Homework 1

Srujan Vaddiparthi - sv6126@g.rit.edu
https://github.com/SrujanVaddiparthi/Applcns_of_ML_remote_sensing/blob/main/eda

My final submission direct:
https://github.com/SrujanVaddiparthi/Applcns_of_ML_remote_sensing/blob/main/eda/FINAL_Submission_HW1.ipynb

Problem 1

My understanding of the “bands”:

- Extracted what each band meant from the following website: https://gisgeography.com/sentinel-2-bands-combinations/
- I was initially concerned with how exactly to recognize each band in the 12 bands of the Sentinel-2 correctly, but later realised I can just assign an order to those 12 bands where B10 was excluded.
- Before I define the plot_band(Args) function, I initiate a variable labels,

```
# defining the labels for each band
labels = [
            "B1 - 60 m - 443 nm - Ultra Blue (Coastal and Aerosol)",
            "B2 - 10 m - 490 nm - Blue",
            "B3 - 10 m - 560 nm - Green",
            "B4 - 10 m - 665 nm - Red",
            "B5 - 20 m - 705 nm - VNIR",
            "B6 - 20 m - 740 nm - VNIR",
            "B7 - 20 m - 783 nm - VNIR",
            "B8 - 10 m - 842 nm - VNIR",
            "B8a - 20 m - 865 nm - VNIR",
            "B9 - 60 m - 940 nm - SWIR",
            "B11 - 20 m - 1610 nm - SWIR",
            "B12 - 20 m - 2190 nm - SWIR"
]
```

Handling of “no data”:

- Hint was given in class, but either ways, I replace the 0.0 with “Nan”, and then exclude them in my visualilzations.

```
#prob 1: handling of no data with nan replacement
def replace_with_nan(arr: np.ndarray, no_data_value = 0):
    nan_replaced = arr.astype(float, copy=True)
    nan_replaced[arr == no_data_value] = np.nan # dam i love the
automcomplete here lol
```

```
    return nan_replaced
no_nan_data = replace_with_nan(data_copy, no_data_value = 0)
# no_nan_data
```

Stretching is needed to improve the contrast. Read online and will be implement a 98 percentile till 90 percentile stretch and decide which one's better for improved constrast.

- Implement two kinds of stretching

```
def minmax_stretch(img):
   low = np.nanmin(img)
   high = np.nanmax(img)
   output = (img-low)/(high-low)
   return np.clip(output, 0, 1).astype(np.float32) #ensuring consistency in
datatypes and clipping the output since did normalization.


def percentile_stretch(img, low_perc=10, high_per=90):
   low = np.nanpercentile(img, low_perc)
   high = np.nanpercentile(img, high_per)
   if not np.isfinite(low) or not np.isfinite(high) or high<=low:
       return np.zeros_like(img, dtype=np.float32)
   output = (img-low)/(high-low)
   return np.clip(output, 0, 1).astype(np.float32)
```

Finally implemented the plot_band function,

- First experimented with multiple cmocean options, and ended up with balance.

```
for i in range(data_copy.shape[-1]):
   plot_band(img = no_nan_data,
             band_i = i,
             cmap_name = "balance",
             stretch = ("percentile",3,97),
             show_colorbar = True)
```

- The plot_band code is in the FINAL_Submission_HW1.ipynb file.

In my opinion, a proper stretching is a percentile stretching, its easy to calibrate the contrast. And I used “balance” from the cmocean as I found it to give the most contrast between the built-up land, vegetation, and water bodies.

Problem 2

A. I calculated the band statistics:

label	mean	median	q1	q3	std	skew	kurt
B1 - 60 m - 443 nm - Ultra Blue (Coastal	0.08868	0.08290	0.07090	0.10070	0.02791	3	27

and Aerosol)							
B2 - 10 m - 490 nm - Blue	0.09254	0.08530	0.07160	0.10530	0.03500	4	46
B3 - 10 m - 560 nm - Green	0.10550	0.09870	0.08580	0.11680	0.03437	5	49
B4 - 10 m - 665 nm - Red	0.09431	0.08500	0.06630	0.10930	0.04445	3	26
B5 - 20 m - 705 nm - VNIR	0.13672	0.13100	0.11520	0.15060	0.04091	3	30
B6 - 20 m - 740 nm - VNIR	0.24359	0.24720	0.21310	0.28300	0.06104	-1	6
B7 - 20 m - 783 nm - VNIR	0.28584	0.29030	0.24440	0.33770	0.07730	-1	5
B8 - 10 m - 842 nm - VNIR	0.29138	0.29760	0.24750	0.34640	0.08033	-1	4
B8a - 20 m - 865 nm - VNIR	0.30351	0.30980	0.26050	0.35950	0.08222	-1	5
B9 - 60 m - 940 nm - SWIR	0.34508	0.34460	0.30520	0.38820	0.07795	0	5
B11 - 20 m - 1610 nm - SWIR	0.19172	0.18890	0.17100	0.20860	0.04836	1	9
B12 - 20 m - 2190 nm - SWIR	0.12918	0.12040	0.10200	0.14270	0.04855	2	11

- All the values inside each band are the reflectance values.
    - Mean = shows us what the average reflectance is in that particular band.
    - Std = how varied is it
    - Median = it is the 50th percentile value and so on for the Q1 and Q3 as well.
    - Skewness = degree of asymmetry around the mean. 3rd standardised moment of distribution around the mean.
        - +ve → right skewed; “bright” anomalies; higher reflectance, so either rooftops or clouds for example.

        - -ve → left skewed; “dull” anomalies; shadowed areas or water bodies; “low” reflectance.
        - 0 → symmetric around the mean.
    - Kurtosis = degree of peakedness or heaviness of tails relative to the normal distribution. 4th moment of distribution. As I understand it in terms of remote sensing, it tells us how sharp the peak is, or how extreme some outliers are.
        - >3 → more number of outliers than normal.
        - <3 → fewer outliers than normal.
    - Essentially we get an idea about the distribution of the reflectance values at each band.

I observe that B2, B3, B4 seem to have similar distributions:

I say this by looking at the mean, median, Q1, Q3 values, and they are almost similar. It would make sense because B2 B3 B4 lie in the visible wavelength, and their reflectances are almost the same in the visible spectrum.

And for skew and kurtosis observations:

I observe that, if kurtosis is super high, then that would mean that some outliers are extremely dominant, which would mean that the other values are pretty plain. So that would mean that the B2 B3 B4 which have super high kurtosis values, and positive skews which would mean that there are brighter anomalies, and they are super bright.

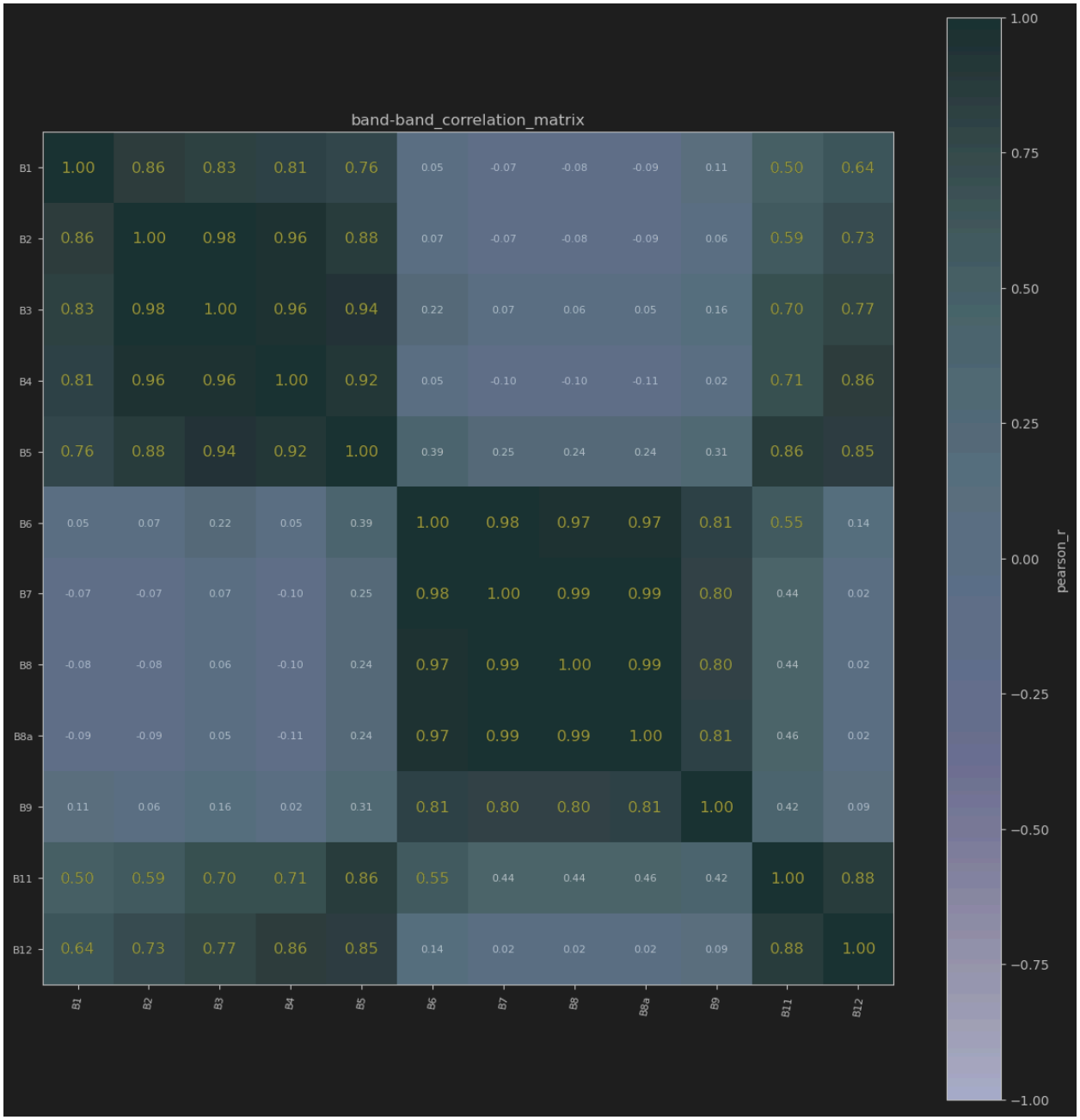
B. I implemented the z_score standardization:

- Standardizing the bands would make it possible for us to compare one band with another. Will help us identify anomalies. it would just standardize around the respective means, letting us properly assess the relative distributions, and easily identify the outliers and infer how each band’s reflectances are like.
- Rather than look at the numbers from above, this gives me an intuitive read of how the reflectances are distributed per band.
- Standardization is essentially, in each band, every pixel’s reflectance value is subtracted by the mean and divided by the std deviation. This centres the data around mean and the whole standardized data’s variance becomes 1. And there is no difference in magnitude between bands, making them easy to compare.
- It becomes easy for us to highlight anomalies by choosing a z threshold (z_thresh) which I fixated at 2.2. This would mean that the datapoints outside of the ~80% of the whole distribution are considered as anomalies.
    - $|z| > k$ is a good heuristic for anomalies detection in remote sensing. I experimented within the range of [0.5,3.0], and settled at 2.2.
    - I do not remember where I read this but 1-1/k is the percentage of distribution out of which the values present are considered to be outliers. Hence 1 - 1/(2.2) =~80%.

I plotted the distributions per band before and after standardization and can be seen in the FINAL_Submission_HW1.ipynb file.

Problem 3

A. Correlation_matrix to be built using pearson r correlation coefficient. We get a square matrix of size 12 * 12.



If |r| >= 0.50 → yellow big numbers → decent to strongly linear-relationship.
Else, → white small numbers → average to weak linear-relationship.

- As expected, the pairs of B2 B3 B4 have a high linear correlation (0.96 to 0.98 r values).
- Rest all I couldn't make sense of it, but the darker regions are where there is high linear correlation amongst the respective bands.

- And at a glance, almost every one of the pairs of bands have a positive correlation ⇒ if band 1's reflectance values increase, then band 2's increase as well. It's like, certain surfaces in the image captured, reflects off certain bands of the spectrum in a similar way.

B. I plotted the correlation plots using hexbin for plotting the correlation density plots. This was done only for the 10 m resolution bands (B2, B3, B4, B8). I wrote a function as instructed which creates two subplots:
    a. pairwise scatter plot b/w every two vectors = this a good "linear" relationship assessment. Similar to a pairplot in my opinion.
    b. density of scatter plot b/w every two vectors. This is useful for visualising areas where data points are more concentrated. I studied different techniques to plot this and picked hexbin charts. I even experimented with hist2d but am not displaying it in the final notebook, but attached a toggle in the function to switch between the density plots. I use cmocean's thermal color to plot my density plots.

- Hexbin charts divide the plot area into hexagonal bins, with each bin's color indicating the number of observations within it, offering a discrete approximation of density. https://datavizproject.com/data-type/hexagonal-binning/
- Whereas Hist2d or 2D histogram plots used square bins instead of hexagonal bins.
- These 2D density visualizations are very useful for large datasets where overlapping points in a scatterplot do not clearly capture the underlying pattern.
- They can be combined with marginal distributions (like histograms or density curves) to show both the joint relationship and the individual distributions of each variable.
- Such a combination helps in assessing correlation while also understanding the shape of each variable's distribution.
- The density plot can reveal patterns such as multimodal distributions or skewness that influence the interpretation of correlation.

Observations of these plots:

1. All the following three pairs have strong r values because they belong in the visible band (Red, Green, Blue). And from the pixel value distributions, their reflectances are very close from any kind of surface (water-bodies/vegetation/builtup-land).
    a. B2 vs B3: (Blue vs Green)
        i. r = 0.98, extremely strong. They rise or fall together.
        ii. They are of low pixel values <=> their reflectances are low.
        iii. Their wavelengths according to the website was B2 = 490 nm, B3 = 560 nm: very close, so most surfaces reflect these similarly.
    b. B4 vs B2: (Red vs Blue)
        i. r = 0.96, similarly strong. They rise or fall together.
    c. B3 vs B4: (Green vs Red)
        i. r = 0.96, similarly strong. They rise or fall together.
2. Whereas wrt to B8 (NIR), they have weak (positive & negative) r values.

    a. Which means it has weak "linear" relationship with the visible bands. Makes sense as it is Near Infrared.
3. From the website, it is mentioned that B8 (NIR) are good at reflecting chlorophyll. Hence where there is denser vegetation, the B8 band values will be higher, and the density associated with the visible spectrum.
4. The density plots associated with (B3 vs B8), (B4 vs B8), and (B2 vs B8) are in that oval shape that is very distinct from the other highly correlated density shapes. Which would suggest that there is some sort of a "non-linear" relationship amongst those bands.

Hence, the density plots which are oval/elongated shape, most likely represent a presence of a strong/weak "non-linear" relationship worth exploring. The others which are "dashed" and "widespread" might hint at a huge coverage of land and water bodies(maybe).

Problem 4

ECOSTRESS data provides high-resolution spectral reflectance across the electromagnetic spectrum.

- The task was to compare ECOSTRESS spectra with Sentinel-2 reflectance, specifically in the 0.35–2.5 μm range.
- Vegetation (oak, Quercus genus): already within this range.
- Manmade (asphalt/road): data originally spanned 0.42–14 μm → clipped to 0.35–2.5 μm to match Sentinel-2 range.

- Preprocessing: parsed .spectrum.txt files into wavelength + reflectance arrays.
- Converted reflectance from 0–100 (%) to 0–1 scale.
- Excluded Sentinel-2 bands affected by atmosphere or not present: B1 (443 nm), B9 (940 nm), B10 (1375 nm).
- Remaining 10 bands were kept (B2, B3, B4, B5, B6, B7, B8, B8A, B11, B12).
- I downsampled ECOSTRESS to Sentinel-2 bandpasses by averaging reflectance within each band window.
- Similarity metric: Spectral Angle Mapper (SAM), which is essentially cosine similarity in angle form.
- Lower SAM angle → higher similarity.

Analysis steps:

Computed SAM between every pixel spectrum in Sentinel-2 image and each ECOSTRESS reference spectrum (oak, asphalt).

Collected top-100 best matches (lowest SAM angles).

Plotted ECOSTRESS spectra vs the 1st, 50th, and 100th matched Sentinel-2 spectra for both oak and asphalt.

Observations:

- Oak: the Sentinel-2 matches followed the general shape of the ECOSTRESS curve, especially in the visible (B2–B4) and NIR (B6–B8A) regions, though Sentinel-2 reflectances were lower in magnitude.
- Asphalt: Sentinel-2 matches were consistently higher in reflectance than the ECOSTRESS asphalt curve, but the increasing trend across SWIR bands was preserved.
- Differences are expected because Sentinel-2 integrates over wide bands and includes mixed pixels, while ECOSTRESS spectra are lab-based and very high resolution.

Thresholding:

- Picked a SAM threshold (experimented with values, settled around 0.15 rad ~ 8.6°).
- Pixels below this angle were labeled as oak/asphalt.
- Created binary masks and also an RGB overlay (green = oak, gray = asphalt) for visualization.

Favorite approach for representation:

- Showing oak and asphalt masks separately for clarity.
- Overlaying both masks on a true-color RGB Sentinel-2 image for context and easy validation.