PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Rodrigo Almeida Bezerra

Algoritmos de Machine Learning aplicado a previsão do tempo na cidade de Salvador-Bahia

BELO HORIZONTE - MG
2023

Rodrigo Almeida Bezerra

Algoritmos de Machine Learning aplicado a previsão do tempo na cidade de Salvador-Bahia

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

BELO HORIZONTE - MG

2023

SUMÁRIO

1. Introdução

Este trabalho propõe uma análise de dados e a implementação de um algoritmo de previsão baseado nos dados fornecidos pelo INMET, visando destacar o impacto desses eventos no cotidiano da população de Salvador. Inicialmente, apresentamos a introdução, na qual contextualizamos o tema e delineamos o problema em questão. Em seguida, fornecemos detalhes sobre a coleta de dados no Capítulo 2, seguido pelo relato dos procedimentos adotados para o tratamento inicial, compreendendo a limpeza e a estruturação dos dados no Capítulo 3.

No capítulo subsequente, dedicamo-nos à análise e exploração dos dados, buscando extrair informações estatísticas relevantes, conforme descrito no Capítulo 4. O Capítulo 5 aborda a aplicação de Modelos de Machine Learning, onde implementamos um algoritmo de previsão e comparamos os resultados obtidos com os dados reais do INMET. Além disso, exploramos a utilização de algoritmos para agrupar os dados com base nos anos.

A conclusão do estudo, apresentada no Capítulo 6, resume e discute os resultados alcançados, oferecendo uma visão geral do impacto dos eventos climáticos em Salvador, conforme analisado e previsto pelo nosso modelo. Este trabalho busca fornecer insights valiosos para compreender as implicações práticas desses fenômenos na vida cotidiana da população.

1.1. Contextualização

Com o passar dos anos, observou-se uma crescente discussão e emissão de alertas relacionados à interação entre o ser humano e o meio ambiente. A industrialização e o avanço no desenvolvimento de novas tecnologias têm sido associados ao aumento da temperatura global, acarretando impactos significativos na qualidade de vida, não apenas para a população humana, mas também para a flora e fauna. No Brasil, tornaram-se frequentes eventos climáticos extremos, como secas intensas e enchentes, enquanto em outros países, episódios de calor extremo têm levado a consequências até mesmo fatais. Esses fenômenos, ao atingirem

áreas densamente povoadas, especialmente aglomerações urbanas, resultam em acidentes, desastres e catástrofes, caracterizados como impactos negativos no sistema socioeconômico (Dias & Herrmann, 2002).

Durante o período de chuvas intensas em Salvador, as ações preventivas coordenadas pela Defesa Civil, com o apoio de órgãos parceiros, tornam-se fundamentais para mitigar os danos causados pelas precipitações. Por meio de um trabalho educativo coordenado, diversas ações são implementadas com o propósito de sensibilizar a comunidade, visando alterar, coletivamente, o cenário de vulnerabilidade da cidade (Defesa Civil de Salvador (CODESAL). Relatório final da Operação Chuva).

Neste estudo, é realizada uma análise utilizando dados obtidos do INMET (Instituto Nacional de Meteorologia) e aplicado um algoritmo de previsão, empregando métricas como MAE, R^2, entre outras, para comparar esses dados com as previsões. Além disso, são realizadas comparações com dados obtidos por meio da API Free Weather API, que utiliza informações baixadas e processadas de diversos serviços meteorológicos nacionais, incluindo o INMET.

Com base nessas informações, é notável que Salvador, capital da Bahia, destaca-se como uma das cidades da Região Nordeste com um dos índices mais elevados de desastres naturais. A topografia da cidade, caracterizada por encostas íngremes e vales profundos, aliada à ocupação desordenada do solo, a torna particularmente vulnerável a impactos adversos provenientes de chuvas intensas (SANTOS, 2008).

MÊS	2019		2020		2021		2022		2023	
	LONA (m²)	ÁREAS (Un)	LONA (m²)	ÁREAS (Un)	LONA (m²)	ÁREAS (Un)	LONA (m²)	ÁREAS (Un)	LONA (m²)	ÁREAS (Un)
JANEIRO	5.842	43	29.165	212	26.984	108	6.438	39	3.934	17
FEVEREIRO	3.958	46	13.888	99	15.450	101	6.706	36	3.964	32
MARÇO	43.442	310	48.474	352	5.440	25	13.390	85	23.714	147
ABRIL	68.518	640	81.670	609	32.342	151	25.972	134	18.660	145
MAIO	45.440	379	123.826	941	30.842	168	22.368	134	36.598	243
JUNHO	29.436	231	74.591	495	21.640	139	26.204	178	30.158	229
*TOTAL	186.836	1.560	328.561	2.397	90.264	483	87.934	531	109.130	764
TOTAL	196.636	1.649	371.614	2.708	132.698	692	101.078	606	117.028	813

Figura1.Fonte: Defesa Civil de Salvador – CODESAL

Com base nisso é necessária analisar como esse volume de chuva impactou a vida na cidade de Salvador e como algoritmos de Machine Learning, podem ajudar com politicas publicas melhores e ações como Operação Chuva mais eficiente.

1.2. Problema proposto

A crise climática vem sendo amplamente discutida em eventos sobre clima e eventos atípicos ocorrem ao redor de todo o planeta. As anomalias da Temperatura da Superfície do Mar (TSM) no Oceano Pacífico Equatorial associadas ao evento El Niño-Oscilação Sul (ENOS) (Ropelewski e Halpert, 1987), e anomalias de TSM no Atlântico Tropical associadas ao Gradiente do Atlântico tem influência sobre a precipitação no Nordeste do Brasil (Hastenrath e Heller, 1977) logo, as diferenças na distribuição da chuva na região, tanto em escala espacial quanto temporal, está intimamente relacionada com alterações na circulação atmosférica de grande escala, devido à interação entre a atmosfera e os oceanos Pacífico e Atlântico.

Em Salvador existem medidas adotadas pela Defesa Civil para diminuir impactos do grande volume de chuvas como geomantas (Figura 2) cm objetivo de diminuir os deslizamentos de terá principalmente em bairros mais carentes, ainda

assim existe relatos de deslizamentos de terras e áreas que precisam ser isoladas deixando pessoas desabrigadas.

Figura 2. Fonte: CODESAL – Relatório Final Operação Chuva (2023)

OBS: Os meses em destaque foi quando ocorreu a Operação Chuva, inclusive o *TOTAL.

A capital baiana ainda conta com um sistema de alarmes para caso de chuvas intensas e possíveis desastres naturais. O CEMADEC monitora uma rede composta por 74 Plataformas Coletoras de Dados Pluviométricas, Meteorológicas, Geotécnicas e Hidrológicas, sendo 35 da Codesal, 37 do Centro Nacional de Monitoramento e Alertas de Desastres Naturais – CEMADEN e 02 do Instituto Nacional de Meteorologia – INMET, que permitem acompanhar os índices pluviométricos em tempo real, contribuindo diretamente para tomada de decisão por parte da Codesal. Além disso, são monitoradas 15 estações geotécnicas, 04 estações hidrológicas (02 da Codesal e 02 do CEMADEN) e 04 estações meteorológicas (02 do INMET e 02 da Codesal) (Relatório Anual 2022 – Codesal). Neste trabalho busca propor através de métodos de machine learning uma nova forma de analisar e abordar a vulnerabilidade da infraestrutura de Salvador diante de chuvas intensas, visando desenvolver estratégias eficazes para mitigar os impactos negativos e promover uma gestão mais eficiente durante eventos climáticos extremos. E com isso responder o método 5W2h:

What (O que): As mudanças climáticas afetam os padrões de chuva em Salvador ao modificar a distribuição temporal e espacial das precipitações. Essas mudanças podem resultar em variações na intensidade, frequência e duração das chuvas na região.

Why (Por que): Compreender esses padrões de chuva e sua relação com as mudanças climáticas é crucial devido aos impactos significativos que eventos de chuva intensa podem causar em áreas urbanas como Salvador. A cidade está sujeita a inundações, deslizamentos de terra e outros danos à infraestrutura, afetando diretamente a qualidade de vida da população. Além disso, a adaptação urbana e a gestão de riscos exigem uma compreensão aprofundada desses padrões para o desenvolvimento de estratégias eficazes.

Who (Quem): A população de Salvador, em particular aqueles que residem em áreas suscetíveis a inundações e deslizamentos, será impactada pelos eventos de chuva intensa. Além dos residentes, setores econômicos, como comércio e transporte, também podem sofrer consequências adversas.

When (Quando): Os eventos de chuva intensa em Salvador tendem a ser mais propensos durante a estação chuvosa, que geralmente ocorre nos meses de abril a julho,porém neste trabalho será analisando no período de 2019 até 2023. No entanto, as mudanças climáticas podem influenciar a sazonalidade, tornando os eventos extremos mais imprevisíveis ao longo do ano.

How (Como): Os modelos de aprendizado de máquina podem contribuir para prever eventos de chuva intensa de maneira mais precisa ao analisar grandes volumes de dados meteorológicos. Esses modelos podem identificar padrões complexos e relações não lineares nos dados históricos, permitindo a criação de previsões mais precisas em comparação com métodos tradicionais. A aplicação desses modelos envolve treinamento com dados passados, validação e ajuste contínuo para melhorar a capacidade de previsão ao longo do tempo.

1.3. Objetivos

Utilizaremos análise exploratória e modelagem preditiva para extrair informações das séries temporais para a partir disso poder prever possíveis temporais e contribuir para aplicação de políticas publicas mais eficientes e auxiliar os órgãos responsáveis por atuar em tempos de temporais.

Objetivo Geral:

O objetivo geral deste estudo é avaliar a os padrões de chuva em Salvador, analisando e usando algoritmos de machine learning para obter resultados, com o objetivo de identificar oportunidades de melhoria na infraestrutura de cidade e promover ações de prevenção mais eficiente e inclusiva na cidade.

Objetivos Específicos:

1. Analisar os padrões históricos de chuva em Salvador em relação às mudanças climáticas.
2. Desenvolver modelos de aprendizado de máquina para prever eventos de chuva intensa.
3. Avaliar a eficácia dos modelos na previsão precisa de eventos climáticos extremos.

2. Coleta de Dados

A metodologia de coleta de dados utilizada neste estudo envolveu a integração de informações provenientes de diferentes fontes. Dados foram extraídos da API Open-Meteo através do link (https://open-meteo.com/en/docs/climate-api#latitude=-12.9711&longitude=-38.5108&start_date=2019-01-01&end_date=2023-12-31)e do Instituto Nacional de Meteorologia (INMET). Os dados do INMET foram coletados por Gregory Oliveira e estão disponíveis no Kaggle(Figura 3) através do link (https://www.kaggle.com/datasets/gregoryoliveira/brazil-weather-information-by-inmet) . O outro dataset utilizado é proveniente de uma API ,a Open-Meteo (Figura 4), onde foi possível ter dados de mudança climática de 2019 á 2023 porém esses dados foram coletados de forma open source , a escolha desse dataset é para diminuir o viés no algoritmo de previsão.

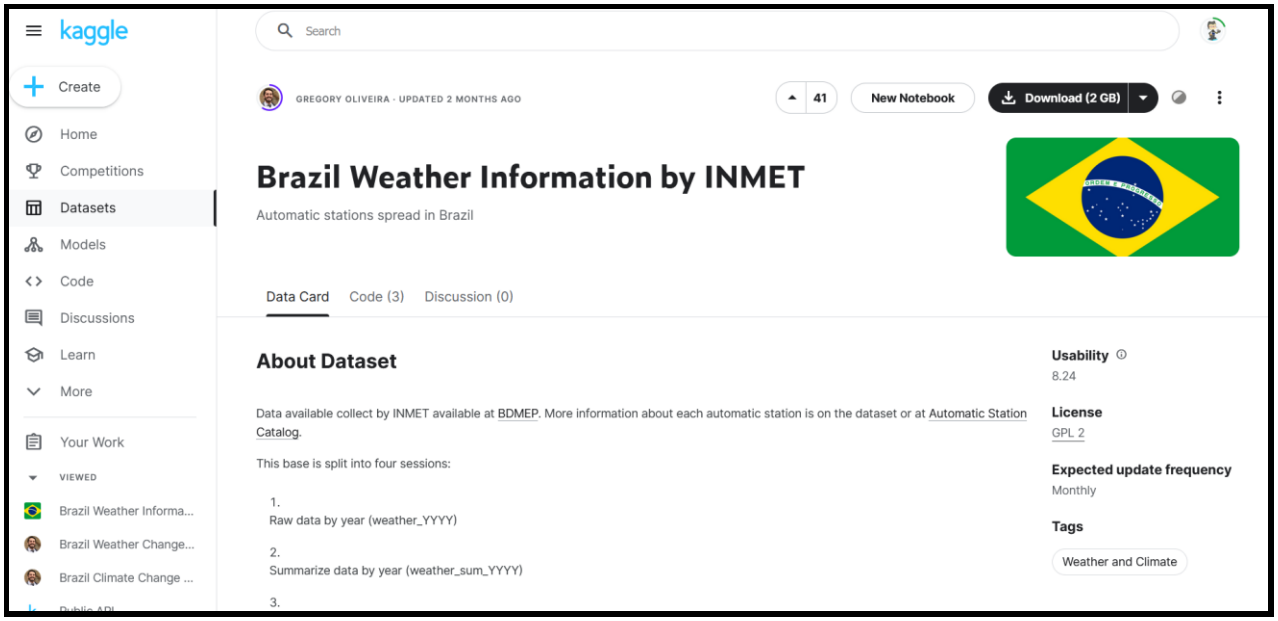

Figura 3. Fonte: Autoria própria.

Dataset - INMET

Nome da coluna/campo	Descrição	Tipo
ESTAÇÃO	CIDADE QUE A ESTAÇÃO METEREOLOGICA SE ENCONTRA	Object
DATA	Dia,mês e ano do registro do dado.	Object
HORA	Fuso Horario	Object
PRECIPITACAO TOTAL HORARIO	É a medida do total de precipitação (chuva) que ocorreu no espaço de 24 hora e o horário.	float64
PRESSAO ATMOSFERICA AO NIVEL DA ESTACAO, HORARIA	Nas estações convencionais é a medida da pressão atmosférica que foi medida na estação, e a partir deste valor, também é calculada a pressão ao nível do mar (para fins sinóticos).	float64
PRESSAO ATMOSFERICA MAX.NA HORA ANT.	Nas estações automáticas é a medida da pressão atmosférica máxima, ocorrida na última hora antes de cada mensagem de dados.	float64
PRESSÃO ATMOSFERI-CA MIN. NA HORA ANT. (AUT)	Nas estações automáticas é medida da pressão atmosférica mínima,	float64

	ocorrida na última hora antes de cada mensagem de dados.	
RADIACAO GLOBAL (KJ/m²)	Nas estações automáticas é a medida de toda radiação solar que chegou a superfície terrestre, na última hora antes de cada mensagem de dados.	float64
TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)	Nas estações convencionais é a medida da temperatura do ar, a partir do termômetro de bulbo seco do psicrômetro (equipamento dotado de um termômetro de bulbo seco e um termômetro de bulbo úmido).	float64
TEMPERATURA DO PONTO DE ORVALHO (°C)	Nas estações convencionais é uma medida determinada de forma indireta (a partir dos valores de temperatura do ar e da umidade relativa), através de valores tabulares.	float64
TEMPERATURA MÁXIMA NA HORA ANT. (AUT) (°C)	Nas estações convencionais é a medida máxima da temperatura do ar, ocorrida no período de 24 horas, a partir do termômetro específico	float64

	para este fim. Nas estações automáticas é medida máxima da temperatura do ar, ocorrida na última hora antes de cada mensagem de dados.	
TEMPERATURA MÍNIMA NA HORA ANT. (AUT) (°C)	Nas estações convencionais é a medida mínima da temperatura do ar, ocorrida no período de 24 horas, a partir do termômetro específico para este fim. Nas estações automáticas é a medida mínima da temperatura do ar, ocorrida na última hora antes de cada mensagem de dados.	float64
TEMPERATURA ORVA-LHO MAX. NA HORA ANT. (AUT) (°C)	Nas estações automáticas este parâmetro é calculado a partir dos valores máximos de temperatura do ar e da umidade relativa, ocorridos na última hora antes de cada mensagem de dados. Este parâmetro não é determinado nas estações convencionais.	float64
TEMPERATURA ORVA-	Nas estações automáticas	float64

LHO MIN. NA HORA ANT. (AUT) (°C)	este parâmetro é calculado a partir dos valores mínimos de temperatura do ar e da umidade relativa, ocorridos na última hora antes de cada mensagem de dados. Este parâmetro não é determinado nas estações convencionais.	
UMIDADE REL. MAX. NA HORA ANT. (AUT) (%)	Nas estações automáticas é a medida máxima da umidade relativa do ar, ocorrida na última hora antes de cada mensagem de dados. Este parâmetro não é determinado nas estações convencionais.	float64
UMIDADE REL. MIN. NA HORA ANT. (AUT) (%)	Nas estações automáticas é a medida mínima da umidade relativa do ar, ocorrida na última hora antes de cada mensagem de dados. Este parâmetro não é determinado nas estações convencionais.	float64
UMIDADE RELATIVA DO AR, HORARIA (%)	Nas estações convencionais é uma medida determinada de forma indireta, a partir do psicrômetro (equipamento dotado de um termômetro	float64

	de bulbo seco e um termômetro de bulbo úmido), e o uso de valores tabulares. Nas estações automáticas é a medida da umidade relativa do ar, ocorrida na última hora antes de cada mensagem de dados.	
VENTO, DIREÇÃO HORARIA ()	Nas estações automáticas é a medida em graus angulares da direção do vento (de onde o vento vem). Este valor é a média dos últimos 10 minutos antes de cada hora, de envio da mensagem de dados.	float64
VENTO, RAJADA MAXIMA (m/s)	Nas estações automáticas é a medida máxima da velocidade do vento, ocorrida na última hora antes de cada mensagem de dados. Nas estações convencionais é a máxima medida da velocidade do vento ocorrida no período de 24 horas.	float64
VENTO, VELOCIDADE HORARIA (m/s)	Nas estações automáticas é a medida da velocidade do vento. Este valor é a média dos últimos 10	float64

	minutos antes de cada hora, de envio da mensagem de dados. Nas estações convencionais é a medida da velocidade do vento, e determinada a partir da média dos últimos 10 minutos antes da hora cheia (09:00, ..., 12:00 UTC) de cada observação.	

Abaixo na figura 3 temos o início do script da API utilizada para a captura dos dados. Então na figura 4 temos a configuração utilizada na interface da API, onde utilizamos dados de Climate Change, com o fuso horário de São Paulo e as coordenadas de Salvador que é possível preencher clicando em Search e adicionando a cidade desejada.

```
import openmeteo_requests

import requests_cache
import pandas as pd
from retry_requests import retry

# Setup the Open-Meteo API client with cache and retry on error
cache_session = requests_cache.CachedSession('.cache', expire_after = 3600)
retry_session = retry(cache_session, retries = 5, backoff_factor = 0.2)
openmeteo = openmeteo_requests.Client(session = retry_session)

# Make sure all required weather variables are listed here
# The order of variables in hourly or daily is important to assign them correctly below
url = "https://climate-api.open-meteo.com/v1/climate"
params = {
    "latitude": -12.9711,
    "longitude": -38.5108,
    "start_date": "2020-01-01",
    "end_date": "2023-12-31",
    "models": ["CMCC_CM2_VHR4", "FGOALS_f3_H", "HiRAM_SIT_HR", "MRI_AGCM3_2_S", "EC_Earth3P_HR", "MPI_ESM1_2_XR", "NICAM16_8S"],
    "timezone": "America/Sao_Paulo",
    "daily": ["temperature_2m_mean", "temperature_2m_max", "temperature_2m_min", "wind_speed_10m_mean", "wind_speed_10m_max", "cl
}
responses = openmeteo.weather_api(url, params=params)
```

Figura 3. Fonte: Autoria própria.

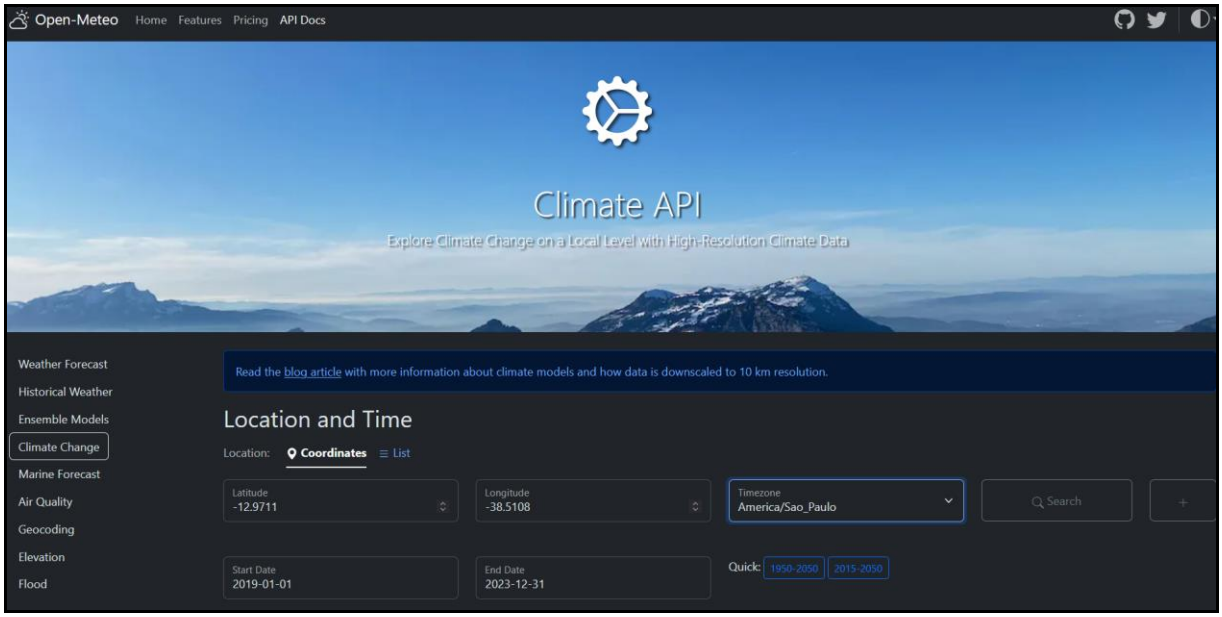

Figura 4. Fonte: Autoria própria.

Dataset – Open-Meteo

Nome da coluna/campo	Descrição	Tipo
date	Data	Object
temperature_2m_mean	Temperatura média a 2m do nível do mar	Object
temperature_2m_max	Temperatura maxima a 2m do nível do mar	float64
temperature_2m_min	Temperatura minima a 2m do nível do mar	float64
wind_speed_10m_mean	Media da velocidade do vento a 10 metros da altura do mar	float64
wind_speed_10m_max	Velocidade máxima do vento a 10 metros do nível do mar	float64
cloud_cover_mean	Cobertura média de formação de nuvem	float64

shortwave_radiation_sum	Soma da formação de nuvens	float64
relative_humidity_2m_mean	Humidade relativa a 2 m do nível do mar	float64
precipitation_sum	Precipitação no dia	Float64

Ambos dataset possuem campos relativo a precipitação, velocidade do vento e temperatura.

3. Processamento/Tratamento de Dados

O projeto foi executado por meio da linguagem de programação interpretada Python 3.9.13 no Jupyter Notebook– ambiente de desenvolvimento interativo baseado na Web – e pelo Google Colab tendo sido utilizadas, primordialmente, as bibliotecas listadas na tabela a seguir:

Biblioteca	Versão
Numpy	1.21.5
Matplotlib	3.5.2
Seaborn	0.11.2
Pandas	1.4.4
pip	23.1.2
sklearn	0.0. post5
scikit-learn	1.0.2
tensorflow math	2.10.0 1.10.1
Datetime	2.8.2
pkg_resources	1.81
imblearn	0.0
Statsmodels	0.13.2

Como o período utilizado é 5 anos, de 2019 á 2023 contido no dataset da API e no o dataset do INMET tá dividido por ano porém para facilitar leitura, manipulação e tratamento dos dados realizei a leitura e agrupei em um único dataframe (Figura 6).

Isso além de facilitar o tratamento e a análise dos dados permitiu no algoritmo de previsão dividir o dataset .

- Diretórios onde estão armazenados os dataset:

```
# Diretório com os 4 dataset do INMET
diretorio = './Dados/INMET'
# Diretório com dataset da API Open Meteo:
diretorio_api = './Dados/API'
```

Figura 5. Fonte: Autoria própria.

- Leitura e agrupamento dos 4 dataset do INMET:

```
# Criação de um dataframe vazio para armazenar dados dos datasets
dataframes = []

# Loop por todos arquivos .csv na pasta
for arquivo in os.listdir(diretorio):
    if arquivo.endswith('.csv'):
        # Cria o caminho completo do arquivo
        caminho_completo = os.path.join(diretorio, arquivo)

        # Lê o arquivo CSV e adiciona o DataFrame à lista
        data = pd.read_csv(caminho_completo)
        dataframes.append(data)

for i in range(1, len(dataframes)):
    dataframes[i] = dataframes[i].reindex(columns=dataframes[0].columns, fill_value=None)

# Concatena todos os DataFrames em um único DataFrame
dados = pd.concat(dataframes, ignore_index=True)

# Cria um arquivo .csv com o DataFrame gerado
dados.to_csv('dataset_INMET-Unico.csv', index=False)
dados
```

Figura 6. Fonte: Autoria própria.

Dataset resultante:

	DATA (YYYY-MM-DD)	Hora UTC	PRECIPITAÇÃO TOTAL, HORÁRIO (mm)	PRESSAO ATMOSFERICA AO NIVEL DA ESTACAO, HORARIA (mB)	PRESSÃO ATMOSFERICA MAX.NA HORA ANT. (AUT) (mB)	PRESSÃO ATMOSFERICA MIN. NA HORA ANT. (AUT) (mB)	RADIACAO GLOBAL (KJ/m²)	TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)	TEMPERATURA DO PONTO DE ORVALHO (°C)	TEMPERATURA MÁXIMA NA HORA ANT. (AUT) (°C)	TEMPERAT MÍNIM/ HORA (AUT)
0	2020-01-01	0000 UTC	0.0	884.1	884.2	883.9	NaN	20.0	18.1	20.2	
1	2020-01-01	0100 UTC	0.0	885.0	885.0	884.1	NaN	19.5	18.1	20.0	
2	2020-01-01	0200 UTC	0.0	885.5	885.5	885.0	NaN	20.0	18.8	20.1	
3	2020-01-01	0300 UTC	0.0	885.3	885.5	885.3	NaN	19.5	17.9	20.2	
4	2020-01-01	0400 UTC	0.0	885.2	885.4	885.2	NaN	19.5	17.3	19.8	
...	...	...	...	...	...	...	...	...	...	...	
19818643	2023-11-30	1900 UTC	0.0	900.9	901.5	900.9	2307.5	28.9	18.1	29.4	
19818644	2023-11-30	2000 UTC	0.0	900.5	900.9	900.5	1547.1	28.0	18.7	29.3	
19818645	2023-11-30	2100 UTC	0.0	900.6	900.6	900.3	1058.2	28.7	17.3	28.8	
19818646	2023-11-30	2200 UTC	0.0	901.2	901.2	900.6	297.6	27.5	16.8	28.8	
19818647	2023-11-30	2300 UTC	0.0	901.6	901.6	901.2	NaN	26.1	17.2	27.5	

19818648 rows × 20 columns

Figura 8. Fonte: Autoria própria.

Através do arquivo Estacoes.csv também do Kaggle é possivel vê todas as estações e informações, como o código delas:

```
#Verificando as estações com base no arquivo Estacoes.csv
estacoes = pd.read_csv('./Dados/Estacoes.csv')

estacoes
```

Figura 9.Fonte Autoria Própria

	region	state	city_station	id_station	lat	lon	lvl	record_first	record_last
0	CO	DF	BRASILIA	A001	-15,78944444	-47,92583332	1159,54	2000-05-07	2022-08-31
1	NE	BA	SALVADOR	A401	-13,01666666	-38,51666666	51,41	2000-05-13	2022-08-31
2	N	AM	MANAUS	A101	-3,10333333	-60,01638888	61,25	2000-05-09	2022-08-31
3	SE	RJ	ECOLOGIA AGRICOLA	A601	-22,8	-43,68333333	33	2000-05-07	2022-08-31
4	S	RS	PORTO ALEGRE	A801	-30,05	-51,16666666	46,97	2000-09-22	2022-08-31
...	...	...	...	...	...	...	...	...	...
608	S	SC	CAMPOS NOVOS	A898	-27,3886111	-51,21583333	963	2019-02-15	2022-08-31
609	SE	SP	CRIOSFERA	C891	-84	-79,49416666	1285	2012-09-01	2020-12-31
610	N	PA	SANTA MARIA DAS BARREIRAS	A256	-8,72972221	-49,85638888	165	2020-12-18	2022-08-31
611	SE	MG	POMPEU	A560	-19,23249999	-44,96416666	705	2015-08-21	2022-08-31
612	N	PA	PORTO DE MOZ	A245	-1,82194443	-52,11166666	19	2022-07-04	2022-08-31

Figura 10.Fonte Autoria Própria

Após isso inicia-se o tratamento buscando filtrar e salvar em .csv para a cidade de Salvador Bahia:

```
#Coletando do dataset original somente a Estação de Salvador (A401)
dados = dados[dados['ESTACAO']=='A401']
dados.to_csv('dataset_INMET-Salvador.csv', index=False)
dados
```

Figura 11. Fonte: Autoria própria.

	DATA (YYYY-MM-DD)	Hora UTC	PRECIPITAÇÃO TOTAL, HORÁRIO (mm)	PRESSAO ATMOSFERICA AO NIVEL DA ESTACAO, HORARIA (mB)	PRESSÃO ATMOSFERICA MAX.NA HORA ANT. (AUT) (mB)	PRESSÃO ATMOSFERICA MIN. NA HORA ANT. (AUT) (mB)	RADIACAO GLOBAL (KJ/m²)	TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)	TEMPERATURA DO PONTO DE ORVALHO (°C)	TEMPERATURA MÁXIMA NA HORA ANT. (AUT) (°C)
1326384	2020-01-01	0000 UTC	0.0	1006.3	1006.3	1005.9	NaN	25.9	22.1	26.4
1326385	2020-01-01	0100 UTC	0.0	1007.0	1007.0	1006.3	NaN	26.0	22.4	26.1
1326386	2020-01-01	0200 UTC	0.0	1006.9	1007.2	1006.9	NaN	26.0	22.7	26.1
1326387	2020-01-01	0300 UTC	0.0	1006.3	1006.9	1006.3	NaN	25.7	22.9	26.0
1326388	2020-01-01	0400 UTC	0.0	1005.2	1006.3	1005.2	NaN	25.7	23.1	25.8
...	...	...	...	...	...	...	...	...	...	...
16347715	2023-11-30	1900 UTC	0.0	1005.3	1005.4	1005.1	1802.4	29.4	21.3	30.3
16347716	2023-11-30	2000 UTC	0.0	1005.5	1005.5	1005.2	520.6	27.5	21.4	29.6
16347717	2023-11-30	2100 UTC	0.0	1005.7	1005.8	1005.5	46.9	26.7	21.3	27.5
16347718	2023-11-30	2200 UTC	0.0	1006.1	1006.2	1005.7	NaN	26.6	21.6	26.7
16347719	2023-11-30	2300 UTC	0.0	1006.5	1006.5	1006.1	NaN	26.3	21.4	26.6

34320 rows × 20 columns

Figura 12.Fonte Autoria Própria

Então é feito a leitura do dataset da API:

```
# Lendo o arquivo dados_api.csv proveniente da consulta na API Open Meteo
# Lê o arquivo CSV e adiciona o DataFrame à lista
dados_api = pd.read_csv(diretorio_api)
# Exibe o DataFrame resultante
dados_api
```

Figura 13. Fonte: Autoria própria.

Dataset API:

	date	temperature_2m_mean	temperature_2m_max	temperature_2m_min	wind_speed_10m_mean	wind_speed_10m_max	cloud_cover_mean
0	2019-01-01	26.815092	25.020996	85.708015	91.317510	77.232990	2.134795
1	2019-01-02	28.864230	24.866518	82.718410	92.345540	65.240410	4.727908
2	2019-01-03	27.511778	23.711008	84.713490	91.363510	80.236770	2.239490
3	2019-01-04	28.759325	24.105497	82.708580	92.381480	68.233120	0.470172
4	2019-01-05	29.206875	24.749987	81.703670	92.399445	66.229490	1.172232

Figura 14.Fonte Autoria Própria

Então como últimas etapas do tratamento dos dados é feito a substituição de por onde tiver NaN(Not a Number) por zero e exclusão de colunas desnecessárias:

```
#Substitui onde tiver NaN por 0
dados = data.fillna(0)
dados_api = dados_api.fillna(0)
```

Figura 15. Fonte: Autoria própria.

```
#Eliminando colunas desnecessarias
dados_api = dados_api.drop(columns=[  'cloud_cover_mean','shortwave_radiation_sum'])
dados = dados.drop(columns=['PRESSÃO ATMOSFERICA MAX.NA HORA ANT. (AUT) (mB)','PRESSÃO ATMOSFERICA MIN. NA HORA ANT. (AUT)(mB)'
                            ,'TEMPERATURA MÁXIMA NA HORA ANT. (AUT) (°C)','TEMPERATURA MÍNIMA NA HORA ANT. (AUT) (°C)',
                            'UMIDADE REL. MAX. NA HORA ANT. (AUT) (%)','UMIDADE REL. MIN. NA HORA ANT. (AUT) (%)'])
```

Figura 16. Fonte: Autoria própria.

Sobrando assim dos dataset do INMET com 20 colunas do dataset original e do dataset da API com as 13 colunas do original**.** Verificando o cabeçalho de cada dataset**.**

Dataset INMET:

	DATA (YYYY-MM-DD)	Hora UTC	PRECIPITAÇÃO TOTAL, HORÁRIO (mm)	PRESSAO ATMOSFERICA AO NIVEL DA ESTACAO, HORARIA (mB)	PRESSÃO ATMOSFERICA MAX.NA HORA ANT. (AUT) (mB)	PRESSÃO ATMOSFERICA MIN. NA HORA ANT. (AUT) (mB)	RADIACAO GLOBAL (KJ/m²)	TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)	TEMPERATURA DO PONTO DE ORVALHO (°C)	TEMPERATURA MÁXIMA NA HORA ANT. (AUT) (°C)	TE
1326384	2020-01-01	0000 UTC	0.0	1006.3	1006.3	1005.9	NaN	25.9	22.1	26.4	
1326385	2020-01-01	0100 UTC	0.0	1007.0	1007.0	1006.3	NaN	26.0	22.4	26.1	
1326386	2020-01-01	0200 UTC	0.0	1006.9	1007.2	1006.9	NaN	26.0	22.7	26.1	
1326387	2020-01-01	0300 UTC	0.0	1006.3	1006.9	1006.3	NaN	25.7	22.9	26.0	
1326388	2020-01-01	0400 UTC	0.0	1005.2	1006.3	1005.2	NaN	25.7	23.1	25.8	
...	...	...	...	...	...	...	...	...	...	...	
16347715	2023-11-30	1900 UTC	0.0	1005.3	1005.4	1005.1	1802.4	29.4	21.3	30.3	
16347716	2023-11-30	2000 UTC	0.0	1005.5	1005.5	1005.2	520.6	27.5	21.4	29.6	
16347717	2023-11-30	2100 UTC	0.0	1005.7	1005.8	1005.5	46.9	26.7	21.3	27.5	
16347718	2023-11-30	2200 UTC	0.0	1006.1	1006.2	1005.7	NaN	26.6	21.6	26.7	
16347719	2023-11-30	2300 UTC	0.0	1006.5	1006.5	1006.1	NaN	26.3	21.4	26.6	

Figura 17.Fonte Autoria Própria

Dataset API**:**

	date	temperature_2m_mean	temperature_2m_max	temperature_2m_min	wind_speed_10m_mean	wind_speed_10m_max	cloud_cover_mean
0	2019-01-01	26.815092	25.020996	85.708015	91.317510	77.232990	2.134795
1	2019-01-02	28.864230	24.866518	82.718410	92.345540	65.240410	4.727908
2	2019-01-03	27.511778	23.711008	84.713490	91.363510	80.236770	2.239490
3	2019-01-04	28.759325	24.105497	82.708580	92.381480	68.233120	0.470172
4	2019-01-05	29.206875	24.749987	81.703670	92.399445	66.229490	1.172232
...	...	...	...	...	...	...	...
1821	2023-12-27	26.944578	26.028261	76.412285	77.011925	73.019875	13.770626
1822	2023-12-28	27.798487	26.076876	74.468605	81.070130	66.060470	6.182674
1823	2023-12-29	28.202393	25.475494	69.524950	79.128340	60.101055	4.703408
1824	2023-12-30	28.806301	25.824110	74.581270	80.186550	66.141655	9.838177
1825	2023-12-31	27.810207	25.622728	76.637600	81.244760	72.182250	4.297015

1826 rows × 13 columns

Figura 18.Fonte Autoria Própria

A utilização de dados abertos provenientes do Kaggle oferece uma vasta gama de conjuntos de dados provenientes de diversas fontes, enriquecendo a diversidade e a representatividade das informações. A plataforma, como repositório colaborativo, proporciona acesso a conjuntos de dados desafiadores, possibilitando a realização de estudos robustos e aprimoramento contínuo das técnicas empregadas.

Com a integração de dados da API Open Meteo adiciona uma camada dinâmica à coleta e tenta diminui o viés do dataset do autor compartilhou o dataset do INMET no Kaggle, permitindo a obtenção de informações climáticas em tempo real. Essa abordagem não apenas enriquece a análise com dados atualizados, mas também oferece uma perspectiva mais imersiva em relação às variabilidades climáticas. A combinação dessas duas fontes de dados, Kaggle e API Open Meteo, confere ao trabalho uma quantidade de dados suficientes para analises de dados distintas assim como interpretar padrões climáticos com menor viés e uma abrangência maior, promovendo assim embasamento robusto para a tomada de decisões em diferentes domínios.

4. Análise e Exploração dos Dados

Nesta seção será demonstrado como os dados estão organizados e se relacionando. A análise demonstrada aqui foi tanto do dataframe feito com os datasets do INMET quanto do dataset da API Open Meteo. Como o principal objetivo é demonstrar como as mudanças climáticas influenciam no tempo e nas chuvas intensas em Salvador, é possível ver algumas estatísticas relacionadas ao dataset do INMET

	count	mean	std	min	25%	50%	75%	max
PRECIPITAÇÃO TOTAL, HORÁRIO (mm)	4545072.0	0.122159	1.154570	0.0	0.0	0.0	0.0	131.0
PRESSAO ATMOSFERICA AO NIVEL DA ESTACAO, HORARIA (mB)	4545072.0	814.901426	351.179680	0.0	914.3	960.1	991.0	1040.1
RADIACAO GLOBAL (KJ/m²)	4545072.0	625.583236	1012.956711	0.0	0.0	0.0	1020.9	9750.3
TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)	4545072.0	19.806248	10.090916	-2.7	16.5	22.4	26.4	43.2
TEMPERATURA DO PONTO DE ORVALHO (°C)	4545072.0	13.931771	8.038817	-35.0	9.7	16.6	20.2	37.4
TEMPERATURA ORVALHO MAX. NA HORA ANT. (AUT) (°C)	4545072.0	14.351977	8.221535	-34.3	10.3	17.1	20.7	38.6
TEMPERATURA ORVALHO MIN. NA HORA ANT. (AUT) (°C)	4545072.0	13.482787	7.892587	-34.6	9.0	16.0	19.7	36.0
UMIDADE RELATIVA DO AR, HORARIA (%)	4545072.0	57.847946	33.813885	0.0	36.0	68.0	86.0	100.0
VENTO, DIREÇÃO HORARIA (gr) (° (gr))	4545072.0	122.977637	108.415862	0.0	18.0	104.0	201.0	360.0
VENTO, RAJADA MAXIMA (m/s)	4545072.0	3.616892	3.243884	0.0	0.0	3.2	5.8	47.3
VENTO, VELOCIDADE HORARIA (m/s)	4545072.0	1.414322	1.599881	0.0	0.0	1.0	2.3	26.0

Figura 19. Fonte Autoria Própria.

Então temos a correlação entre as variáveis:

	PRECIPITAÇÃO TOTAL, HORÁRIO (mm)	PRESSAO ATMOSFERICA AO NIVEL DA ESTACAO, HORARIA (mB)	RADIACAO GLOBAL (KJ/m²)	TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)
PRECIPITAÇÃO TOTAL, HORÁRIO (mm)	1.000000	0.044004	-0.045653	0.006731

Figura 20. Fonte: Autoria própria.

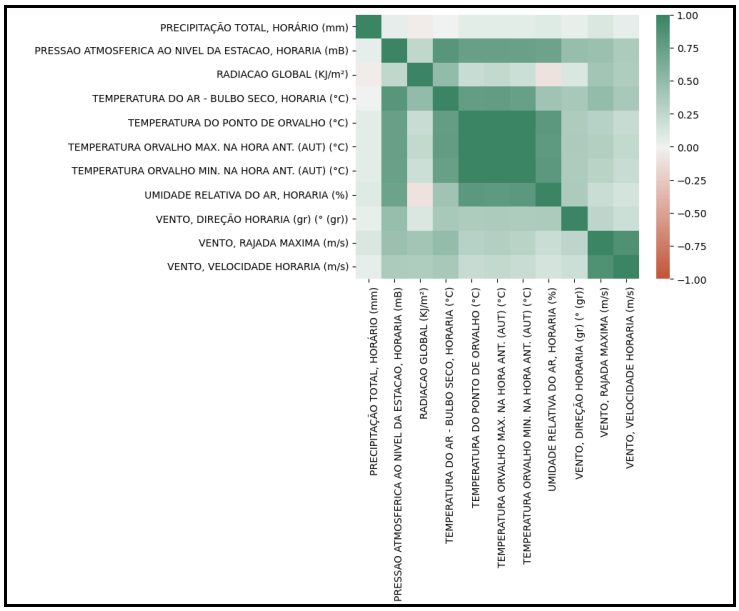

Figura 21. Fonte: Autoria própria.

Através da Figura 21, é possível observar a correlação entre as variáveis Precipitação e Temperatura do Ar, a qual é de 0,006731. Vale ressaltar que a temperatura do ar exerce influência sobre a capacidade do ar de conter vapor d'água. Dada a vasta extensão territorial do Brasil, diferentes regimes de precipitação e temperatura são observados de norte a sul, apresentando características regionais distintas.

No norte do país, é predominante um clima equatorial chuvoso, praticamente sem estação seca. Já no Nordeste, a estação chuvosa, com baixos índices pluviométricos, é restrita a poucos meses, caracterizando um clima semiárido. Kousky (1980) observou que o máximo de chuvas no leste do Nordeste, de maio a julho, está possivelmente associado à máxima convergência dos alísios com a brisa terrestre, sendo mais forte nas estações de outono e inverno devido ao maior contraste de temperatura entre a terra e o mar.

Cavalcanti (1982) destacou a contribuição das linhas de instabilidade para a precipitação na costa norte/nordeste da América do Sul, sendo mais frequentes nos meses de outono/inverno e menos na primavera e verão. A presença da Zona de Convergência Intertropical (ZCIT) próxima à região, provocando baixas pressões, favorece

o desenvolvimento de cumulus nimbus na costa, indicando uma associação entre sistemas locais e de grande escala.

Quanto à correlação entre umidade e temperatura, esta se situa entre 0,25 e 0,5. No contexto do sul do Nordeste, o principal mecanismo de precipitação são os sistemas frontais estacionários alimentados pela umidade do Atlântico Sul, definindo a Zona de Convergência do Atlântico Sul (ZCAS), além de sistemas pré-frontais, convecção local e brisas marítimas e terrestres. Nessa região, os índices pluviométricos variam de 300 mm/ano no interior a mais de 3000 mm/ano no litoral, assemelhando-se ao regime de chuvas da parte norte da região Sudeste do Brasil (Molion & Bernardo, 2002).

Para mitigar possíveis viés nos dados do INMET, coletados indiretamente via Kaggle, foi utilizado outro dataset proveniente de um projeto de código aberto, o qual compila dados das principais organizações e institutos de clima dos países. A Figura 22 exibe algumas estatísticas desse dataset, proporcionando uma análise mais abrangente.

	count	mean	std	min	25%	50%	75%	max
PRECIPITAÇÃO TOTAL, HORÁRIO (mm)	34316.0	0.226617	1.378074	0.0	0.0	0.0	0.00	49.8
PRESSAO ATMOSFERICA AO NIVEL DA ESTACAO, HORARIA (mB)	34316.0	1008.791051	2.884071	996.4	1006.7	1008.6	1010.80	1019.1
PRESSÃO ATMOSFERICA MAX.NA HORA ANT. (AUT) (mB)	34316.0	1009.023662	2.871746	996.8	1007.0	1008.8	1011.00	1019.3
PRESSÃO ATMOSFERICA MIN. NA HORA ANT. (AUT) (mB)	34316.0	1008.558722	2.886149	996.2	1006.5	1008.3	1010.60	1019.1
RADIACAO GLOBAL (KJ/m²)	18611.0	1341.931422	1112.103873	0.0	280.1	1154.0	2247.55	4202.3
TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)	34316.0	25.732169	2.189678	19.3	24.2	25.5	27.00	33.8
TEMPERATURA DO PONTO DE ORVALHO (°C)	34316.0	21.711618	1.494763	15.2	20.7	21.9	22.80	26.6
TEMPERATURA MÁXIMA NA HORA ANT. (AUT) (°C)	34315.0	26.211246	2.321087	20.1	24.5	25.9	27.60	34.2
TEMPERATURA MÍNIMA NA HORA ANT. (AUT) (°C)	34316.0	25.274432	2.048771	19.1	23.8	25.1	26.50	32.4
TEMPERATURA ORVALHO MAX. NA HORA ANT. (AUT) (°C)	34316.0	22.165043	1.514813	15.8	21.1	22.3	23.20	27.4
TEMPERATURA ORVALHO MIN. NA HORA ANT. (AUT) (°C)	34316.0	21.268959	1.499910	14.5	20.2	21.4	22.40	25.4
UMIDADE REL. MAX. NA HORA ANT. (AUT) (%)	34316.0	81.164588	8.620830	49.0	76.0	82.0	88.00	97.0
UMIDADE REL. MIN. NA HORA ANT. (AUT) (%)	34316.0	76.953957	10.147333	40.0	70.0	78.0	84.00	97.0
UMIDADE RELATIVA DO AR, HORARIA (%)	34316.0	79.140080	9.427787	43.0	73.0	80.0	86.00	97.0
VENTO, DIREÇÃO HORARIA (gr) (° (gr))	34316.0	121.650076	66.857790	1.0	68.0	117.0	163.00	360.0
VENTO, RAJADA MAXIMA (m/s)	34316.0	5.037819	1.939719	0.5	3.8	4.8	6.10	16.8
VENTO, VELOCIDADE HORARIA (m/s)	34316.0	1.330368	0.670891	0.1	0.9	1.2	1.60	6.0

Figura 22. Fonte Autoria Própria.

Importante enfatizar a diferença de registros, na coluna count da figura 11 temos 4545072 enquanto a mesma coluna na figura 13 possui 1826, porém na primeira figura não tá filtrada para cidade de Salvador, quando é realizado o filtro, temos:

	count	mean	std	min	25%	50%	75%	max
temperature_2m_mean	1826.0	27.713473	1.869221	22.594217	26.313768	27.819698	29.040950	34.191418
temperature_2m_max	1826.0	24.487504	1.288919	20.107874	23.580047	24.663429	25.456020	27.537766
temperature_2m_min	1826.0	78.020217	7.743078	44.989944	73.502755	78.701453	83.550264	97.986930
wind_speed_10m_mean	1826.0	86.421988	7.218810	46.189903	82.820074	88.178195	91.505838	95.502040
wind_speed_10m_max	1826.0	68.205139	10.080415	35.005020	61.349067	68.018210	75.620114	99.254500
relative_humidity_2m_mean	1826.0	25.947967	1.340363	22.327370	24.846890	26.130876	27.021330	29.409084
relative_humidity_2m_max	1826.0	27.713473	1.869221	22.594217	26.313768	27.819698	29.040950	34.191418
relative_humidity_2m_min	1826.0	24.487504	1.288919	20.107874	23.580047	24.663429	25.456020	27.537766
precipitation_sum	1826.0	12.309181	2.240993	4.858694	10.889948	12.209179	13.634810	23.848590
rain_sum	1826.0	16.851117	2.847733	5.918991	15.218531	16.867107	18.617309	27.955820

Figura 23. Fonte Autoria Própria.

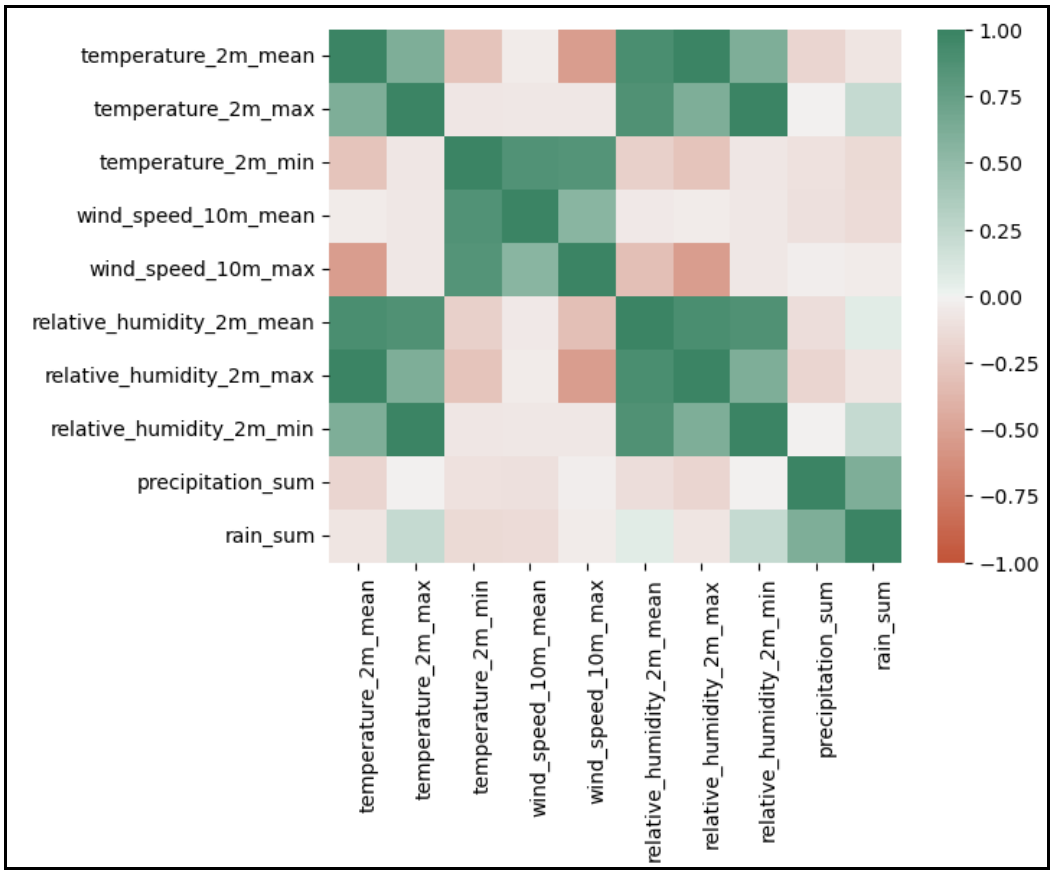

Figura 24. Fonte Autoria Própria.

No dataset da API já demonstra uma correlação negativa,sendo mais um ponto sobre a qualidade dos dados utilizados e por qual tratamento passam. Na figura 25 é possível verificar a correlação nesse dataset:

	temperature_2m_mean	te
temperature_2m_mean	1.000000	
temperature_2m_max	0.642067	
temperature_2m_min	-0.288211	
wind_speed_10m_mean	-0.040934	
wind_speed_10m_max	-0.540430	
relative_humidity_2m_mean	0.908871	
relative_humidity_2m_max	1.000000	
relative_humidity_2m_min	0.642067	
precipitation_sum	-0.210628	
rain_sum	-0.104029	

Figura 25. Fonte Autoria própria

Então a analise chegou aos gráficos abaixo que comprovam a baixa correlação entre ambas variáveis:

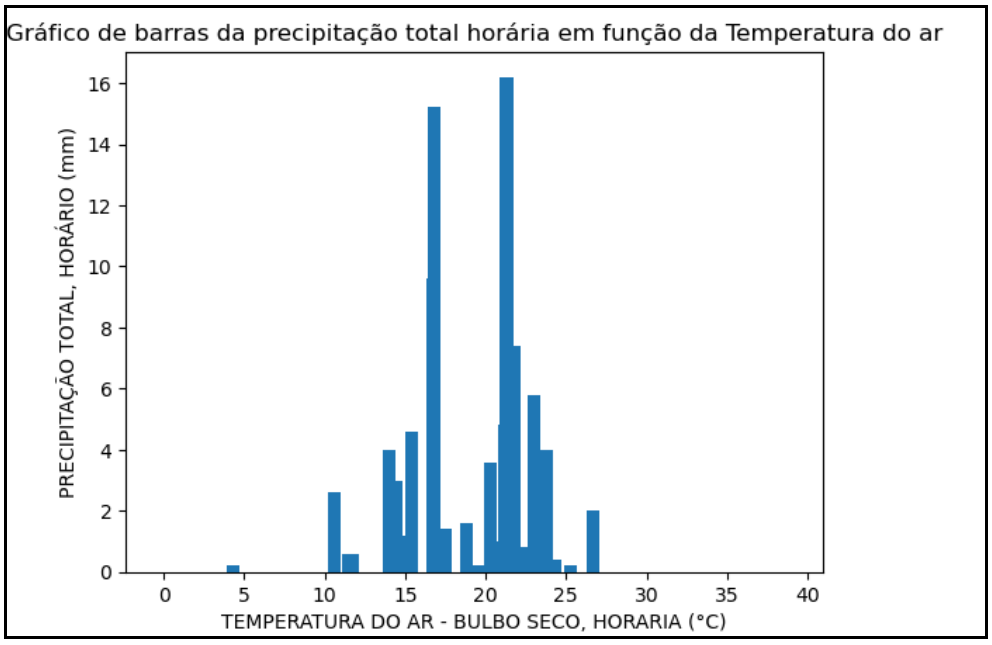

Figura 26. Fonte Autoria Própria.

É possivel verificar que houve precipitação quando a temperatura não estava nem tão alta, acima de 30ºC nem tão baixa exceto com um pequeno outlier que teve uma precipitação menor que 2 mm quando estava a 4ºC.Então os dados aqui expostos possuem entre si pequenas contradições quanto ao volume de chuva, como na figura 27:

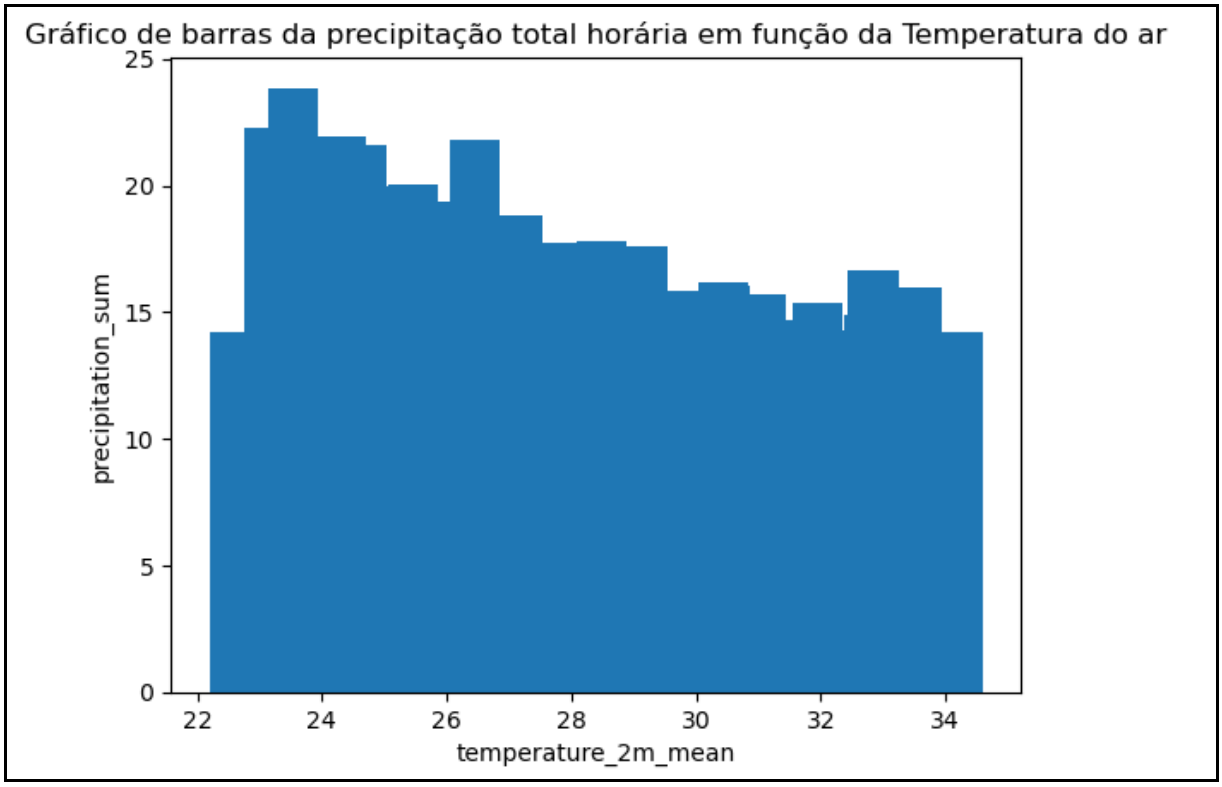

Figura 27. Fonte Autoria Própria

Através da analise dos dados exposta, alguns pontos a serem considerados:

1. Dados abertos são ótimos para analises mas não possuem um tratamento padrão que valide os dados e possa garantir que o que de fato aconteceu foi o que tava no dataset do Kaggle e não da API, vice-versa.

2. Os dados coletados algumas colunas não foram tratadas de forma correta ou não foram coletadas.

3. É possivel deduzir alguns fatores com base nas variaveis porém como será mostrado no capitulo 5 , prevê se terá chuva intensa não é possível.

Os dados aqui manipulados, explorados podem ser fontes de muitos insights para mais diferentes áreas, mesmo tendo que realizar alguns ajustes como a colunas com NaN. Ao conduzir análises exploratórias com esses dados, é possível identificar tendências, anomalias e relações complexas entre variáveis, contribuindo para a construção de modelos preditivos mais precisos. A visualização de dados desempenha um papel crucial nesse processo, permitindo a comunicação eficaz de resultados complexos e facilitando a interpretação por parte de diversas audiências.

A combinação de dados abertos do Kaggle e informações em tempo real provenientes da API Open Meteo proporciona uma base robusta para investigações profundas e compreensão abrangente dos fenômenos climáticos. Ao utilizar conjuntos de dados diversificados do Kaggle, provenientes de fontes diversas e confiáveis, os cientistas de dados podem explorar uma ampla gama de cenários e padrões, enriquecendo suas análises com informações detalhadas e contextualmente relevantes.

5. Criação de Modelos de Machine Learning

Para analisar os dados dos datasets selecionados foram utilizados os algoritmos K-Nearest Neighbors, Linear Regression, DecisionTreeClassifier, RandomForestClassifier. Esses algoritmos são algoritmos de aprendizado supervisionado, foram escolhidos de acordo com os objetivos deste trabalho, porém como da figura 26 é possível ver vários tipos de algoritmos que poderiam ser utilizados a depender do objetivo.

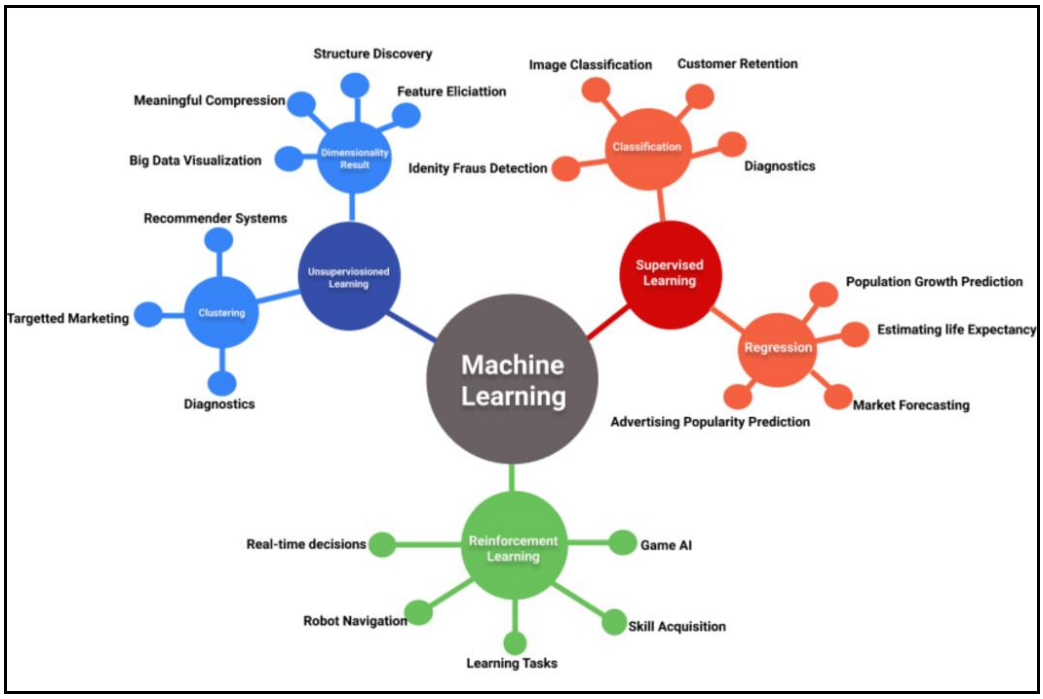

Figura 28. Fonte Autoria Própria

Então temos o modelo de previsão, primeiro a leitura do dataset a set utilizado, no caso o dataset_INMET-Salvador.csv que foi gerado durante a análise e contem somente registros da Estação de Salvador:

```
# Diretório com os 4 dataset do INMET agrupados em um unico dataset
diretorio = './Dados/INMET/Unificados/dataset_INMET-Salvador.csv'
diretorio api = './Dados/API/dados api ate 2022.csv'
```

Figura 29. Fonte Autoria Própria

Também foi feito a leitura do dataset da API mas somente até o ano de 2022 porque no dataset do INMET o ano de 2023 a coluna precipitação contem poucos registros de chuva:

```
Não choveu          1315
Choveu pouco         144
Choveu moderado        2
Name: CATEGORIA_PRECIPITACAO, dtype: int64
```

Figura 30. Fonte Autoria Própria

Enquanto no dataset da API :

```
Choveu moderado    1281
Choveu pouco        180
Name: CATEGORIA_PRECIPITACAO, dtype: int64
```

Figura 31. Fonte Autoria Própria

Como foi utilizado um modelo de regressão linear foi necessário a sintonização de hiperparâmetros pois é um processo importante ao desenvolver modelos de aprendizado de máquina. Os hiperparâmetros são configurações externas ao modelo que afetam seu desempenho e são definidos antes do treinamento. Exemplos comuns incluem a profundidade máxima de uma árvore de decisão, o número de vizinhos em um k-Nearest Neighbors.Então com esse objetivos, tem a função:

```python
#Sintonizando hiperparâmetros para o  modelo usando validação cruzada
def tuner(clf, dist, X, y):
  rs_clf = RandomizedSearchCV(estimator=clf, random_state=1, n_jobs=-1, param_distributions=dist, cv=10)

  search = rs_clf.fit(X=X, y=y)

  return search.best_params_
```

Figura 32. Fonte Autoria Própria

Foi criada logo em seguida mais três funções:

```
#Função para gerar e exibir uma matriz de confusão para avaliar o desempenho do modelo
def plot_confusion_matrix(y_true, y_pred, labels):
  cm = confusion_matrix(y_true=y_true, y_pred=y_pred, labels=labels)

  disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)

  disp.plot(cmap=plt.cm.Blues)

  plt.show()
```

Figura 33. Fonte Autoria Própria

```
#Função para gerar relatório para avaliação
def reporter(clf, X, y, labels):

  predictions = clf.predict(X)

  plot_confusion_matrix(y_true=y, y_pred=predictions, labels=labels)

  print(classification_report(y_true=y, y_pred=predictions))
```

Figura 34. Fonte Autoria Própria

A função da figura 33 tem como objetivo gerar as matrizes de confusão mais adiante e a da figura 34 é somente para gerar o relatório quanto as métricas.Então como é um modelo de classificação foi necessário criar uma nova coluna com variavés categóricas:

```
# Transformando dados numericos da coluna precipitação em labels de acordo com intervalos
def categorize_precipitation(value):
    if value == 0:
        return 'Não choveu'
    elif 0 < value <= 10:
        return 'Choveu pouco'
    elif 10 < value <= 30:
        return 'Choveu moderado'
    elif 30 < value <= 100:
        return 'Chuva intensa'
    else:
        return 'Valor fora dos intervalos especificados'

dados['CATEGORIA_PRECIPITACAO'] = dados['PRECIPITAÇÃO TOTAL, HORÁRIO (mm)'].apply(categorize_precipitation)


print(dados['CATEGORIA_PRECIPITACAO'].value_counts())

print(dados.head())
```

Figura 35. Fonte Autoria Própria

```python
# Transformando dados numericos da coluna precipitação em labels de acordo com intervalos
def categorize_precipitation(value):
    if value == 0:
        return 'Não choveu'
    elif 0 < value <= 10:
        return 'Choveu pouco'
    elif 10 < value <= 30:
        return 'Choveu moderado'
    elif 30 < value <= 100:
        return 'Chuva intensa'
    else:
        return 'Valor fora dos intervalos especificados'

dados_api['CATEGORIA_PRECIPITACAO'] = dados_api['precipitation_sum'].apply(categorize_precipitation)


print(dados_api['precipitation_sum'].value_counts())

print(dados_api.head())
```

Figura 37. Fonte Autoria Própria

Teve como resultado:

```
Name: precipitation_sum, Length: 1461, dtype: int64
         date  temperature_2m_mean  temperature_2m_max  temperature_2m_min  \
0  2019-01-01            26.815092           25.020996           85.708015
1  2019-01-02            28.864230           24.866518           82.718410
2  2019-01-03            27.511778           23.711008           84.713490
3  2019-01-04            28.759325           24.105497           82.708580
4  2019-01-05            29.206875           24.749987           81.703670

   wind_speed_10m_mean  wind_speed_10m_max  cloud_cover_mean  \
0            91.317510            77.23299          2.134795
1            92.345540            65.24041          4.727908
2            91.363510            80.23677          2.239490
3            92.381480            68.23312          0.470172
4            92.399445            66.22949          1.172232

   shortwave_radiation_sum  relative_humidity_2m_mean  \
0                      0.0                  25.799965
1                      0.0                  26.298769
2                      0.0                  25.646866
3                      0.0                  26.244965
4                      0.0                  26.693064

   relative_humidity_2m_max  relative_humidity_2m_min  precipitation_sum  \
0                 26.815092                 25.020996          13.768294
1                 28.864230                 24.866518          11.890161
2                 27.511778                 23.711008          12.206211
3                 28.759325                 24.105497          11.270189
4                 29.206875                 24.749987          10.646830

    rain_sum CATEGORIA_PRECIPITACAO
0  17.705230        Choveu moderado
1  16.821505        Choveu moderado
2  16.824470        Choveu moderado
3  15.941787        Choveu moderado
4  14.172985        Choveu moderado
```

Figura 38. Fonte Autoria Própria

Esse processo foi feito em ambos datasets o da figura 35 é dos dados do INMET e

da figura 36 é o dataset da API Open Meteo. Em seguida foi feito a contagem das classes que resultou nos resultados das figuras 30 e 31. Contagem das classes nos datasets da API e do INMET respectivamente:

```
contagem_categorias = dados_api['CATEGORIA_PRECIPITACAO'].value_counts()
contagem_categorias
```

Figura 39. Fonte Autoria Própria

```
contagem_categorias = dados['CATEGORIA_PRECIPITACAO'].value_counts()
contagem_categorias
```

Figura 40. Fonte Autoria Própria

Após o tratamento dos datasets, foi selecionado as features que é o que vai ser utilizado para prevê os targets:

```
# Selecionar características relevantes
features = dados_api[[ 'temperature_2m_mean', 'relative_humidity_2m_max', 'wind_speed_10m_mean']]
#Define a variavel a ser prevista
targets = dados['CATEGORIA_PRECIPITACAO']
```

Figura 41. Fonte Autoria Própria

Dividir o dataset em treino, validação e teste utilizando a proporção 30% do dataset para testes e 70% para treino e teste isso irá contribuir para evitar overfitting e um bom equilíbrio entre teste e treino.

```
X_train, X_test_valid, y_train, y_test_valid = train_test_split(features, targets,test_size=0.3)

X_test, X_valid, y_test, y_valid = train_test_split(X_test_valid, y_test_valid,test_size=0.7)
```

Figura 42. Fonte Autoria Própria

Criando uma lista chamada max_depths que contém valores de profundidade máxima para serem testados durante a sintonização de hiperparâmetros de um modelo de árvore de decisão.

```
max_depths = [i for i in range(1, 30, 2)]
```

Figura 43. Fonte Autoria Própria

Na figura 42 é um range de 1 á 29 que diz respeito a profundidade da arvore, buscando evitar muita profundidade para não ter overfitting. Isso é frequentemente usado em um processo de validação cruzada (cross-validation) para avaliar o desempenho do modelo de árvore de decisão com diferentes profundidades máximas e, assim, escolher a melhor profundidade que otimiza o desempenho do modelo no conjunto de dados específico.

Como exposto anteriormente o conjunto de dados não possui uma boa qualidade para o objetivo proposto neste TCC, então uma normalização dos conjuntos de dados foi feita usando o MinMaxScaler:

```
scaler = MinMaxScaler()

X_train_normalized = scaler.fit_transform(X_train)
X_valid_normalized = scaler.transform(X_valid)
X_test_normalized = scaler.transform(X_test)
```

Figura 44. Fonte Autoria Própria

Após isso iremos realizar um sintonização de hiperparâmetros, treinar e avaliar o modelo de K-Nearest Neighbors(KNN):

```
dist = dict(n_neighbors=[i for i in range(1, 100, 2)])
n_samples = X_test.shape[0]
knc = KNeighborsClassifier(n_jobs=-1)

best = tuner(knc, dist, X_valid_normalized, y_valid)

print(f'Melhores parâmetros: {best}')

knc = KNeighborsClassifier(n_jobs=-1, n_neighbors=best['n_neighbors'])

knc.fit(X_train_normalized, y_train)

labels = np.unique(y_train)

print('Train')
reporter(knc, X_train_normalized, y_train, labels)

print('Cross Validation')
reporter(knc, X_valid_normalized, y_valid, labels)

print('Test')
reporter(knc, X_test_normalized, y_test, labels)
```

Figura 45. Fonte Autoria Própria

O experimento obteve três resultados a serem discutidos no capitulo 5 deste trabalho. Porém foram todos utilizando o mesmo padrão da figura 45:

	precision	recall	f1-score	support
Choveu pouco	0.00	0.00	0.00	103
Não choveu	0.90	1.00	0.95	919
accuracy			0.90	1022
macro avg	0.45	0.50	0.47	1022
weighted avg	0.81	0.90	0.85	1022

Figura 46. Fonte Autoria Própria

A árvore de decisão é um algoritmo de aprendizado supervisionado adequado para problemas de classificação, pois é capaz de ordenar as classes em um nível preciso. Usa uma estrutura de árvore para representar um número de possíveis caminhos de decisão. Diferente de outros modelos a arvore de decisão pode ser útil para lidar com uma mistura de atributos números e categóricos e podem até classificar os da-

dos faltantes. Porem ao construir uma árvore de decisão é necessário decidir quais perguntas fazer e em qual ordem. Pois em cada etapa uma árvore há algumas possibilidades que eliminamos e outras que não. O KNN é um algoritmo de aprendizado de máquina utilizado para classificação e regressão. Baseia-se no princípio de que instâncias similares devem ter rótulos semelhantes. No KNN, os pontos de dados são classificados com base na maioria dos rótulos de seus vizinhos mais próximos. Pode ser computacionalmente caro para grandes conjuntos de dados, já que exige a computação das distâncias entre todos os pontos.

Foi utilizado também o RandomForestClassifier é um algoritmo de aprendizado supervisionado que cria uma “floresta” de árvores de decisão aleatórias e as combina para obter uma previsão mais precisa e estável. Ele é eficaz para evitar o sobreajuste, que é uma limitação comum das árvores de decisão.

```
dist = dict(max_depth=max_depths)

rfc = RandomForestClassifier(n_jobs=-1, random_state=1)

best = tuner(rfc, dist, X_valid, y_valid)

print(f'Melhores parâmetros: {best}')

rfc = RandomForestClassifier(n_jobs=-1, random_state=1, max_depth=best['max_depth'])

rfc.fit(X_train, y_train)

print('Train')
reporter(rfc, X_train_resampled, y_train_resampled, labels)
print('Cross Validation')
reporter(rfc, X_train_resampled, y_train_resampled, labels)
print('Test')
reporter(rfc, X_test, y_test, labels)
```

Figura 47. Fonte Autoria Própria

Em contra partida ao modelo de classificação foi feito um de previsão para vê como seria o comportamento dos dados. Foi utilizado e realizado a leitura dos mesmos dataset da figura 29. Porém neste modelo foi feito com somente uma amostra do dataset do INMET :

```
# Substituição de  registros NaN por 0
dados = dados.fillna(0)
dados =dados.sample(1461)
dados_api = dados_api.fillna(0)
```

Figura 48. Fonte Autoria Própria

Foi feito uma conversão de tipo do campo data:

```
#Transforma o tipo do campo data
dados['DATA (YYYY-MM-DD)'] = pd.to_datetime(dados['DATA (YYYY-MM-DD)'])
# Substituir 'A401' por 'Salvador' na coluna 'ESTACAO'
dados['ESTACAO'] = dados['ESTACAO'].replace({'A401': 'Salvador'})
```

Figura 49. Fonte Autoria Própria

Como pós analise já se tem o conhecimento da qualidade baixa dos dados, foi realizado primeiro somente com dados do INMET para no capitulo 6 discutir os resultados comparando somente dados do INMET e os resultados dos dados INMET e da API:

Previsão com dados do INMET

```
# Selecionar características relevantes
features = dados[[ 'TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)', 'UMIDADE RELATIVA DO AR, HORARIA (%
#Define a variavel a ser prevista
targets = dados['PRECIPITAÇÃO TOTAL, HORÁRIO (mm)']
```

Figura 50. Fonte Autoria Própria

Divisão e normalização dos dados:

```
 # Dividir em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(features, targets, test_size=0.4, random_state=42)


# Normalização dos dados

scaler = MinMaxScaler()

train_normalized = (X_train - X_train.mean())/X_train.std()
test_normalized = (X_test - X_test.mean())/X_test.std()

# Removendo a primeira linha a cada 1 hora (resultados)
train_result = train_normalized

# Removendo a última linha (não há resultados para a última linha)
train_normalized = train_normalized

# Removendo a primeira linha a cada 1 hora (resultados)
test_result = test_normalized

# Removendo a última linha (não há resultados para a última linha)
test_normalized = test_normalized

# Verificar os dados normalizados
print("Dados de treino normalizados:")
print(train_normalized[:-1])
print("\nDados de teste normalizados:")
print(test_normalized[:-1])
```

Figura 51. Fonte Autoria Própria

E como resultados, tivemos:

```
df_test = pd.DataFrame(test_normalized)
print(df_test)
```

```
      TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)  \
582                                       -0.390540
1314                                       2.156353
4844                                      -0.343375
3749                                      -0.626363
2965                                       0.882907
...                                             ...
1512                                       0.128272
1237                                       1.684706
2421                                       0.175437
4931                                      -0.673528
6826                                      -0.390540

      UMIDADE RELATIVA DO AR, HORARIA (%)  VENTO, VELOCIDADE HORARIA (m/s)
582                              0.734339                        -0.408957
1314                            -1.686767                        -0.231611
4844                            -0.423581                        -0.231611
3749                            -0.107785                         1.009811
2965                             0.102746                         0.300427
...                                   ...                              ...
1512                             0.102746                        -0.940994
1237                            -0.844643                        -0.586303
2421                            -0.213050                        -0.763648
4931                             0.208012                         0.477773
6826                            -0.528847                        -1.118340

[585 rows x 3 columns]
```

Figura 52. Fonte Autoria Própria

```
df_train = pd.DataFrame(train_normalized)
print(df_train)
```

```
      TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)  \
162                                       1.393885
7822                                     -0.221192
4505                                      0.317167
4352                                     -1.163321
3555                                     -0.849278
...                                            ...
7426                                      0.451757
1429                                      1.977108
4877                                     -1.297911
2403                                     -0.086602
2228                                      0.765799

      UMIDADE RELATIVA DO AR, HORARIA (%)  VENTO, VELOCIDADE HORARIA (m/s)
162                             -1.159233                         0.405484
7822                             0.135524                        -0.250479
4505                            -1.258830                        -0.578461
4352                             0.035928                         3.029338
3555                             0.932298                        -1.234424
...                                   ...                              ...
7426                            -0.262863                         0.733466
1429                            -1.159233                        -0.086488
4877                             0.235121                        -0.578461
2403                             0.235121                        -1.070434
2228                            -0.462056                        -0.250479

[876 rows x 3 columns]
```

Figura 53. Fonte Autoria Própria

Cria um modelo de regressão linear e treiná-lo com um conjunto específico de dados de treinamento (X_train e y_train).

```
modelo = LinearRegression()
modelo.fit(X_train, y_train)
```

Figura 54. Fonte Autoria Própria

Faz as previsões com o modelo treinado, calcule várias métricas de avaliação de desempenho (MSE, MAE, R-squared) e as exiba para avaliar quão bem o modelo se ajusta aos dados de teste.

```
y_pred = modelo.predict(X_test)

#
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean squared error:', mse)
print('Mean absolute error:', mae)
print('R-squared:', r2)
```

Figura 55. Fonte Autoria Própria

```
#Função que realiza calculo da metrica MAE
def root_mean_squared_error(x, y):
        return Kb.sqrt(Kb.mean(Kb.square(x - y), axis=-1))
```

Figura 56. Fonte Autoria Própria

O mean_squared_error é uma função do scikit-learn que calcula o erro médio quadrático (MSE) entre as previsões (y_pred) e os rótulos reais (y_test). O MSE é uma métrica comum para avaliar a qualidade de um modelo de regressão. Ele calcula a média dos quadrados das diferenças entre as previsões e os valores reais. Quanto menor o MSE, melhor o desempenho do modelo.

O r2_score calcula o coeficiente de determinação (R-squared) entre as previsões (y_pred) e os rótulos reais (y_test). O R-squared fornece uma medida de quão bem as previsões do modelo se ajustam aos dados reais. Ele varia de 0 a 1, onde 1 indica um ajuste perfeito. Quanto mais próximo de 1, melhor o modelo está ajustando os dados. mean_absolute_error é outra função do scikit-learn, e ela calcula o erro médio absoluto (MAE) entre as previsões (y_pred) e os rótulos reais (y_test). O MAE é a média das diferenças absolutas entre as previsões e os valores reais. Como o MSE, o MAE é uma métrica de erro, mas é menos sensível a valores discrepantes, pois não utiliza quadrados.

```
#Treinando Epoch
regularizer=tf.compat.v1.keras.regularizers.l2(0.001)
model = keras.Sequential([
layers.Dense(20, activation='relu', input_shape = (8,), kernel_regularizer=regularizer),
layers.Dense(12, activation='relu', kernel_regularizer=regularizer),
layers.Dense(4, activation='relu', kernel_regularizer=regularizer),
layers.Dense(1)
])

model.compile(loss='mean_squared_error', optimizer='adam',#Utilizando metrica MAE e MSE
              metrics=['mean_absolute_error', 'mean_squared_error', root_mean_squared_error])

model.fit(train_normalized, train_result,
  epochs=100, batch_size=1,validation_split = 0.0, verbose=1)
```

Figura 57. Fonte Autoria Própria

O modelo é uma rede neural sequencial com quatro camadas, sendo a primeira com 20 neurônios, a segunda com 12 neurônios, a terceira com 4 neurônios e a última com 1 neurônio, que é a camada de saída. O treinamento é realizado por 100 épocas, onde cada época representa um ciclo completo através de todo o conjunto de treinamento. O otimizador escolhido é o Adam, e a função de perda utilizada é a mean_squared_error, que é a média dos quadrados das diferenças entre as previsões do modelo e os rótulos reais. O treinamento é supervisionado pelos dados de entrada (train_normalized) e os resultados esperados (train_result). Além disso, é incorporada uma regularização L2 nos pesos das camadas ocultas com um parâmetro de regularização de 0.001 para evitar o overfitting. Durante o treinamento, são monitoradas métricas como o erro médio absoluto (mean_absolute_error), o erro quadrático médio (mean_squared_error), e uma métrica personalizada root_mean_squared_error. O objetivo é ajustar os pesos da rede de forma a minimizar a função de perda, tornando o modelo capaz de fazer previsões mais precisas sobre os dados de entrada.

Agora para o modelo é realizada diversas etapas em um contexto de treinamento e avaliação de um modelo de aprendizado de máquina. Inicialmente, o modelo é treinado usando o conjunto de treinamento (X_train e y_train). Durante o treinamento, o algoritmo ajusta seus parâmetros com base nos dados de treinamento para fazer previsões mais precisas. Em seguida, o modelo é utilizado para fazer previsões no

conjunto de teste (X_test), e essas previsões são comparadas com os valores reais (y_test).

Para avaliar o desempenho do modelo, são calculadas métricas de erro, como o Erro Quadrático Médio (MSE), o Erro Absoluto Médio (MAE) e o R^2 (coeficiente de determinação). Essas métricas quantificam a discrepância entre as previsões do modelo e os valores reais. A exibição dessas métricas proporciona uma compreensão do quão bem o modelo está performando.

Além disso, é utilizado o modelo treinado para fazer previsões adicionais no conjunto de teste, criando um DataFrame (df1) que armazena as previsões e os valores reais em colunas separadas. Isso permite uma análise mais detalhada das previsões, fornecendo uma visão tabular das discrepâncias entre as previsões do modelo e os dados reais. Em resumo, o código realiza treinamento, avaliação e visualização de um modelo de regressão, proporcionando insights sobre sua eficácia por meio de métricas e gráficos.

```python
# Treinando o modelo
model.fit(X_train, y_train)

# Fazendo previsões no conjunto de teste
y_pred = model.predict(X_test)

# Avaliando o desempenho do modelo
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean Squared Error:', mse)
print('Mean Absolute Error:', mae)
print('R-squared:', r2)

# Visualizando as previsões em relação aos valores reais
plt.plot(y_test.values, label='Actual Precipitation', color='blue')
plt.plot(y_pred, label='Predicted Precipitation', color='red')
plt.xlabel('Sample Index')
plt.ylabel('Precipitation')
plt.title('Actual vs Predicted Precipitation')
plt.legend()
plt.show()

# Realizando previsões utilizando sklearn
predictions = model.predict(X_test)
df1 = pd.DataFrame({"Pred": predictions.flatten(), "Real": y_test.values.flatten()})

# Visualizando as previsões em um DataFrame
print(df1)
```

Figura 58. Fonte Autoria Própria

```python
# Realizando previsões utilizando sklearn
predictions = model.predict(X_test)
df1 = pd.DataFrame(df1)

# Visualizando as previsões em um DataFrame
print(df1)
```

Figura 59. Fonte Autoria Própria

Depois é realizado o mesmo processo porem com ambos datasets INMET E OPEN METEO:

Previsão com dados do INMET e Open Meteo

```python
# Selecionar características relevantes
features = dados_api[[ 'temperature_2m_mean', 'relative_humidity_2m_max', 'wind_speed_10m_mean']]
#Define a variavel a ser prevista
targets = dados['PRECIPITAÇÃO TOTAL, HORÁRIO (mm)']
features
```

Figura 60. Fonte Autoria Própria

O esultado de ambos experimentos, classificação e previsão, será discutido no capitulo a seguir

6. Interpretação dos Resultados

No modelo de classificação tivemos os seguintes resultados:

```
dist = dict(n_neighbors=[i for i in range(1, 100, 2)])
n_samples = X_test.shape[0]
knc = KNeighborsClassifier(n_jobs=-1)

best = tuner(knc, dist, X_valid_normalized, y_valid)

print(f'Melhores parâmetros: {best}')

knc = KNeighborsClassifier(n_jobs=-1, n_neighbors=best['n_neighbors'])

knc.fit(X_train_normalized, y_train)

labels = np.unique(y_train)

print('Train')
reporter(knc, X_train_normalized, y_train, labels)

print('Cross Validation')
reporter(knc, X_valid_normalized, y_valid, labels)

print('Test')
reporter(knc, X_test_normalized, y_test, labels)
```

Figura 61. Fonte Autoria Própria

Com isso teve o seguinte resultado:

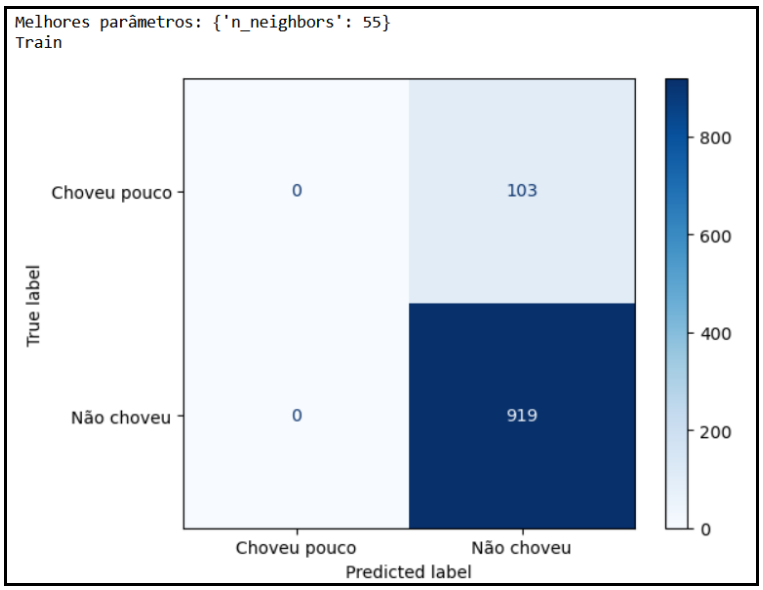

Figura 62. Fonte Autoria Própria

	precision	recall	f1-score	support
Choveu pouco	0.00	0.00	0.00	103
Não choveu	0.90	1.00	0.95	919
accuracy			0.90	1022
macro avg	0.45	0.50	0.47	1022
weighted avg	0.81	0.90	0.85	1022

Figura 63. Fonte Autoria Própria

Pelo relatório é possível perceber que o modelo não previu bem quando choveu pouco, nem identificou nenhum, porém o de Não Choveu ele acertou com uma precisão de 90%, identificou todas as vezes. Com um F1-score baixo, pois é uma média harmônica de precisão e recall. Neste caso, a precisão é zero, o que impacta negativamente o F1-score. Já a classe Não Choveu teve um F1 alto indicando um bom equilíbrio entre precisão e recall.

As métricas macros e ponderada (macro avg e weighted avg) são médias das métricas para cada classe. A macro avg atribui igual peso a cada classe, enquanto a weighted avg leva em consideração o desequilíbrio de classes.

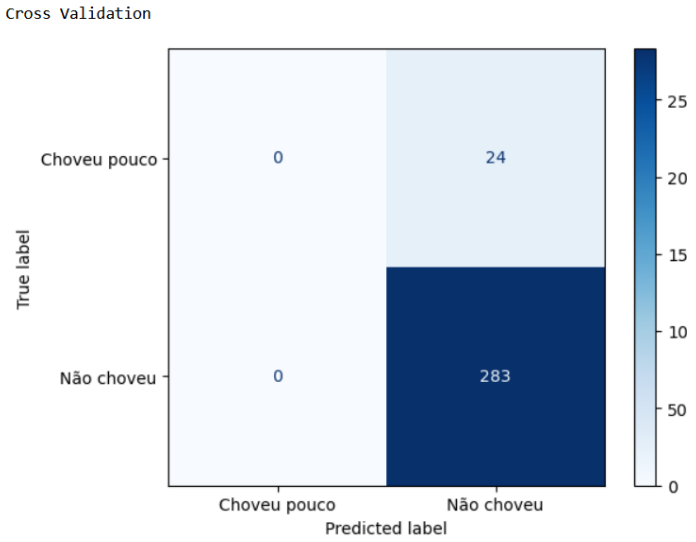

Figura 64. Fonte Autoria Própria

	precision	recall	f1-score	support
Choveu moderado	0.00	0.00	0.00	1
Choveu pouco	0.00	0.00	0.00	24
Não choveu	0.92	1.00	0.96	283
accuracy			0.92	308
macro avg	0.31	0.33	0.32	308
weighted avg	0.84	0.92	0.88	308

Figura 65. Fonte Autoria Própria

Agora foi obtido resultados diferentes. Para a classe "Choveu moderado" e "Choveu pouco", a precisão é zero (0.00). Isso significa que o modelo não previu corretamente nenhuma instância dessas classes. Para a classe "Choveu moderado" e "Choveu pouco", a recall é zero (0.00). Isso significa que o modelo não conseguiu recuperar corretamente nenhuma instância dessas classes. O F1-Score para a classe "Choveu moderado" e "Choveu pouco" é zero, o que reflete a falta de previsões corretas para essas classes.

Para a classe "Não choveu", a revocação é alta (1.00), indicando que o modelo recuperou corretamente todas as instâncias dessa classe. O F1-Score para a classe é alto (0.96), indicando um bom equilíbrio entre precisão e recall para essa

classe. A acurácia geral do modelo é de 92% e A média ponderada das métricas (precision, recall, f1-score) mostra valores relativamente altos, principalmente devido à alta acurácia para a classe majoritária "Não choveu". A média não ponderada das métricas (macro avg) mostra valores mais baixos, indicando que o desempenho do modelo é mais desafiador quando consideramos todas as classes de maneira uniforme.

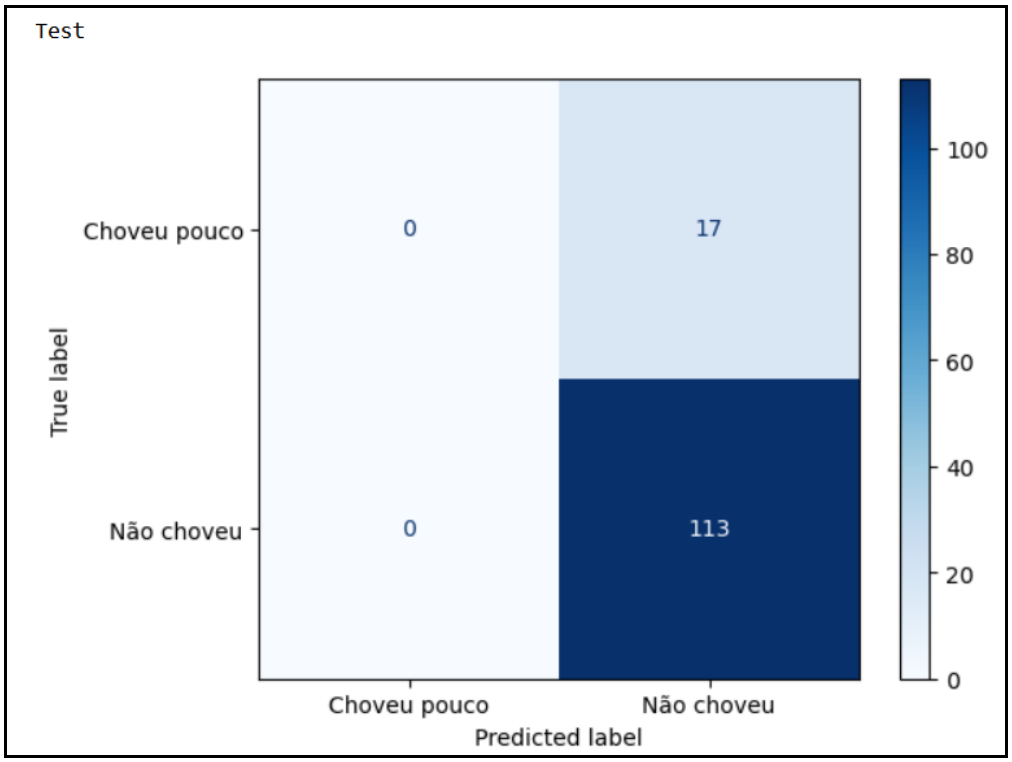

Figura 66. Fonte Autoria Própria

	precision	recall	f1-score	support
Choveu moderado	0.00	0.00	0.00	1
Choveu pouco	0.00	0.00	0.00	17
Não choveu	0.86	1.00	0.93	113
accuracy			0.86	131
macro avg	0.29	0.33	0.31	131
weighted avg	0.74	0.86	0.80	131

Figura 67. Fonte Autoria Própria

Com esse último resultado tivemos um desempenho elevado na previsão da classe "Não choveu". A precisão de 0.86 indica que, dentre as instâncias previstas como "Não choveu", 86% são corretas. O recall de 1.00 mostra que o modelo identificou corretamente todas as instâncias reais de "Não choveu". O F1-score, que combina precisão e recall, é alto em 0.93. O modelo não conseguiu prever corretamente a classe "Choveu pouco", pois todas as métricas (precision, recall, F1-score) são zero. Isso sugere que o modelo não está conseguindo identificar adequadamente os casos dessa classe. Similar à classe "Choveu pouco", o modelo não acerta nenhum caso da classe "Choveu moderado", com todas as métricas sendo zero.

Com o intuito de melhorar os resultados do modelo, diminuir a diferença entre as classes foi gerado dados sintéticos através de SMOTE (Synthetic Minority Oversampling Technique) com a estratégia ‘auto’ para ajustar automaticamente a proporção entre as classes:

```
# Aplicando SMOTE para balancear as classes
smote = SMOTE(sampling_strategy='auto', k_neighbors=3, random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)


# Treinando um modelo (por exemplo, RandomForest)

clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_resampled, y_train_resampled)

# Fazendo previsões
y_pred = clf.predict(X_test)

# Avaliando o desempenho
print(classification_report(y_test, y_pred))
```

Figura 68. Fonte Autoria Própria

	precision	recall	f1-score	support
Choveu moderado	0.00	0.00	0.00	1
Choveu pouco	0.13	0.35	0.19	17
Não choveu	0.87	0.65	0.75	113
accuracy			0.61	131
macro avg	0.33	0.34	0.31	131
weighted avg	0.77	0.61	0.67	131

Figura 69. Fonte Autoria Própria

A precisão indica a proporção de instâncias positivas identificadas corretamente em relação ao total de instâncias identificadas como positivas. Valores mais altos indicam melhor precisão. O recall, também conhecido como sensibilidade, indica a proporção de instâncias positivas que foram identificadas corretamente em relação ao total de instâncias positivas. Valores mais altos indicam melhor capacidade de recuperar instâncias positivas. O F1-score é a média harmônica entre precisão e recall, o modelo tem dificuldade em equilibrar precisão e recall para todas as classes. A acurácia indica a proporção de instâncias corretamente classificadas em relação ao total de instâncias. Valores mais altos indicam melhor desempenho geral do modelo. A acurácia geral é razoável, mas a interpretação do desempenho do modelo deve levar em consideração as características específicas do problema e as necessidades de cada aplicação.

Utilizando os dados sintéticos para fazer o mesmo experimento de arvore de decisão, porém com os novos dados:

```
dist = dict(max_depth=max_depths)

rfc = RandomForestClassifier(n_jobs=-1, random_state=1)

best = tuner(rfc, dist, X_valid, y_valid)

print(f'Melhores parâmetros: {best}')

rfc = RandomForestClassifier(n_jobs=-1, random_state=1, max_depth=best['max_depth'])

rfc.fit(X_train, y_train)

print('Train')
reporter(rfc, X_train_resampled, y_train_resampled, labels)
print('Cross Validation')
reporter(rfc, X_train_resampled, y_train_resampled, labels)
print('Test')
reporter(rfc, X_test, y_test, labels)
```

Figura 70. Fonte Autoria Própria

Tendo como resultados:

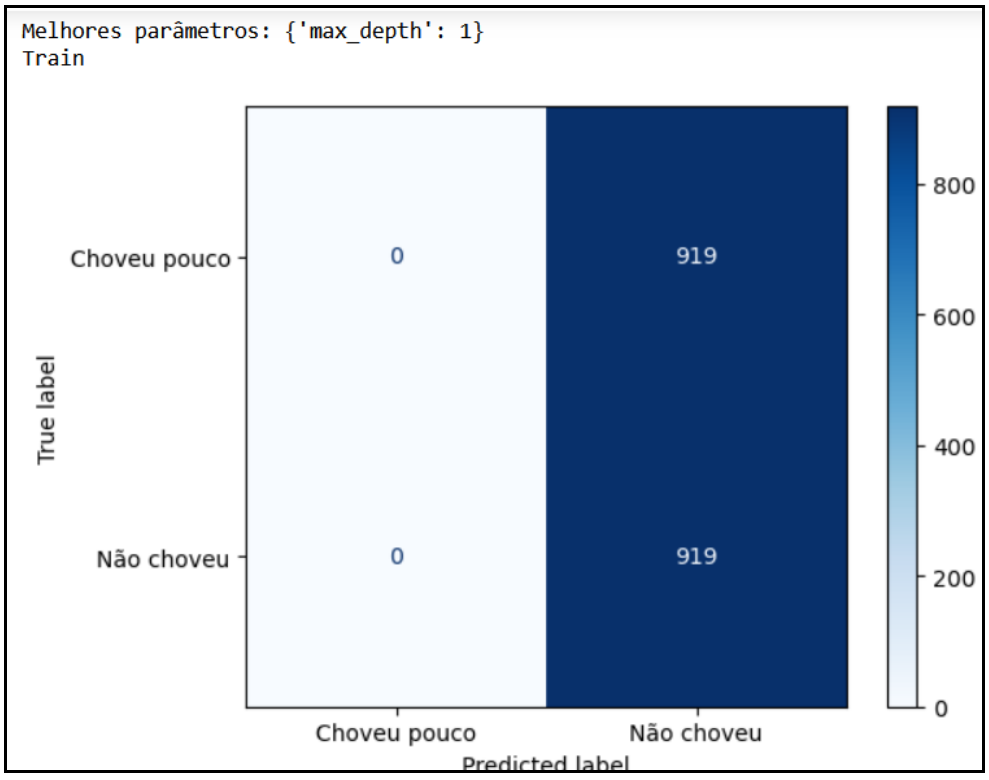

Figura 71. Fonte Autoria Própria

	precision	recall	f1-score	support
Choveu pouco	0.00	0.00	0.00	919
Não choveu	0.50	1.00	0.67	919
accuracy			0.50	1838
macro avg	0.25	0.50	0.33	1838
weighted avg	0.25	0.50	0.33	1838

Figura 72. Fonte Autoria Própria

Aparentemente falhou na previsão da classe "Choveu pouco". Com uma precision, recall e F1-Score de zero para essa classe, o modelo não conseguiu identificar corretamente nenhum caso de chuva fraca. Por outro lado, o desempenho para a classe "Não choveu" é melhor, com uma precision de 0.50 e recall de 1.00, indicando que o modelo é mais eficaz na previsão da ausência de chuva. A acurácia geral é 0.50, refletindo a precisão balanceada entre as duas classes, mas essa média pode ser enganadora, já que a incapacidade do modelo em prever "Choveu pouco" é preocupante. Em suma, é crucial aprimorar a capacidade do modelo em identificar chuvas leves, possivelmente através de ajustes nos dados de treinamento, seleção de características relevantes ou alterações no algoritmo de aprendizado utilizado.

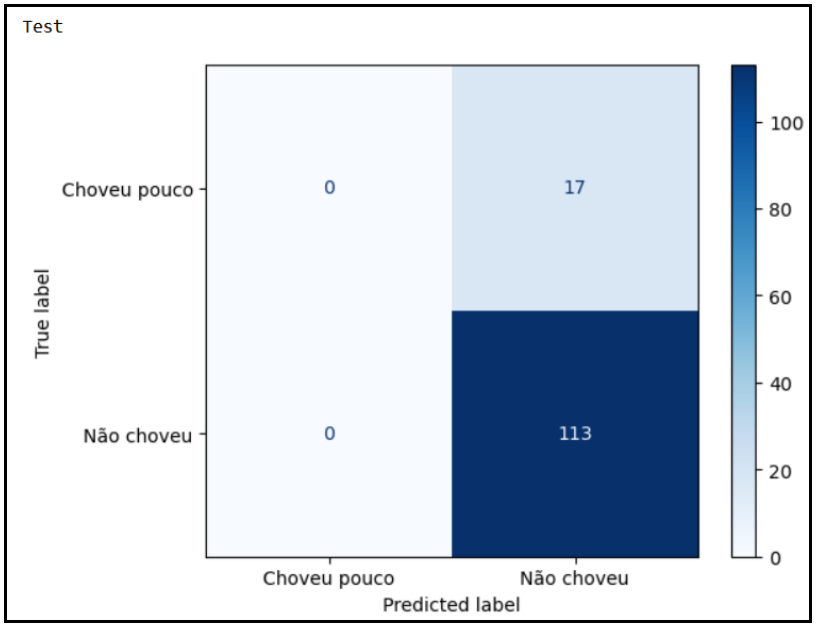

Figura 73. Fonte Autoria Própria

	precision	recall	f1-score	support
Choveu moderado	0.00	0.00	0.00	1
Choveu pouco	0.00	0.00	0.00	17
Não choveu	0.86	1.00	0.93	113
accuracy			0.86	131
macro avg	0.29	0.33	0.31	131
weighted avg	0.74	0.86	0.80	131

Figura 74. Fonte Autoria Própria

Para a classe "Choveu moderado", "Choveu pouco" e "Não choveu", as precisões são 0.00, 0.00 e 0.86, respectivamente. Isso indica que o modelo tem um bom desempenho em prever quando não chove, mas tem problemas sérios nas previsões de chuvas moderadas e pequenas. O recall para "Não choveu" é excelente, indicando que o modelo está identificando corretamente a ausência de chuva. O F1-Score com valor alto para "Não choveu" mostra que o modelo está equilibrando bem precision e recall para essa classe. a acurácia é 0.86, indicando que o modelo está correto em 86% das previsões.

O mesmo experimento, mas com os dados da API como targets e o do INMET como features. Obteve os resultados:

```
# Selecionar características relevantes
features = dados[[ 'TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)', 'UMIDADE RELATIVA DO AR, HORARIA (%)', 'VENTO, VELOCIDADE HORA
#Define a variavel a ser prevista
targets = dados_api['CATEGORIA_PRECIPITACAO']
```

Figura 75. Fonte Autoria Própria

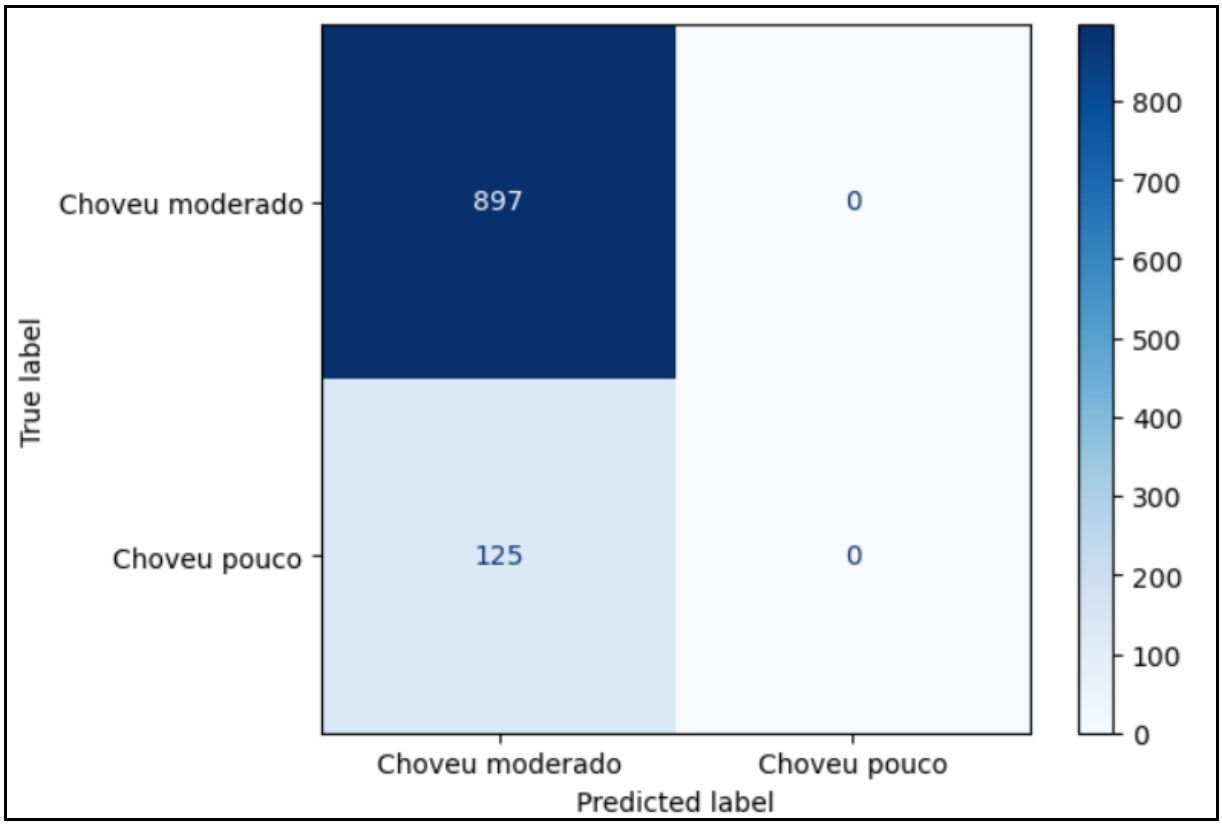

Figura 75. Fonte Autoria Própria

	precision	recall	f1-score	support
Choveu moderado	0.88	1.00	0.93	897
Choveu pouco	0.00	0.00	0.00	125
accuracy			0.88	1022
macro avg	0.44	0.50	0.47	1022
weighted avg	0.77	0.88	0.82	1022

Figura 76. Fonte Autoria Própria

As métricas de desempenho do modelo apresentado revela uma precisão global de 88%, sugerindo uma capacidade razoável de identificar corretamente as condições meteorológicas de chuva moderada. No entanto, a precisão para prever chuva leve é preocupantemente baixa, atingindo 0%. Isso indica uma limitação significativa do modelo na detecção de chuvas leves. Além disso, o recall para chuva leve também é nulo, indicando que o modelo não conseguiu identificar corretamente nenhum caso de chuva leve entre as amostras. A acurácia global de 88% pode mascarar a incapacidade do modelo de lidar efetivamente com a classe "Choveu pouco". As métricas macro avg e weighted avg apontam para um desempenho médio e ponderado, respectivamente, destacando a importância de considerar ambas as classes ao avaliar o desempenho geral do modelo. Em resumo, enquanto o modelo demonstra uma boa capacidade de prever chuva moderada, é crucial aprimorar sua sensibilidade para detectar chuvas leves, a fim de garantir uma performance mais equilibrada e confiável em diversas condições meteorológicas.

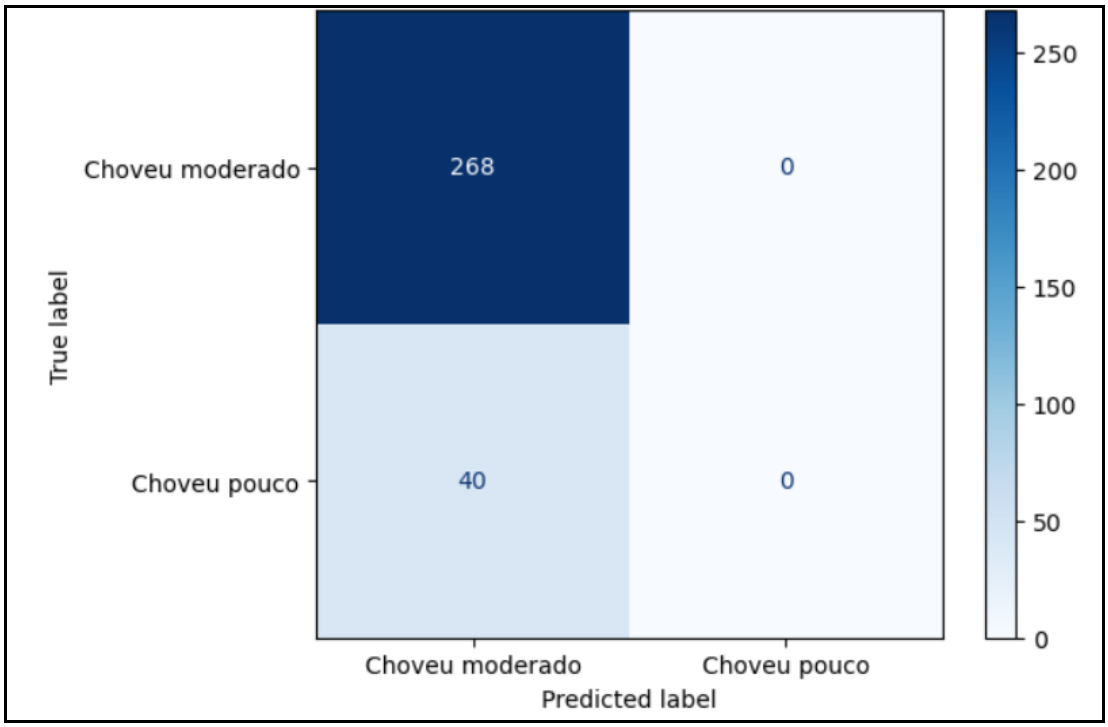

Figura 77. Fonte Autoria Própria

	precision	recall	f1-score	support
Choveu moderado	0.87	1.00	0.93	268
Choveu pouco	0.00	0.00	0.00	40
accuracy			0.87	308
macro avg	0.44	0.50	0.47	308
weighted avg	0.76	0.87	0.81	308

Figura 78. Fonte Autoria Própria

As métricas de desempenho apresentadas, revela informações importantes sobre o modelo utilizado para classificar eventos de chuva moderada e chuva fraca. Observa-se que o modelo demonstra um desempenho notável na identificação de eventos de chuva moderada, com uma precision de 0.87, recall de 1.00 e f1-score de 0.93. Isso sugere que, quando o modelo prevê chuva moderada, ele tende a estar correto na grande maioria das vezes e também captura efetivamente todos os casos reais de chuva moderada.

No entanto, ao analisar a classe "Choveu pouco", percebe-se um desempenho insatisfatório, com precision, recall e f1-score todos iguais a 0.00. Isso indica que o modelo não foi capaz de identificar corretamente nenhum caso de chuva fraca, o que pode ser atribuído a limitações na capacidade do modelo em lidar com esse tipo específico de evento.

A acurácia global do modelo é de 0.87, refletindo a precisão média entre as duas classes. A macro avg e weighted avg das métricas apontam para um desempenho geral que não é uniforme entre as classes, indicando a necessidade de aprimoramentos, especialmente na identificação de chuva fraca.

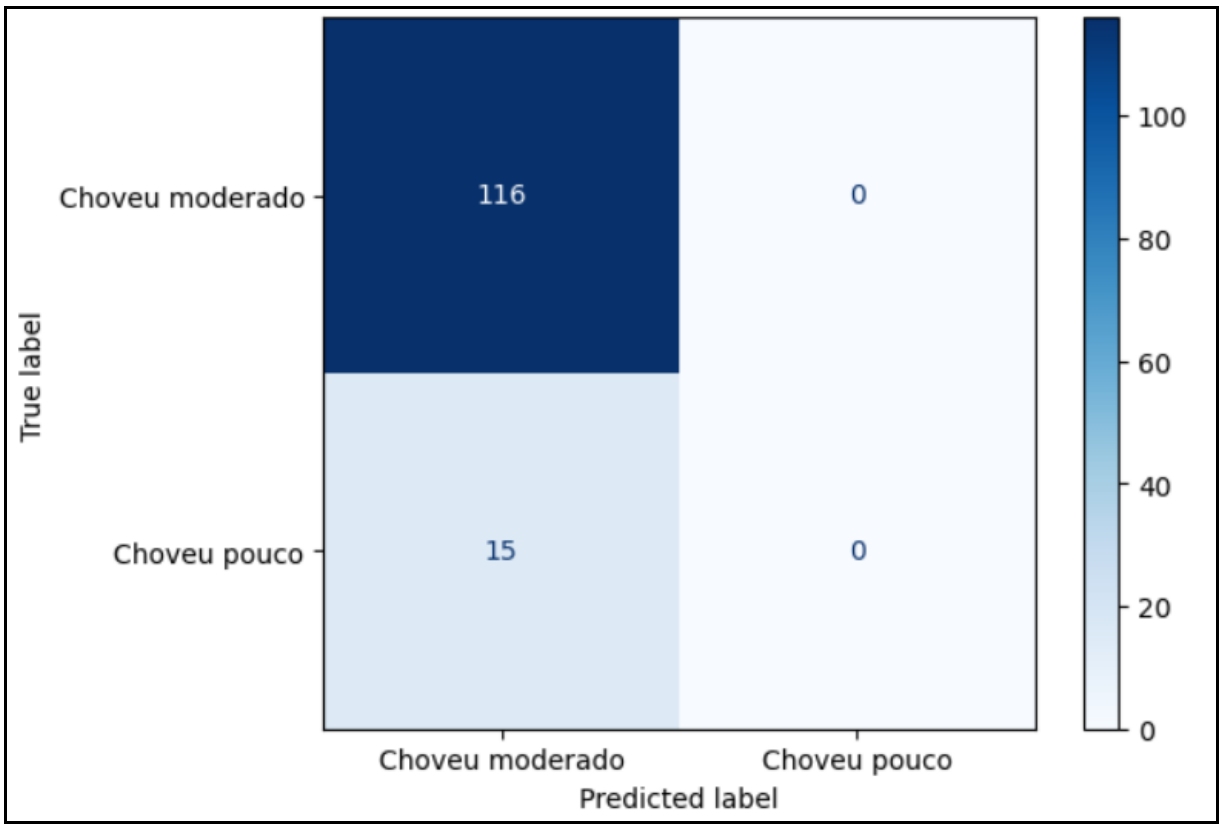

Figura 79. Fonte Autoria Própria

	precision	recall	f1-score	support
Choveu moderado	0.89	1.00	0.94	116
Choveu pouco	0.00	0.00	0.00	15
accuracy			0.89	131
macro avg	0.44	0.50	0.47	131
weighted avg	0.78	0.89	0.83	131

Figura 80. Fonte Autoria Própria

A análise das métricas de desempenho de um modelo de classificação apresenta algumas conclusões importantes. Inicialmente, observamos que o modelo demonstra uma alta precisão (0.89) para a classe "Choveu moderado", indicando que, quando prevê a ocorrência de chuva moderada, está correta em 89% das vezes.

Além disso, a recall para essa classe é de 1.00, o que sugere que o modelo é capaz de identificar efetivamente todos os casos de chuva moderada.

No entanto, a performance para a classe "Choveu pouco" é problemática. Com uma precisão e recall de 0.00, o modelo não é capaz de identificar corretamente casos de chuva leve. Isso pode indicar um desequilíbrio nos dados de treinamento, falta de representação adequada para a classe minoritária ou limitações intrínsecas ao modelo para detectar padrões específicos associados a chuvas leves.

A acurácia geral do modelo é de 0.89, o que inicialmente pode parecer satisfatório, mas é importante considerar o desafio representado pela classe minoritária. A macro avg e weighted avg para precision, recall e f1-score mostram uma performance média, sugerindo que o modelo tem dificuldades em generalizar para ambas as classes de forma equilibrada.

Para melhorar o desempenho, seria necessário explorar estratégias como o ajuste de hiperparâmetros, a coleta de mais dados para a classe minoritária, ou mesmo a utilização de técnicas avançadas como o uso de modelos mais complexos. Agora os resultados do modelo de previsão, mesmo método primeiro só com dados do inmet depois com ambos datasets INMET e API Open Meteo

Previsão com dados do INMET

```
# Selecionar características relevantes
features = dados[[ 'TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)', 'UMIDADE RELATIVA DO AR, HORARIA (%)', 'VENTO, VELOCIDADE HORA
#Define a variavel a ser prevista
targets = dados['PRECIPITAÇÃO TOTAL, HORÁRIO (mm)']
```

Figura 81. Fonte Autoria Própria

Depois de todo tratamento feito com os dados, como mostrado no capitulo 5, primeiro resultado das métricas:

```
y_pred = modelo.predict(X_test)

#
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean squared error:', mse)
print('Mean absolute error:', mae)
print('R-squared:', r2)
```

```
Mean squared error: 0.7813943273685928
Mean absolute error: 0.3529619602455774
R-squared: 0.0941128333543475
```

Figura 82. Fonte Autoria Própria

A avaliação de um modelo de regressão é fundamental para compreender sua capacidade de prever dados. Os resultados apresentados revelam que o modelo possui um Mean Squared Error (Erro Quadrático Médio) de 0.7814, indicando que, em média, os quadrados das diferenças entre as previsões e os valores reais são relativamente altos. O Mean Absolute Error (Erro Médio Absoluto) de 0.353 demonstra uma certa precisão nas previsões, sendo menor em magnitude do que o MSE. No entanto, o R-squared de 0.0941 indica que apenas uma pequena proporção da variabilidade nos dados é explicada pelo modelo, sugerindo uma capacidade limitada de capturar a complexidade do fenômeno em estudo.

Logo depois, o treinamento das epoch. As epoch é a quantidade de vezes que todo o conjunto de dados de treinamento é utilizado para atualizar os pesos da rede neural. Cada época é composta por iterações, onde o modelo faz previsões, calcula a perda (erro), e ajusta seus parâmetros para minimizar essa perda. Ao longo das épocas, o modelo aprende padrões nos dados, refinando sua capacidade de fazer previsões. Um número adequado de épocas é crucial para evitar

underfitting (modelo subajustado) ou overfitting (modelo superajustado). Monitorar métricas, como a perda, ao longo das épocas ajuda a avaliar o desempenho do modelo e garantir sua eficácia na generalização para dados não vistos.

```
#Treinando Epoch
regularizer=tf.compat.v1.keras.regularizers.l2(0.001)
model = keras.Sequential([
layers.Dense(20, activation='ELU', input_shape = (8,), kernel_regularizer=regularizer),
layers.Dense(12, activation='ELU', kernel_regularizer=regularizer),
layers.Dense(4, activation='ELU', kernel_regularizer=regularizer),
layers.Dense(1)
])

model.compile(loss='mean_squared_error', optimizer='adam',#Utilizando metrica MAE e MSE
              metrics=['mean_absolute_error', 'mean_squared_error', root_mean_squared_error])

model.fit(train_normalized, train_result,
  epochs=100, batch_size=1,validation_split = 0.0, verbose=1)
```

```
Epoch 1/100
876/876 [==============================] - 2s 2ms/step - loss: 0.5248 - mean_absolute_error: 0.5342 - mean_squared_error: 0.524
8 - root_mean_squared_error: 0.5781
Epoch 2/100
876/876 [==============================] - 1s 2ms/step - loss: 0.4522 - mean_absolute_error: 0.4945 - mean_squared_error: 0.452
2 - root_mean_squared_error: 0.5315
Epoch 3/100
876/876 [==============================] - 1s 2ms/step - loss: 0.4494 - mean_absolute_error: 0.4931 - mean_squared_error: 0.449
4 - root_mean_squared_error: 0.5268
Epoch 4/100
876/876 [==============================] - 1s 2ms/step - loss: 0.4483 - mean_absolute_error: 0.4925 - mean_squared_error: 0.448
3 - root_mean_squared_error: 0.5245
Epoch 5/100
876/876 [==============================] - 1s 2ms/step - loss: 0.4478 - mean_absolute_error: 0.4927 - mean_squared_error: 0.447
8 - root_mean_squared_error: 0.5239
Epoch 6/100
876/876 [==============================] - 1s 1ms/step - loss: 0.4480 - mean_absolute_error: 0.4930 - mean_squared_error: 0.448
0 - root_mean_squared_error: 0.5242
```

Figura 82. Fonte Autoria Própria

Nas 100 épocas do treinamento indicam que o modelo passou por uma diminuição gradual na função de perda (loss) ao longo do tempo. Inicialmente, na primeira época, a loss foi de 0.5248, e essa métrica foi decrescendo até atingir um valor de 0.4472 na última época. O mesmo padrão é observado nas métricas adicionais, como mean_absolute_error, mean_squared_error e root_mean_squared_error, que também apresentaram uma redução consistente ao longo das épocas.

É importante destacar que a convergência dessas métricas sugere que o modelo está aprendendo e ajustando-se aos dados de treinamento. No entanto, é essencial avaliar o desempenho do modelo em conjuntos de validação ou teste para verificar se ele generaliza bem para dados não vistos durante o treinamento.A consistência na diminuição da loss ao longo das épocas indica que o modelo está progredindo de maneira estável.

Então começa o treinamento do modelo:

```
# Treinando o modelo
model.fit(X_train, y_train)

# Fazendo previsões no conjunto de teste
y_pred = model.predict(X_test)

# Avaliando o desempenho do modelo
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean Squared Error:', mse)
print('Mean Absolute Error:', mae)
print('R-squared:', r2)

# Visualizando as previsões em relação aos valores reais
plt.plot(y_test.values, label='Actual Precipitation', color='blue')
plt.plot(y_pred, label='Predicted Precipitation', color='red')
plt.xlabel('Sample Index')
plt.ylabel('Precipitation')
plt.title('Actual vs Predicted Precipitation')
plt.legend()
plt.show()

# Realizando previsões utilizando sklearn
predictions = model.predict(X_test)
df1 = pd.DataFrame({"Pred": predictions.flatten(), "Real": y_test.values.flatten()})

# Visualizando as previsões em um DataFrame
print(df1)
```

Figura 83. Fonte Autoria Própria

E temos os resultados:

```
28/28 [==============================] - 1s 3ms/step - loss: 224.6268 - mean_absolute_error: 11.2177 - mean_squared_error:
224.6268 - root_mean_squared_error: 11.2177
19/19 [==============================] - 0s 3ms/step
Mean Squared Error: 25.688933529551132
Mean Absolute Error: 4.943961664998634
R-squared: -28.781730419775933
```

Figura 84. Fonte Autoria Própria

Os resultados apresentados revelam que o modelo possui algumas deficiências em sua capacidade de previsão. A métrica de loss, com um valor de 224.6268, indica uma considerável disparidade entre as previsões do modelo e os valores reais. As métricas adicionais, como Mean Squared Error (MSE) de 25.6889 e Mean Absolute Error (MAE) de 4.9439, apontam para uma precisão moderada, mas ainda existem áreas para melhoria. O R-squared negativo de -28.7817 sugere que o modelo não está conseguindo explicar a variabilidade dos dados de maneira adequada. Isso indica a necessidade de ajustes na arquitetura do modelo, possíveis revisões nos hiperparâmetros ou a inclusão de mais dados para aprimorar seu desempenho.

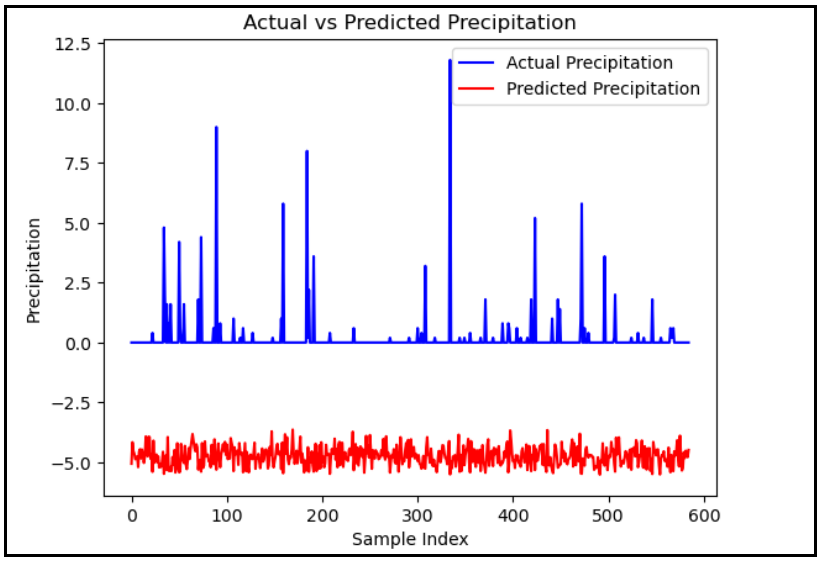

Figura 85. Fonte Autoria Própria

Seguindo agora para dados do INMET com do Open METEO:

Previsão com dados do INMET e Open Meteo

```
# Selecionar características relevantes
features = dados_api[[ 'temperature_2m_mean', 'relative_humidity_2m_max', 'wind_speed_10m_mean']]
#Define a variavel a ser prevista
targets = dados['PRECIPITAÇÃO TOTAL, HORÁRIO (mm)']
features
```

Figura 86. Fonte Autoria Própria

```
y_pred = modelo.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean squared error:', mse)
print('Mean absolute error:', mae)
print('R-squared:', r2)
```

```
Mean squared error: 0.8618109450142968
Mean absolute error: 0.33090119752214964
R-squared: 0.0008841274900289742
```

Figura 87. Fonte Autoria Própria

Os resultados são sobre desempenho do modelo, indicando que a métrica Mean Squared Error (MSE) atingiu um valor de 0.8618, refletindo uma baixa dispersão entre as previsões e os valores reais. Da mesma forma, o Mean Absolute Error (MAE) de 0.3309 sugere uma precisão moderada nas previsões do modelo, com erros médios relativamente baixos. No entanto, o R-squared de 0.0009 indica uma explicação muito limitada da variabilidade dos dados pelo modelo, sugerindo que a capacidade de previsão é mínima. Essa baixa correlação entre as previsões e os resultados reais pode indicar que o modelo não está capturando adequadamente os padrões nos dados. Portanto, ajustes na arquitetura do modelo, revisão de hiperparâmetros ou a inclusão de mais dados podem ser necessários para melhorar a capacidade de generalização do modelo.

```
#Treinando Epoch
regularizer=tf.compat.v1.keras.regularizers.l2(0.001)
model = keras.Sequential([
layers.Dense(20, activation='ELU', input_shape = (8,), kernel_regularizer=regularizer),
layers.Dense(12, activation='ELU', kernel_regularizer=regularizer),
layers.Dense(4, activation='ELU', kernel_regularizer=regularizer),
layers.Dense(1)
])

model.compile(loss='mean_squared_error', optimizer='adam',#Utilizando metrica MAE e MSE
              metrics=['mean_absolute_error', 'mean_squared_error', root_mean_squared_error])

model.fit(train_normalized, train_result,
  epochs=100, batch_size=1,validation_split = 0.0, verbose=1)
```

```
Epoch 1/100
876/876 [==============================] - 2s 2ms/step - loss: 0.5248 - mean_absolute_error: 0.5342 - mean_squared_error:
0.5248 - root_mean_squared_error: 0.5781
Epoch 2/100
876/876 [==============================] - 1s 2ms/step - loss: 0.4522 - mean_absolute_error: 0.4945 - mean_squared_error:
0.4522 - root_mean_squared_error: 0.5315
Epoch 3/100
876/876 [==============================] - 1s 2ms/step - loss: 0.4494 - mean_absolute_error: 0.4931 - mean_squared_error:
0.4494 - root_mean_squared_error: 0.5268
Epoch 4/100
876/876 [==============================] - 1s 2ms/step - loss: 0.4483 - mean_absolute_error: 0.4925 - mean_squared_error:
0.4483 - root_mean_squared_error: 0.5245
Epoch 5/100
876/876 [==============================] - 1s 2ms/step - loss: 0.4478 - mean_absolute_error: 0.4927 - mean_squared_error:
0.4478 - root_mean_squared_error: 0.5239
```

Figura 88. Fonte Autoria Própria

Os resultados das métricas do modelo ao longo de 100 epochs indicam uma progressão consistente em direção à otimização. No início, o valor do Mean Squared Error (MSE) foi significativamente alto, atingindo 0.5248 na primeira epoch, mas esse valor reduziu consistentemente ao longo do treinamento, chegando a 0.4472 na última epoch. O mesmo padrão é observado nas métricas Mean Absolute Error (MAE) e Root Mean Squared Error (RMSE), onde valores mais altos no início diminuíram gradualmente.

A análise dessas métricas sugere que o modelo está aprendendo com o conjunto de dados de treinamento, ajustando-se aos padrões e melhorando suas previsões ao longo das epochs. É importante observar que a diminuição dessas métricas indica um processo eficaz de treinamento, onde o modelo se torna mais preciso em suas previsões. Contudo, para avaliar a eficácia do modelo, é crucial validar seu desempenho em um conjunto de dados de teste independente.

Esses resultados também destacam a importância do monitoramento contínuo do treinamento do modelo, ajustando hiperparâmetros ou refinando a arquitetura para otimizar ainda mais o desempenho. Em última análise, a avaliação do modelo em

cenários do mundo real será crucial para determinar sua utilidade prática e capacidade de generalização além dos dados de treinamento.

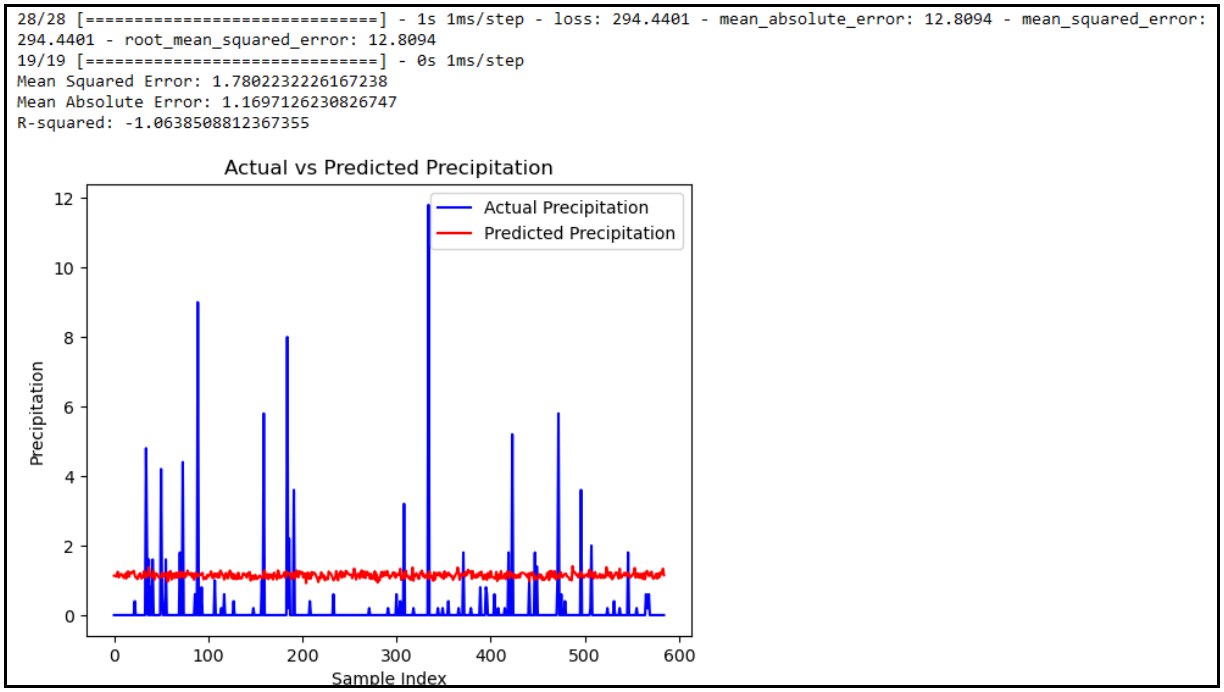

Figura 89. Fonte Autoria Própria

A métrica Loss, que é uma combinação das métricas de erro, atingiu um valor elevado de 294.4401. No entanto, as métricas individuais, como o Mean Squared Error (MSE) de 1.7802 e o Mean Absolute Error (MAE) de 1.1697, indicam que o modelo está conseguindo fazer previsões com um nível razoável de precisão em relação aos dados de treinamento.

Surpreendentemente, o R-squared apresenta um valor negativo de -1.0639, o que pode indicar que o modelo não está se ajustando bem aos dados ou que há uma falta de linearidade nas relações capturadas. Isso sugere a necessidade de uma investigação mais profunda, seja refinando a arquitetura do modelo, ajustando hiperparâmetros ou explorando características específicas do conjunto de dados.

O fato de o R-squared ser negativo levanta questões sobre a capacidade do modelo de superar um ajuste linear simples, indicando que pode haver complexidades não capturadas ou informações relevantes ausentes no processo de treinamento.

O uso de modelos de aprendizado de máquina, como árvores de decisão, para abordar questões relacionadas aos padrões climáticos em Salvador, se revelou uma estratégia promissora. Ao empregar esses modelos tanto para classificação quanto para previsão, foi possível obter resultados que, embora razoáveis, apresentaram nuances significativas. No contexto da classificação, a capacidade do modelo de árvore de decisão em identificar padrões climáticos específicos, como eventos de chuva intensa, oferece uma ferramenta valiosa para a gestão de riscos. No entanto, é crucial reconhecer que o desempenho razoável indica a necessidade contínua de otimização e refinamento do modelo para aumentar sua precisão na identificação desses padrões.

No que diz respeito ao modelo de previsão, embora tenha alcançado resultados razoáveis em termos de acertos e erros, é vital compreender as limitações intrínsecas à previsão de eventos climáticos extremos. A imprevisibilidade inerente a fenômenos meteorológicos, especialmente diante das mudanças climáticas, impõe desafios significativos. O período analisado de 2019 a 2023 revela a dinâmica temporal dos eventos de chuva intensa, mas a sazonalidade e a complexidade do clima demandam uma abordagem contínua de aprimoramento do modelo. Assim, é imperativo reconhecer os resultados como um ponto de partida, sinalizando a importância de ajustes constantes e refinamentos para aprimorar a capacidade de previsão ao longo do tempo.

Em relação aos objetivos propostos, a análise exploratória e modelagem preditiva são ferramentas cruciais na extração de insights das séries temporais climáticas. O enfoque na previsão de temporais contribui diretamente para a eficiência na aplicação de políticas públicas, permitindo uma resposta mais ágil dos órgãos responsáveis diante desses eventos. Os objetivos específicos delineados, desde a análise dos padrões históricos até a avaliação da eficácia dos modelos de aprendizado de máquina, refletem um compromisso abrangente em entender e antecipar as mudanças climáticas em Salvador. O objetivo geral de identificar oportunidades de melhoria na infraestrutura da cidade e promover ações preventivas ganha destaque,

ressaltando a importância do uso inovador da tecnologia para enfrentar desafios climáticos e fortalecer a resiliência urbana.

7. Links

Aqui você deve disponibilizar os links para o vídeo com sua apresentação de 5 minutos e para o repositório contendo os dados utilizados no projeto, scripts criados, etc.

Link para o vídeo: https://youtu.be/8YgDsXmAUzw

Link para o repositório: https://github.com/RodrigoAB93/TCC_PUC_MG

REFERÊNCIAS

Cavalcanti, I.A, 1982: Um estudo sobre as interações entre os sistemas de circulação de escala sinótica e circulações locais.

Defesa Civil de Salvador (CODESAL). **Relatório final da Operação Chuva 2019-2023.** (acesso em: 11 de janeiro de 2024 no site: http://www.defesacivil.salvador.ba.br/).

DIAS, F. P.; HERRMANN, M. L. P. **Susceptibilidade a deslizamentos: Estudo de caso no bairro Saco Grande, Florianópolis - SC.** Caminhos de Geografia – Revista On Line.Programa de Pós- graduação em Geografia, 2002

HASTENRATH, S.; HELLER, L. Dynamics of climatic hazards in north-east Brazil. **Quarterly Journal of the Royal Meteorological Society**, v. 103, n. 435, p. 77-92, 1977.

Kousky , V.E, 1980: **Diurnal rainfall variation in Northeast Brazil**. Mon. Weather Rev, 108, 488-498.

Molion, L. C. B.; Bernardo, S. O. **Uma revisão das chuvas no Nordeste brasileiro**, Revista Brasileira de Meteorologia, v.17, n.1, p.1-10, 2002.

ROPELEWSKI, C. F.; HALPERT, M. S. **Global and Regional scale precipitation patterns associated with El Niño/Southern Oscillation**. Monthly Weather Review, v. 115, n. 8, p. 1606-1626, 1987.

SANTOS, A. H. M. **Eventos Extremos de Chuva em Salvador, Bahia: Condições Atmosféricas e Impactos Ambientais**. 2008. 65 f. Dissertação (Mestrado em Meteorologia) -Universidade Federal de Campina Grande, Campina Grande, 2008

APÊNDICE

Programação/Scripts

API.ipynb

```
#Instalando bibliotecas ausentes
!pip install openmeteo_requests
!pip install requests_cache
!pip install retry_requests

#  Importando bibliotecas necessárias
import openmeteo_requests

import requests_cache
import pandas as pd
from retry_requests import retry



cache_session = requests_cache.CachedSession('.cache', ex-
pire_after = 3600)
retry_session = retry(cache_session, retries = 5,
backoff_factor = 0.2)
openmeteo = openmeteo_requests.Client(session = retry_session)

# Make sure all required weather variables are listed here
# The order of variables in hourly or daily is important to
assign them correctly below
url = "https://climate-api.open-
meteo.com/v1/climate?latitude=-12.9711&longitude=-
38.5108&start_date=2019-01-01&end_date=2023-12-
31&models=CMCC_CM2_VHR4,FGOALS_f3_H,HiRAM_SIT_HR,MRI_AGCM3_2_S
,EC_Earth3P_HR,MPI_ESM1_2_XR,NICAM16_8S&timezone=America%2FSao
_Paulo&daily=temperature_2m_max,temperature_2m_min,relative_hu
midi-
ty_2m_mean,relative_humidity_2m_max,relative_humidity_2m_min,p
recipitation_sum,rain_sum"
params = {
  "latitude": -12.9711,
```

```
  "longitude": -38.5108,
  "start_date": "2020-01-01",
  "end_date": "2023-12-31",
  "models": ["CMCC_CM2_VHR4", "FGOALS_f3_H", "HiRAM_SIT_HR",
"MRI_AGCM3_2_S", "EC_Earth3P_HR", "MPI_ESM1_2_XR",
"NICAM16_8S"],
  "timezone": "America/Sao_Paulo",
  "daily": ["temperature_2m_mean", "temperature_2m_max", "tem-
perature_2m_min", "wind_speed_10m_mean", "wind_speed_10m_max",
"cloud_cover_mean", "shortwave_radiation_sum", "rela-
tive_humidity_2m_mean", "relative_humidity_2m_max", "rela-
tive_humidity_2m_min", "precipitation_sum", "rain_sum"]
}
responses = openmeteo.weather_api(url, params=params)


response = responses[0]
print(f"Coordinates {response.Latitude()}°E {respon-
se.Longitude()}°N")
print(f"Elevation {response.Elevation()} m asl")
print(f"Timezone {response.Timezone()} {re-
sponse.TimezoneAbbreviation()}")
print(f"Timezone difference to GMT+0 {re-
sponse.UtcOffsetSeconds()} s")

# Process daily data. The order of variables needs to be the
same as requested.
daily = response.Daily()
daily_temperature_2m_mean = daily.Variables(0).ValuesAsNumpy()
daily_temperature_2m_max = daily.Variables(1).ValuesAsNumpy()
daily_temperature_2m_min = daily.Variables(2).ValuesAsNumpy()
daily_wind_speed_10m_mean = daily.Variables(3).ValuesAsNumpy()
daily_wind_speed_10m_max = daily.Variables(4).ValuesAsNumpy()
daily_cloud_cover_mean = daily.Variables(5).ValuesAsNumpy()
daily_shortwave_radiation_sum = dai-
ly.Variables(6).ValuesAsNumpy()
daily_relative_humidity_2m_mean = dai-
ly.Variables(7).ValuesAsNumpy()
daily_relative_humidity_2m_max = dai-
ly.Variables(8).ValuesAsNumpy()
```

```
daily_relative_humidity_2m_min = dai-
ly.Variables(9).ValuesAsNumpy()
daily_precipitation_sum = daily.Variables(10).ValuesAsNumpy()
daily_rain_sum = daily.Variables(11).ValuesAsNumpy()

daily_data = {"date": pd.date_range(
  start = pd.to_datetime(daily.Time(), unit = "s"),
  end = pd.to_datetime(daily.TimeEnd(), unit = "s"),
  freq = pd.Timedelta(seconds = daily.Interval()),
  inclusive = "left"
)}
daily_data["temperature_2m_mean"] = daily_temperature_2m_mean
daily_data["temperature_2m_max"] = daily_temperature_2m_max
daily_data["temperature_2m_min"] = daily_temperature_2m_min
daily_data["wind_speed_10m_mean"] = daily_wind_speed_10m_mean
daily_data["wind_speed_10m_max"] = daily_wind_speed_10m_max
daily_data["cloud_cover_mean"] = daily_cloud_cover_mean
daily_data["shortwave_radiation_sum"] = dai-
ly_shortwave_radiation_sum
daily_data["relative_humidity_2m_mean"] = dai-
ly_relative_humidity_2m_mean
daily_data["relative_humidity_2m_max"] = dai-
ly_relative_humidity_2m_max
daily_data["relative_humidity_2m_min"] = dai-
ly_relative_humidity_2m_min
daily_data["precipitation_sum"] = daily_precipitation_sum
daily_data["rain_sum"] = daily_rain_sum

daily_dataframe = pd.DataFrame(data = daily_data)
print(daily_dataframe) daily_data)
print(daily_dataframe)

# Caminho do arquivo CSV
caminho_arquivo_csv = 'dataset_api.csv'

# Salvando o DataFrame como um arquivo CSV
daily_dataframe.to_csv(caminho_arquivo_csv, index=False)
```

Analise_Mudança_Climatica.ipynb

```
#Importando bibliotecas necessárias
import os
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import pkg_resources

# Listando bibliotecas e suas versões
bibliotecas = ['numpy', 'pandas', 'matplotlib', 'seaborn',
'scikit-learn','pkg_resources',
               'tensor-
flow','math','datetime','stats','sklearn','sys','statsmodels',
'os',
               'csv','imblearn']

# Dicionário para armazenar as versões
versoes = {}

for biblioteca in bibliotecas:
    try:
        versao =
pkg_resources.get_distribution(biblioteca).version
        versoes[biblioteca] = versao
    except pkg_resources.DistributionNotFound:
        versoes[biblioteca] = 'Não encontrada'

# Exibir as versões
for biblioteca, versao in versoes.items():
    print(f'{biblioteca}: {versao}')


# Diretório com os 4 dataset do INMET
diretorio = './Dados/INMET/Anos/'
# Diretório com dataset da API Open Meteo:
diretorio_api = './Dados/API/dataset_api.csv'
```

```
## Tratamento dos Dados



# Criação de um dataframe vazio para armazenar dados dos da-
tasets
dataframes = []

# Loop por todos arquivos .csv na pasta
for arquivo in os.listdir(diretorio):
    if arquivo.endswith('.csv'):
        # Cria o caminho completo do arquivo
        caminho_completo = os.path.join(diretorio, arquivo)

        # Lê o arquivo CSV e adiciona o DataFrame à lista
        data = pd.read_csv(caminho_completo)
        dataframes.append(data)

for i in range(1, len(dataframes)):
    dataframes[i] = datafra-
mes[i].reindex(columns=dataframes[0].columns, fill_value=None)

# Concatena todos os DataFrames em um único DataFrame
dados = pd.concat(dataframes, ignore_index=True)

# Cria um arquivo .csv com o DataFrame gerado
dados.to_csv('dataset_INMET-Unico.csv', index=False)
dados



#Coletando do dataset original somente a Estação de Salvador
(A401)
dados = dados[dados['ESTACAO']=='A401']
dados.to_csv('dataset_INMET-Salvador.csv', index=False)
dados
```

```
# Coletando Estatisticas dos dados do INMET
dados.describe().transpose()




# Lendo o arquivo dados_api.csv proveniente da consulta na API
Open Meteo
# Lê o arquivo CSV e adiciona o DataFrame à lista
dados_api = pd.read_csv(diretorio_api)
# Exibe o DataFrame resultante
dados_api




#Quantidade de registros 0.

print("-" * 120)
print("=" * 120)
print(" " * 50 + "DADOS INMET")
print("=" * 120)
display(dados.isnull().sum())

print("-" * 120)
print("=" * 120)
print(" " * 50 + "DADOS OPEN-METEO")
print("=" * 120)

display(dados_api.isnull().sum())

#Substitui onde tiver NaN por 0
```

```
dados = data.fillna(0)
dados_api = dados_api.fillna(0)
dados
```

```
#Criando um novo dataset com dados somente até 2022

# Convertendo a coluna 'DATA (YYYY-MM-DD)' para o tipo de dado
datetime
dados_api['date'] = pd.to_datetime(dados_api['date'])

# Somente registros até 2022
dados_api_2022 = dados_api['date'].dt.year <= 2022

dados_ate_2022 = dados_api.loc[dados_api_2022]

#Cria um novo arquivo csv com os dados
dados_api=dados_api_2022.to_csv('dados_api.csv', index=False)
```

```
dados = pd.DataFrame(dados)
# Tipo de cada coluna

tipos_colunas = dados.dtypes

# Exibindo os tipos de cada coluna

print("-" * 120)
print("=" * 120)
print(" " * 50 + "DADOS INMET")
print("=" * 120)
print("-" * 120)
print(tipos_colunas)
```

```
dados_api = pd.read_csv('./Dados/API/dados_api_ate_2022.csv')

tipos_colunas_api = dados_api.dtypes

print("-" * 120)
print("=" * 120)
print(" " * 50 + "DADOS OPEN-METEO")
print("=" * 120)
print("-" * 120)
print(tipos_colunas_api)

#Eliminando colunas desnecessarias
dados_api = dados_api.drop(columns=[
 'cloud_cover_mean','shortwave_radiation_sum'])
dados = dados.drop(columns=['PRESSÃO ATMOSFERICA MAX.NA HORA
ANT. (AUT) (mB)','PRESSÃO ATMOSFERICA MIN. NA HORA ANT. (AUT)
(mB)'
                            ,'TEMPERATURA MÁXIMA NA HORA ANT.
(AUT) (°C)','TEMPERATURA MÍNIMA NA HORA ANT. (AUT) (°C)',
                            'UMIDADE REL. MAX. NA HORA ANT.
(AUT) (%)','UMIDADE REL. MIN. NA HORA ANT. (AUT) (%)'])




#Primeiros 10 exemplos do dataset de amostra
dados_api.head(10)

#Estatisticas sobre o dataset INMET
dados.describe().transpose()



#Estatisticas sobre o dataset INMET
dados_api.describe().transpose()


print(dados.shape)
print(dados_api.shape)
```

```
corr = dados.iloc[:,0:].corr()
corr



corr = dados_api.iloc[:,0:].corr()
corr



corr = dados.corr()
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 150, n=200),
    square=False
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=90,
    horizontalalignment='left'
);



corr = dados_api.corr()
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 150, n=200),
    square=False
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=90,
    horizontalalignment='left'
);

dados = dados.sample(1000)
```

```
#Criando gráfico relacionando Velocidade do vento e precipita-
ção total
plt.bar(dados['TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)'],
dados['PRECIPITAÇÃO TOTAL, HORÁRIO (mm)'])
plt.xlabel('TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)')
plt.ylabel('PRECIPITAÇÃO TOTAL, HORÁRIO (mm)')
plt.title('Gráfico de barras da precipitação total horária em
função da Temperatura do ar')
plt.show()

#Criando gráfico relacionando Velocidade do vento e precipita-
ção total
plt.bar(dados_api['temperature_2m_mean'], da-
dos_api['precipitation_sum'])
plt.xlabel('temperature_2m_mean')
plt.ylabel('precipitation_sum')
plt.title('Gráfico de barras da precipitação total horária em
função da Temperatura do ar')
plt.show()

# Certifique-se de que a coluna 'date' está no formato dateti-
me
dados_api['date'] = pd.to_datetime(dados_api['date'])

# Extração do ano e criação de uma nova coluna 'year'
dados_api['year'] = dados_api['date'].dt.year

# Agrupamento por ano e soma da precipitação
precipitacao_por_ano = da-
dos_api.groupby('year')['precipitation_sum'].sum().reset_index
()

# Criando o gráfico de barras
plt.figure(figsize=(7, 3))
plt.bar(precipitacao_por_ano['year'], precipita-
cao_por_ano['precipitation_sum'], color='blue')
plt.xlabel('Ano')
```

```
plt.ylabel('Precipitação Total')
plt.title('Precipitação por Ano')
plt.show()



#Verificando as estações com base no arquivo Estacoes.csv
estacoes = pd.read_csv('./Dados/Estacoes.csv')

estacoes




#estacoes = estacoes[estacoes['region']=='NE']
estacoes = estacoes[estacoes['city_station']=='SALVADOR']
```

Modelo_Previsão_Mudança_Climatica.ipynb

```
# Importando bibliotecas
import tensorflow as tf
import pandas as pd
import math
import datetime
from scipy import stats
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score
from tensorflow import keras
from tensorflow.python.keras import models
from tensorflow.python.keras import layers
```

```
from tensorflow.python.keras import backend as Kb
from tensorflow.python.data import Dataset
# Helper Libraries
import numpy as np
import matplotlib.pyplot as plt
import sys
import statsmodels as sm
from sklearn.metrics import classification_report
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
import pkg_resources

import os
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import confusion_matrix, ConfusionMa-
trixDisplay
from matplotlib import pyplot as plt
import csv
from sklearn import preprocessing
from sklearn.metrics import mean_squared_error,
mean_absolute_error,  r2_score




# Diretório com os 4 dataset do INMET agrupados em um unico
dataset
diretorio = './Dados/INMET/Unificados/dataset_INMET-
Salvador.csv'
diretorio_api = './Dados/API/dados_api_ate_2022.csv'




# Lista de bibliotecas que você quer obter as versões
bibliotecas = ['numpy', 'pandas', 'matplotlib', 'seaborn',
'scikit-learn','pkg_resources',
```

```
                'tensorflow','sympy','python-
dateutil','stats','sklearn','sys','statsmodels','os',
                'csv','imblearn']

# Dicionário para armazenar as versões
versoes = {}

for biblioteca in bibliotecas:
    try:
        versao =
pkg_resources.get_distribution(biblioteca).version
        versoes[biblioteca] = versao
    except pkg_resources.DistributionNotFound:
        versoes[biblioteca] = 'Não encontrada'

# Exibir as versões
for biblioteca, versao in versoes.items():
    print(f'{biblioteca}: {versao}')



# Leitura do dataset
dados = pd.read_csv(diretorio)
dados_api = pd.read_csv(diretorio_api)


# Substituição de  registros NaN por 0
dados = dados.fillna(0)
dados =dados.sample(1461)
dados_api = dados_api.fillna(0)


#Transforma o tipo do campo data
dados['DATA (YYYY-MM-DD)'] = pd.to_datetime(dados['DATA (YYYY-
MM-DD)'])
# Substituir 'A401' por 'Salvador' na coluna 'ESTACAO'
dados['ESTACAO'] = dados['ESTACAO'].replace({'A401': 'Salva-
dor'})
```

```
## Previsão com dados do INMET



# Selecionar características relevantes
features = dados[[ 'TEMPERATURA DO AR - BULBO SECO, HORARIA
(°C)', 'UMIDADE RELATIVA DO AR, HORARIA (%)', 'VENTO, VELOCI-
DADE HORARIA (m/s)']]
#Define a variavel a ser prevista
targets = dados['PRECIPITAÇÃO TOTAL, HORÁRIO (mm)']




 # Dividir em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(features,
targets, test_size=0.4, random_state=42)




# Normalização dos dados

scaler = MinMaxScaler()

train_normalized = (X_train - X_train.mean())/X_train.std()
test_normalized = (X_test - X_test.mean())/X_test.std()

# Removendo a primeira linha a cada 1 hora (resultados)
train_result = train_normalized

# Removendo a última linha (não há resultados para a última
linha)
train_normalized = train_normalized

# Removendo a primeira linha a cada 1 hora (resultados)
test_result = test_normalized

# Removendo a última linha (não há resultados para a última
linha)
test_normalized = test_normalized
```

```
# Verificar os dados normalizados
print("Dados de treino normalizados:")
print(train_normalized[:-1])
print("\nDados de teste normalizados:")
print(test_normalized[:-1])


df_test = pd.DataFrame(test_normalized)
print(df_test)


df_train = pd.DataFrame(train_normalized)
print(df_train)









modelo = LinearRegression()
modelo.fit(X_train, y_train)

y_pred = modelo.predict(X_test)

#
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean squared error:', mse)
print('Mean absolute error:', mae)
print('R-squared:', r2)
```

```
    Mean squared error: 0.7813943273685928
    Mean absolute error: 0.3529619602455774
    R-squared: 0.0941128333543475



#Função que realiza calculo da metrica MAE
def root_mean_squared_error(x, y):
        return Kb.sqrt(Kb.mean(Kb.square(x - y), axis=-1))



#Treinando Epoch
regularizer=tf.compat.v1.keras.regularizers.l2(0.001)
model = keras.Sequential([
layers.Dense(20, activation='relu', input_shape = (8,), ker-
nel_regularizer=regularizer),
layers.Dense(12, activation='relu', ker-
nel_regularizer=regularizer),
layers.Dense(4, activation='relu', ker-
nel_regularizer=regularizer),
layers.Dense(1)
])

model.compile(loss='mean_squared_error', optimiz-
er='adam',#Utilizando metrica MAE e MSE
              metrics=['mean_absolute_error',
'mean_squared_error', root_mean_squared_error])

model.fit(train_normalized, train_result,
  epochs=100, batch_size=1,validation_split = 0.0, verbose=1)




# Treinando o modelo
```

```
model.fit(X_train, y_train)

# Fazendo previsões no conjunto de teste
y_pred = model.predict(X_test)

# Avaliando o desempenho do modelo
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean Squared Error:', mse)
print('Mean Absolute Error:', mae)
print('R-squared:', r2)

# Visualizando as previsões em relação aos valores reais
plt.plot(y_test.values, label='Actual Precipitation', col-
or='blue')
plt.plot(y_pred, label='Predicted Precipitation', color='red')
plt.xlabel('Sample Index')
plt.ylabel('Precipitation')
plt.title('Actual vs Predicted Precipitation')
plt.legend()
plt.show()

# Realizando previsões utilizando sklearn
predictions = model.predict(X_test)
df1 = pd.DataFrame({"Pred": predictions.flatten(), "Real":
y_test.values.flatten()})

# Visualizando as previsões em um DataFrame
print(df1)



# Realizando previsões utilizando sklearn
predictions = model.predict(X_test)
df1 = pd.DataFrame(df1)

# Visualizando as previsões em um DataFrame
print(df1)
```

```
## Previsão com dados do INMET e Open Meteo



# Selecionar características relevantes
features = dados_api[[ 'temperature_2m_mean', 'relati-
ve_humidity_2m_max', 'wind_speed_10m_mean']]
#Define a variavel a ser prevista
targets = dados['PRECIPITAÇÃO TOTAL, HORÁRIO (mm)']
features









 # Dividir em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(features,
targets, test_size=0.4, random_state=42)




# Normalização dos dados

scaler = MinMaxScaler()

train_normalized = (X_train - X_train.mean())/X_train.std()
test_normalized = (X_test - X_test.mean())/X_test.std()

# Removendo a primeira linha a cada 1 hora (resultados)
train_result = train_normalized
```

```
# Removendo a última linha (não há resultados para a última linha)
train_normalized = train_normalized

# Removendo a primeira linha a cada 1 hora (resultados)
test_result = test_normalized

# Removendo a última linha (não há resultados para a última linha)
test_normalized = test_normalized

# Verificar os dados normalizados
print("Dados de treino normalizados:")
print(train_normalized[:-1])
print("\nDados de teste normalizados:")
print(test_normalized[:-1])




df_test = pd.DataFrame(test_normalized)
print(df_test)




df_train = pd.DataFrame(train_normalized)
# Remove the comment below to see the table
print(df_train)




modelo = LinearRegression()
modelo.fit(X_train, y_train)
```

```
y_pred = modelo.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean squared error:', mse)
print('Mean absolute error:', mae)
print('R-squared:', r2)


#Função que realiza calculo da metrica MAE
def root_mean_squared_error(x, y):
        return Kb.sqrt(Kb.mean(Kb.square(x - y), axis=-1))



#Treinando Epoch
regularizer=tf.compat.v1.keras.regularizers.l2(0.001)
model = keras.Sequential([
layers.Dense(20, activation='ELU', input_shape = (8,), ker-
nel_regularizer=regularizer),
layers.Dense(12, activation='ELU', ker-
nel_regularizer=regularizer),
layers.Dense(4, activation='ELU', ker-
nel_regularizer=regularizer),
layers.Dense(1)
])

model.compile(loss='mean_squared_error', optimiz-
er='adam',#Utilizando metrica MAE e MSE
              metrics=['mean_absolute_error',
'mean_squared_error', root_mean_squared_error])
```

```
model.fit(train_normalized, train_result,
  epochs=100, batch_size=1,validation_split = 0.0, verbose=1)

# Treinando o modelo
model.fit(X_train, y_train)

# Fazendo previsões no conjunto de teste
y_pred = model.predict(X_test)

# Avaliando o desempenho do modelo
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean Squared Error:', mse)
print('Mean Absolute Error:', mae)
print('R-squared:', r2)

# Visualizando as previsões em relação aos valores reais
plt.plot(y_test.values, label='Actual Precipitation', col-
or='blue')
plt.plot(y_pred, label='Predicted Precipitation', color='red')
plt.xlabel('Sample Index')
plt.ylabel('Precipitation')
plt.title('Actual vs Predicted Precipitation')
plt.legend()
plt.show()

# Realizando previsões utilizando sklearn
predictions = model.predict(X_test)
df1 = pd.DataFrame({"Pred": predictions.flatten(), "Real":
y_test.values.flatten()})

# Visualizando as previsões em um DataFrame
print(df1)



# Realizando previsões utilizando sklearn
predictions = model.predict(X_test)
```

```
df1 = pd.DataFrame(df1)

# Visualizando as previsões em um DataFrame
print(df1)
```

Modelo_Classificação_Mudança_Climatica.ipynb

```
  `
#Importando bibliotecas necessárias
import numpy as np
import pandas as pd
import sklearn
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import confusion_matrix, ConfusionMa-
trixDisplay
from matplotlib import pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE
import pkg_resources



# Diretório com os 4 dataset do INMET agrupados em um unico
dataset
diretorio = './Dados/INMET/Unificados/dataset_INMET-
Salvador.csv'
```

```
diretorio_api = './Dados/API/dados_api_ate_2022.csv'



# Leitura do dataset
dados = pd.read_csv(diretorio)
dados_api = pd.read_csv(diretorio_api)



# Retirando uma amostra do dataset
dados = dados.sample(1461)


## Classificação com dados INMET e Meteo Open



#Sintonizando hiperparâmetros para o  modelo usando validação
cruzada
def tuner(clf, dist, X, y):
  rs_clf = RandomizedSearchCV(estimator=clf, random_state=1,
n_jobs=-1, param_distributions=dist, cv=10)

  search = rs_clf.fit(X=X, y=y)

  return search.best_params_



#Função para gerar e exibir uma matriz de confusão para avali-
ar o desempenho do modelo
def plot_confusion_matrix(y_true, y_pred, labels):
  cm = confusion_matrix(y_true=y_true, y_pred=y_pred, la-
bels=labels)

  disp = ConfusionMatrixDisplay(confusion_matrix=cm, dis-
play_labels=labels)
```

```
  disp.plot(cmap=plt.cm.Blues)

  plt.show()



#Função para gerar relatório para avaliação
def reporter(clf, X, y, labels):

  predictions = clf.predict(X)

  plot_confusion_matrix(y_true=y, y_pred=predictions, la-
bels=labels)

  print(classification_report(y_true=y, y_pred=predictions))




# Transformando dados numericos da coluna precipitação em la-
bels de acordo com intervalos
def categorize_precipitation(value):
    if value == 0:
        return 'Não choveu'
    elif 0 < value <= 10:
        return 'Choveu pouco'
    elif 10 < value <= 30:
        return 'Choveu moderado'
    elif 30 < value <= 100:
        return 'Chuva intensa'
    else:
        return 'Valor fora dos intervalos especificados'

dados['CATEGORIA_PRECIPITACAO'] = dados['PRECIPITAÇÃO TOTAL,
HORÁRIO (mm)'].apply(categorize_precipitation)
```

```
print(dados['CATEGORIA_PRECIPITACAO'].value_counts())

print(dados.head())

contagem_categorias = da-
dos['CATEGORIA_PRECIPITACAO'].value_counts()
contagem_categorias



# Transformando dados numericos da coluna precipitação em la-
bels de acordo com intervalos
def categorize_precipitation(value):
    if value == 0:
        return 'Não choveu'
    elif 0 < value <= 10:
        return 'Choveu pouco'
    elif 10 < value <= 30:
        return 'Choveu moderado'
    elif 30 < value <= 100:
        return 'Chuva intensa'
    else:
        return 'Valor fora dos intervalos especificados'

dados_api['CATEGORIA_PRECIPITACAO'] = da-
dos_api['precipitation_sum'].apply(categorize_precipitation)


print(dados_api['precipitation_sum'].value_counts())

print(dados_api.head())


contagem_categorias = da-
dos_api['CATEGORIA_PRECIPITACAO'].value_counts()
contagem_categorias



# Selecionar características relevantes
```

```
features = dados_api[[ 'temperature_2m_mean', 'relati-
ve_humidity_2m_max', 'wind_speed_10m_mean']]
#Define a variavel a ser prevista
targets = dados['CATEGORIA_PRECIPITACAO']



X_train, X_test_valid, y_train, y_test_valid =
train_test_split(features, targets,test_size=0.3)



X_test, X_valid, y_test, y_valid =
train_test_split(X_test_valid, y_test_valid,test_size=0.7)


max_depths = [i for i in range(1, 30, 2)]


scaler = MinMaxScaler()

X_train_normalized = scaler.fit_transform(X_train)
X_valid_normalized = scaler.transform(X_valid)
X_test_normalized = scaler.transform(X_test)



dist = dict(n_neighbors=[i for i in range(1, 100, 2)])
n_samples = X_test.shape[0]
knc = KNeighborsClassifier(n_jobs=-1)

best = tuner(knc, dist, X_valid_normalized, y_valid)

print(f'Melhores parâmetros: {best}')

knc = KNeighborsClassifier(n_jobs=-1,
n_neighbors=best['n_neighbors'])

knc.fit(X_train_normalized, y_train)
```

```
labels = np.unique(y_train)

print('Train')
reporter(knc, X_train_normalized, y_train, labels)

print('Cross Validation')
reporter(knc, X_valid_normalized, y_valid, labels)

print('Test')
reporter(knc, X_test_normalized, y_test, labels)



dist = dict(max_depth=max_depths)

dtc = DecisionTreeClassifier(random_state=1)

best = tuner(dtc, dist, X_valid, y_valid)

print(f'Melhores parâmetros: {best}')

dtc = DecisionTreeClassifier(random_state=1,
max_depth=best['max_depth'])
result = dtc.fit(X_train, y_train)




# Aplicando SMOTE para balancear as classes
smote = SMOTE(sampling_strategy='auto', k_neighbors=3, ran-
dom_state=42)
X_train_resampled, y_train_resampled =
smote.fit_resample(X_train, y_train)


# Treinando um modelo (por exemplo, RandomForest)

clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_resampled, y_train_resampled)
```

```
# Fazendo previsões
y_pred = clf.predict(X_test)

# Avaliando o desempenho
print(classification_report(y_test, y_pred))



dist = dict(max_depth=max_depths)

rfc = RandomForestClassifier(n_jobs=-1, random_state=1)

best = tuner(rfc, dist, X_valid, y_valid)

print(f'Melhores parâmetros: {best}')

rfc = RandomForestClassifier(n_jobs=-1, random_state=1,
max_depth=best['max_depth'])

rfc.fit(X_train, y_train)

print('Train')
reporter(rfc, X_train_resampled, y_train_resampled, labels)
print('Cross Validation')
reporter(rfc, X_train_resampled, y_train_resampled, labels)
print('Test')
reporter(rfc, X_test, y_test, labels)



# Selecionar características relevantes
features = dados[[ 'TEMPERATURA DO AR - BULBO SECO, HORARIA
(°C)', 'UMIDADE RELATIVA DO AR, HORARIA (%)', 'VENTO, VELOCI-
DADE HORARIA (m/s)']]
#Define a variavel a ser prevista
targets = dados_api['CATEGORIA_PRECIPITACAO']


X_train, X_test_valid, y_train, y_test_valid =
train_test_split(features, targets,test_size=0.3)
```

```
X_test, X_valid, y_test, y_valid =
train_test_split(X_test_valid, y_test_valid,test_size=0.7)


max_depths = [i for i in range(1, 30, 2)]


scaler = MinMaxScaler()

X_train_normalized = scaler.fit_transform(X_train)
X_valid_normalized = scaler.transform(X_valid)
X_test_normalized = scaler.transform(X_test)



dist = dict(n_neighbors=[i for i in range(1, 100, 2)])
n_samples = X_test.shape[0]
knc = KNeighborsClassifier(n_jobs=-1)

best = tuner(knc, dist, X_valid_normalized, y_valid)

print(f'Melhores parâmetros: {best}')

knc = KNeighborsClassifier(n_jobs=-1,
n_neighbors=best['n_neighbors'])

knc.fit(X_train_normalized, y_train)

labels = np.unique(y_train)

print('Train')
reporter(knc, X_train_normalized, y_train, labels)

print('Cross Validation')
reporter(knc, X_valid_normalized, y_valid, labels)

print('Test')
reporter(knc, X_test_normalized, y_test, labels)
```