Escola Politécnica da USP

PTC3567 - Ciência dos Dados em Automação e Engenharia

Prof. Dr.Pedro Luiz Pizzigatti Correa

Estimador de presença de nuvem

Grupo 3

Mentora: Dra. Jeaneth Machicao

	NUSP:
Bruna Matsumoto Silva	11262146
Gustavo S. Torrico	10770429
Rodrigo Madruga Tavares	11234029

São Paulo

2023

Sumário

1 Introdução

No momento atual da Ciência de Dados, o acesso a dados geoespaciais de alta resolução tem se tornado crucial em uma variedade de contextos como monitoramento climático, agricultura, monitoramento ambiental, gestão de recursos naturais, segurança e defesa e pesquisa científica. Um componente fundamental dessas informações é a utilização de imagens de satélite, como as fornecidas pelo Sentinel 2[1], que oferecem uma visão abrangente e consistente da superfície terrestre.

Nesse contexto, a presença de nuvens nas imagens pode representar um desafio substancial durante a análise e interpretação desses dados, bem como para o uso deles no treinamento de modelos de aprendizado de máquina. Para contornar esse problema, muitos fornecedores de imagens de satélite fornecem informações sobre o nível de nuvem presente na imagem ou, ainda, em cada pixel. No entanto, nem todos os conjuntos de imagens de satélite possuem essa informação e os que possuem podem ter uma informação imprecisa.

Dessa forma, este projeto abordou essa complexidade, concentrando-se no planejamento e desenvolvimento de uma aplicação de Ciência de Dados voltada para a estimação da probabilidade de existência de nuvens em cada pixel de imagens de satélite. No projeto, treinou-se uma rede neural para realizar essa tarefa, a qual foi construída com base em referências do Github[2], e comparou-se as estimações dela com os dados fornecidos no dataset. Esse tipo de modelo contribui para aprimorar a qualidade do pré-processamento de dados em projetos de Ciência de Dados que envolvem informações geoespaciais.

A importância da estimação da cobertura de nuvens reside na necessidade de mitigar os efeitos adversos nas imagens de satélite, como distorções na interpretação geográfica, perda de informações e comprometimento da precisão dos algoritmos de análise. Este projeto contribui para aprimorar a confiabilidade e utilidade das análises em Ciência de Dados, permitindo avanços significativos no desenvolvimento de modelos preditivos que dependem da integridade dos dados geoespaciais. Ao compreender e mitigar a interferência das nuvens, o projeto representa um passo crucial na otimização do pré-processamento de imagens de satélite, maximizando a eficácia e precisão das análises subsequentes em projetos de Ciência de Dados. Esse pré-processamento se fez necessário, por exemplo, em trabalho realizado anteriormente na disciplina de Ciência de Dados, cujo objetivo foi realizar a extração

[1] `https://sentinel.esa.int/web/sentinel/missions/sentinel-2`

automática de corpos d'água ou de corpos vegetativos, no qual foi necessário realizar uma filtragem das imagens de satélite com base na quantidade de nuvens presentes nela [1].

2 Estratégias de Gestão dos Dados

No âmbito de um projeto de Ciência de Dados, a gestão dos dados desempenha um papel crucial para assegurar a transparência, reprodutibilidade e integridade dos resultados alcançados. A seguir, são listadas as estratégias adotadas para essa gestão neste projeto com o intuito de facilitar o compartilhamento e a reutilização futura.

1. **Canvas de Gestão de Dados:** Na etapa de planejamento, fizemos um Canvas (Seção 7) para fornecer uma representação visual das principais etapas e considerações relacionadas ao ciclo de vida dos dados no projeto. Este Canvas serviu como uma ferramenta ágil para mapear estratégias de coleta, processamento e armazenamento, permitindo uma compreensão rápida e abrangente do fluxo de trabalho que seria realizado.

2. **Data Management Plan (DMP):** Paralelamente, desenvolvemos um Data Management Plan (Seção 8) detalhado que complementou o Canvas, oferecendo uma visão mais aprofundada e documentada das práticas de gestão de dados adotadas. O DMP foi feito usando a ferramenta DMP Tool[2] e delineou de forma sistemática as diretrizes para a coleta, documentação, armazenamento e compartilhamento dos dados. Além disso, proporcionou um roteiro para a preservação a longo prazo, assegurando a sustentabilidade e acessibilidade dos dados gerados.

3. **DOI para Identificação do Dataset:** A fim de estabelecer uma identificação única e permanente para o conjunto de dados gerado, atribuímos um Digital Object Identifier (DOI). Este identificador exclusivo possibilita a referência inequívoca do dataset, contribuindo para a rastreabilidade e reconhecimento da fonte em futuras pesquisas e citações acadêmicas. O identificador gerado para o dataset por meio do Zenodo foi `10.5281/zenodo.10185776`.

4. **Fonte dos Dados - Google Earth Engine:** Os dados utilizados foram provenientes da Google Earth Engine (GEE). A escolha dessa fonte se deu pela sua capacidade de gerenciar e fornecer acesso eficiente a imagens de satélite, simplificando assim a obtenção e processamento das informações necessárias para o projeto. A GEE foi fundamental na agilidade e confiabilidade no acesso aos dados de satélite do Sentinel 2, bem como o acesso à documentação deles.

[2] `https://dmptool.org/`

5. **Armazenamento de Dados sem Serviço em Nuvem:** Embora muitas vezes a prática comum seja o armazenamento em serviços de nuvem, optamos por utilizar a API do Google Earth Engine diretamente para acessar os dados e já processá-los. Isso se mostrou eficiente, eliminando a necessidade de armazenamento adicional em nuvem e otimizando o fluxo de trabalho, uma vez que os dados eram acessados e processados dinamicamente.

6. **Repositório no GitHub para Código e Dados:** Para manter o compromisso com a transparência e reprodutibilidade, o código-fonte e os dados utilizados foram armazenados em um repositório público no GitHub. Essa abordagem não apenas facilita o compartilhamento com a comunidade científica, mas também permite que outros pesquisadores revisem, reproduzam e construam sobre o trabalho realizado. O repositório pode ser acessado em `https://github.com/rodtav/ptc3567-cloud_detector`.

Essas estratégias de gestão de dados buscam garantir a qualidade e confiabilidade do presente projeto, além de seguir um padrão para boas práticas em Ciência de Dados, fortalecendo a credibilidade e utilidade dos resultados alcançados.

3 Caracterização e Preparação dos Dados

O conjunto de dados COPERNICUS/S2_SR, disponível no Google Earth Engine, representa uma fonte de informações valiosa sobre a superfície terrestre. Ele é gerado a partir dos satélites Sentinel-2 do programa Copernicus da ESA e tem uma grande importância para monitorar mudanças no ambiente, bem como para apoiar diversos segmentos como agricultura, recursos hídricos e detecção de desastres naturais.

Esses satélites coletam imagens de alta resolução em várias faixas espectrais, o que possibilita a identificação de diferentes características da superfície como vegetação, água e áreas urbanas. Além disso, os dados são processados para corrigir efeitos atmosféricos e para detectar a presença de água, nuvens, vegetação, dentre outras coisas, o que os torna adequados para análises quantitativas.

Cada imagem do dataset possui um atributo "properties", com 76 metadados sobre a imagem. Além do atributo "properties", existe o atributo "bands", que possui a informação sobre as 23 bandas existentes na imagem. As bandas assumem um valor para cada pixel da imagem, porém cada banda tem sua resolução, ou seja, existem bandas com resolução de 10 metros, bandas com resolução de 20 metros e outras com resolução de 60 metros. Assim, a quantidade de pixels em cada banda de uma imagem varia de acordo com a resolução dela.

3.1 Caracterização dos Dados

Inicialmente, fizemos a classificação dos dados presentes nas propriedades da imagem. Como existem 76 colunas, foram selecionadas somente as mais importantes para o trabalho e que serão usadas. A classificação delas é mostrada na Tabela 1.

Propriedades	**Tipo de Dado**	**Descrição**
Coordinates	Quantitativo contínuo	Coordenadas geográficas da imagem
CLOUDY_PIXEL_PERCENTAGE	Quantitativo contínuo	Porcentagem de cobertura de nuvem da imagem
system:index	Qualitativo	Código de identificação único da imagem

Tabela 1: Classificação das propriedades das imagens

Em seguida, fizemos a classificação dos dados presentes nas bandas da imagem. Como existem 23 bandas, foram selecionadas somente as mais importantes para o trabalho e que serão usadas. A classificação delas é mostrada na Tabela 2.

Tipo de Banda	**Tipo de Dado**	**Descrição**
B2	Qualitativo	Código RGB após aplicação de filtro azul
B3	Qualitativo	Código RGB após aplicação de filtro verde
B4	Qualitativo	Código RGB após aplicação de filtro vermelho
B8	Qualitativo	Código RGB após aplicação de filtro NIR
MSK_CLDPRB	Quantitativo Contínuo	Probabilidade de existir nuvem no pixel
B9	Qualitativo	Código RGB após aplicação de filtro de vapor de água
SCL	Qualitativo	Classificação de cena do pixel

Tabela 2: Classificação das bandas dos pixels

As bandas B2, B3, B4, B9 são classificadas como dados qualitativos, pois o retorno dos dados representa uma cor RGB após filtragem em determinados comprimentos de onda.

A banda MSK_CLDPRB contém a probabilidade contínua de existir uma máscara de nuvem no pixel. No entanto, apesar de ser um dado contínuo, ele sofre discretização para que possa ser representado dentro dos 256 valores.

A banda SCL é classificada como dado qualitativo porque pode apresentar valores limitados ao conjunto de cores criadas na classificação da imagem, os quais são apresentados na tabela 3.

Cor (Hexadecimal)	**Descrição**
#ff0004	Saturado ou defeituoso
#868686	Pixels de área escura
#774b0a	Sombras de Nuvem
#10d22c	Vegetação
#ffff52	Solos nus
#0000ff	Água
#818181	Nuvens de baixa probabilidade / Sem classificação
#c0c0c0	Nuvens de média probabilidade
#f1f1f1	Nuvens de alta probabilidade
#bac5eb	Cirrus
#52fff9	Neve / Gelo

Tabela 3: Classificação SCL

3.2 Preparação dos Dados

Os dados foram coletados na região do Vale do Ribeira, localizado no Estado de São Paulo. Essa região apresenta características ambientais e climáticas bastante singulares, o que o torna uma área interessante para os testes do estimador de presença de nuvem. Dentre os tópicos que justificam a escolha da região para a extração dos dados, estão:

1. **Diversidade climática:** O Vale do Ribeira abriga uma diversidade climática significativa, com áreas de Mata Atlântica, serras, rios e uma variedade de microclimas. Essa diversidade cria um ambiente propício para a formação de diferentes tipos de nuvens, desde as mais baixas e densas até as mais altas e finas.

2. **Topografia variada:** A topografia variada da região, caracterizada por ser-

ras, vales e planícies, pode influenciar a formação e dispersão de nuvens. A interação entre o relevo e os padrões climáticos pode criar condições específicas que impactam a presença e o comportamento das nuvens.

3. **Ecossistemas Únicos:** A presença de ecossistemas únicos, como a Mata Atlântica, pode contribuir para padrões específicos de formação de nuvens. A vegetação densa e a umidade característica dessa região podem influenciar a dinâmica atmosférica, afetando a presença e a frequência de nuvens.

4. **Variação sazonal:** O Vale do Ribeira experimenta variações sazonais significativas, com períodos de chuva e seca. Essas mudanças sazonais podem impactar diretamente a formação de nuvens, sendo essencial para um estimador considerar tais variações ao prever a presença de nuvens ao longo do ano.

5. **Aplicações Locais:** A capacidade de estimar a presença de nuvens no Vale do Ribeira pode ter aplicações práticas locais, como previsão do tempo, gestão hídrica e agricultura. Compreender os padrões de nuvens na região é valioso para atividades que dependem das condições atmosféricas.

Definiu-se uma geometria poligonal que englobasse a maior parte da região do Vale do Ribeira, como pode ser vista na figura 1. As coordenadas dos vértices da geometria são (-47.013116, -24.254516), (-47.320733, -24.074090), (-48.378168, -24.739387) e (-48.021112, -24.958709). O período escolhido para coleta de imagens foi entre 13 de Novembro de 2023 a 17 de Novembro de 2023.

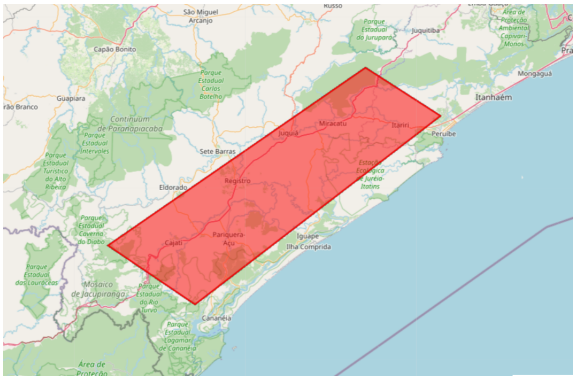


Figura 1: Geometria escolhida no Vale do Ribeira.

Uma vez filtrado os dados para região e período escolhidos, a coleção de imagens pode ser obtida utilizando o método ImageCollection, da biblioteca do Earth Engine, e as imagens podem ser acessadas diretamente pela API sem ser necessário fazer o download delas. Para conseguir utilizar os dados no modelo supervisionado, é necessário converter as imagens das bandas para o formato correto, e para isso foi desenvolvida a função ee_array_to_df(), que converte os arrays das imagens de duas dimensões para arrays de uma dimensão (mais detalhes podem ser vistos na Seção 9). A partir desses arrays, é construído um dataframe de entrada do modelo, em que os dados das bandas para cada pixel podem ser vistos na figura 2.

```
sattelite_images_data.head()
```

	MSK_CLDPRB	B2	B3	B4	B8	B9	SCL
0	0.00	0.020432	0.023072	0.019821	0.058747	0.060761	4
1	0.00	0.020340	0.022919	0.019516	0.059586	0.061509	4
6	0.01	0.028443	0.030808	0.027573	0.065583	0.065034	4
7	0.01	0.027604	0.029801	0.026535	0.063127	0.066300	4
12	0.03	0.030030	0.031815	0.029404	0.061738	0.065034	4

Figura 2: Primeiras 5 linhas do dataframe de dados.

4 Modelagem e Análise dos Dados

4.1 Pré-processamento

O pré-processamento dos dados consiste numa etapa crucial de preparação de dados realizada como parte do processo de desenvolvimento de um modelo de aprendizado de máquina para análise de imagens de satélite. O objetivo desta fase é garantir que os dados estejam estruturados e normalizados adequadamente para que o modelo possa aprender de maneira eficaz e fornecer previsões precisas.

Após definir como variável do modelo a banda *MSK_CLDPRB*, através do método *Holdout*, que consiste em dividir o conjunto total de dados em dois subconjuntos mutuamente exclusivos (um para treinamento e estimação dos parâmetros e outro para teste), o grupo submeteu 80% dos dados para treino e 20% dos dados para teste. Reconhecendo a importância da normalização dos dados, também foi aplicado o método de *StandardScaler* da biblioteca *sklearn.preprossing*. Esse processo padroniza as características, assegurando que todas tenham média zero e desvio padrão igual a 1. O conjunto de treinamento foi utilizado para calcular a média e o desvio padrão, e ambos os conjuntos de treinamento e teste foram transformados para esta nova escala.

4.2 Rede neural

4.3 Configuração da rede neural

A rede neural possui arquitetura sequencial e três camadas densas (conectadas) com ativações *relu* nas duas primeiras camadas e *sigmoid* na última. A função *relu* serve para introduzir não-linearidades, promovendo a capacidade da rede de aprender padrões complexos. É eficiente e frequentemente aplicada em camadas ocultas. Por outro lado, a função *sigmoid* transforma a saída em uma escala de probabilidade entre 0 e 1, sendo ideal para problemas de classificação binária ou problemas em que a interpretação probabilística é importante, como é o caso do estimador. Para assegurar a reprodutibilidade dos resultados, sementes são fixadas para a geração de números aleatórios. Adicionalmente, são feitas configurações para suprimir logs indesejados durante o processo de treinamento.

O treinamento é conduzido ao longo de 32 épocas, utilizando um tamanho de lote de 12. Uma parte do conjunto de treinamento (20%) é reservada para validação,

monitorando o desempenho do modelo em dados não vistos. Um mecanismo de parada antecipada é implementado para interromper o treinamento caso não haja melhorias significativas após um determinado número de épocas (*earlyStopping* de 5 épocas).

Uma vez treinado, o modelo é testado no conjunto de teste, e suas previsões são comparadas com os valores reais. Métricas de avaliação, como a Mean Squared Error (MSE), são utilizadas para quantificar o desempenho tanto no conjunto de treinamento quanto no conjunto de teste.

Para a função *loss*, que corresponde à função que será minimizada pelo modelo, foi utilizada a distribuição de *Poisson*, que utiliza (1) como equação.

$$\text{loss } = y_{pred} - y_{true} \cdot log(y_{pred}) \tag{1}$$

Além disso, buscando criar, também, um classificador binário, o grupo utilizou um *threshold* de 50%, de modo que previsões do modelo que tivessem as probabilidade superiores 50% fossem classificadas como positivas para nuvem e, caso contrário, negativas para nuvem. Assim, foi possível gerar o relatório de classificação com as métricas mais importantes e a matriz de confusão.

Os resultados gerados tanto para o estimador quanto para o classificador podem ser consultados na seção 5.

5 Apresentação e discussão dos resultados

Nas figuras 3 a 6, é possível observar que a diferença entre os dados de treino e validação foram pequenos de forma majoritária, não ultrapassando uma diferença de 0,008 no primeiro caso e 0,01 no segundo caso. Destaca-se, também, a não ocorrência de *overfitting*, uma vez que os dados possuem formas diferentes de comportamento e erro pequeno.

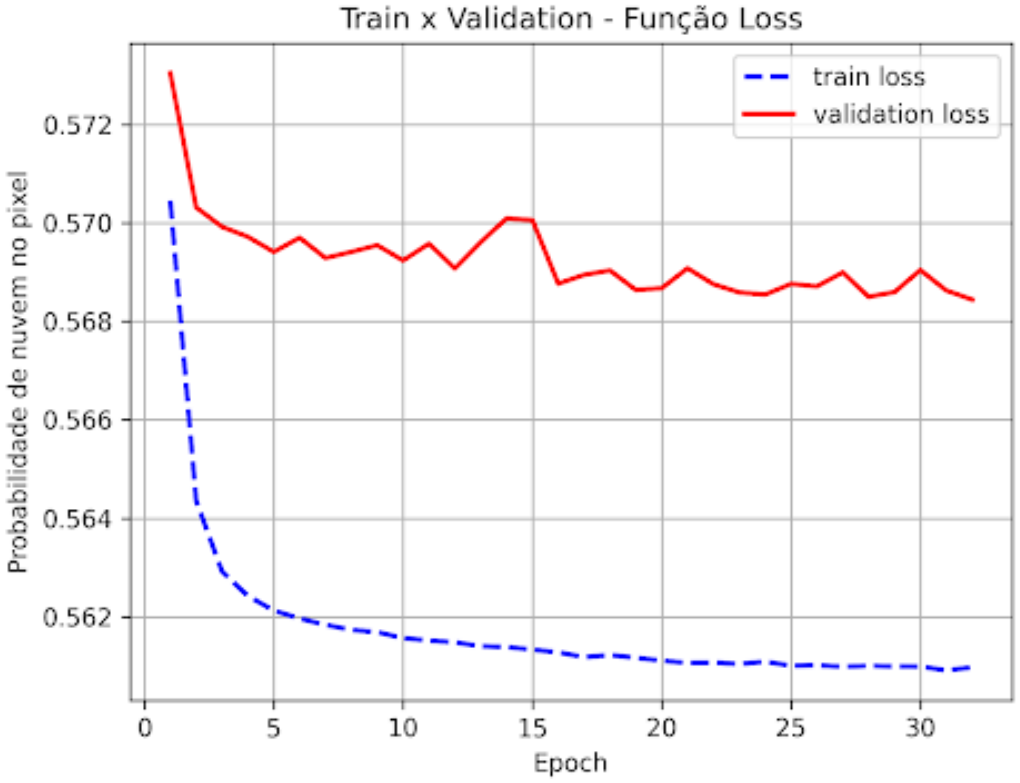


Figura 3: Loss de treino e validação em cada epoch.

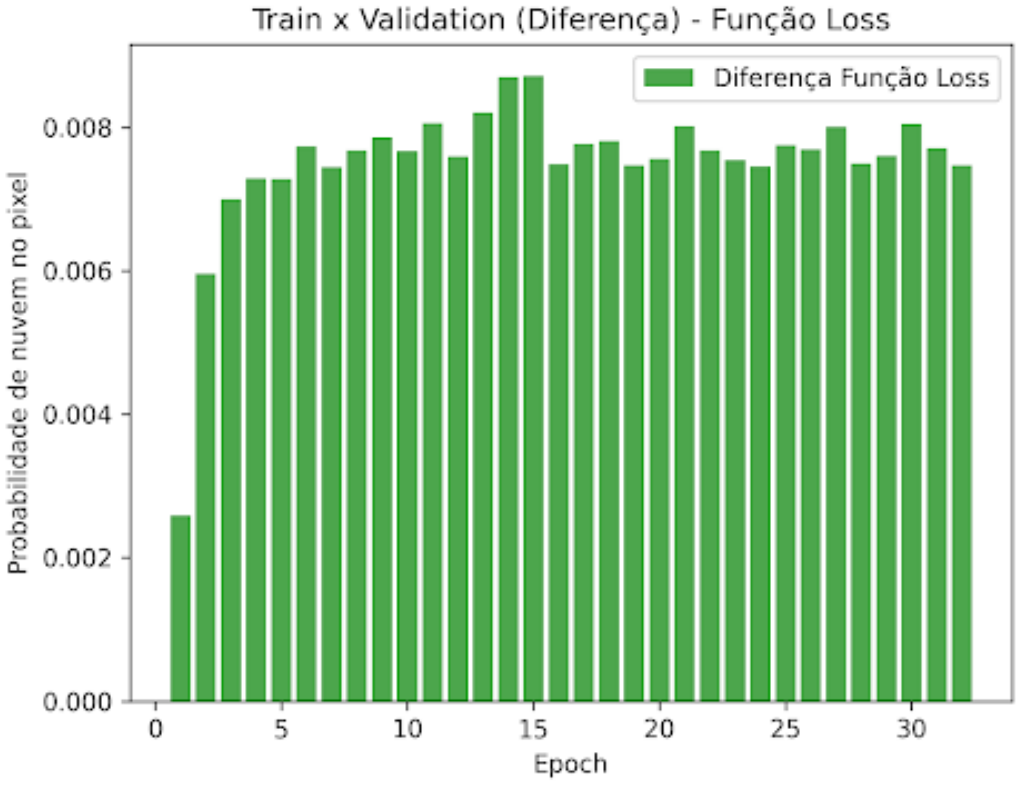


Figura 4: Diferença entre Loss de treino e validação em cada epoch.

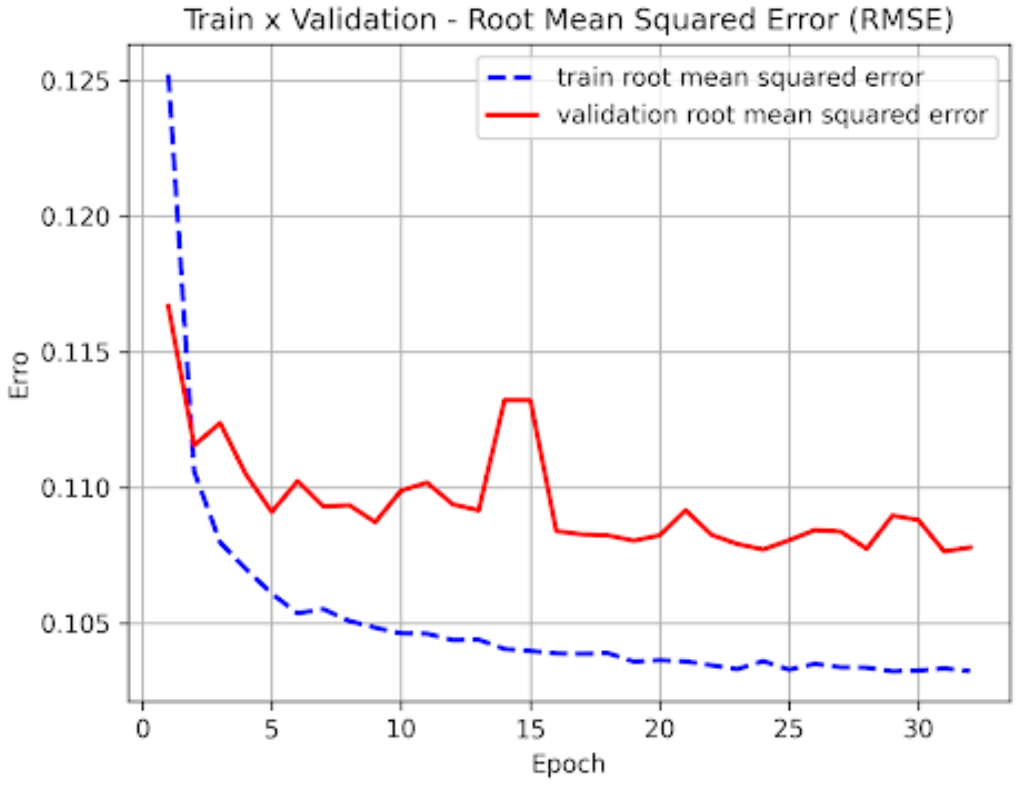


Figura 5: RMSE de treino e validação em cada epoch.

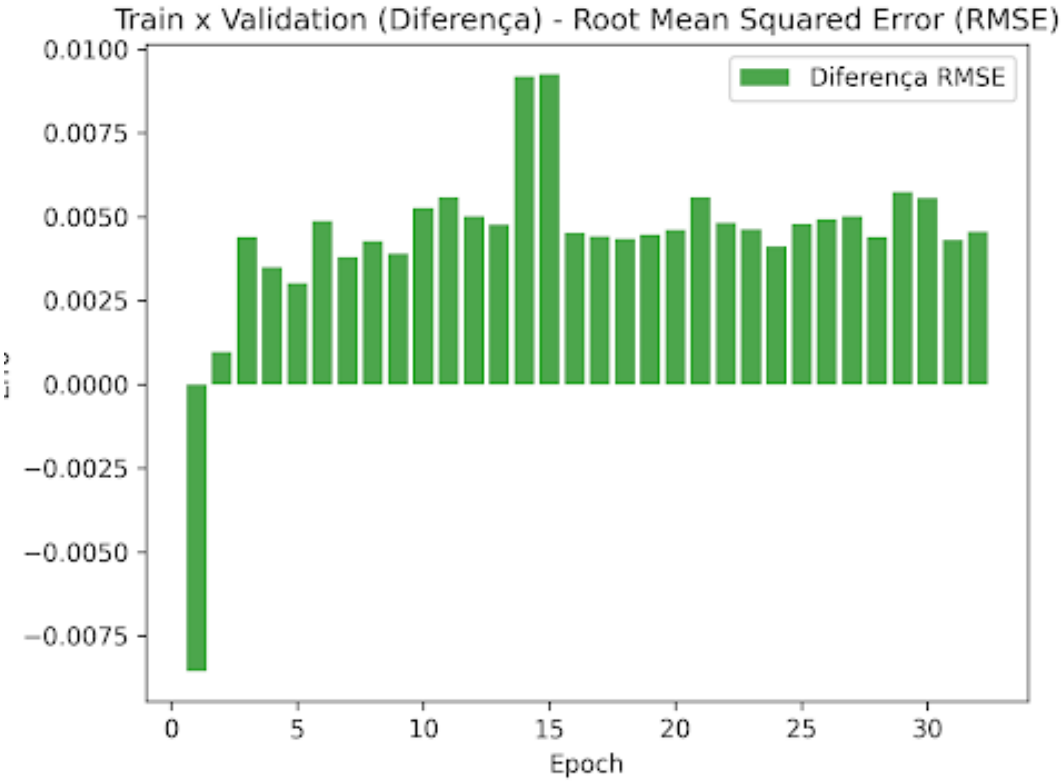


Figura 6: Diferença entre RMSE de treino e validação em cada epoch.

Na figura 7, consta a matriz de confusão gerada para o classificador binário. Analisando a mesma, percebe-se que ocorreram 177 falsos negativos e 215 falsos positivos. Em termos de porcentagem, os falsos negativos representam cerca de 3,59% das predições negativas e os falsos positivos 5,35% das predições positivas. As demais métricas encontradas para o classificador podem ser encontradas na tabela 4.

Classificador	**Precisão**	**Recall**	**F1-score**
Sem Nuvem	96%	96%	96%
Com Nuvem	96%	95%	95%

Tabela 4: Métricas do modelo binário

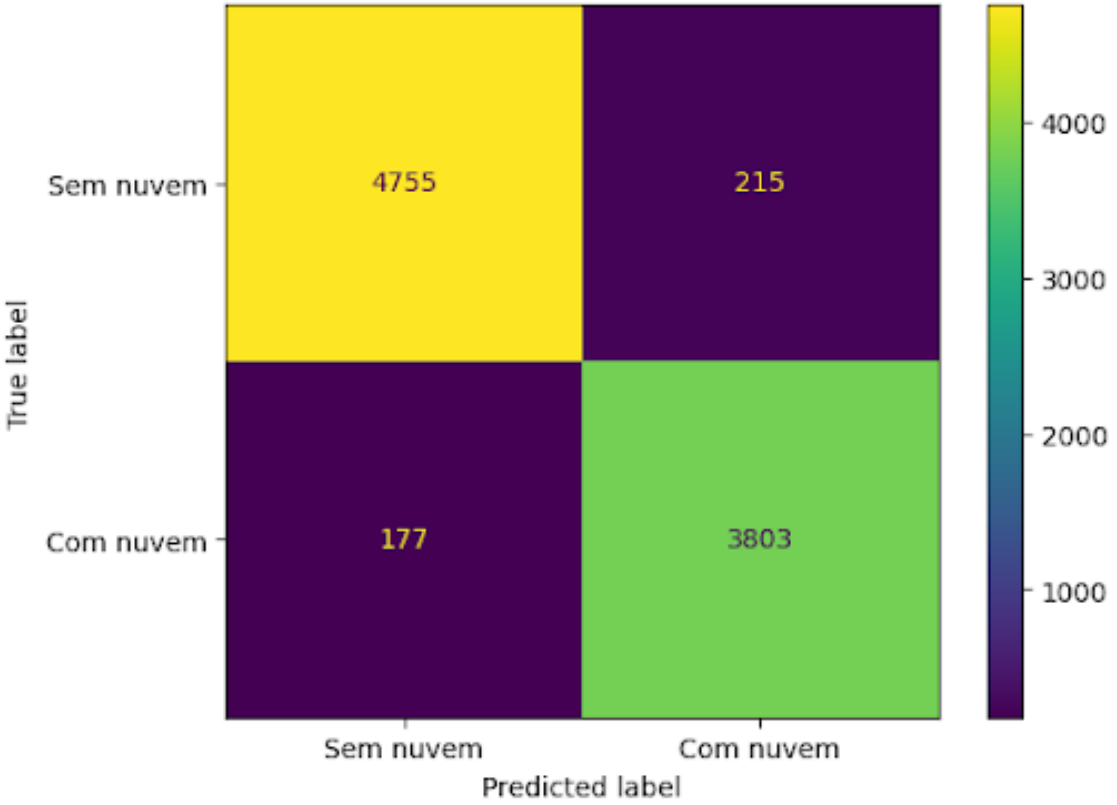
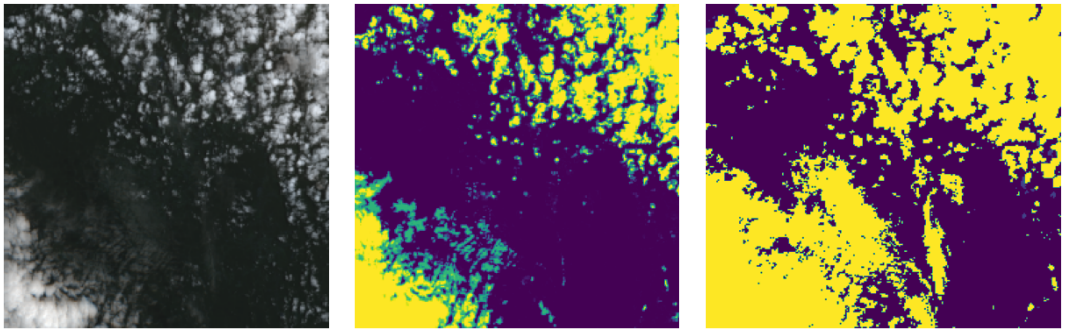


Figura 7: Matriz de confusão.

Na figura 8, pode-se visualizar a estimação do modelo em uma imagem de satélite. Nota-se que o modelo performa muito bem, porém mostra-se sensível, pois ele também identifica regiões claras no meio da imagem como nuvem. Além disso, as outras partes da imagem tem uma estimação correta, porém grosseira, com as bordas das nuvens não identificadas de forma suave. Para mudar isso, uma rede neural diferente poderia ser usada ou novas bandas poderiam ser adicionadas ao treino do modelo.



(a) Original (b) Máscara de nuvem original (c) Máscara de nuvem estimada

Figura 8: Comparação visual das máscaras de nuvem original e estimada.

Com base, nos resultados obtidos, nota-se que a performance do modelo desenvolvido foi bastante satisfatória.

6 Conclusões

A disciplina PTC3567 - Ciência dos Dados em Automação e Engenharia abordou diversos aspectos da Ciência de Dados, explorando desde o ciclo de vida dos dados até a implementação de técnicas avançadas como modelos de Deep Learning e computação em nuvem. Nosso projeto buscou integrar esses conhecimentos, aplicando-os em um contexto prático por meio do acesso às imagens de satélite disponíveis na Google Earth Engine.

O projeto começou pelo planejamento do projeto, por meio da elaboração de um Canvas (Seção 7) e de um Data Management Plan (Seção 8). Além disso, a preparação dos dados foi uma etapa necessária, uma vez que lidamos com imagens de satélite de grande escala, exigindo uma transformação para garantir a qualidade e a relevância das informações.

A escolha de utilizar Deep Learning para estimar a probabilidade de cobertura de nuvens em cada pixel revelou-se uma abordagem interessante e eficaz. A utilização de uma rede neural permitiu a modelagem de relações complexas nos dados, proporcionando uma precisão notável na estimação.

Este projeto não apenas alcançou os objetivos estabelecidos, mas também destacou a complexidade e as diferentes etapas envolvidas na aplicação da ciência de dados em um contexto do mundo real. A capacidade de integrar conhecimentos teóricos, habilidades técnicas e inovação prática oferece uma base sólida para futuros desafios e explorações na Ciência de Dados.

O projeto não só consolidou os conhecimentos adquiridos em sala de aula, mas também proporcionou uma compreensão mais profunda da aplicação prática da ciência de dados. A experiência de pensar e planejar a manipulação de grandes conjuntos de dados, aliada à implementação de técnicas avançadas, mostra o valor crescente dessa disciplina.

Referências

[1] Erick A. Schwarzelmuller; Lucas Fiori e Rodrigo Fill Rangel. Extração automática de corpos d'água ou de corpos vegetativos através de índices espectrais. Acessado em 02 de setembro de 2023.

[2] Mikael Vaaltola; Pauliina Paasivirta. project for cloud recognition from sentinel-2 satellite imagery using machine learning. `https://github.com/GispoCoding/sentinel2-geo-ai`. Acessado em 20 de novembro de 2023.

7 Apêndice 1 - Canvas

📊 Data Science Workflow Canvas*

Start here. The sections below are ordered intentionally to make you state your goals first, followed by steps to achieve those goals. You're allowed to switch orders of these steps!

Title:

1 Problem Statement What problem are you trying to solve? What larger issues do the problem address? Identificar a presença de nuvens em imagens de satélite	**2 Outcomes/Predictions** What prediction(s) are you trying to make? Identify applicable predictor (X) and/or target (y) variables. A predição é a probabilidade de haver nuvem em um pixel As variáveis preditoras X são as bandas de cor dos pixels e a variável alvo Y é a probabilidade de presença de nuvem no pixel	**3 Data Acquisition** Where are you sourcing your data from? Is there enough data? Can you work with it? Os dados serão retirados do dataset Google Earth Engine
4 Modeling What models are appropriate to use given your outcomes? Modelos de Aprendizado de Máquina Supervisionado	**5 Model Evaluation** How can you evaluate your model's performance? A intenção é fazer uma divisão do dataset após a remoção de outliers em dados de treino e teste, conseguindo avaliar qualitativamente e quantitativamente os acertos do modelo.	**6 Data Preparation** What do you need to do to your data in order to run your model and achieve your outcomes? Rodar análise descritiva dos dados Filtrar as colunas de interesse Separar o dataset em treino e teste

✅ Activation

When you finish filling out the canvas above, now you can begin implementing your data science workflow in roughly this order.

1 Problem Statement → 2 Data Acquisition → 3 Data Prep → 4 Modeling → 5 Outcomes/Preds → 6 Model Eval

* **Note:** This canvas is intended to be used as a starting point for your data science projects. Data science workflows are typically nonlinear.

Conceptualized by Jasmine Vasandani using notes from General Assembly's Data Science Immersive. Format inspired by Business Model Canvas.

8 Apêndice 2 - DMPTool

Estimador da Presença de Nuvens em Imagens de Satélite - Descrição dos Dados e Metadados produzidos pelo projeto

Descrição dos dados e metadados produzidos

Que dados serão coletados ou criados?

Os dados coletados serão imagens de satélite coletadas pelo Sentinel-2.

Como os dados serão coletados ou criados

Dados serão coletados por meio da API do Google Earth Engine.

Estimador da Presença de Nuvens em Imagens de Satélite - Restrições legais ou éticas

Descrição das restrições éticas e Legais

Como serão tratadas as questões éticas e legais?

O uso dos dados do Google Earth é permitido para fins de pesquisa e não comerciais.

Como serão tratadas as questões de direito e propriedade intelectual do autor?

O uso dos dados do Google Earth é permitido para fins de pesquisa e não comerciais.

Estimador da Presença de Nuvens em Imagens de Satélite - Política de preservação e compartilhamento

Política de preservação e compartilhamento

Como os dados serão armazenados e como serão realizadas as cópias de segurança durante a pesquisa?

Os dados serão coletados pela API do Google Earth Engine e automaticamente processados.

Como serão tratadas as questões de acesso e segurança?

O acesso será feito usando Python com a autenticação feita pela API do Google Earth Engine.

Estimador da Presença de Nuvens em Imagens de Satélite - Descrição de mecanismos, formatos e padrões para armazenamento

Descrição de mecanismos, formatos e padrões para armazenamento

---

Quem será responsável pelo gerenciamento dos dados?

Todos os participantes serão conjuntamente responsáveis.

Que recursos serão necessários para manter esse plano?

Os recursos utilizados serão os do Google Colabratory.

---

9 Apêndice 3 - Código

```
# -*- coding: utf-8 -*-
"""PLOTS S2CD_ImageToCsv + keras model.ipynb

Automatically generated by Colaboratory.

Original file is located at

    ↪ https://colab.research.google.com/drive/1U3PeTQlLvCnJJSUSoTOsIMhMn9icpUWk

# Reading data from Sentinel 2
"""

# Commented out IPython magic to ensure Python compatibility.
# %reload_ext autoreload
# %autoreload 2
# %matplotlib inline

import datetime as dt
import matplotlib.pyplot as plt
import numpy as np
import geemap
import pandas as pd
import ee

ee.Authenticate()

ee.Initialize()

# VALE DO RIBEIRA
geometry = ee.Geometry.Polygon(
  [[-47.013116, -24.254516], # direita em cima
  [-47.320733, -24.074090], # esquerda em cima
  [-48.378168, -24.739387], # esquerda embaixo
```

```
  [-48.021112, -24.958709]]) # direita embaixo
Map = geemap.Map()
Map.centerObject(geometry)
Map.addLayer(geometry,
             {'color': 'red'},
             'Geometry [red]: polygon')
display(Map)

# Seleciona imagens na regiao especificada no período especificado
sentinel2 = ee.ImageCollection("COPERNICUS/S2_SR") \
  .filterDate('2023-11-13', '2023-11-17') \
  .filterBounds(geometry)

def ee_array_to_df(arr, list_of_bands):

    df = pd.DataFrame(arr)

    # Rearrange the header.
    headers = df.iloc[0]
    df = pd.DataFrame(df.values[1:], columns=headers)

    # Drop null values
    df = df[[*list_of_bands]].dropna()

    # Convert the data to numeric values.
    for band in list_of_bands:
        df[band] = pd.to_numeric(df[band], errors='coerce')
        if band in ['B2', 'B3', 'B4', 'B8', 'B9']:
          df[band] = df[band]/65535
        if band == 'MSK_CLDPRB':
          df[band] = df[band]/100

    return df

# Define as bandas a serem extraídas pro csv
```

```
bands = ['MSK_CLDPRB', 'B2', 'B3', 'B4', 'B8', 'B9', 'SCL']

# Objeto ImageCollection da área de interesse
collection_area_of_interest = sentinel2 \
  .select(bands) \
  .getRegion(geometry, scale=500) \
  .getInfo()

# Dataframe com os dados dos pixels
sattelite_images_data = ee_array_to_df(collection_area_of_interest,
↪  bands)

sattelite_images_data.head()

sattelite_images_data.to_csv('PTC3567 - Identificador de presenca de
↪  nuvem.csv')

image_viz_params = {
    'bands': ['B2', 'B3', 'B4'],
}

geometry = ee.Geometry.Polygon([[-47.013116, -24.254516], # direita
↪  em cima
                                [-47.320733, -24.074090], # esquerda
                                ↪  em cima
                                [-48.378168, -24.739387], # esquerda
                                ↪  embaixo
                                [-48.021112, -24.958709]]) # direita
                                ↪  embaixo

# Add the image layer to the map and display it.
Map = geemap.Map(zoom=10)
Map.centerObject(geometry)
image0 = ee.Image(sentinel2.toList(sentinel2.size()).get(0))
image1 = ee.Image(sentinel2.toList(sentinel2.size()).get(1))
```

```
image2 = ee.Image(sentinel2.toList(sentinel2.size()).get(2))
image3 = ee.Image(sentinel2.toList(sentinel2.size()).get(3))
image4 = ee.Image(sentinel2.toList(sentinel2.size()).get(4))
Map.add_layer(image0, image_viz_params, 'image 0')
Map.add_layer(image1, image_viz_params, 'image 1')
Map.add_layer(image2, image_viz_params, 'image 2')
Map.add_layer(image3, image_viz_params, 'image 3')
Map.add_layer(image4, image_viz_params, 'image 4')
display(Map)

"""## Neural network"""

# Commented out IPython magic to ensure Python compatibility.
import tensorflow as tf
import numpy as np
import random
import matplotlib.pyplot as plt
import pandas as pd
import gzip

# %matplotlib inline

"""## Data preprocessing"""

from sklearn.model_selection import train_test_split

data = sattelite_images_data

X = data.drop('MSK_CLDPRB', axis=1)
y = data[['MSK_CLDPRB']]

X_train, X_test, y_train, y_test = train_test_split(X, y,
↪  test_size=0.2, random_state=42)

from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

"""## Rede neural"""

from tensorflow import keras
from keras.callbacks import EarlyStopping
from keras.models import Sequential
from keras.layers import Dense
from keras import backend as K
from keras.metrics import RootMeanSquaredError

# Let's make results reproducable
from numpy.random import seed
seed(1)
from tensorflow import random
random.set_seed(2)

tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)

early_stopping_monitor = EarlyStopping(patience=5)

model = Sequential()
n_cols = np.shape(X_train)[1]
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
↪  loss='poisson',
↪  metrics=[tf.keras.metrics.MeanSquaredError(),RootMeanSquaredError()])
print("fit begins")
```

```
history = model.fit(X_train, y_train, epochs=32, batch_size=12,
 ↪  validation_split=0.2, callbacks=[early_stopping_monitor],
          use_multiprocessing=True, verbose=2)
print("predicting begins")
mlppre = model.predict(X_test)

mlppre = pd.DataFrame(mlppre,columns=['Predicted'])
y_mlppre = pd.concat([mlppre,y_test.reset_index(drop=True)],axis=1)

print(model.evaluate(X_train, y_train)[1])
print(model.evaluate(X_test, y_test)[1])

train_loss = history.history['loss']
validation_loss = history.history['val_loss']
loss_diff  = [a - b for a, b in zip(validation_loss, train_loss)]
train_mean_squared_error = history.history['mean_squared_error']
validation_mean_squared_error =
 ↪  history.history['val_mean_squared_error']
mean_squared_error_diff  = [a - b  for a, b in
 ↪  zip(validation_mean_squared_error, train_mean_squared_error)]
train_root_mean_squared_error =
 ↪  history.history['root_mean_squared_error']
validation_root_mean_squared_error =
 ↪  history.history['val_root_mean_squared_error']
root_mean_squared_error_diff  = [a - b  for a, b in
 ↪  zip(validation_root_mean_squared_error,
 ↪  train_root_mean_squared_error)]
num_epochs = np.arange(1,len(train_loss)+1)
plt.plot(num_epochs, train_loss, label='train loss', color='blue',
 ↪  linestyle='--', linewidth=2)
plt.plot(num_epochs, validation_loss, label='validation loss',
 ↪  color='red', linestyle='-', linewidth=2)

plt.title('Train x Validation - Função Loss')
plt.xlabel('Epoch')
```

```
plt.ylabel('Probabilidade de nuvem no pixel')
plt.legend()

plt.grid(True)
plt.savefig('comparacao_loss.png', dpi=300)
plt.close()

plt.plot(num_epochs, train_root_mean_squared_error, label='train root
↪  mean squared error', color='blue', linestyle='--', linewidth=2)
plt.plot(num_epochs, validation_root_mean_squared_error,
↪  label='validation root mean squared error', color='red',
↪  linestyle='-', linewidth=2)

plt.title('Train x Validation - Root Mean Squared Error (RMSE)')
plt.xlabel('Epoch')
plt.ylabel('Probabilidade de nuvem no pixel')
plt.legend()
plt.grid(True)
plt.savefig('comparacao_rmse.png', dpi=300)
plt.close()

plt.bar(num_epochs, loss_diff, label='Diferença Loss', color='green',
↪  alpha=0.7)
plt.title('Train x Validation (Diferença) - Função Loss')
plt.xlabel('Epoch')
plt.ylabel('Probabilidade de nuvem no pixel')
plt.legend()
plt.savefig('diferenca_loss.png', dpi=300)
plt.close()

plt.bar(num_epochs, root_mean_squared_error_diff, label='Diferença
↪  RMSE', color='green', alpha=0.7)
plt.title('Train x Validation (Diferença) - Root Mean Squared Error
↪  (RMSE)')
plt.xlabel('Epoch')
```

```
plt.ylabel('Probabilidade de nuvem no pixel')
plt.legend()
plt.savefig('diferenca_rmse.png', dpi=300)
plt.close()

from sklearn.metrics import confusion_matrix, classification_report,
↪  ConfusionMatrixDisplay

previsoes_modelo = y_mlppre["Predicted"]
rotulos_verdadeiros = y_mlppre["MSK_CLDPRB"]

# Definindo o threshold
threshold = 0.5

# Convertendo as previsões contínuas para previsões binárias usando
↪  o threshold
previsoes_binarias = (previsoes_modelo > threshold).astype(int)
rotulos_binarios = (rotulos_verdadeiros > threshold).astype(int)


# Criando a matriz de confusão
matriz_confusao = confusion_matrix(rotulos_binarios,
↪  previsoes_binarias)

labels = ['Sem nuvem','Com nuvem']
disp = ConfusionMatrixDisplay(matriz_confusao, display_labels=labels)
disp.plot()

report = classification_report(previsoes_binarias, rotulos_binarios)
print(report)

y_mlppre

# Let's check that we are not predicting always the same label
print(np.count_nonzero(mlppre<0.5))
```

```
print(np.count_nonzero(mlppre>=0.5))
```