**Wildfire Vulnerability Analysis Training Manual**

**Prepared For:** Paul Keane  
**Project Context:** Wildfire risk identification and analysis using geospatial and statistical techniques in Python and ArcGIS Pro

**📘 Overview**

This manual documents the end-to-end process for preparing, executing, debugging, and exporting a geospatial wildfire analysis Python notebook that you developed and ran both locally and on an AWS EC2 virtual machine. It includes your technical challenges, how they were resolved, and recommendations for future workflows.

**🔧 Environment Setup and File Management**

**✔️ Local Setup:**

* Installed Anaconda Navigator.
* Opened .ipynb notebook directly **from the folder containing shapefiles and data**.
* Verified that Anaconda Navigator or Jupyter Notebook **does not need to be opened first**. Simply right-click and choose *Open With → Jupyter Notebook* to preserve paths.

**⚠️ Initial Mistake:**

* Opening the notebook from a different directory (or through Anaconda Navigator first) **broke relative file paths** like California\_Fire\_Perimeters\_2017.shp, causing file-not-found errors.

**🗂️ Recommendation:**

* Always unzip and run your notebook from the same directory where all shapefiles, rasters, and CSVs are located.

**🧪 Path Fixes and Dataset Integration**

**📂 Data Files Required:**

* California\_Fire\_Perimeters\_2017.shp
* firestations.shp
* DSM, DTM, Canopy Height rasters
* StoryMap and prior 2020 analysis references (for reuse and comparison)

**✅ Fixes Made:**

* Updated gpd.read\_file() and raster paths to relative paths (e.g., gpd.read\_file('California\_Fire\_Perimeters\_2017.shp') instead of an absolute path).

**🧮 Geometry, CRS & Projection Challenges**

**🚫 CRS Errors:**

* Encountered ValueError while setting CRS that already existed.
* Some geometries were invalid (fire\_centroids.is\_valid.value\_counts() → False entries).

**✅ Fixes:**

* Used GeoSeries.set\_crs(..., allow\_override=True) only when necessary.
* Ran fire\_centroids = fire\_centroids[fire\_centroids.is\_valid] to clean invalid geometries.
* Reprojected everything to **UTM Zone 11N (EPSG:32611)** before calculating distances.

fire\_centroids\_proj = fire\_centroids.to\_crs('EPSG:32611')

fire\_stations\_proj = fire\_stations.to\_crs('EPSG:32611')

**📏 Distance Analysis**

**Original Error:**

* Incorrect use of \* 111 for calculating distances in kilometers from lat/lon (WGS84) without projection.
* Resulted in impossible values like 1.797693e+305 (overflow).

**✅ Fix:**

* Corrected method using UTM projection and .distance() in meters:

fire\_centroids\_proj['nearest\_dist\_km'] = fire\_centroids\_proj.geometry.apply(

lambda pt: fire\_stations\_proj.distance(pt).min() / 1000 # meters to km

)

**📊 Scatterplot & Correlation Plotting Issues**

**Initial Problems:**

* Scatterplot failed to render due to SystemError: tile cannot extend outside image (PIL + matplotlib bug).

**Fixes:**

* Ensured that no labels or outliers extend beyond figure dimensions.
* Added padding to .text() placement.

for \_, row in highlight\_fires.iterrows():

ax.text(row['nearest\_dist\_km'] + 0.2, row['area\_km2'], row['FIRE\_NAME'], fontsize=9)

**Pearson Correlation Result:**

* After cleaning, a valid Pearson r and p-value were computed and visualized.

**💾 Saving & Exporting Results**

**Save Strategy:**

* Use **File → Save and Checkpoint** to preserve notebook and results.
* Optionally export as PDF:
  + Jupyter: File → Export Notebook As → PDF.
  + Ensure PDF printer is installed.

**Uploading to GitHub:**

* Push both .ipynb and exported .pdf.
* If using large data files, push notebook only and link to zipped data storage.

**📈 Final Interpretation Summary**

* 🔥 Fires with larger area tend to be farther from fire stations.
* 📉 Correlation may not be strong but highlights gaps in infrastructure.
* 🎯 South-facing slopes, >40% slope, and dense vegetation (via LiDAR canopy rasters) identify highest-risk wildfire zones.
* 🛰️ Use of LiDAR (DSM - DEM) enables fine-grained site vulnerability classification.

**📋 Recommendations for Future Use**

1. **Always keep your notebook in the same directory as data**.
2. Validate geometry and CRS immediately after reading shapefiles.
3. Reproject before performing spatial operations.
4. Export to PDF after verifying all outputs visually.
5. Push to GitHub along with README.md explaining how to unzip and run.

**✅ Status Summary**

|  |  |
| --- | --- |
| **Task** | **Completed?** |
| Local run with correct paths | ✅ |
| EC2-compatible ZIP package | ✅ |
| Geometry and CRS fixes | ✅ |
| Nearest station distance analysis | ✅ |
| Correlation and labeling | ✅ |
| Manual export options | ✅ |
| Personalized training manual | ✅ |

**Wildfire Vulnerability Analysis Training Manual**

**Prepared For:** Paul Keane  
**Project Context:** Wildfire risk identification and analysis using geospatial and statistical techniques in Python and ArcGIS Pro

**📘 Overview**

This manual documents the end-to-end process for preparing, executing, debugging, and exporting a geospatial wildfire analysis Python notebook that you developed and ran both locally and on an AWS EC2 virtual machine. It includes your technical challenges, how they were resolved, and recommendations for future workflows.

**🔧 Environment Setup and File Management**

**✔️ Local Setup:**

* Installed Anaconda Navigator.
* Opened .ipynb notebook directly **from the folder containing shapefiles and data**.
* Verified that Anaconda Navigator or Jupyter Notebook **does not need to be opened first**. Simply right-click and choose *Open With → Jupyter Notebook* to preserve paths.

**⚠️ Initial Mistake:**

* Opening the notebook from a different directory (or through Anaconda Navigator first) **broke relative file paths** like California\_Fire\_Perimeters\_2017.shp, causing file-not-found errors.

**🗂️ Recommendation:**

* Always unzip and run your notebook from the same directory where all shapefiles, rasters, and CSVs are located.

**✅ AWS EC2-Specific Setup:**

* Made sure that the extracted ZIP project folder was fully self-contained on the remote desktop.
* Ensured Anaconda and Jupyter Notebook were launched using a path consistent with the project folder.
* Verified that all shapefiles and raster layers were accessible by Python from within EC2 without broken paths.

**🧪 Path Fixes and Dataset Integration**

**📂 Data Files Required:**

* California\_Fire\_Perimeters\_2017.shp
* firestations.shp
* LiDAR-derived rasters: DSM, DEM, Canopy Height (nDSM)
* 2020 StoryMap and prior analysis references

**✅ Fixes Made:**

* Updated gpd.read\_file() and raster paths to relative paths.
* Moved the notebook into the folder containing the data — the key fix that prevented FileNotFound errors.

**🧮 Geometry, CRS & Projection Challenges**

**🚫 CRS Errors:**

* Encountered ValueError while setting CRS that already existed.
* Some geometries were invalid (fire\_centroids.is\_valid.value\_counts() → False entries).

**✅ Fixes:**

* Used GeoSeries.set\_crs(..., allow\_override=True) only when necessary.
* Ran fire\_centroids = fire\_centroids[fire\_centroids.is\_valid] to clean invalid geometries.
* Reprojected everything to **UTM Zone 11N (EPSG:32611)** before calculating distances.

fire\_centroids\_proj = fire\_centroids.to\_crs('EPSG:32611')

fire\_stations\_proj = fire\_stations.to\_crs('EPSG:32611')

**📏 Distance Analysis**

**Original Error:**

* Incorrect use of \* 111 for calculating distances in kilometers from lat/lon (WGS84) without projection.
* Resulted in impossible values like 1.797693e+305 (float overflow).

**✅ Fix:**

* Corrected method using UTM projection and .distance() in meters:

fire\_centroids\_proj['nearest\_dist\_km'] = fire\_centroids\_proj.geometry.apply(

lambda pt: fire\_stations\_proj.distance(pt).min() / 1000

)

* Verified correct distance range (~0–180 km), resolved NaN values with cleaned geometries and valid projections.

**📊 Scatterplot & Correlation Plotting Issues**

**Initial Problems:**

* Scatterplot failed to render due to PIL/matplotlib issue: SystemError: tile cannot extend outside image.

**Fixes:**

* Added margin padding to avoid rendering overflow.
* Cleaned data before Pearson correlation.
* Retained labels with adjusted offset:

for \_, row in highlight\_fires.iterrows():

ax.text(row['nearest\_dist\_km'] + 0.2, row['area\_km2'], row['FIRE\_NAME'], fontsize=9)

**Pearson Correlation Result:**

* Successfully calculated Pearson r and p-values between fire area and distance to nearest station.
* Plotted scatter with sns.scatterplot() colored by YEAR\_.

**💾 Saving & Exporting Results**

**Save Strategy:**

* Used **File → Save and Checkpoint** to preserve notebook and outputs.
* Exported notebook as PDF via Jupyter: File → Export Notebook As → PDF.
* Alternative: used browser print to PDF (better format on some systems).

**GitHub Push Instructions:**

1. Export notebook as .pdf and save .ipynb.
2. Push both to GitHub.
3. Add README.md explaining structure, how to unzip, and run.

**📈 Final Interpretation Summary**

* 🔥 Fires with larger area tend to occur farther from fire stations — but the correlation is weak.
* 📊 Nearest station analysis shows potential fire station service gaps.
* 🌲 South/Southwest-facing slopes, steep slopes (>40%), and high canopy vegetation identify high-risk WUI zones.
* 🛰️ LiDAR DSM–DEM subtraction provided Canopy Height model, enhancing the terrain-based risk mapping.

**🧭 Full Challenge-to-Solution Timeline**

|  |  |
| --- | --- |
| **Challenge** | **Fix/Resolution** |
| Notebook paths broke | Launched notebook from correct folder |
| Shapefiles not loading | Corrected file paths |
| Invalid geometries | Filtered invalid with .is\_valid |
| CRS errors | Used set\_crs(..., allow\_override=True) |
| Distance calculation overflow | Reprojected to UTM and used .distance() properly |
| PIL rendering crash | Adjusted matplotlib text labels |
| Pearson r was NaN | Cleaned distance data and recalculated |
| PDF text cut off | Used browser Print > PDF instead |

**📋 Recommendations for Future Use**

1. **Always run notebooks from the correct folder.**
2. Validate geometries immediately after read\_file().
3. Confirm CRS and projection before any spatial math.
4. Run full notebook before exporting to PDF.
5. Archive zipped version of all files used.
6. Push .ipynb, .pdf, and a helpful README.md to GitHub.

**✅ Status Summary**

|  |  |
| --- | --- |
| Task | Completed? |
| Local run with correct paths | ✅ |
| EC2-compatible ZIP package | ✅ |
| Geometry and CRS fixes | ✅ |
| Nearest station distance analysis | ✅ |
| Correlation and labeling | ✅ |
| Manual export options | ✅ |
| Personalized training manual | ✅ |
| Full retrospective challenge analysis | ✅ |

**Supplementary Wildfire Analysis Training Manual (Extended Challenges & Learnings)**

**Purpose:** This document expands on the original training manual by capturing additional key challenges, workaround strategies, troubleshooting steps, and learning experiences that occurred throughout your wildfire analysis journey.

**🔁 File Handling & Notebook Launch Challenges**

**❌ Problem:**

Opening the .ipynb file via Anaconda Navigator or from a directory other than the data folder resulted in path errors (e.g., missing shapefiles).

**✅ Fix:**

You learned that Jupyter notebooks must be opened **directly from the data folder** (via right-click > Open With > Jupyter Notebook). This maintains relative path references.

**🔄 Learning:**

* Jupyter's working directory defaults to the launch path.
* Opening from an unrelated location results in FileNotFoundError when using gpd.read\_file() or raster functions.

**🧠 Tip:**

Always verify with os.getcwd() or check !ls in the notebook to confirm file visibility.

**🧭 Running Cells: Run All vs. Sequential Execution**

**⚠️ Misstep:**

After running Restart & Run All, downstream cells failed because earlier errors blocked valid variable definitions.

**🧪 Adjustment:**

Instead of re-running everything after an error, you manually executed each cell **sequentially** post-fix.

**📌 Key Insight:**

* Errors high in the cell order block outputs later on.
* If restarting the kernel, rerun everything only if all errors have been corrected.

**📎 File Extension Issues & Default App Behavior**

**Confusion:**

Windows did not recognize .ipynb files with Jupyter Notebook.

**✅ Resolution:**

You selected jupyter-notebook.exe from Anaconda Scripts, and manually checked **"Always use this app"**.

**📌 Best Practice:**

Once associated, notebooks will open consistently with correct environment context.

**🧼 Distance Errors & Geometry Fixes**

**🔍 Observation:**

Distances showed a value of 1.797693e+305 — a floating-point max value, indicating a calculation problem.

**Root Cause:**

* Lat/lon data was used without projecting to a local CRS.
* Some geometries were **invalid**, causing distance methods to return junk values.

**🔧 Fix Strategy:**

* Use .is\_valid.value\_counts() to identify and filter invalid geometries.
* Reproject all data to EPSG:32611 (UTM Zone 11N).

fire\_centroids = fire\_centroids[fire\_centroids.is\_valid]

fire\_centroids\_proj = fire\_centroids.to\_crs('EPSG:32611')

**🎨 Scatterplot Rendering Failures**

**Error:**

tile cannot extend outside image — when plotting with matplotlib and seaborn.

**🔍 Root Cause:**

* Label placement may have extended beyond figure bounds.
* Internal PIL bug triggered during automatic rendering.

**✅ Fix:**

* Added plt.tight\_layout() or margin adjustments.
* Ensured label placement offsets didn’t go beyond plot boundaries.

**🧪 Statistical Correlation (Pearson r) Returns NaN**

**🔍 Diagnosis:**

Some nearest\_dist\_km values were NaN or invalid floats due to broken distance calculations.

**✅ Fix:**

* Recomputed distances only **after** verifying geometry validity and projection.
* Then recalculated Pearson correlation:

corr, p\_value = pearsonr(

analysis\_df['nearest\_dist\_km'],

analysis\_df['area\_km2']

)

**📦 Manual & Output Management**

**You Asked:**

Should I use Save or Checkpoint to preserve everything?

**✅ Answer:**

* Use **Save and Checkpoint** while working.
* Before exporting or uploading, save outputs manually using File > Save and Export As PDF or Print to PDF in browser for cleaner formatting.

**🔧 Strategy Recap: Countering Common Pitfalls**

|  |  |
| --- | --- |
| **Problem** | **Strategy** |
| Broken paths | Always open notebook from data directory |
| CRS overwrite error | Use allow\_override=True only if necessary |
| NaN distances | Clean geometries + verify CRS before measuring |
| PIL tile error | Avoid overflowing plot bounds with text |
| Notebook won’t open | Re-associate .ipynb with Jupyter Notebook |

**✅ Final Reflection**

This extended training manual now includes:

* A deeper look at **problem-solving mindset** during geospatial Python work.
* Emphasis on **order of operations** — from reading data, cleaning geometry, setting CRS, and calculating.
* Visual and statistical issues were intertwined — **valid geometry** was foundational to every success downstream.

Combined with your original PDF + DOCX manual, this gives you a complete retrospective for replication, improvement, or sharing on GitHub.

Let me know when you're ready for export!

Thanks — this confirms a few important things. Here's what we now know:

**✅ What’s Working:**

1. **Valid Geometry**:  
   fire\_stations\_proj.is\_valid = ✅ All 307 fire station geometries are valid.
2. **Reprojection is Correct**:  
   CRS changed successfully from EPSG:4326 → EPSG:32611 ✔️
3. **Geometry Column Exists**:  
   The geometry field is visible with POINT geometries and valid projected coordinates (e.g., 5.19e+05, 3.69e+06)

**❗What’s Still Failing:**

Despite the geometries being valid and projected, your .distance() calculation still returned 1.797693e+305 — a sign it’s silently failing.

That can happen when:

* 🔍 **The input geometries are not compatible spatially** — i.e., your **fire centroids and fire stations** are actually **very far apart** or **mismatched in region**.
* 🔒 The .distance() operation fails silently if a **geometry is empty** or .min() receives no valid value.

**✅ Next Diagnostic Step:**

Let’s check **your fire centroids** now — just like we did for fire stations.

**📌 New Diagnostic Cell:**

python

CopyEdit

fire\_centroids\_proj.is\_valid.value\_counts()

If this also shows all True, we'll then run:

python

CopyEdit

fire\_centroids\_proj.head(3)

Let's confirm both the geometry and CRS of the fire centroids are valid **before** distance measurement. We’re almost there!

**You said:**

do this fire\_centroids\_proj.is\_valid.value\_counts() under the 3 new cells?

**ChatGPT said:**

✅ Yes — place it **under the 3 new cells** you just ran for the fire stations.

This keeps the logic clean:

1. First block → validates fire\_stations\_proj
2. Next block → validates fire\_centroids\_proj

Then we can directly compare and isolate the issue.

You're good to run it there!

**You said:**



**ChatGPT said:**

💥 There it is — the root cause!

**❗ All fire\_centroids\_proj geometries are invalid:**

plaintext

CopyEdit

False 608

This means every single centroid is broken or corrupted after projection — which fully explains the failed .distance() calculation and that huge 1.797693e+305 output.

**✅ Fix Path:**

Let’s go back and **validate the centroids list before projection** to ensure they were created correctly.

**🔍 New Diagnostic Cell — Check the original geometry:**

python

CopyEdit

fire\_centroids.is\_valid.value\_counts()

Run this just before you call .to\_crs(...).

If fire\_centroids is also False across the board, it means the issue began **when the centroids were calculated**, not during projection.

We'll confirm and fix from there. You’re just one step away now.

Top of Form

Bottom of Form