Libraries and datapath used

```python
import tifffile
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans

# TIFF dosyasını yükleme
tiff_path = r'C:\Users\tunah\PycharmProjects\pythonProject\multispectral.tif'
tiff_image = tifffile.imread(tiff_path)
```

Band 1

```python
# Bantları ayırma
red_band = tiff_image[:, :, 0]    # İlk bant kırmızı (Red)
green_band = tiff_image[:, :, 1]  # İkinci bant yeşil (Green)
blue_band = tiff_image[:, :, 2]   # Üçüncü bant mavi (Blue)
nir_band = tiff_image[:, :, 3]    # Dördüncü bant NIR (Near Infrared)

# Verileri normalize etme fonksiyonu
4 usages
def normalize_band(band):
    band_min = band.min()
    band_max = band.max()
    normalized_band = (band - band_min) / (band_max - band_min)
    return normalized_band

# Bantları normalize etme
red_band_normalized = normalize_band(red_band)
green_band_normalized = normalize_band(green_band)
blue_band_normalized = normalize_band(blue_band)
nir_band_normalized = normalize_band(nir_band)

# Bantları ayrı ayrı figürlerde çizdirme ve kaydetme
bands = {
    'Red Band': red_band_normalized,
    'Green Band': green_band_normalized,
    'Blue Band': blue_band_normalized,
    'NIR Band': nir_band_normalized
}
```

Band -2

```python
output_paths = {
    'Red Band': 'red_band_output.tiff',
    'Green Band': 'green_band_output.tiff',
    'Blue Band': 'blue_band_output.tiff',
    'NIR Band': 'nir_band_output.tiff'
}

# Renk paleti
colors = {
    'Red Band': 'Reds',
    'Green Band': 'Greens',
    'Blue Band': 'Blues',
    'NIR Band': 'Purples'
}

# Bantları sırayla gösterme
for band_name, band_data in bands.items():
    plt.figure(figsize=(6, 6))
    plt.imshow(band_data, cmap=colors[band_name])
    plt.title(band_name)
    plt.axis('off')
    plt.savefig( *args: output_paths[band_name], format='tiff')
    plt.show()
```

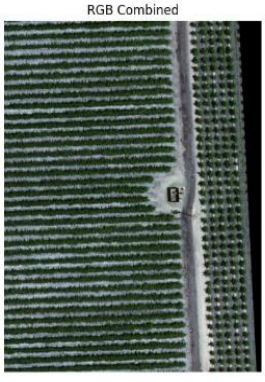
RGB Combination

```python
# RGB bantlarını birleştirme ve normalize etme
rgb_combined = np.dstack((red_band_normalized, green_band_normalized, blue_band_normalized))

# RGB kombinasyonu görüntüleme
plt.figure(figsize=(6, 6))
plt.imshow(rgb_combined)
plt.title('RGB Combined')
plt.axis('off')
plt.show()

# RGB kombinasyonu kaydetme
rgb_combined_path = 'rgb_combined_output.tiff'
tifffile.imwrite(rgb_combined_path, (rgb_combined * 255).astype(np.uint8))

# K-means kümeleme işlemi
# Veriyi (satır, sütun, bant) -> (satır*sütun, bant) şeklinde yeniden şekillendiriyoruz
rows, cols, bands = rgb_combined.shape
```



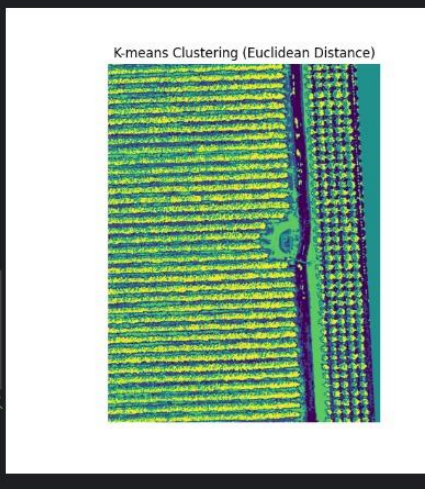
K-Means

```python
# K-means kümeleme işlemi
# Veriyi (satır, sütun, bant) -> (satır*sütun, bant) şeklinde yeniden şekillendiriyoruz
rows, cols, bands = rgb_combined.shape
flat_rgb_combined = rgb_combined.reshape((rows * cols, bands))

# K-means kümeleme, Öklidyen mesafe ile benzerliği ölçer
n_clusters = 5
kmeans = KMeans(n_clusters=n_clusters, random_state=0)
kmeans.fit(flat_rgb_combined)
clustered = kmeans.labels_.reshape((rows, cols))

# Kümeleme sonuçlarını görselleştirme
plt.figure(figsize=(6, 6))
plt.imshow(clustered, cmap='viridis')
plt.title('K-means Clustering (Euclidean Distance)')
plt.axis('off')
plt.show()

# Kümeleme sonuçlarını normalize etme ve kaydetme
# Etiketler [0, n_clusters - 1] aralığında olduğundan normalize ederek [0, 255] aralığına taşıyor
clustered_normalized = (clustered / (n_clusters - 1) * 255).astype(np.uint8)
clustered_path = 'kmeans_clustered_output.tiff'
tifffile.imwrite(clustered_path, clustered_normalized)
```



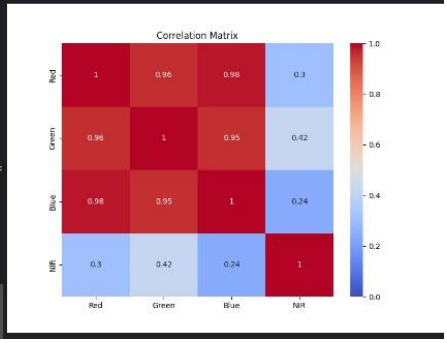
Correlation Matrix

```python
# Korelasyon matrisini hesaplama
normalized_bands = np.stack( arrays: (red_band_normalized, green_band_normalized, blue_band_normaliz
band_names = ['Red', 'Green', 'Blue', 'NIR']

# Korelasyon hesaplama
correlation_matrix = np.corrcoef(normalized_bands.reshape(-1, 4), rowvar=False,)

# Korelasyon matrisini görselleştirme
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, xticklabels=band_names, yticklabels=band_names, cmap=
plt.title('Correlation Matrix')
plt.savefig( *args: 'correlation_matrix.png', format='png')  # PNG olarak kaydet
plt.show()

# NDVI hesaplama
denominator = nir_band_normalized + red_band_normalized
denominator[denominator == 0] = np.nan  # Sıfıra bölme hatasını önlemek için sıfırları NaN yap
```



NDVI

```python
# NDVI hesaplama
denominator = nir_band_normalized + red_band_normalized
denominator[denominator == 0] = np.nan  # Sıfıra bölme hatasını önlemek için sıfırları NaN yap
ndvi = (nir_band_normalized - red_band_normalized) / denominator

# NDVI görüntüsünü görselleştirme
plt.figure(figsize=(6, 6))
plt.imshow(ndvi, cmap='RdYlGn')
plt.title('NDVI')
plt.colorbar(label='NDVI Value')
plt.axis('off')
plt.show()

# NDVI çıktısını kaydetme
ndvi_path = 'ndvi_output.tiff'
tifffile.imwrite(ndvi_path, (ndvi * 255).astype(np.uint8))
```