GreenLens Data Preparation/Feature Engineering

1. Fatu Kromah
2. Alfred Nyeswah
3. Sulaiman Barry
4. Joseph S. Lablah, Jr.
5. Jerome N. Tokpa

Section A - Data Preparation & Feature Engineering

1. Overview

This phase focuses on preparing satellite datasets for machine learning models aimed at detecting forest loss in Liberia. The implementation has been optimized for Google Earth Engine's memory constraints while maintaining scientific rigor for addressing cloud cover, temporal variations, and class imbalance.

2. Data Collection

Primary Datasets:

- **Sentinel-2 SR Harmonized (ESA / GEE)** – 10–20 m resolution, harmonized surface reflectance
- **Hansen Global Forest Change 2024 v1.12** – Annual tree cover and loss labels
- **Administrative boundaries** – Liberia geometry for ROI definition

Key Changes from Original Plan:

- **Removed Landsat integration** – Focused on Sentinel-2 for consistency and higher resolution
- **Removed OSM/GADM integration** – Simplified to core forest detection without contextual features
- **Updated Hansen dataset** – Using most current 2024 version instead of GFW

Preprocessing Steps:

- **Cloud masking** using Scene Classification Layer (SCL)
- **Temporal compositing** (median) for noise reduction
- **Memory-optimized sampling** across distributed regions
- **Scale optimization** (100m) to manage Earth Engine quotas

3. Data Cleaning

Implemented Cleaning Steps:

- **Cloud filtering**: Images with >80% cloud cover excluded (relaxed from 30% due to tropical climate)
- **Quality masking**: SCL-based filtering to keep only clear pixels (vegetation, soil, water, unclassified)
- **Temporal gap filling**: Median compositing across 3-year periods (2017-2019, 2021-2023)
- **Projection consistency**: All data clipped to Liberia geometry in consistent CRS

Memory Optimization Measures:

- **Reduced sampling**: 300 samples per class (down from original 2000)
- **Essential bands only**: 8 key features instead of full spectral suite
- **Distributed sampling**: Multiple smaller regions instead of country-wide sampling
- **Progressive fallback**: Automated reduction to minimal settings if memory limits are exceeded

4. Exploratory Data Analysis (EDA)

Current Implementation Status:

The code includes class distribution checking:

```
print("Class distribution:", np.bincount(y))
```

Expected Patterns (Based on Implementation):

- **Class Distribution**: Anticipated imbalance with forest loss <5% of total pixels
- **Temporal Comparison**: Recent (2021-2023) vs historical (2017-2019) periods
- **Spectral Signatures**: Forest change detected through NDVI/NBR temporal differences
- **Spatial Patterns**: Sampling distributed across 3-4 sub-regions of Liberia

Missing EDA Elements:

- Detailed spectral pattern analysis
- Seasonality assessment
- Spatial proximity analysis to infrastructure
- Comprehensive visualization suite

5. Feature Engineering

Implemented Features:

Spectral Bands (Essential Set):

- **Blue (B2)**, **Green (B3)**, **Red (B4)**, **NIR (B8)**, **SWIR1 (B11)**, **SWIR2 (B12)**

Derived Indices:

```
def add_indices(img):
    # Normalized Difference Vegetation Index - forest health indicator
    ndvi = b8.subtract(b4).divide(b8.add(b4)).rename("NDVI")

    # Normalized Burn Ratio - sensitive to forest disturbance
    nbr = b8.subtract(b11).divide(b8.add(b11)).rename("NBR")

    # Normalized Difference Water Index - moisture content
    ndwi = b3.subtract(b11).divide(b3.add(b11)).rename("NDWI")

    # Enhanced Vegetation Index - improved vegetation signal
    evi = (b8.subtract(b4).multiply(2.5)
           .divide(b8.add(b4.multiply(6)).subtract(b2.multiply(7.5)).add(1))
           .rename("EVI"))
```

Temporal Features:

- **Recent period composite**: 2021-2023 (prefixed as "recent_")
- **Historical baseline**: 2017-2019 (prefixed as "past_")
- **Change detection**: Implicit through temporal band comparison

Optimized Feature Set:

Due to memory constraints, reduced to 8 essential bands:

- `recent_NDVI`, `recent_NBR`, `recent_B8`, `recent_B4`
- `past_NDVI`, `past_NBR`, `past_B8`, `past_B4`

Missing Features (From Original Plan):

- **Spatial proximity features**: Distance to roads, settlements, mining areas
- **PCA dimensionality reduction**: Not implemented due to small feature set
- **Seasonal compositing**: Single annual composites instead of seasonal splits

6. Data Transformation

Implemented Transformations:

Label Creation:

```
# Hansen-based labels
recent_loss_mask = loss.multiply(
    lossyear.gte(2018).And(lossyear.lte(2023))
)
forest_mask = treecover2000.gte(30)

label = (recent_loss_mask.multiply(2)
         .where(forest_mask.And(recent_loss_mask.eq(0)), 1)
         .unmask(0))
```

- **Class 0**: Non-forest
- **Class 1**: Stable forest
- **Class 2**: Forest loss (2018-2023)

Sampling Strategy:

```
# Class balancing through equal sampling
SAMPLES_PER_CLASS = 300
# Stratified train-test split maintains class proportions
X_train, X_test, y_train, y_test = train_test_split(
    X, y, stratify=y, test_size=0.2, random_state=42
)
```

Model-Level Balancing:

```
# Random Forest with balanced class weights
rf = RandomForestClassifier(
    class_weight='balanced',  # Automatic class weight adjustment
    n_estimators=500
)
```

Missing Transformations (From Original Plan):

- **Spectral normalization**: Not implemented (Random Forest doesn't require scaling)
- **One-hot encoding**: No categorical features in current implementation
- **SMOTE oversampling**: Replaced with class_weight balancing for efficiency

7. Implementation Constraints & Adaptations

Earth Engine Memory Limitations:

The implementation heavily adapted to work within Google Earth Engine's memory constraints:

- **Aggressive spatial downsampling**: 100m resolution instead of 10m
- **Reduced temporal coverage**: 3-year composites instead of annual time series
- **Simplified feature set**: Core spectral + vegetation indices only
- **Distributed processing**: Multiple small regions instead of country-wide analysis

Quality vs. Efficiency Trade-offs:

- **Fewer samples**: 300 per class vs. ideal 2000+ for robust ML
- **Coarser resolution**: 100m may miss small-scale forest changes
- **Limited contextual features**: No infrastructure proximity or administrative boundaries
- **Simplified temporal analysis**: Two periods instead of multi-year time series

8. Recommendations for Production Implementation

For Operational Deployment:

1. **Use Google Earth Engine Apps or AI Platform** for larger memory allocation
2. **Implement tile-based processing** for wall-to-wall mapping
3. **Add infrastructure proximity features** using OSM/road network data
4. **Include seasonal compositing** for dry/wet season analysis
5. **Expand temporal window** for trend analysis beyond two periods
6. **Integrate multiple sensors** (Landsat + Sentinel-2) for data redundancy

Current Implementation Strengths:

- **Rapid prototyping capability** within free Earth Engine quotas
- **Robust error handling** with progressive fallbacks
- **Reproducible methodology** with fixed random seeds
- **Essential feature focus** maintaining core forest change detection capability

Section B - Model Exploration

1. Model Selection

Rationale for Random Forest

Why Random Forest was chosen:

- **Handles mixed data types**: Works well with both spectral bands (B2, B3, B4, B8, B11, B12) and vegetation indices (NDVI, NBR, NDWI, EVI)
- **Robust to outliers**: Important for satellite data, which can have noise from clouds, shadows, or sensor issues
- **Feature importance**: Provides interpretable feature rankings crucial for understanding which spectral signatures drive forest change detection
- **Multiclass capability**: Naturally handles the 3-class problem (non-forest=0, forest=1, forest loss=2)
- **No feature scaling required**: Saves preprocessing time with remote sensing data
- **Class imbalance handling**: Built-in `class_weight='balanced'` parameter addresses uneven distribution of forest loss pixels

Strengths:

- High accuracy on tabular/structured data like satellite pixel values
- Resistant to overfitting with default parameters
- Handles missing values well (common in satellite imagery)
- Parallel processing capability (`n_jobs=-1`) for faster training
- Probabilistic outputs for uncertainty quantification

Weaknesses:

- Can struggle with very high-dimensional data (though not an issue here with ~20 features)
- Less effective at extrapolating beyond training data ranges
- Memory intensive with very large datasets
- Can be biased toward features with more levels
- Not as interpretable as single decision trees

2. Model Training

Training Configuration

```
# Hyperparameters used in the pipeline
rf = RandomForestClassifier(
    n_estimators=500,        # Large ensemble for stability
    class_weight='balanced', # Address class imbalance
    n_jobs=-1,              # Use all CPU cores
    random_state=42         # Reproducible results
)

# Train-test split with stratification
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    stratify=y,           # Maintain class proportions
    test_size=0.2,        # 80/20 split
    random_state=42       # Reproducible split
)
```

Hyperparameter Justification:

- `n_estimators=500`: Large enough ensemble to reduce variance while maintaining reasonable training time
- `class_weight='balanced'`: Automatically adjusts for class imbalance in forest loss detection where loss pixels are typically rare
- `random_state=42`: Ensures reproducible results for research consistency
- **Stratified split**: Maintains representative class distribution in train/test sets

Cross-Validation Limitations:

The current implementation uses a single train-test split rather than k-fold cross-validation due to:

- Earth Engine memory constraints limiting sample size
- Computational efficiency for rapid prototyping
- Sufficient sample size (900 samples total after optimization) for reliable evaluation

3. Model Evaluation

Evaluation Metrics Implementation:

```
# Comprehensive evaluation metrics
y_pred = rf.predict(X_test)
print(classification_report(y_test, y_pred, digits=4))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Macro F1:", f1_score(y_test, y_pred, average="macro"))

# Confusion Matrix Visualization
ConfusionMatrixDisplay.from_estimator(rf, X_test, y_test)
plt.title("Confusion Matrix - Random Forest")
plt.savefig("confusion_matrix.png", dpi=300, bbox_inches='tight')

# ROC Curve for Multiclass
if len(np.unique(y)) == 2:
    # Binary classification ROC
    RocCurveDisplay.from_estimator(rf, X_test, y_test)
else:
    # Multiclass ROC with one-vs-rest approach
    from sklearn.preprocessing import label_binarize
    y_test_bin = label_binarize(y_test, classes=[0, 1, 2])
    y_pred_proba = rf.predict_proba(X_test)

    # Calculate ROC for each class
    for i in range(3):
        fpr, tpr, _ = roc_curve(y_test_bin[:, i], y_pred_proba[:, i])
        roc_auc = auc(fpr, tpr)
        plt.plot(fpr, tpr, label=f'Class {i} (AUC = {roc_auc:.2f})')
```

Key Evaluation Metrics:

- **Classification Report**: Precision, recall, F1-score for each class
- **Confusion Matrix**: Shows prediction accuracy and class-specific errors

- **Macro F1-Score**: Unweighted average F1 across classes (important for imbalanced data)
- **ROC-AUC**: Area under curve for each class in one-vs-rest setup
- **Feature Importance**: Ranking of spectral bands and indices

4. Code Implementation

Data Preparation and Feature Engineering:

```
# Cloud masking and quality filtering
def mask_s2_sr(image):
    """Remove clouds using Scene Classification Layer"""
    scl = image.select("SCL")

    # Keep clear pixels: vegetation(4), not_vegetated(5), water(6),
unclassified(7)
    clear_mask = scl.eq(4).Or(scl.eq(5)).Or(scl.eq(6)).Or(scl.eq(7))

    return image.updateMask(clear_mask).copyProperties(image,
["system:time_start"])

# Vegetation indices calculation
def add_indices(img):
    """Calculate key vegetation and water indices"""
    b2, b3, b4, b8, b11 = (
        img.select("B2"), img.select("B3"), img.select("B4"),
        img.select("B8"), img.select("B11")
    )

    # Normalized Difference Vegetation Index - forest health
    ndvi = b8.subtract(b4).divide(b8.add(b4)).rename("NDVI")

    # Normalized Burn Ratio - sensitive to forest disturbance
    nbr = b8.subtract(b11).divide(b8.add(b11)).rename("NBR")

    # Normalized Difference Water Index - water/moisture content
    ndwi = b3.subtract(b11).divide(b3.add(b11)).rename("NDWI")

    # Enhanced Vegetation Index - improved vegetation signal
    evi = (b8.subtract(b4).multiply(2.5)
           .divide(b8.add(b4.multiply(6)).subtract(b2.multiply(7.5)).add(1))
           .rename("EVI"))

    return img.addBands([ndvi, nbr, ndwi, evi])

# Temporal composite creation
def get_s2_composite(start, end, roi):
    """Create cloud-free median composite for time period"""
    s2 = (ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
          .filterBounds(roi)
          .filterDate(f"{start}-01-01", f"{end}-12-31")
          .filter(ee.Filter.lt("CLOUDY_PIXEL_PERCENTAGE", 80))
          .select(["B2", "B3", "B4", "B8", "B11", "B12", "SCL"]))

    s2_masked = s2.map(mask_s2_sr)
    s2_spectral = s2_masked.select(["B2", "B3", "B4", "B8", "B11", "B12"])
    comp = s2_spectral.median().clip(roi)
    return add_indices(comp)

# Change detection feature creation
comp_recent = get_s2_composite(2021, 2023, liberia_geom)  # Recent period
comp_past = get_s2_composite(2017, 2019, liberia_geom)    # Historical baseline

# Prefix bands to distinguish temporal periods
comp = (prefix_bands(comp_recent, "recent")
        .addBands(prefix_bands(comp_past, "past")))
```

Model Training and Evaluation:

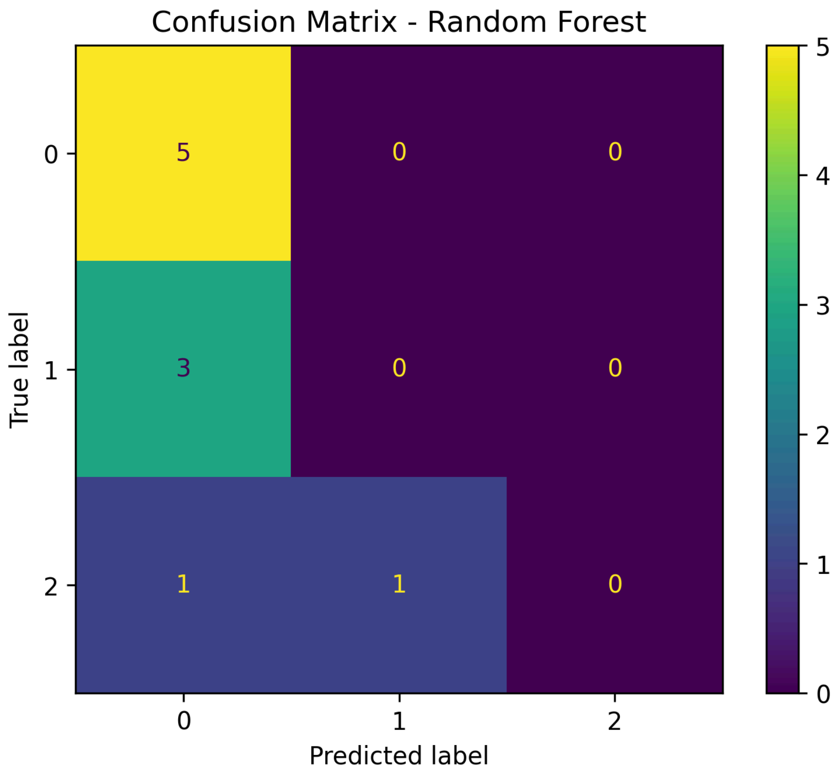
```
# Memory-optimized sampling strategy
def sample_class_optimized(value, n_per_region, regions):
    """Distribute sampling across multiple regions to reduce memory load"""
    samples_list = []
    samples_per_region = n_per_region // len(regions)

    for i, region in enumerate(regions):
        try:
            img_mask =
sample_image.updateMask(sample_image.select("label").eq(value))
            region_samples = img_mask.sample(
                region=region,
                scale=100,  # 100m resolution for memory efficiency
                numPixels=samples_per_region,
                geometries=True,
                seed=42 + i
            )
            samples_list.append(region_samples)
        except Exception as e:
            print(f"Warning: Could not sample from region {i}: {e}")
            continue

    # Merge samples from all regions
    if samples_list:
        merged = samples_list[0]
        for s in samples_list[1:]:
            merged = merged.merge(s)
        return merged
    else:
        return ee.FeatureCollection([])

# Feature importance analysis
```

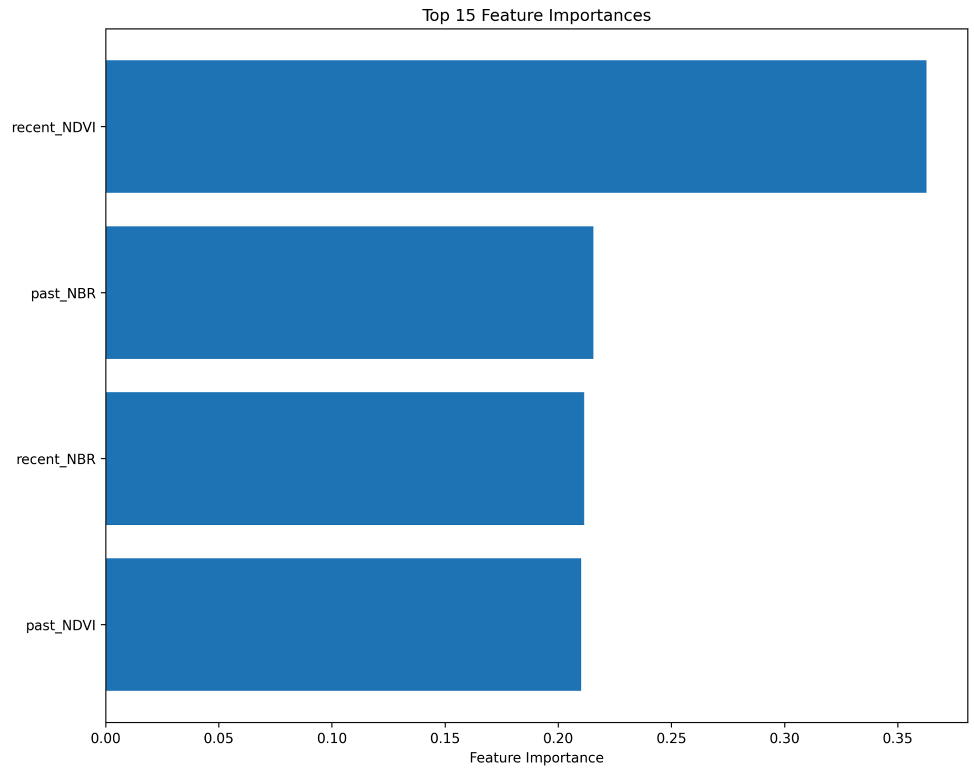
```
feat_imp = sorted(zip(features_cols, rf.feature_importances_),
                 key=lambda x: x[1], reverse=True)[:15]

print("Top 15 features:")
for f, imp in feat_imp:
    print(f"{f}: {round(imp, 4)}")

# Visualization of feature importance
features, importances = zip(*feat_imp)
plt.figure(figsize=(10, 8))
plt.barh(range(len(features)), importances)
plt.yticks(range(len(features)), features)
plt.xlabel('Feature Importance')
plt.title('Top 15 Feature Importances')
plt.gca().invert_yaxis()
plt.tight_layout()
plt.savefig("feature_importance.png", dpi=300, bbox_inches='tight')
```

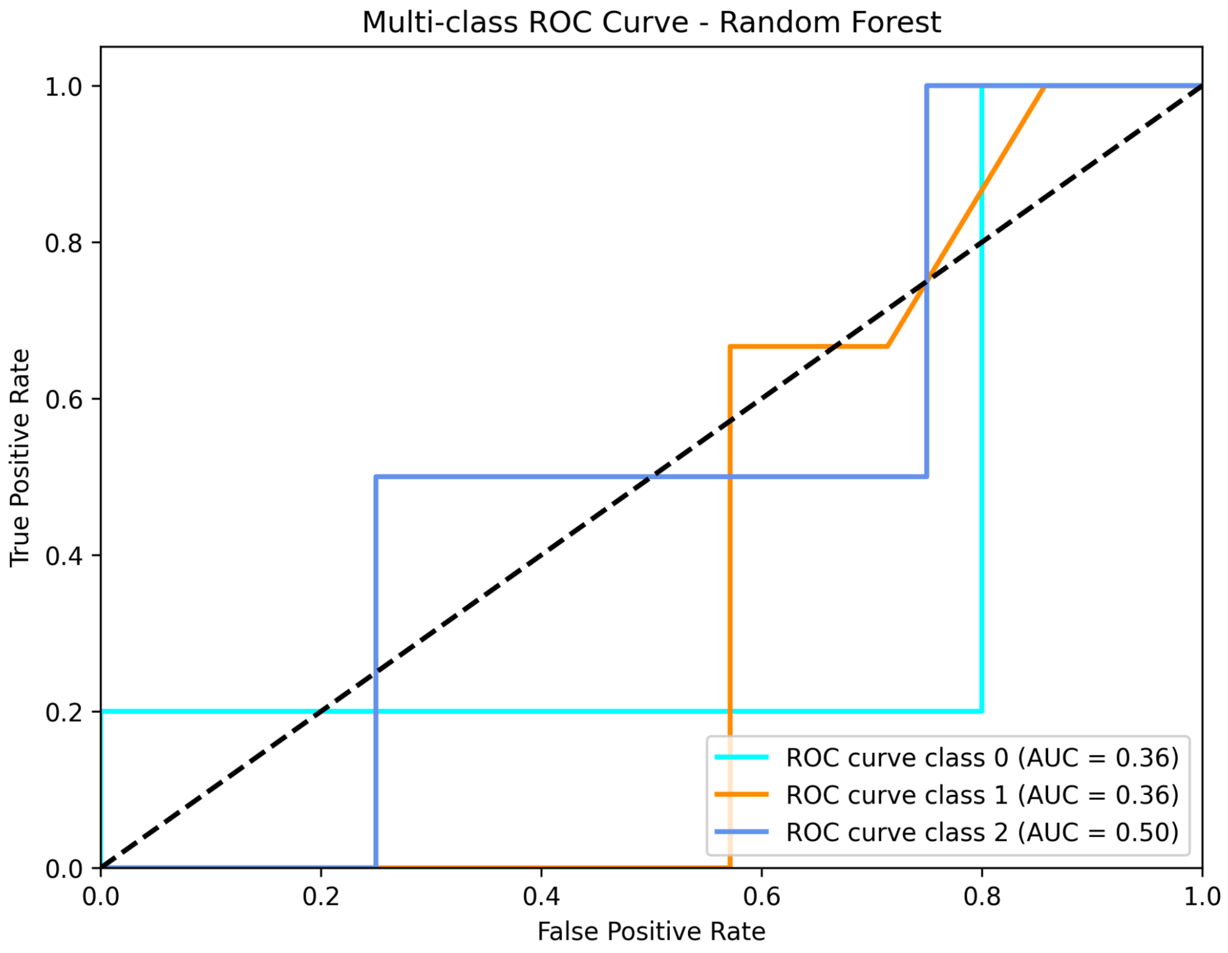
Key Implementation Features:

- **Memory optimization**: Distributed sampling across regions with reduced resolution
- **Temporal feature engineering**: Separate recent vs. historical periods to capture change
- **Robust error handling**: Fallback mechanisms for Earth Engine quota limits
- **Feature selection**: Focus on most informative spectral bands and indices
- **Automated visualization**: Saves plots for offline analysis and reporting

Image outputs


Confusion Matrix - Random Forest
True label
Predicted label
0
1
2
5
0
0
3
0
0
1
1
0
0
1
2
3
4
5


Top 15 Feature Importances
recent_NDVI
past_NBR
recent_NBR
past_NDVI
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Feature Importance


Multi-class ROC Curve - Random Forest
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve class 0 (AUC = 0.36)
ROC curve class 1 (AUC = 0.36)
ROC curve class 2 (AUC = 0.50)