Creating the Datasets

This notebook will show how we created the datasets used in the CNN.

First we import the libraries, which includes some different satelites, standard pythin libraries and google drive api clients.

In [42]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import ee
import urllib.request
import datetime
import os
import base64
import requests
import warnings
warnings.filterwarnings("ignore")
import tempfile
from google.oauth2 import service_account
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
from googleapiclient.http import MediaFileUpload
from sentinelhub import SentinelHubRequest, MimeType, CRS, BBox, DataCollection
from sentinelhub import SHConfig
from sentinelhub import WmsRequest, CRS, MimeType, CustomUrlParam
from PIL import Image
import random
from math import radians, sin, cos, sqrt, atan2
from sklearn.neighbors import DistanceMetric
from scipy.spatial.distance import cdist
```

Getting the Satellite Images

To find the images of places where a fire will occur, so we have a chance to see if we can predict it, we have found a dataset of fires occuring in the state in Oregon in the US from 2000-2022. This includes the time of detection, exact location, size, and what caused the fire, plus many other details.

In [2]:
```
full_data = pd.read_csv('ODF_Fire_Occurrence_Data_2000-2022.csv')
```

In [3]:
```
full_data.T
```

Out[3]:

	0	1	2	3	4	5	6
Serial	102649	131239	58256	59312	61657	98529	63735
FireCategory	STAT	STAT	STAT	STAT	STAT	STAT	STAT
FireYear	2015	2022	2000	2000	2001	2014	2002
Area	EOA	EOA	EOA	EOA	SOA	SOA	NOA
DistrictName	Klamath-Lake	Walker Range - WRFPA	Central Oregon	Northeast Oregon	Southwest Oregon	Douglas - DFPA	West Oregon
UnitName	Klamath	Crescent	John Day	La Grande	Grants Pass	DFPA Central	Philomath

FullFireNumber	15-981082-16	22-991220-23	00-952011-01	00-971024-01	01-712133-02	14-733192-15	02-551001-03
FireName	Bass 497	Hay Fire	Slick Ear #2	Woodley	QUEENS BRANCH	Chilcoot	WREN
Size_class	B	A	B	C	A	A	A
EstTotalAcres	3.2	NaN	0.75	80.0	0.1	0.01	0.01
Protected_Acres	3.2	0.2	0.75	80.0	0.1	0.01	0.01
HumanOrLightning	Human	Human	Lightning	Lightning	Human	Lightning	Human
CauseBy	Other-Public	NaN	Lightning	Lightning	Motorist	Lightning	Motorist
GeneralCause	Under Invest	Miscellaneous	Lightning	Lightning	Smoking	Lightning	Recreation
SpecificCause	NaN	NaN	Lightning	Lightning	Other - Smoker Related	Lightning	Fireworks
Cause_Comments	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Lat_DD	42.13361	43.59358	44.91519	45.08509	42.53671	43.45583	44.58709
Long_DD	-122.04083	-121.49422	-119.28863	-118.3344	-123.21215	-122.74889	-123.42779
LatLongDD	POINT (-122.04083 42.13361)	POINT (-121.49422 43.59358)	POINT (-119.28863 44.91519)	POINT (-118.3344 45.08509)	POINT (-123.21215 42.53671)	POINT (-122.74889 43.45583)	POINT (-123.42779 44.58709)
FO_LandOwnType	Industrial	Other Public	BLM	Other Private	BLM	BLM	State
Twn	39S	23S	07S	05S	35S	24S	11S
Rng	7E	10E	29E	36E	04W	1W	06W
Sec	31.0	11.0	31.0	32.0	7.0	25.0	28.0
Subdiv	NWSE	NWSE	NESW	NESW	SESE	NWSE	SENW
LandmarkLocation	Topsy Area	Hwy 31	11 MI SE Ritter LO	Woodley C.G	7 N ROGUE RIVER	NaN	Kings Valley
County	Klamath	Klamath	Grant	Union	Jackson	Douglas	Benton
RegUseZone	KF1	WC2	EC2	NE3	SW3	DG1	W01
RegUseRestriction	Reg Use Closure	NaN	Reg Use Closure	Reg Use Closure	Reg Use Closure	Reg Use Closure	Closed Fire Season Lvl 1
Industrial_Restriction	Does Not Apply - Eastern OR	NaN	Does Not Apply - Eastern OR	Does Not Apply - Eastern OR	Lvl 3 Restricted Shutdown	Lvl 3 Restricted Shutdown	Lvl 1 Fire Season Only
Ign_DateTime	09/02/2015 05:00:00 PM	NaN	07/18/2000 07:00:00 PM	08/24/2000 05:30:00 AM	08/10/2001 05:40:00 PM	08/12/2014 06:15:00 PM	07/06/2002 01:01:00 PM
ReportDateTime	09/02/2015 05:05:00 PM	08/16/2022 06:56:00 PM	07/19/2000 01:20:00 PM	08/24/2000 01:07:00 PM	08/10/2001 05:47:00 PM	08/13/2014 04:01:00 PM	07/06/2002 01:04:00 PM

Discover_DateTime	09/02/2015 05:00:00 PM	NaN	07/19/2000 01:15:00 PM	08/24/2000 01:07:00 PM	08/10/2001 05:45:00 PM	08/13/2014 04:00:00 PM	07/06/2002 01:02:00 PM
Control_DateTime	09/02/2015 11:00:00 PM	NaN	07/20/2000 12:50:00 AM	09/01/2000 09:30:00 PM	08/10/2001 06:30:00 PM	08/14/2014 06:30:00 PM	07/06/2002 01:07:00 PM
CreationDate	09/05/2015 12:00:00 AM	08/18/2022 12:00:00 AM	07/20/2000 09:13:00 AM	08/29/2000 03:59:00 PM	08/10/2001 06:42:00 PM	08/21/2014 12:00:00 AM	07/07/2002 09:16:00 AM
ModifiedDate	10/13/2015 08:39:00 AM	08/18/2022 09:11:00 AM	11/14/2000 09:16:00 AM	12/21/2000 04:22:00 PM	08/17/2001 11:45:00 AM	08/24/2014 11:54:00 AM	07/28/2002 10:08:00 AM
DistrictCode	98	99	95	97	71	73	55
UnitCode	981	991	952	971	712	733	551
DistFireNumber	082	220	011	024	133	192	001

38 rows × 23490 columns

Here we collect some of the columns that may be relevant for our analysis

In [3]:
```
# Collecting relevant columns
data = full_data[['FireYear','Size_class','FullFireNumber','EstTotalAcres','CauseBy','La
```

Cleaning the data by dropping NA rows, converting to datetime for the satelite scraping and creating the coordinates form the latitude and longitude columns

In [4]:
```
#Cleaning the data
data = data.dropna() # Drop rows with no discover time
data['Discover_DateTime'] = pd.to_datetime(data['Discover_DateTime']) # Convert to datet
data['Control_DateTime'] = pd.to_datetime(data['Control_DateTime']) # Convert to datetim
data['ReportDateTime'] = pd.to_datetime(data['ReportDateTime']) # Convert to datetime
data['Coordinates'] = data.apply(lambda row: [row['Lat_DD'], row['Long_DD']], axis=1) #
```

Creating columns 1, 3, and 6 months before the fire was detected, to use for different intervals for the scraping

In [5]:
```
# Getting the dates for the images
data['6months'] = data['Discover_DateTime'] - pd.DateOffset(months=6)
data['3months'] = data['Discover_DateTime'] - pd.DateOffset(months=3)
data['1months'] = data['Discover_DateTime'] - pd.DateOffset(months=1)
```

Getting the images

Here we are using the Earth Engine to scrape the images

Use this to get access to the Earth Engine: https://developers.google.com/earth-engine/guides/python_install

Username: forestfiresgroupglobaldtu

In [9]:
```
# Authenticate Earth Engine
ee.Authenticate()
```

To authorize access needed by Earth Engine, open the following URL in a web browser and follow the instructior

https://code.earthengine.google.com/client-auth?
scopes=https%3A//www.googleapis.com/auth/earthengine%20https%3A//www.googleapis.com/auth/devstorage.
6F7loXHE&tc=sG2XcqEKhulWSFxZyGYaYWle5eE7I_oi9QAQUnO2LFM&cc=k2T9UIrUQI1_MYdaUj5IBg6uVAR

The authorization workflow will generate a code, which you should paste in the box below.

```
Enter verification code: 4/1AVHEtk5CrvfHqz949qbW8XO205OoPj4i4750n0rUacxjkCpO6iGskFDtA4g

Successfully saved authorization token.
```

Earth Engine

In [10]:
```
# initialize Earth Engine
ee.Initialize()
```

Google Drive Save

All the images was directly saved to google drive with the google api client, to save space on the local computer.

The first satelite we tried was the Landsat 8, which has a lot of data available, but after scarping the images we decided to not use it as the quality was to low when trying to get images that was under 2km x 2km.

Landsat 8

Landsat 8, launched in February 2013, is an Earth observation satellite operated by the United States Geological Survey (USGS) and NASA. Landsat 8 operates in a sun-synchronous orbit, capturing images of the Earth's surface in a continuous and systematic manner.

Landsat 8 has a 16-day revisit time for any specific location on Earth. This means that Landsat 8 captures images of the same location approximately once every 16 days. The satellite acquires images in multiple spectral bands, including visible, near-infrared, and thermal infrared bands.

It's worth noting that Landsat 8 works in tandem with Landsat 9, which was launched in September 2021. Both satellites share the same orbital plane and follow the same ground track, with Landsat 9 offset by 8 days from Landsat 8. This effectively increases the revisit time for the Landsat program to approximately once every 8 days for any specific location on Earth.

In [20]:
```
# Function to get the image URL
def get_image_url(coordinates, timestamp):
    latitude, longitude = coordinates
    # convert the timestamp to a string in the format 'YYYY-MM-DD'
    date_str = timestamp.strftime('%Y-%m-%d')

    # convert the string to a datetime object
    date = datetime.datetime.strptime(date_str, '%Y-%m-%d')

    # Define the image collection and filter
    image_collection = ee.ImageCollection("LANDSAT/LC08/C01/T1_SR") \
        .filterDate(date - datetime.timedelta(days=30), date + datetime.timedelta(days=3
        .filterBounds(ee.Geometry.Point(longitude, latitude))

    # Check if there are any images within the 30-day window
    image_count = image_collection.size().getInfo()
    if image_count == 0:
        #print(f'No images found within 30 days of {timestamp} for coordinates {coordina
```

```
        return

    # Get the least cloudy image
    least_cloudy = image_collection.sort('CLOUD_COVER').first()

    # Define the visualization parameters
    vis_params = {
        'bands': ['B4', 'B3', 'B2'],
        'min': 0,
        'max': 3000,
        'gamma': 1.4
    }

    # Define the region to get the image
    region = ee.Geometry.Point(longitude, latitude).buffer(1000).bounds().getInfo()['coo

    # Get the image URL
    image_url = least_cloudy.getThumbURL({
        'region': region,
        'scale': 12,  # Set the scale to match the native resolution
        'format': 'png',
        'resampling_method': 'bicubic',
        **vis_params
    })

    return image_url

# Function to save the image to Google Drive
def save_image_to_drive(image_url, file_name, folder_id=None):
    # Set up Google Drive API
    creds = service_account.Credentials.from_service_account_file('forestfiredtu-ef1cef2
    service = build('drive', 'v3', credentials=creds)

    # Download the image
    response = requests.get(image_url)
    if response.status_code != 200:
        print(f"Failed to download image: {file_name}.png")
        return
    image_data = response.content

    # Save the image to a temporary file
    with tempfile.NamedTemporaryFile(suffix='.png', delete=False) as temp_image:
        temp_image.write(image_data)
        temp_image.flush()

        # Save the image to Google Drive
        file_metadata = {
            'name': f'{file_name}.png',
            'mimeType': 'image/png'
        }

        if folder_id:
            file_metadata['parents'] = [folder_id]

        media = MediaFileUpload(temp_image.name, mimetype='image/png', resumable=True)

        try:
            file = service.files().create(body=file_metadata, media_body=media,
                                          fields='id').execute()
            print(F'File ID: "{file.get("id")}".')
        except HttpError as error:
            print(F'An error occurred: {error}')
            file = None

        # Remove the temporary file
        os.unlink(temp_image.name)
```

```
    return file
# Specify your Google Drive folder ID
folder_id = '1As3zsmcQIIGIAwTdwgtPIcLQuLCclXsZ'

# Iterate over the DataFrame and call the functions
for index, row in df.iterrows():
    image_url = get_image_url(row['Coordinates'], row['6months'])
    if image_url:
        save_image_to_drive(image_url, row['FireName'], folder_id)
```

```
File ID: "1D_0XphtFY0jqAX5DEIyRWjP6pTxAUnA3".
File ID: "1wT2M-mDI7s3zRrTyHACLZQvrGdH2hSUG".
File ID: "1iWIZz_02_Jir7hoIpE4KAItqFC_YtOc_".
File ID: "1v1gvPIu8f8QGLhg_9Y_G490qo_HNsJvv".
File ID: "1VUX9paJhbWGYsIPm4lhinefqJ-GLWOEE".
File ID: "18S8sPyNPIxoBtKdDKxAe0KYOO6QPpZNX".
File ID: "1tTVldD5M6a1l0WFQJ99RrhkwT5OXbJpn".
File ID: "1hD86j5kD93QOj3yfd1D-0xdD7Oz-GtXu".
File ID: "1QBOdWyh-PMP5HiqGIQ58s_T2DDymVxG1".
File ID: "1oPhWrwfX-IQG6JDYGQ9UNCN42-BfPyiq".
File ID: "1d6Ng8cyRPAW8yKD7qUPPSyA8-14wvD6M".
File ID: "1rS2RZbFbaaUfBLiveIeEFBFolWHrBqJz".
File ID: "1E2vkNR8m55zSlzEQKp9JuNX65FW2Wwgc".
File ID: "1zhmPlei-8pACmcfEzrWnuTFCyxSwC9Nt".
File ID: "1RIM5nGkslI6WWoy3Zj7bhE9pjLGPgLsh".
File ID: "1bN7tEzwBzXCNA4oDDvyLScXYIgtcBgMl".
File ID: "14oqGf140uCtsryinf0vC17Ck7aLKV8Sc".
File ID: "1C9DWz38v2fZk3S5UL7-6ZOu-wyV-WVHG".
File ID: "15Wvgy3fR9iDWbDuhUB7vxDFk0lgq2SPR".
File ID: "1Uo3GNJcyimiZosuOL7ajtzfRW6fJ3TGr".
File ID: "1Z0WgUz0ZaAYHdq0-sOx-8k7EI2rEeWkv".
File ID: "1S29-bRlbYmWaP-ztnOGPPrMR6DKNDysD".
File ID: "1OC1ZoOppXw4akQjNCEWj_LMoRpGVi7i5".
File ID: "1XjBXAnf0mafhZCeSPaBzEENmukPBil1_".
File ID: "1ZWuqXWEaTk9cZkLWB_uIe1ywhoRy0w3i".
File ID: "1QcBMPaztOSwbyJj21MqMm1kIA4w9trJK".
File ID: "1Wj39Y0II58ErYCx3zpnr3i3ze-mTue7l".
File ID: "12c4iw4urlAM0gLI1ki1sbbbq7Vzmvy98".
File ID: "1UAX_guuJqMz0yCHdN-1yA3kzlyC0X3op".
File ID: "1cAlTPKpfU-AIy67ZsGzj4yU_70mRpchp".
File ID: "1qxLO_5BmOjDfYApfOXpUX-gF_E1qtYHH".
File ID: "1b8awHAT-BAUXUjdlXZoGLctuhqLpSIv8".
```

The next satelite we tried was the Sentinel 2, which provided better pictures, but it could still not produce images of a good enough quality with images smaller than 1km x 1km.

Sentinel 2 - only data from 2015

In [30]:

```
# Function to get the image URL
def get_image_url(coordinates, timestamp):
    latitude, longitude = coordinates
    # convert the timestamp to a string in the format 'YYYY-MM-DD'
    date_str = timestamp.strftime('%Y-%m-%d')

    # convert the string to a datetime object
    date = datetime.datetime.strptime(date_str, '%Y-%m-%d')

    # Define the image collection and filter
    image_collection = ee.ImageCollection("COPERNICUS/S2") \
        .filterDate(date - datetime.timedelta(days=30), date + datetime.timedelta(days=3
        .filterBounds(ee.Geometry.Point(longitude, latitude)) \
        .filterMetadata('CLOUDY_PIXEL_PERCENTAGE', 'less_than', 5)
    # Check if there are any images within the 30-day window
```

```
    image_count = image_collection.size().getInfo()
    if image_count == 0:
        #print(f'No images found within 30 days of {timestamp} for coordinates {coordina
        return

    # Get the least cloudy image
    least_cloudy = image_collection.sort('CLOUDY_PIXEL_PERCENTAGE').first()

    # Define the visualization parameters
    vis_params = {
        'bands': ['B4', 'B3', 'B2'],
        'min': 0,
        'max': 3000,
        'gamma': 1.4
    }

    # Define the region to get the image
    region = ee.Geometry.Point(longitude, latitude).buffer(750).bounds().getInfo()['coor

    # Get the image URL
    image_url = least_cloudy.getThumbURL({
        'region': region,
        'scale': 10,  # Set the scale to match the native resolution
        'format': 'png',
        'resampling_method': 'bicubic',
        **vis_params
    })

    return image_url

# Function to save the image to Google Drive
def save_image_to_drive(image_url, file_name, folder_id=None):
    # Set up Google Drive API
    creds = service_account.Credentials.from_service_account_file('forestfiredtu-ef1cef2
    service = build('drive', 'v3', credentials=creds)

    # Download the image
    response = requests.get(image_url)
    if response.status_code != 200:
        print(f"Failed to download image: {file_name}.png")
        return
    image_data = response.content

    # Save the image to a temporary file
    with tempfile.NamedTemporaryFile(suffix='.png', delete=False) as temp_image:
        temp_image.write(image_data)
        temp_image.flush()

        # Save the image to Google Drive
        file_metadata = {
            'name': f'{file_name}.png',
            'mimeType': 'image/png'
        }

        if folder_id:
            file_metadata['parents'] = [folder_id]

        media = MediaFileUpload(temp_image.name, mimetype='image/png', resumable=True)

        try:
            file = service.files().create(body=file_metadata, media_body=media,
                                          fields='id').execute()
            print(F'File ID: "{file.get("id")}".')
        except HttpError as error:
            print(F'An error occurred: {error}')
            file = None
```

```
        # Remove the temporary file
        os.unlink(temp_image.name)

    return file
# Specify your Google Drive folder ID
folder_id = '13Yklp_2PlRh-HV1dmOMYA77L4fARfuIE'

# Iterate over the DataFrame and call the functions
for index, row in test.iterrows():
    image_url = get_image_url(row['Coordinates'], row['3months'])
    if image_url:
        save_image_to_drive(image_url, row['FireName'], folder_id)
```

```
File ID: "1a2fKCzTze1qxQXD_fpuECrF7FxZgCSLO".
```

The satelite that we ended up using was the NAIP which can provide high quality images, and we decided on images with 250m x 250m. The only disadvantage with NAIP is that it is only updated ever 2-3 years so we could not get as many pictures as we had hoped. The scraped images are between 2-7 months before the fire occured

NAIP

The National Agriculture Imagery Program (NAIP) imagery is typically updated on a two to three-year cycle. This means that for a specific area within the United States, new NAIP images are usually acquired every two or three years. However, this frequency may vary depending on factors like budget, weather conditions, and other factors that can impact aerial image acquisition.

NAIP primarily focuses on capturing imagery during the agricultural growing season (late spring through early fall) to support various agricultural programs and applications. The images have a resolution of 1 meter, and they are available in natural color (RGB) and, in some cases, near-infrared (NIR) bands.

In [24]:
```
#Split the data, as the scrape usually stops after 4000-5000 rows
data1 = data[:4000]
data2 = data[4000:8000]
data3 = data[8000:12000]
data4 = data[12000:16000]
data5 = data[16000:20000]
data6 = data[20000:]
```

In [19]:
```
# Function to get the image URL
def get_image_url(coordinates, timestamp):
    latitude, longitude = coordinates
    # convert the timestamp to a string in the format 'YYYY-MM-DD'
    date_str = timestamp.strftime('%Y-%m-%d')

    # convert the string to a datetime object
    date = datetime.datetime.strptime(date_str, '%Y-%m-%d')

    # Define the image collection and filter
    image_collection = ee.ImageCollection("USDA/NAIP/DOQQ") \
        .filterDate(date - datetime.timedelta(days=120), date + datetime.timedelta(days=
        .filterBounds(ee.Geometry.Point(longitude, latitude))

    # Check if there are any images within the 1-year window
    image_count = image_collection.size().getInfo()
    if image_count == 0:
        #print(f'No images found within 1 year of {timestamp} for coordinates {coordinat
        return

    # Get the most recent image
```

```
    most_recent = image_collection.sort('system:time_start', False).first()

    # Define the visualization parameters
    vis_params = {
        'bands': ['R', 'G', 'B'],
        'min': 0,
        'max': 255,
    }

    # Define the region to get the image
    region = ee.Geometry.Point(longitude, latitude).buffer(250).bounds().getInfo()['coor

    # Get the image URL
    image_url = most_recent.getThumbURL({
        'region': region,
        'scale': 1,  # Set the scale to match the native resolution
        'format': 'png',
        'resampling_method': 'bicubic',
        **vis_params
    })

    return image_url

# Function to save the image to Google Drive
def save_image_to_drive(image_url, file_name, folder_id=None):
    # Set up Google Drive API
    creds = service_account.Credentials.from_service_account_file('forestfiredtu-ef1cef2
    service = build('drive', 'v3', credentials=creds)

    # Download the image
    response = requests.get(image_url)
    if response.status_code != 200:
        print(f"Failed to download image: {file_name}.png")
        return
    image_data = response.content

    # Save the image to a temporary file
    with tempfile.NamedTemporaryFile(suffix='.png', delete=False) as temp_image:
        temp_image.write(image_data)
        temp_image.flush()

        # Save the image to Google Drive
        file_metadata = {
            'name': f'{file_name}.png',
            'mimeType': 'image/png'
        }

        if folder_id:
            file_metadata['parents'] = [folder_id]

        media = MediaFileUpload(temp_image.name, mimetype='image/png', resumable=True)

        try:
            file = service.files().create(body=file_metadata, media_body=media,
                                          fields='id').execute()
            print(F'File ID: "{file.get("id")}".')
        except HttpError as error:
            print(F'An error occurred: {error}')
            file = None

        # Remove the temporary file
        os.unlink(temp_image.name)

    return file
# Specify your Google Drive folder ID
folder_id = '1ats_A6B5WJx2Gdkd426zOpVUSL6N_LEz'
```

```
j = 0
# Iterate over the DataFrame and call the functions
for index, row in data6.iterrows():
    image_url = get_image_url(row['Coordinates'], row['3months'])
    j += 1
    if j % 100 == 0:
        print(j)
    if image_url:
        save_image_to_drive(image_url, row['FullFireNumber'], folder_id)
```

```
File ID: "1qF3pMZr84DVCR_mm4obsEXwHuYiovdJP".
File ID: "14PE3z2Hi04sm5lre7v9i7oCv3Ni9Mvzg".
File ID: "1XyKdwpG0bYsTEK3mFKlpfepCwB_cC5x5".
File ID: "1F9Ya65s6ADBbEORz6s3kDjVpdt0qWfrm".
File ID: "1SQjla7YHxCZUublJ3qB9OCcfVeIK2MmT".
File ID: "1a1qiTOx2f52sJ-3dC7bE91WnvMiT1QFq".
File ID: "1y7rgsbU1HpNiyymm6MsMkg6vGffYr7aR".
File ID: "1m7k_AEnTrYcuxvDsvVWctyqo594lfzn4".
File ID: "1rAfZudhUNs45P5_X9sNKfN7cWWY5YEdV".
File ID: "10t0Dtfi64nBhdVAnzteU3Omi7EwDf18I".
File ID: "1f-teEgL5do1bsu7yGolF9_VcU6a0-7M5".
File ID: "1WQG9rp-NCC0OXK0vJC9Om2wfpP9x7FfH".
100
File ID: "15rMUwc7OSHgscrRpe_ILmUHAGTYBScJI".
File ID: "10230sm_Q1q6-W16TR-FFLxWJnmULgr78".
File ID: "1KSp1gOzt2LXg6rsRj1nTFQ4LeLfTLvFI".
File ID: "12JTgbSqBrHxquIY8a9l81fdIOB48ugTO".
File ID: "1I5-6J6Ivb1MVTmz6mYP8U43C_f0cItsn".
File ID: "1Hduz4JEjxzWxKK-rHtiLMe0kGM6pbXVE".
File ID: "1b8F9ldx60-A1cfSkjHJkRm3WYqQIZkmZ".
File ID: "1Ui2AS-M6WfTMdIH7F-eKPqWxLD_T5kTa".
File ID: "1ZLXmzJpHYHKOa_SFJwKruisBSSmppYKf".
File ID: "1BPkvMCTTPY5ruoTLJToolcSRaexfly-V".
200
File ID: "1MzPu7UtXHuZF4GdnF7HnbAXdUL6m43gf".
File ID: "17I_TMnkMzV8FqHUtHv_hyzOldQePnr16".
File ID: "1jrLZjsuCYmZgimX65pvipuKCqJi0jUKE".
File ID: "1nzkX72FbynYOjtHL73VppreOIb8T-j-e".
File ID: "1Zvm2l0tzegcOrWKZmGteAfB-_tYH_3wi".
File ID: "1GaZltA3YkBOiuMCNustLud-WUSgl1bsp".
File ID: "1f_VdATxtYMm2MZr9hqtLYIc_6eyuLyLd".
File ID: "1g3CjDSip5Fajkc_PhcQgdSel5wObAgQs".
File ID: "1lHQruR4N936gRLwedduD0gj4mnnzUUXO".
File ID: "1iNaLlHkzCS_4QMOFX59Lh6dAl8Ntw-AN".
300
File ID: "1nseSzWIjWcY7njb9YO6HLW0o-3Ho4aLj".
File ID: "1w_bB8fV2zZ-BEF3VH358RLVUmY45Siur".
File ID: "1MGpMSxeX6YIqPffiLv3KSt1yO1VIvU6N".
File ID: "1RrylLyERtfF4_xQ-BER3yrdiUQpd4NFs".
File ID: "152MVynVfB8FTmBMGV2_bnTYuvqOP9hUb".
File ID: "1Pjg0bVFffJ2Mac0nrwiowxQJlmaaWa0j".
File ID: "1fpV8um-56M-_ieNiSLE7DRkD8gDUK7ci".
File ID: "1304o3We0y7CzDO-5tBYKRgXeBkwYv3yc".
File ID: "1shaYuvCCiHVGHBuDHhX7_za-_PesqRJp".
File ID: "1bmJyvTRJXemsywTy8sBbjl_84QXsd-ZW".
File ID: "1Z31cAz94wewoaVEAKvNQloJTmjV8rRN4".
File ID: "1FAiwgIf6qdZQAauDKmPEcQhV5apUgk0_".
File ID: "1znSpej1xAd8veJFHAwAQH3got8r55Hmr".
File ID: "1U2YQeVg48voQZTBjW_t79VYiq4bVybcO".
File ID: "1qxhEggag1Ik6So-BLjDWybZfBwBn31i7".
File ID: "1b0b1F6MSGxqbF7Ml5Zd2wi4yOiLCHIWh".
400
File ID: "1DnfQ7pqTna5Lv6E-8mUiTWA1Q3-9g-Ka".
File ID: "1bkruL06rsScz2XSdsu8Op-HSduxOOkli".
File ID: "1O1miUSOGtsoUYQbDJoKCw1vcbFAHcMzK".
File ID: "11IQJZZ-c8jljAYRKMs75DKBYgySTPJV9".
```

File ID: "1QblgXzwNdFgvzwjoVtTaOwElveB9LujI".
File ID: "16Mpj7ZIfUtL6C2k6KCbBDFQ1IfPhpAUC".
File ID: "1yHU8TQ6VjWprnpVtoFaApTPh-03ZGnja".
File ID: "1KWMH7E3Bu4ZCVOkJYzD1PcVk8Nr9Cfgh".
File ID: "1ECJZzs8O2RlfYnD5Z2vgq0e69995DfVO".
File ID: "1eAV0zVMXu8YNNOF9ynay2cuUhKO5YZ1T".
500
File ID: "1DpGoQ-ojj3KUAB2M0q1gmkt61m6lvxRe".
File ID: "1kLBfxIjEWccacwTE3S7OydoNQ3lm7uGk".
File ID: "1LvIyMtgH-N9-dQEkJUlv7vYqPS29u58l".
File ID: "1KWoqgTw958hkO1SR-rsgNWugbIybEcQv".
File ID: "17FGodIimc_HCD-pUSk9DtkjQ4ijiOW_X".
File ID: "1DRETr8CKLM2HTFjfYWqhYEvSii9NPB8i".
600
File ID: "1y40O89aSZs89Lmg5ycZgz8bFkhX_0lvX".
File ID: "1aUSLE4Q_kQXdvTp6RlG0wRGFxVsIjYBY".
File ID: "1J-g0W6Jt7LQSPolebYOfg8VSZKZ_frQF".
File ID: "1OMzMhe9UumqZHg4Ij6WMPMJppiZAtTSq".
File ID: "1pIL-yS9DqAcSazYEuSWgx_YQxImAzQYK".
File ID: "1kPndppBls8dMKgHbQGcaQQxqcsjj-6mD".
File ID: "1N6BlXNXFr69fV0Z5tNXdIY1ByblxHgfu".
File ID: "1XIIIxZOJQBToomn_NHpFu-k_KU4fgNYj".
File ID: "1ZVi2it2VJY1_Y9P1XJLyU3k9bbv-SXWR".
File ID: "1O4OsJO0gHaz3_hbseylAPW1MpYhrOplu".
File ID: "146EsM3qaEHQBBX1HW_dUhibtAvo9yrie".
700
File ID: "1N1N9dfaMuSooviYEg4TUoZrgbTK4Yh4T".
File ID: "1XX4P46amsE0obD1YkoBNMCMa3VrERDUp".
File ID: "1pwf1C8uIDgbEnJckHFFbSt0QgV2HpOKV".
File ID: "1F88aIT60M0VRtRfRS3JaFsuA5m4D67B6".
File ID: "1Gm6egJCfblFaXlxeJH6lPHD2OMQONWA6".
File ID: "1BzSndZMU3O2TmbS3WaNOxZQVniNzerQt".
File ID: "1TSPF7CD3k_UMhrI3EOn41B-7D02vNmF6".
File ID: "11JLanPuZRr7WkP-F7W_MFXAuGXYG8nUG".
File ID: "1yWyyghroHzmlLnv5l5mi0JMmPvEVJXpY".
File ID: "1lRZ38FmS8g-gpDPldxlts2DWqQTnNUVJ".
File ID: "1Etv9zc9EWch1OwTuePPFDpRNrufuRvXk".
File ID: "1DlEZR-LGaFdVZhaERwzjeniKctDVsnrt".
800
File ID: "174uEsEV3S9e6l0uE4GmWCf0TH88b_5DP".
File ID: "1lIcrBXMGURdWCaseHM0pAg6Vmkvkjxij".
File ID: "1tN9q-Z-r6KMuZhBMn6FOWDjR-PQ2-lmk".
File ID: "1XRZRsDXwarpsYUpYUlLRqj6oWIquinw7".
File ID: "1nLVb1tGXKBWEnIBLJjGGnWG-5mVi1qjB".
File ID: "1Z1qJYDPF82D9aCQc5KprUbItXdNopqh0".
File ID: "11V_VShSbbAQuM1degDtrob8D8xKKAaOI".
File ID: "1uvYPIb8uRGIagwcO4Kx6uBJzIHMIGOT1".
File ID: "10A4HbZHZP6BSnNFR0WR4C13sqAXZdtXu".
File ID: "1GmQk0K8gcfb5OM5zAFprBTqTbeq_2tsx".
File ID: "1tiGsGXLgo_EomS8JKZSMuDRTGtw_KyoW".
900
File ID: "1v-qf1DfzpvKWZHoFAdDQVj5LbZ3FnmwW".
File ID: "1L9Wx9QM_PBNWUHX5LDG7ev3b82kQ6-G8".
File ID: "1IXsEF9vEZt_mST36IjFLTIPbjHKRRf31".
File ID: "1IIVWI7c2vp8-7pnV8Ql0eOgwX_C3ZOPZ".
File ID: "1TNAPnWgw3c6DrRM7i1p24Q6iLYrxAiBU".
File ID: "1IAbY0e58VnhMqz4dCnRfwbSGwsPLJKbE".
File ID: "18WuR2WZ5O5ewqhIvlqWEmzFmelDf1uuQ".
File ID: "1J5AnovFY6Dr9o4gkx2Y5O3YSY7s8zPp0".
File ID: "1JsfJ2-DXumkCm4Ci1HgTVJ56l-VVifdB".
File ID: "1dWFo2rPCn4IMTheaNgKoiG0ir0rpiLdd".
1000
File ID: "1e_eq7-W0TU6uX53seeKT1zlZUZEdMsuV".
File ID: "1FyvCdzCj3awF1Vc7DgNuPSsGDfc3A8RK".
File ID: "1WOqBFhI70JZZTPQ9sorCM4xeN0fnq60W".
File ID: "13t5s7KLI7e9w7MrB1GYpsjB7ll_vQzvO".

File ID: "15Ofy3NKyzjAUh1qLvI5ac-rINSZ6Raia".
File ID: "1cxQX7OZDO0vF9nB2kpk-vuo97tS5BGnL".
File ID: "1hJAdMnp2oxVxyJX2Qh-PYyG2zl0Kc0WB".
File ID: "1GSzZHa45FitG78bFGiL0b5LCxofjN8dj".
File ID: "1r8SudrmdHpm3ZzUBXX-h4WyVDfxvcpOy".
File ID: "1JbssAcX_rRxYcrfH20-ylJyp8uMba4Pi".
File ID: "1eiwMJjJAEddUwpsPKdMtASrWzOMN9Ysl".
File ID: "1Nin0hvCxxNZFQvrqWpQBZ1kKBysxuCL_".
File ID: "1F_uPwnQ2vO_d1MybQnjBfCUK6WcnJe2u".
1100
File ID: "1m32pVxh58TZShs3f-AQRLjhe-kRsqL-z".
File ID: "1IAZ6_uiq8Xp2YLNkMZwq8N0hnTtCVsy1".
File ID: "1O_15qdYmrMjP7FlQlgbAPYSWemaHMvka".
File ID: "1RTWnqro8O6FMh07GUps2UIdmagTAgxsL".
File ID: "13ouMsGgOH7nlZxCY2NMadBfrckvd7EKW".
File ID: "1UZCGxDdNSKrWzKDESTgCXAZBqoiIBLyI".
File ID: "1q2qiGlDEUxXdcawWaP54zKTnCfWgsapf".
File ID: "1RWhRc4p0MqCTlPGm_NktPcLnVyfjHBJ-".
File ID: "1EA_T2PRYEuLnpwjVZdMs_1bGTyPBF3Vc".
File ID: "1nc591xXTOKbAXHiJTZf8nfjVsz_dupH2".
File ID: "1O0-nlaHsWipM7fKEZWq5c6CiOHxoSQcQ".
File ID: "17NLprxL9Z-JarXAlpzn6VG72gtqK7iBX".
File ID: "1XTevLLiFZ7pAWb7Jk6yovrYpEO95wMj1".
1200
File ID: "10jf-lStSeMOj8XA6kbIPNqjIPVLAyQj1".
File ID: "11pBZpbI0s5sHPTLo8fNWB9PFMb4pTaQQ".
File ID: "1wou4OcbJjD8Pe7ZakGSxeKirN3y_IdaB".
File ID: "1jM5hqj8J84x1apcE6kQ9QyKHZ8L_USvv".
File ID: "1nAU8JgUfMdf2z0Yo7lSqhbwSc4X0pHaq".
File ID: "1pK8YIHelERDwxjwOGdFP0hwURiKF8E-z".
File ID: "1DU-nmN9AKqwvk9kD1U4B_wfvt2T_2meq".
File ID: "1v4NFmrJBWgXAePCX6BfoQjxJSzDOw4ql".
File ID: "16kujwx28aUSq_Q7RES1eNSPO1VgXqCq3".
File ID: "1IEh5iF-jiM9052F8ax_8QrYo0Y59QMn4".
File ID: "10dPBx1es-8loHs3bcCXIC5eGTocctM4Z".
1300
File ID: "14D7e4DHKIOI9q2h-928qXL_ITV0wuHUi".
File ID: "1nK7Wi2lmoKjxPX4etOptuxU3DqaIHYyP".
File ID: "1_NvJyFlvdqrXkhayA_cA5LvezMKkHp26".
File ID: "1umobyUEFsw-3Ea2w2jhiiovL-DQ8EPC_".
File ID: "1jBXJ1HeK1nrJ98wTsZzv77WkH9uq2aDX".
File ID: "1BhdVfg2tr0Nib4VS4pUmaGWTFwO8Dpsq".
File ID: "1JEExFWPQpsMLeEVv930PejEet-SP8WUd".
File ID: "1dss1ryRsHurb9GFoqbG9mhJphoa3A4IS".
File ID: "1WOXXmu2Jcqo6dZxRiGQ9LEpNYghVBs3w".
File ID: "1J1iKwBA-V8vNj6oU7-80xrbG0hbbPaaK".
File ID: "1B4phya8QQ37oQLOJCQ4ydWf90wgE3HxI".
1400
File ID: "1qB7osFae6t0Gb1dRPq_qpu3k4ZntR73d".
File ID: "13HqCyyuEgzb63IuZ0erq3GL-hp3zEMNO".
File ID: "1qbBMUhnImFuNT3cpWD010A1x-vXUBkTL".
File ID: "1HEm6kiCzdvbRzbSYHdKDLxFQiiQVdZKc".
File ID: "12k4pozaXOrBo0AWk2sn0Xjt7RfQKD5pk".
File ID: "1iP9FlYUFJvG9c645Oez-bQT0TyWF0VC5".
File ID: "1egr787po4BautDpQcJ_bspg29P292cUG".
File ID: "1XdQ_vi6KsekECpEJFnoSJlj53SmZbBn-".
1500
File ID: "1IInjGoLR7NxF4H2-_cE2hLQXCOKDI9ms".
File ID: "1C2-uwYv5aEXJkq4DmTNGKwHdfwKC3LVe".
File ID: "1xiFwhdwTOBovHKGfKXvFMocnxnza6Ri0".
File ID: "16IP_DfPqFfknuhwf7IQe-wRFyxrVhYKU".
File ID: "1WYME1lcun4xRKv4_yRaMqPhCPJvl6Fse".
File ID: "1xsweQTCJ6BxXWag6ZXDSZykamhq7cgdN".
File ID: "1vomRRGw6tUQY7DMoUlP6lYastubxbdlA".
File ID: "1Wg94yiw-7OSNztoMGEjuOjPxlnTMr2zb".
File ID: "1NEydXQZDTiUoCbGY_tMLOV7YUVDL8PP5".

```
File ID: "12YmO6A_lEUPsedOrP933sxtDkd1Ulyl6".
File ID: "15Nbw4lzze1_O9lJgwBZx_llJkhSkRoVB".
1600
File ID: "1Bi3bbT-DmkUsJAtpx5JlAqX-GOUhg1Tt".
File ID: "1w4zfNy0eh5U_e7CXlROtxpWwzqykXkn6".
File ID: "1YOJxvdg5Z2RZtOc6X_5beYD_Zv8zlwvd".
File ID: "1E3c3kLHzQ9RbhZigy3ikwBUm8vBHkXu6".
File ID: "1b_qKz9UQpmhr5kZXMTHx_YyRKnoTzWnn".
File ID: "12IMp1MjCkfaPAbELsLHNzjSEFGDKCs3p".
File ID: "1ci0ezqsjKp7oosSAasvneX9VfeCZoT-X".
File ID: "1VjH_WECBYJX4ZrrrTSZPtX53dyfvw9R1".
File ID: "1l-9voKQ2BCY5NLF17NOz3n4vtb2nVNrQ".
File ID: "1zi1R07Ybky4lMnewW1-KfmVctcZM5O7s".
1700
File ID: "12laswlmBdOyM5xdsqsRQ5X9ZeiPwqXlF".
File ID: "1X9a7QprA6TQwn8cWM6gej-PH5JrgmpXW".
File ID: "1rvSwqqCiwTLHzOUNC5w9vbHH--lBBmK0".
File ID: "1C4WBof5rL37beWn3bH8ssDBOSLIpUXM1".
File ID: "1trUFpkcIER9uEP_rrPAzmSN4E60doq-3".
File ID: "1Splv1LwHIHdNCqM0Y1ZpHNuJIbCpkV0F".
File ID: "1IiZBRB14h0qYqvo3R6FRpjNJcT7EN-iU".
File ID: "18RNRFFhJT8RbJFA0n7Ppor53FoB4WIoR".
File ID: "1aBBwc_18HusPh72UPOZGvsW29jXhzGlI".
1800
File ID: "1v1-MZI2dmo9YOt8hCICfW4BiTUdwSpMf".
File ID: "1kD_nuobJMZh0WOzVDSkUY5LWIux0a0JF".
File ID: "1x88WvmJfiN2nzCp8f4UmisZcGEGc2NJx".
1900
2000
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-19-bea9dae9ed10> in <module>
     89 # Iterate over the DataFrame and call the functions
     90 for index, row in data6.iterrows():
---> 91     image_url = get_image_url(row['Coordinates'], row['3months'])
     92     j += 1
     93     if j % 100 == 0:

<ipython-input-19-bea9dae9ed10> in get_image_url(coordinates, timestamp)
     14
     15     # Check if there are any images within the 1-year window
---> 16     image_count = image_collection.size().getInfo()
     17     if image_count == 0:
     18         #print(f'No images found within 1 year of {timestamp} for coordinates {c
oordinates}. Skipping row.')

~/anaconda3/lib/python3.7/site-packages/ee/computedobject.py in getInfo(self)
     94       The object can evaluate to anything.
     95     """
---> 96     return data.computeValue(self)
     97
     98   def encode(self, encoder):

~/anaconda3/lib/python3.7/site-packages/ee/data.py in computeValue(obj)
    900           body=body,
    901           project=_get_projects_path(),
--> 902           prettyPrint=False))['result']
    903
    904

~/anaconda3/lib/python3.7/site-packages/ee/data.py in _execute_cloud_call(call, num_retr
ies)
    327   """
    328   try:
--> 329     return call.execute(num_retries=num_retries)
```

```
    330   except googleapiclient.errors.HttpError as e:
    331     raise _translate_cloud_exception(e)

~/anaconda3/lib/python3.7/site-packages/googleapiclient/_helpers.py in positional_wrappe
r(*args, **kwargs)
    128                 elif positional_parameters_enforcement == POSITIONAL_WARNING:
    129                     logger.warning(message)
--> 130             return wrapped(*args, **kwargs)
    131
    132         return positional_wrapper

~/anaconda3/lib/python3.7/site-packages/googleapiclient/http.py in execute(self, http, n
um_retries)
    930             method=str(self.method),
    931             body=self.body,
--> 932             headers=self.headers,
    933         )
    934

~/anaconda3/lib/python3.7/site-packages/googleapiclient/http.py in _retry_request(http,
 num_retries, req_type, sleep, rand, uri, method, *args, **kwargs)
    189         try:
    190             exception = None
--> 191             resp, content = http.request(uri, method, *args, **kwargs)
    192         # Retry on SSL errors and socket timeout errors.
    193         except _ssl_SSLError as ssl_error:

~/anaconda3/lib/python3.7/site-packages/google_auth_httplib2.py in request(self, uri, me
thod, body, headers, redirections, connection_type, **kwargs)
    223             redirections=redirections,
    224             connection_type=connection_type,
--> 225             **kwargs
    226         )
    227

~/anaconda3/lib/python3.7/site-packages/ee/_cloud_api_utils.py in request(***failed reso
lving arguments***)
     61       session.max_redirects = redirections
     62       response = session.request(
---> 63           method, uri, data=body, headers=headers, timeout=self._timeout)
     64       headers = dict(response.headers)
     65       headers['status'] = response.status_code

~/anaconda3/lib/python3.7/site-packages/requests/sessions.py in request(self, method, ur
l, params, data, headers, cookies, files, auth, timeout, allow_redirects, proxies, hook
s, stream, verify, cert, json)
    585         }
    586         send_kwargs.update(settings)
--> 587         resp = self.send(prep, **send_kwargs)
    588
    589         return resp

~/anaconda3/lib/python3.7/site-packages/requests/sessions.py in send(self, request, **kw
args)
    699
    700         # Send the request
--> 701         r = adapter.send(request, **kwargs)
    702
    703         # Total elapsed time of the request (approximately)

~/anaconda3/lib/python3.7/site-packages/requests/adapters.py in send(self, request, stre
am, timeout, verify, cert, proxies)
    497                     decode_content=False,
    498                     retries=self.max_retries,
--> 499                     timeout=timeout,
    500                 )
```

```
    501
```

~/anaconda3/lib/python3.7/site-packages/urllib3/connectionpool.py in urlopen(self, method, url, body, headers, retries, redirect, assert_same_host, timeout, pool_timeout, release_conn, chunked, body_pos, **response_kw)

```
    708                 body=body,
    709                 headers=headers,
--> 710                 chunked=chunked,
    711             )
    712
```

~/anaconda3/lib/python3.7/site-packages/urllib3/connectionpool.py in _make_request(self, conn, method, url, timeout, chunked, **httplib_request_kw)

```
    447                     # Python 3 (including for exceptions like SystemExit).
    448                     # Otherwise it looks like a bug in the code.
--> 449                     six.raise_from(e, None)
    450         except (SocketTimeout, BaseSSLError, SocketError) as e:
    451             self._raise_timeout(err=e, url=url, timeout_value=read_timeout)
```

~/anaconda3/lib/python3.7/site-packages/urllib3/packages/six.py in raise_from(value, from_value)

~/anaconda3/lib/python3.7/site-packages/urllib3/connectionpool.py in _make_request(self, conn, method, url, timeout, chunked, **httplib_request_kw)

```
    442                 # Python 3
    443                 try:
--> 444                     httplib_response = conn.getresponse()
    445                 except BaseException as e:
    446                     # Remove the TypeError from the exception chain in
```

~/anaconda3/lib/python3.7/http/client.py in getresponse(self)

```
   1334         try:
   1335             try:
-> 1336                 response.begin()
   1337             except ConnectionError:
   1338                 self.close()
```

~/anaconda3/lib/python3.7/http/client.py in begin(self)

```
    304         # read until we get a non-100 response
    305         while True:
--> 306             version, status, reason = self._read_status()
    307             if status != CONTINUE:
    308                 break
```

~/anaconda3/lib/python3.7/http/client.py in _read_status(self)

```
    265
    266     def _read_status(self):
--> 267         line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")
    268         if len(line) > _MAXLINE:
    269             raise LineTooLong("status line")
```

~/anaconda3/lib/python3.7/socket.py in readinto(self, b)

```
    587         while True:
    588             try:
--> 589                 return self._sock.recv_into(b)
    590             except timeout:
    591                 self._timeout_occurred = True
```

~/anaconda3/lib/python3.7/ssl.py in recv_into(self, buffer, nbytes, flags)

```
   1069                   "non-zero flags not allowed in calls to recv_into() on %s" %
   1070                   self.__class__)
-> 1071             return self.read(nbytes, buffer)
   1072         else:
   1073             return super().recv_into(buffer, nbytes, flags)
```

~/anaconda3/lib/python3.7/ssl.py in read(self, len, buffer)

```
    927         try:
    928             if buffer is not None:
--> 929                 return self._sslobj.read(len, buffer)
    930             else:
    931                 return self._sslobj.read(len)

KeyboardInterrupt:
```

Creating non fire dataset

To create the dataset for non fires we have taken a random timestamp between 2017-2022, random coordinates in the state of Oregon, and a ID number

In [19]:
```
# Function to generate a random timestamp between May and November of 2017-2021
def random_timestamp():
    year = random.choice(range(2020, 2021))
    month = random.choice(range(7, 12))
    day = random.choice(range(1, 30))
    return datetime.datetime(year, month, day)

# Function to generate random coordinates within Oregon
def random_coordinates():
    lat_min, lat_max = 42.0, 46.3
    lon_min, lon_max = -124.6, -116.5
    lat = random.uniform(lat_min, lat_max)
    lon = random.uniform(lon_min, lon_max)
    return [lat, lon]

# Function to generate a unique 9-digit ID number
def unique_id(existing_ids):
    id_num = random.randint(100000000, 999999999)
    while id_num in existing_ids:
        id_num = random.randint(100000000, 999999999)
    return id_num

# Generate dataset
timestamps = [random_timestamp() for _ in range(1000)]
coordinates = [random_coordinates() for _ in range(1000)]

ids = set()
unique_ids = []
for _ in range(1000):
    unique_id_num = unique_id(ids)
    ids.add(unique_id_num)
    unique_ids.append(unique_id_num)

data = {'Timestamp': timestamps, 'Coordinates': coordinates, 'ID Number': unique_ids}
df = pd.DataFrame(data)
```

In [20]:
```
df['3months'] = df['Timestamp'] - pd.DateOffset(months=3)
```

We then scraped the non fire images

In [26]:
```
# Function to get the image URL
def get_image_url(coordinates, timestamp):
    latitude, longitude = coordinates
    # convert the timestamp to a string in the format 'YYYY-MM-DD'
    date_str = timestamp.strftime('%Y-%m-%d')

    # convert the string to a datetime object
    date = datetime.datetime.strptime(date_str, '%Y-%m-%d')
```

```python
    # Define the image collection and filter
    image_collection = ee.ImageCollection("USDA/NAIP/DOQQ") \
        .filterDate(date - datetime.timedelta(days=120), date + datetime.timedelta(days=
        .filterBounds(ee.Geometry.Point(longitude, latitude))

    # Check if there are any images within the 1-year window
    image_count = image_collection.size().getInfo()
    if image_count == 0:
        #print(f'No images found within 1 year of {timestamp} for coordinates {coordinat
        return

    # Get the most recent image
    most_recent = image_collection.sort('system:time_start', False).first()

    # Define the visualization parameters
    vis_params = {
        'bands': ['R', 'G', 'B'],
        'min': 0,
        'max': 255,
    }

    # Define the region to get the image
    region = ee.Geometry.Point(longitude, latitude).buffer(250).bounds().getInfo()['coor

    # Get the image URL
    image_url = most_recent.getThumbURL({
        'region': region,
        'scale': 1,  # Set the scale to match the native resolution
        'format': 'png',
        'resampling_method': 'bicubic',
        **vis_params
    })

    return image_url

# Function to save the image to Google Drive
def save_image_to_drive(image_url, file_name, folder_id=None):
    # Set up Google Drive API
    creds = service_account.Credentials.from_service_account_file('forestfiredtu-ef1cef2
    service = build('drive', 'v3', credentials=creds)

    # Download the image
    response = requests.get(image_url)
    if response.status_code != 200:
        print(f"Failed to download image: {file_name}.png")
        return
    image_data = response.content

    # Save the image to a temporary file
    with tempfile.NamedTemporaryFile(suffix='.png', delete=False) as temp_image:
        temp_image.write(image_data)
        temp_image.flush()

        # Save the image to Google Drive
        file_metadata = {
            'name': f'{file_name}.png',
            'mimeType': 'image/png'
        }

        if folder_id:
            file_metadata['parents'] = [folder_id]

        media = MediaFileUpload(temp_image.name, mimetype='image/png', resumable=True)

        try:
            file = service.files().create(body=file_metadata, media_body=media,
```

```
                                            fields='id').execute()
            print(F'File ID: "{file.get("id")}".')
        except HttpError as error:
            print(F'An error occurred: {error}')
            file = None

        # Remove the temporary file
        os.unlink(temp_image.name)

    return file
# Specify your Google Drive folder ID
folder_id = '10tExCQHC4pEz7M8qrTBQ6EcS-lqfDDft'
j = 0
# Iterate over the DataFrame and call the functions
for index, row in df2.iterrows():
    image_url = get_image_url(row['Coordinates'], row['3months'])
    j += 1
    if j % 100 == 0:
        print(j)
    if image_url:
        save_image_to_drive(image_url, row['ID Number'], folder_id)
```

```
File ID: "1PTaAiyPQo-kUwHIYx97W-oCBKpy_Z_r7".
File ID: "1StEkr4I5eG1jGcfnb0iZgnN_ca5MkTPf".
File ID: "1cEAEu0dP9CUGBjW-4q0Z0-wf8v071vM9".
File ID: "14T9oVBwtXo-KKGmNqup_FtZ_Pckcik_w".
File ID: "1FRXkb32Eo-oStUZSZSj1KBxlNJo-LCIi".
File ID: "1MUvpn-45MFBeefMrwre-XpdRrJ2rm2q9".
File ID: "1dIh4nUnNH0s3drZRm0JPi2zfLqUihhnd".
File ID: "1FkLE1zRWYOc5_nRaGVpoo5xOc6YcXA5C".
File ID: "1VUzQPcT4dta_9I2VH9MLdBJTG5M6TxMm".
File ID: "1Y-fzxFhTGPvz88u5MWe_u0sm2Mmqf9HX".
File ID: "1I-5Qhc9xTzzGCdI0HmjHl69YULakP3Yq".
File ID: "1zGw95b9wtH6Dlj_uvXOvefzwLpPaRKdj".
File ID: "10-scKOcxP88RATVFH8ghRfZq5Be_1WJY".
File ID: "1VCAPjfm_kHLSWA7bHnY4lyrox-ZtJpIe".
File ID: "1cTCfiak2T8kvUqW-ODgoMI_HQ8GtMp2G".
File ID: "1YiXieLj8J-LXluDdWL2uhgzvrVxCXDX3".
File ID: "1rWBYi3wLrxruQOFahTlDOlErY8hZCJ9R".
File ID: "1mI84scZCiFo4FxTOL8VWS2LEMMDg61Kv".
File ID: "1U9rc3qOwon67-YR4on0Ex_SzCa2FeHJS".
File ID: "1dRLxUc3qJezps029ntCm-AsRBbgursDc".
File ID: "13J3UjssmWU_egDO8GzdUUtOALMVeLxza".
File ID: "1ggxvjDota9qhL5BJW3MjFUeKW84SJEH7".
File ID: "1ms1Q_Zof4o1ABhLZNiAwtvbozLd1hBup".
File ID: "1SXluz5IlMtO26Ke7zuPp9470OBQDw0kb".
File ID: "1j8QaHVw2iVAdgehpj1dYhCrbJFN-GrTL".
File ID: "1NlU6fjIweTRkGfFOqVR2MNa0yHKHhY8l".
File ID: "14U3kSTgXXfOsDe_inBLt1lO8eIPKMfYt".
File ID: "1lyKmD3KgcVFT4AXzj3wgwzZNB3HhrL59".
File ID: "1bZw50b0Q-GAzNVN5CF1o1q0yrU-6CVAT".
File ID: "12ZKdx6L5-O2P34XZLFjBygVq6RV6ZJOr".
File ID: "1sV_7JMypBn4ijCSR22kkE1h4vkdrzpNQ".
File ID: "1cDTWFEbtME0KPTkm7-G4trqE2PgCINGD".
File ID: "19-z3bxKYOP9O-Jf_Sr8k2KQaRw3UEKKk".
File ID: "157gMNvqQSPH9e6ibbPYD8QiO9SjoPPzq".
File ID: "1vKMVk18EAjy8fF7HNpWy_-rxSmhnVzFz".
File ID: "1SLtDLvdraKyIcSyDeo-OhRG9WU3GLadJ".
File ID: "1IKfBIWkSjcb5EQVoGL2jXZdcOWDSCc30".
File ID: "1gIRtigueiUytQAF-zjo_dVF0vXFOJ86S".
File ID: "1mNRVDGU7yH_30A-0nXgYYDw2Qca3S40Q".
File ID: "1lEK1rsgrqS9_EFwVBDw9G78gxIM9CIHg".
File ID: "1HfTiuZZgyK7gO0QQddk8rwv_f3-YoCao".
File ID: "1WDfAB0paueVbWfUgvwOyPObzAHyImD0T".
File ID: "1vxrDisYT2M82IBdLS6gzLFpDJvELAPQk".
100
```

File ID: "1lstj0rXpzeJ0ET_8mHcjP1xrBNoOC5Be".
File ID: "1OlpaSpp6fQxPUa2MxrhIj1UaQBBTXX9l".
File ID: "1W9HJCj4CPE0FWXdWtG4v8u6SE8Pb8w8N".
File ID: "1A-dwyHhYo7sRJcAxVbX23ZS0m9CC3GJ-".
File ID: "1m4dsj2SyU-HdK9LTIl-X2IT2Gd1Fo9hn".
File ID: "1AmDVV2gmvv7h81-WpgzmahVtEvomRljU".
File ID: "1xA49l7f5LPvXg7KqbciU5DSKtlomqAeE".
File ID: "1Fh_F1lG7udXBugJmE6xW6nbYRHgljVr1".
File ID: "1lrJ1p_QqzB4D7NKtAW9-uNKZkxxn9lFv".
File ID: "1oq8hktwTbH3-OGCctIrsSgBDFIxIspv3".
File ID: "11sIn_4_wEtxnBnpPjjWLlsOXo_YOLMaq".
File ID: "1D-M9-ob4jFFgiaJ4S2eTHuM2WTN1z5Ka".
File ID: "1WLgdpVBOF0vo4iOnzkT_HBgY1aXDWrs7".
File ID: "1ILTYoeGJYmvP4P_mnFj-wxW_8nUV15p4".
File ID: "12sbHgBMU5jM1yr9eId3yfW3oNtyQxxw2".
File ID: "1i4qhAkT7V4jXUjcyzkSN6xtCQPElUrDO".
File ID: "1gumh5SwLilu7GOUWaTR_lBD3aueeHVvQ".
File ID: "1HipqBMCiXCSRLZGGpq9OG8_qifXXqpfK".
File ID: "1XiN3RY2kqpbDy0p70YbJqs2wMmyfudIb".
File ID: "1zuMwzvwU_zES81GchaPY_2SaWlVO1XOC".
File ID: "1lfLZzY1q_azs1btky-mPjI4uMi7qkrZC".
File ID: "1nm6mvzPi_ZyDh0h23JjlZpubUI26zv6w".
File ID: "1g1-JHm0XxgGoDykCOgbfomizZV3Ij-Bb".
File ID: "1Dh9jqJP_LmU36g05R3vIUAny22k_uGFa".
File ID: "1_fvmXm5AQo6AXMA1tAvNM4rZIsWDifFy".
File ID: "17fz8EADtS2siodLuGuExM8uOIMbr0Laf".
File ID: "1RMg58Oru3KcA2kUiE6SlFbagIe-7T2pn".
File ID: "1-Xt2x3uTBpfiMTCBZK_el2uT8cpi_res".
File ID: "1XGMuwdbXywraqd0DkO6ORZFr811PcrWl".
File ID: "1xNzW66q5QayQyljYjToBoZ4eepyFO1Wf".
File ID: "1boSgr59qpsgsTL8YP-BNYynoN9nvpLWo".
File ID: "1rIa5ULIV--bH-AEx_5fqsGGpfdpZsTRp".
File ID: "1LDpapdXdrPN1lxnD52DCHYChWmpGSy1J".
File ID: "1uln-hHsFbz7dbemNhy3CaCZy1caveGEh".
File ID: "12tIkfURk-BGC9kx-A1cvndUb9dO3muHa".
File ID: "1DKcoTEWdLWly31xB3_dHQiQGM08uK1wp".
File ID: "1pmH0HtFbxwaFQ_X6ztA-pCOzuLSvAGON".
File ID: "1_Mwgn-ICo08rD2pZZk9PcC1_-kxTaJsH".
File ID: "1M7aoPZYQ56AKsCwyqDcjbRiYUUKkN_ID".
200
File ID: "1AbpNcbvAoQPFYnY0Q8ze0mXwP7tSLqDF".
File ID: "1Z1v634mXQxLPwSKq6ZhWmt7GN5IexSUy".
File ID: "1htVmDtYoPE1nrJnhTuhAcVu4P6wfAtCL".
File ID: "1hp39AuuL7r-gKKs1q3v7BLTmO5JkmWuI".
File ID: "1tK2r8JIp0vF9M4iVZD1rSgWB4Sr3a2CV".
File ID: "1PPEaxP_eKxH6ey88Ftov2XTfl0kWTaG5".
File ID: "1IgIDhbP_xMkJkDHR6jD8g3VrKL5FjV_y".
File ID: "1zQkL_r2brAhumsrgknho8gTOdAi_2Ylo".
File ID: "1cjGV4oehkAeEt9eoCEFfpfHscZ-_fPwq".
File ID: "1JBF2fkLXZrMcwc32liVAkWpAzz5z6z7h".
File ID: "1QbcjolhlwGaBGhhoraDr0TXJMqThrd33".
File ID: "1RMAsLSTar0htn7uSVk5YENVNsJQfvi5b".
File ID: "1RsJ6ZfNHb6GKfyO_gjvFesQkZ4JfUY6a".
File ID: "1pN7cfLq0gXFFBj6BekVdcmaBeY2EPCcw".
File ID: "13gNmmgOs3EV-hZrxbI4EP5QyOxQlgXRI".
File ID: "1FxiDuy7k_65FO3vmm18GvBVmCR1zi-Pl".
File ID: "1-fT8RTRLEeiYLsxOukL3DhpNy9B7oCak".
File ID: "121S61v7f2BIV4GsnSSqgNSVQO25_FnjL".
File ID: "1I6sCV9Y_AC_VXNbvKyTy3s4dmR5Ji8po".
File ID: "1TVil36AMFoQ72ldEw3hoI8e1eW63InZE".
File ID: "127J-qLym4gX-saUvzQQB32hveMmg65Rl".
File ID: "1EGpY7DiGouDYTnRiW4tFBbYUeWcAYH4U".
File ID: "1FT-byGQuOMCaCdQ7weYH6Zq3lmrkLtyn".
File ID: "1L2imG1-NQsmWeJIIKh3SEjUSxWbeV3xg".
File ID: "1lrCPgG6gPsho0oE5WKZMWRMsLiNbj7AV".
File ID: "1nABqTPPYQp0g-3WV-qBDfwjMBgswjU2t".

```
File ID: "1nmCqcoT0v7XYFvX6Q6wkSh-WIjVAS_LD".
File ID: "1UTaSrt7GsSOFvk5y-H7hY5TPpaCeYdRH".
File ID: "1VPci747PJ2qAyCF-kv9TP_-fV7V6JfgD".
File ID: "11K_GjZrh_0ETTmf8EdmvrrqJSAuhzvTQ".
File ID: "1xR5ZUkEbBMlB11xU83nexIlOROkeYX6M".
File ID: "1AbMNtGzbVITNLKsvdz8xLJQ8U36xG4lj".
File ID: "1EUun7sOfKclVnfTpmEkTreY_xBS7MeNG".
File ID: "1_GI1zWdyKQrd3_XxFX9oxfrkXzgEMe8T".
File ID: "1NI273lTlMFmYkVqvlPc6drxsmPSHbSfa".
File ID: "1Uq7wt9kpi_YyIpANggDMvFNvWENyV5IK".
File ID: "1TIsVC7oB-t_LwAzFIqPY3RR1gvdlQ5NP".
File ID: "1L_xbuTt-NPRyrJvkNTAVy2ZJ8o5a4puQ".
File ID: "15YmCOzSGm41pT5bCpX-ejgyn41StvOaH".
File ID: "1Yl_15E1Tbk3Qw-CbNealxsiwkh6arsYQ".
File ID: "1GCCzr7mTDeGY_FjIJg0p5iHnlsLRQDTO".
File ID: "16U-pCc4LoShD8DJudUtG-MVeKElSM60m".
File ID: "1LRQyi2ITpFDvtEOWSzc6clfXSrul3PxZ".
File ID: "1UlRj1YeZ82GQap9Wl1PT923qg1HV3Om0".
File ID: "1nLXCzX7tZZzcU8MDL3P4sW1UlcKn-_wo".
300
File ID: "1qjLlwcgsS2tlkPDzXQ717ObQ2Hk-y6ni".
File ID: "1ful6261IbQ0C8cuaDdrPE6g5tnMJ5BuJ".
File ID: "1sAx-r5ILfiAUVSZX8S-tI6C19-lHRQUz".
File ID: "1-17Z1_2XubgkSLVQYjmwSVs3TMhOaeRD".
File ID: "1nIYJIxbTgzVzMcSc48rBeRYACmhlay_v".
File ID: "1Kv1adLitBM6PRSMm1P4-F7NzncRPfnqp".
File ID: "1k27BAaVWCYb1A-fjW86tsrzDUnRs-YJ8".
File ID: "1R2iNOcKPDXJSHrxGYmvaQSv7xSfTKsTH".
File ID: "1qufaqLbSeVNJmk9Ad3mHTioi6wijjM1E".
File ID: "1HttHJKbfi1H0cWqYEdEEDP2xDm26BuJ6".
File ID: "1GxhSYHrERnnmV2W1qZux4m4xVgyoVbUE".
File ID: "1a8DZcrRVa0uJiGg7ampVB6rz_oIi3_hp".
File ID: "1P1lbRWE5DZYQRuoojD7OlgfEAswmeYE6".
File ID: "1AN6zxsAQdlqL83jHrYyzQwVxvmeIHTym".
File ID: "1cGOZmvqpfQLC62SKmUsy69gKD8eOw1wN".
File ID: "1GzFZSMPNYN-ORsVtsUO1S3CDpqf_v_kU".
File ID: "1J-nlX0BiLXzZjaNRFk1oXgNhnmdFJgLg".
File ID: "1KV8sntdArHKV4TinwbjFybyFzAxNl3gZ".
File ID: "1ffZfQYtqsqEuPW9dQ-hP8hdRbZovLTuj".
File ID: "1N5ueE7KyT69nH5ZvuWBAfe2eCfKOZN2m".
File ID: "1gdqqF7qnDseJhq99usypwRNfMh90gfV5".
File ID: "1MRs9cbl7F_iniFJ3UOaeqQzKUQtMQt_u".
File ID: "1xKhQu9qd_yff-rIm2bVFblSrcZG9vOI6".
File ID: "1peR2iKaVO2IMyTvf4lQ2zfnsQSZzZskW".
File ID: "1W3XDoVWrRKYeMQ5wTpiADPbOHb80LAxi".
File ID: "1a-dxJpIbxvb4iROsgHh8EcJQ793FvO-8".
File ID: "1Wj8m_91KvwIv8nXSkwdGUx4QZWmDw9Cc".
File ID: "1syArcG5LEkkZ3lJ_aDAi5STYk0pq_74Y".
File ID: "16Etsw7gSopJHJ3rXuZY_JfH1Fw5TFIFq".
File ID: "1iVZaoU95NO1u4Rw5Eubol1-aGFN0ZLJn".
File ID: "1dVJYn7mNTEYczXMp9afU0ZZBDByRn-A_".
File ID: "1zF1vdD4JZ3DzleigoaW6Q-oUnP4tmE51".
File ID: "1qEQGUxivSUDJGxQBBmuvskMj0Mdtcg58".
File ID: "1rQ5Vg1M67OW9D23KeJdMLYc-RMvqYI63".
File ID: "16w89VZCl2uokRFvq4X5shq6F0aBTSt4F".
File ID: "1AnEFhQtOmQT2wvo9qur37K_6sRBQx6MZ".
File ID: "1SVTSpxBzKNtzKN0H33UwytCz3R3-arFn".
File ID: "1wLLDAynANp3m5Q9Z9F4xJwCLl5qLhI-y".
File ID: "1DASb7gCin94P-8FqW4NYRtpFQpFsSMzb".
File ID: "1ed2kAQsw9RH13Fd5qdJ-mN0OAngZ6Rtt".
400
```

This notebook showed how we used the Earth Engine API to create our dataset, that we will use in the rest of the project to predict fire hazard in Oregon.