# 

# **GreenLens ML Project Documentation**

Group 5 – GreenLens

Team Members:

- Fatu Kromah

- Alfred Nyeswa

- Sulaiman Barry

- Jerome N. Tokpa

- Joseph S. Lablah Jr

## **Model Refinement**

### **1. Overview**

Refined RandomForest classifier through systematic hyperparameter tuning to improve forest loss prediction accuracy and deployment readiness.

### **2. Model Evaluation**

**Initial Results:**

* Macro F1: 0.3447
* High class imbalance: 31.2% / 35.8% / 33.0%
* 31.3% low-confidence predictions (<70%)

**Improvement Areas:** Hyperparameter optimization, class balancing, feature engineering

### **3. Refinement Techniques**

* GridSearchCV for automated tuning
* Class weight balancing for imbalanced data
* Stratified cross-validation for consistent splits

### **4. Hyperparameter Tuning**

param\_grid = {

"n\_estimators": [100, 200, 500],

"max\_depth": [None, 10, 20],

"min\_samples\_split": [2, 5],

"min\_samples\_leaf": [1, 2]

}

**Best:** n\_estimators=500, max\_depth=20, min\_samples\_split=2, min\_samples\_leaf=1

### **5. Cross-Validation**

StratifiedKFold (3 splits) maintained class distribution across folds, ensuring reliable F1-macro scoring.

### **6. Feature Selection**

Excluded: ['label', 'lon', 'lat']. Top 8 features selected based on importance analysis (threshold >0.10).

## **Test Submission**

### **1. Overview**

Deployed production-ready Streamlit app with validation pipeline, monitoring dashboard, and automated testing.

### **2. Data Preparation**

* Validated: no missing values, numeric types, feature alignment
* Applied same preprocessing as training data
* Minimum 100 samples required

### **3. Model Application**

# Load model

model, features, metadata = load\_model()

# Predict

X\_input = data[features].values

predictions = model.predict(X\_input)

probabilities = model.predict\_proba(X\_input)

### **4. Test Metrics**

**Test Set (1000 samples):**

* Speed: 6,765 samples/sec
* Avg Confidence: 68.97%
* Class Distribution: Balanced (31-36%)

**Training vs Test:** Similar F1 scores indicate no overfitting, but both below target (0.75).

### **5. Model Deployment**

**Stack:** Streamlit (8501), Monitoring (8502), Docker, Health checks, JSONL logging

**Command:** ./deploy.sh production

### **6. Code Implementation**

# Training

grid = GridSearchCV(rf, param\_grid, cv=cv, scoring="f1\_macro")

grid.fit(X\_train, y\_train)

joblib.dump(grid.best\_estimator\_, "model.joblib")

# Validation

def validate\_dataframe(df, features):

missing = set(features) - set(df.columns)

if missing: return False, f"Missing: {missing}"

if df.isnull().any().any(): return False, "NaN values"

return True, None

# Prediction

preds = model.predict(X\_input)

confidence = model.predict\_proba(X\_input).max(axis=1)

## **Conclusion**

**Achieved:**

* Production infrastructure with monitoring
* 100% automated test pass rate
* Real-time drift detection

**Challenges:**

* F1 (0.3447) below threshold (0.75)
* Need more diverse training data

**Next Steps:** Collect additional data, try XGBoost/LightGBM, implement SMOTE

## **References**

* scikit-learn 1.3.0
* Streamlit docs
* GridSearchCV best practices