```
import pandas as pd
import numpy as np
pd.set_option('display.max_columns', 50)
from scipy.spatial import cKDTree
from math import *
import matplotlib.pyplot as plt
%matplotlib inline
np.random.seed(42)
import urllib.request
import urllib, os


from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True)
```

```
import pandas as pd
import xarray as xr
import math
from datetime import date, timedelta
from dateutil.relativedelta import relativedelta


# Importing libraries
import os, shutil
from keras import models, layers, optimizers, regularizers
import datetime
import matplotlib.pyplot as plt
import scipy
import numpy as np
from PIL import Image
from scipy import ndimage
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
import itertools
```

```
from sklearn.metrics import confusion_matrix, classification_report
np.random.seed(123)
import math
import tensorflow as tf
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.optimizers import RMSprop
#from keras.applications import VGG19
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import preprocessing


df_grace = xr.open_dataset(r'/content/drive/MyDrive/ugp/2_deg/regrid_2_deg/grace_04-2002_11-2021.nc')
df_preci = xr.open_dataset(r'/content/drive/MyDrive/ugp/2_deg/regrid_2_deg/Precipitation_04-2002_11-2021.nc' )
df_temp = xr.open_dataset(r'/content/drive/MyDrive/ugp/2_deg/regrid_2_deg/Land_and_sea_temp_04-2002_11-2021.nc')
df_ssh = xr.open_dataset(r'/content/drive/MyDrive/ugp/2_deg/regrid_2_deg/zos_AVISO_04-2002_12-2010.nc' )


df_preci·=·xr.open_dataset(r'/content/drive/MyDrive/ugp/2_deg/regrid_2_deg/Precipitation_04-2002_11-2021.nc'·)
df_preci
```

xarray.Dataset

| | | | | |
|---|---|---|---|---|
| ▶ Dimensions: | (**lat**: 90, **lon**: 180, **time**: 236) | | | |
| ▼ Coordinates: | | | | |
| **time** | (time) | datetime64[ns] | 2002-04-01 ... 2021-11-01 | |
| **lon** | (lon) | int64 | 0 2 4 6 8 ... 350 352 354 356 358 | |
| **lat** | (lat) | float64 | -90.0 -88.0 -86.0 ... 86.0 88.0 | |
| ▼ Data variables: | | | | |
| precipitation | (time, lat, lon) | float64 | ... | |
| ▶ Attributes: (0) | | | | |

```
df_grace
```

xarray.Dataset

▶ Dimensions: (**lat**: 90, **lon**: 180, **time**: 236)

▼ Coordinates:

| | | | | |
|---|---|---|---|---|
| **time** | (time) | datetime64[ns] | 2002-04-01 ... 2021-11-01 | |
| **lon** | (lon) | int64 | 0 2 4 6 8 ... 350 352 354 356 358 | |
| **lat** | (lat) | float64 | -90.0 -88.0 -86.0 ... 86.0 88.0 | |

▼ Data variables:

| | | | | |
|---|---|---|---|---|
| lwe_thickness | (time, lat, lon) | float64 | ... | |

▶ Attributes: (0)

```
df_grace = df_grace['lwe_thickness'][:,25:66,10:71]
df_grace
```

xarray.DataArray 'lwe_thickness' (**time**: 236, **lat**: 41, **lon**: 61)

```
[590236 values with dtype=float64]
```

▼ Coordinates:

| | | | | |
|---|---|---|---|---|
| **time** | (time) | datetime64[ns] | 2002-04-01 ... 2021-11-01 | |
| **lon** | (lon) | int64 | 20 22 24 26 28 ... 134 136 138 140 | |
| **lat** | (lat) | float64 | -40.0 -38.0 -36.0 ... 38.0 40.0 | |

▶ Attributes: (0)

```
df_grace[0]
```

xarray.DataArray 'lwe_thickness' (**lat**: 41, **lon**: 61)

```
array([[ -4.820502,  -4.86268 ,  -4.722223, ...,  -1.114338,  -1.147498,
         -1.73444 ],
       [ -3.360285,  -4.353694,  -4.195386, ...,  -1.221529,  -1.99629 ,
         -3.216284],
       [ -1.494723,  -2.233424,  -3.287187, ...,  -3.25616 ,  -3.630314,
         -1.493465],
       ...,
       [  1.336078,   2.07022 ,   3.497176, ...,   1.111395,   0.682665,
          1.516902],
       [  1.413195,   8.259285,   8.574983, ...,  -3.197883,  -0.446197,
         27.000797],
       [  7.10087 ,  10.261107,  10.605567, ...,  -1.413481, -18.366465,
         -7.015009]])
```

▼ Coordinates:

```
df_preci = df_preci['precipitation'][:,25:66,10:71]
df_preci
```

xarray.DataArray 'precipitation' (**time**: 236, **lat**: 41, **lon**: 61)

```
[590236 values with dtype=float64]
```

▼ Coordinates:

| | | | | |
|---|---|---|---|---|
| **time** | (time) | datetime64[ns] | 2002-04-01 ... 2021-11-01 | |
| **lon** | (lon) | int64 | 20 22 24 26 28 ... 134 136 138 140 | |
| **lat** | (lat) | float64 | -40.0 -38.0 -36.0 ... 38.0 40.0 | |

▶ Attributes: (0)

```
df_preci[0]
```

xarray.DataArray 'precipitation' (**lat**: 41, **lon**: 61)

```
array([[2.446428, 2.346867, 1.993499, ..., 0.684222, 0.578498, 0.589409],
       [1.524702, 1.621783, 1.537544, ..., 0.528166, 0.456726, 0.445193],
       [0.85676 , 0.958198, 1.021826, ..., 0.463121, 0.411438, 0.35833 ],
       ...,
       [1.972414, 2.29273 , 1.837716, ..., 4.396911, 3.6905  , 3.253795],
       [1.885994, 2.106746, 2.204469, ..., 3.292836, 3.594714, 3.648915],
       [1.695725, 2.141913, 2.767659, ..., 2.042926, 2.716503, 3.196219]])
```

▼ Coordinates:

```
df_temp = df_temp['temperature'][:,25:66,10:71]
df_temp
```

xarray.DataArray 'temperature' (**time**: 236, **lat**: 41, **lon**: 61)

```
[590236 values with dtype=float64]
```

▼ Coordinates:

| | | | |
|---|---|---|---|
| **time** | (time) | datetime64[ns] | 2002-04-01 ... 2021-11-01 |
| **lon** | (lon) | int64 | 20 22 24 26 28 ... 134 136 138 140 |
| **lat** | (lat) | float64 | -40.0 -38.0 -36.0 ... 38.0 40.0 |

► Attributes: (0)

```
df_temp[0]
```

xarray.DataArray 'temperature' (**lat**: 41, **lon**: 61)

```
array([[ 1.352317,  1.144489,  1.058138, ..., -0.335429, -0.251992, -0.139329],
       [ 1.027193,  0.755376,  0.648633, ..., -0.48993 , -0.417516, -0.208664],
```

```
df_ssh = df_ssh['zos'][:,25:66,10:71]
df_ssh
```

xarray.DataArray 'zos' (**time**: 105, **lat**: 41, **lon**: 61)

```
[262605 values with dtype=float64]
```

▼ Coordinates:

| | | | | |
|---|---|---|---|---|
| **time** | (time) | datetime64[ns] | 2002-04-01 ... 2010-12-01 | |
| **lon** | (lon) | int64 | 20 22 24 26 28 ... 134 136 138 140 | |
| **lat** | (lat) | float64 | -40.0 -38.0 -36.0 ... 38.0 40.0 | |

► Attributes: (0)

```
df_ssh[0]
```

xarray.DataArray 'zos' (**lat**: 41, **lon**: 61)

```
array([[0.863975, 0.732271, 0.409582, ..., 0.494863, 0.483598, 0.476015],
       [0.756291, 1.010434, 0.971179, ..., 0.506403, 0.491956, 0.489491],
       [0.297212, 0.558802, 0.917073, ..., 0.508088, 0.495943,      nan],
       ...,
       [     nan,      nan,      nan, ..., 0.653466,      nan,      nan],
       [     nan,      nan,      nan, ..., 0.536126, 0.537222,      nan],
       [     nan,      nan,      nan, ..., 0.440976, 0.448936, 0.438511]])
```

▼ Coordinates:

| | | | | |
|---|---|---|---|---|
| time | () | datetime64[ns] | 2002-04-01 | |
| **lon** | (lon) | int64 | 20 22 24 26 28 ... 134 136 138 140 | |
| **lat** | (lat) | float64 | -40.0 -38.0 -36.0 ... 38.0 40.0 | |

► Attributes: (0)

# Data Interpolation

## SSH

```
import numpy as np
import scipy.interpolate
```

```
loni = np.array(df_ssh['lon'])
lati = np.array(df_ssh['lat'])
```

```
X, Y = np.meshgrid(loni, lati)
XI, YI = np.meshgrid(loni,lati)
```

```
for i in range(0,105):
  df_ssh[i] = scipy.interpolate.griddata((X.flatten(),Y.flatten()),np.array(df_ssh[i]).flatten() , (XI,YI),method='cubic')
```

## Grace

```
df_grace
```

xarray.DataArray 'lwe_thickness' (**time**: 236, **lat**: 41, **lon**: 61)

```
loni = np.array(df_grace['lon'])
lati = np.array(df_grace['lat'])
```

```
X, Y = np.meshgrid(loni, lati)
XI, YI = np.meshgrid(loni,lati)
```

```
for i in range(0,236):
  df_grace[i] = scipy.interpolate.griddata((X.flatten(),Y.flatten()),np.array(df_grace[i]).flatten() , (XI,YI),method='cubic')
```

## Precip

```
loni = np.array(df_preci['lon'])
lati = np.array(df_preci['lat'])
```

```
X, Y = np.meshgrid(loni, lati)
XI, YI = np.meshgrid(loni,lati)
for i in range(0,236):
  df_preci[i] = scipy.interpolate.griddata((X.flatten(),Y.flatten()),np.array(df_preci[i]).flatten() , (XI,YI),method='cubic')
```

## Normalization

## SSH

```
ssh_mini = 999999
ssh_maxi = -999999

for i in range(0,105):
  ssh_maxi = np.maximum(ssh_maxi,np.amax(np.array(df_ssh[i]),where=~np.isnan(np.array(df_ssh[i])), initial=-1))
  ssh_mini = np.minimum(ssh_mini,np.amin(np.array(df_ssh[i]),where=~np.isnan(np.array(df_ssh[i])), initial=-1))


print(ssh_mini, ssh_maxi)
```

```
-1.0 -1.0
```

```
for i in range(0,105):
  x = df_ssh[i]
  x = (x-ssh_mini)/(ssh_maxi-ssh_mini)
  df_ssh[i] = x


df_ssh[0]
```

xarray.DataArray 'zos' (**lat**: 41, **lon**: 61)

```
array([[nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       ...,
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan]])
```

▼ Coordinates:

| | | | |
|---|---|---|---|
| time | () | datetime64[ns] | 2002-04-01 |
| **lon** | (lon) | int64 | 20 22 24 26 28 ... 134 136 138 140 |
| **lat** | (lat) | float64 | -40.0 -38.0 -36.0 ... 38.0 40.0 |

▶ Attributes: (0)

## Temperature

```
temp_mini = 999999
temp_maxi = -999999
for i in range(0,236):
  temp_maxi = np.maximum(temp_maxi,np.amax(np.array(df_temp[i]),where=~np.isnan(np.array(df_temp[i])), initial=-1))
  temp_mini = np.minimum(temp_mini,np.amin(np.array(df_temp[i]),where=~np.isnan(np.array(df_temp[i])), initial=-1))
for i in range(0,105):
  x = df_temp[i]
  x = (x-temp_mini)/(temp_maxi-temp_mini)
  df_temp[i] = x
```

```
temp_maxi
```

```
7.646506613402127
```

```
temp_mini
```

```
-9.476898891484783
```

## Precipitation

```
preci_mini = 999999
preci_maxi = -999999
for i in range(0,236):
  preci_maxi = np.maximum(preci_maxi,np.amax(np.array(df_preci[i]),where=~np.isnan(np.array(df_preci[i])), initial=-1))
  preci_mini = np.minimum(preci_mini,np.amin(np.array(df_preci[i]),where=~np.isnan(np.array(df_preci[i])), initial=-1))
for i in range(0,105):
  x = df_preci[i]
  x = (x-preci_mini)/(preci_maxi-preci_mini)
  df_preci[i] = x
```

## Grace

```
grace_mini = 999999
grace_maxi = -999999
for i in range(0,236):
  grace_maxi = np.maximum(grace_maxi,np.amax(np.array(df_grace[i]),where=~np.isnan(np.array(df_grace[i])), initial=-1))
  grace_mini = np.minimum(grace_mini,np.amin(np.array(df_grace[i]),where=~np.isnan(np.array(df_grace[i])), initial=-1))
for i in range(0,105):
  x = df_grace[i]
  x = (x-grace_mini)/(grace_maxi-grace_mini)
  df_grace[i] = x


df_grace= df_grace.fillna(0)
df_preci= df_preci.fillna(0)
df_temp= df_temp.fillna(0)
df_ssh= df_ssh.fillna(0)
```

## Data procressing

```
data_0 = np.array([df_grace[0],df_ssh[0], df_temp[0]]).T


data_0.shape
```

```
(61, 41, 3)
```

```
lat = df_grace['lat']
lon = df_grace['lon']
time = df_grace['time'][0]


m_data = [data_0]
for i in range(1,105):
  data_i = np.array([df_grace[i],df_ssh[i], df_temp[i]]).T
```

```
  m_data = np.append(np.array(m_data),[data_i],axis=0)
```

```
m_data.shape
```

```
     (105, 61, 41, 3)
```

```
m_data_x = m_data
m_data_y = df_preci[0:105]
```

```
X_train, X_test, y_train, y_test = train_test_split(m_data_x, m_data_y, test_size=0.10, random_state=42)
```

```
X_train[0].shape
```

```
     (61, 41, 3)
```

```
y_train[0].shape
```

```
     (41, 61)
```

```
X_train, X_val, y_train, y_val = train_test_split(X_train[:], y_train[:], test_size=0.10, random_state=42)
```

```
np.random.seed(42)

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='sigmoid',
                        input_shape=(61,41,3)))

model.add(layers.Conv2D(56, (2, 2), activation='sigmoid'))

model.add(layers.Conv2D(64, (2, 2), activation='sigmoid'))
```

```
model.add(layers.Flatten())
model.add(layers.Dense(2501, activation='sigmoid'))
model.add(tf.keras.layers.Reshape((41, 61)))

model.compile(loss='mean_squared_error', optimizer="adam", metrics=['mse'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 59, 39, 32)        896

 conv2d_1 (Conv2D)           (None, 58, 38, 56)        7224

 conv2d_2 (Conv2D)           (None, 57, 37, 64)        14400

 flatten (Flatten)           (None, 134976)            0

 dense (Dense)               (None, 2501)              337577477

 reshape (Reshape)           (None, 41, 61)            0

=================================================================
Total params: 337,599,997
Trainable params: 337,599,997
Non-trainable params: 0
_________________________________________________________________
```

```
X_train[0].shape
```

```
(61, 41, 3)
```

```
history = model.fit(X_train,
                    y_train,
```

```
                epochs=10,
                validation_data=(X_val, y_val))
```

```
WARNING:tensorflow:Keras is training/fitting/evaluating on array-like data. Keras may not be optimized for this format, so if y
Epoch 1/10
3/3 [==============================] - 10s 3s/step - loss: 0.0156 - mse: 0.0156 - val_loss: 0.0143 - val_mse: 0.0143
Epoch 2/10
3/3 [==============================] - 10s 3s/step - loss: 0.0156 - mse: 0.0156 - val_loss: 0.0143 - val_mse: 0.0143
Epoch 3/10
3/3 [==============================] - 10s 3s/step - loss: 0.0156 - mse: 0.0156 - val_loss: 0.0143 - val_mse: 0.0143
Epoch 4/10
3/3 [==============================] - 10s 3s/step - loss: 0.0156 - mse: 0.0156 - val_loss: 0.0143 - val_mse: 0.0143
Epoch 5/10
3/3 [==============================] - 10s 3s/step - loss: 0.0156 - mse: 0.0156 - val_loss: 0.0143 - val_mse: 0.0143
Epoch 6/10
3/3 [==============================] - 10s 3s/step - loss: 0.0156 - mse: 0.0156 - val_loss: 0.0143 - val_mse: 0.0143
Epoch 7/10
3/3 [==============================] - 10s 3s/step - loss: 0.0156 - mse: 0.0156 - val_loss: 0.0143 - val_mse: 0.0143
Epoch 8/10
3/3 [==============================] - 10s 3s/step - loss: 0.0156 - mse: 0.0156 - val_loss: 0.0143 - val_mse: 0.0143
Epoch 9/10
3/3 [==============================] - 10s 3s/step - loss: 0.0156 - mse: 0.0156 - val_loss: 0.0143 - val_mse: 0.0143
Epoch 10/10
3/3 [==============================] - 10s 3s/step - loss: 0.0156 - mse: 0.0156 - val_loss: 0.0143 - val_mse: 0.0143
```

```
model.save_weights('./checkpoints/my_checkpoint')
```

```
loss, acc = model.evaluate(X_test, y_test, verbose=2)
print("Restored model, mse: {:5.2f}".format(acc))
```

```
WARNING:tensorflow:Keras is training/fitting/evaluating on array-like data. Keras may not be optimized for this format, so if y
1/1 - 0s - loss: 0.0161 - mse: 0.0161 - 493ms/epoch - 493ms/step
Restored model, mse:  0.02
```

✓ 0s completed at 11:23 PM