landUseClassif

November 23, 2021

1 Usage De Sol Par Classification Supervisee (Imagerie Sentinel 2)

Je vais suivre les etapes suivantes: 1. Import de l'image et la transformer en table (Dataframe) 2. Import des zones de verite terrain + Transformation + Division en Train/Test 3. Entrainement du modele 4. Evaluation de la performance

1.1 1. Import de l'image Sentinel-2

[158]:
```
import rasterio
import rasterio.features
import rasterio.warp
from matplotlib import pyplot
from rasterio.plot import show
import pandas as pd
import numpy as np

with rasterio.open('./train-test-data/4BANDS-SOURCE.tif') as src:
    #Lire l'image
    array = src.read()

#creation d'array numpy
array = np.array(array)

sentinel2 = pd.DataFrame(array.reshape([4,-1]).T)
```

1.2 2. Import des zones de verite terrain

- Eau **(0)**

[86]:
```
eau_ = rio.open("./train-test-data/WATER_.tif")

#Lire l'image
array_ea = eau_.read()
#creation d'array numpy
array_ea = np.array(array_ea)

eau = pd.DataFrame(array_ea.reshape([4,-1]).T)
```

```
eau = eau[eau[0] != 0]
eau['target'] = 0
```

- Vegetation **(1)**

[87]:
```
vegetation_ = rio.open("./train-test-data/VEGETATION_.tif")

#Lire l'image
array_vg = vegetation_.read()
#creation d'array numpy
array_vg = np.array(array_vg)

vegetation = pd.DataFrame(array_vg.reshape([4,-1]).T)
vegetation = vegetation[vegetation[0] != 0]
vegetation['target'] = 1
```

- Urbain **(2)**

[88]:
```
urbain_ = rio.open("./train-test-data/URBAN_.tif")

#Lire l'image
array_ur = urbain_.read()
#creation d'array numpy
array_ur = np.array(array_ur)

urbain = pd.DataFrame(array_ur.reshape([4,-1]).T)
urbain = urbain[urbain[0] != 0]
urbain['target'] = 2
```

- Terrain nu **(3)**

[98]:
```
bareland_ = rio.open("./train-test-data/BARELAND_.tif")

#Lire l'image
array_bl = bareland_.read()
#creation d'array numpy
array_bl = np.array(array_bl)

bareland = pd.DataFrame(array_bl.reshape([4,-1]).T)
bareland = bareland[bareland[0] != 0]
bareland['target'] = 3
```

[100]:
```
frames = [eau,vegetation, urbain, bareland]

data = pd.concat(frames)
```

1.3 3. Entrainement du modele

[121]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

X = data.drop("target", axis=1)
y = data["target"]

# scaler = StandardScaler().fit(X)
# X_scaled = scaler.transform(X)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.30)
```

[122]:
```
from sklearn.neighbors import KNeighborsClassifier

# K-NNC
knn = KNeighborsClassifier(n_neighbors=9)

knn.fit(X_train, y_train)

# Predict the labels of test data

knn_pred = knn.predict(X_test)
```

1.4 4. Evaluation de la performance

[123]:
```
print(f"Accuracy: {accuracy_score(y_test, knn_pred)*100}")

print(classification_report(y_test, knn_pred))
```

```
Accuracy: 99.57829116244959
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      8878
           1       0.99      1.00      0.99       610
           2       1.00      0.99      0.99      6783
           3       0.99      1.00      0.99      5545

    accuracy                           1.00     21816
   macro avg       0.99      1.00      0.99     21816
weighted avg       1.00      1.00      1.00     21816
```

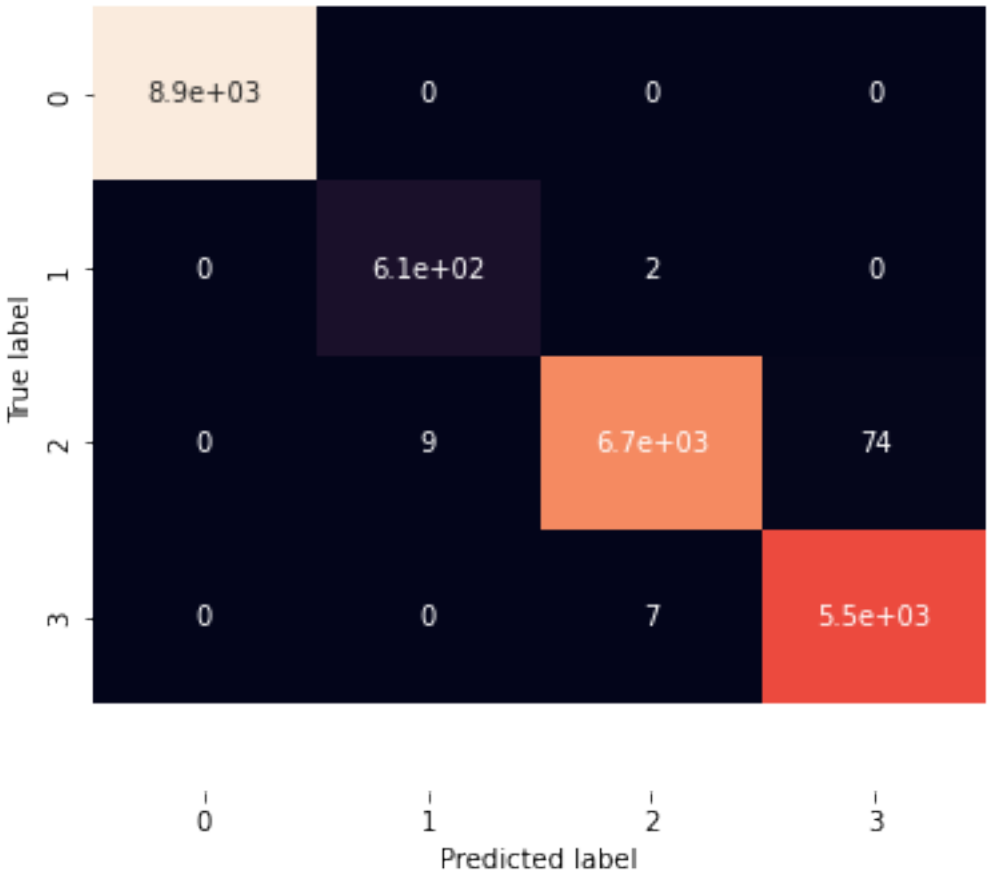
[124]:
```
from sklearn.metrics import confusion_matrix

# Create a confusion matrix using the confusion_matrix function
confusion_matrix(y_test, knn_pred)
# Import seaborn for improving visualisation of confusion matrix
import seaborn as sns

# Make confusion matrix more visual
def plot_conf_mat(y_test, y_preds):
    """
    Plots a confusion matrix using Seaborn's heatmap().
    """
    fig, ax = plt.subplots(figsize=(6, 6))
    ax = sns.heatmap(confusion_matrix(y_test, y_preds),
                     annot=True, # Annotate the boxes
                     cbar=False)
    plt.xlabel("Predicted label")
    plt.ylabel("True label")

    # Fix the broken annotations (this happened in Matplotlib 3.1.1)
    bottom, top = ax.get_ylim()
    ax.set_ylim(bottom + 0.5, top - 0.5);

plot_conf_mat(y_test, knn_pred)
```



[ ]:
```
s = sentinel2.to_numpy().reshape([4, 3527, 2996])
full = knn.predict(sentinel2)
```

[164]:
```
full = full.reshape([1 ,3527, 2996])
```

[165]:
```
profile = src.profile
profile.update(dtype=rasterio.uint8, count=1, compress='lzw')

with rasterio.open('./output.tif', 'w', **profile) as dst:
    dst.write(full.astype(rasterio.uint8))
```

[177]:
```
#import required libraries
import rasterio
from rasterio import plot
import matplotlib.pyplot as plt
from rasterio.plot import show
```

```
%matplotlib inline

#open the raster
src=rasterio.open('./output.tif')

#display one band:
fig, ax = plt.subplots(figsize=(16, 16))
img = ax.imshow(src.read(1), cmap='terrain')
fig.colorbar(img, ax=ax)
```

[177]: <matplotlib.colorbar.Colorbar at 0x1f0ba198c10>