# Supress Warnings

import warnings

warnings.filterwarnings('ignore')

# Visualization

import ipyleaflet

import matplotlib.pyplot as plt

from IPython.display import Image

import seaborn as sns

# Data Science

import numpy as np

import pandas as pd

import statsmodels.api as sm

# Feature Engineering

from sklearn.model\_selection import train\_test\_split

# Machine Learning

from sklearn.ensemble import ExtraTreesRegressor

from sklearn.metrics import r2\_score

# Planetary Computer Tools

import pystac

import pystac\_client

import odc

from pystac\_client import Client

from pystac.extensions.eo import EOExtension as eo

from odc.stac import stac\_load

import planetary\_computer as pc

# Please pass your API key here

pc.settings.set\_subscription\_key('1c08f190faaf461db5a4cf29338be8dc')

# Others

import requests

import rich.table

from itertools import cycle

from tqdm import tqdm

tqdm.pandas()

from tqdm.notebook import tqdm\_notebook

tqdm\_notebook.pandas()

crop\_yield\_data = pd.read\_csv("Crop\_Yield\_Data.csv")

crop\_yield\_data.head()

def get\_sentinel\_data(longitude, latitude, season,assests):

 '''

 Returns a list of VV,VH, VV/VH values for a given latitude and longitude over a given time period (based on the season)

 Attributes:

 longitude - Longitude

 latitude - Latitude

 season - The season for which band values need to be extracted.

 assets - A list of bands to be extracted

 '''

 bands\_of\_interest = assests

 if season == 'SA':

 time\_slice = "2022-05-01/2022-08-31"

 if season == 'WS':

 time\_slice = "2022-01-01/2022-04-30"

 vv\_list = []

 vh\_list = []

 vv\_by\_vh\_list = []

 bbox\_of\_interest = [longitude , latitude, longitude, latitude]

 time\_of\_interest = time\_slice

 catalog = pystac\_client.Client.open("https://planetarycomputer.microsoft.com/api/stac/v1")

 search = catalog.search(collections=["sentinel-1-rtc"], bbox=bbox\_of\_interest, datetime=time\_of\_interest)

 items = list(search.get\_all\_items())

 item = items[0]

 items.reverse()

 data = stac\_load([items[1]],bands=bands\_of\_interest, patch\_url=pc.sign, bbox=bbox\_of\_interest).isel(time=0)

 for item in items:

 data = stac\_load([item], bands=bands\_of\_interest, patch\_url=pc.sign, bbox=bbox\_of\_interest).isel(time=0)

 if(data['vh'].values[0][0]!=-32768.0 and data['vv'].values[0][0]!=-32768.0):

 data = data.where(~data.isnull(), 0)

 vh = data["vh"].astype("float64")

 vv = data["vv"].astype("float64")

 vv\_list.append(np.median(vv))

 vh\_list.append(np.median(vh))

 vv\_by\_vh\_list.append(np.median(vv)/np.median(vh))

 return vv\_list, vh\_list, vv\_by\_vh\_list

## Get Sentinel-1-RTC Data

assests = ['vh','vv']

train\_band\_values=crop\_yield\_data.progress\_apply(lambda x: get\_sentinel\_data(x['Longitude'], x['Latitude'],x['Season(SA = Summer Autumn, WS = Winter Spring)'],assests), axis=1)

vh = [x[0] for x in train\_band\_values]

vv = [x[1] for x in train\_band\_values]

vv\_by\_vh = [x[2] for x in train\_band\_values]

vh\_vv\_data = pd.DataFrame(list(zip(vh,vv,vv\_by\_vh)),columns = ["vv\_list","vh\_list","vv/vh\_list"])