabel('Tempo(h)')
196 plt.ylabel('% de Fe')
197 plt.legend()
198 plt.savefig('prev_LSTM_Fe.eps', format = 'eps', bbox_inches='tight') #
    salva em uma figura
```

```
plt.show()

# apresenta as metricas
rmse_v = rmse(base_teste, previsoes)
mape_v = mape(base_teste, previsoes)
u = UStatistic(base_teste, previsoes)
rows = []
rows.append([rmse_v, mape_v, u])
pd.DataFrame(rows, columns=['RMSE', 'MAPE', 'U'])
```

Código B.1 – Código usado para criar a rede neural para predição de Ferro

B.2 Código série temporal nebulosa para predição de porcentagem de Ferro

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Criado em Domingo 2 dezembro 18:40:48 2019

@author: santinobitaraes
"""


# instalacao da biblioteca pyFTS
!pip3 install -U git+https://github.com/PYFTS/pyFTS

# importacoes iniciais
from pyFTS.common import Membership # importa as funcoes de pertinencia
from pyFTS.models import chen, cheng, hofts, pwfts # importa os modelos
    pyFTS
from pyFTS.partitioners import Grid, FCM, CMeans, Entropy # importa os
    particionadores
from pyFTS.benchmarks import Measures # pacote de metricas do pyFTS
from pyFTS.common import Util # utilidades do pyFTS

import numpy as np  # permite trabalhar com vetores e matrizes
import pandas as pd # permite manipulacao e analise de dados
import matplotlib.pyplot as plt # ferramenta para gerar graficos
import warnings
warnings.filterwarnings('ignore')


plt.rcParams.update({'font.size': 16}) # parametro para fonte dos graficos

# importacao dos dados
df = pd.read_excel('https://query.data.world/s/
    ubbxamr7q7tyi3fyg7wu664ggwq3ju')
```

```
31 dados = df.iloc[0:224, 1:2].values
32 dados = dados.flatten().tolist()
33 fim = len(dados)
34
35 # dados para treino
36 qtde_dt_tr = 200 # sera usado os 200 primeiros valores para treinamento da
       rede
37 dados_treino = dados[:qtde_dt_tr]
38 ttr = list(range(len(dados_treino))) # cria apenas um vetor para plotar
39
40 ordem = 1 # ordem do modelo, indica quantos ultimos valores serao usados
41
42 # dados para teste da rede
43 dados_teste = dados[qtde_dt_tr - ordem:224]
44 tts = list(range(len(dados_treino) - ordem, len(dados_treino) + len(
       dados_teste) - ordem))
45
46 # visualiza os dados importados
47 '''
48 fig, ax = plt.subplots(nrows = 1, ncols = 1,figsize = [16, 5])
49 plt.plot(dados, color = 'blue', label = 'Dados', marker = "o")
50 plt.title('Dados Amostrados')
51 plt.xlabel('Tempo(h)')
52 plt.ylabel('% de Fe')
53 plt.legend()
54 plt.savefig('dados_gerais_Fe.eps', format = 'eps', bbox_inches = 'tight')
55 plt.show()
56 '''
57
58 # particiona os dados
59
60 # Grid.GridPartitioner, Entropy.EntropyPartitioner, FCM.FCMPartitioner,
       CMeans.CMeansPartitioner
61 # Particionamento do Universo de Discurso
62 # data -> recebe os dados que serao usados no particionador
63 # npart -> numero de particoes que os dados serao divididos
64 # func -> funcao de pertinencia
65 # Membership.trimf, Membership.trapmf, Membership.gaussmf]
66
67 particionador = Grid.GridPartitioner(data = dados_treino, npart = 9, func =
        Membership.trimf)
68
69 # plota como os dados ficaram particionados
70 fig, ax = plt.subplots(nrows = 1, ncols = 1, figsize = [16, 4])
71 particionador.plot(ax)
72
73
```

```
74 # plotar os tipos de particionadores somente
75
76 '''
77 metodos = [Grid.GridPartitioner, Entropy.EntropyPartitioner, FCM.
       FCMPartitioner, CMeans.CMeansPartitioner]
78
79
80 fig, ax = plt.subplots(nrows=4, ncols=1, figsize=[15, 10])
81
82 for contador, metodo in enumerate(metodos):
83     particionador = metodo(data = dados_treino, npart = 9)
84     particionador.plot(ax[contador])
85
86 plt.tight_layout()
87 plt.savefig('metodos_part_FTS.eps', format = 'eps', bbox_inches = 'tight')
88 '''
89
90 # plota apenas as funcoes de pertinencia
91 '''
92 functions = [Membership.trimf, Membership.trapmf, Membership.gaussmf]
93
94 fig, ax = plt.subplots(nrows = 3, ncols = 1, figsize = [15, 10])
95
96 for contador, mf in enumerate(functions):
97     particionador = Grid.GridPartitioner(data = dados_treino, npart = 9,
       func = mf)
98     particionador.plot(ax[contador])
99
100 plt.tight_layout()
101 plt.savefig('func_pert_FTS.eps', format = 'eps', bbox_inches = 'tight')
102
103 '''
104
105 # Cria um modelo vazio
106 modelo = hofts.HighOrderFTS(partitioner = particionador, order = ordem)
107 #modelo = hofts.WeightedHighOrderFTS(partitioner = particionador, order =
       ordem)
108 #modelo = pwfts.ProbabilisticWeightedFTS(partitioner = particionador, order
        = ordem)
109
110
111
112 # Todo o procedimento de treinamento e feito pelo metodo fit
113 modelo.fit(dados_treino)
114
115 # Todo o procedimento de inferencia e feito pelo metodo predict
116 predicoes = modelo.predict(dados_teste)
```

```
117
118 # insere um espaco vazio no vetor para se encaixar melhor nos graficos
119 for k in range(modelo.order):
120     predicoes.insert(0, None)
121
122
123 # metricas
124
125 rows = []
126 rmse = []
127 mape = []
128 u = []
129 rmse, mape, u = Measures.get_point_statistics(dados_teste, modelo)
130
131 rows.append([ rmse, mape, u])
132
133 tts1 = list(range(len(dados_treino) - ordem, len(dados_treino) + len(
    dados_teste)- ordem))
134
135 fig, ax = plt.subplots(nrows = 1, ncols = 1, figsize = [18, 6])
136 plt.plot(tts, dados_teste, color = 'red', label = 'Valores Reais', marker =
     "+")
137 plt.plot(tts1, predicoes, color = 'black', label = 'Previsoes', marker = "X
    ")
138 plt.title('Previsao de % de Fe usando FTS')
139 plt.xlabel('Tempo(h)')
140 plt.ylabel('% de Fe')
141 plt.legend()
142 plt.savefig('prev_FTS_Fe.eps', format = 'eps', bbox_inches='tight')
143 plt.show()
144
145
146 pd.DataFrame(rows, columns=['RMSE', 'MAPE', 'U'])
```

Código B.2 – Código usado para criar a rede neural para predição de Ferro