PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Inteligência Artificial e Aprendizado de Máquina

Marina Micas Jardim

REDE NEURAL COM PREDIÇÕES DE VOLUME ÚTIL DO RESERVATÓRIO DE FUNIL (RJ) BASEADO EM VARIÁVEIS CLIMÁTICAS

Rio de Janeiro
Abril de 2022

Marina Micas Jardim

REDE NEURAL COM PREDIÇÕES DE VOLUME ÚTIL DO RESERVATÓRIO DE FUNIL (RJ) BASEADO EM VARIÁVEIS CLIMÁTICAS

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Inteligência Artificial e Aprendizado de Máquina, como requisito parcial à obtenção do título de *Especialista*.

Rio de Janeiro

Abril de 2022

SUMÁRIO

1. Introdução

Um dos elementos fundamentais para sobrevivência humana é o consumo de água potável. E um dos grandes constructos do homem foi a criação de enormes reservatórios de água para suprir tanto de sua necessidade vital quanto a de lazer aumentando até a economia local.

Porém o represamento de água não garante que haverá volume útil para munir a população desse bem precioso. Dessa forma se fazem necessários planejamentos acerca do reservatório, onde dados meteorológicos são fundamentais para decisões importantes como por exemplo, ao uso do volume morto, que podem impactar na vida aquática que mantém a salubridade da água.

De acordo com NAGASELVI e DEEPA (2015) a experiência com variáveis meteorológicas tem sido amplamente utilizadas com sucesso em Redes Neurais Artificiais (ANNs), prevendo como por exemplo a quantidade de chuva sendo imprescindível em alertas para enchentes e na identificação de nevoeiros com muita utilidade para aeroportos.

2. Contextualização

De acordo com G1 (2014) o volume morto ou reserva técnica de um reservatório são “milhões de metros cúbicos de água situado abaixo das comportas das represas” e que nunca deveria atender a população, pois especialistas alegam contaminação com a inclusão de metais pesados impossibilitando o consumo, além de afetar a vida aquática que mantém as águas do reservatório potável.

Para auxiliar em planejamentos do volume útil de reservatórios, a inteligência artificial pode ser treinada com dados dos últimos 10 ou 20 anos para poder estimar variáveis climáticas que podem detectar a tal ocorrência com predições automatizadas utilizando a Feed-forward Neural Network, que no dicionário técnico de tradução trata-se de uma Rede Neural de Alimentação Direta ou Rede Neural de Alimentação Antecipada, da sigla em inglês FFNN. Uma FFNN, de acordo com a WIKIPEDIA (2022) apud ZELL (1994), “é uma rede neural artificial onde as conexões entre os nós não formam um ciclo”. Com as informações da entrada passando pelos

nós (camadas) ocultas indo à saída. Na pesquisa de NAGASELVI e DEEPA (2015) ao utilizar a FFNN, eles obtiveram resultados satisfatórios com dados meteorológicos, mesmo com um dataset pequeno e com dados ausentes.

3. Descrição do Problema e da Solução Proposta

Na entrevista realizada pela FOLHA DE SÃO PAULO (2015) com Carlos Bocuhy, Presidente do Instituto Brasileiro de Proteção Ambiental - o Proam, parte da resposta dele sobre a importância do Cantareira sair do volume morto foi: “O volume morto é uma reserva técnica e não deveria ser explorado. Ele é uma margem de segurança para deixar o reservatório equilibrado, sob o ponto de vista da diluição de poluentes e da recomposição do ecossistema.” E segundo WATANABE (2015) relata que “o repovoamento da fauna e da flora pode demorar muitos anos para acontecer. Enquanto isso, a água fica sem a sua salubridade, mesmo que potável, e não é boa para beber e cozinhar alimentos.”

De acordo com esses conhecimentos, nota-se como pode ser problemático um reservatório chegar a esses níveis, além de também impactar financeiramente ao utilizar esse volume. No caso do sistema Cantareira, a instalação das bombas custou R$ 80 milhões, pois esse volume é armazenado abaixo do ponto de captação impossibilitando retirarem só por gravidade.

Dessa forma, o objetivo da pesquisa foi verificar se há padrões climáticos na região do reservatório, utilizando a FFNN para prever variações do volume útil para que possam ser planejadas de antemão as devidas decisões que impactam em uma possível utilização do volume morto.

4. Coleta de Dados

A região escolhida para o estudo foi o município do Rio de Janeiro por sua alta densidade populacional. E de acordo com a notícia do G1 de 2015 os seus principais reservatórios são: Paraibuna (SP), Santa Branca (SP), Jaguari (SP) e Funil (RJ) que estavam na época com níveis baixíssimos de água.

Para escolher as estações meteorológicas próximas aos reservatórios da pesquisa, foi utilizado alguns trechos de códigos com o recurso em Python do Meteostat para obter as estações no raio de até 32 km de distância do local de estudo, essa distância segundo RAY (2017) quando não se tem um mapa da zona climática ou outras medidas é o mais adequado a ser adotado. No Instituto Nacional de Meteorologia (INMET) estavam disponíveis apenas uma das três estações meteorológicas que foram as que o Meteostat retornou. E de acordo com o raio de 32 km estabelecido anteriormente, a estação meteorológica selecionada foi a de Resende (RJ) e o reservatório foi o de Funil (RJ). Em ambos os dados foram extraídos no formato CSV.

Os dados analisados da estação meteorológica de Resende compreendem os anos de 1983 a 2022, com a obtenção dos dados pelo INMET realizada através do link <https://bdmep.inmet.gov.br/> no dia 29 de março de 2022, com dados diários e convencionais para contemplar o efeito da evaporação da água. A tabela 1 dispõe dos campos utilizados após a exploração dos dados, alguns campos foram escolhidos pelo fato da evaporação estar correlacionada a temperatura do ar, umidade do ar, velocidade do vento e da radiação solar, onde tais variáveis climáticas impactam no volume final contido no reservatório.

Nome da coluna	**Coluna renomeada**	**Tipo**
Data Medicao	data	OBJECT
EVAPORACAO DO PICHE, DIARIA(mm)	evap	FLOAT64
INSOLACAO TOTAL, DIARIO(h)	inso	FLOAT64
PRECIPITACAO TOTAL, DIARIO(mm)	prec	FLOAT64
TEMPERATURA MEDIA COMPENSADA, DIARIA(°C)	temp	FLOAT64
UMIDADE RELATIVA DO AR, MEDIA DIARIA(%)	umid	FLOAT64
VENTO, VELOCIDADE MEDIA DIARIA(m/s)	vent	OBJECT

Tabela 1 – Colunas dos dados da estação meteorológica de Resende (RJ)

No dia 3 de abril de 2022 foi coletado os dados do reservatório através do link <https://www.ana.gov.br/sar0/MedicaoSin?dropDownListEstados=20&dropDownListReservatorios=19093&dataInicial=01%2F01%2F1993&dataFinal=05%2F12%2F2017

&button=Buscar>. A aquisição desses dados foram pelo Sistema de Acompanhamento de Reservatórios (SAR), com dados diários entre o período de 1993 a 2017, na tabela 2 dispõe os campos utilizados durante a exploração dos dados.

Nome da coluna	Coluna renomeada	Tipo
Data da Medição	data	OBJECT
Volume Útil (%)	volu	OBJECT

Tabela 2 – Colunas dos dados do reservatório de Funil (RJ)

Todos os locais de coleta dos dados: INMET, SAR e Meteostat disponibilizam os dados publicamente. Sendo no Meteostat necessário mencionar o autor na obra com a licença CC BY-NC 4.0, e trata-se de uma iniciativa independente que depende de doações, patrocínios e outras fontes de rendimento.

Os dados da estação meteorológica e do reservatório foram pareados e gerado um único dataset com as datas correspondentes. Sendo assim, neste estudo acerca dos dados coletados, tem-se por objetivo a análise de 7 variáveis e 1 variável destino. Onde a variável destino é o volume útil, em que o modelo de aprendizagem de máquina escolhida realizou as previsões.

5. Processamento/Tratamento de Dados

O primeiro passo foi decidir o ambiente a ser utilizado e o escolhido foi o Colaboratory, conhecido mais como Colab, que é um serviço da Google permitindo execuções na linguagem Python utilizando gratuitamente recursos como GPUs. Uma GPU de acordo com a Intel “é um processador composto por muitos núcleos menores e mais especializados”, onde para atividades relacionadas a inteligência artificiais propõem um cenário ideal, isso tudo bastando ter uma conta gratuita na Google.

Após ter criado uma conta google, caso não a tenha, o acesso pode ser diretamente pelo Drive da Google ou na página do Colab e independente do caminho a ser percorrido deverá instalar o Colab, porém essa instalação é no ambiente nuvem da própria Google e não localmente em seu computador.

Ao abrir um novo arquivo Jupyter notebook no Colab, antes de iniciar o tratamento dos dados, foi necessário utilizar o recurso do Meteostat para localizar as estações com no máximo 32 km da região escolhida. Na figura 1 é apresentado o código utilizado e devidamente comentado explicando o que foi realizado nesta etapa.

```
# Dependências do Meteostat https://github.com/meteostat
!pip install meteostat
from datetime import datetime
from meteostat import Stations

# criação de um dicionário que guarda o nome de cada reservatório e de cada coordenada correspondente a ele
dic_coordenadas_reservatorios = {"FUNIL":  (-22.5311, -44.5681), "JAGUARI": (-23.1956, -46.0075), "PARAIBUNA": (-23.415, -45.6025), "SANTA BRANCA":
(-23.3743, -45.8727)}

# criação de um dicionário vazio que guardará cada nome de dados da estação mais próxima relativa a um reservatório
dic_reservatorios_data_estacoes = {}

# iteração sobre cada nome de reservatório e cada coordenada guardando os nomes em variáveis (chave e valor)
for (chave, valor) in dic_coordenadas_reservatorios.items():

    # separação de cada coordenada do reservatório em variáveis diferentes
    lat_reservatorio = valor[0]
    long_reservatorio = valor[1]

    # criação do objeto Stations para procura e retorno de uma ou mais estações
    stations = Stations()

    # captura de 6 estações perto do reservatório, passando as coordenadas dele
    stations = stations.nearby(lat_reservatorio, long_reservatorio).fetch(6)

    for index, linha in stations.iterrows():

        # extração do id, nome da estação e distância de no máximo 32km em relação ao reservatório
        id, nome, dist = linha.wmo, linha["name"], linha['distance']
        if (dist < 32000.0):
            print('Estação meteorológica: ' + nome + ' | Id: ' + id + ' | Reservatório: ' + chave + ' | Distância: ' + str(round(dist / 1000.0)) + 'km')
        else:
            pass
```

Figura 1 – Código de localização das estações próximas aos reservatórios do Rio de Janeiro (fonte: figura da autora via recurso do Meteostat)

O código acima retornou três estações meteorológicas próximas aos reservatórios que abastecem o Rio de Janeiro, que foram:

- Resende (RJ) – ID: 83738 com 12km do reservatório de Funil (RJ)
- Resende (RJ) – ID: 86874 com 15km do reservatório de Funil (RJ)
- São Luís do Paraitinga/Rio Afonsos – ID: 86912 com 28km do reservatório de Paraibuna (SP)

Porém no INMET, apenas a estação de Resende do ID: 83738 estava disponível para coleta dos dados. E para esses dados serem visíveis ao Jupyter notebook que foi criado, foi realizado o upload dos dataset para o Drive do Google e para que seja possível que o Colab consiga ver seus dados no Drive é necessário montar o Google Drive na máquina virtual usando o trecho visualizado na figura 2 a seguir.

```
# Importação dos arquivos do seu Google Drive pelo Colab
from google.colab import drive
drive.mount('/content/drive')
```

Figura 2 – Montagem do Google Drive
(fonte: figura da autora)

5.1 Estação meteorológica de Resende (RJ)

Os dados para a estação meteorológica de Resende foram no período entre 17/09/1983 a 18/03/2022 totalizando 14.062 dados para obter chances de continuar com uma quantidade de dados considerável após seu tratamento. O tipo de dados foi o diário, pois o local em que os dados do reservatório foram extraídos após essa etapa possui apenas essa possibilidade, e a fim de manter a equidade dos dados para uni-los em um único dataframe esse tipo de dados coletado foi o adequado.

A próxima etapa é a leitura desses dados e como o arquivo extraído possui um formato csv, foi utilizado o método read_csv do Pandas. O Pandas é uma ferramenta de código aberto que facilita manipular os dados com a linguagem Python. Para usá-lo além de conhecer seus recursos, deve-se importar para dentro do Colab com o trecho: import pandas as pd. Após a leitura dos dados pelo *read_csv*, o Colab vai exibir uma tabela, que se um dos últimos registros ou então os iniciais estiverem com ausência de algum dado com o termo *Null* ou *NaN* ele será rapidamente notado. Mas isso não deve ser parâmetro em determinar que seus dados estão perfeitos para uso, portanto foi utilizado o código disponibilizado por OLIVEIRA (2019) no Kaggle para detalhar os dados ausentes presentes em cada coluna, como visto na figura 3.

```
total = train_df.isnull().sum().sort_values(ascending=False)
percent_1 = train_df.isnull().sum()/train_df.isnull().count()*100
percent_2 = (round(percent_1, 1)).sort_values(ascending=False)
missing_data = pd.concat([total, percent_2], axis=1, keys=['Total', '%'])
missing_data.head(5)
```

Figura 3 – Detalhar dados ausentes
(fonte: OLIVEIRA, 2019)

Após utilizar a codificação da figura 3 que usa métodos como o *isnull* do Pandas para detectar os dados ausentes entre outros cálculos para resultar o percentual de cada coluna, a coluna que retornou mais falhas foi a *VENTO, VELOCIDADE MEDIA DIARIA(m/s)* com 45% de dados faltantes. E antes de iniciar o tratamento desse e das demais que também retornaram tais ausências, foi utilizado o método *rename* do Pandas para facilitar todo o processamento. Onde a coluna *VENTO, VELOCIDADE MEDIA DIARIA(m/s)* passou a ser chamada apenas *vent*, seguindo a mesma lógica para as restantes mantendo as quatros letras iniciais de cada variável.

Com as colunas renomeadas, o método *dropna* do Pandas retirou os dados faltantes, porém o índice retornou uma ordenação estranha trazendo confusão dos dados totais, para isto se usou o método *reset_index* do Pandas para reiniciar o índice em zero novamente. Além desse tratamento foi necessário a conversão de duas colunas que tinham tipos de dados diferentes das que deveriam possuir, essa percepção foi realizada com o método *info* do Pandas que exibe além da quantidade de dados um comparativo de ausentes. As colunas foram a *data* e o *vent*, que estavam com o tipo *object*, que é vista como um tipo de dados texto ou string, portando nesses campos foram utilizados o método *assign* do Pandas para a conversão. Onde para a data foi combinada com o método *to_datetime* do Pandas para converter para um tipo data, e para o *vent* que é um tipo numérico *float* (com “vírgulas”), foi utilizado o *astype(float)* combinado com o *str.replace('.', '')* para ajustar a quantidade de pontos que havia antes, e ambos os métodos pertencentes também ao Pandas.

Para finalizar o tratamento dos dados da estação meteorológica de Resende, foi realizado um cálculo com o método *len* para totalizar e o método *round* para arredondar as casas decimais para nenhuma casa decimal e por fim o método *str* para converter em texto para realizar a junção do valor calculado juntamente com o símbolo de porcentagem ao lado, e todos esses métodos do Pandas. O retorno de perdas detectado em relação a quantidade inicial de dados foi de 57%, exibindo nessa etapa um novo intervalo de período de datas, que foi utilizado para pareamento com os dados do reservatório de Funil.

5.2 Reservatório de Funil (RJ)

Para os dados do reservatório foi realizado as mesmas etapas realizadas nos dados da estação meteorológica:

- Lido com o método *read_csv*, pois o arquivo também apresentava a extensão *csv*;
- Detalhado os dados ausentes com o código do OLIVEIRA (2019);
- Renomeado somente as colunas que seriam utilizadas com o método *rename*, onde o campo *Data da Medição* que também foi renomeado para *data*, a fim de usá-lo como a coluna “coringa” que irá unir os dois dataframes;
- E após a renomeação foi feito uma etapa que não foi necessária na estação meteorológica, pois nessa foi possível selecionar as variáveis desejadas antes da coleta dos dados. Portanto para o reservatório foi utilizado o método *drop* do Pandas para eliminar as colunas que não serão utilizadas na pesquisa. Com isso os dados que tinham falhas foram retirados fazendo com que a etapa do método *dropna* não fosse necessária;
- Foi visualizado os tipos das colunas com o método *info* para detectar se precisava realizar a conversão, e para as duas colunas a serem utilizadas teve a necessidade de converter. Uma delas a *data*, foi realizado da mesma forma que da estação meteorológica, e para a coluna *volu*, que é o volume útil, foi o mesmo utilizado para o *vent* com exceção do método *replace* que ao invés de deixar vazio, foi optado em trocar a vírgula para ponto, pois nesses os dados não tinham várias vírgulas espalhadas como no *vent* que havia vários pontos entre os números.

Não houve necessidade de realizar o cálculo de perdas de dados após o tratamento, pois nas duas colunas a serem utilizadas não existiam dados faltantes, permanecendo assim com a mesma quantidade dos dados iniciais.

5.3 Unindo os dados da estação meteorológica de Resende com o reservatório de Funil

Terminado o tratamento das duas fontes de dados: da estação meteorológica e do reservatório, chegou o momento de torná-los únicos, para tal foi utilizado o método *merge* do Pandas com o argumento *how* contendo o valor *inner*, e o argumento *on* com o valor *data*, pois a intenção é unir a partir das datas que coincidirem em ambos dataframes sem duplicatas. E para conferir se de fato não resultou em dados duplicados, foi utilizado o método *duplicated* do Pandas, e de fato não houve existências de dados duplicados.

As próximas etapas foram verificar se os tipos de dados ainda continuavam com os que tinham sido convertidos na etapa do tratamento individual dos dados, seguido com o cálculo da perda de dados após a junção de ambos os dataframes, e retornou uma perda de 58%, totalizando 5.891 de dados finais para a etapa da aprendizagem de máquina.

Depois foi exportado o dataframe que tinha tanto os dados da estação meteorológica quanto do reservatório em três formatos: *csv*, *xlsx* e *pkl*. Utilizado respectivamente os seguintes métodos do Pandas: *to_csv*, *to_excel* e *to_pickle*.

No gráfico 1 abaixo, todas as barras indicam a relação da variável evaporação (evap) com a precipitação (prec), ou seja, as barras azuis indicam a presença de chuva em relação ao quantitativo da evaporação, e o mesmo é apresentado nas barras vermelhas, onde nessas marcam a ausência da chuva. Portanto percebe-se que no ano 2015 há uma queda grande no volume útil, corroborando com a notícia do G1 (2015) onde relata que o reservatório de Funil chega ao seu nível mais baixo. Chegando próximo a um trecho detectado com ocorrências frequentes na ausência da chuva e com evaporações altas.

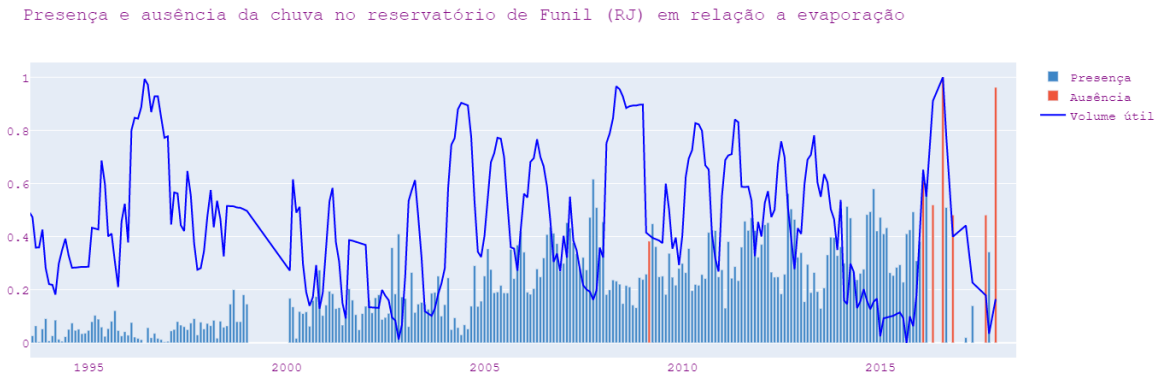

Gráfico 1 – Variáveis climáticas da precipitação e evaporação ao reservatório de Funil (RJ) (fonte: figura da autora)

Outro ponto interessante é que mesmo havendo concentrações massivas de chuvas nesse período isso não impediu o volume útil do reservatório cair drasticamente. Pode ser que a variável evaporação não apresente uma correlação determinística em relação ao volume útil, ou então os arredores da localidade do reservatório absorveu a maior parte da água advinda dessas precipitações.

Para a criação do gráfico foi utilizado recursos do matplotlib, seaborn e plotly. E necessitou importação do matplotlib e do plotly e da instalação do seaborn e plotly. Conforme visualizado na figura 4.

```
import matplotlib.pyplot as plt
!pip install seaborn
!pip install plotly
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

Figura 4 – Recursos na criação do gráfico (fonte: figura da autora)

Na figura 5 é exibido o código, devidamente comentado, utilizado para a criação do gráfico. Esse conjunto de recursos proporcionou interatividade ao gráfico sendo possível visualizar o quantitativo da evaporação ao passar o mouse por cima e podendo realizar zoom além de outras funcionalidades.

```
# LEGENDA:
# prec = precipitação mm    | volu = volume útil %
# ------------------------------------------------------------------------------------------
# usar uma das variáveis abaixo na linha do comentário => testar variável
# temp = temperatura do ar °C | umid = umidade relativa % | vent = velocidade do vento m/s
# evap = evaporação mm        | inso = insolação total h

# agrupando a data para um período mensal
df_mensal = df_estacao_Resende_E_df_reservatorio_Funil.groupby(pd.Grouper(key='data', freq='M')).agg('mean')
df_mensal.dropna(inplace=True)
series_data = df_mensal.index.astype('string')
series_data_presente = df_mensal[df_mensal['prec'] > 0].index.astype('string')
series_data_ausente = df_mensal[df_mensal['prec'] == 0].index.astype('string')

# normalizando os valores
df_mensal_normal = (df_mensal-df_mensal.min())/(df_mensal.max()-df_mensal.min())

# criação da área do gráfico
fig = make_subplots()

# criação das barras com presença de precipitação/chuva
fig.add_trace(go.Bar(name='Presença', x=series_data_presente, y=df_mensal_normal[df_mensal_normal['prec'] > 0]['evap'], marker=dict(color='#3d85c6'))) # testar variável

# criação das barras com ausência de precipitação/chuva
fig.add_trace(go.Bar(name='Ausência', x=series_data_ausente, y=df_mensal_normal[df_mensal_normal['prec'] == 0]['evap'])) # testar variável

# criação das linha do volume útil
fig.add_trace(go.Line(name='Volume útil', x=series_data, y=df_mensal_normal['volu'], marker=dict(color='#0000ff')))

# título do gráfico
fig.update_layout(title="Presença e ausência da chuva no reservatório de Funil (RJ) em relação a evaporação", font=dict(family="Courier New, monospace", size=15, color="Purple"))

# exibição do gráfico
fig.show()
```

Figura 5 – Codificação para criação do gráfico
(fonte: figura da autora)

6. Análise Exploratória dos Dados e Análise com Modelos de Machine Learning

Assim como na parte do tratamento dos dados, nessa etapa do modelo a ser utilizado para a aprendizagem de máquina também foi utilizado o Colab, pois exatamente nessa parte em que a ferramenta adotada fez jus a sua escolha. E o modelo adotado foi a Rede Neural de Alimentação Direta, a FFNN.

Inicialmente foi lido o novo dataframe da junção das variáveis da estação meteorológica com o reservatório através do método *read_excel* do Pandas, pois ao ler o dataframe no formato de *csv*, no momento da conversão da data para data *Unix* ele apresentou erros. Na conversão para a data *Unix*, que é o formato em que a aprendizagem de máquina compreende, foi utilizado o seguinte trecho de código presente na figura 6.

```
# conversão da data para um formato que a machine learning compreenda com data unix
df_estacao_reservatorio["DataUnix"] = df_estacao_reservatorio["data"].values.astype(np.int64) // 10 ** 9
df_estacao_reservatorio
```

Figura 6 – Conversão da data para data Unix
(fonte: figura da autora)

Onde no código da figura 6 exibe que a coluna *data* é um valor do Numpy como um tipo inteiro que para convertê-lo para um tempo *Unix* foi feita uma divisão inteira dele por 10 elevado a 9 (1 bilhão), por se tratar de um tipo de data que representa uma cadeia de caracteres de um determinado horário ocorrido em

nanossegundos. E em seguida foi atribuído a uma nova coluna do dataframe com o nome de *DataUnix*. O próximo passo foi retirar a coluna *data* antiga com o método *drop* e manter apenas a convertida, renomeando-a em seguida para apenas *data* com o método *rename*. E para verificar se a conversão foi realizada com sucesso, foi utilizado um conversor online (https://www.epochconverter.com/) para verificar se a data convertida bate com a data original.

Neste momento os dados ficaram prontos para a aprendizagem de máquina, logo utilizar o método *list* do Python facilitou para pegar as colunas rapidamente do dataframe e atribuir a varíavel *X* em maiúsculo transformando-o em um *array* do Numpy utilizando o método *value*. Essa variável *X* representou as variáveis de entrada da rede neural, e o *y* em minúsculo foi a variável de saída que também foi utilizado o recurso do método *value*, a variável y foi o retorno do aprendizado da rede neural.

Para a definição do tamanho da amostra foi utilizado o método *train_test_split* do sklearn, que se trata de uma biblioteca Python para aprendizagem de máquina e que realiza a divisão do dataset em dois conjuntos de dados: treino e teste. Para tanto, os argumentos utilizados foram realizar um “chaveamento” entre as variáveis *X* e *y* (onde ambas já foram criadas anteriormente como um *array* do Numpy) e dessa forma foi incluindo também a quantidade a essas variáveis através do argumento *test_size*, que ao adotar o valor 0.25 indica-se que 25% foi utilizado para as variáveis de teste e 75% para as variáveis de treino, e por fim, a “semente” que é um valor randômico da amostra argumentado pelo *random_state* que evita o “vício” da rede neural, pois se enviar os dados sequenciais ao detectar algum padrão prejudica sua aprendizagem. Aonde foram retornado as seguintes amostras das variáveis:

- Quantidade de dados para o treino do X (*X_treino*): 4419
- Quantidade de dados para o teste do X (*X_teste*): 1473
- Quantidade de dados para o treino do y (*y_treino*): 4419
- Quantidade de dados para o treino do y (*y_teste*): 1473

A próxima etapa foi criar o escalonamento dos dados instanciando o método *StandardScaler* do sklearn para a variável *escalador* que foi utilizado para realizar a normalização dos dados em conjunto com os métodos *fit_tranform* e *transform*, onde foram padronizados as variáveis *X_treino* e *X_teste* respectivamente. O *fit_tranform* transformou o *array* do Numpy com valores pequenos normalizando os valores entre 0 e 1 para diminuir os ruídos e a convergência na rede neural, pois a rede neural funciona melhor com este tipo de dado, até poderia ser utilizado em seu formato normal, mas para evitar conflitos desnecessários se realiza essas normalizações. Porém as variáveis *y_treino* e *y_teste* não podem ser normalizadas, para que possam representar a realidade de um teste para a rede neural mantendo, portanto, seus dados em formato bruto da forma que vieram no dataset unificado, mas com a *data Unix*.

Com as variáveis de treino e teste criadas e normalizadas, a variável *parar_rede_neural* utilizada no método *EarlyStopping* do Tensorflow com o Keras, em que o Tensorflow é um código aberto para aprendizagem de máquina que juntamente com o API do Keras, onde nesse trata-se de uma biblioteca de rede neural em Python, que facilitou algumas operações do Tensorflow. Como por exemplo, ao que foi feito com a variável *parar_rede_neural*, que mata a rede neural quando perceber que não ocorre mais variações em seu aprendizado, pois já alcançou o melhor resultado possível para aquele treino específico. E a importância deste procedimento é reduzir o ciclo da aprendizagem ao perceber que não há mais melhoria no rendimento da rede neural, controlando dessa forma a ocorrência de overfitting, onde a rede neural decora o treino sem de fato aprender, dificultando dessa forma quando for testada sua aprendizagem. Contendo como argumentos:

- monitor com o valor *val_loss*, que determina a perda na qualidade dos dados como uma métrica para verificar se a rede neural está indo bem ou não;
- mode com o valor *min* que monitora a perda;
- verbose com o valor *1* que determina a visualização da aprendizagem da rede com uma barra que descreve o progresso;

- patience com o valor *21* que determina a quantidade de ciclos que a rede neural terá, mas se já foi detectado uma reta considerável da aprendizagem não precisa ser alterada.

Para a variável *velocidade_aprendizagem* foi utilizado o método *Nadam* do Tensorflow com o Keras, onde seu argumento *learning_rate* determina a velocidade com que a rede neural aprende, e quanto menor o número mais lento é seu aprendizado. E é um dos hiper parâmetros, ou seja, um valor que deve ser ajustado para que a aprendizagem de máquina realize treinos para aproximar cada vez mais em acertos.

O início da configuração da aprendizagem de máquina propriamente dita com a FFNN, foi ao utilizar o método *Sequential* do Tensorflow com o Keras, que foi instanciada a variável *modelo*, que em conjunto com os métodos *add* e *Dense* foram criadas as camadas de entrada, saída e as ocultas contendo em seus argumentos a quantidade de neurônios, ou nódulos, e o tipo de ativação. O tipo de ativação escolhida foi a linear retificada - *relu*, que ao utilizá-la nas camadas do modelo proporciona eficiência na aprendizagem. As camadas de entrada foram as variáveis do *X_treino* que são as variáveis climáticas incluindo a *data Unix*, nas camadas ocultas podem ter uma diversidade de neurônios sem uma receita padrão e a camada de saída foi o *y_treino*, que foi a variável do volume útil (volu) e é a variável que foi desejada descobrir o quanto a rede neural aprendeu. No método *compile* que se trata da compilação do modelo FFNN, foi utilizado a variável *velocidade_aprendizagem* no argumento *optimizer*, determinando se a máquina terá um aprendizado rápido ou lento e no argumento *loss* foi utilizado o Erro Quadrado Médio – *mse*, que se trata das médias de diferenças quadráticas entre o valor previsto e o real, onde o resultado é sempre positivo evitando erros maiores.

No método *fit* foi realizado o ajuste do modelo neural, usando a variável x do X_treino e o y de y_treino que além desses teve outros argumentos como:

- *epochs*, que é a quantidade máxima de iterações numa sessão de treino, é um dos hiper parâmetros;

- *validation_data*, que foi comparada com as variáveis de x e y (onde a rede neural já conhece);

- *callbacks*, que é o *patience*, onde vai parar na quantidade determinada na variável *parar_rede_neural,* caso a eficiência da máquina pare de aumentar.

Ao executar o método *fit* a rede iniciou seu treinamento em comparação ao seu teste de forma síncrona, onde ao visualizar o *loss* caindo significa que a rede neural está aprendendo.

7. Discussão dos Resultados

Foi utilizado 7 neurônios na camada de entrada, sendo essas as variáveis climáticas (evaporação, insolação, precipitação, temperatura, umidade e vento) além da data em tempo Unix. Para a camada oculta foi necessária apenas uma com seiscentos e sessenta e seis neurônios e para a camada de saída apenas a variável volume útil. Outras configurações foram:

- random_state = 42
- patience = 21
- learning_rate = 0.0003
- epochs = 2500

Para verificar as previsões da aprendizagem da máquina com o modelo adotado, foi utilizado o método *predict* do Pandas contendo em seu argumento a variável *X_teste*. Onde é exibido na tabela 3 a previsão da rede versus os valores corretos, que é o que já se sabe, mas que não foi apresentada a rede.

	Previsão da rede	Valores corretos
0	45.38	38.817844
1	50.17	52.617275
2	12.20	20.334005
3	46.03	65.042511
4	47.21	61.835258
...	...	...
1468	45.33	38.289410
1469	14.16	26.554461
1470	27.05	59.906822
1471	62.68	49.801525
1472	64.53	55.381088

Tabela 3 – Aprendizagem de máquina com modelo FFNN
(fonte: figura da autora via ferramenta do Colab)

Com as informações da tabela 3, foi realizado o gráfico 2, para visualizar a eficiência da aprendizagem de uma forma mais rápida, onde apesar de sua eficiência ter sido apenas de 37%, a linha laranja que representa o Teste convergiu suavemente com a linha azul representada pelo Treino, iniciando com valores alto e progredindo até a formação de uma assíntota horizontal, isso significa que no começo a rede ainda não fez uma correlação boa dos dados e pode passar por um aumento de eficiência.

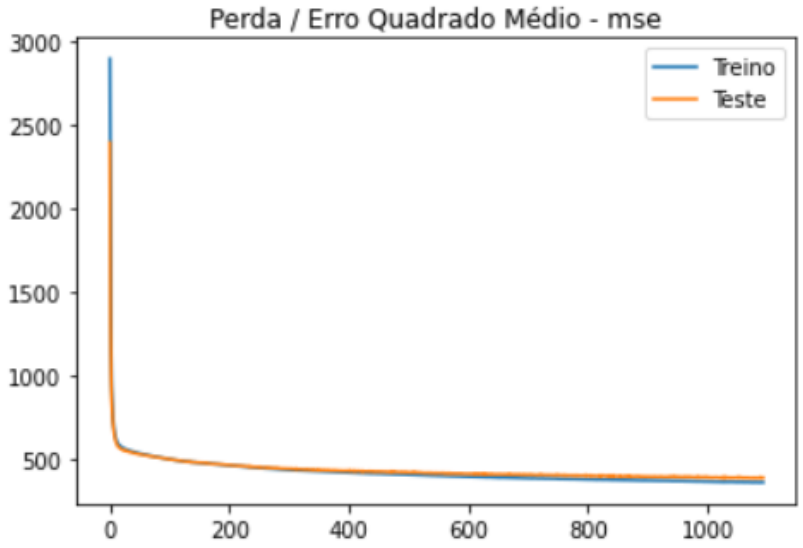

Gráfico 2 – Aprendizagem de máquina com modelo FFNN
(fonte: figura da autora via ferramenta do Colab)

Para as métricas foi utilizado recursos do sklearn a fim de extrair a eficiência da rede neural e a média da previsão da rede em relação a variável destino, que nesse se trata da coluna do volume útil utilizada na camada de saída do modelo da rede neural, na tabela 4 é exibido as métricas citadas.

Método do sklearn	Descrição da métrica	Resultado
explained_variance_score	Eficiência da rede neural	37%
mean_squared_error	Média da previsão da rede	20%

Tabela 4 – Métricas da a Aprendizagem de máquina com modelo FFNN

8. Conclusão

Correlacionar o volume útil de um reservatório com a estação meteorológica mais próxima para predizer o quanto uma Rede Neural de Alimentação Direta auxilia em previsões é uma tarefa complexa, a eficiência máxima obtida nesta tentativa inicial foi de 37%. Esses resultados serão melhorados em etapas futuras na continuidade deste trabalho através da otimização de hiper parâmetros, para tanto, propõem-se como trabalho futuro a utilização do Keras Tuner, onde no site oficial do Keras, relata que este recurso “é uma estrutura de otimização de hiper parâmetros escaláveis e fácil de usar que resolve os pontos de dor da pesquisa”.

Outra proposta é utilizar na fase de pré-processamento dos dados, o recurso do conceito Fuzzy, que na pesquisa NAGASELVI e DEEPA (2015) relataram que quando há um dataframe com uma diversidade de unidades presentes, como são os da meteorologia que contém milhas por horas, graus etc., a técnica Fuzzy remove essa dependência de unidades ao normalizar os dados.

9. Links

No repositório (https://github.com/MeusEstudos/RedesNeuraisFFNN) contém os dataframes assim como os Jupiter notebooks realizados no Colab onde exibe toda a codificação realizada na pesquisa. Também foi criado uma versão visual

deste repositório utilizando o recurso de páginas no GitHub (https://meusestudos.github.io/RedesNeuraisFFNN/).

10. Referências

ARAÚJO, M. **Nível do reservatório de Paraibuna, no Rio, atinge o volume morto.** Rio de Janeiro, 23 de janeiro de 2015. Disponível em: <https://g1.globo.com/jornal-da-globo/noticia/2015/01/nivel-do-reservatorio-de-paraibuna-no-rio-atinge-o-volume-morto.html>. Acesso em: 21 de março de 2022.

DICIONÁRIO TÉCNICO. **Resultados da busca para "Feed-forward Neural Network".** 2022. Disponível em: <https://www.dicionariotecnico.com/traducao.php?termo=Feed-forward+Neural+Network>. Acesso em: 17 de março de 2022.

EPOCH CONVERTER. **Epoch & Unix Timestamp Conversion Tools.** Disponível em: <https://www.epochconverter.com/>. Acesso em: 31 de março de 2022.

FOLHA DE SÃO PAULO. **Saiba mais sobre o volume morto.** 2015. Disponível em: <https://www1.folha.uol.com.br/cotidiano/2015/01/1574108-saiba-mais-sobre-o-volume-morto.shtml>. Acesso em: 15 de março de 2022.

FOLHA DE SÃO PAULO. **Volume morto é prefácio de clima cada vez mais extremo, diz ambientalista.** 2015. Disponível em: <https://www1.folha.uol.com.br/cotidiano/2015/12/1724210-volume-morto-e-prefacio-de-clima-cada-vez-mais-extremo-diz-ambientalista.shtml>. Acesso em: 14 de março de 2022.

G1, Rio de Janeiro. **Reservatório de Funil, no RJ, chega ao nível mais baixo desde 1969.** GLOBO.COM, 2015. Disponível em: <https://g1.globo.com/rio-de-janeiro/noticia/2015/01/reservatorio-de-funil-no-rj-chega-ao-nivel-mais-baixo-desde-1969.html>. Acesso em: 06 de abril de 2022.

G1, São Paulo. **Entenda o que é o volume morto do Sistema Cantareira.** GLOBO.COM, 2014. Disponível em: <https://g1.globo.com/sao-paulo/noticia/2014/05/entenda-o-que-e-o-volume-morto-do-sistema-cantareira.html>. Acesso em: 11 de março de 2022.

GITHUB. **Where the world builds software.** Disponível em: <https://github.com/>. Acesso em: 01 de abril de 2022.

GOOGLE. **Colaboratory.** Disponível em: <https://colab.research.google.com/>. Acesso em: 01 de abril de 2022.

GOVERNO DO BRASIL. **População brasileira chega a 213,3 milhões de habitantes, estima IBGE.** 27 de agosto de 2021. Disponível em: <https://www.gov.br/pt-br/noticias/financas-impostos-e-gestao-

publica/2021/08/populacao-brasileira-chega-a-213-3-milhoes-de-habitantes-estima-ibge>. Acesso em: 21 de março de 2022.

INMET – Instituto Nacional de Meteorologia. **Banco de Dados Meteorológicos do INMET.** Disponível em: <https://bdmep.inmet.gov.br/>. Acesso em: 29 de março de 2022.

INTEL. **CPU versus GPU: qual é a diferença**. Disponível em: <https://www.intel.com.br/content/www/br/pt/products/docs/processors/cpu-vs-gpu.html#:~:text=O%20que%20%C3%A9%20uma%20GPU,dividida%20executada%20em%20muitos%20n%C3%BAcleos>. Acesso em: 06 de abril de 2022.

KERAS. **Keras Turner.** Disponível em: <https://keras.io/keras_tuner/>. Acesso em: 08 de abril de 2022.

MATPLOTLIB. **Matplotlib: Visualization with Python.** Disponível em: <https://matplotlib.org/>. Acesso em: 01 de abril de 2022.

METEOSTAT. **The Weather's Record Keeper.** Canadá. Disponível em: <https://github.com/meteostat>. 16 de março de 2022.

NAGASELVI, M.; DEEPA, Dr. T. **Weather Forecasting using Deep Feed Forward Neural Network (DFFNN) and Fuzzy Outlier Removal.** Journal on Science Engineering & Technology, 2015. Volume 2, No. 04. Disponível em: <http://jset.sasapublications.com/wp-content/uploads/2017/10/6702690.pdf>. Acesso em: 21 de março de 2022.

NUMPY. **The fundamental package for scientific computing with Python.** Disponível em: <https://numpy.org/>. Acesso em: 01 de abril de 2022.

OLIVEIRA, S. S. **Titanic Passo a Passo com 8 Modelos ML Pt-br.** 2019. Disponível em: <https://www.kaggle.com/code/samukaunt/titanic-passo-a-passo-com-8-modelos-ml-pt-br/notebook>. Acesso em: 06 de abril de 2022.

PANDAS. **pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.** Disponível em: <https://pandas.pydata.org/>. Acesso em: 01 de abril de 2022.

PLOTLY. **Dash Enterprise.** Disponível em: <https://plotly.com/>. Acesso em: 07 de abril de 2022.

PYTHON. **Python is a programming language that lets you work quickly and integrate systems more effectively.** Disponível em: <https://www.python.org/>. Acesso em: 01 de abril de 2022.

RAY, R. **Re: Data and weather station distance.** 2017. Disponível em: <https://www.researchgate.net/post/Data-and-weather-station-distance/594d2b50217e20b58d25659f/citation/download>. Acesso em: 06 de abril de 2022.

SAR – Sistema de Acompanhamento de Reservatórios. **Dados de operação dos reservatórios SIN – FUNIL. 01/01/1993 – 05/12/2017.** Disponível em: <https://www.ana.gov.br/sar0/MedicaoSin?dropDownListEstados=20&dropDownList

Reservatorios=19093&dataInicial=01%2F01%2F1993&dataFinal=05%2F12%2F2017&button=Buscar>. Acesso em: 03 de abril de 2022.

SCIKIT-LEARN. **Machine Learning in Python.** Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 01 de abril de 2022.

SEABORN. **seaborn: statistical data visualization.** Disponível em: <https://seaborn.pydata.org/>. Acesso em: 07 de abril de 2022.

SOUZA, V. A. de et al. **Influência na variação dos volumes dos reservatórios na geração de energia: um estudo comparativo dos reservatórios do Nordeste – Sobradinho e Itaparica.** 2017. Disponível em: <https://repositorio.ufc.br/handle/riufc/54617>. Acesso em: 21 de março de 2022.

TENSORFLOW. **Keras.** Disponível em: <https://www.tensorflow.org/guide/keras?hl=pt-br>. Acesso em: 01 de abril de 2022.

WATANABE, R. M. **As partes de um reservatório**. 2015. Disponível em: <https://www.ebanataw.com.br/talude/barragem.htm>. Acesso em: 14 de março de 2022.

WIKIPEDIA, the free encyclopedia. **Era Unix**. 2022. Disponível em: <https://pt.wikipedia.org/wiki/Era_Unix>. Acesso em: 31 de março de 2022.

WIKIPEDIA, the free encyclopedia. **Feedforward neural network**. 2022. Disponível em: <https://en.wikipedia.org/wiki/Feedforward_neural_network>. Acesso em: 17 de março de 2022.