

Entregável 3.1. Manual de utilização operacional da metodologia implementada na cadeia de produção da DGT

Versão v01

Índice

1.	Introdução ao PyCCD (v01).....	3
2.	Instalação da plataforma e configuração de ambientes virtuais.....	3
2.1.	Instalação do Anaconda Navigator	3
2.2.	Criação do Ambiente Virtual para a execução do PyCCD (ccdISA_v2).....	4
2.2.1.	Passos para instalação do Ambiente Virtual no Windows	4
2.2.2.	Carregar e executar scripts em Spyder	7
3.	Estrutura e organização de dados e scripts	9
4.	Instalação do PyCCD na máquina local.....	10
5.	Inputs do PyCCD	12
6.	Pré-processamento do PyCCD	13
7.	Processamento do PyCCD.....	14
8.	Outputs do PyCCD	14
9.	Download Imagens Sentinel-2 do GEE	17
9.1.	Criação do Ambiente virtual para download de Imagens Sentinel-2 do GEE (gee_env) 17	
9.2.	Configuração da Autenticação do Google Earth Engine em Spyder	19
9.3.	Imagens Sentinel-2 para processamento no CCD	21
9.4.	Imagens Sentinel-2 para análise visual	24

Índice de Figuras

Figura 1 – Download do ficheiro yml para Windows no repositório GitHub.	5
Figura 2 – Abrir o “Anaconda Prompt” através do menu Iniciar.	5
Figura 3 – Comando que cria o ambiente virtual ccdISA_v2.....	5
Figura 4 – Lista de ambientes virtuais instalados no Anaconda através de linha de comandos...	5
Figura 5 – Lista de ambientes virtuais instalados visíveis no Anconda Navigator.	6
Figura 6 – Linha de comandos para exibir as versões dos packages instalados no ambiente virtual ccdISA_v2 (à esquerda) ficheiro yml com as versões dos packages (à direita).	6
Figura 7 –Versões dos packages instalados no ambiente virtual ccdISA_v2 visíveis no Anaconda Navigator.	7
Figura 8 – Caminho onde o ambiente virtual está instalado.....	7
Figura 9 – Instalação do Spyder via Anaconda Navigator.	7
Figura 10 – Iniciar o Spyder no ambiente virtual ccdISA_v2 via Anaconda Navigator.	8

Figura 11 – Spyder com o ambiente virtual ccdISA_v2 visível na barra inferior canto esquerdo.	8
Figura 12 – Abrir scripts no Spyder.....	8
Figura 13 – Botão para correr scripts, destacado com um retângulo a vermelho.	8
Figura 14 – Download da pasta scripts do repositório S2CHANGE/PyCCD do GitHub.....	11
Figura 15 – Localização da PASTA_DE_SCRIPTS no script main.py.	11
Figura 16 – Localização dos inputs no script main.py.....	12
Figura 17 – Localização dos parâmetros de pré-processamento no script main.py.....	13
Figura 18 – Estrutura de pastas de outputs por tiles.	15
Figura 19 – Localização dos parâmetros para construir o plot no script main.py.	15
Figura 20 – Gráfico gerado pelo CCD indicando as alterações detetadas no pixel situado em latitude 40.97141 e longitude -8.12198.....	15
Figura 21 – Informação do ficheiro shapefile para o mesmo pixel representado na Figura 20. .	16
Figura 22 – Localização dos outputs no script main.py (acima) e estrutura das pastas de outputs na máquina local (abaixo).	17
<i>Figura 23 – Criação de ambientes virtuais na plataforma Anaconda.</i>	<i>18</i>
<i>Figura 24 – Definição do nome do ambiente virtual a ser criado.</i>	<i>18</i>
<i>Figura 25 – Abrir um terminal de comandos dentro do ambiente virtual criado.....</i>	<i>18</i>
<i>Figura 26 – Terminal de comandos dentro do ambiente virtual criado.</i>	<i>18</i>
Figura 27 – Iniciar o Spyder no ambiente virtual gee_env via Anaconda Navigator.	19
<i>Figura 28 – Instrução que deverá ser escrita na consola do spyder.</i>	<i>19</i>
<i>Figura 29 – Mensagem que deverá aparecer após executar o comando da Figura 28.</i>	<i>19</i>
<i>Figura 30 – Gerar o token de autenticação do GEE.....</i>	<i>20</i>
<i>Figura 31 – Instrução da consola de Spyder onde deverá ser introduzido o token gerado na Figura 30. A mensagem da 2ª linha indica sucesso na autenticação.....</i>	<i>20</i>
Figura 32 – Validação da inicialização do Google Earth Engine (EE) no ambiente Python, sem erros após a execução dos comandos import ee e ee.Initialize().....	21
Figura 33 – Inputs do script download-sentinel2-from-gee.py.....	22
Figura 34 – Exemplos de pastas com os nomes dos tiles S2 na pasta pessoal do Google Drive.	23
Figura 35 – Parâmetros a escolher caso se opte pela divisão das geometrias.....	23
Figura 36 – Exemplos de pastas quando o script realiza a divisão automática das geometrias.	24
Figura 37 – Inclusão da banda NDVI às bandas requeridas para o download de imagens S2 via GEE.	24
Figura 38 – Implementação da transformação de escala do NDVI no script download-sentinel2-from-gee.py, com a opção para desativar a transformação, utilizando a função destacada a vermelho.	25

Índice de Tabelas

Tabela 1 – Referências de caminhos e pastas utilizadas como exemplo no manual.....	10
Tabela 2 – Pastas que o utilizador deverá ter de definir.....	11
Tabela 3 – Etapas de redução das bandas Sentinel-2 para as bandas finais utilizadas no processamento PyCCD.	13

1. Introdução ao PyCCD (v01)

O PyCCD (*Python Continuous Change Detection*) é um algoritmo originalmente desenvolvido para a análise de séries temporais de dados do satélite LANDSAT, disponível em <https://code.usgs.gov/lcmap/pyccd>. No contexto do projeto “Desenvolvimento de mapas de perdas recentes de floresta e mato em Portugal derivados de imagens de satélite”, o algoritmo foi adaptado para lidar com dados do satélite Sentinel-2, ampliando a sua aplicabilidade.

Esta versão modificada, também chamada de PyCCD (v01), teve como objetivo principal ajustar as diferenças entre os dados provenientes dos dois satélites. Um dos principais ajustes feitos consistiu na exclusão da banda térmica, presente no LANDSAT, mas ausente no Sentinel-2. Além disso, os dados do Sentinel-2 já incluem correções atmosféricas, diferentemente dos dados do LANDSAT, onde esse processamento era feito dentro do próprio algoritmo.

Adicionalmente, foram adicionados alguns parâmetros ao modelo que anteriormente não estavam incorporados, como 'CHISQUAREPROB' e 'ALPHA' sendo este último responsável por determinar a intensidade da regularização aplicada ao modelo.

As adaptações feitas no PyCCD proporcionaram uma validação eficaz do modelo, alcançando um F1-Score superior a 80% (ver Entregável 1.3), o que possibilitou uma deteção precisa e confiável das alterações na vegetação ao longo do tempo.

2. Instalação da plataforma e configuração de ambientes virtuais

Neste capítulo, abordaremos o processo de instalação do Anaconda Navigator, que será utilizado para gerenciar os ambientes virtuais necessários para executar o PyCCD. A criação de dois ambientes virtuais distintos será essencial:

1. O primeiro ambiente será dedicado à execução do CCD → ccdISA_v2. Nota: Este procedimento é obrigatório para executar PyCCD sobre imagens Sentinel-2 de que o utilizador dispõe (descarregadas através de GEE ou obtidas por outra via, e.g. imagens Theia).
2. O segundo ambiente será usado para realizar o download das imagens Sentinel-2 a partir do Google Earth Engine (GEE) → gee_env. Nota: este procedimento é opcional, caso os utilizadores já tenham acesso a imagens Sentinel-2 de outra origem (ver secção 9.1).

2.1. Instalação do Anaconda Navigator

O Anaconda Navigator é uma plataforma gráfica que facilita a gestão de packages e ambientes para projetos em Python, permitindo a criação de ambientes isolados para diferentes aplicações. Abaixo estão as etapas para sua instalação:

1. Acessar o site oficial do Anaconda, disponível em: <https://www.anaconda.com/products/navigator>
2. Clique no botão **"Download"** para baixar a versão mais adequada para o seu sistema operacional (Windows, macOS ou Linux).
3. Após o download, siga as instruções de instalação específicas para o seu sistema.

Além do Anaconda Navigator, o Anaconda Prompt também será instalado. O Anaconda Prompt é uma ferramenta de linha de comando que permite a instalação e gestão de packages e ambientes diretamente pelo terminal, oferecendo controle total sobre as operações realizadas em ambientes Python.

Para garantir que o Anaconda foi instalado corretamente, siga os passos abaixo:

- Abra o `Anaconda Navigator`. Caso tenha instalado no Windows, poderá encontrá-lo no menu Iniciar. No macOS ou Linux, poderá abrir o Anaconda digitando `anaconda-navigator` no terminal.
- A interface gráfica do Anaconda Navigator deverá ser exibida.

Além disso, poderá abrir o Anaconda Prompt (no Windows, disponível no menu Iniciar) para realizar operações mais avançadas de instalação de packages, criação e ativação de ambientes virtuais diretamente por comandos sem a necessidade da interface gráfica, conforme será mostrado mais à frente.

2.2. Criação do Ambiente Virtual para a execução do PyCCD (ccdISA_v2)

Agora que o Anaconda Navigator está instalado, o próximo passo é criar o ambiente virtual para isolar as dependências do projeto CCD. O objetivo é configurar um ambiente onde as versões corretas dos packages e a versão de Python necessários para o processamento do PyCCD serão instaladas. Para isso, utilizamos os ficheiros “yml” disponíveis no repositório GitHub do projeto. O ficheiro yml é específico para um sistema operacional, garantindo que todas as dependências sejam instaladas de forma compatível com o ambiente. Dessa forma, podemos assegurar que o projeto será executado de maneira estável e sem conflitos de versão entre packages:

- **WINDOWS**
https://github.com/manuelcampagnolo/S2CHANGE/blob/main/ccdISA_win.yml
- **LINUX**
https://github.com/manuelcampagnolo/S2CHANGE/blob/main/ccdISA_linux.yml

2.2.1. Passos para instalação do Ambiente Virtual no Windows

Os passos seguintes só precisam de ser realizados uma única vez para instalar o ambiente virtual.

1. Download do `ficheiro yml` (apropriado para o seu sistema operacional):

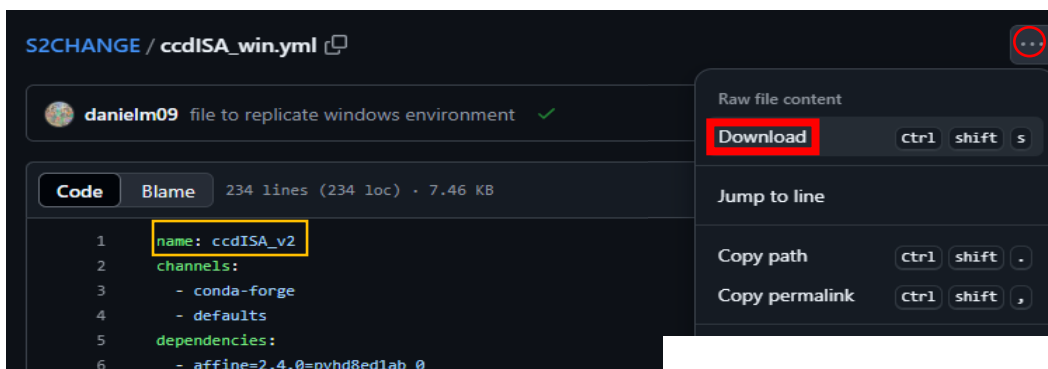


Figura 1 – Download do ficheiro yml para Windows no repositório GitHub.

2. Aceder ao terminal do Anaconda, escrevendo no menu Iniciar do Windows “Anaconda Prompt (anaconda3)” e abrir (note que também pode abrir o terminal no Anaconda Navigator):

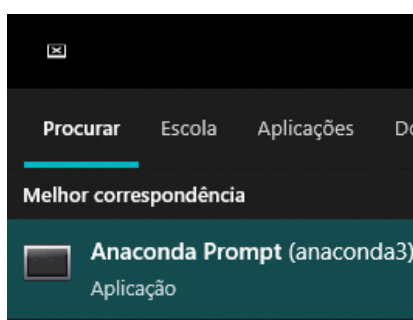


Figura 2 – Abrir o “Anaconda Prompt” através do menu Iniciar.

3. Criar o ambiente virtual executando o seguinte comando no terminal: `conda env create -f C:/Users/Utilizador/Downloads/ccdISA_win.yml` (**ATENÇÃO:** o Utilizador varia). Este comando criará um ambiente virtual com o nome especificado no cabeçalho do que, neste caso, é `ccdISA_v2` (ver Figura 1 caixa a amarelo). O comando visa instalar todos os packages necessários e a versão de Python, neste caso 3.10.14, no ambiente, usando as versões listadas no ficheiro yml. Pretende-se algo como:

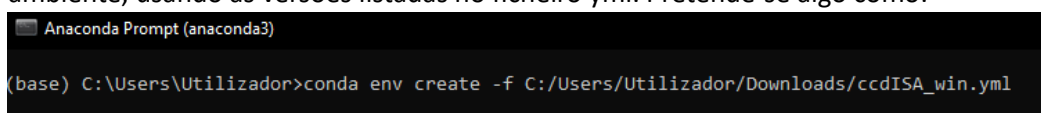


Figura 3 – Comando que cria o ambiente virtual `ccdISA_v2`.

4. Verificar se o ambiente foi criado corretamente: `conda env list` e verificar se consta o nome `ccdISA_v2` na lista:

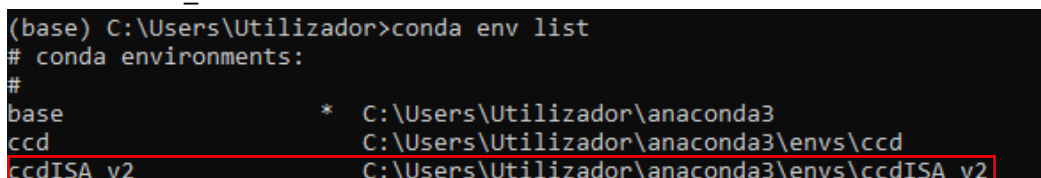


Figura 4 – Lista de ambientes virtuais instalados no Anaconda através de linha de comandos.

O ambiente criado, `ccdISA_v2`, também será automaticamente listado no Anaconda Navigator sob a aba “**Environments**”, permitindo um fácil acesso:

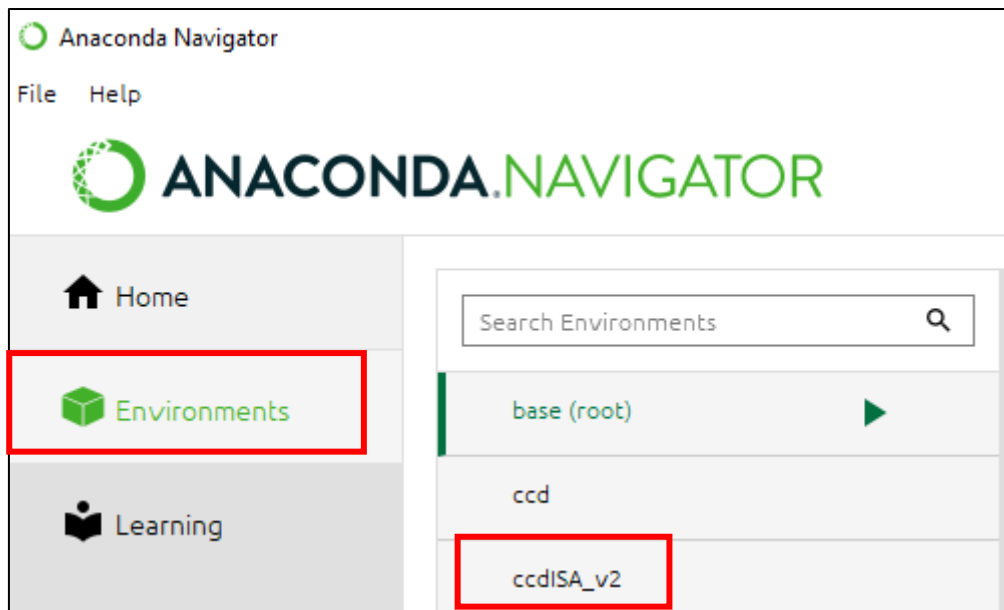


Figura 5 – Lista de ambientes virtuais instalados visíveis no Anaconda Navigator.

5. Ativar o ambiente virtual no terminal: (base) `conda activate ccdISA_v2`
6. O utilizador pode verificar que o ambiente `ccdISA_v2` contém as versões corretas dos packages listados no ficheiro yml: (ccdISA_v2) `conda list`

```
(ccdISA_v2) C:\Users\Utilizador>conda list
# packages in environment at C:\Users\Utiliz
#
# Name                        Version
affine                        2.4.0
attrs                        23.2.0
aws-c-auth                    0.7.16
aws-c-cal                     0.6.10
aws-c-common                  0.9.14
aws-c-compression             0.2.18
aws-c-event-stream            0.4.2
aws-c-http                    0.8.1
aws-c-io                      0.14.6
aws-c-mqtt                    0.10.3
aws-c-s3                      0.5.5
aws-c-sdkutils                0.1.15
aws-checksums                 0.1.18
aws-crt-cpp                   0.26.4
```

```
! ccdISA_win.yml X
C: > Users > Utilizador > Downloads > ! ccdISA_win.yml
1  name: ccdISA_v2
2  channels:
3    - conda-forge
4    - defaults
5  dependencies:
6    - affine=2.4.0=pyhd8ed1ab_0
7    - attrs=23.2.0=pyh71513ae_0
8    - aws-c-auth=0.7.16=h7613915_8
9    - aws-c-cal=0.6.10=hf6fcf4e_2
10   - aws-c-common=0.9.14=hcfcfb64_0
11   - aws-c-compression=0.2.18=hf6fcf4e_2
12   - aws-c-event-stream=0.4.2=h3df98b0_6
13   - aws-c-http=0.8.1=h4e3df0f_7
14   - aws-c-io=0.14.6=hf0b8b6f_2
15   - aws-c-mqtt=0.10.3=h96fac68_2
16   - aws-c-s3=0.5.5=h08df315_0
17   - aws-c-sdkutils=0.1.15=hf6fcf4e_2
18   - aws-checksums=0.1.18=hf6fcf4e_2
19   - aws-crt-cpp=0.26.4=h944602d_3
```

Figura 6 – Linha de comandos para exibir as versões dos packages instalados no ambiente virtual `ccdISA_v2` (à esquerda) ficheiro yml com as versões dos packages (à direita).

Também poderá visualizar as versões dos packages e do Python através da interface gráfica do Anaconda Navigator, selecionando a aba "**Environments**" e clicando em `ccdISA_v2`, onde as versões serão exibidas à direita:

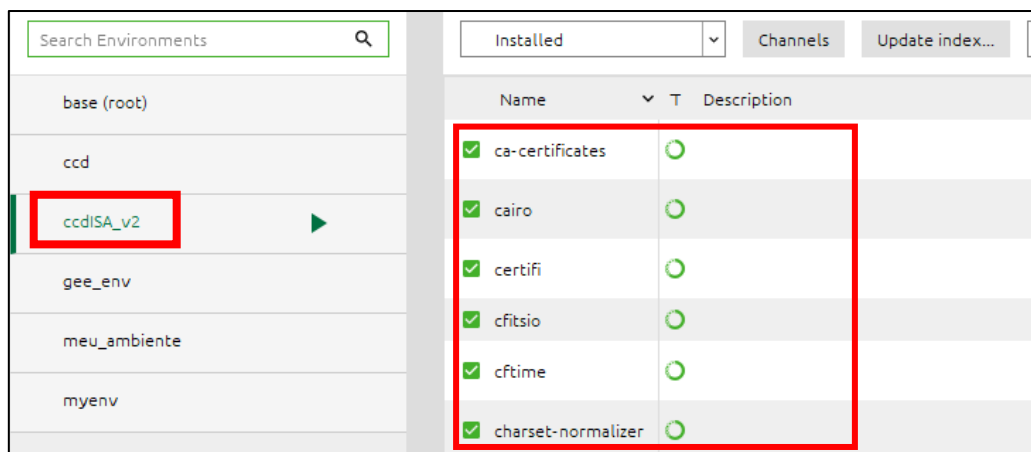


Figura 7 – Versões dos packages instalados no ambiente virtual `ccdISA_v2` visíveis no Anaconda Navigator.

7. O interpretador de Python do ambiente virtual estará instalado na seguinte diretoria:

```
(ccdISA_v2) C:\Users\Utilizador>where python
C:\Users\Utilizador\anaconda3\envs\ccdISA_v2\python.exe
```

Figura 8 – Caminho onde o ambiente virtual está instalado.

8. Para instalar o editor Spyder no ambiente `ccdISA_v2` e executar os scripts: aceda à secção **"Home"** do Anaconda Navigator, certifique-se de que o ambiente ativo é o `ccdISA_v2`, e clique em **"Install"** na aplicação do Spyder (pode levar algum tempo a instalar):

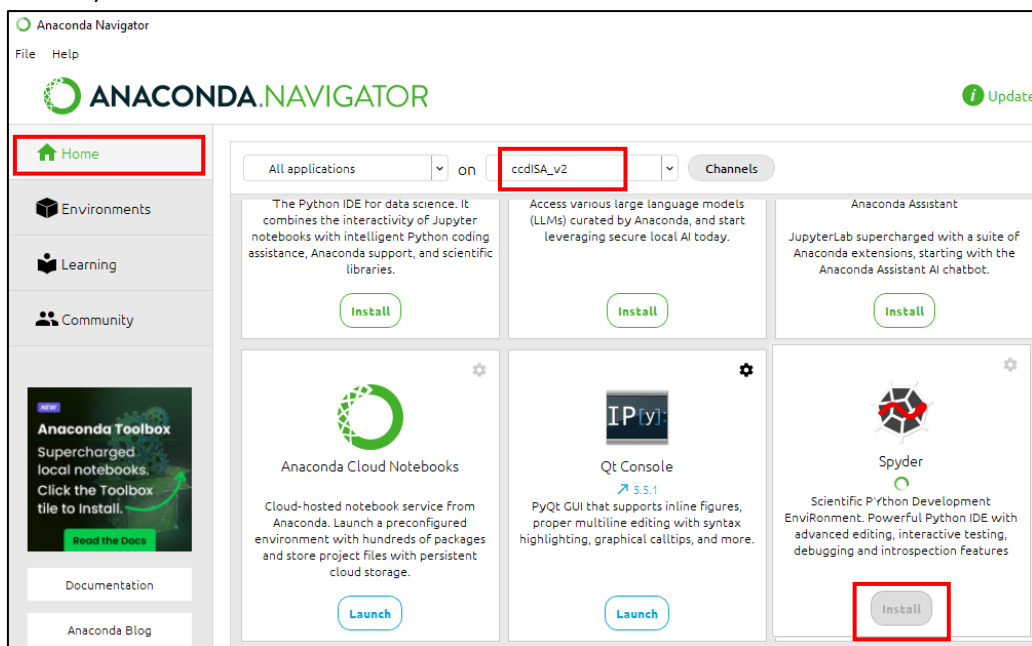


Figura 9 – Instalação do Spyder via Anaconda Navigator.

2.2.2. Carregar e executar scripts em Spyder

Numa sessão de trabalho, o utilizador acede ao ambiente `ccdISA_v2` através do Anaconda Navigator.

1. Para abrir o Spyder, acesse a seção **"Home"** do Anaconda Navigator, confirme que o ambiente ativo é o `ccdISA_v2`, e clique em **"Launch"** na aplicação do Spyder:

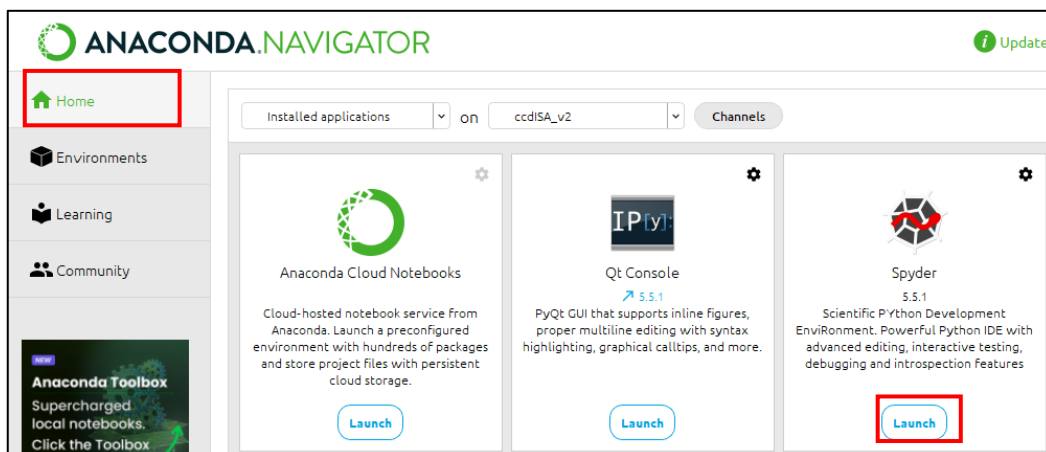


Figura 10 – Iniciar o Spyder no ambiente virtual `ccdISA_v2` via Anaconda Navigator.

2. É possível verificar se está a trabalhar no ambiente virtual `ccdISA_v2` no Spyder, assim como a versão do Python (3.10.14), observando a barra inferior no canto esquerdo, onde é exibido o nome do ambiente ativo:

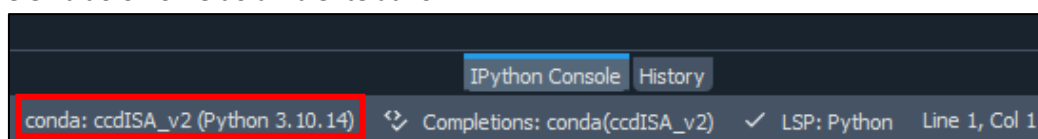


Figura 11 – Spyder com o ambiente virtual `ccdISA_v2` visível na barra inferior canto esquerdo.

3. Para abrir os scripts no Spyder faça **File > Open**:

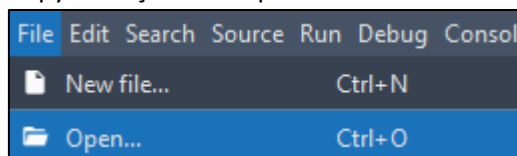


Figura 12 – Abrir scripts no Spyder.

4. Para executar os scripts no Spyder deve usar o botão indicado abaixo (ou **Run > Run**):

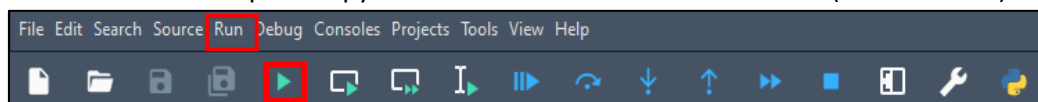
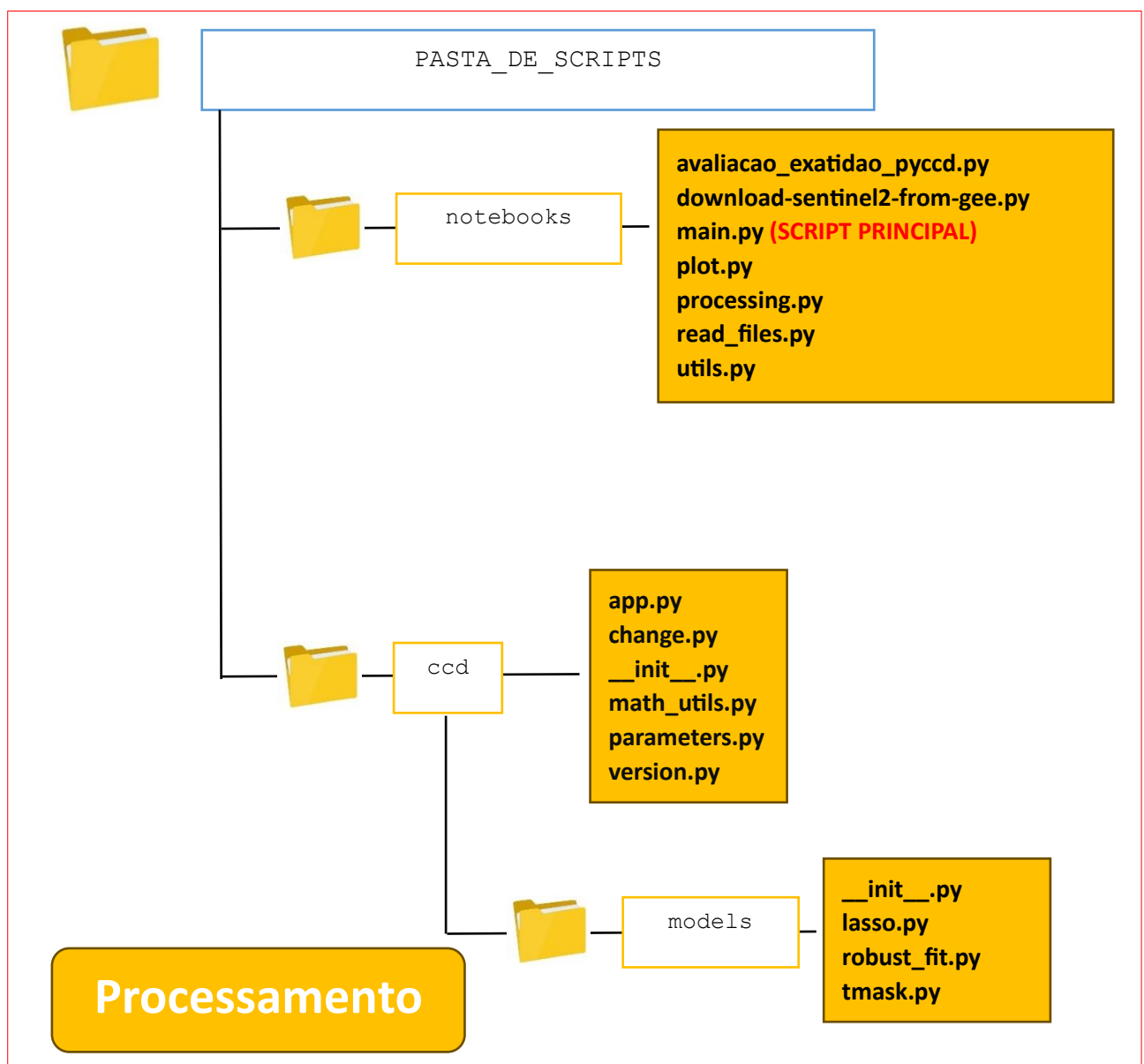
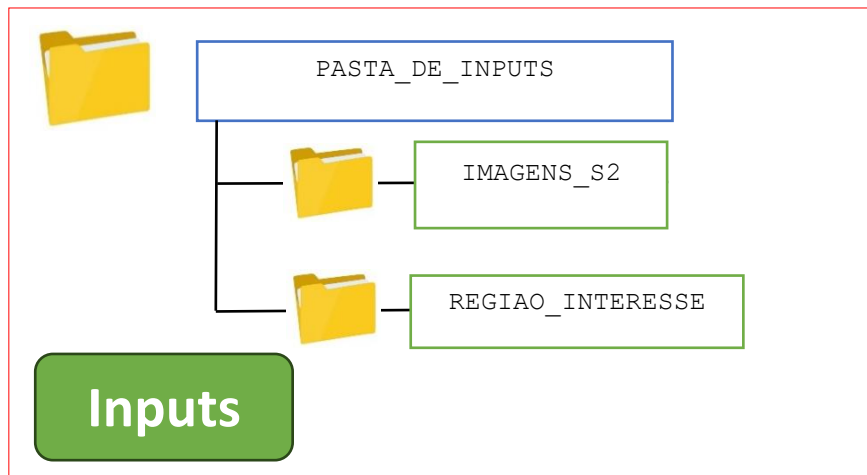
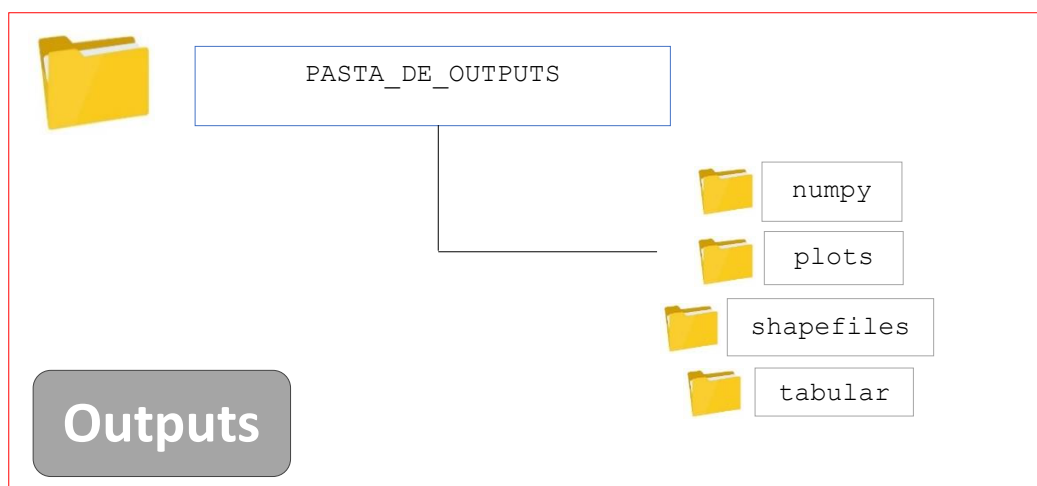


Figura 13 – Botão para correr scripts, destacado com um retângulo a vermelho.

3. Estrutura e organização de dados e scripts





Nos exemplos do manual, as pastas utilizadas correspondem a:

PASTA_DE_INPUTS	C:\Users\Public\Documents
IMAGENS_S2	...\imagens_theia ou ...\imagens_gee
REGIAO_INTERESSE	...\BDR_300_artigo ou ...\BDR_Navigator
PASTA_DE_SCRIPTS	C:\Users\Utilizador\Desktop\CCD_yaml_win\S2CHANGE\scripts\pyccd_theia\
PASTA_DE_OUTPUTS	C:\Users\Public\Documents
OUTPUTS_RI	...\outputs_BDR-NAV ou ...\outputs_BDR300

Tabela 1 – Referências de caminhos e pastas utilizadas como exemplo no manual.

O utilizador é livre para escolher os caminhos que preferir, bastando configurar os caminhos adequados nos scripts. No manual, como região de interesse, utilizou-se o conjunto de dados geográficos BDR_DGT e BDR_NAVIGATOR, que correspondem às bases de dados de referência usadas na avaliação do algoritmo. No entanto, é importante reforçar que o uso do PyCCD não requer, necessariamente, uma base de dados de referência; qualquer região de interesse pode ser utilizada para a análise.

O processo inicia-se com o pré-processamento dos dados, ajustado de acordo com os inputs fornecidos. Em seguida, são executadas as funções que implementam o algoritmo CCD para detetar mudanças na vegetação, resultando na produção dos outputs.

4. Instalação do PyCCD na máquina local

Os ficheiros do PyCCD estão disponíveis num repositório em GitHub. Como o repositório contém outros ficheiros usa-se a seguinte funcionalidade para descarregar apenas os ficheiros das pastas de scripts. Após a instalação do ambiente virtual, é necessário acessar o site DownGit (disponível em: <https://downgit.evecalm.com/#/home>) e inserir o seguinte URL: https://github.com/manuelcampagnolo/S2CHANGE/tree/main/scripts/pyccd_theia. Em

seguida, clique em **"Download"**, o que resultará numa pasta ZIP, que deverá ser descompactada na pasta PASTA_DE_SCRIPTS:

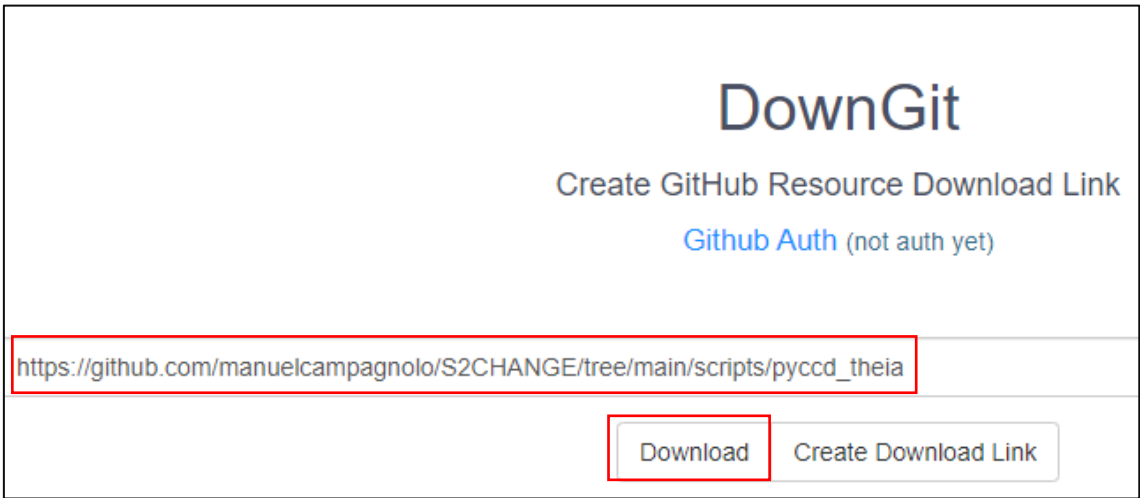


Figura 14 – Download da pasta scripts do repositório S2CHANGE/PyCCD do GitHub.

O utilizador irá ter de criar as pastas listadas na Tabela 2.

PASTA_DE_INPUTS	Pasta onde terá de colocar os ficheiros de input de acordo com as pastas indicadas abaixo.
IMAGENS_S2	Ficheiros em formato GeoTiff
REGIAO_INTERESSE	Ficheiro em formato Geopackage ou Shapefile
PASTA_DE_SCRIPTS	Pasta onde terá de colocar os scripts descompactados, descarregados com o DownGit. Há duas subpastas que a estrutura é preciso manter “ccd” e “notebooks”.
PASTA_DE_OUTPUTS	Pasta onde são colocados os outputs do processamento do PyCCD.

Tabela 2 – Pastas que o utilizador deverá ter de definir.

Para iniciar o spyder no ambiente virtual ccdISA_v2, deve seguir os passos da secção 2.2.2. Os próximos passos são os seguintes:

1. Abrir o ficheiro main.py em Spyder. Este ficheiro é o ficheiro principal do processamento do algoritmo, está localizado em: PASTAS_DE_SCRIPTS\notebooks.
2. Re-definir as variáveis: PASTA_DE_SCRIPTS (ver Figura 15), PASTA_DE_INPUTS e subpastas IMAGENS_S2 e REGIAO_INTERESSE (ver Figura 16) e PASTA_DE_OUTPUTS (ver Figura 22).

```
from pathlib import Path
# Assumir onde está a pasta dos scripts do PyCCD
PASTA_DE_SCRIPTS = Path(__name__ ).parent.absolute() / 'S2CHANGE' / 'scripts' / 'pyccd_theia'

if PASTA_DE_SCRIPTS not in sys.path:
    sys.path.append(str(PASTA_DE_SCRIPTS))
```

Figura 15 – Localização da PASTA_DE_SCRIPTS no script main.py.

3. Colocar as imagens Sentinel-2 (GeoTIFFs) na subpasta `IMAGENS_S2` e o ficheiro geopackage ou shapefile que delimita a região de interesse na subpasta `REGIAO_INTERESSE`.
4. Executar o ficheiro `main.py` como explicado em 2.2.2.
5. Os outputs produzidos serão colocados automaticamente na `PASTA_DE_OUTPUTS`, como descrito na secção 8.

Note: na versão atual do código, em cada execução do código os conteúdos da `PASTA_DE_OUTPTUS` serão reescritos.

5. Inputs do PyCCD

Os inputs do algoritmo consistem em três variáveis essenciais:

1. **Coleção de imagens Sentinel-2:** as imagens estão no formato GeoTIFF, compostas por 10 bandas espectrais do sensor Sentinel-2 (na seguinte ordem: ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B8A', 'B11', 'B12']) e podem ser provenientes dos processamentos Theia/MAJA ou GEE/s2cloudless. Os pixels identificados pelo MAJA ou s2cloudless estão mascarados. As imagens do s2cloudless são obtidas diretamente do Google Earth Engine (GEE), através de um script Python executado no ambiente virtual `gee_env`, que foi configurado especificamente para esta finalidade (conforme descrito na secção 2.2.1). Caso seja necessário realizar o download das imagens, as instruções estão disponíveis na secção 9.
2. **Região de interesse:** trata-se de um ficheiro que pode estar no formato shapefile ou geopackage, contendo polígonos simples (singlepolygons) ou múltiplos (multipolygons). O algoritmo PyCCD processa os centros dos pixels localizados no interior dessas geometrias.

Caso seja necessário alterar os diretórios e/ou nomes de pastas, essa modificação deverá ser realizada no ficheiro `main.py`:

```
# -----
#                INPUTS
# -----
var = 'Theia' # choose variable: Theia or GEE
S2_tile = 'T29TNE' # escolher o tile S2

# Caminho onde estão os inputs todos
PASTA_DE_INPUTS = Path('C:/Users/Public/Documents/')
# -> Shapefile ou Geopackage que contem a regioao de interesse
REGIAO_INTERESSE = PASTA_DE_INPUTS / 'BDR_300_artigo' / 'BDR_CCDC_TNE_Adjusted.shp'

# -> IMAGENS SENTINEL:
IMAGENS_S2 = PASTA_DE_INPUTS / f'imagens_{str(var)}'

tiles = IMAGENS_S2 / S2_tile
```

Figura 16 – Localização dos inputs no script `main.py`.

6. Pré-processamento do PyCCD

Esta etapa do pré-processamento tem como objetivo preparar os dados necessários para a detecção de mudanças no PyCCD. Neste ponto, é necessário escolher duas variáveis principais:

1. **Período de processamento:** Determinado pelas datas inicial e final, que delimitam o intervalo temporal a ser analisado.
2. **Bandas utilizadas para o pré-processamento:** BLUE (B2), GREEN (B3), RED (B04), NIR (B8), SWIR2 (B12).

Se for necessário realizar ajustes, deverão ser feitos no ficheiro `main.py`:

```
# -----  
#   PARAMETROS PRÉ PROCESSAMENTO  
# -----  
min_year = 2017 # ano inicial da corrida do CCD  
max_date = datetime(2023, 12, 31) # data até onde se co  
bandas_desejadas = [1, 2, 3, 7, 10] # bandas usadas par
```

Figura 17 – Localização dos parâmetros de pré-processamento no script `main.py`.

A Tabela 3 apresenta as etapas de redução das bandas até chegarmos às bandas necessárias para o processamento no PyCCD.

Bandas Sentinel-2 dos ficheiros GeoTIFF	['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B8A', 'B11', 'B12']
Índices das bandas S2 no Python (variável <code>bandas_desejadas</code> na Figura 17)	['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
Bandas do S2 guardadas no ficheiro <code>numpy</code>	BLUE (B2), GREEN (B3), RED (B4), NIR (B8) e SWIR2 (B12)
Bandas usadas para calcular o NDVI	As bandas RED e NIR são utilizadas para calcular o Índice de Vegetação por Diferença Normalizada (NDVI), que é um dos parâmetros principais do algoritmo.
Substituição da banda BLUE	NDVI, GREEN (B3), RED (B4), NIR (B8) e SWIR2 (B12) Após o cálculo do NDVI, este substituirá a banda BLUE.
Bandas finais para o PyCCD	NDVI, GREEN, SWIR2

Tabela 3 – Etapas de redução das bandas Sentinel-2 para as bandas finais utilizadas no processamento PyCCD.

Após a definição das datas, o pré-processamento inicia com a leitura dos ficheiros TIFF e a criação de uma série temporal para os pixels contidos nas geometrias especificados pela região de interesse (RI). Apenas as cinco bandas selecionadas (BLUE, RED, GREEN, NIR e SWIR2) são utilizadas na construção desta série temporal.

O processo é realizado por meio da biblioteca `xarray`, que gera um ficheiro no formato `numpy`. Após esta etapa, é calculado o Índice de Vegetação por Diferença Normalizada (NDVI), utilizando as bandas RED e NIR. O cálculo do NDVI resulta na substituição da banda BLUE na série temporal. Para o processamento subsequente do PyCCD, são mantidas apenas as bandas NDVI, GREEN e SWIR2, pois estas bandas possuem propriedades espectrais que são particularmente sensíveis às mudanças na vegetação.

7. Processamento do PyCCD

Os parâmetros do algoritmo foram ajustados para otimizar o desempenho do modelo, conforme detalhado no Entregável 1.3, sendo os seguintes:

- 'PEEK_SIZE': 6
- 'MIN_YEARS': 1
- 'DETECTION_BANDS': NDVI, Green, SWIR2
- 'TMASK_BANDS': Green, SWIR2
- 'LASSO_MAX_ITER': 1000
- 'ALPHA': 20
- 'CHISQUAREPROB': 0.999

Caso seja necessário alterar os parâmetros do modelo deverá ser feito no script `parameters.py` contido no diretório: `PASTA_DE_SCRIPTS\ccd`.

8. Outputs do PyCCD

O nome final dos ficheiros de output gerados segue um formato específico, permitindo uma identificação clara dos parâmetros utilizados no processamento:

```
imagens_GEE-NDVI_XX999YM1NOBS6LDA20ITER1000_START20170408_END20231225_N51856_RS42
```

Este formato revela informações importantes sobre o processamento:

- **Coleção Sentinel-2:** O prefixo "GEE" indica que os dados foram extraídos do Google Earth Engine, enquanto "Theia" denotaria o uso de dados processados pela plataforma Theia.
- **Banda utilizada:** O termo "NDVI" especifica que o Índice de Vegetação por Diferença Normalizada foi a banda escolhida para identificar mudanças.
- **Parâmetros do modelo:** O número "20" representa o valor do parâmetro 'ALPHA', enquanto "ITER1000" representa o parâmetro 'LASSO_MAX_ITER'.
- **Datas de processamento:** "START20170408" e "END20231225" significam, respectivamente, a data de início e de fim da execução do modelo.
- **Número de pixels:** "N51856" representa o total de pixels analisados dentro de todas as geometrias da região de interesse.
- **Estado aleatório:** O "RS42" indica o estado aleatório utilizado, neste caso, um valor fixo de 42, que assegura a reprodutibilidade dos resultados.

É importante notar que o nome do ficheiro não contém o nome do tile, pois os ficheiros são armazenados em pastas específicas nomeadas de acordo com o tile correspondente:

Local Disk (C:) > Users > Public > Public Documents > output_BDR-NAV > shapefiles >		
Name	Date modified	Type
T29SNB	8/28/2024 7:09 PM	File folder
T29SNB(2)	8/29/2024 7:52 PM	File folder
T29TNE	9/2/2024 2:52 PM	File folder
T29TNF	9/3/2024 5:05 PM	File folder
T29TPE	9/1/2024 2:25 PM	File folder

Figura 18 – Estrutura de pastas de outputs por tiles.

Após o processamento do PyCCD, o algoritmo gera três tipos principais de output:

1. Plots: gráficos que apresentam as informações geradas pelo algoritmo CCD. Apresenta os segmentos ajustados aos dados bem como as datas de quebra entre segmentos. Estes gráficos são salvos na máquina local com o formato especificado no início desta secção e é complementado por '_RowIndex95.png', que indica o índice (linha) do pixel no ficheiro CSV. Para gerar o gráfico de um pixel específico, é necessário definir `EXECUTAR_PLOT = TRUE` no script `main.py` e indicar a linha correspondente ao pixel na variável `ROW_INDEX`.

```
EXECUTAR_PLOT = False # (false para não fazer; true para fazer)
ROW_INDEX = 8 # plot para uma linha do CSV (escolher a linha no row_index)
```

Figura 19 – Localização dos parâmetros para construir o plot no script `main.py`.

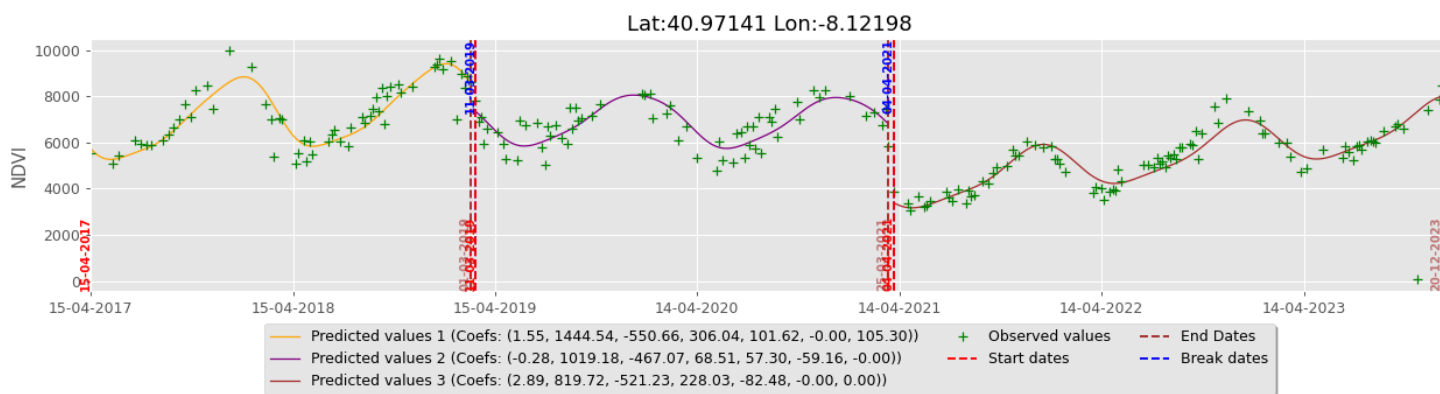


Figura 20 – Gráfico gerado pelo CCD indicando as alterações detetadas no pixel situado em latitude 40.97141 e longitude -8.12198.

2. Ficheiro no formato shapefile: armazena os resultados do CCD, permitindo a visualização e manipulação em softwares GIS como o QGIS. Este ficheiro é salvo na máquina local com o formato especificado anteriormente complementado por '.shp'. Cada linha do shapefile representa um ponto contido dentro de um polígono, e os atributos são os indicados abaixo. Note que cada evento é descrito por 3 datas que são o `tEnd`, `tBreak`, `tStart` nesta ordem cronológica.
 - o lat: latitude do pixel.
 - o lon: longitude do pixel.

Variáveis calculadas para cada segmento:

- tBreak: data de quebra identificada pelo CCD em milissegundos a partir de 1 de janeiro de 1970.
- tBreak_ddm: data de quebra identificada pelo CCD em formato %dd-%mm%yyyy (Break dates na Figura 20).
- tEnd: data de fim do segmento ajustado pelo CCD antes do tBreak, em milissegundos a partir de 1 de janeiro de 1970.
- tEnd_ddmmy: datas de fim dos segmentos ajustados pelo CCD em formato %dd-%mm%yyyy (End dates na Figura 20).
- tStart: data de início do segmento ajustado pelo CCD após tBreak, em milissegundos a partir de 1 de janeiro de 1970 (Start dates na).
- changeProb: confiança do algoritmo na deteção de uma quebra significativa nas séries temporais.
- coeficientes: coeficientes dos segmentos ajustados pelo CCD.
- ndvi_magni: diferença entre a magnitude do NDVI do último valor do segmento anterior (NDVI em tEnd) e o primeiro valor do segmento seguinte (NDVI em tStart). Quando não houver mais segmentos, a magnitude do NDVI terá o valor 65535, correspondendo a *nodata*.

Variáveis calculadas para as datas das imagens Sentinel-2:

- ndvis: valores da série temporal de NDVI multiplicado por 10000 (observed_values)
- dates: datas em milissegundos a contar desde o dia 1 de janeiro de 1970.
- mask: máscara utilizada para filtrar outliers dos dados. Se mask = False a observação é considerada um outlier pelo algoritmo, não sendo por isso considerada pelo CCD.

Variáveis calculadas diariamente (para a sequência completa de dias):

- prediction_dates: datas diárias em milissegundos a partir de 1 de janeiro de 1970
- predicted_values: série temporal dos valores de NDVI correspondentes a cada prediction_dates ajustados pelo CCD.

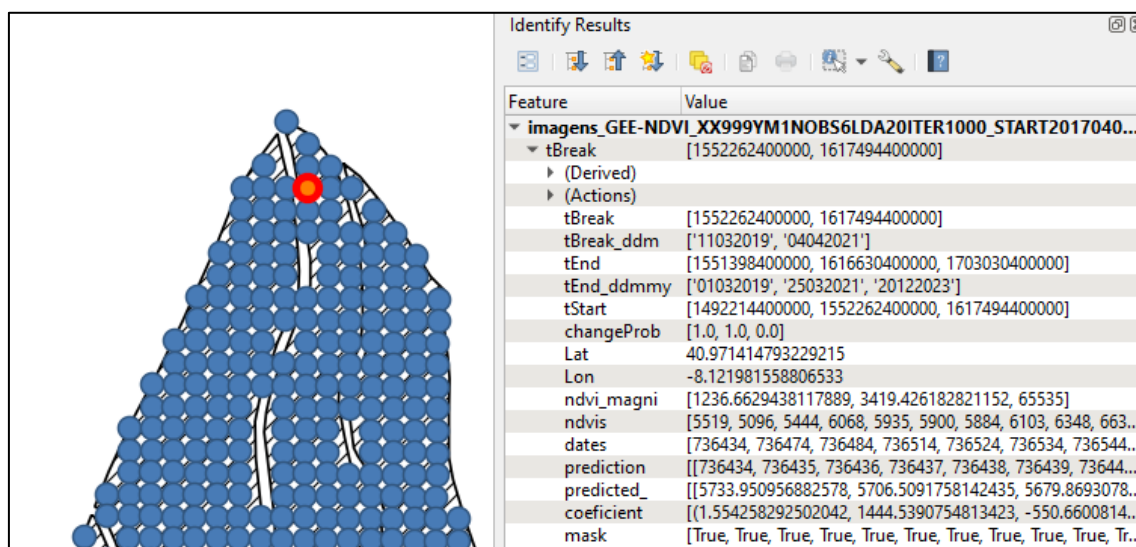
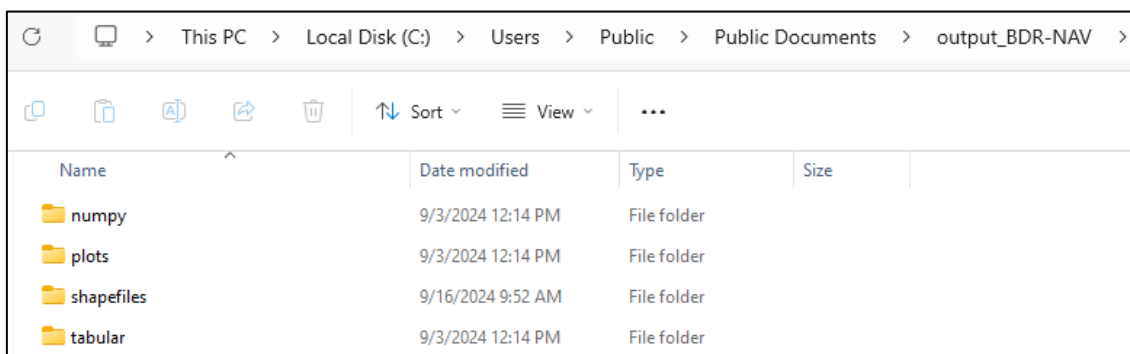


Figura 21 – Informação do ficheiro shapefile para o mesmo pixel representado na Figura 20.

3. Ficheiro CSV: contém a tabela de atributos do shapefile. O ficheiro tem extensão '.csv' em vez de '.shp'.

Caso seja necessário alterar os diretórios e/ou nomes de pastas, essa modificação deverá ser realizada no ficheiro `main.py`:

```
# -----  
#           OUTPUTS  
# -----  
PASTA_DE_OUTPUTS = Path('C:/Users/Public/Documents/outputs_RI')  
  
FOLDER_NPY = PASTA_DE_OUTPUTS / 'numpy' / S2_tile  
FOLDER_PLOTS = PASTA_DE_OUTPUTS / 'plots' / S2_tile  
FOLDER_CSV = PASTA_DE_OUTPUTS / 'tabular' / S2_tile  
FOLDER_SHP = PASTA_DE_OUTPUTS / 'shapefiles' / S2_tile  
  
# Função para criar diretórios se não existirem  
def create_directory_if_not_exists(path):  
    if not path.exists():  
        path.mkdir(parents=True, exist_ok=True)  
        print(f"Diretório criado: {path}")  
    else:  
        print(f"Diretório já existe: {path}")  
  
# Criar os diretórios  
create_directory_if_not_exists(FOLDER_NPY)  
create_directory_if_not_exists(FOLDER_PLOTS)  
create_directory_if_not_exists(FOLDER_CSV)  
create_directory_if_not_exists(FOLDER_SHP)
```



Name	Date modified	Type	Size
numpy	9/3/2024 12:14 PM	File folder	
plots	9/3/2024 12:14 PM	File folder	
shapefiles	9/16/2024 9:52 AM	File folder	
tabular	9/3/2024 12:14 PM	File folder	

Figura 22 – Localização dos outputs no script `main.py` (acima) e estrutura das pastas de outputs na máquina local (abaixo).

9. Download Imagens Sentinel-2 do GEE

9.1. Criação do Ambiente virtual para download de Imagens Sentinel-2 do GEE (`gee_env`)

Este ambiente será utilizado para fazer o download de imagens do Sentinel-2 a partir do Google Earth Engine (GEE). Os passos seguintes só precisam de ser realizados uma única vez para instalar o ambiente virtual:

1. No Anaconda Navigator, clique na aba “**Environments**” e no canto inferior esquerdo, clique em “**Create**”:

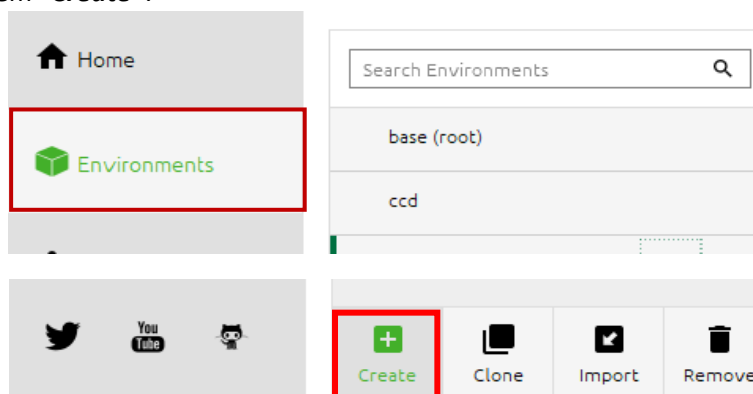


Figura 23 – Criação de ambientes virtuais na plataforma Anaconda.

2. Escolha um nome para o seu ambiente (por exemplo, gee_env). Selecionar a versão do Python necessária (neste caso, Python 3.8). Clique em create:

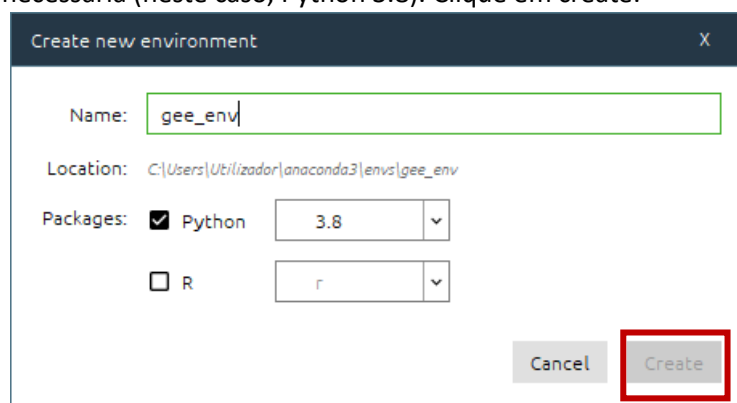


Figura 24 – Definição do nome do ambiente virtual a ser criado.

3. Selecione o ambiente gee_env criado.
4. Clique no botão “**Play**” e, de seguida, “**Open Terminal**”:

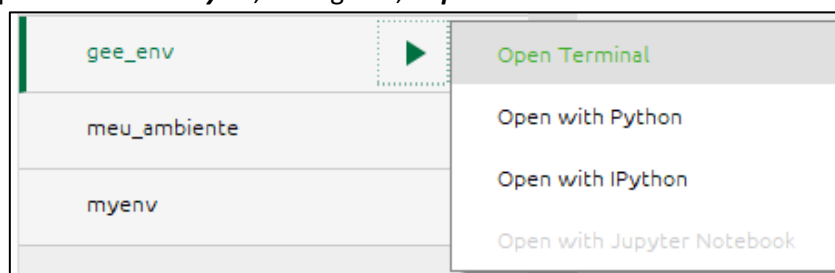


Figura 25 – Abrir um terminal de comandos dentro do ambiente virtual criado.

5. Verifique se está dentro do ambiente virtual gee_env, aparecendo este entre () no terminal:

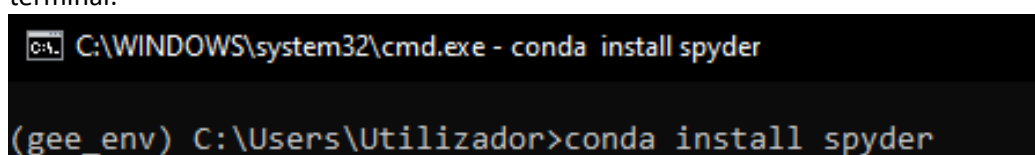


Figura 26 – Terminal de comandos dentro do ambiente virtual criado.

6. Para configurar o ambiente e garantir que todos os packages necessários para executar scripts do Google Earth Engine estejam instalados, siga os passos abaixo. (Note que durante a instalação, pode surgir a mensagem “Proceed ([y]/n)?”. Nesse caso, simplesmente pressione a tecla Y e tecla Enter para continuar):

```
conda install spyder
conda install matplotlib
pip install fiona==1.8.22
pip install geopandas==0.13.2
pip install earthengine-api
```

9.2. Configuração da Autenticação do Google Earth Engine em Spyder

Numa sessão de trabalho, após o ambiente virtual `gee_env` estar instalado é preciso fazer autenticação do Google Earth Engine com o Spyder. Os passos seguintes só precisam de ser realizados uma única vez para autenticar a conta do Google Earth Engine ao Spyder:

1. Para abrir o Spyder, aceda à seção “Home” do Anaconda Navigator, confirme que o ambiente ativo é o `gee_env`, e clique em “Launch” na aplicação do Spyder:

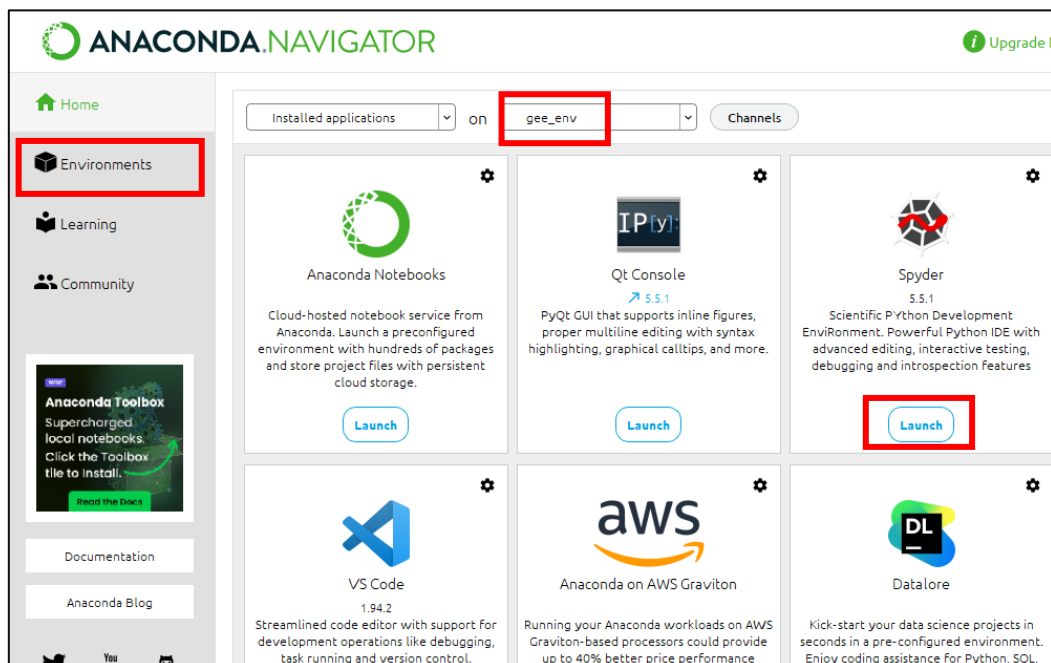


Figura 27 – Iniciar o Spyder no ambiente virtual `gee_env` via Anaconda Navigator.

2. Para autenticar ao Google Earth Engine, na consola de Spyder (localizada no canto inferior direito da interface) deve escrever as seguintes instruções: `import ee` e `ee.Authenticate(force=True)`

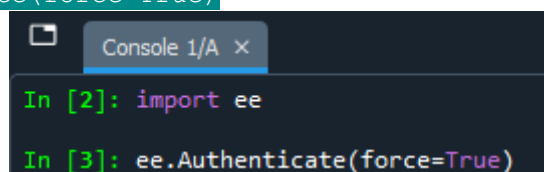


Figura 28 – Instrução que deverá ser escrita na consola do spyder.

3. Uma nova página web deverá surgir: `<IPython.core.display.HTML object>`

Figura 29 – Mensagem que deverá aparecer após executar o comando da Figura 28.

4. Gere o token de autenticação e copie-o. Em seguida, cole-o no terminal quando for solicitado o "**Enter verification code**":

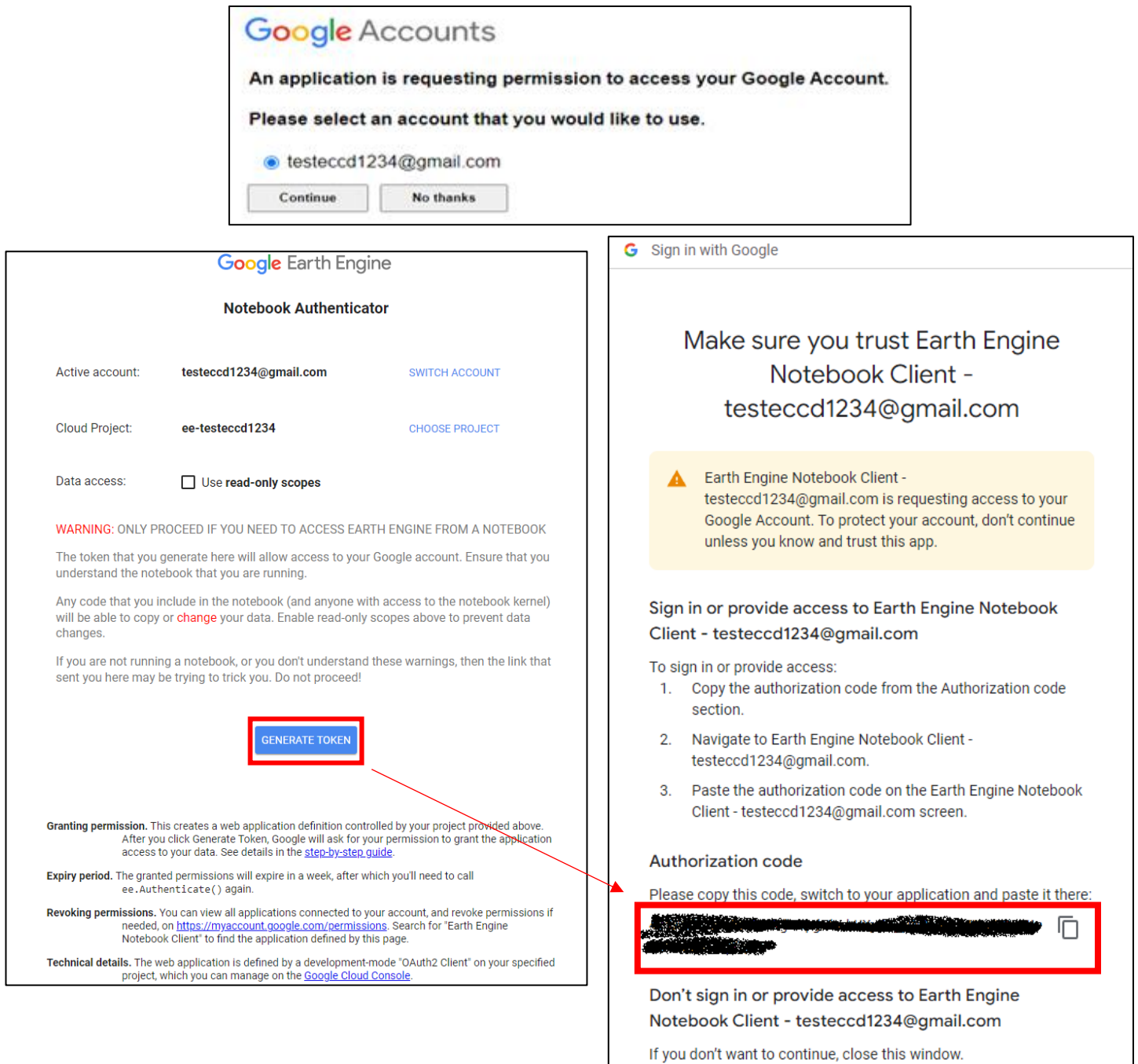
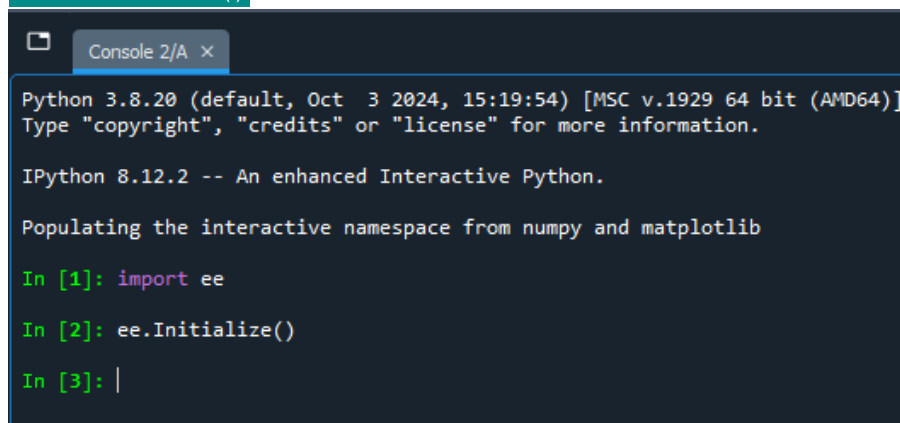


Figura 30 – Gerar o token de autenticação do GEE.



Figura 31 – Instrução da consola de Spyder onde deverá ser introduzido o token gerado na Figura 30. A mensagem da 2ª linha indica sucesso na autenticação.

5. Após validação do código deverá executar na mesma consola `import ee` e `ee.Initialize()` e não dar erro:



```
Python 3.8.20 (default, Oct 3 2024, 15:19:54) [MSC v.1929 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.12.2 -- An enhanced Interactive Python.

Populating the interactive namespace from numpy and matplotlib

In [1]: import ee
In [2]: ee.Initialize()
In [3]: |
```

Figura 32 – Validação da inicialização do Google Earth Engine (EE) no ambiente Python, sem erros após a execução dos comandos `import ee` e `ee.Initialize()`.

9.3. Imagens Sentinel-2 para processamento no CCD

Para iniciar o download das imagens Sentinel-2 do Google Earth Engine (GEE), o primeiro passo é abrir o Spyder dentro do ambiente virtual `gee_env`. Para isso, aceda à seção **"Home"** do Anaconda Navigator, certifique-se de que o ambiente ativo é o `gee_env` e clique em **"Launch"** na aplicação do Spyder (conforme mostrado na Figura 27).

Depois de abrir o Spyder, vá até **File > Open...** (cf. Figura 12), localize o diretório `PASTA_DE_SCRIPTS\notebooks` e abra o script denominado por `download-sentinel2-from-gee.py`. Este é o script Python responsável por executar o download das imagens Sentinel-2 diretamente do GEE. Para executar o script basta fazer **"Run"** ou carregar no botão **"Play"** (cf. Figura 13).

O script utiliza os seguintes inputs para a execução:

- `file_path_original`: caminho para o ficheiro geopackage que contém as geometrias de interesse (pode incluir multipolígonos ou polígonos simples).
- `file_path_tiles`: Caminho para o ficheiro shapefile que delimita os tiles Sentinel-2. Este ficheiro pode ser encontrado na pasta `PASTA_DE_SCRIPTS` após o download do repositório, com o nome `sentinel2_tiles_PT_terra_tm06.shp`.
- `file_path`: caminho onde será salvo o novo geopackage, agora com a coluna extra referente aos tiles.
- `tile_to_test`: nome do tile Sentinel-2 que deseja processar.
- `date_start`: data inicial.
- `date_end`: data final.
- `bandas`: bandas espectrais de interesse para o processamento.
- `tamanho_buffer`: tamanho do buffer que será aplicado em torno das geometrias.
- `cloud_filter`, `cld_prob_thresh`, `nir_drk_thresh`, `cld_prj_dist`: parâmetros do `s2cloudless`.

```

# -----
#           INPUTS
# -----
# Configuracoes e execucao principal
file_path_original = r'C:\Users\scaetano\Downloads\nvg_2018_ccd.gpkg'
file_path_tiles = r'C:\Users\scaetano\Downloads\sentinel2_tiles_PT_terra_tm06.shp'
file_path = r'C:\Users\scaetano\Downloads\nvg_2018_ccd_with_tiles.gpkg' # Nome dos

tile_to_test = 'T29TNF' # Escolher o tile
# Caso especial para o tile 'T29SNB', adicionar a variável de divisão
tile_to_split = 'T29SNB'
num_parts = 2 # número de divisões das geometrias
geometry_part = 'second' # 'first' ou 'second' para escolher a parte da geometria

date_start = '2023-08-01' # Escolher a data inicial para fazer o download das image
date_end = '2023-08-10' # Escolher a data final para fazer o download das imagens
bandas = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B8A', 'B11', 'B12'] # 10 banda
# bandas = ['ndvi', 'B2', 'B3', 'B4', 'B8']
tamanho_buffer = 500

NODATA = 65535

# Parâmetros S2cloudless
CLOUD_FILTER = 60
CLD_PRB_THRESH = 50
NIR_DRK_THRESH = 0.2
CLD_PRJ_DIST = 1
BUFFER = 50

```

Figura 33 – Inputs do script *download-sentinel2-from-gee.py*

Ao executar o script, os usuários devem começar por fornecer um geopackage que contenha as geometrias de interesse, bem como especificar as datas de início e fim do período a ser analisado. Além disso, é necessário definir os parâmetros do s2cloudless, o tamanho do buffer e as bandas desejadas para o processamento.

O script inicia processando o geopackage e, para cada geometria, adiciona uma coluna extra que indica o tile correspondente. É fundamental que os usuários escolham o tile específico que desejam processar, garantindo assim que as geometrias sejam corretamente selecionadas para o download.

O resultado final consiste num conjunto de ficheiros TIFF multibanda, com um ficheiro gerado para cada data dentro do intervalo especificado. A nomenclatura dos ficheiros TIFF segue o formato: `S2SR_image_1691235601301_tile_T29TNF`, onde o número "1691235601301" representa a data em milissegundos desde 1 de janeiro de 1970. Os ficheiros gerados serão salvos nas pastas dos tiles dentro da pasta pessoal do Google Drive. Note que é recomendável criar a pasta com o nome do tile que irá processar na sua pasta pessoal do Google Drive antes de executar o código, pois, em algumas situações, apenas executar o código pode não resultar na criação da pasta do tile.

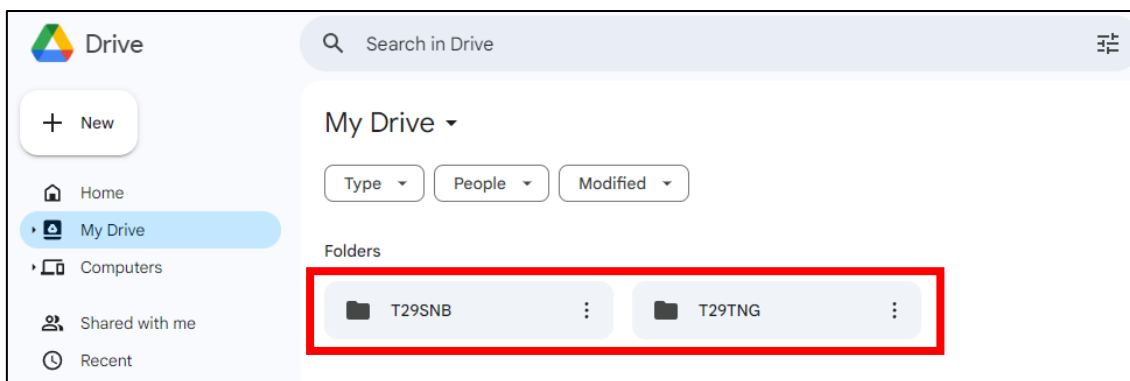


Figura 34 – Exemplos de pastas com os nomes dos tiles S2 na pasta pessoal do Google Drive.

Caso os tiles contenham um número elevado de geometrias, o script possui a funcionalidade de dividi-los automaticamente, facilitando o processamento. Nas experiências realizadas com o tile T29SNB, que inclui 250 geometrias multipoligonos referentes à nossa região de interesse e cobre uma área total de cerca de 44.587.948,63 m², essa divisão foi essencial. Sem esta divisão, o download simultâneo de todas as geometrias consumiria muito tempo e espaço em disco, tornando o processo menos eficiente. Para otimizar o desempenho, as geometrias foram divididas em duas partes, permitindo um download mais rápido e eficiente. Abaixo estão os parâmetros relevantes caso esta funcionalidade seja usada:

- `tile_to_test`: tile para o qual se quer processar a divisão das geometrias, neste caso T29SNB.
- `tile_to_split`: tile utilizado para comparar com `tile_to_test`, determinando se as geometrias devem ser divididas ao meio, neste exemplo apenas foi para o tile T29SNB que se fez a divisão.
- `num_parts`: número de partes em que se quer dividir as geometrias, neste caso, só está implementado para dividir em duas partes.
- `geometry_part`: permite selecionar se será processada a primeira ou a segunda metade das geometrias (aceita apenas 'first' ou 'second').

```
tile_to_test = 'T29SNB' # Escolher o tile
# Caso especial para o tile 'T29SNB', adicionar
tile_to_split = 'T29SNB'
num_parts = 2 # número de divisões das geometrias
geometry_part = 'second' # 'first' ou 'second'
```

Figura 35 – Parâmetros a escolher caso se opte pela divisão das geometrias.

Note que é aconselhável criar uma pasta com o nome do tile na sua área pessoal do Google Drive. Se estiver a processar a primeira parte das geometrias (ou seja, `geometry_part = first`), deve criar uma pasta com o nome do tile apenas. Caso esteja a processar a segunda parte das geometrias (`geometry_part = second`), a pasta deve ser nomeada com o nome do tile seguido de (2):

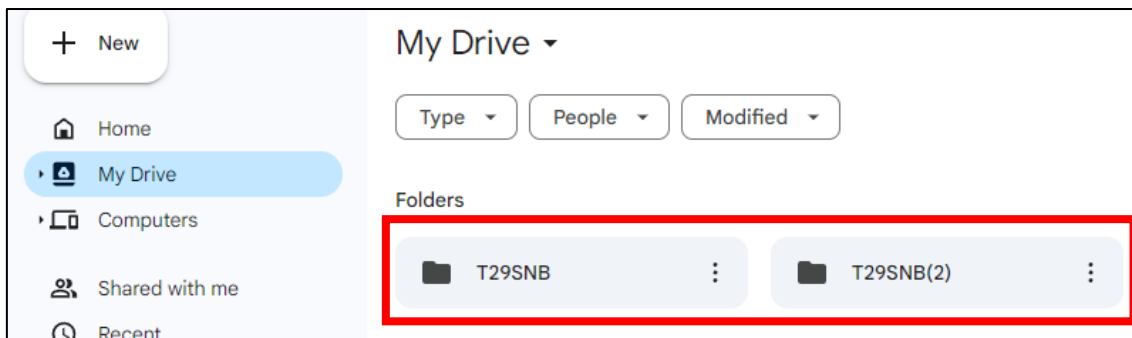


Figura 36 – Exemplos de pastas quando o script realiza a divisão automática das geometrias.

9.4. Imagens Sentinel-2 para análise visual

O script `download-sentinel2-from-gee.py` oferece a possibilidade de gerar imagens Sentinel-2 especificamente para análise visual, utilizando as bandas NDVI, B2, B3, B4 e a B8:

```
# bandas = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B8A', 'B11', 'B12']
bandas = ['ndvi', 'B2', 'B3', 'B4', 'B8']
```

Figura 37 – Inclusão da banda NDVI às bandas requeridas para o download de imagens S2 via GEE.

Durante este processo, a banda NDVI passa por uma transformação de escala para aumentar o contraste nas imagens de NDVI, seguindo as seguintes regras:

- Se $NDVI \geq 5000$, então $NDVI = 5000$;
- Se $NDVI < 0$, então $NDVI = 0$;
- Se $0 \leq NDVI < 5000$, o valor original é mantido.

A Figura 38 mostra a parte do script `download-sentinel2-from-gee.py` onde a transformação de escala do NDVI mencionada acima é implementada. No entanto, se não quiser esta transformação basta usar a função representada a vermelho na figura referida.


```

# -----
# SCRIPT PARA DOWNLOAD DAS IMAGENS SENTINEL-2 DIRETAMENTE DO GEE
# -----

# Inicializar o Earth Engine
ee.Initialize()

# -----
#      Funcoes Auxiliares
# -----

# # Função para adicionar banda NDVI s/ transformacao da escala
# def addNDVI(image):
#     ndvi = image.normalizedDifference(['B8', 'B4']).multiply(10000).int16()
#     return image.addBands(ndvi.rename('ndvi'))

# Funcao para adicionar banda NDVI c/ transformacao da escala
def addNDVI(image):
    # Calcula o NDVI
    ndvi = image.normalizedDifference(['B8', 'B4']).multiply(10000).int16()

    # Aplicação da transformação:
    # Se NDVI >= 5000 -> NDVI = 5000
    # Se NDVI < 0 -> NDVI = 0
    # Se 0 <= NDVI < 5000 -> mantém o valor original
    ndvi_clipped = ndvi.expression(
        '((ndvi < 0) ? 0 : (ndvi >= 5000 ? 5000 : ndvi))',
        {'ndvi': ndvi}
    ).int16()

    return image.addBands(ndvi_clipped.rename('ndvi'))

```

Figura 38 – Implementação da transformação de escala do NDVI no script `download-sentinel2-from-gee.py`, com a opção para desativar a transformação, utilizando a função destacada a vermelho.