Subject: GIS Programming Class Project 2 Report

Name: Parvin Momenian

LSUID: 895384980

Course: 2025 Spring GEOG 4057 for Lei Wang

GIS Programming Class Project 2 Report

1. Project Summary

Problem Statement

This project aims to solve a common geospatial challenge: extracting accurate elevation values for a set of geographic points that represent boundary locations of a flood-affected area. Elevation is a critical variable for modeling flood risk, runoff, and hydrological behavior. The objective is to build an automated solution that reads a CSV of coordinates, extracts elevation values from a high-resolution dataset using Google Earth Engine (GEE), and visualizes the results through a map in ArcGIS Pro.

Data Sources

- boundary.csv: A CSV file containing X, Y coordinates that define the locations of interest. This file was generated from a classified flood raster.
- USGS 3DEP 10m Elevation Dataset: A digital elevation model (DEM) provided through Google Earth Engine. This raster dataset provides elevation data at a 10-meter resolution across the United States.

Methods and Tools

The following tools and libraries were used to implement the solution:
- Python: Main programming language for script development.
- ArcPy: For spatial operations within ArcGIS Pro.
- GeoPandas and Pandas: For reading CSVs and creating GeoDataFrames.
- Earth Engine API: For accessing and sampling the USGS elevation data.
- ArcGIS Pro: For toolbox integration, visualization, and exporting maps.
- Jupyter Notebook: Used for iterative development and testing.

2. Development of GIS Solution

The core solution was developed using Python and integrated into ArcGIS Pro as a script tool. The script performs the following tasks:
1. Reads the CSV using Pandas and converts it into a GeoDataFrame using GeoPandas.
2. Defines a spatial reference (EPSG:4326) and saves the data as a shapefile.
3. Initializes Google Earth Engine and retrieves elevation values for each point.
4. Writes the elevation values back into the shapefile using ArcPy.
5. A custom ArcGIS Toolbox was created so users can run the script with a graphical interface.

3. Project Documentation

Introduction

Elevation data is foundational in geographic information systems. It plays a significant role in environmental modeling, hydrology, and land-use planning. This project demonstrates how to extract elevation values for flood boundary points by leveraging Google Earth Engine and integrating the results into ArcGIS Pro for visualization and analysis.

Methodology

The CSV file containing X and Y coordinates was first read using Pandas. The points were then converted to a shapefile using GeoPandas and projected to EPSG:4326. Each point was passed to the Earth Engine API, which sampled the USGS 3DEP DEM and returned elevation values. These values were added as a new field in the shapefile. The final output was symbolized using graduated color ramps and exported to a PDF map layout.

Code Documentation

The code includes the following logical steps:
- Input: Read CSV → `pandas.read_csv()`
- Convert to shapefile → `geopandas.GeoDataFrame()` with geometry from X/Y.
- Projection: Set CRS to EPSG:4326.
- Earth Engine access → `ee.Initialize()`, image → `ee.Image("USGS/3DEP/10m")`
- Sampling: Use `sampleRegions()` to extract elevation per point.
- Output: Update shapefile using `arcpy.da.UpdateCursor()` and save.

Results and Visualization

The output shapefile contained 338 points with elevation data successfully added. In ArcGIS Pro, the data was symbolized using 5 classes with natural breaks. Thematic mapping revealed small but significant variations in elevation across the boundary. A layout was created with title, legend, north arrow, and scale bar. The final map was exported to PDF for presentation.

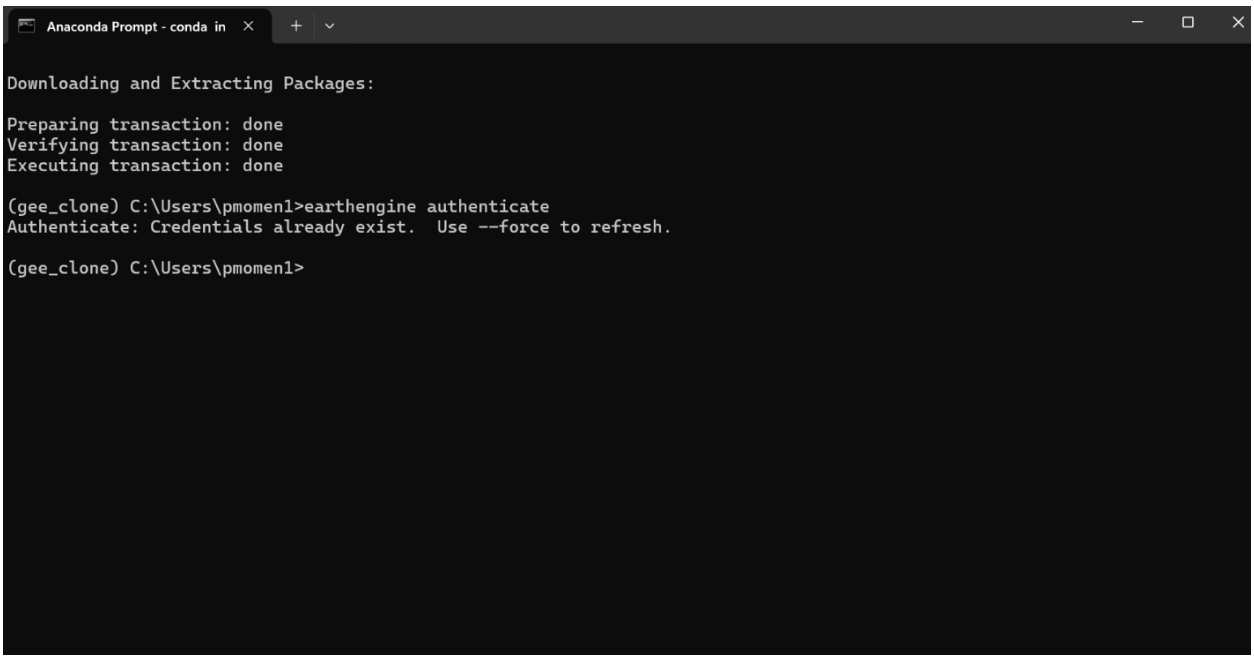
The steps of Project 2

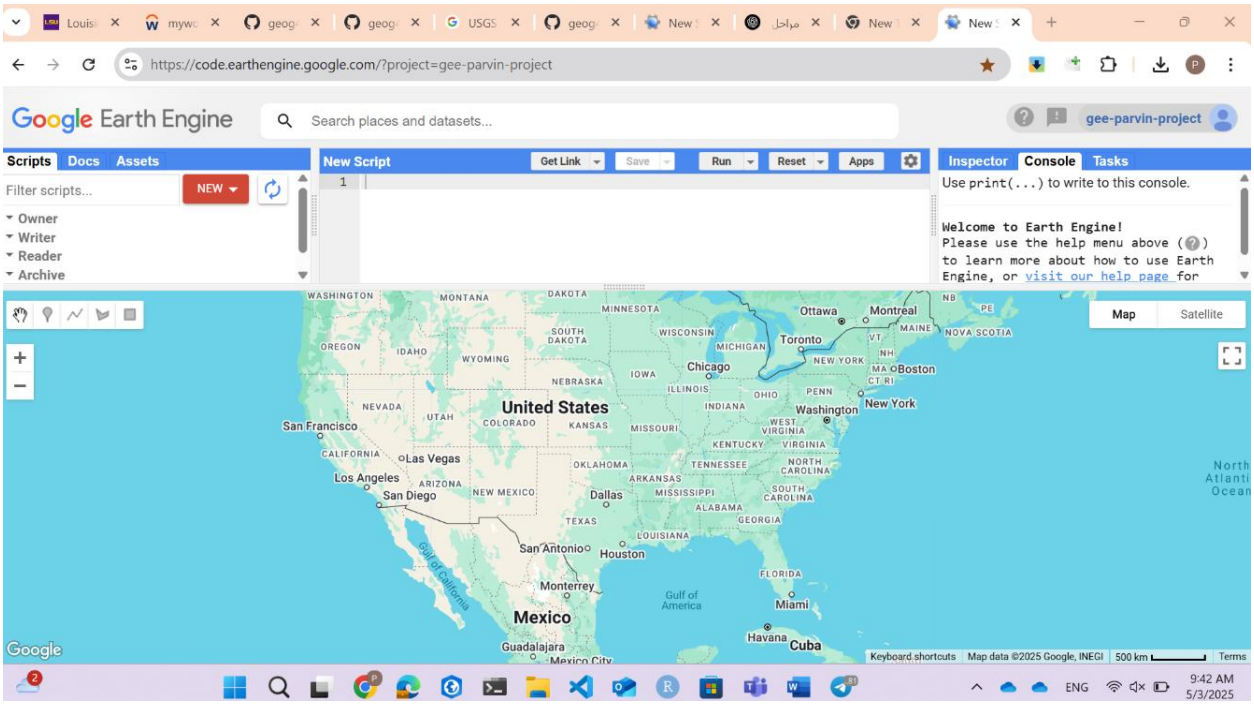
Step 1 – Environment Setup and Authentication

In this step, we set up the Python environment and authenticated access to Google Earth Engine.
A Conda environment named gee_clone (cloned from the ArcGIS Pro environment) was

```
# In Anaconda Prompt

conda activate "C:\Users\pmomen1\AppData\Local\ESRI\conda\envs\gee_clone"

conda install -c conda-forge earthengine-api

earthengine authenticate
```

activated. Required packages such as earthengine-api, geopandas, pandas, and shapely were installed. Earth Engine authentication was performed via terminal and then initialized in the notebook using the project ID.






```
# In Jupyter Notebook

import ee

ee.Initialize(project='gee-parvin-project')
```

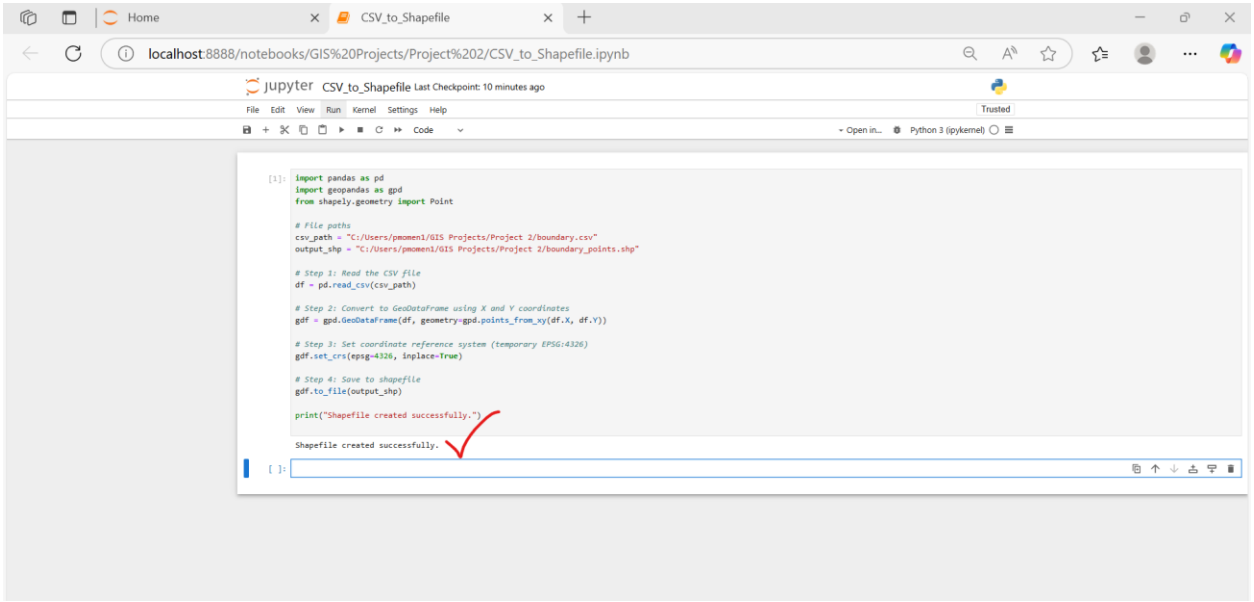
Step 2 – Converting CSV to Shapefile

In this step, we converted the point coordinates stored in the `boundary.csv` file into a shapefile using Python.

First, we read the CSV file with `pandas` and created a `GeoDataFrame` using the X and Y columns as point geometries. Then, we assigned a temporary coordinate reference system (CRS) of EPSG:4326. The shapefile was saved in the project directory as `boundary_points.shp`.

This shapefile will later be used to extract elevation values for each point from Google Earth Engine.

```
import pandas as pd

import geopandas as gpd

from shapely.geometry import Point

csv_path = "C:/Users/pmomen1/GIS Projects/Project 2/boundary.csv"

output_shp = "C:/Users/pmomen1/GIS Projects/Project 2/boundary_points.shp"

df = pd.read_csv(csv_path)

gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df.X, df.Y))

gdf.set_crs(epsg=4326, inplace=True)

gdf.to_file(output_shp)

print("Shapefile created successfully.")
```



Step 3 – Setting CRS Based on TIFF Metadata

In this step, we determined the correct coordinate reference system (CRS) for the shapefile by inspecting the GeoTIFF raster flood_2class.tif. The CRS was extracted using ArcPy, and it was found to be EPSG:32119 (NAD_1983_StatePlane_North_Carolina_FIPS_3200).

```
import arcpy

tif_path = r"C:\Users\pmomen1\GIS Projects\Project 2\flood_2class.tif"

desc = arcpy.Describe(tif_path)

crs = desc.spatialReference

print("CRS Name:", crs.name)

print("Factory Code (EPSG):", crs.factoryCode)
```

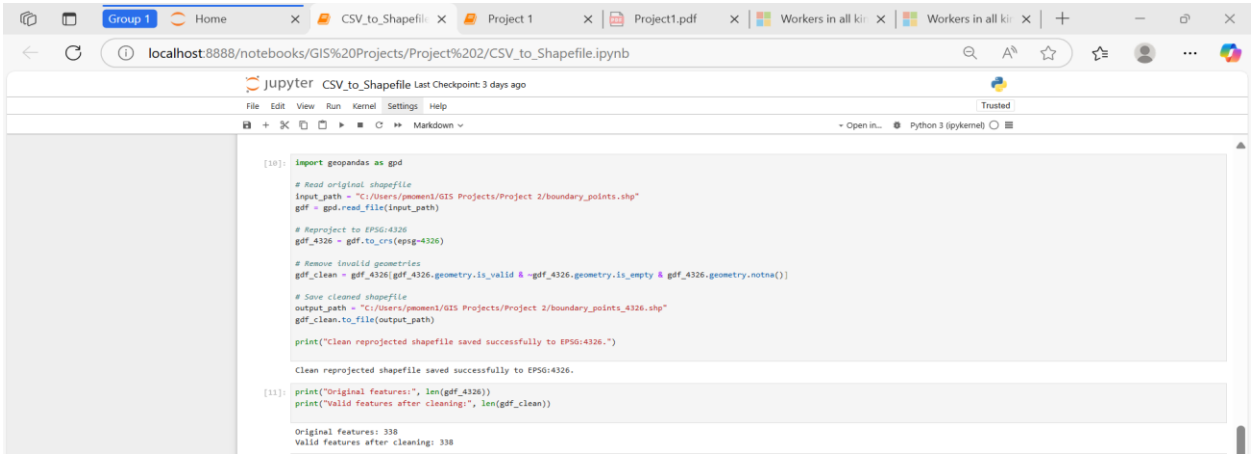
This EPSG code was later used when constructing the shapefile from the CSV.

Step 4 – Reproject Shapefile to EPSG:4326

Google Earth Engine requires coordinates in WGS 84 (EPSG:4326), so we reprojected the shapefile.
After reprojection, we checked the validity of geometries. All 338 points remained valid.

```
import geopandas as gpd

# Read original shapefile

input_path = "C:/Users/pmomen1/GIS Projects/Project 2/boundary_points.shp"

gdf = gpd.read_file(input_path)

# Reproject to EPSG:4326

gdf_4326 = gdf.to_crs(epsg=4326)

# Remove invalid geometries (none were found)

gdf_clean = gdf_4326[gdf_4326.geometry.is_valid & ~gdf_4326.geometry.is_empty & gdf_4326.geometry.notna()]

# Save cleaned shapefile

output_path = "C:/Users/pmomen1/GIS Projects/Project 2/boundary_points_4326.shp"

gdf_clean.to_file(output_path)

print("Clean reprojected shapefile saved successfully.")
```

```
print("Original features:", len(gdf_4326))

print("Valid features after cleaning:", len(gdf_clean))
```

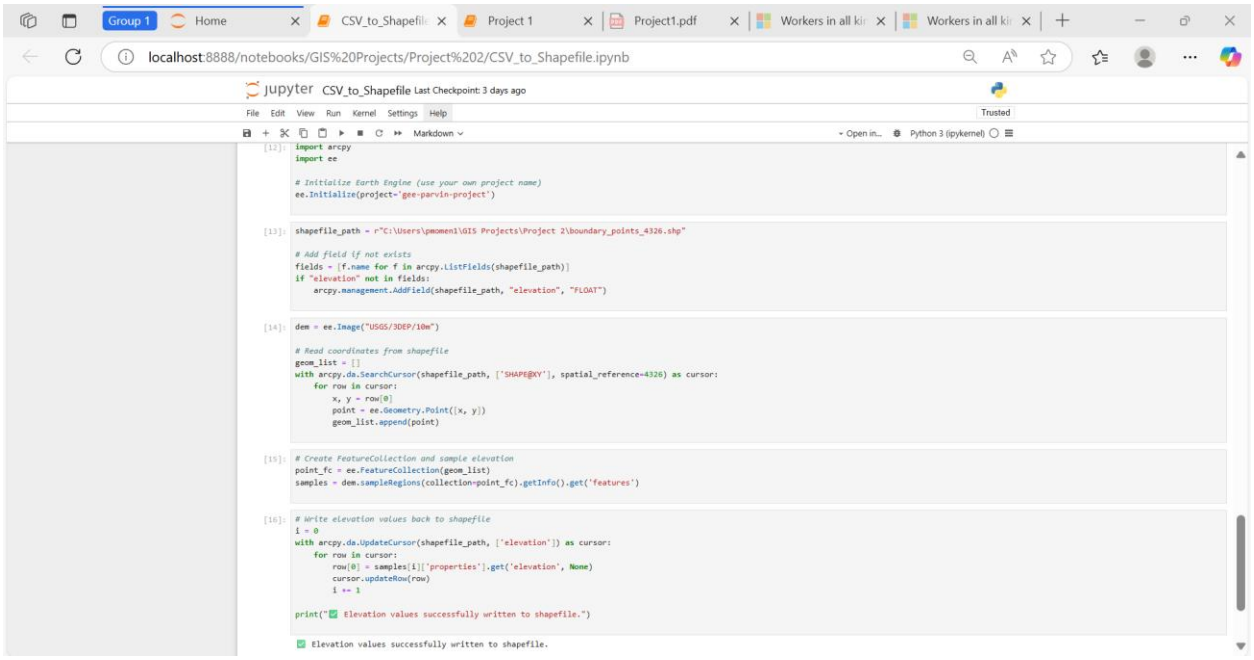


Step 5 – Extracting Elevation from Google Earth Engine

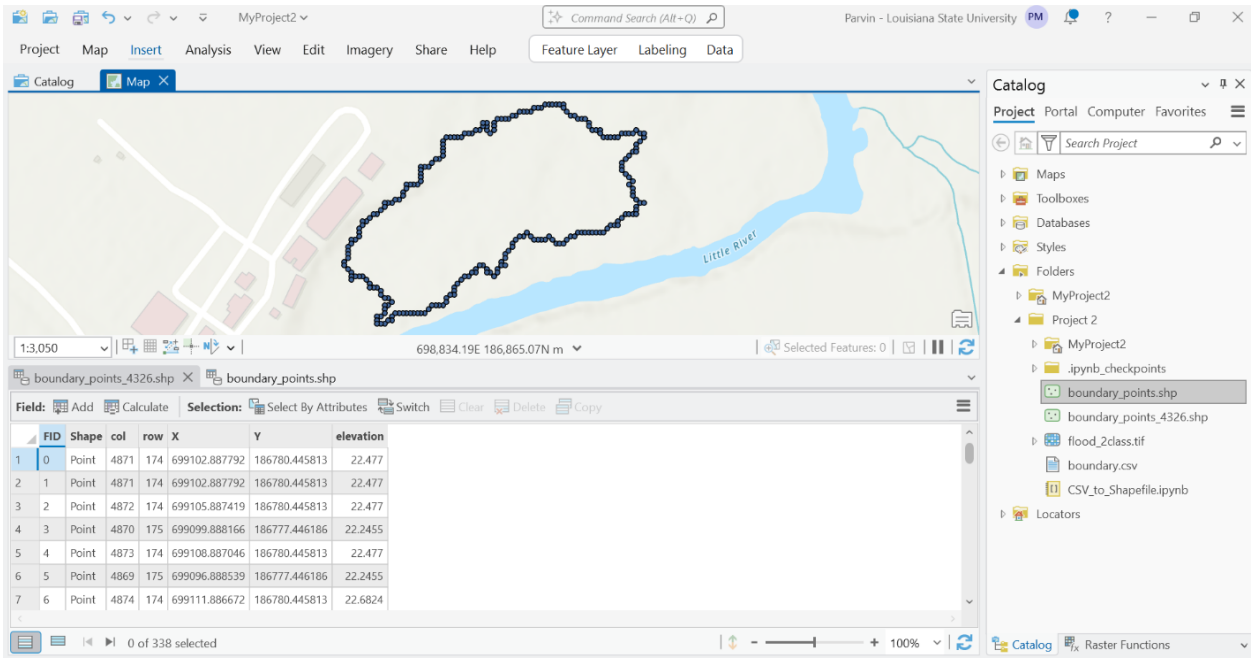
In this step, we extracted elevation values from the USGS 3DEP 10-meter Digital Elevation Model using Google Earth Engine for each point in the reprojected shapefile.
The shapefile boundary_points_4326.shp, containing 338 valid point features in WGS 84 (EPSG:4326), was used as input. A new field named elevation was added to store the elevation values.

Using arcpy, we read the coordinates from the shapefile, sent them to GEE as ee.Geometry.Point, and used the sampleRegions() function to extract elevation values from the DEM. These values were written back to the shapefile using an UpdateCursor.

```
import arcpy
import ee
# Initialize Earth Engine
ee.Initialize(project='gee-parvin-project')
# Input shapefile path
input_shp = r"C:\Users\pmomen1\GIS Projects\Project 2\boundary_points_4326.shp"
# Add elevation field if not present
fields = [f.name for f in arcpy.ListFields(input_shp)]
if "elevation" not in fields:
    arcpy.management.AddField(input_shp, "elevation", "FLOAT")
# Load DEM from Earth Engine
dem = ee.Image("USGS/3DEP/10m")
# Extract coordinates from shapefile
geom_list = []
with arcpy.da.SearchCursor(input_shp, ['SHAPE@XY'], spatial_reference=4326) as cursor:
    for row in cursor:
        x, y = row[0]
        point = ee.Geometry.Point([x, y])
        geom_list.append(point)
# Create FeatureCollection from points
point_fc = ee.FeatureCollection(geom_list)
# Sample elevation values
samples = dem.sampleRegions(collection=point_fc).getInfo().get('features')
# Write elevation values back to shapefile
i = 0
with arcpy.da.UpdateCursor(input_shp, ['elevation']) as cursor:
    for row in cursor:
        row[0] = samples[i]['properties'].get('elevation', None)
        cursor.updateRow(row)
        i += 1
print(" Elevation values successfully written to shapefile.")
```



The shapefile now contains elevation data for all 338 points and is ready to be visualized in ArcGIS Pro.



Step 6 – Interface Design for a Toolbox

In this final step, we created a custom geoprocessing tool inside ArcGIS Pro using a Python toolbox (.atbx) that allows users to extract elevation values from a CSV file using Google Earth Engine.

Tool Function

The tool named "Extract Elevation from CSV" takes a CSV file containing X and Y coordinates, creates a shapefile, and then fills in the elevation values by querying the USGS 3DEP dataset in GEE. It requires two inputs:

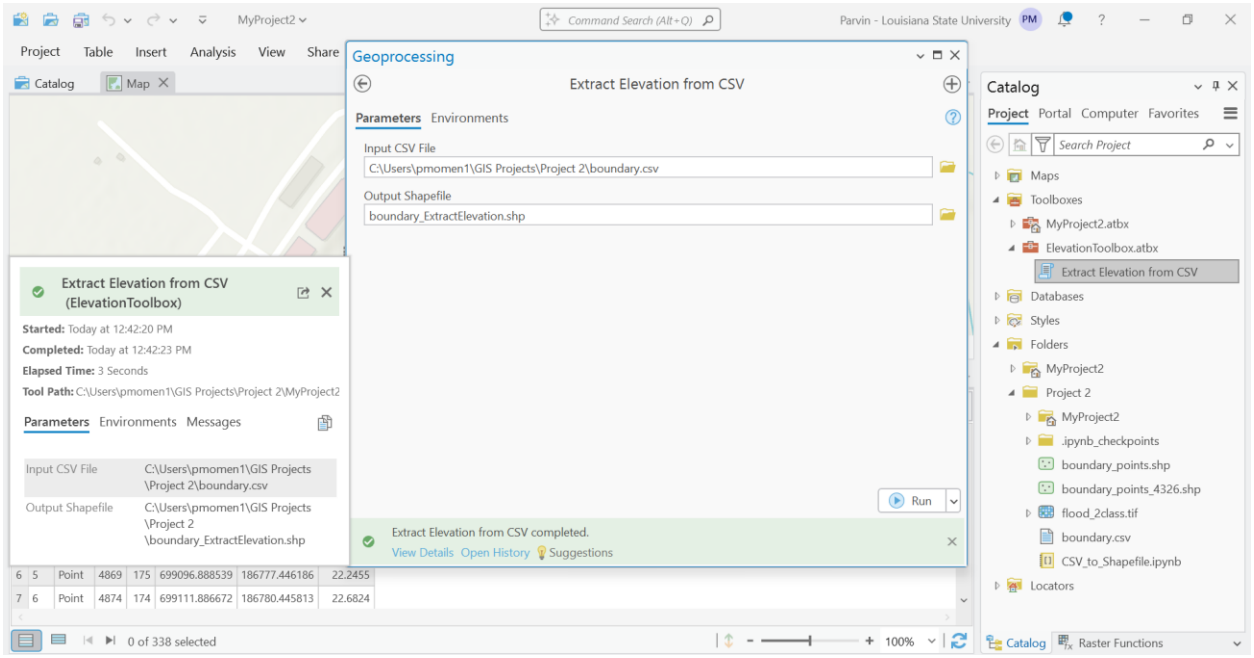
- Input CSV File – the file containing point coordinates
- Output Shapefile – the path to save the resulting shapefile with elevation values

Implementation Steps:

1. Python Toolbox (.atbx) was created and added to the ArcGIS Pro project.
2. A new tool named Extract Elevation from CSV was implemented using Python code.
3. The tool was run from the Geoprocessing pane using graphical input.
4. It successfully generated the shapefile boundary_ExtractElevation.shp, containing:
    - 338 valid point features
    - An added field elevation with values from GEE

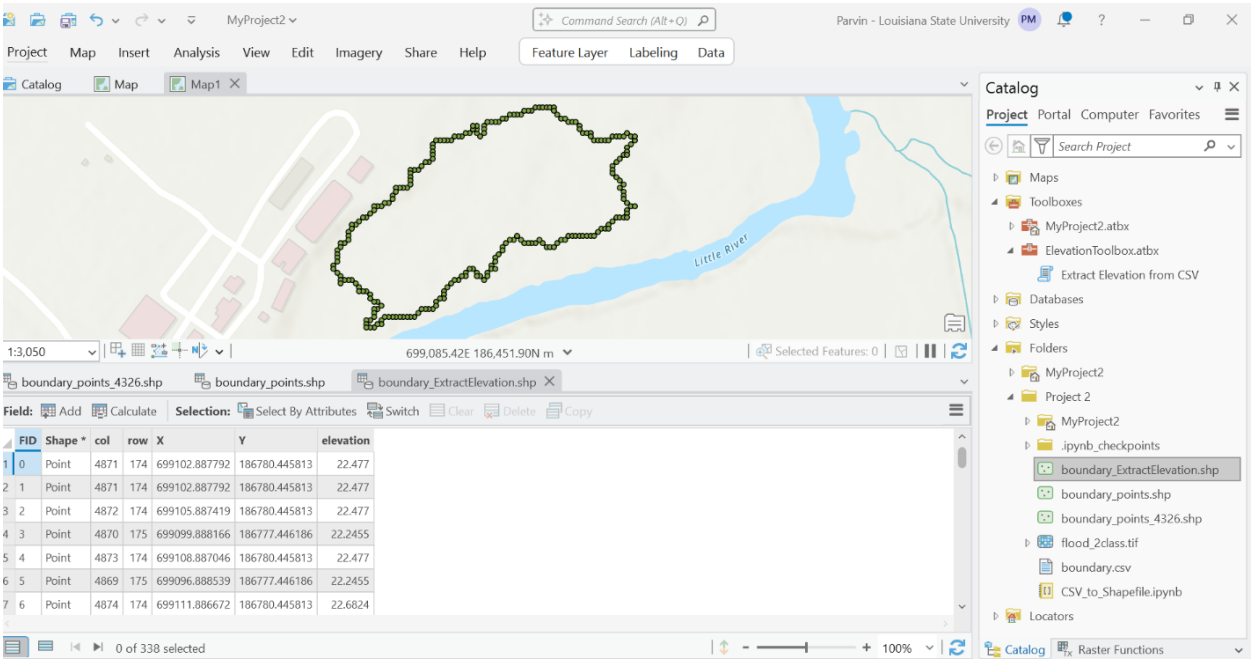
```
import arcpy
import ee
import os
import pandas as pd
import geopandas as gpd
from shapely.geometry import Point
class ExtractElevation:
    def __init__(self):
        self.label = "Extract Elevation from CSV"
        self.description = "This tool extracts elevation values from Google Earth Engine for points in a CSV file and saves them to a shapefile."
    def getParameterInfo(self):
        params = [
            arcpy.Parameter(
                displayName="Input CSV File",
                name="input_csv",
                datatype="DEFile",
                parameterType="Required",
                direction="Input"
            ),
            arcpy.Parameter(
                displayName="Output Shapefile",
                name="output_shp",
                datatype="DEFeatureClass",
                parameterType="Required",
                direction="Output"
            )
        ]
        return params
    def execute(self, parameters, messages):
        input_csv = parameters[0].valueAsText
        output_shp = parameters[1].valueAsText
```

```
# Step 1: Read CSV
df = pd.read_csv(input_csv)
gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df['X'], df['Y']))
gdf.set_crs(epsg=4326, inplace=True)
# Save temp shapefile
gdf.to_file(output_shp)
# Initialize GEE
ee.Initialize(project='gee-parvin-project')
dem = ee.Image("USGS/3DEP/10m")
# Add elevation field
if "elevation" not in [f.name for f in arcpy.ListFields(output_shp)]:
    arcpy.management.AddField(output_shp, "elevation", "FLOAT")
# Read geometry for each point
geom_list = []
with arcpy.da.SearchCursor(output_shp, ['SHAPE@XY'], spatial_reference=4326) as cursor:
    for row in cursor:
        x, y = row[0]
        point = ee.Geometry.Point([x, y])
        geom_list.append(point)
fc = ee.FeatureCollection(geom_list)
elevations = dem.sampleRegions(fc).getInfo().get("features")
# Write back to shapefile
i = 0
with arcpy.da.UpdateCursor(output_shp, ['elevation']) as cursor:
    for row in cursor:
        row[0] = elevations[i]['properties'].get('elevation', None)
        cursor.updateRow(row)
        i += 1
arcpy.AddMessage("Elevation values written successfully.")
```



Results

The tool ran successfully and produced a shapefile visualized in ArcGIS Pro with all elevation values correctly extracted and displayed. The tool can now be reused for similar tasks by providing different CSV files as input.

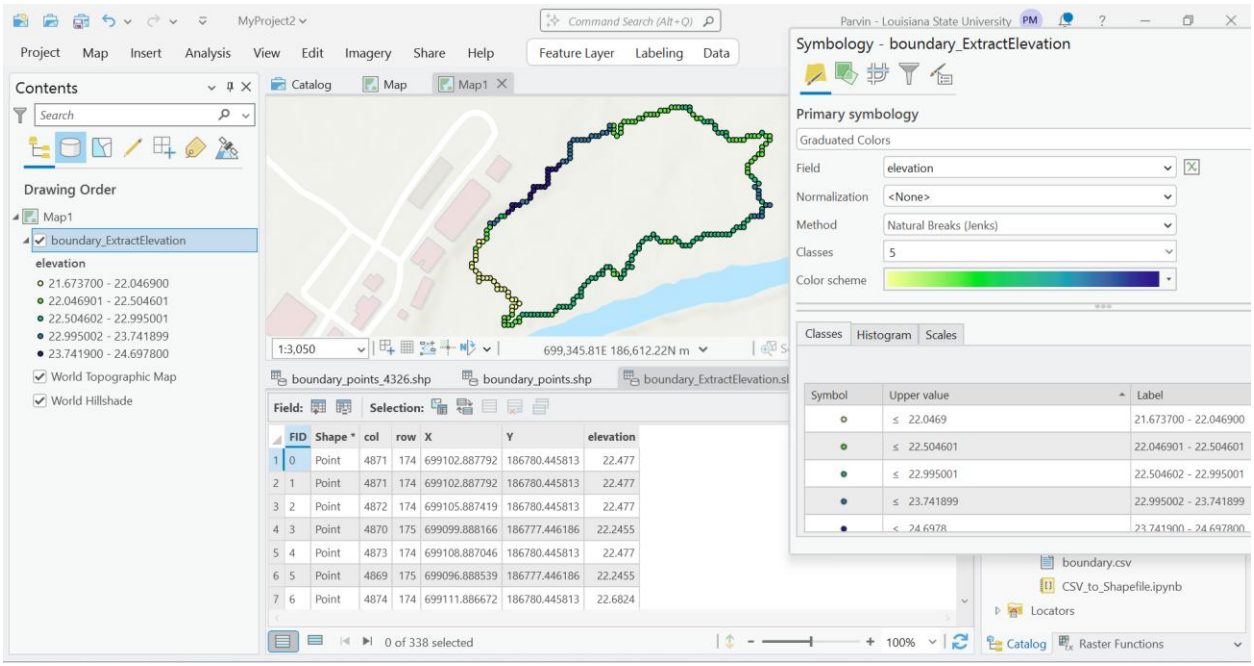


Step 6 – Data Visualization: Change Symbology

In this step, we applied symbology to our elevation shapefile to visualize the spatial variation of elevation values.

- We right-clicked on the boundary_ExtractElevation.shp layer in the **Contents** panel.
- Then we opened the **Symbology** pane and chose **Graduated Colors**.
- The field used for classification was **elevation**.
- We selected **Natural Breaks (Jenks)** as the classification method and set the number of classes to **5**.
- We applied a green-to-blue color ramp to visualize the elevation gradient across the boundary points.

This visualization helps highlight how elevation changes across the points along the boundary.

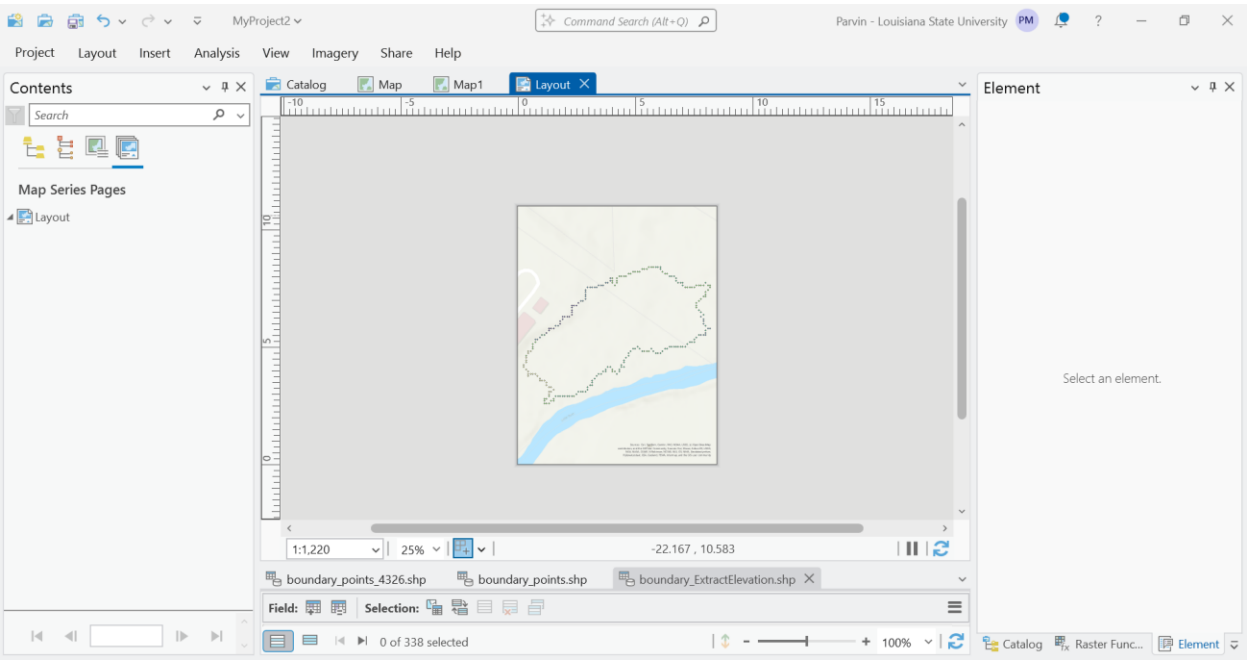


Adding Map Frame to Layout

In this step, we created a printable layout for visualizing elevation values.

First, we opened the layout view and inserted a new **Map Frame** containing the styled map layer boundary_ExtractElevation.shp. The symbology was preserved from the map view, and the map frame was adjusted to fit the layout area.

This layout will serve as the base for further additions like legend, scale bar, north arrow, and title before exporting the final map.



Adding Title to the Map Layout

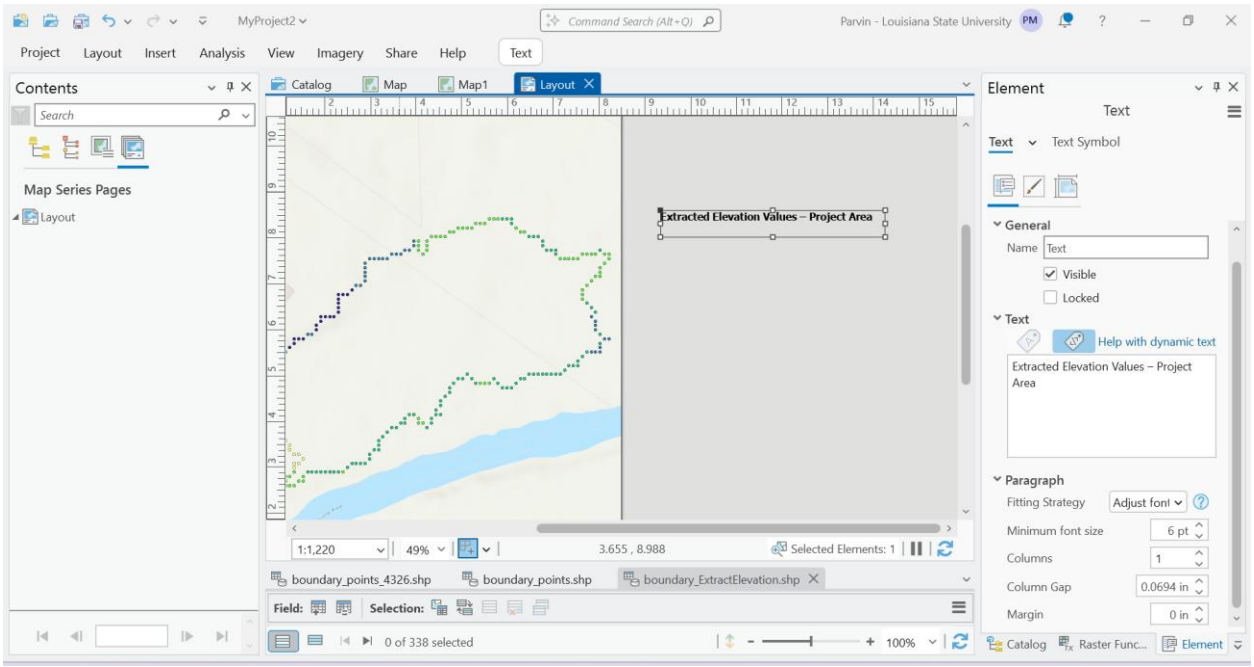
A map title was added to the layout using the *Insert* > *Rectangle Text* tool. The title "Extracted Elevation Values – Project Area" was placed above the map frame to describe the content. The font style and size were adjusted for better visibility.

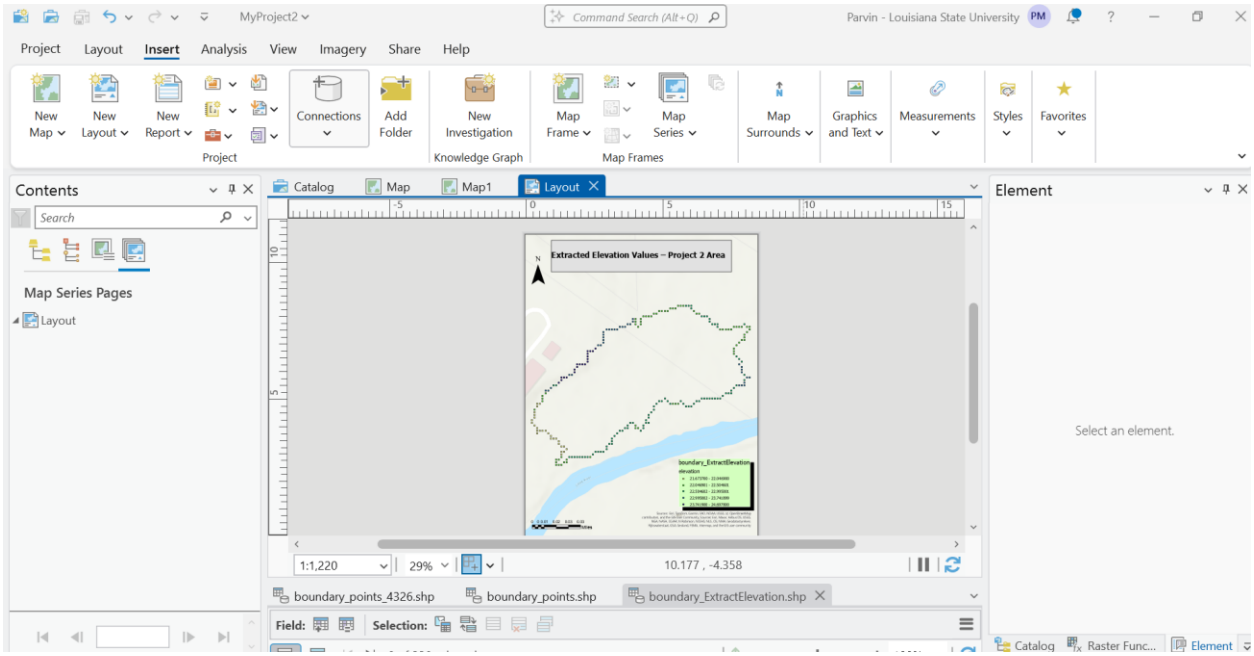
Step 6: Data Visualization and Layout Export

Objective:
Visualize the elevation shapefile and create a printable map layout.

Actions Completed:

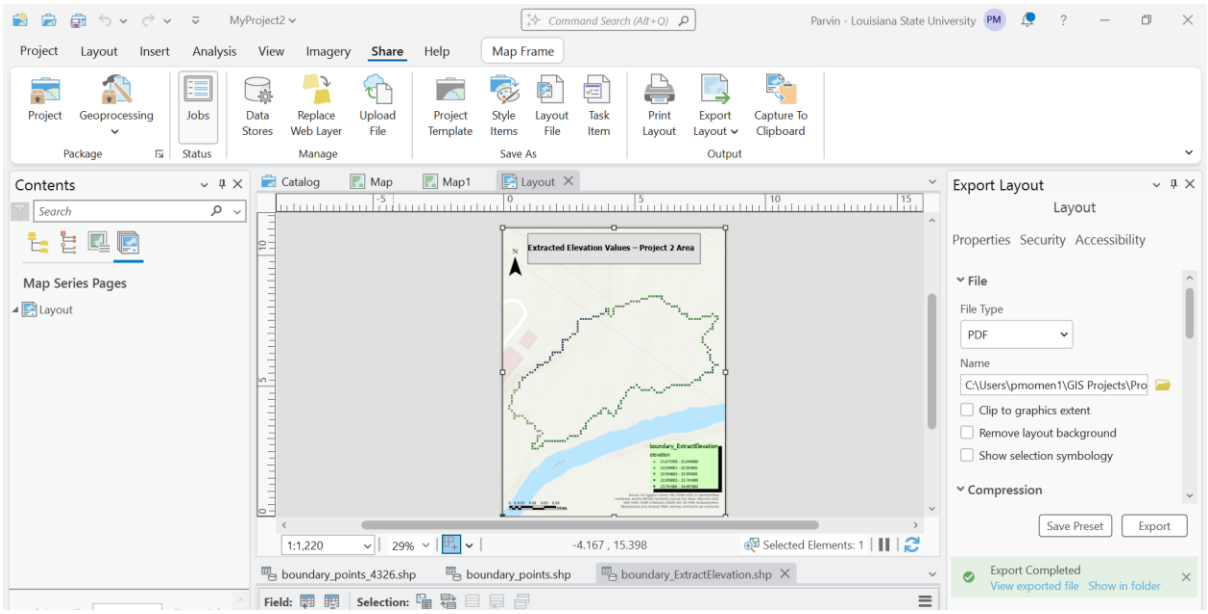
1. **New Map Layout:**
   A new layout was added to the existing ArcGIS Pro project.
2. **Add Map Frame:**
   The map containing boundary_ExtractElevation.shp was inserted into the layout using the *Map Frame* option.
3. **Symbology Settings:**
   The elevation data was symbolized using **graduated colors** with Natural Breaks (Jenks) classification into 5 classes, to show spatial variation in elevation values.
4. **Add Elements:**
   - **Title**: "Extracted Elevation Values – Project 2 Area"
   - **Legend**: A legend showing the elevation class ranges.
   - **North Arrow**: Added to the top-left corner.
   - **Scale Bar**: Positioned at the bottom of the layout.
   - **Credits** and **Basemap info** (auto-included at the bottom right).

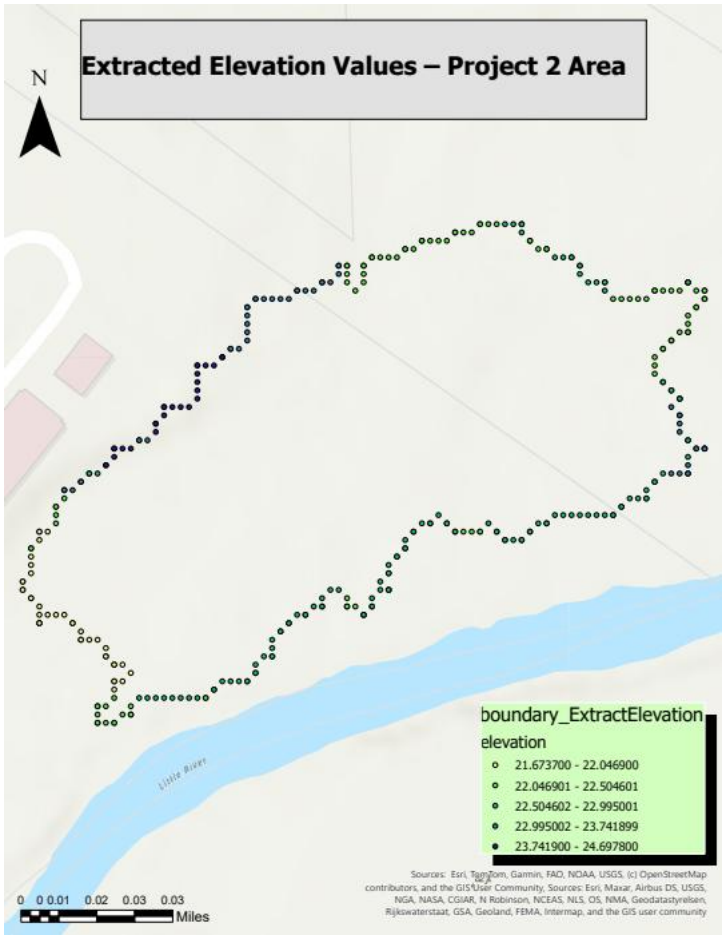




5. **Export Layout:**

   - Exported as a **Vector PDF** for high-resolution output.





Academic Integrity Note

This project report and code were developed with the assistance of OpenAI ChatGPT for code review.