Prirodno-matematički fakultet
Informatika

Projekat iz predmeta Mikroprocesorski sistemi

“Detekcija bespravne seče šume i alarm”

Student:
Nikola Vukašinović 27/2019

Profesor:
dr Aleksandar Peulić

Avgust/Septembar 2024.

Sadržaj

1. Uvod

Bespravna seča šuma predstavlja veliki problem, kako zbog nestanka prirodnog staništa mnogih biljaka i životinja, tako i zbog materijalnih gubitaka vlasnika tih šuma. Naime ova nelegalna aktivnost se uglavnom i događa upravo zbog tih materijalnih dobitaka pojedinaca ili određene grupe ljudi na račun vlasnika šuma kao i životinja čiji je to dom.

U ovom projektu pokušao sam da napravim jednostavan sistem za praćenje bespravne seče šuma određenih oblasti preko satelitskih snimaka koristeći Python za kodiranje i slanje alarma, kao i Google Earth Engine za analizu satelitskih snimaka.

2. Analiza koda

```
import numpy as np
import pandas as pd
import cv2  # OpenCV
import rasterio
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import ee  # Earth Engine API
import geemap
import os
import leaflet
import ipywidgets as widgets

import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import MinMaxScaler

import serial
```

U prvom jupyter notebook delu za kod se nalazi import svih potrebnih biblioteka na jednom mestu.

2.1. Učitavanje satelitskih snimaka, prikaz i čuvanje podataka

```
#os.remove(os.path.expanduser('~/.config/earthengine/credentials'))
ee.Authenticate()
ee.Initialize(project='nikolavukasinovic')

# Definišemo područje interesa (AOI) - Stavio da bude moje selo - to ce da bude test
aoi = ee.Geometry.Polygon([
    [
        [20.611538, 43.045487],  # Ugao 1 (Jugo Zapadna granica)
        [20.619155, 43.045487],  # Ugao 2 (Jugo Istočna granica)
        [20.619155, 43.051737],  # Ugao 3 (Severo Istočna granica)
        [20.611538, 43.051737]   # Ugao 4 (Severo Zapadna granica)
    ]
])

# Definišemo drugo područje interesa (AOI) - Stavio da bude pokrcena suma u Amazonu - train
aoi2 = ee.Geometry.Polygon([
    [
        [-60.230, -3.540],  # Ugao 1 (Jugo Zapadna granica)
        [-60.220, -3.540],  # Ugao 2 (Jugo Istočna granica)
        [-60.220, -3.530],  # Ugao 3 (Severo Istočna granica)
        [-60.230, -3.530]   # Ugao 4 (Severo Zapadna granica)
    ]
])

# Definišemo drugo područje interesa (AOI) - Stavio da bude gusta vegetacija iz Severne amerike - train
aoi3 = ee.Geometry.Polygon([
    [
        [-83.540, 35.625],  # Ugao 1 (Jugo Zapadna granica)
        [-83.530, 35.625],  # Ugao 2 (Jugo Istočna granica)
        [-83.530, 35.635],  # Ugao 3 (Severo Istočna granica)
        [-83.540, 35.635]   # Ugao 4 (Severo Zapadna granica)
    ]
])
```

Prvi korak je napraviti projekat na Google earth engine cloud-u I povezati ga sa projektom iz VSC-a.

Nakon toga definišemo područja od interesa, izabrao sam 2 različite oblasti kako bi imali više različitih podataka za treniranje(aoi2, aoi3). Dok je aoi oblast za koju želimo da vidimo da li je bilo bespravne seče šuma pa ćemo nju koristiti za kasniju proveru.

```
def calculate_ndvi_evi(image):
    ndvi = image.normalizedDifference(['B8', 'B4']).rename('NDVI')
    evi = image.expression(
        '2.5 * ((NIR - RED) / (NIR + 6 * RED - 7.5 * BLUE + 1))',
        {
            'NIR': image.select('B8'),
            'RED': image.select('B4'),
            'BLUE': image.select('B2')
        }).rename('EVI')
    return image.addBands([ndvi, evi])


datasets = [
    ee.ImageCollection('COPERNICUS/S2_HARMONIZED').filterDate('2015-01-01', '2022-12-31').filterBounds(aoi2).map(calculate_ndvi_evi),
    ee.ImageCollection('COPERNICUS/S2_HARMONIZED').filterDate('2015-01-01', '2022-12-31').filterBounds(aoi3).map(calculate_ndvi_evi),
]

combined_dataset = ee.ImageCollection(datasets[0])

for dataset in datasets[1:]:
    combined_dataset = combined_dataset.merge(dataset)
images = combined_dataset.median()
```

Učitavanje slika korišćenjem Sentinel-2 Harmonized skupa podataka iz perioda [2015-2022] u gore navedenim područjima. Za sve filtrirane slike se koristi funkcija calculte_ndvi_evi koja računa NDVI(Normalized Difference Vegetation Index) i EVI (Enhanced Vegetation Index).

Pokušaj kombinovanja podataka sa oba područja u jednu sliku.

```
dataset_test = ee.ImageCollection('COPERNICUS/S2_HARMONIZED').filterDate('2015-01-01', '2022-12-31').filterBounds(aoi).map(calculate_ndvi_evi)
image = ee.Image(dataset_test.first())

vis_params = {
    'bands': ['B4', 'B3', 'B2'],  # RGB bands
    'min': 0,
    'max': 3000,
    'gamma': 1.4
}

# Pravimo mapu za vizuelizaciju
Map = geemap.Map()
Map.centerObject(aoi, 13)
Map.addLayer(image, vis_params, 'RGB Image')
Map.addLayer(aoi, {}, 'AOI')
Map.addLayerControl()
Map
```

Učitavanje i prikaz željenog područja za testiranje.

```python
def calculate_ndvi(image):
    # Racunamo NDVI: (NIR - Red) / (NIR + Red)
    ndvi = image.normalizedDifference(['B8', 'B4']).rename('NDVI')
    return image.addBands(ndvi)

# Uzimamo prosecnu sliku za period 2015-2022 za NDVI i EVI
median_ndvi = combined_dataset.select('NDVI').median()
median_evi = combined_dataset.select('EVI').median()

# Selo
median_ndvi_test = dataset_test.select('NDVI').median().clip(aoi)
median_evi_test = dataset_test.select('EVI').median().clip(aoi)

# Amazon
median_ndvi2 = datasets[0].select('NDVI').median()
median_evi2 = datasets[0].select('EVI').median()

# Amerika
median_ndvi3 = datasets[1].select('NDVI').median()
median_evi3 = datasets[1].select('EVI').median()

# Postavljamo parametre za NDVI (Normalized Difference Vegetation Index)
ndvi_params = {
    'min': -1,
    'max': 1,
    'palette': ['blue', 'white', 'green']
}

# Visualize NDVI
Map = geemap.Map()
Map.centerObject(aoi, 13)
Map.addLayer(median_ndvi_test, ndvi_params, 'Median NDVI')
Map.addLayer(aoi, {}, 'AOI')
Map.addLayerControl()
Map
```

✓ 1.6s

Funkcija calculate_ndvi koristimo kada želimo jednostavno da izračunamo vrednost samo za ndvi.

Pravimo prosečne slike za definisani period i definisana područja.

Prikaz mape sa prosečnim ndvi za testno područje.

```
images_folder = os.path.join(os.getcwd(), 'images')
export_path = os.path.join(images_folder, 'median_ndvi_test.tif')
export_path1 = os.path.join(images_folder, 'median_evi_test.tif')
export_path2 = os.path.join(images_folder, 'median_ndvi_train1.tif')
export_path3 = os.path.join(images_folder, 'median_evi_train1.tif')
export_path4 = os.path.join(images_folder, 'median_ndvi_train2.tif')
export_path5 = os.path.join(images_folder, 'median_evi_train2.tif')

# Exportujemo test slike na lokal
geemap.ee_export_image(
    median_ndvi_test,
    filename=export_path,
    scale=5,
    region=aoi.getInfo()['coordinates']
)

geemap.ee_export_image(
    median_evi_test,
    filename=export_path1,
    scale=5,
    region=aoi.getInfo()['coordinates']
)

# Exportujemo train slike na lokal
geemap.ee_export_image(
    median_ndvi2,
    filename=export_path2,
    scale=10,
    region=aoi2.getInfo()['coordinates']
)

geemap.ee_export_image(
    median_evi2,
    filename=export_path3,
    scale=10,
    region=aoi2.getInfo()['coordinates']
)
```

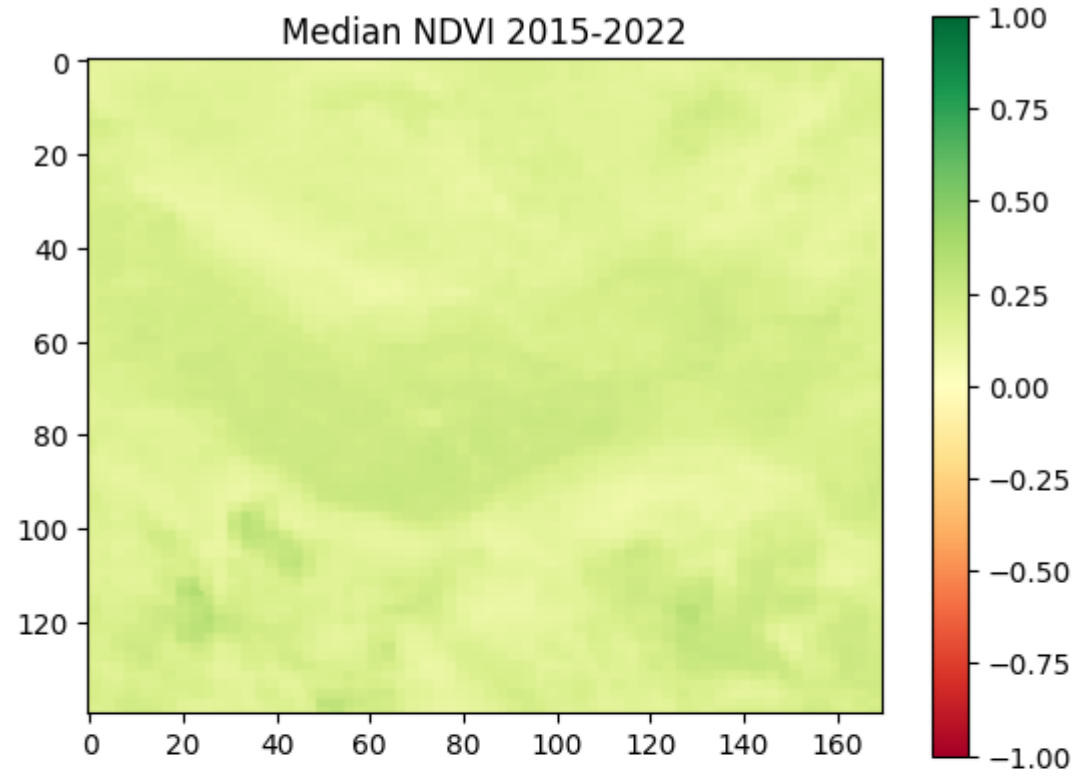
```
geemap.ee_export_image(
    median_ndvi3,
    filename=export_path4,
    scale=10,
    region=aoi3.getInfo()['coordinates']
)

geemap.ee_export_image(
    median_evi3,
    filename=export_path5,
    scale=10,
    region=aoi3.getInfo()['coordinates']
)
```

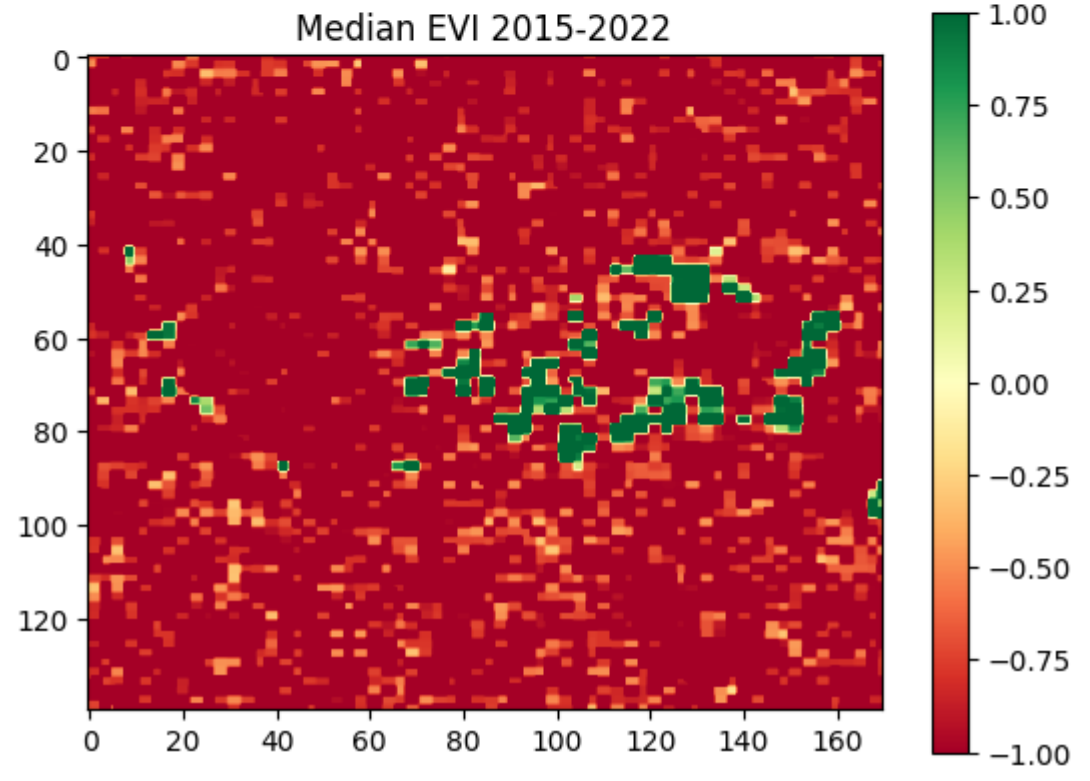
✓ 1m 33.7s

Exportovanje dobijenih slika sa prosečnim ndvi I evi vrednostima na lokalni uređaj.

```python
# Ucitavamo exportovanu NDVI test sliku
with rasterio.open(export_path) as src:
    ndvi_data_test = src.read(1)  # Citamo prvi band
    profile = src.profile

# Ucitavamo exportovanu EVI test sliku
with rasterio.open(export_path1) as src:
    evi_data_test = src.read(1)  # Citamo prvi band
    profile = src.profile

# Ucitavamo exportovanu NDVI train1 sliku
with rasterio.open(export_path2) as src:
    ndvi_data1 = src.read(1)  # Citamo prvi band
    profile = src.profile

# Ucitavamo exportovanu EVI train1 sliku
with rasterio.open(export_path3) as src:
    evi_data1 = src.read(1)  # Citamo prvi band
    profile = src.profile

# Ucitavamo exportovanu NDVI train2 sliku
with rasterio.open(export_path4) as src:
    ndvi_data2 = src.read(1)  # Citamo prvi band
    profile = src.profile

# Ucitavamo exportovanu EVI train2 sliku
with rasterio.open(export_path5) as src:
    evi_data2 = src.read(1)  # Citamo prvi band
    profile = src.profile

# Prikaz test NDVI slike
plt.imshow(ndvi_data_test, cmap='RdYlGn', vmin=-1, vmax=1)
plt.colorbar()
plt.title('Median NDVI 2015-2022')
plt.show()

# Prikaz test EVI slike
plt.imshow(evi_data_test, cmap='RdYlGn', vmin=-1, vmax=1)
plt.colorbar()
plt.title('Median EVI 2015-2022')
plt.show()
```

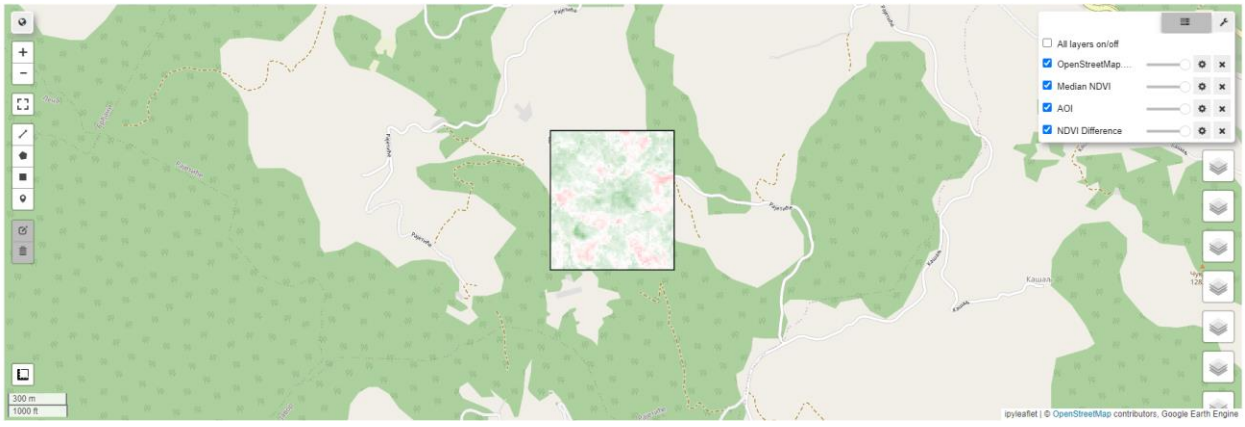
Učitavanje slika sa lokala pomoću biblioteke rasterio. Prikaz ndvi i evi slika za testno područje.


Median NDVI 2015-2022
0
20
40
60
80
100
120
0
20
40
60
80
100
120
140
160
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Median EVI 2015-2022
0
20
40
60
80
100
120
0
20
40
60
80
100
120
140
160
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

```
dataset2023 = ee.ImageCollection('COPERNICUS/S2_HARMONIZED') \
              .filterDate('2023-01-01', '2023-12-31') \
              .filterBounds(aoi)

dataset_2023_ndvi = dataset2023.map(calculate_ndvi)
median_ndvi_2023 = dataset_2023_ndvi.select('NDVI').median().clip(aoi)

dataset2015 = ee.ImageCollection('COPERNICUS/S2_HARMONIZED') \
              .filterDate('2015-01-01', '2015-12-31') \
              .filterBounds(aoi)

dataset_2015_ndvi = dataset2015.map(calculate_ndvi)
median_ndvi_2015 = dataset_2015_ndvi.select('NDVI').median().clip(aoi)

# Razlika izmedju dva perioda
ndvi_difference = median_ndvi_2023.subtract(median_ndvi_2015)


diff_params = {
    'min': -0.5,
    'max': 0.5,
    'palette': ['red', 'white', 'green']
}

Map.addLayer(ndvi_difference, diff_params, 'NDVI Difference')
Map.addLayerControl()
Map

✓ 0.5s
```

Vizuelno prikazana razlika za ndvi vrednosti za testno područje za 2015. i 2023. Godinu



2.2. Priprema i kreiranje modela RandomForest

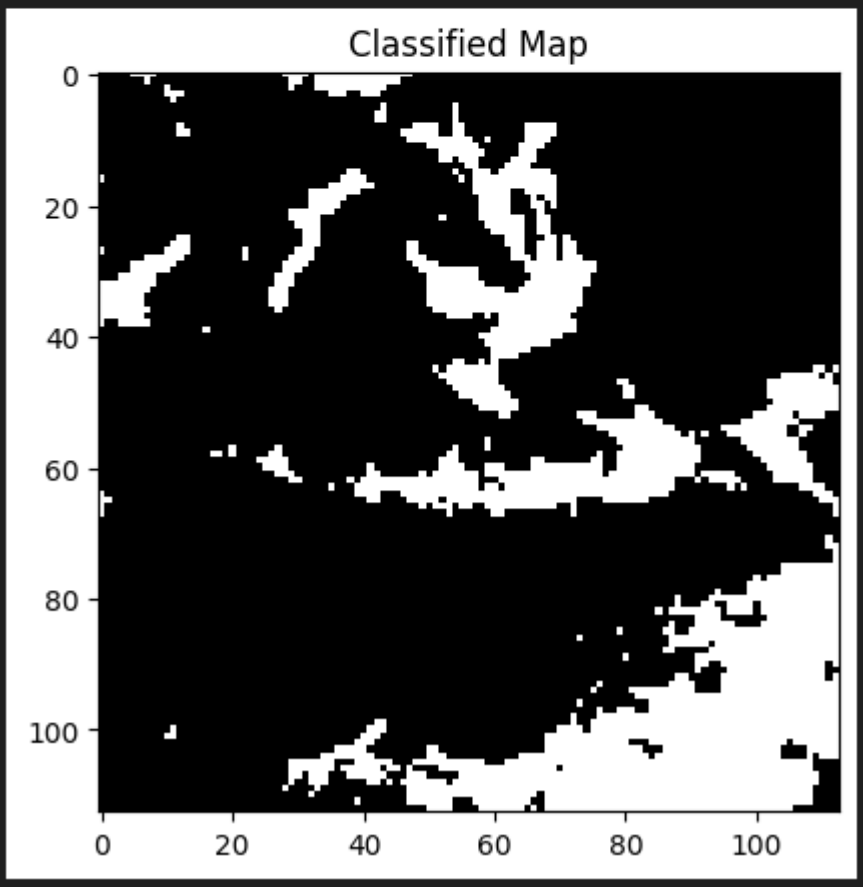
```
ndvi_data = np.median([ndvi_data1, ndvi_data2], axis=0)
evi_data = np.median([evi_data1, evi_data2], axis=0)

# Priprema feature-a za trening, kombinovanje NDVI i EVI u feature
X = np.vstack([
    ndvi_data.flatten(),
    evi_data.flatten()
]).T

# Gde su ndvi i evi manji od 0.21 racunacemo da je pokrceno(deforestation) a ostali ce da upadnu u sume
labels = np.where((ndvi_data < 0.21) & (evi_data < 0.21), 1, 0)
y = labels.flatten()  # Labels (0 i dalje sume, 1 pokrceno)

scaler = MinMaxScaler(feature_range=(0, 1))
X_scaled = scaler.fit_transform(X)

# Deljenje podataka na train i test skupove u odnosu 75/25
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.25, random_state=42)
```

Spajanje ndvi i evi vrednosti sa različitih područja, dodela labela 1 ako je pokrčeno, a 0 ako je šuma i dalje. Podela na trening i test skupove od čega 75% za trening skupa i 25% za testni skup.

```
# Inicijalizacija RandomForestClassifier sa 100 jedinica
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Treniranje modela
rf_classifier.fit(X_train, y_train)

# Prediktovanje
y_pred = rf_classifier.predict(X_test)

# Evaluacija modela
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Vizualizacija klasifikovane mape
classified_map = rf_classifier.predict(X_scaled).reshape(ndvi_data.shape)
plt.imshow(classified_map, cmap='gray')
plt.title('Classified Map')
plt.show()

✓ 0.5s
```

Inicijalizacija modela koristeći RandomForest tehniku za mašinsko učenje. Prikaz Konfuzione matrice modela i odgovarajućih parametara za proveru tačnosti modela.

```
Confusion Matrix:
[[2515    0]
 [   0  678]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2515
           1       1.00      1.00      1.00       678

    accuracy                           1.00      3193
   macro avg       1.00      1.00      1.00      3193
weighted avg       1.00      1.00      1.00      3193
```



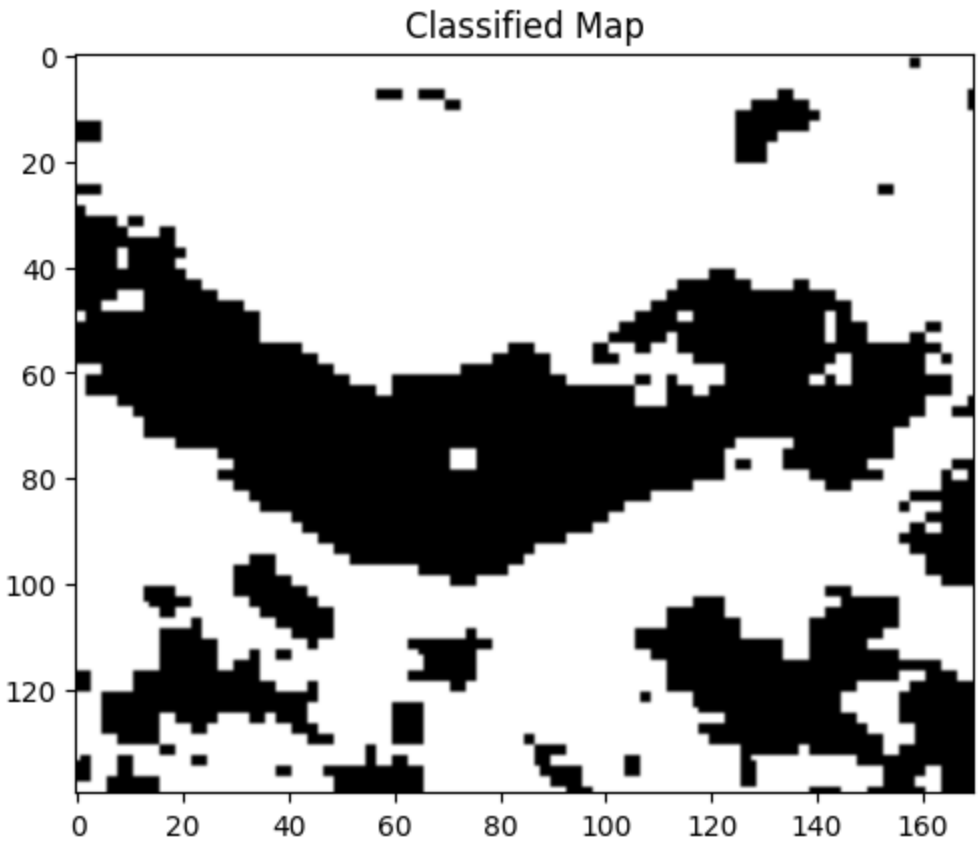
Nažalost moj model je overfittovan, tj previše se oslanja na podatke iz kojih je učio. Pa tako ima samo True Positive i True Negative vrednosti.

```python
new_X = np.vstack([
    ndvi_data_test.flatten(),
    evi_data_test.flatten()
]).T

new_X_scaled = scaler.transform(new_X)

labels2 = np.where((ndvi_data_test < 0.21) & (evi_data_test < 0.21), 1, 0)
y_pred2 = rf_classifier.predict(new_X_scaled)
y_true2 = labels2.flatten()

# Evaluacija modela na novim nepoznatim do sad podacima
print("Confusion Matrix:")
print(confusion_matrix(y_true2, y_pred2))

print("\nClassification Report:")
print(classification_report(y_true2, y_pred2))

new_classified_map = rf_classifier.predict(new_X_scaled).reshape(ndvi_data_test.shape)
plt.imshow(new_classified_map, cmap='gray')
plt.title('Classified Map')
plt.show()
```

Da proverimo ipak ono što nas zanima kako se pokazuje model na, njemu do sad, neviđenim podacima. Tj kako predviđa vrednosti iz našeg testnog područja od interesa.

```
Confusion Matrix:
[[ 6741   106]
 [ 1322 15631]]

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.98      0.90      6847
           1       0.99      0.92      0.96     16953

    accuracy                           0.94     23800
   macro avg       0.91      0.95      0.93     23800
weighted avg       0.95      0.94      0.94     23800
```

Vidimo da je stanje sada realnije nego kada se koriste podaci iz istog skupa za validaciju.


Classified Map
0
20
40
60
80
100
120
0
20
40
60
80
100
120
140
160

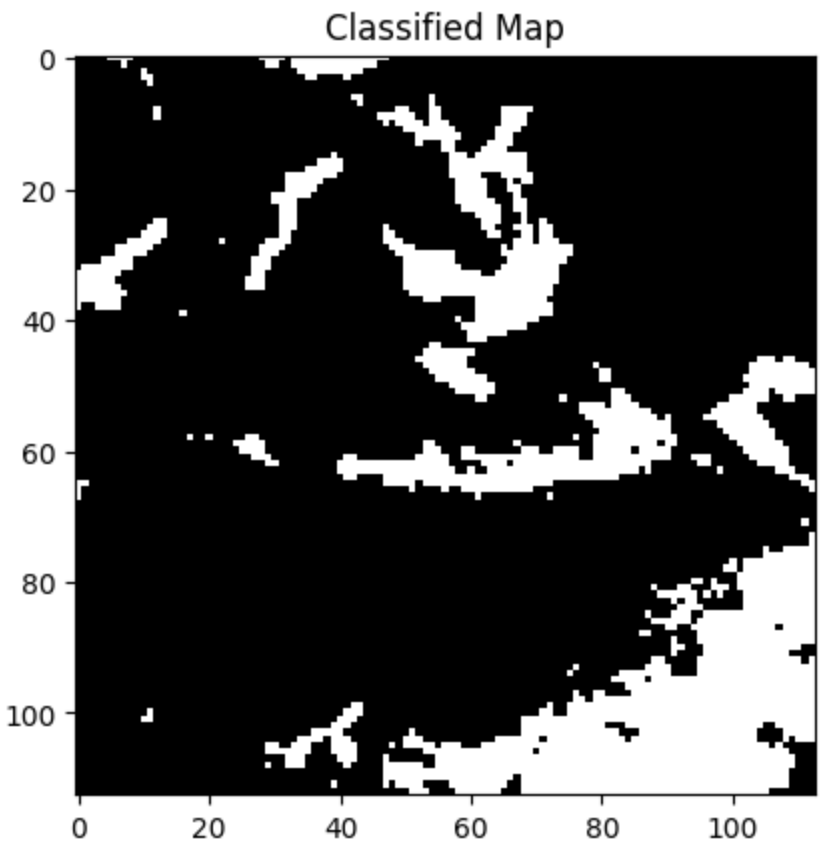
2.3. Kreiranje modela Logističke regresije

```python
# Inicijalizacija logisticke regresije
logistic_classifier = LogisticRegression(max_iter=1000, random_state=42)

# Treniarnje modela
logistic_classifier.fit(X_train, y_train)

# Predikcija
y_pred3 = logistic_classifier.predict(X_test)

# Provera modela
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Vizuelizacija rezultata logisticke regresije
classified_map2 = logistic_classifier.predict(X_scaled).reshape(ndvi_data.shape)
plt.imshow(classified_map2, cmap='gray')
plt.title('Classified Map')
plt.show()
```

✓ 0.1s

Prikaz Konfuzione matrice modela I odgovarajućih parametara za proveru tačnosti modela

```
Confusion Matrix:
[[2515    0]
 [   0  678]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2515
           1       1.00      1.00      1.00       678

    accuracy                           1.00      3193
   macro avg       1.00      1.00      1.00      3193
weighted avg       1.00      1.00      1.00      3193
```

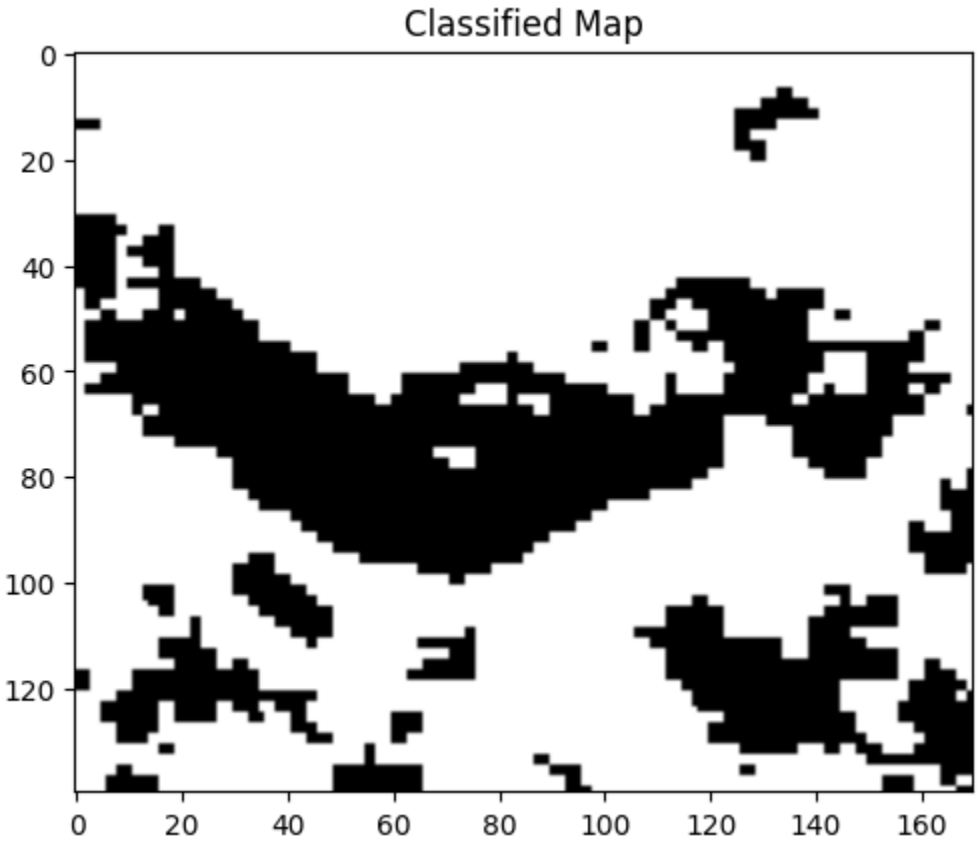


Provera modela logističke regresije na testnom području.

```
y_pred4 = logistic_classifier.predict(new_X_scaled)
y_true4 = labels2.flatten()

# Evaluacija modela logisticke regresije na novim, do sad nepoznatim, podacima
print("Confusion Matrix:")
print(confusion_matrix(y_true4, y_pred4))

print("\nClassification Report:")
print(classification_report(y_true4, y_pred4))

new_classified_map2 = logistic_classifier.predict(new_X_scaled).reshape(ndvi_data_test.shape)
plt.imshow(new_classified_map2, cmap='gray')
plt.title('Classified Map')
plt.show()
```

✓ 0.1s

```
Confusion Matrix:
[[ 6601   246]
 [  196 16757]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.96      0.97      6847
           1       0.99      0.99      0.99     16953

    accuracy                           0.98     23800
   macro avg       0.98      0.98      0.98     23800
weighted avg       0.98      0.98      0.98     23800
```



Kao i primenom Random Forest tehnike i ovde sada postoje lažno pozitivne i lažno negativne vrednosti.

2.4. Alarm – slanje poruke

```python
def send_email(subject, body, to_email):
    from_email = 'mojemail@gmail.com'
    from_password = 'lozinka'

    msg = MIMEMultipart()
    msg['From'] = from_email
    msg['To'] = to_email
    msg['Subject'] = subject

    msg.attach(MIMEText(body, 'plain'))

    try:
        server = smtplib.SMTP('smtp.example.com', 587)
        server.starttls()
        server.login(from_email, from_password)
        text = msg.as_string()
        server.sendmail(from_email, to_email, text)
        server.quit()
        print("Email sent successfully")
    except Exception as e:
        print(f"Failed to send email: {e}")
```

Funkcija send_email podiže smtp server za slanje mejlova, postavlja email sa kojeg će da se šalje poruka kao i njegovu lozinku kako bi mu pristupio. Iz bezbednosnih razloga nije navedena odgovarajuća email adresa, zbog toga što se ovaj kod nalazi javno na githubu.

```python
def load_data():
    aoinew = ee.Geometry.Polygon([
    [
        [20.615000, 43.046000],  # Ugao 1 (Jugo Zapadna granica)
        [20.619000, 43.046000],  # Ugao 2 (Jugo Istočna granica)
        [20.619000, 43.050000],  # Ugao 3 (Severo Istočna granica)
        [20.615000, 43.050000]   # Ugao 4 (Severo Zapadna granica)
    ]
    ])

    # Trebalo bi da se uzimaju datumi u zadnjih nedelju dana ili samo za juce npr, ali nemaju jos podaci za taj period
    dataset_new = ee.ImageCollection('COPERNICUS/S2_HARMONIZED').filterDate('2024-01-01', '2024-03-31').filterBounds(aoi).map(calculate_ndvi_evi)

    median_ndvi_new = dataset_new.select('NDVI').median().clip(aoinew)
    median_evi_new = dataset_new.select('EVI').median().clip(aoinew)

    export_path_new = os.path.join(images_folder, 'median_ndvi_new.tif')
    export_path_new2 = os.path.join(images_folder, 'median_evi_new.tif')

    geemap.ee_export_image(
        median_ndvi_new,
        filename=export_path_new,
        scale=10,
        region=aoi.getInfo()['coordinates']
    )

    geemap.ee_export_image(
        median_evi_new,
        filename=export_path_new2,
        scale=10,
        region=aoi.getInfo()['coordinates']
    )

    with rasterio.open(export_path_new) as src:
        ndvi_data_new = src.read(1)

    with rasterio.open(export_path_new2) as src:
        evi_data_new = src.read(1)

    X_new = np.vstack([
        ndvi_data_new.flatten(),
        evi_data_new.flatten()
    ]).T

    X_scaled_new = scaler.transform(X_new)

    y_pred_new = rf_classifier.predict(X_scaled_new)
    predicted_map = y_pred_new.reshape(ndvi_data_new.shape)

    return predicted_map
```
✓ 0.0s

Funkcija load_data() služi da se ubace podaci za željeno područje kako bi se kreirale slike i izračunali neophodni podaci, kao i pripremili podaci na način koji je bio primenjem u testiranju i kreiranju modela.

```python
# Proveravamo da li ima vise pokrcenih piksela od navedenog broja piksela (5% za sad).
def check_deforestation_total(predicted_map):
    total_deforested_pixels = np.sum(predicted_map == 1)

    # Ako ima ukupno vise od 5% pokrcenih piksela onda se aktivira alarm
    if total_deforested_pixels >= (predicted_map.size * 0.05):
        return True
    return False

def monitor_deforestation_and_notify():
    predicted_map = load_data()

    # Mozemo da vidimo koliko ukupno ima piksela kako bi promenili treshold
    print(predicted_map.size)

    # Ako postoji jedan piksel da je oznacen kao 1, tj da spada u pokrcen deo sume (deforest)
    if check_deforestation_total(predicted_map):
        send_email('Alarm: Detekcija bespravne sece sume', 'Bespravna seca je zabelezena u tvojoj oblasti.', 'nekome@gmail.com')

monitor_deforestation_and_notify()
```
✓ 5.2s

Funkcija check_deforestation_total proverava za poslatu prediction_map da li je više od 5 posto ukupnog područja pokrčeno i ako jeste to će okinuti proces slanja mejla sa nazivom: “Alarm: Detekcija bespravne sece sume” i sadržajem: ”Bespravna seca je zabelezena u tvojoj oblasti.” i to će biti poslato na izabrani mejl.

```
Generating URL ...
Downloading data from https://earthengine.googleapis.com/v1/projects/nikolavukasinovic/thumbnails/0c23dfe68a23443236072acc916a21d5-943c61e9811c0bfc7dfc52b93c4b808d:getPixels
Please wait ...
Data downloaded to k:\DetekcijaSuma\images\median_ndvi_new.tif
Generating URL ...
Downloading data from https://earthengine.googleapis.com/v1/projects/nikolavukasinovic/thumbnails/b3912093bb44575159c98039667c178c-28b014746ca1f808462f6dab3dff3ee4:getPixels
Please wait ...
Data downloaded to k:\DetekcijaSuma\images\median_evi_new.tif
6020
Failed to send email: [Errno 11001] getaddrinfo failed
```

Za primer kada je uočeno više od 5% pokrčenosti, okida se slanje mejla, što je važno, ali trenutno dolazi do greške jer kredencijali nisu nameštеni.

```
Generating URL ...
Downloading data from https://earthengine.googleapis.com/v1/projects/nikolavukasinovic/thumbnails/0c23dfe68a23443236072acc916a21d5-43752a90522e1c034e40bb2c7d0ef547:getPixels
Please wait ...
Data downloaded to k:\DetekcijaSuma\images\median_ndvi_new.tif
Generating URL ...
Downloading data from https://earthengine.googleapis.com/v1/projects/nikolavukasinovic/thumbnails/b3912093bb44575159c98039667c178c-f0d80f9c7718c14e779e521763da0d05:getPixels
Please wait ...
Data downloaded to k:\DetekcijaSuma\images\median_evi_new.tif
6020
```

Kada uslov nije ispunjen, ne dobijamo poruku za grešku pri slanju mejla jer ne dolazi do poziva te funkcije.

3. Dorada projekta

U dorađenom projektu je izmenjena funkcija send_email() koja umesto podizanja smtp servera i slanje mejla sa porukom sada koristi serijski port za slanje poruke (alarma) direktno na mikrokontroler. Za ovu svrhu, uvežena je biblioteka pyserial, koja omogućava komunikaciju preko serijskih portova sa mikrokontrolerima.

```python
def send_email(message):
    try:
        ser = serial.Serial('COM3', 9600, timeout=1)
        ser.write(message.encode())
        ser.close()
        print(f"Alarm: '{message}' poslat na mikrokontroler.")

    except serial.SerialException as e:
        print("Serijski port nije dostupan. Simuliracu slanje alarma na mikrokontroler.")
        print(f"Simulirani alarm: '{message}' poslat na mikrokontroler.")

    except Exception as e:
        print(f"Greška pri slanju alarma: {e}")
```

Za potrebe testiranja dodat je izuzetak koji simulira slanje poruke na mikrokontroler, što omogućava testiranje slanja alarma i izbegavanje greške ukoliko mikrokontroler fizički nije spojen sa računarom.

```
PS C:\Users\nikol> & C:/Python312/python.exe k:/DetekcijaSuma/doradjen_projekat.py
Generating URL ...
Downloading data from https://earthengine.googleapis.com/v1/projects/nikolavukasinovic/thumbnails/5050d3b00ff4c91337baa666017d6b5e-99bc470488b
96bb7952a86da4191bce6:getPixels
Please wait ...
Data downloaded to K:\DetekcijaSuma\images\median_ndvi_new.tif
Generating URL ...
Downloading data from https://earthengine.googleapis.com/v1/projects/nikolavukasinovic/thumbnails/5d1eeff2baa83fa5838eea869a11e6ab-7b1fd6b5fc4
121168f57fb64fa6d5f5d:getPixels
Please wait ...
Data downloaded to K:\DetekcijaSuma\images\median_evi_new.tif
Serijski port nije dostupan. Simuliracu slanje alarma na mikrokontroler.
Simulirani alarm: 'Bespravna seca je zabelezena u tvojoj oblasti.' poslat na mikrokontroler.
```

4. Problemi sa kojima sam se susretao u toku izrade projekta

Prvi problem na koji sam naišao u izradi projekta je bilo povezivanje na Google earth engine, naime više nije moguće kreirati projekat u VSC-u i samo autentifikovati nalog. Sada je neophodno da se prvo napravi cloud projekat makar i prazan, pa da se zatim inicijalizuje iz koda.

Drugi problem bio je vezan za vizuelni prikaz mapa unutar VSC-a, u početku su mape bile prikazivane samo na jupyter notebook-u, ali uz par ekstenzija i biblioteka i ovo je prevaziđeno.

Treći i najveći problem je taj što sam izabrao odgovarajuću površinu sa koje bih uzeo podatke za precizno treniranje modela i pisalo je da će ti podaci biti dostupni za preuzimanje u roku od 12-24h. Međutim i nakon 2 nedelje čekanja i ponovnih zahteva ta opcija mi i dalje nije omogućena. U pitanju je open-source web aplikacija Global Forest Watch. Na slici ispod je prikaza aplikacija GFW sa porukom koja i dalje glasi za 12-24h.

Četvrti problem je nastao kao rešenje prethodnog, pokušao sam da napravim sintetičke podatke iz postojeće slike za AOI (Aria of interest, tj područje od interesa) na kojima bih pratio da li ima bespravne seče šuma. Međutim podaci su se pokazali kao veoma loši.

Peti problem je nastavak na četvrti gde sam umesto pravljenja sintetičkih podataka iz AOI, koji ćemo da proveravamo, izabrao dve potpuno različite oblasti jedna je slabo pošumljena oblast Amazona, dok je druga gusto pošumljena oblast iz Severne Amerike kako podaci ne bi bili previše pristrasni kasnijem predviđanju. Pošto je kombinovanje ta dva skupa u jedan zajednički prelazilo dozvoljenu količinu podataka za izvoz morao sam da smanjim oblasti za treniranje i da njihove posebno slike exportujem pa kasnije spajam u pripremi za treniranje. Međutim i sa takvim podacima primećujemo značajan overfitting.

Šesti problem je taj što za korišćenje stmp servera za slanje mejla mora da se unese email i lozinka sa kog bi se slao mejl upozorenja, sad postoji i dvostruka autentifikacija pa bi morala da se ubaci posebna lozinka vezana za taj mejl i aplikaciju ali pošto će rad imati javni pristup na github-u ostavio sam samo primer umesto pravog email-a i lozinke.