Random Forest Final Backup

April 16, 2023

[ ]:
```
import numpy as np
import gdal
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import rasterio
from rasterio.plot import show
```

[ ]:
```
# define inputs
inpRaster = 'C:/Users/Minteb2yobb/Desktop/RF/NewHypermap.tif'
outRaster = 'RFClassification.tif'
df = pd.read_csv('C:/Users/Minteb2yobb/Desktop/RF/Minte_Training_used.csv')
```

[ ]:
```
# read training data and labels
data =␣
 ↪df[['Band_1','Band_2','Band_3','Band_4','Band_5','Band_6','Band_7','Band_8','Band_9','Band_
label = df['id']
del df

# split the data into training and validation sets
from sklearn.model_selection import train_test_split
train_data, val_data, train_label, val_label = train_test_split(data, label,␣
 ↪test_size=0.2, random_state=42)
```

[ ]:
```
# open raster file
ds = gdal.Open(inpRaster, gdal.GA_ReadOnly)

# get raster info
rows = ds.RasterYSize
cols = ds.RasterXSize
bands = ds.RasterCount
geo_transform = ds.GetGeoTransform()
projection = ds.GetProjectionRef()

# read raster data as array and reshape it
array = ds.ReadAsArray()
ds = None
```

```
array = np.stack(array, axis=2)
array = np.reshape(array, [rows*cols, bands])
test = pd.DataFrame(array, dtype='int16')
del array
```

```
C:\Users\Minteb2yobb\AppData\Local\Temp\ipykernel_16932\2579082112.py:16:
FutureWarning: In a future version, passing float-dtype values and an integer
dtype to DataFrame will retain floating dtype if they cannot be cast losslessly
(matching Series behavior). To retain the old behavior, use
DataFrame(data).astype(dtype)
  test = pd.DataFrame(array, dtype='int16')
```

[ ]:
```
# train classifier
clf = RandomForestClassifier(n_estimators=100, n_jobs=-1, oob_score=True)
clf.fit(train_data, train_label)
del train_data, train_label
```

[ ]:
```
# Evaluate the model's OOB score and print feature importances
print("OOB score:", clf.oob_score_)
importances = clf.feature_importances_
for i, importance in enumerate(importances):
    print(f"Band {i+1}: {importance:.3f}")
```

```
OOB score: 0.97
Band 1: 0.045
Band 2: 0.078
Band 3: 0.037
Band 4: 0.039
Band 5: 0.086
Band 6: 0.074
Band 7: 0.099
Band 8: 0.078
Band 9: 0.014
Band 10: 0.075
Band 11: 0.075
Band 12: 0.074
Band 13: 0.068
Band 14: 0.070
Band 15: 0.089
```

[ ]:
```
# predict classes for validation set and calculate accuracy
y_pred = clf.predict(val_data)
val_accuracy = accuracy_score(val_label, y_pred)
print("Validation accuracy:", val_accuracy)
```

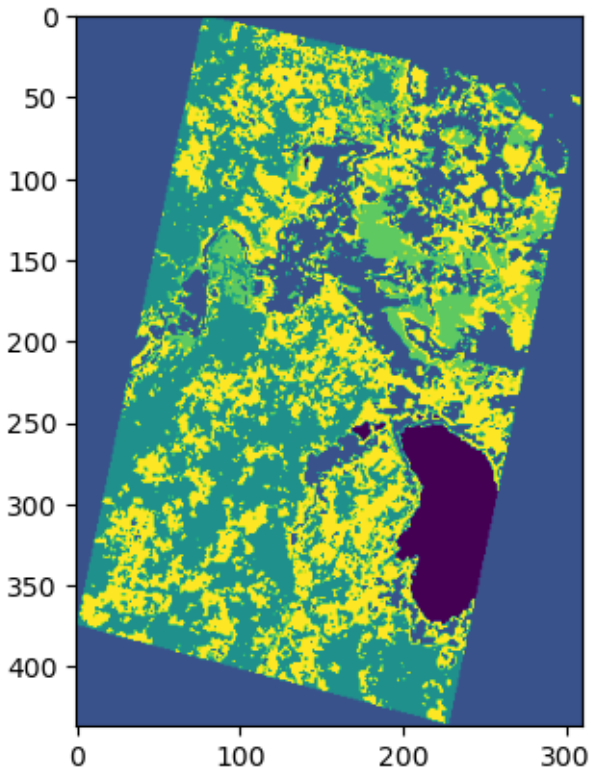
```
Validation accuracy: 0.98
```

[ ]:
```
# predict classes for the whole image
y_pred = clf.predict(test)
del test

# reshape the predicted classes to match the image dimensions
classification = y_pred.reshape((rows,cols))
del y_pred
```

```
c:\Users\Minteb2yobb\anaconda3\lib\site-packages\sklearn\base.py:450:
UserWarning: X does not have valid feature names, but RandomForestClassifier was
fitted with feature names
  warnings.warn(
```

[ ]:
```
# create a GeoTIFF file for the classified image
def createGeotiff(outRaster, data, geo_transform, projection):
    driver = gdal.GetDriverByName('GTiff')
    rows, cols = data.shape
    rasterDS = driver.Create(outRaster, cols, rows, 1, gdal.GDT_Int32)
    rasterDS.SetGeoTransform(geo_transform)
    rasterDS.SetProjection(projection)
    band = rasterDS.GetRasterBand(1)
    band.WriteArray(data)
    rasterDS = None

createGeotiff(outRaster, classification, geo_transform, projection)
```

[ ]:
```
# open the classified image using rasterio and display it
with rasterio.open('RFClassification.tif') as src:
    data = src.read(1)
    show(data, cmap='viridis')
```



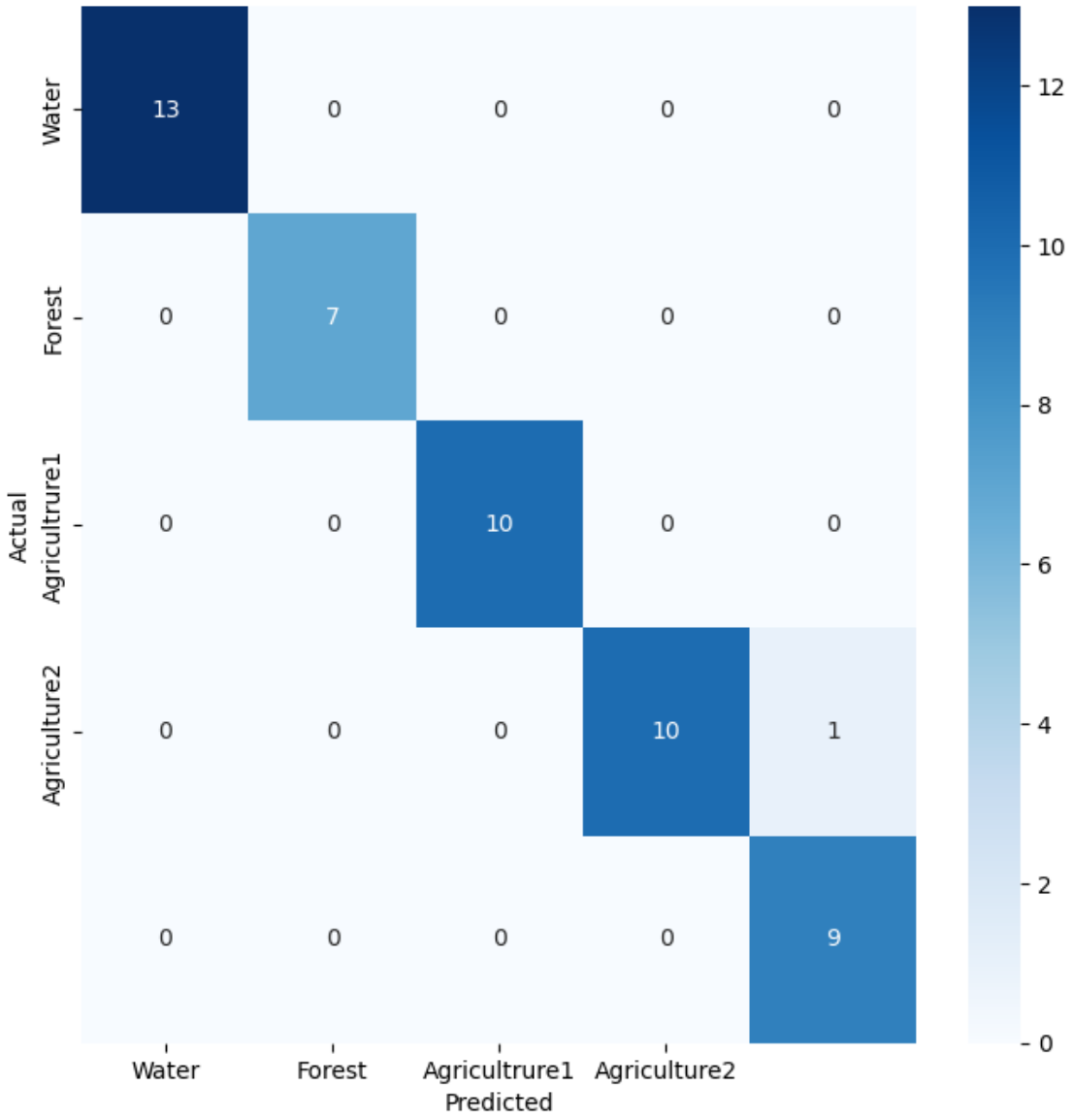
[ ]:
```
from sklearn.metrics import (accuracy_score, confusion_matrix,␣
 ↪classification_report)
```

[ ]:
```
y_TestPredicted = clf.predict(val_data)
print("Accuracy score:", accuracy_score(val_label,y_TestPredicted))
print("Confusion matrix:\n", confusion_matrix(val_label,y_TestPredicted))
print("Classification report:\n",␣
 ↪classification_report(val_label,y_TestPredicted))
```

```
Accuracy score: 0.98
Confusion matrix:
 [[13  0  0  0  0]
 [ 0  7  0  0  0]
 [ 0  0 10  0  0]
 [ 0  0  0 10  1]
 [ 0  0  0  0  9]]
Classification report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         7
           3       1.00      1.00      1.00        10
```

```
           4       1.00      0.91      0.95        11
           5       0.90      1.00      0.95         9

    accuracy                           0.98        50
   macro avg       0.98      0.98      0.98        50
weighted avg       0.98      0.98      0.98        50
```

[ ]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# calculate confusion matrix
cm = confusion_matrix(val_label, y_TestPredicted)
class_names = ['Water', 'Forest', 'Agricultrure1', 'Agriculture2']

# create heatmap
fig, ax = plt.subplots(figsize=(8, 8))
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d", xticklabels=class_names,␣
 ↪yticklabels=class_names)
ax.set_xlabel("Predicted")
ax.set_ylabel("Actual")
plt.show()
```



[ ]:
```
classes = {
    1 : ("Water", "#0000FF"),
    2 : ("Forest", "#006400"),
    3 : ("Uncultivated Land", "#9B870C"),
    4 : ("Agricultural Land", "#FFFF00"),
    5 : ("Agricultural Land2", "#FFFF00")}


classes_colors = []
classes_labels = []
```

```
for key in classes:
    values = classes.get(key)
    label = values[0]
    color = values[1]

    classes_labels.append(label)
    classes_colors.append(color)

n_classes = classification.max()
```

[ ]:
```
fig, ax = plt.subplots(figsize=(15, 15), subplot_kw={"xticks": [], "yticks":␣
 ↪[]})
cmap = plt.matplotlib.colors.ListedColormap(classes_colors, N=n_classes)␣
 ↪#colormap
cax = ax.imshow(classification,cmap=cmap)

plt.title("RF-Classification of Postfire Scene - Paradise, CA (10.12.2018)",␣
 ↪fontsize=22, pad=15)
cbar = fig.colorbar(cax, ax=ax, fraction=0.034, pad=0.065)
cbar.ax.set_yticklabels(classes_labels)
plt.show()
```

```
C:\Users\Minteb2yobb\AppData\Local\Temp\ipykernel_16932\4132391990.py:7:
UserWarning: FixedFormatter should only be used together with FixedLocator
  cbar.ax.set_yticklabels(classes_labels)
```

RF-Classification of Postfire Scene - Paradise, CA (10.12.2018)