RF_hyper parameter

March 28, 2024

1 Project title:-

Advancing Earth Observation Data and ResUNet-Deep Learning Model for Irrigated Area Mapping: The Case of Along the Awash Valley, Ethiopia

2 Grid search Hyperparameter optimization to get the best Random Forest (RF) parameters and model

This Jupyter notebook demonstrates how to apply Grid search hyperparameter optimization to get the best RF parameters and model with the ESA EO-Africa inovation lab cloud computing environment.

Prerequisites for running this notebook

Several packages need to be installed and/or imported for running this script:

The `rasterio, sklearn,and geopandas` modules should be installed first to apply Grid search Hyperparameter optimization;

2.1 Import the Relevant Modules

[1]:
```
import rasterio
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score
import geopandas as gpd
```

2.2 Load Sentinel-2 image

[2]:
```
sentinel_image_path = "/home/eoafrica/Sentinel2_AWbasin/sentinel2_layerstack/
 ↪stacked_image.tif"
with rasterio.open(sentinel_image_path) as src:
    sentinel_image = src.read()
```

2.3 Load ground truth GCP shapefile data

[3]:
```
gcp_shapefile_path = "/home/eoafrica/Sentinel2_AWbasin/GCP_LULCawash/lulcgcp.
↪shp"
gcp_data = gpd.read_file(gcp_shapefile_path)
```

2.4 Extract the values of the image pixels at the locations of the GCPs

[4]:
```
gcp_points = gcp_data.geometry.apply(lambda geom: (geom.x, geom.y)).tolist()
gcp_values = []
for point in gcp_points:
    row, col = src.index(point[0], point[1])
    gcp_values.append(sentinel_image[:, row, col])

gcp_values = np.array(gcp_values)
```

2.5 Extract corresponding class labels from the GCP data

[5]:
```
class_labels = gcp_data["class_labe"].values.astype(int)
```

2.6 Split data into training and testing sets

[6]:
```
X_train, X_test, y_train, y_test = train_test_split(gcp_values, class_labels,␣
↪test_size=0.3, random_state=42)
```

2.7 Define parameter grid for grid search

[7]:
```
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

2.8 Initialize Random Forest classifier

[8]:
```
rf_classifier = RandomForestClassifier()
```

2.9 Perform grid search

[9]:
```
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid,␣
↪cv=3, scoring='accuracy')
grid_search.fit(X_train, y_train)
```

[9]:
```
GridSearchCV(cv=3, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [None, 10, 20],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [100, 200, 300]},
             scoring='accuracy')
```

2.10 Get the best parameters and model

[12]:
```
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
```

2.11 Evaluate the model

[13]:
```
y_pred = best_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Best Parameters:", best_params)
print("Accuracy:", accuracy)
```

```
Best Parameters: {'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split':
2, 'n_estimators': 200}
Accuracy: 0.5925925925925926
```

[ ]:
```

```