

# SVM\_hyperparameters

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 Support Vector Machine (SVM) parameters and model

This Jupyter notebook demonstrates how to apply Grid search hyperparameter optimization to get the best SVM parameters and model with the ESA EO-Africa innovation 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;

### 2.0.1 Import Relevant Packages

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

### 2.0.2 Load Sentinel-2 MSI image

```
[2]: sentinel_image_path = "/home/eoafrika/Sentinel2_AWbasin/outputs_rgb/stacked_rgb.
    ↪tif"
with rasterio.open(sentinel_image_path) as src:
    sentinel_image = src.read()
```

### 2.0.3 Load ground truth GCP shapefile data

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

### 2.0.4 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)
```

### 2.0.5 Extract corresponding class labels from the GCP data

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

### 2.0.6 Split data into training and testing sets

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

### 2.0.7 Define parameter grid for grid search

```
[8]: param_grid = {  
      'C': [0.1, 1, 10],  
      'kernel': ['linear', 'rbf'],  
      'gamma': ['scale', 'auto']  
      }
```

### 2.0.8 Initialize Support Vector Classifier

```
[9]: svm_classifier = SVC()
```

### 2.0.9 Perform grid search

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

### 2.0.10 Get the best parameters and model

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

### 2.0.11 Evaluate the model

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

```
[ ]:
```