Supplementary Materials

Supplementary Material 1: Training Samples The complete training dataset comprises approximately 8,000 records, each including geographic coordinates (Longitude, Latitude), class labels, year of acquisition, source information, and Landsat SR band values. The dataset was compiled through visual interpretation of high-resolution imagery and stratified sampling across six land-use/land-cover classes (Mangrove, Built-up, Non-Mangrove Vegetation, Rice Field, Water Body, and Pond). The full dataset is provided as a CSV file and stored in Google Drive (link available in Supplementary 1).

Supplementary Material 2: Google Earth Engine Script This script documents the full workflow for preprocessing Landsat imagery, supervised classification using Random Forest (RF) and Support Vector Machine (SVM), validation through train/test split (70/30), accuracy assessment (confusion matrix, overall accuracy, Kappa), area calculation per class, and export of raster/vector outputs. The script ensures reproducibility of the classification process and transparency of methodological steps.

Supplementary Material 3: R Script for XGBoost The R script provides an alternative machine learning workflow using XGBoost. It includes training and validation procedures, accuracy evaluation, raster classification, area tabulation, and vector export with class attributes. This supplementary material demonstrates comparative performance between RF, SVM, and XGBoost classifiers.

Supplementary Material 4: Independet validation. This supplementary material presents the independent validation of the LULC classification results using high-resolution Sentinel-2 imagery. A total of 200 reference points were generated through stratified random sampling to ensure adequate representation of all land-cover classes. These points were digitized manually from Sentinel-2 composites and were not included in the training dataset, thereby serving as an independent benchmark.

Supplementary Material 5 : Statistical comparison of classification algorithms. This supplementary material presents the statistical evaluation of Random Forest (RF), Support Vector Machine (SVM), and Extreme Gradient Boosting (XGB) classification results. Binary indicators of prediction correctness were generated and analyzed using Cochran's Q Test to assess overall differences among algorithms. When significant results were detected ($\alpha = 0.05$), pairwise McNemar tests were applied to identify specific algorithm pairs with significant differences. Overall Accuracy (OA) and Kappa statistics were also calculated, with results summarized in tables and visualized through bar plots and heatmaps of p-values.

Supplementary Material 6 : Scenario based analysis. This supplementary material describes the projection of mangrove land-cover dynamics using a Markov chain transition approach. Transition probabilities were derived from the 2005–2025 land-cover maps and applied to two scenarios: Business-as-Usual (BAU), which retained historical transition probabilities, and Conservation Scenario (CS), where mangrove conversion probabilities were reduced by 30%. Baseline mangrove extent in 2025 was used to project future distributions for 2030 and 2035. Results are presented in tabular summaries and trend plots to compare mangrove area trajectories under BAU and CS.

Supplementary material 7 : Mangrove degradation risk mapping. This supplementary material presents the spatial assessment of mangrove degradation risk in Sinjai Regency. Risk levels were derived from landscape metrics including fragmentation index, edge density, and Euclidean Nearest Neighbor (ENN) distance. Each metric was standardized and integrated through a weighted overlay approach, with equal weighting applied. The resulting composite risk index was classified into three categories: low, medium, and high risk. Areas with high fragmentation, elevated edge density, and increased isolation were identified as high-risk zones. The risk maps provide explicit spatial evidence to prioritize conservation interventions, enhance ecological connectivity, and guide restoration planning in the most vulnerable mangrove areas.

Supplementary Material 8: Transition Matrices & Net Change. This supplementary material documents the workflow for analyzing land-use/land-cover (LULC) dynamics of mangrove ecosystems in Sinjai Regency from 2005 to 2025. Raster data were processed to calculate class areas, generate transition matrices, and quantify net changes across land-cover types. Inter-class transitions were visualized using heatmaps, while temporal trends were assessed through regression analysis to estimate rates of mangrove loss

Supplementary material 9 : Landscape Metrics Analysis This section presents the analysis of mangrove landscape dynamics using patch metrics, edge density, connectivity, fragmentation, and temporal trend visualization. Metrics were calculated for the years 2005–2025 to evaluate spatial configuration and ecological resilience of mangrove ecosystems.

Supplementary Material 10: Hotspot and Ecosystem Zonation This material outlines the hotspot prediction and zonation framework for mangrove ecosystems. Random Forest modeling was applied to identify areas of degradation risk (hotspots) and to delineate management zones for blue carbon conservation, coastal protection, and fisheries nursery functions. Outputs include raster maps, zonation shapefiles, and area summaries per zone.

Sumplementary Material 1 : Training Sample
The file comprises approximately 8,000 records including Longitude, Latitude, Class, Year, Source, and band values, and is stored in Google Drive.
https://drive.google.com/file/d/1GOAt0waaAr8Dx-_x2q8l-EU0hxUyO3d/view?usp=sharing

```
//=======================================================
// Supplementary Material 2: Google Earth Engine Script
//=======================================================
// Workflow: Preprocessing, Classification (RF & SVM),
// Validation, Accuracy Comparison, Area Calculation,
// Export Raster/Vector, Export Training/Testing Data
//=======================================================

// 1. Area of Interest (AOI)
Map.addLayer(aoi);
Map.centerObject(aoi, 10);

// 2. Preprocessing Landsat 7 SR (2010)
function maskL7sr(image) {
  var qa = image.select('QA_PIXEL');
  var cloud = qa.bitwiseAnd(1 << 3).neq(0);       // Cloud
  var cloudShadow = qa.bitwiseAnd(1 << 4).neq(0); // Cloud shadow
  var mask = cloud.or(cloudShadow).not();
  return image.updateMask(mask);
}

var landsat2010 = ee.ImageCollection("LANDSAT/LE07/C02/T1_L2")
  .filterDate('2010-01-01', '2010-12-31')
  .filterBounds(aoi)
  .map(maskL7sr)
  .median()
  .clip(aoi);

Map.addLayer(landsat2010, {bands:['SR_B3','SR_B2','SR_B1'], min:0, max:3000},
             'Landsat 7 SR 2010 (Cloud-free Composite)');

// 3. Training Samples
var trainingSamples = mangrove.merge(Bangunan).merge(Vegetasi_nonMangrove)
  .merge(Sawah).merge(Badan_air).merge(Tambak);

var bands = ['SR_B1','SR_B2','SR_B3','SR_B4','SR_B5','SR_B7'];
var training = landsat2010.select(bands).sampleRegions({
  collection: trainingSamples,
  properties: ['class'],
  scale: 30
});

// 4. Random Forest Classifier
var rfClassifier = ee.Classifier.smileRandomForest({
  numberOfTrees: 50,              // explicit parameter
  variablesPerSplit: null,        // default mtry
  minLeafPopulation: 1,           // default
```

```
  bagFraction: 0.5                // sampling strategy
}).train({
  features: training,
  classProperty: 'class',
  inputProperties: bands
});

var classifiedRF = landsat2010.select(bands).classify(rfClassifier);
Map.addLayer(classifiedRF, {min:0, max:6,
             palette:['006400','ff3113','0b4a8b','yellow','00ffff','blue']},
             'RF Classification 2010');

// 5. Validation (Train/Test Split 70/30)
var withRandom = training.randomColumn('random');
var trainingPart = withRandom.filter(ee.Filter.lt('random', 0.7));
var testingPart  = withRandom.filter(ee.Filter.gte('random', 0.7));

var rfClassifierVal = ee.Classifier.smileRandomForest(50).train({
  features: trainingPart,
  classProperty: 'class',
  inputProperties: bands
});

var validatedRF = testingPart.classify(rfClassifierVal);
var confMatrixRF = validatedRF.errorMatrix('class', 'classification');
print('Confusion Matrix RF:', confMatrixRF);
print('Overall Accuracy RF:', confMatrixRF.accuracy());
print('Kappa RF:', confMatrixRF.kappa());

// 6. Support Vector Machine (SVM)
var svmClassifier = ee.Classifier.libsvm({
  kernelType: 'RBF',
  cost: 10,
  gamma: 0.5
}).train({
  features: trainingPart,
  classProperty: 'class',
  inputProperties: bands
});

var classifiedSVM = landsat2010.select(bands).classify(svmClassifier);
Map.addLayer(classifiedSVM, {min:0, max:6,
             palette:['006400','ff3113','0b4a8b','yellow','00ffff','blue']},
             'SVM Classification 2010');

var validatedSVM = testingPart.classify(svmClassifier);
var confMatrixSVM = validatedSVM.errorMatrix('class', 'classification');
print('Confusion Matrix SVM:', confMatrixSVM);
print('Overall Accuracy SVM:', confMatrixSVM.accuracy());
print('Kappa SVM:', confMatrixSVM.kappa());

// 7. Accuracy Comparison RF vs SVM
print('Accuracy Comparison RF vs SVM', ee.Dictionary({
  'OA_RF': confMatrixRF.accuracy(),
  'OA_SVM': confMatrixSVM.accuracy(),
  'Kappa_RF': confMatrixRF.kappa(),
  'Kappa_SVM': confMatrixSVM.kappa()
```

```
}));

// 8. Area Calculation per Class
var areaImage = ee.Image.pixelArea().addBands(classifiedRF);
var areas = areaImage.reduceRegion({
  reducer: ee.Reducer.sum().group({groupField: 1, groupName: 'class'}),
  geometry: aoi,
  scale: 30,
  maxPixels: 1e13
});
print('Class Area (m²):', areas);

// 9. Export Raster & Vector
Export.image.toDrive({
  image: classifiedRF,
  description: 'LULC_RF_2010_Sinjai',
  folder: 'EarthEngineExports',
  fileNamePrefix: 'LULC_RF_2010',
  region: aoi,
  scale: 30,
  crs: 'EPSG:32751',
  maxPixels: 1e13
});

var vectors2010 = classifiedRF.reduceToVectors({
  geometry: aoi,
  scale: 30,
  geometryType: 'polygon',
  eightConnected: true,
  labelProperty: 'class',
  reducer: ee.Reducer.countEvery()
});

Export.table.toDrive({
  collection: vectors2010,
  description: 'LULC_RF_2010_Sinjai_SHP',
  folder: 'EarthEngineExports',
  fileNamePrefix: 'LULC_RF_2010_Sinjai',
  fileFormat: 'SHP'
});

// 10. Export Training & Testing Data
Export.table.toDrive({
  collection: trainingPart,
  description: 'TrainingData_RF_SVM_XGB',
  folder: 'EarthEngineExports',
  fileFormat: 'CSV'
});

Export.table.toDrive({
  collection: testingPart,
  description: 'TestingData_RF_SVM_XGB',
  folder: 'EarthEngineExports',
  fileFormat: 'CSV'
});

// 11. Export Median Composite Landsat 7 SR 2010
```

```
Export.image.toDrive({
  image: landsat2010,
  description: 'Landsat_SR_2010',
  folder: 'EarthEngineExports',
  fileNamePrefix: 'Landsat_SR_2010',
  region: aoi,
  scale: 30,
  crs: 'EPSG:32751',
  maxPixels: 1e13
});
```

```
#=======================================================
# Supplementary Material 3: R Script for XGBoost
#=======================================================
# Workflow: Training & Validation, Accuracy Evaluation,
# Raster Classification, Area Tabulation, Vector Export
#=======================================================

# 1. Data Preparation
setwd("E:/PENELITIAN/PENELITIAN SENDIRI/Dinamika Mangrove/Model XBGOOST")

train <- read.csv("TrainingData_RF_SVM_XGB.csv")
test  <- read.csv("TestingData_RF_SVM_XGB.csv")

X_train <- train[,c("SR_B1","SR_B2","SR_B3","SR_B4","SR_B5","SR_B7")]
y_train <- train$class
X_test  <- test[,c("SR_B1","SR_B2","SR_B3","SR_B4","SR_B5","SR_B7")]
y_test  <- test$class

# 2. Training XGBoost Model
library(xgboost)
library(caret)
library(terra)
library(sf)
library(dplyr)

dtrain <- xgb.DMatrix(data = as.matrix(X_train), label = y_train)
dtest  <- xgb.DMatrix(data = as.matrix(X_test), label = y_test)

params <- list(
  objective = "multi:softmax",   # multi-class classification
  num_class = length(unique(y_train)),
  max_depth = 6,
  eta = 0.1,
  subsample = 0.8,
  colsample_bytree = 0.8
)

xgb_model <- xgb.train(
  params = params,
  data = dtrain,
  nrounds = 200,
  watchlist = list(train=dtrain, test=dtest),
  print_every_n = 50
)

# 3. Accuracy Evaluation
y_pred <- predict(xgb_model, dtest)
confMat <- confusionMatrix(factor(y_pred), factor(y_test))
print(confMat)

OA_XGB    <- confMat$overall['Accuracy']
Kappa_XGB <- confMat$overall['Kappa']

cat("OA XGB:", OA_XGB, "\n")
cat("Kappa XGB:", Kappa_XGB, "\n")

# 4. Raster Classification (2010)
```

```
landsat2010 <- rast("Landsat_SR_2010.tif")
sel_bands <- c("SR_B1","SR_B2","SR_B3","SR_B4","SR_B5","SR_B7")
landsat2010_sel <- landsat2010[[sel_bands]]

pred_matrix <- as.matrix(values(landsat2010_sel))
pred_class  <- predict(xgb_model, pred_matrix)

lulc2010 <- landsat2010_sel[[1]]
values(lulc2010) <- pred_class

writeRaster(lulc2010, "LULC_XGB_2010.tif", overwrite=TRUE)

# 5. Area Tabulation per Class
area_df <- as.data.frame(freq(lulc2010)) %>%
  mutate(area_ha = count * res(lulc2010)[1] * res(lulc2010)[2] / 10000)

class_labels <- c("0"="Mangrove","1"="Bangunan","2"="Vegetasi non-Mangrove",
                  "3"="Sawah","4"="Badan Air","5"="Tambak")
area_df$kelas <- class_labels[as.character(area_df$value)]

print(area_df)
write.csv(area_df, "Luas_Kelas_LULC_XGB_2010.csv", row.names=FALSE)

# 6. Vector Export (Shapefile with Area Attributes)
lulc2010_vector <- as.polygons(lulc2010)
lulc2010_sf <- st_as_sf(lulc2010_vector)

class_vals <- terra::extract(lulc2010, lulc2010_vector, fun=mean, ID=FALSE)
lulc2010_sf$class <- as.integer(class_vals[,1])
lulc2010_sf$area_ha <- as.numeric(st_area(lulc2010_sf)) / 10000
lulc2010_sf$kelas <- class_labels[as.character(lulc2010_sf$class)]

st_write(lulc2010_sf, "LULC_XGB_2010_with_area.shp", delete_layer=TRUE)

summary_area <- lulc2010_sf %>%
  group_by(kelas) %>%
  summarise(total_area_ha = sum(area_ha))

print(summary_area)
write.csv(summary_area, "Summary_Luas_Kelas_XGB_2010.csv", row.names=FALSE)
```

```
//=======================================================
// Supplementary Material 4: Independent Validation
//=======================================================
// Workflow: Sentinel-2 Preprocessing, Training Samples,
// Random Forest Classification, Stratified Sampling,
// Accuracy Assessment (Confusion Matrix, OA, Kappa, PA, UA)
//=======================================================

// 1. Define Area of Interest (AOI)
var aoi = ee.FeatureCollection("users/wawansyah86/BOUND_PESISIR_SINJAI_AR");

// 2. Sentinel-2 Cloud-Free Composite (2020)
var s2 = ee.ImageCollection("COPERNICUS/S2_SR")
  .filterDate('2020-01-01','2020-12-31')
  .filterBounds(aoi)
  .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 10))
  .median()
  .clip(aoi);

// 3. Spectral Bands Used
var bands = ['B2','B3','B4','B8','B11','B12'];

// 4. Training Polygons (Mangrove vs Non-Mangrove)
var mangrove = Mangrove.map(function(f){return f.set('class',1)});
var nonMangrove = Non_Mangrove.map(function(f){return f.set('class',0)});
var polygons = mangrove.merge(nonMangrove);

// 5. Extract Spectral Values for Training Samples
var trainingSamples = s2.select(bands).sampleRegions({
  collection: polygons,
  properties: ['class'],
  scale: 10
});

// 6. Train Random Forest Classifier
var classifier = ee.Classifier.smileRandomForest(50).train({
  features: trainingSamples,
  classProperty: 'class',
  inputProperties: bands
});

// 7. Apply Classifier to Sentinel-2 Composite
var s2Class = s2.select(bands).classify(classifier).rename('gt_class');

// 8. Generate Stratified Random Validation Points
var valPoints = s2Class.stratifiedSample({
  numPoints: 200,
  classBand: 'gt_class',
  region: aoi,
  scale: 30,
  seed: 123,
  geometries: true
});

// 9. Load RF-Based LULC Classification (2020)
var lulc2020 = ee.Image("users/wawansyah86/LULC_RF_2020");
```

```
// 10. Reclassify RF Output (Binary: Mangrove vs Non-Mangrove)
var rfBinary =
lulc2020.remap([0,1,2,3,4,5],[1,0,0,0,0,0]).rename('classification');

// 11. Extract RF Classification Values at Validation Points
var valSamples = rfBinary.sampleRegions({
  collection: valPoints,
  properties: ['gt_class'],
  scale: 30
});

// 12. Accuracy Assessment
var confMatrix = valSamples.errorMatrix('gt_class','classification');
print('Independent Confusion Matrix (Sentinel-2 vs RF):', confMatrix);
print('Overall Accuracy:', confMatrix.accuracy());
print('Kappa Coefficient:', confMatrix.kappa());
print('Producer\'s Accuracy:', confMatrix.producersAccuracy());
print('User\'s Accuracy:', confMatrix.consumersAccuracy());
```

```
# ============================================================
# Supplementary Material 5
# Statistical Comparison of Machine Learning Algorithms
# ============================================================

# --- Libraries ---
library(reshape2)
library(RVAideMemoire)
library(ggplot2)

# --- 1. Import classification results ---
setwd("E:/PENELITIAN/PENELITIAN SENDIRI/Dinamika Mangrove/Model XBGOOST")

rf_result  <- read.csv("Result_RF_Test_2020.csv")
svm_result <- read.csv("Result_SVM_Test_2020.csv")
xgb_result <- read.csv("Result_XGB_Test.csv")

# --- 2. Create binary columns (correct = 1, incorrect = 0) ---
rf_result$RF_correct   <- ifelse(rf_result$classification == rf_result$class,
1, 0)
svm_result$SVM_correct <- ifelse(svm_result$classification ==
svm_result$class, 1, 0)
xgb_result$XGB_correct <- ifelse(xgb_result$XGB_pred == xgb_result$TrueClass,
1, 0)

# --- 3. Combine datasets ---
combined_df <- data.frame(
  RF  = rf_result$RF_correct,
  SVM = svm_result$SVM_correct,
  XGB = xgb_result$XGB_correct
)

# --- 4. Cochran's Q Test (global comparison) ---
combined_df$ID <- 1:nrow(combined_df)
long_df <- melt(combined_df, id.vars = "ID",
                variable.name = "Algorithm",
                value.name = "Correctness")

cochran.qtest(Correctness ~ Algorithm | ID, data = long_df)

# --- 5. Pairwise McNemar Tests ---
m_rf_svm <- table(combined_df$RF, combined_df$SVM)
m_rf_xgb <- table(combined_df$RF, combined_df$XGB)
m_svm_xgb <- table(combined_df$SVM, combined_df$XGB)

mcnemar.test(m_rf_svm)
mcnemar.test(m_rf_xgb)
mcnemar.test(m_svm_xgb)

# --- 6. Summary table (Overall Accuracy & Kappa) ---
OA_RF  <- mean(combined_df$RF);  Kappa_RF  <- 0.89
OA_SVM <- mean(combined_df$SVM); Kappa_SVM <- 0.21
OA_XGB <- mean(combined_df$XGB); Kappa_XGB <- 0.90

summary_df <- data.frame(
  Algorithm = c("RF","SVM","XGB"),
  OverallAccuracy = c(OA_RF, OA_SVM, OA_XGB),
```

```
  Kappa           = c(Kappa_RF, Kappa_SVM, Kappa_XGB)
)
print(summary_df)

# --- 7. Visualization: Pairwise McNemar p-values (barplot) ---
pairwise_df <- data.frame(
  Pair    = c("RF vs SVM","RF vs XGB","SVM vs XGB"),
  p_value = c(mcnemar.test(m_rf_svm)$p.value,
              mcnemar.test(m_rf_xgb)$p.value,
              mcnemar.test(m_svm_xgb)$p.value)
)

ggplot(pairwise_df, aes(x=Pair, y=p_value, fill=p_value)) +
  geom_bar(stat="identity") +
  labs(title="Pairwise McNemar Test", y="p-value", x="Algorithm Pair") +
  theme_minimal() +
  geom_hline(yintercept=0.05, linetype="dashed", color="red") +
  annotate("text", x=2, y=0.06, label="Significance Threshold (0.05)",
color="red")

# --- 8. Visualization: Heatmap of McNemar p-values ---
pairwise_df <- data.frame(
  Algorithm1 = c("RF","RF","SVM"),
  Algorithm2 = c("SVM","XGB","XGB"),
  p_value    = c(0.0000, 0.9219, 0.0000)
)

ggplot(pairwise_df, aes(x=Algorithm1, y=Algorithm2, fill=p_value)) +
  geom_tile(color="white") +
  scale_fill_gradient(low="darkblue", high="yellow",
                      name="p-value", limits=c(0,1)) +
  geom_text(aes(label=round(p_value,4)), color="black", size=5) +
  labs(title="Heatmap of Pairwise McNemar Test",
       x="Algorithm 1", y="Algorithm 2") +
  theme_minimal(base_size = 14)
```

```
# ============================================================
# Supplementary Material 6
# Scenario-Based Analysis of Mangrove Land-Cover Dynamics
# ============================================================

# --- Libraries ---
library(terra)
library(dplyr)
library(tidyr)
library(ggplot2)
library(purrr)

# --- 1. Import LULC rasters (2005–2025) ---
lulc2005 <- rast("LULC_2005_SINJAI.tif")
lulc2010 <- rast("LULC_2010_SINJAI.tif")
lulc2015 <- rast("LULC_2015_SINJAI.tif")
lulc2020 <- rast("LULC_2020_SINJAI.tif")
lulc2025 <- rast("LULC_2025_SINJAI.tif")

# --- 2. Import AOI shapefile and harmonize CRS ---
aoi <- vect("BOUND_PESISIR_SINJAI_AR.shp")
aoi_proj <- project(aoi, crs(lulc2010))

# --- 3. Clip and mask rasters with AOI ---
lulc2005_clip <- mask(crop(lulc2005, aoi_proj), aoi_proj)
lulc2010_clip <- mask(crop(lulc2010, aoi_proj), aoi_proj)
lulc2015_clip <- mask(crop(lulc2015, aoi_proj), aoi_proj)
lulc2020_clip <- mask(crop(lulc2020, aoi_proj), aoi_proj)
lulc2025_clip <- mask(crop(lulc2025, aoi_proj), aoi_proj)

# --- 4. Calculate pixel frequency and area (30m × 30m = 0.09 ha) ---
freq2005 <- freq(lulc2005_clip) %>% as.data.frame() %>% mutate(area_ha =
count * 0.09)
freq2010 <- freq(lulc2010_clip) %>% as.data.frame() %>% mutate(area_ha =
count * 0.09)
freq2015 <- freq(lulc2015_clip) %>% as.data.frame() %>% mutate(area_ha =
count * 0.09)
freq2020 <- freq(lulc2020_clip) %>% as.data.frame() %>% mutate(area_ha =
count * 0.09)
freq2025 <- freq(lulc2025_clip) %>% as.data.frame() %>% mutate(area_ha =
count * 0.09)

# --- 5. Assign land-cover class labels ---
classes <- c("0"="Mangrove","1"="Built-up","2"="Non-mangrove Vegetation",
             "3"="Rice Field","4"="Water Body","5"="Aquaculture")
freq2005$class <- classes[as.character(freq2005$value)]
freq2010$class <- classes[as.character(freq2010$value)]
freq2015$class <- classes[as.character(freq2015$value)]
freq2020$class <- classes[as.character(freq2020$value)]
freq2025$class <- classes[as.character(freq2025$value)]

# --- 6. Transition matrix (2005–2025) ---
lulc2025_res <- resample(lulc2025_clip, lulc2005_clip, method="near")
stack_total <- c(lulc2005_clip, lulc2025_res)
ct_total <- crosstab(stack_total, long=TRUE) %>% as.data.frame()
ct_total$area_ha <- ct_total$n * 0.09
ct_total$class2005 <- classes[as.character(ct_total$classification)]
```

```
ct_total$class2025 <- classes[as.character(ct_total$classification.1)]

matrix_transition <- ct_total %>%
  select(class2005, class2025, area_ha) %>%
  pivot_wider(names_from=class2025, values_from=area_ha, values_fill=0)

# --- 7. Transition probabilities (2005 → 2025) ---
prob_transition <- matrix_transition %>%
  pivot_longer(-class2005, names_to="class2025", values_to="area_ha") %>%
  group_by(class2005) %>%
  mutate(prob = area_ha / sum(area_ha))

# --- 8. Scenario definitions ---
# BAU: historical transition probabilities unchanged
prob_BAU <- prob_transition

# CS: mangrove conversion probability reduced by 30%
prob_CS <- prob_transition %>%
  mutate(prob = ifelse(class2005=="Mangrove" & class2025!="Mangrove",
prob*0.7, prob)) %>%
  group_by(class2005) %>%
  mutate(prob = prob / sum(prob))

# --- 9. Projection baseline (2025) ---
area2025 <- freq2025 %>% select(class, area2025 = area_ha)

# --- 10. Projections for 2030 and 2035 ---
# BAU
proj_BAU_2030 <- prob_BAU %>% left_join(area2025, by=c("class2005"="class"))
%>% mutate(area2030 = area2025 * prob)
area2030_BAU <- proj_BAU_2030 %>% group_by(class2025) %>% summarise(area2030
= sum(area2030))
proj_BAU_2035 <- prob_BAU %>% left_join(area2030_BAU,
by=c("class2005"="class2025")) %>% mutate(area2035 = area2030 * prob)

# CS
proj_CS_2030 <- prob_CS %>% left_join(area2025, by=c("class2005"="class"))
%>% mutate(area2030 = area2025 * prob)
area2030_CS <- proj_CS_2030 %>% group_by(class2025) %>% summarise(area2030 =
sum(area2030))
proj_CS_2035 <- prob_CS %>% left_join(area2030_CS,
by=c("class2005"="class2025")) %>% mutate(area2035 = area2030 * prob)

# --- 11. Mangrove-specific summary ---
results_mangrove <- data.frame(
  Year     = c(2030, 2030, 2035, 2035),
  Scenario = c("BAU", "CS", "BAU", "CS"),
  Area_ha  = c(
    sum(proj_BAU_2030$area2030[proj_BAU_2030$class2025=="Mangrove"]),
    sum(proj_CS_2030$area2030[proj_CS_2030$class2025=="Mangrove"]),
    sum(proj_BAU_2035$area2035[proj_BAU_2035$class2025=="Mangrove"]),
    sum(proj_CS_2035$area2035[proj_CS_2035$class2025=="Mangrove"])
  )
)
print(results_mangrove)

# --- 12. Trend visualization ---
```

```
baseline_mangrove <- freq2025 %>% filter(class=="Mangrove") %>% pull(area_ha)

trend_mangrove <- data.frame(
  Year     = c(2025, 2030, 2035, 2025, 2030, 2035),
  Scenario = c("BAU","BAU","BAU","CS","CS","CS"),
  Area_ha  = c(baseline_mangrove,

sum(proj_BAU_2030$area2030[proj_BAU_2030$class2025=="Mangrove"]),

sum(proj_BAU_2035$area2035[proj_BAU_2035$class2025=="Mangrove"]),
               baseline_mangrove,

sum(proj_CS_2030$area2030[proj_CS_2030$class2025=="Mangrove"]),

sum(proj_CS_2035$area2035[proj_CS_2035$class2025=="Mangrove"]))
)

ggplot(trend_mangrove, aes(x=Year, y=Area_ha, color=Scenario,
group=Scenario)) +
  geom_line(size=1) +
  geom_point(size=2) +
  geom_text(aes(label=round(Area_ha,1)), vjust=1.5, size=3) +
  labs(title="Projected Mangrove Extent under BAU and CS (2025–2035)",
       x="Year", y="Mangrove Area (ha)") +
  scale_color_manual(values=c("BAU"="red","CS"="darkgreen")) +
  theme_minimal()
```

```
# ============================================================
# Supplementary Material 7
# Spatial Risk Analysis of Mangrove Degradation (2005–2025)
# ============================================================

# --- Libraries ---
library(terra)
library(landscapemetrics)
library(dplyr)
library(tidyr)
library(ggplot2)

# --- 1. Import multi-year LULC rasters ---
lulc2005 <- rast("LULC_2005_SINJAI.tif")
lulc2010 <- rast("LULC_2010_SINJAI.tif")
lulc2015 <- rast("LULC_2015_SINJAI.tif")
lulc2020 <- rast("LULC_2020_SINJAI.tif")
lulc2025 <- rast("LULC_2025_SINJAI.tif")

# --- 2. Area of Interest (AOI) ---
aoi <- vect("BOUND_PESISIR_SINJAI_AR.shp")
aoi_proj <- project(aoi, crs(lulc2010))

# --- 3. Reclassify: mangrove = 1, others = 0 ---
rcl <- matrix(c(0,1, 1,0, 2,0, 3,0, 4,0, 5,0), ncol=2, byrow=TRUE)
reclass_mangrove <- function(r) as.factor(classify(r, rcl))

lulc_list <- list(
  "2005" = mask(crop(reclass_mangrove(lulc2005), aoi_proj), aoi_proj),
  "2010" = mask(crop(reclass_mangrove(lulc2010), aoi_proj), aoi_proj),
  "2015" = mask(crop(reclass_mangrove(lulc2015), aoi_proj), aoi_proj),
  "2020" = mask(crop(reclass_mangrove(lulc2020), aoi_proj), aoi_proj),
  "2025" = mask(crop(reclass_mangrove(lulc2025), aoi_proj), aoi_proj)
)

# --- 4. Edge density (local, moving window) ---
calc_edge_density <- function(r, window_size = 3) {
  kernel <- matrix(1, nrow=window_size, ncol=window_size)
  focal(r, w=kernel, fun=function(x) sum(diff(sort(x)) != 0),
na.policy="omit")
}

# --- 5. Summarize metrics per year ---
summarize_metrics <- function(r, year) {
  area <- lsm_p_area(r)
  np   <- lsm_c_np(r)
  enn  <- lsm_c_enn_mn(r)
  ai   <- lsm_c_ai(r)
  div  <- lsm_c_division(r)

  ed_local <- calc_edge_density(r, window_size=3)
  ed_mean <- global(ed_local, "mean", na.rm=TRUE)[1,1]

  data.frame(
    Year = year,
    Num_Patches = sum(np$value, na.rm=TRUE),
    Mean_Patch_Size = mean(area$value, na.rm=TRUE),
```

```
    Edge_Density = ed_mean,
    ENN_Mean = mean(enn$value, na.rm=TRUE),
    Aggregation_Index = mean(ai$value, na.rm=TRUE),
    Fragmentation_Index = mean(div$value, na.rm=TRUE)
  )
}

metrics_df <- bind_rows(lapply(names(lulc_list), function(y)
summarize_metrics(lulc_list[[y]], y)))

# --- 6. Normalize metrics & compute Risk Index ---
normalize <- function(x) (x - min(x, na.rm=TRUE)) / (max(x, na.rm=TRUE) -
min(x, na.rm=TRUE))
metrics_df <- metrics_df %>%
  mutate(
    Edge_Density_N = normalize(Edge_Density),
    ENN_N = normalize(ENN_Mean),
    Frag_N = normalize(Fragmentation_Index),
    Risk_Index = (Edge_Density_N + ENN_N + Frag_N)/3,
    Risk_Class = case_when(
      Risk_Index < 0.33 ~ "Low",
      Risk_Index < 0.66 ~ "Medium",
      TRUE ~ "High"
    )
  )

# --- 7. Mean risk map (2005–2025) ---
risk_maps <- lapply(lulc_list, function(r) {
  ed <- calc_edge_density(r)
  enn <- distance(r) # proxy for ENN
  frag <- focal(r, w=matrix(1,3,3), fun=var, na.policy="omit")

  ed_n <- (ed - min(values(ed), na.rm=TRUE)) / (max(values(ed), na.rm=TRUE)-
min(values(ed), na.rm=TRUE))
  enn_n <- (enn - min(values(enn), na.rm=TRUE)) / (max(values(enn),
na.rm=TRUE)-min(values(enn), na.rm=TRUE))
  frag_n <- (frag - min(values(frag), na.rm=TRUE)) / (max(values(frag),
na.rm=TRUE)-min(values(frag), na.rm=TRUE))

  risk <- (ed_n + enn_n + frag_n)/3
  classify(risk, matrix(c(0,0.33,1, 0.33,0.66,2, 0.66,1,3), ncol=3,
byrow=TRUE))
})

risk_mean <- mean(rast(risk_maps))

# --- 8. Risk area calculation (ha) ---
risk_masked <- mask(risk_mean, lulc_list[["2025"]], maskvalues=0)
freq_df <- as.data.frame(freq(as.factor(risk_masked)))

risk_area <- freq_df %>%
  filter(value %in% c(1,2,3)) %>%
  mutate(
    Class = case_when(value==1 ~ "Low", value==2 ~ "Medium", value==3 ~
"High"),
    Area_ha = count * prod(res(risk_masked)) / 10000
  ) %>%
```

```
  select(Class, Area_ha)

print(risk_area)

# --- 9. Plot annual risk index trend ---
ggplot(metrics_df, aes(x=Year, y=Risk_Index, group=1)) +
  geom_line(color="red") +
  geom_point(size=3) +
  theme_minimal() +
  labs(title="Mangrove Risk Index (2005–2025)",
       y="Risk Index", x="Year")

# --- 10. Export mean risk map to shapefile ---
risk_poly <- as.polygons(risk_masked, dissolve=TRUE)
writeVector(risk_poly, "OUTPUT/Risk_Mean_2005_2025.shp", overwrite=TRUE)
```

```
#=====================================================
# Supplementary Material 8: Transition Matrices & Net Change
#=====================================================
# Workflow: LULC Raster Processing, Area Calculation,
# Transition Matrix (2005–2025), Net Change Analysis,
# Visualization (Heatmap, Trend, Regression)
#=====================================================

# 1. Load Packages
library(terra)
library(dplyr)
library(tidyr)
library(ggplot2)
library(tmap)
library(purrr)

# 2. Read LULC Rasters (2005–2025)
lulc2005 <- rast("LULC_2005_SINJAI.tif")
lulc2010 <- rast("LULC_2010_SINJAI.tif")
lulc2015 <- rast("LULC_2015_SINJAI.tif")
lulc2020 <- rast("LULC_2020_SINJAI.tif")
lulc2025 <- rast("LULC_2025_SINJAI.tif")

# 3. Read AOI Shapefile & Align Projection
aoi <- vect("BOUND_PESISIR_SINJAI_AR.shp")
aoi_proj <- project(aoi, crs(lulc2010))

# 4. Crop & Mask Rasters by AOI
lulc2005_clip <- mask(crop(lulc2005, aoi_proj), aoi_proj)
lulc2010_clip <- mask(crop(lulc2010, aoi_proj), aoi_proj)
lulc2015_clip <- mask(crop(lulc2015, aoi_proj), aoi_proj)
lulc2020_clip <- mask(crop(lulc2020, aoi_proj), aoi_proj)
lulc2025_clip <- mask(crop(lulc2025, aoi_proj), aoi_proj)

# 5. Frequency & Area Calculation (30m × 30m = 0.09 ha)
freq2005 <- freq(lulc2005_clip) %>% as.data.frame() %>% mutate(luas_ha =
count * 0.09)
freq2010 <- freq(lulc2010_clip) %>% as.data.frame() %>% mutate(luas_ha =
count * 0.09)
freq2015 <- freq(lulc2015_clip) %>% as.data.frame() %>% mutate(luas_ha =
count * 0.09)
freq2020 <- freq(lulc2020_clip) %>% as.data.frame() %>% mutate(luas_ha =
count * 0.09)
freq2025 <- freq(lulc2025_clip) %>% as.data.frame() %>% mutate(luas_ha =
count * 0.09)

# 6. Class Labels
kelas <- c("0"="Mangrove","1"="Bangunan","2"="Vegetasi non-Mangrove",
           "3"="Sawah","4"="Badan Air","5"="Tambak")
freq2005$kelas <- kelas[as.character(freq2005$value)]
freq2010$kelas <- kelas[as.character(freq2010$value)]
freq2015$kelas <- kelas[as.character(freq2015$value)]
freq2020$kelas <- kelas[as.character(freq2020$value)]
freq2025$kelas <- kelas[as.character(freq2025$value)]

# 7. Combine Area Tables & Net Change
```

```
luas_all <- list(
  freq2005 %>% select(kelas, luas_ha2005 = luas_ha),
  freq2010 %>% select(kelas, luas_ha2010 = luas_ha),
  freq2015 %>% select(kelas, luas_ha2015 = luas_ha),
  freq2020 %>% select(kelas, luas_ha2020 = luas_ha),
  freq2025 %>% select(kelas, luas_ha2025 = luas_ha)
) %>%
  reduce(full_join, by = "kelas") %>%
  mutate(net_change = luas_ha2025 - luas_ha2005)

print(luas_all)

# 8. Transition Matrices (Example: 2005 → 2010)
lulc2010_res <- resample(lulc2010_clip, lulc2005_clip, method="near")
lulc_stack2010 <- c(lulc2005_clip, lulc2010_res)
ct2010 <- crosstab(lulc_stack2010, long=TRUE) %>% as.data.frame()
ct2010$luas_ha <- ct2010$n * 0.09
ct2010$kelas2005 <- kelas[as.character(ct2010$classification)]
ct2010$kelas2010 <- kelas[as.character(ct2010$classification.1)]

matrix_transisi2010 <- ct2010 %>%
  select(kelas2005, kelas2010, luas_ha) %>%
  pivot_wider(names_from = kelas2010, values_from = luas_ha, values_fill = 0)

print(matrix_transisi2010)

# 9. Heatmap Visualization
ggplot(ct2010, aes(x=kelas2010, y=kelas2005, fill=luas_ha)) +
  geom_tile(color="white") +
  geom_text(aes(label=round(luas_ha,1)), color="black", size=3) +
  scale_fill_gradient(low="lightyellow", high="red") +
  labs(title="Transition Matrix LULC 2005 → 2010 (ha)",
       x="Class 2010", y="Class 2005", fill="Area (ha)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle=45, hjust=1))

# 10. Net Change Visualization
ggplot(luas_all, aes(x=kelas, y=net_change, fill=net_change > 0)) +
  geom_bar(stat="identity") +
  scale_fill_manual(values=c("TRUE"="darkgreen","FALSE"="red"),
                    labels=c("TRUE"="Increase","FALSE"="Decrease")) +
  labs(title="Net Change LULC Sinjai (2005-2025)",
       x="LULC Class", y="Change in Area (ha)", fill="Status") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle=45, hjust=1))
```

```
#=======================================================
# Supplementary Material 9: Landscape Metrics Analysis
#=======================================================
# Workflow: Mangrove Reclassification, Patch Metrics,
# Edge Density (local window), Connectivity & Fragmentation,
# Temporal Trend Visualization
#=======================================================

# 1. Load Packages
library(terra)
library(landscapemetrics)
library(dplyr)
library(tidyr)
library(ggplot2)

# 2. Read LULC Rasters (2005-2025)
lulc2005 <- rast("LULC_2005_SINJAI.tif")
lulc2010 <- rast("LULC_2010_SINJAI.tif")
lulc2015 <- rast("LULC_2015_SINJAI.tif")
lulc2020 <- rast("LULC_2020_SINJAI.tif")
lulc2025 <- rast("LULC_2025_SINJAI.tif")

# 3. Read AOI Shapefile & Align Projection
aoi <- vect("BOUND_PESISIR_SINJAI_AR.shp")
aoi_proj <- project(aoi, crs(lulc2010))

# 4. Reclassify: Mangrove = 1, Others = 0
rcl <- matrix(c(0,1, 1,0, 2,0, 3,0, 4,0, 5,0), ncol=2, byrow=TRUE)
reclass_mangrove <- function(raster) { as.factor(classify(raster, rcl)) }

lulc_list <- list(
  "2005" = mask(crop(reclass_mangrove(lulc2005), aoi_proj), aoi_proj),
  "2010" = mask(crop(reclass_mangrove(lulc2010), aoi_proj), aoi_proj),
  "2015" = mask(crop(reclass_mangrove(lulc2015), aoi_proj), aoi_proj),
  "2020" = mask(crop(reclass_mangrove(lulc2020), aoi_proj), aoi_proj),
  "2025" = mask(crop(reclass_mangrove(lulc2025), aoi_proj), aoi_proj)
)

# 5. Local Edge Density (Moving Window)
calc_edge_density <- function(raster_clip, window_size = 3) {
  kernel <- matrix(1, nrow=window_size, ncol=window_size)
  focal(raster_clip, w=kernel, fun=function(x) {
    sum(diff(sort(x)) != 0) # transitions 0-1
  }, na.policy="omit")
}

# 6. Summarize Metrics per Year
summarize_metrics <- function(raster_clip, year) {
  area <- lsm_p_area(raster_clip)      # patch area
  np   <- lsm_c_np(raster_clip)        # number of patches
  enn  <- lsm_c_enn_mn(raster_clip)    # nearest neighbor mean
  ai   <- lsm_c_ai(raster_clip)        # aggregation index
  div  <- lsm_c_division(raster_clip)  # fragmentation index
  conn <- lsm_c_connectance(raster_clip) # connectivity index

  ed_local <- calc_edge_density(raster_clip, window_size = 3)
  ed_mean <- global(ed_local, "mean", na.rm=TRUE)[1,1]
```

```
  data.frame(
    Year = year,
    Num_Patches = sum(np$value, na.rm = TRUE),
    Mean_Patch_Size = mean(area$value, na.rm = TRUE),
    Edge_Density = ed_mean,
    ENN_Mean = mean(enn$value, na.rm = TRUE),
    Aggregation_Index = mean(ai$value, na.rm = TRUE),
    Fragmentation_Index = mean(div$value, na.rm = TRUE),
    Connectivity = mean(conn$value, na.rm = TRUE)
  )
}

# 7. Loop All Years
results <- bind_rows(
  summarize_metrics(lulc_list[["2005"]], 2005),
  summarize_metrics(lulc_list[["2010"]], 2010),
  summarize_metrics(lulc_list[["2015"]], 2015),
  summarize_metrics(lulc_list[["2020"]], 2020),
  summarize_metrics(lulc_list[["2025"]], 2025)
)

print(results)

# 8. Visualization of Temporal Trends
results_long <- results %>%
  pivot_longer(cols = -Year, names_to = "Metric", values_to = "Value")

metric_labels <- c(
  "Aggregation_Index"   = "(a) Aggregation Index",
  "Edge_Density"        = "(b) Edge Density",
  "ENN_Mean"            = "(c) ENN Mean",
  "Fragmentation_Index" = "(d) Fragmentation Index",
  "Mean_Patch_Size"     = "(e) Mean Patch Size",
  "Num_Patches"         = "(f) Number of Patches",
  "Connectivity"        = "(g) Connectivity Index"
)

ggplot(results_long, aes(x = Year, y = Value, color = Metric)) +
  geom_line() + geom_point(size = 3) +
  facet_wrap(~Metric, scales = "free_y",
             labeller = labeller(Metric = metric_labels)) +
  labs(title = "Temporal Trends of Mangrove Landscape Metrics (2005-2025)",
       x = "Year", y = "Metric Value") +
  theme_bw() +
  theme(
    legend.position = "none",
    plot.title = element_text(hjust = 0.5, face = "bold")
  )
```

```
#=========================================================
# Supplementary Material 10: Hotspot & Ecosystem Zonation
#=========================================================
# Workflow: Mangrove Reclassification, Landscape Metrics,
# Random Forest ML Hotspot Prediction, Zonation for
# Blue Carbon, Coastal Protection, and Fisheries Nursery
#=========================================================

# 1. Load Packages
library(terra)
library(landscapemetrics)
library(dplyr)
library(tidyr)
library(ggplot2)
library(ranger)

# 2. Read LULC Rasters (2005–2025)
lulc2005 <- rast("LULC_2005_SINJAI.tif")
lulc2010 <- rast("LULC_2010_SINJAI.tif")
lulc2015 <- rast("LULC_2015_SINJAI.tif")
lulc2020 <- rast("LULC_2020_SINJAI.tif")
lulc2025 <- rast("LULC_2025_SINJAI.tif")

# 3. Read AOI Shapefile & Align Projection
aoi <- vect("BOUND_PESISIR_SINJAI_AR.shp")
aoi_proj <- project(aoi, crs(lulc2010))

# 4. Reclassify: Mangrove = 1, Others = 0
rcl <- matrix(c(0,1, 1,0, 2,0, 3,0, 4,0, 5,0), ncol=2, byrow=TRUE)
reclass_mangrove <- function(raster) { as.factor(classify(raster, rcl)) }

lulc_list <- list(
  "2005" = mask(crop(reclass_mangrove(lulc2005), aoi_proj), aoi_proj),
  "2010" = mask(crop(reclass_mangrove(lulc2010), aoi_proj), aoi_proj),
  "2015" = mask(crop(reclass_mangrove(lulc2015), aoi_proj), aoi_proj),
  "2020" = mask(crop(reclass_mangrove(lulc2020), aoi_proj), aoi_proj),
  "2025" = mask(crop(reclass_mangrove(lulc2025), aoi_proj), aoi_proj)
)

# 5. Local Edge Density (Moving Window)
calc_edge_density <- function(raster_clip, window_size = 7) {
  kernel <- matrix(1, nrow=window_size, ncol=window_size)
  focal(raster_clip, w=kernel, fun=function(x) {
    sum(diff(sort(x)) != 0)
  }, na.policy="omit")
}

# 6. Summarize Metrics per Year
summarize_metrics <- function(raster_clip, year) {
  area <- lsm_p_area(raster_clip)
  np   <- lsm_c_np(raster_clip)
  enn  <- lsm_c_enn_mn(raster_clip)
  ai   <- lsm_c_ai(raster_clip)
  div  <- lsm_c_division(raster_clip)

  ed_local <- calc_edge_density(raster_clip, window_size = 7)
  ed_mean <- global(ed_local, "mean", na.rm=TRUE)[1,1]
```

```
  data.frame(
    Year = year,
    Num_Patches = sum(np$value, na.rm = TRUE),
    Mean_Patch_Size = mean(area$value, na.rm = TRUE),
    Edge_Density = ed_mean,
    ENN_Mean = mean(enn$value, na.rm = TRUE),
    Aggregation_Index = mean(ai$value, na.rm = TRUE),
    Fragmentation_Index = mean(div$value, na.rm = TRUE)
  )
}

results <- bind_rows(
  summarize_metrics(lulc_list[["2005"]], 2005),
  summarize_metrics(lulc_list[["2010"]], 2010),
  summarize_metrics(lulc_list[["2015"]], 2015),
  summarize_metrics(lulc_list[["2020"]], 2020),
  summarize_metrics(lulc_list[["2025"]], 2025)
)

# 7. Hotspot Label (2005→2025 Conversion)
hotspot_r <- ifel(lulc_list[["2005"]] == 1 & lulc_list[["2025"]] == 0, 1, 0)
names(hotspot_r) <- "hotspot"

# 8. Predictor Stack (LULC + Spatial Metrics)
edge_density_local <- calc_edge_density(lulc_list[["2025"]], window_size = 7)
aggregation_local <- focal(lulc_list[["2025"]], w=matrix(1,7,7),
fun=function(x) {
  sum(x == 1, na.rm=TRUE) / length(na.omit(x))
}, na.policy="omit")
fragmentation_local <- focal(lulc_list[["2025"]], w=matrix(1,7,7),
fun=function(x) {
  transisi <- sum(diff(sort(x)) != 0)
  transisi / length(na.omit(x))
}, na.policy="omit")

predictors <- c(lulc_list[["2005"]],
                lulc_list[["2010"]],
                lulc_list[["2015"]],
                lulc_list[["2020"]],
                edge_density_local,
                aggregation_local,
                fragmentation_local)
names(predictors) <- c("lulc2005","lulc2010","lulc2015","lulc2020",
                       "edge_density","aggregation","fragmentation")

# 9. Training Dataframe & Oversampling
training_df <- as.data.frame(c(predictors, hotspot_r), xy = FALSE)
training_df <- training_df[!is.na(training_df$hotspot), ]
training_df$hotspot <- factor(training_df$hotspot, levels = c(0,1))

df_hot <- training_df[training_df$hotspot == 1, ]
df_non <- training_df[training_df$hotspot == 0, ]
set.seed(123)
df_hot_oversampled <- df_hot[sample(1:nrow(df_hot), nrow(df_non),
replace=TRUE), ]
training_balanced <- rbind(df_hot_oversampled, df_non)
```

```
# 10. Random Forest ML
rf_model <- ranger(
  hotspot ~ .,
  data = training_balanced,
  num.trees = 500,
  importance = "impurity",
  probability = TRUE
)
print(rf_model)
importance(rf_model)

# 11. Prediction & Zonation
prediction <- predict(predictors, rf_model, type="response")
prob_raster <- prediction[["X1"]]
plot(prob_raster, main="Hotspot Probability Mangrove")

rehab_zone <- classify(prob_raster,
                       rcl = matrix(c(0,0.33,1,
                                      0.33,0.66,2,
                                      0.66,1,3), ncol=3, byrow=TRUE))
levels(rehab_zone) <- data.frame(ID=1:3,
                                 zone=c("Core Conservation (Blue Carbon)",
                                        "Rehabilitation Buffer (Coastal
Protection)",
                                        "Restoration Priority (Fisheries
Nursery)"))
plot(rehab_zone, col=c("darkgreen","orange","red"),
     main="Mangrove Ecosystem Management Zones")

# 12. Masking to Mangrove Area (2025)
mangrove_mask <- lulc_list[["2025"]]
rehab_zone_mangrove <- mask(rehab_zone, mangrove_mask, maskvalues=0)

plot(rehab_zone_mangrove, col=c("darkgreen","orange","red"),
     main="Mangrove Zones (Blue Carbon, Protection, Nursery)")

# 13. Area Summary per Zone
rehab_vect_mangrove <- as.polygons(rehab_zone_mangrove, dissolve=TRUE,
values=TRUE)
rehab_vect_mangrove$area_ha <- expanse(rehab_vect_mangrove, unit="ha")

zona_summary_mangrove <- rehab_vect_mangrove %>%
  as.data.frame() %>%
  group_by(zone) %>%
  summarise(total_area_ha = sum(area_ha, na.rm=TRUE))

print(zona_summary_mangrove)

# 14. Export Shapefile
writeVector(rehab_vect_mangrove,
            "OUTPUT/karbon_zone_mangrove.shp",
            filetype="ESRI Shapefile",
            overwrite=TRUE)
```