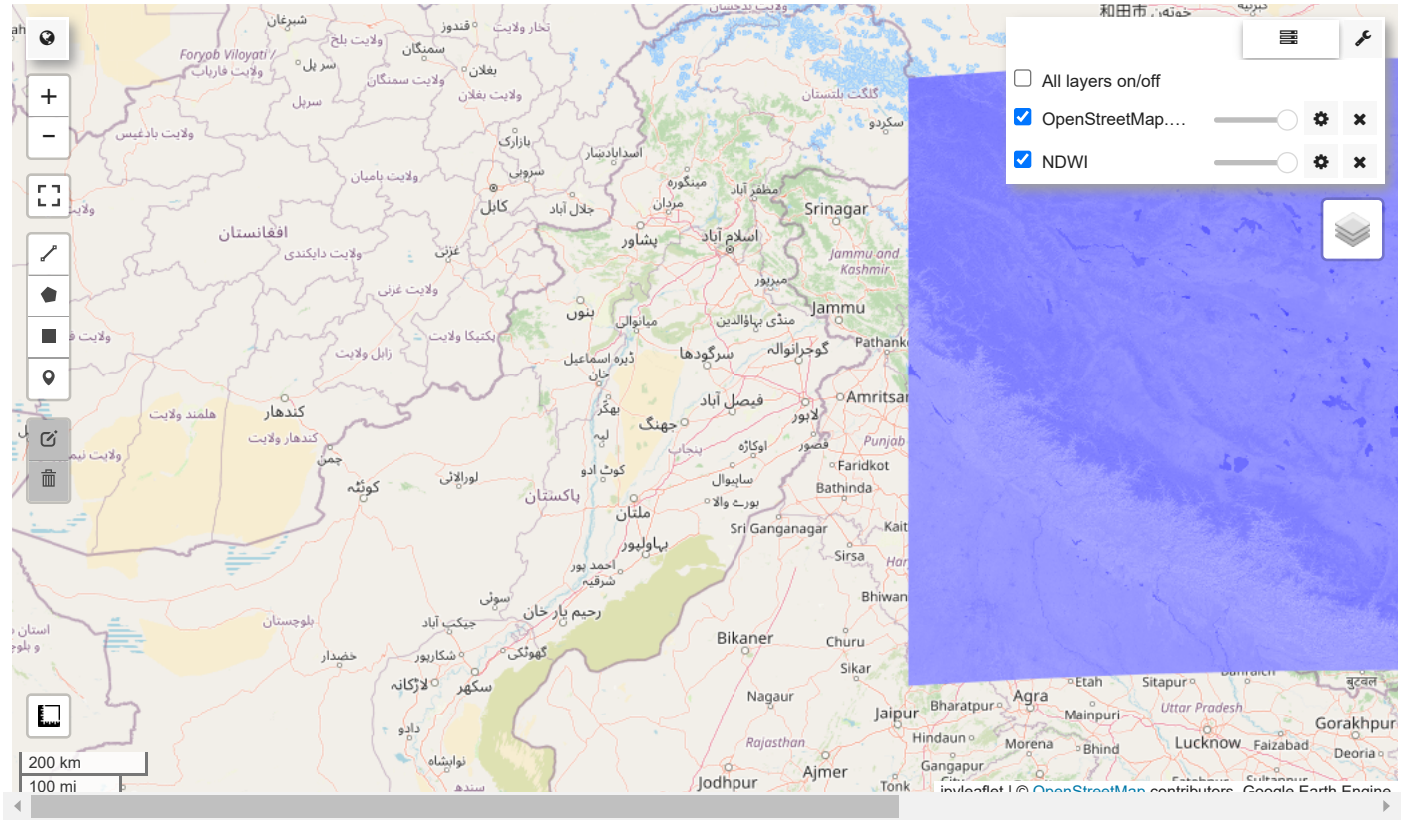
```
import ee
import geemap


# Trigger the authentication flow.
ee.Authenticate()

# Initialize the library.
ee.Initialize()
```

```
WARNING:googleapiclient.http:Encountered 403 Forbidden with reason "PERMISSION_DENIED"
---------------------------------------------------------------------------
EEException                               Traceback (most recent call last)
<ipython-input-4-7392b91b4f51> in <cell line: 9>()
      7
      8 # Initialize the library.
----> 9 ee.Initialize()

                                 1 frames
/usr/local/lib/python3.10/dist-packages/ee/__init__.py in Initialize(credentials, url, cloud_api_key, http_transport, project)
    179     matches = re.search(api_err, str(e))
    180     if (adc_err in str(e)) or (matches and oauth.is_sdk_project(matches[1])):
--> 181       raise EEException(NO_PROJECT_EXCEPTION) from None
    182     raise e
    183   Array.initialize()

EEException: ee.Initialize: no project found. Call with project= or see http://goo.gle/ee-auth.
```

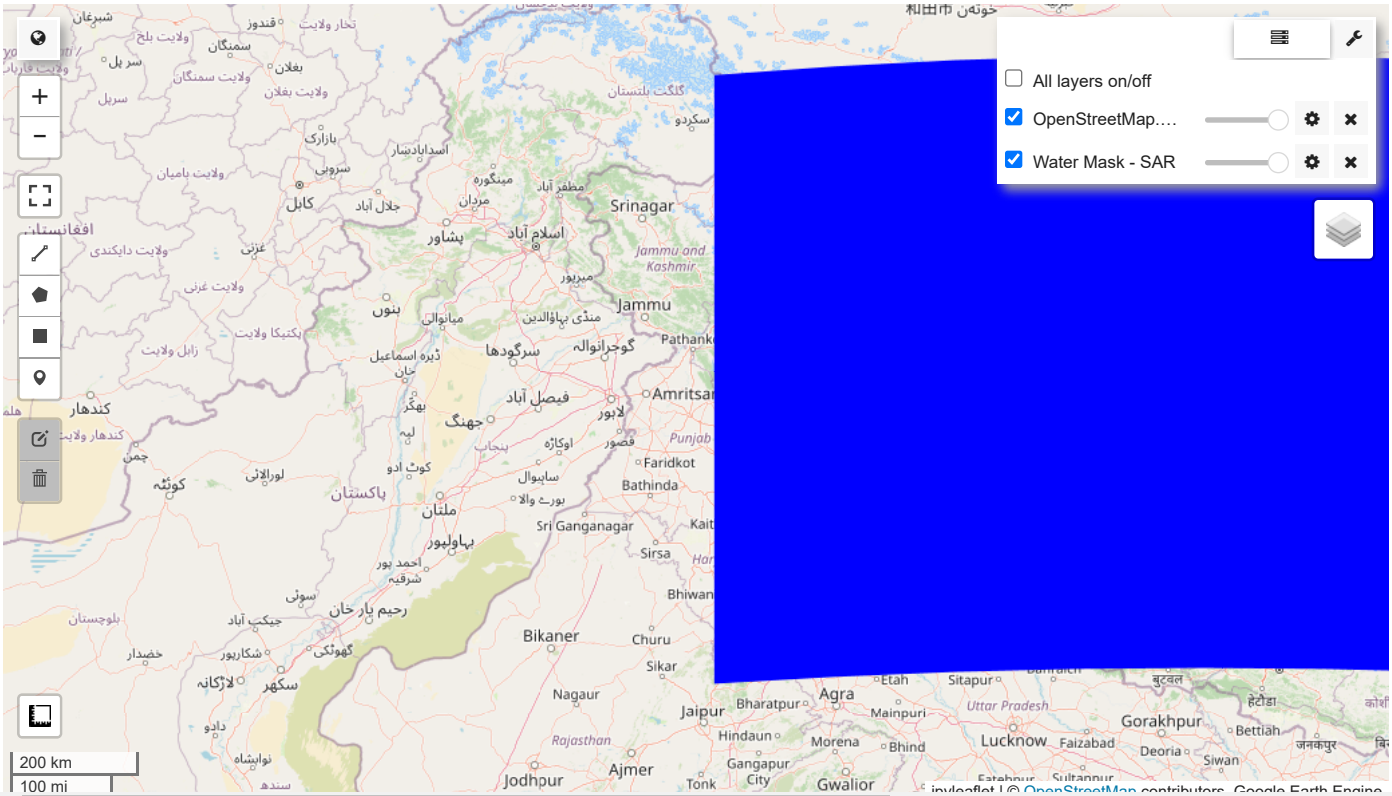
Next steps: Explain error

```
import ee

# Authenticate Google Earth Engine account
ee.Authenticate()

# Initialize the Earth Engine library
ee.Initialize(project='ee-auroradestiny123')
```

```
roi = ee.Geometry.Rectangle([76.0, 27.5, 92.0, 36.0])

# Load Sentinel-2 data and filter by date and cloud coverage
sentinel_2_dataset = ee.ImageCollection('COPERNICUS/S2') \
    .filterBounds(roi) \
    .filterDate('2023-01-01', '2023-12-31') \
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 10))  # Filter for cloud cover less than 10%

# Select relevant bands (B3 = Green, B4 = Red, B8 = NIR) for NDWI computation
def compute_ndwi(image):
    ndwi = image.normalizedDifference(['B3', 'B8']).rename('NDWI')  # NDWI using Green (B3) and NIR (B8)
    return image.addBands(ndwi)

# Apply NDWI computation on Sentinel-2 collection
ndwi_collection = sentinel_2_dataset.map(compute_ndwi)

# Visualization parameters for NDWI (Normalized Difference Water Index)
ndwi_viz = {'min': -1.0, 'max': 1.0, 'palette': ['white', 'blue']}

# Create a geemap interactive map centered at a point within the ROI
Map = geemap.Map(center=[31.5, 84.0], zoom=6)  # Center roughly within the ROI

# Add the median of the NDWI collection clipped to the ROI
ndwi_median = ndwi_collection.select('NDWI').median().clip(roi)
Map.addLayer(ndwi_median, ndwi_viz, 'NDWI')

# Add layer controls
Map.addLayerControl()

# Display the map
Map
```




```
sentinel_1_dataset = ee.ImageCollection('COPERNICUS/S1_GRD') \
    .filterBounds(roi) \
    .filterDate('2023-01-01', '2023-12-31') \
    .filter(ee.Filter.eq('instrumentMode', 'IW')) \
    .filter(ee.Filter.eq('orbitProperties_pass', 'ASCENDING')) \
    .select('VV')

# Apply a speckle filter (e.g., Lee filter)
def speckle_filter(image):
    return image.focal_mean(radius=50, units='meters').rename('Filtered_VV')

# Apply the speckle filter on the Sentinel-1 collection
filtered_sentinel_1 = sentinel_1_dataset.map(speckle_filter)

# Thresholding to detect water (based on the low backscatter values of water)
water_threshold = -17.0  # Define a threshold for water backscatter (adjust as necessary)
water_mask = filtered_sentinel_1.median().lt(water_threshold)

# Create a geemap interactive map
Map = geemap.Map(center=[31.5, 84.0], zoom=6)

# Display the water mask
Map.addLayer(water_mask.clip(roi), {'palette': ['blue']}, 'Water Mask - SAR')

# Add layer control and display the map
Map.addLayerControl()
Map
```




```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
import pandas as pd

water_threshold = -17.0
water_mask = filtered_sentinel_1.median().lt(water_threshold)


sensor_data = pd.DataFrame({
    'timestamp': pd.date_range('2023-01-01', periods=365, freq='D'),
    'water_level': [50 + i*0.1 for i in range(365)],  # Simulated water level data
    'precipitation': [10 + i*0.05 for i in range(365)],  # Simulated precipitation data
    'flood_occurred': [0 if i % 50 != 0 else 1 for i in range(365)]  # Simulated flood events
})

# Step 7: Feature Engineering - Combine satellite and sensor data
# Extract NDWI from satellite imagery
ndwi_collection = filtered_sentinel_1.map(lambda img: img.normalizedDifference(['Filtered_VV', 'VV']).rename('NDWI'))
ndwi_median = ndwi_collection.median().clip(roi)
```

```
sensor_data['ndwi'] = [water_threshold] * len(sensor_data)
```

```
X = sensor_data[['water_level', 'precipitation', 'ndwi']].fillna(0)
y = sensor_data['flood_occurred']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")
```

```
def predict_flood(real_time_data):
    real_time_data['ndwi'] = water_threshold  # Placeholder for real-time NDWI
    real_time_data = real_time_data[['water_level', 'precipitation', 'ndwi']].fillna(0)
    return model.predict([real_time_data])

# Simulate a new sensor reading for prediction
new_sensor_data = {'water_level': 40, 'precipitation': 0}
flood_prediction = predict_flood(pd.Series(new_sensor_data))
print(f"Flood Prediction: {'Flood Expected' if flood_prediction == 1 else 'No Flood'}")

# Step 9: Visualize the results on an interactive map
Map = geemap.Map(center=[31.5, 84.0], zoom=6)
Map.addLayer(water_mask.clip(roi), {'palette': ['blue']}, 'Water Mask - SAR')
Map.addLayerControl()
Map
```

```
Flood Prediction: No Flood
```