**Land use Landcover Area Calculation Tool Preparation Using Python Script**

**1. Introduction**

**1.1 Problem Statement**

Manual LULC analysis requires multiple disjointed tools (e.g., clipping rasters, calculating areas in Excel, generating charts), leading to inefficiencies and potential errors. This toolbox automates the entire process, ensuring accuracy and saving time. The custom ArcGIS Python Toolbox (pyt) is designed to simplify the analysis of land use/land cover (LULC) data within a defined geographic area. It processes LULC raster data by clipping it to a user-defined boundary shapefile, calculates the area of each class, and outputs a detailed statistics table alongside a bar chart for visual interpretation. Additionally, it generates a dissolved vector shapefile, merging polygons by class and annotating them with area attributes. The output is automatically added to the current ArcGIS Pro map, ensuring an efficient and seamless workflow. The script utilizes **arcpy** for GIS operations, **numpy** and **pandas** for data processing, and **matplotlib** for visualizations, offering a comprehensive approach to spatial analysis and effective presentation.

**1.2. Toolbox Importance**

The toolbox is important for several reasons. It enhances automation by reducing repetitive tasks such as clipping, area calculation, and charting, thereby saving time and effort. It promotes standardization by ensuring a consistent methodology across different users and projects. The toolbox also enables seamless integration by combining raster and vector processing, data analysis, and visualization within a single platform. Its accessibility simplifies complex workflows, making it easier for non-experts such as planners and students to use. Additionally, it supports scalability by efficiently handling large datasets through tools like NumPy and Pandas.

**1.3 Goal of the project**

To automate LULC analysis by streamlining raster-to-vector conversion, area calculation, and visualization within ArcGIS Pro, enhancing efficiency and accuracy for data-driven decision-making.

**2. Methodology**

**2.1. Import Libraries**

**arcpy:** For GIS operations (projecting/clipping rasters, converting to polygons).

**os:** To manage file paths and folders.

**numpy and pandas:** To analyze raster data and export results to CSV.

**matplotlib:** To visualize area statistics as a bar chart.

**A screenshot of a computer

AI-generated content may be incorrect.**

**2.2. Toolbox & Tool Setup**

The Toolbox and Tool classes define the ArcGIS Pro toolbox structure. The Tool class sets metadata (name, description) and links to the execute method where the workflow runs.

**A computer code with text

AI-generated content may be incorrect.**

**2.3. Define Input Parameters**

The getParameterInfo method configures user inputs:

Raster layer, boundary shapefile, class values, pixel size, and output paths. Parameters are validated (e.g., ensuring the number of classes matches input values).

**A screenshot of a computer program

AI-generated content may be incorrect.**

**2.4. Execute the Workflow**

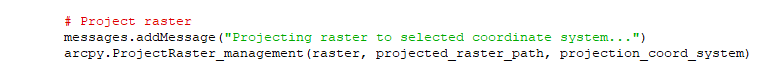
In execution, user inputs (e.g., raster path, class values) are parsed. Class values are split into a list of integers (e.g., 1,2,3 → [1,2,3]).

**A computer code with text

AI-generated content may be incorrect.**

**2.5. Project the Raster**

The input raster is projected to a user-specified coordinate system (e.g., UTM) using arcpy. ProjectRaster\_management. This ensures area calculations use linear units (meters/km²) instead of degrees.

****

**2.6. Clipping the Raster and Calculate Class Areas**

The projected raster is clipped to the boundary shapefile using arcpy. Clip management. This restricts analysis to the study area, ignoring pixels outside the boundary.

The clipped raster is converted to a NumPy array. For each class:

* Pixel count: np.sum(raster\_array == class\_val) counts pixels matching the class.
* Area: Calculated as (pixel count × pixel size²), converted to km².
* Results are stored in a panda DataFrame and saved to a CSV file.

**A screenshot of a computer code

AI-generated content may be incorrect.**

**2.7. Generate a Bar Chart**

A bar chart is plotted with matplotlib, showing area per class. The chart is saved as a PNG file for quick visual analysis.

**A white background with text

AI-generated content may be incorrect.**

**2.8. Convert Raster to Vector & Dissolve**

The clipped raster is converted to polygons (arcpy.RasterToPolygon). Polygons of the same class are merged using arcpy.Dissolve. A new field Area\_km2 is added to store polygon areas calculated with SHAPE@.getArea.

**A screenshot of a computer code

AI-generated content may be incorrect.**

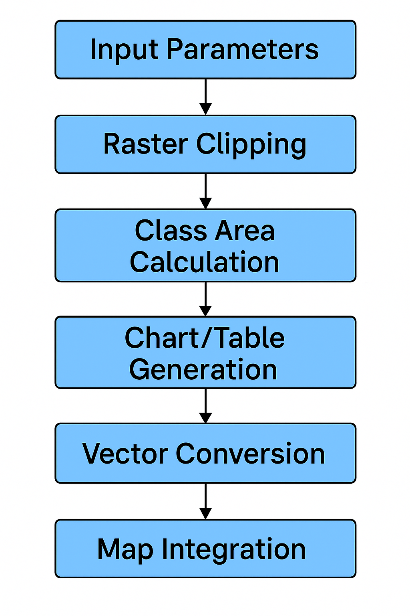
**2.9. Add Results to ArcGIS Pro Map**

The dissolved polygon layer is added to the active ArcGIS Pro map using arcpy.mp.ArcGISProject allowing users to visualize class boundaries directly in GIS.

**A computer code with text

AI-generated content may be incorrect.**

**2.10 Methodological Framework of the Toolbox**



**3.** **Running the toolbox in Arc GIS Pro**

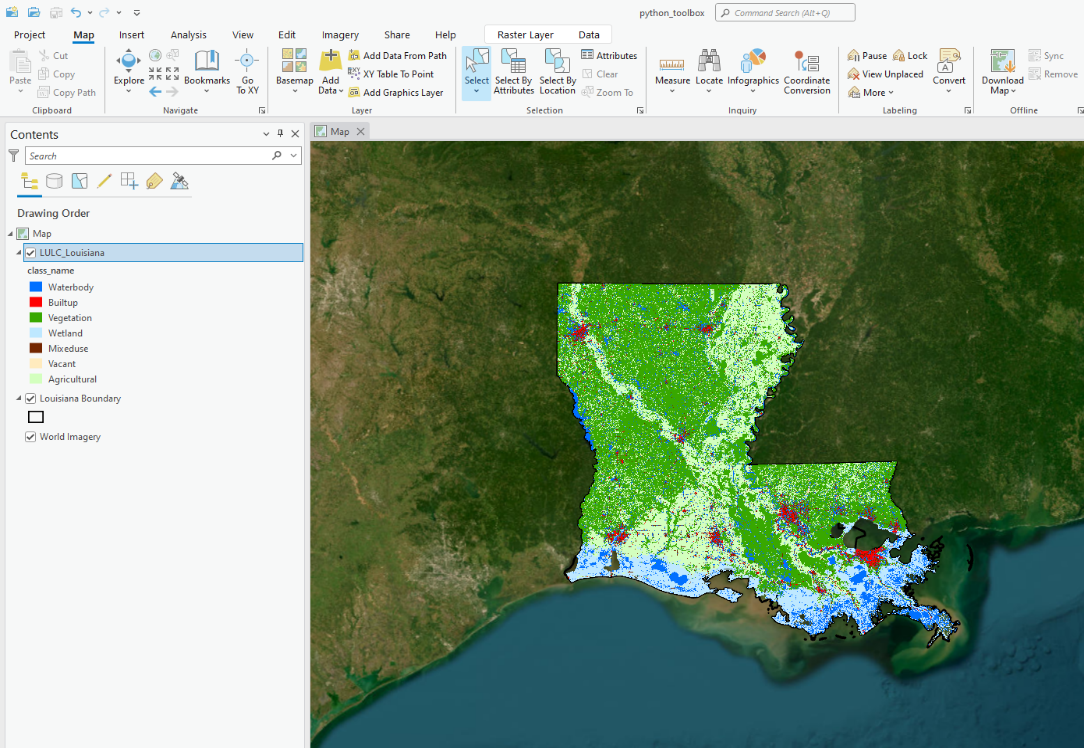
**3.1 Adding the Datasets**

**A screenshot of a computer

AI-generated content may be incorrect.**

**3.2 Map Layer in Arc GIS Pro**

From the add data section, LULC raster data has been added.

****

**3.3 Adding the Toolbox**

In this step the toolbox was added from add toolbox option.

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**3.4 The Toolbox**

The script starts by extracting and validating the user-defined parameters to ensure the correct format and completeness. It converts the Class Values input into a list of integers, checks the consistency of the Number of Classes with the provided class values, and verifies the output directory's existence, creating it if needed.

The LULC raster is reprojected into the specified coordinate system for spatial consistency, a critical step when working with datasets from multiple sources. The raster is then clipped using the boundary shapefile to focus on the area of interest, minimizing unnecessary processing.

The clipped raster is converted to a NumPy array to facilitate efficient pixel-wise analysis. For each LULC class, the script counts the pixels and calculates the corresponding area in square kilometers, with the results exported to a .csv file for easy access and further analysis.

A bar chart is generated to visually represent the area of each LULC class in square kilometers. This provides a clear, intuitive comparison of the land cover distribution. The chart is saved as a .png image, suitable for inclusion in reports or presentations.

The script converts the raster into vector format, turning raster cells into polygons, each associated with a class value (gridcode). It then dissolves the polygons by class, merging adjacent polygons with the same value, and calculating the area for each class. The dissolved polygons are embedded with area attributes, and the resulting shapefile is ready for further analysis.

To complete the workflow, the script automatically adds the dissolved shapefile to the current ArcGIS Pro map. This integration allows for immediate visualization and spatial analysis.

**A screenshot of a computer

AI-generated content may be incorrect.**