Exploratory Data Analysis

March 23, 2025

1 Personal Information

Name: **Selma Dissing**

StudentID: **14133156**

Email: **selma.dissing@student.uva.nl**

Submitted on: **23.03.2025**

Github Link

2 Data Context

For this thesis, I'm using satellite imagery from the Bolívar region in the Amazon, where artisanal small-scale gold mining is a growing environmental concern. The primary dataset comes from the open-source **Mining Detector** project by Earthrise Media. It uses imagery from **Sentinel-2**, a satellite operated by the European Space Agency that captures high-resolution images of the Earth's surface. These images are processed into small 48×48 pixel patches, each labeled either as *positive* (containing visible signs of mining activity) or *negative* (no mining present). These patches allow for pixel-level analysis of land use in dense forest regions.

To provide additional context, I've also included geographic data from **OpenStreetMap (OSM)**—a collaborative mapping project that offers information on roads, rivers, buildings, and land use. The idea is to combine these two data sources: satellite imagery and human-mapped infrastructure. This added layer could help detect patterns around known mining sites, such as proximity to roads or water bodies, and improve the model's ability to distinguish mining activity from other similar-looking areas.

Map of the Bolívar region

2.0.1 Satellite Patch Generation

To generate the Sentinel-2 training data used in this project, I relied on the Earthrise Mining Detector pipeline—specifically the `TrainingData` and `GEE_Data_Extractor` classes. The process begins with a set of annotated GPS points provided by Earthrise, identifying known illegal mining sites (positives) and areas confirmed to be non-mining (negatives). These annotations are stored in GeoJSON files and serve as the foundation for sampling.

For each point, a corresponding 48×48 pixel tile is created based on its latitude and longitude, essentially forming a spatial window around the location. These tiles are then used to query the Google Earth Engine API for Sentinel-2 imagery.

To ensure image quality, cloud scoring is applied using a defined threshold, and a median composite image is generated for the entire year of 2023. This reduces noise and improves consistency across samples. Each resulting image patch includes all 12 spectral bands from Sentinel-2 and is saved as a NumPy array in a `.pkl` file, alongside a class label (`0` for negative, `1` for positive).

The final dataset consists of **332 negative patches** and **170 positive patches**—a class imbalance that reflects the real-world rarity of mining relative to forest regions.

Positive Patches

Negative Patches

3 Data Description

[ ]:
```python
# Imports
import os
import pickle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import geopandas as gpd
import seaborn as sns
import osmnx as ox
```

3.0.1 Data Loading

[2]:
```python
# Load Sentinel-2 data
def load_data(basepath):
    # Load patch data from pickle files
    with open(basepath + "_patch_arrays.pkl", "rb") as f:
        data = pickle.load(f)
    with open(basepath + "_patch_array_labels.pkl", "rb") as f:
        labels = pickle.load(f)
    return np.array(data), np.array(labels)

# Define paths for negative and positive samples
neg_path = "data/patches/v2.0_bolivar_negatives_2023-01-01_2023-12-31"
pos_path = "data/patches/v2.1_bolivar_positives_2023-01-01_2023-12-31"

# Load negative and positive samples
neg_data, neg_labels = load_data(neg_path)
pos_data, pos_labels = load_data(pos_path)

# Combine negative and positive samples
X = np.concatenate([neg_data, pos_data])
y = np.concatenate([neg_labels, pos_labels])
```

[ ]:
```python
# Load your Bolivar AOI GeoJSON
bolivar_gdf = gpd.read_file("data/boundaries/BolivarAmazonas.geojson")
bolivar_geom = bolivar_gdf.union_all()

# Define tags to fetch relevant features
tags = {
    "highway": True,
    "waterway": True,
    "building": True,
    "landuse": True
}
```

```
# Use the correct module for geometries (this takes a while)
osm_gdf = ox.features.features_from_polygon(bolivar_geom, tags=tags)
```

```
/Users/selmadissing/miniforge3/envs/mining-detector/lib/python3.9/site-
packages/osmnx/_overpass.py:267: UserWarning: This area is 204 times your
configured Overpass max query area size. It will automatically be divided up
into multiple sub-queries accordingly. This may take a long time.
  multi_poly_proj = utils_geo._consolidate_subdivide_geometry(poly_proj)
```

[4]:
```
# Save to GeoJSON
os.makedirs("data/contextual", exist_ok=True)
osm_gdf.to_file("../data/contextual/osm_bolivar_features.geojson",␣
 ↪driver="GeoJSON")
```

[5]:
```
# Load OSM data
osm_path = "../data/contextual/osm_bolivar_features.geojson"
osm_gdf = gpd.read_file(osm_path)
```

```
/Users/selmadissing/miniforge3/envs/mining-detector/lib/python3.9/site-
packages/pyogrio/raw.py:198: RuntimeWarning: Several features with id =
313500437 have been found. Altering it to be unique. This warning will not be
emitted anymore for this layer
  return ogr_read(
```

3.0.2 Overview of Data

This code provides a high-level overview of the patch dataset. It prints - the total number of patches - the dimensions of a single patch - the overall shape of the dataset array - how many patches are labeled as positive (indicating mining activity) versus negative (no mining)

This helps confirm the dataset's structure and provides a quick look at class imbalance.

[6]:
```
# Overview
print(f"Total patches: {len(X)}")
print(f"Patch shape: {X[0].shape}")
print(f"Class balance: {np.bincount(y)} (0=No Mine, 1=Mine)")

print("Shape of patch array:", X.shape)  # (502, 48, 48, 12)
print("Number of positive samples:", np.sum(y))
print("Number of negative samples:", len(y) - np.sum(y))
print("Label distribution:", pd.Series(y).value_counts(normalize=True))
```

```
Total patches: 502
Patch shape: (48, 48, 12)
Class balance: [332 170] (0=No Mine, 1=Mine)
Shape of patch array: (502, 48, 48, 12)
Number of positive samples: 170
Number of negative samples: 332
Label distribution: 0    0.661355
```

```
1    0.338645
Name: proportion, dtype: float64
```

[7]:
```
# Calculate baseline accuracy by always predicting the majority class.
# This is the performance threshold the model must beat to be considered useful.
baseline_acc = max(np.mean(y), 1 - np.mean(y))
print(f"Baseline classifier accuracy (majority class): {baseline_acc:.2f}")
```

```
Baseline classifier accuracy (majority class): 0.66
```

3.0.3 Outlier Detection

To check the integrity of the training data, I ran two tests to catch potential outliers like empty or corrupted patches. - First, I looked for patches containing at least one pixel where all 12 spectral bands are zero—this could indicate a dead pixel or masked-out region. - Second, I checked for patches where an entire spectral band contains only zeros, which might point to a sensor issue or download error.

Both checks returned zero cases, suggesting that all patches were successfully retrieved and contain valid, usable data across all bands.

[8]:
```
# Check if any pixel (per patch) has all bands as zero
zero_pixel_mask = np.all(X == 0, axis=-1)  # Shape: (N, 48, 48)
patches_with_zero_pixels = np.any(zero_pixel_mask, axis=(1, 2))
num_patches_with_zero_pixels = np.sum(patches_with_zero_pixels)

print(f"Found {num_patches_with_zero_pixels} patches with at least one␣
↪zero-only pixel")
```

```
Found 0 patches with at least one zero-only pixel
```

[9]:
```
# Check for bands with all-zero values in each patch
zero_band_mask = np.all(X == 0, axis=(1, 2))  # Shape: (N, 12)
patches_with_zero_bands = np.any(zero_band_mask, axis=1)
num_zero_band_patches = np.sum(patches_with_zero_bands)

print(f"Found {num_zero_band_patches} patches with at least one band of␣
↪all-zero values")
```

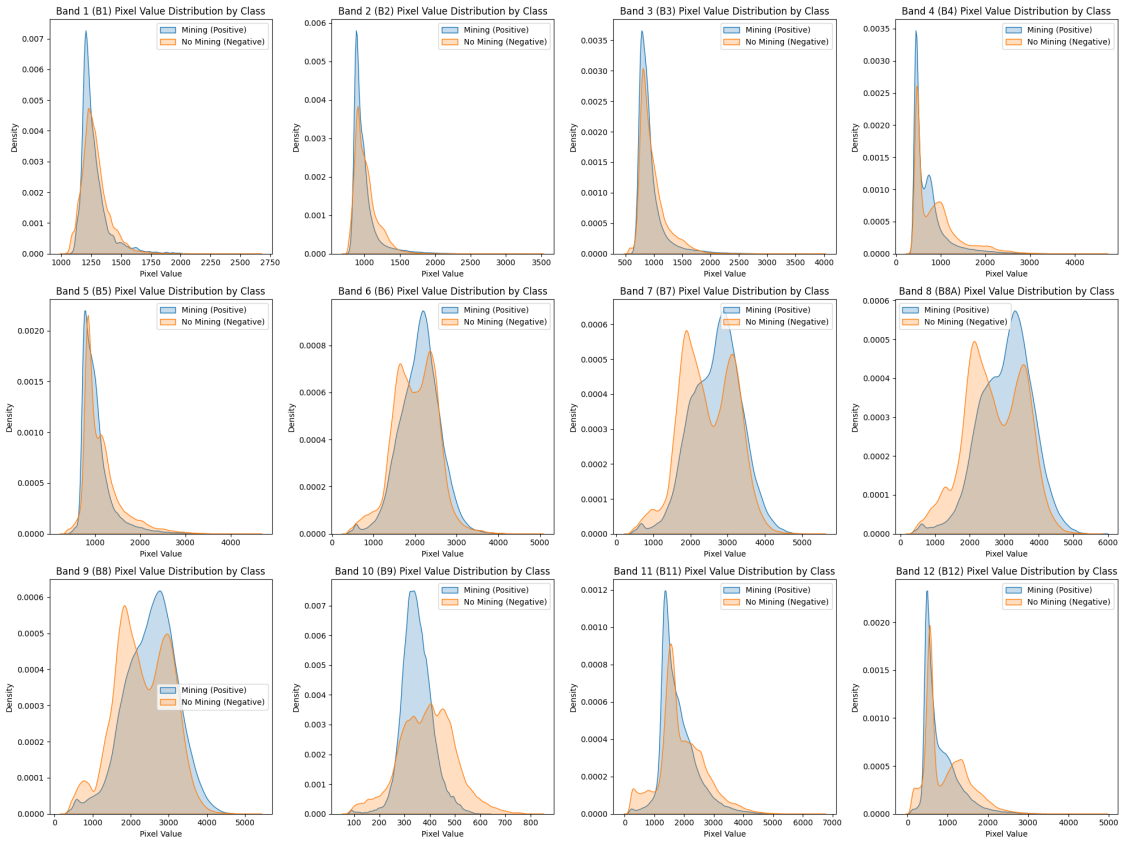
```
Found 0 patches with at least one band of all-zero values
```

3.0.4 Per-Band Class Distribution Differences

These plots show how pixel values are distributed across each Sentinel-2 band for mining (positive) and non-mining (negative) patches.

While the differences between classes are subtle, several bands—especially B4 (red), B8 (near-infrared), and B11/B12 (shortwave infrared)—show noticeable shifts in distribution. This suggests that mining activity alters surface properties like vegetation and soil reflectance, which these bands capture well. Although the distributions overlap, the patterns indicate that spectral information could be useful for classification when combined in a machine learning model.

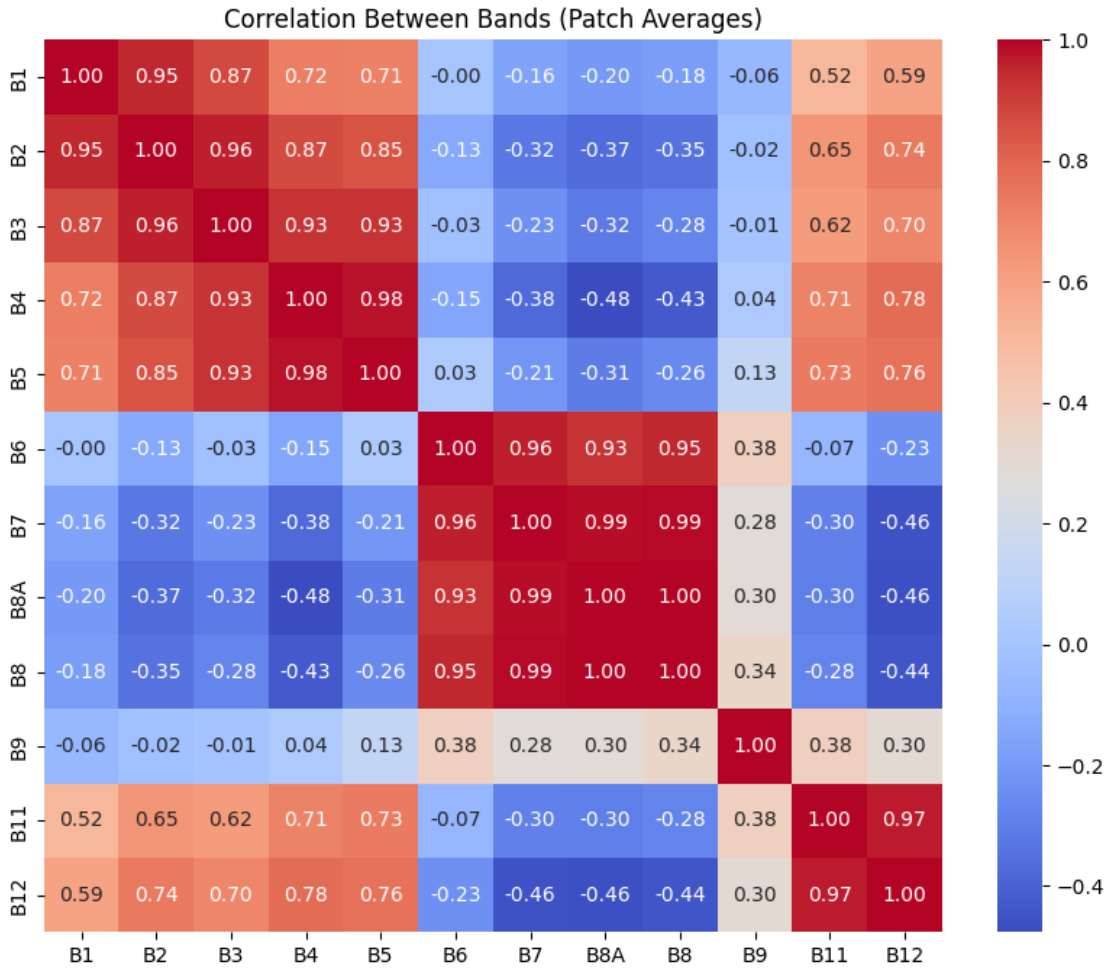
[12]:
```
# Flatten per class
flat_X_pos = X[y == 1].reshape(-1, 12)
flat_X_neg = X[y == 0].reshape(-1, 12)

band_names = [
    "B1", "B2", "B3", "B4", "B5", "B6",
    "B7", "B8A", "B8", "B9", "B11", "B12"
]

plt.figure(figsize=(20, 15))
for band_index in range(12):
    plt.subplot(3, 4, band_index + 1)
    sns.kdeplot(flat_X_pos[:, band_index], label="Mining (Positive)", fill=True)
    sns.kdeplot(flat_X_neg[:, band_index], label="No Mining (Negative)",␣
 ↪fill=True)
    plt.title(f"Band {band_index + 1} ({band_names[band_index]}) Pixel Value␣
 ↪Distribution by Class")
    plt.xlabel("Pixel Value")
    plt.ylabel("Density")
    plt.legend()

plt.tight_layout()
plt.show()
```



3.0.5 Band Correlation

The heatmap reveals strong correlations among several spectral bands, especially within the red-edge group (B5–B8A) and the visible spectrum (B1–B4). This redundancy suggests potential for dimensionality reduction without major information loss. In contrast, the SWIR bands (B11 and B12) show lower correlations with most other bands, indicating they contribute more unique spectral information. These insights can guide feature selection and help avoid multicollinearity in later modeling steps.

[13]:
```
# Take mean per patch per band
band_means = X.mean(axis=(1, 2))  # shape: (502, 12)
corr_matrix = pd.DataFrame(band_means, columns=band_names).corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Between Bands (Patch Averages)")
plt.show()
```



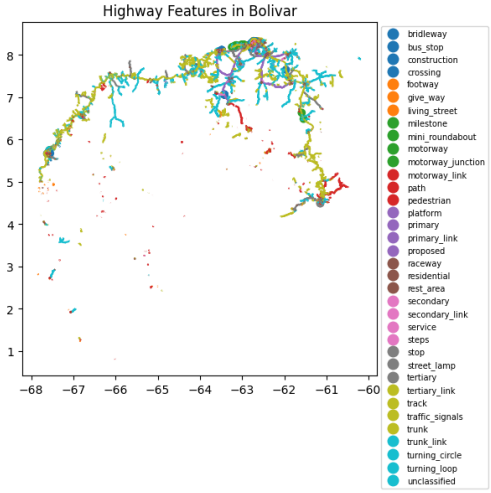
3.0.6 OSM Data

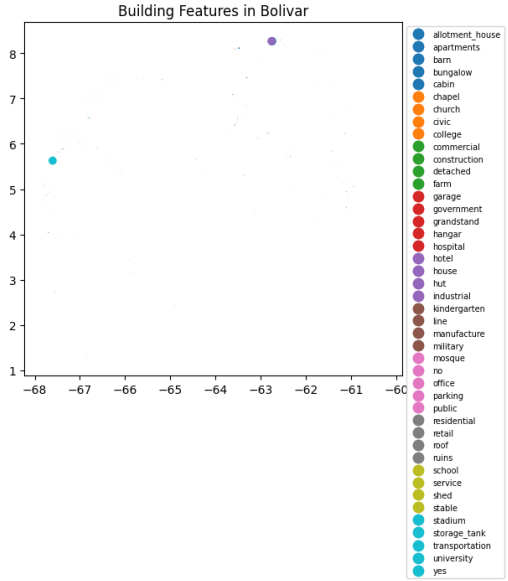
[14]:
```
# Dataset summary
print(f"OSM features loaded: {len(osm_gdf)}")

# Analyze OSM features
print("\n--- OSM Feature Overview ---")
print(osm_gdf[['highway', 'building', 'landuse', 'waterway']].
  ↪describe(include='all'))
```

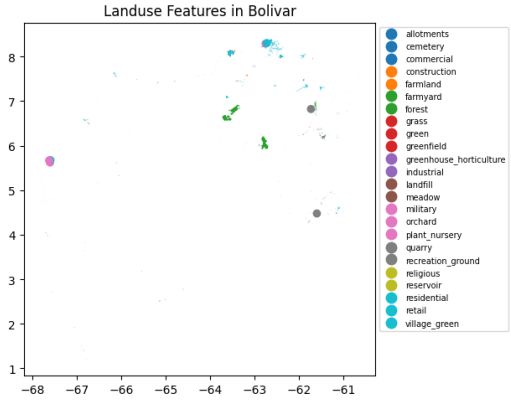
```
OSM features loaded: 121054

--- OSM Feature Overview ---
            highway building      landuse waterway
count         28463    64565         1721    26308
unique           36       44           24        9
top     residential      yes  residential    river
freq          11479    58964          843    21718
```

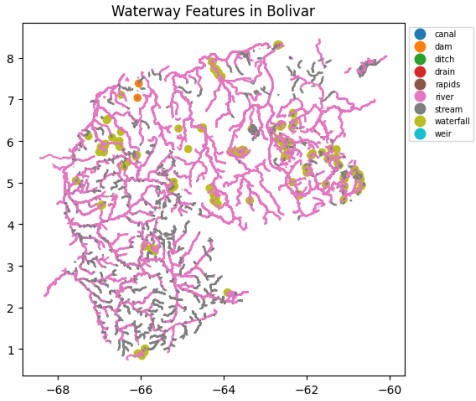
This set of maps provides a spatial overview of contextual features in the Bolívar region extracted from OpenStreetMap. The data is divided into four categories: highways, buildings, land use, and waterways. We see that waterways are densely mapped and widespread across the region, particularly rivers and streams, which are highly relevant given the relationship between water access and illegal gold mining. In contrast, buildings and land use features are more sparsely annotated, with a few dense pockets—possibly corresponding to settlements or known industrial zones. Highway data is relatively detailed, especially in the northern and eastern parts of the region, which may suggest accessibility corridors. These contextual layers can be useful for enriching the satellite-based classification and exploring correlations between mining presence and infrastructure.

[15]:
```
# Plot each feature type in a 2x2 grid
feature_types = ['highway', 'building', 'landuse', 'waterway']
fig, axes = plt.subplots(2, 2, figsize=(15, 12))

for ax, feature in zip(axes.flatten(), feature_types):
    if feature in osm_gdf.columns:
        osm_gdf[osm_gdf[feature].notnull()].plot(column=feature, ax=ax,␣
  ↪legend=True, legend_kwds={'bbox_to_anchor': (1, 1), 'loc': 'upper left',␣
  ↪'fontsize': 7})
        ax.set_title(f"{feature.capitalize()} Features in Bolivar")

plt.tight_layout()
plt.show()
```


Highway Features in Bolivar
bridleway
bus_stop
construction
crossing
footway
give_way
living_street
milestone
mini_roundabout
motorway
motorway_junction
motorway_link
path
pedestrian
platform
primary
primary_link
proposed
raceway
residential
rest_area
secondary
secondary_link
service
steps
stop
street_lamp
tertiary
tertiary_link
track
traffic_signals
trunk
trunk_link
turning_circle
turning_loop
unclassified


Building Features in Bolivar
allotment_house
apartments
barn
bungalow
cabin
chapel
church
civic
college
commercial
construction
detached
farm
garage
government
grandstand
hangar
hospital
hotel
house
hut
industrial
kindergarten
kine
manufacture
military
mosque
no
office
parking
public
residential
retail
roof
ruins
school
service
shed
stable
stadium
storage_tank
transportation
university
yes


Landuse Features in Bolivar
allotments
cemetery
commercial
construction
farmland
farmyard
forest
grass
green
greenfield
greenhouse_horticulture
industrial
landfill
meadow
military
orchard
plant_nursery
quarry
recreation_ground
religious
reservoir
residential
retail
village_green


Waterway Features in Bolivar
canal
dam
ditch
drain
rapids
river
stream
waterfall
weir