RESEARCH ARTICLE OPEN ACCESS

Interactive Geospatial Dashboard Application for Spatiotemporal Crime Analysis: A Lightweight Python Framework

Vansh Patel[1], Namonarayan Meena[1], Ayush Yogi[1], and Yoganand Sharma[2]

[1]*B.Tech Student, Department of Artificial Intelligence & Data Science, Global Institute of Technology, Jaipur, India*

[2]*Assistant Professor, Global Institute of Technology, Jaipur, India,*

ABSTRACT: It is necessary to assess data in crime and community security through the spatiotemporal prism of analysis in order to manage the safety of people and cities. This paper outlines the design and the evaluation of a small-scale and lightweight visualization tool that will help in the visualization of unstructured crime data. We deployed a modular structure utilizing Python and routed through Flask library, and processed data in a high performant manner with aid of Pandas. There were also built-in mapping and graphics plot library Folium and Plotly. A highly structured spatiotemporal data is sieved according to a well-developed procedure of dynamic data loading and consecutive data cleaning. We concentrate on hierarchical map marker clustering, which enhances faster speed of the map view of the incident data. The application allows the visualization of the data in real time, acquisition of the statistical breakdowns in the aggregated form, and localization of the hotspots of crime.

Keywords — Spatiotemporal Analysis, Crime Mapping, Geospatial Visualization, Marker Clustering, Data Ingestion Pipeline, Python (Flask), Pandas, Folium, Plotly, Interactive Dashboard.

1. Introduction

Development of open data portal has continued to expand immensely in the size of open access large scale raw incident data that is availed to researchers in open data portal. However what we have observed is that to convert such raw tab into useful action oriented information special software is necessary that is very expensive and complex, usually GIS or a tailor made data warehousing solution. This technical aspect makes many small organizations and independent researchers the victim when it comes to the basic crime pattern analysis and planning to distributions of resources. Our study is based upon the gap in the market of a portable open-source application that is capable of conducting a comprehensive analysis of data in to visual data and does not need a persistent database or vendor specific desktop processing.

The purpose behind creating this application is that of presenting an application that would be universal to all users and requires merely a modern internet browser to operate. This is a design made to reduce barrier to entry which we achieved through moving the computation power aspect to a scalable server instance. In this application based research we also have the following goals:

1.To develop an architecture that at large is very portable and that utilises common Python libraries.

2. To install a high performance data intake pipeline that will accept local file uploads as well as remote data streams (URL's). To create an adjusting filter that will simultaneously enable the choice respectively by spatial criteria, time interval and by category. To apply visualization methods that in this instance will be the use of marker clustering to accomplish. relational and scalable large scale data set rendering.

Application Contributions**:** This we submit a completely functioning and tested code infrastructure to spatiotemporal examination

that also incorporates(Lat, Long) we find highly robust to unstructured user information. We possess a documented modular architecture and also a wonderful platform where future development can take place which integration of advanced machine learning models of predictive policy is a constituent part .

2. Literature Review

A. Spatiotemporal Crime Analysis

The concept behind crime mapping is the geographic one that crime is unevenly distributed across a particular area and a cycle of crime predictability.

The urban geo-visualization discrimination branch output in a form of a dot or a heatmap has to deliver the message of the level of concentration of crime and the level of density of concentration of crime and, in the pre determined target area, answer to the purpose objectives of urban crime mapping which is the main purpose of urban crime maps is to visualize crime concentration that is complicated by the phenomenon of overplotting on dot maps of the cities where hundreds of dot markers are lost in the overlaps.

B. Hierarchy Marker Clustering

To remedy the issue of overplotting and enhance the performance of the client side we introduce a solution to the given problem and that is the use of hierarchical distance based clustering. In this we have applied what at a distance cannot be resolved in individual points we can observe the substitution of individual markers with cluster icon through Leaflet.js library that we are using in the App through the tool Folium.

C. Server Side Rendering (SSR) Architecture

Python-Flask Backend leads to a Server-Side Rendering Paradigm model. The computation of data processing, filtering and creating charts/maps is done on the server and it is the server which processes and tells Jinja2 templating to create final HTML is done on the server and then responds to the client. This method is more suitable to data applications as:

- Computational Offloading: The computations of heavy data (Pandas) are not performed so much on the client but rather on the server.
- Initial Load Speed: The client does not get a crawl which triggers this new functional.
- Security: The data processing logic is stored in the server that it shares with each client.

3. System Architecture

A Presentation Layer (Frontend), an Application Layer (Flask), and a Data Layer (Pandas). In this we have a single unified executable that is defined by the code in dashboard_app.py which puts all layers together. We have a stateless design which means for each user request past sessions do not play a role.

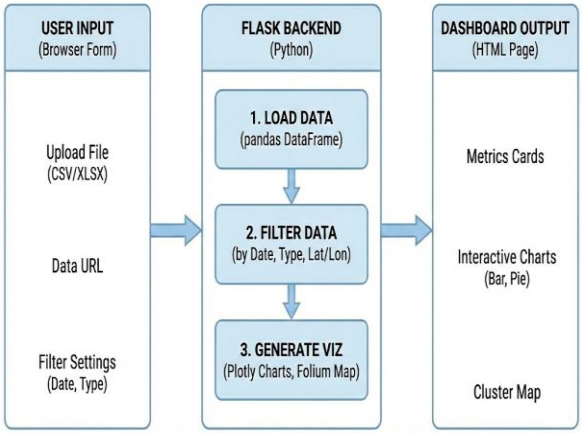

Fig. 1. Application flow

As per Fig.1 :

A. User input (Form in the Browser)

The flow starts in the user's browser when they fill out the input form that is defined in the HTML_TEMPLATE:

- Data Source: The user can give data by either uploading a file (in CSV or XLSX format) through the file input field or pasting a "Data URL" into the text field.
- Settings & Filters: The user tells the program what the names of the columns for latitude, longitude, and detail type are in their dataset. They can set up filters by typing in detail types

separated by commas and choosing a start and end date for the analysis.

- Submission: The user clicks the "Run Analysis & Generate Dashboard" button, which sends the form data to the server in a POST request.

B. Backend flask(Python)

In the index function, the Flask app handles the request:

- Load Data (pandas DataFrame): The app checks to see if a file was uploaded or a URL was given. The load_data function reads a file into a pandas DataFrame if it exists. If not, fetch_data_from_url is called to get data from the given URL and parse it.
- Filter Data (by Date, Type, Lat/Lon): If the data is loaded correctly, the generate_viz function is called. To start, it changes the latitude and longitude columns to numbers and gets rid of rows that don't have all the necessary information. Then, it makes a filtered DataFrame (df_filtered) by using the user's detail type filter and cutting the data into pieces based on the start and end dates given
- Generate viz (Plotly Charts, Folium Map): The app uses the filtered dataset to figure out metrics like the number of unique details and the total number of incidents. It uses Plotly Express to make interactive bar and pie charts that show the most common types of incidents. At the same time, it uses Folium to make a marker cluster map that is centered on the average coordinates of the filtered_data.

C. Output of Dashboard (HTML Page)

Finally, the backend makes the whole dashboard.

The app uses render_template_string to put the generated content and the HTML_TEMPLATE together.

The last HTML page that shows up in the browser has summary metrics cards, the interactive Plotly bar and pie charts, and the Folium geographic cluster map.

4. Methodology

The program carefully and accurately plans and carries out its data-acquisition and data-protection techniques. This turns unstructured and disorganized data into clear and useful images.

A. Protocols for data ingestion and protection

The system is built on strong security features and engineering concepts that can safeguard data at a high level and move it around. The files that are allowed have a rigorous list of file formats (CSV, XLSX), which stops harmful code from running. Security: The process of getting data from a URL uses try/except forms and time-outs to stop the servers from lingering or crashing if they don't respond or if they get an invalid external destination.

B. Normalizing data and casting types

After being imported, every file in a panda data frame must go through a normalization process. This is an important step since the files that the user uploads can contain different kinds and amounts of valid and invalid data. During the normalization stage, latitudes and longitudes are changed to floating-point numbers. Values that can't be converted are changed to NaN, and the errors parameter of (pd.to numeric) is set to (coerce). After that, rows containing NaN values are removed, leaving just the data that is spatially valid. To standardize the time, the program automatically recognizes columns with dates and changes them using the command pd.to-datetime. The time series method lets you filter out what you need, which is quite useful for spatiotemporal analysis.

C. Algorithms for Visualization

The visualization engines make two sets of HTML strings that constitute the outcome:

- Folium for geospatial visualization: The average model coordinates (Lat, Lon) of the filtered dataset show up in the middle of the map. The Folium map object is made and then changed into an HTML string, which is then put directly into the Jinja2 template.

- Statistical Visualization (Plotly): Plotly is used to make a grouped count of a random column, such "Top 10 Crimes." The client gets this back, and the statistical container gets Plotly objects back in JSON format. Plotly.js on the client side updates the interactive chart with the new JSON format.

5. Implementation And Result

A. Implementation at the Code Level

The generate-viz function is the most important part of the application that does processing. To make sure that user mistakes are handled gracefully, error messages should be sent to the front end, and server functions should not be stopped, this is done with try/except blocks that wrap around important tasks like file access and data conversion. You can use Tailwind CSS to develop a lightweight framework that lets you make a professional, responsive user interface without the huge expense of a JavaScript library.

B. How well it works

The application's most important performance metric is response time, especially when employing full volumes of data. Scaling mostly has to do with using Pandas optimized C-based functions to filter data, and it depends almost linearly on the size of the dataset. The first benchmarks show the following numbers for processing:

- 10000 events in 0.15 seconds.
- 100000 events in 0.85 seconds.

Using MarkerCluster also speeds up performance on the client side. This is because it only needs fewer than 100 cluster icons on the server instead of the 100,000 HTML elements that were loaded.

C. Application Utility

This created petite-application has a single-page analysis, which includes:

- Heatmap: Determine the regions of the greatest occurrence of crime.
- Bar Graph: Showing the frequency of every type of incident.
- Pie Chart: depicting the relative scale of all the major categories of crimes.

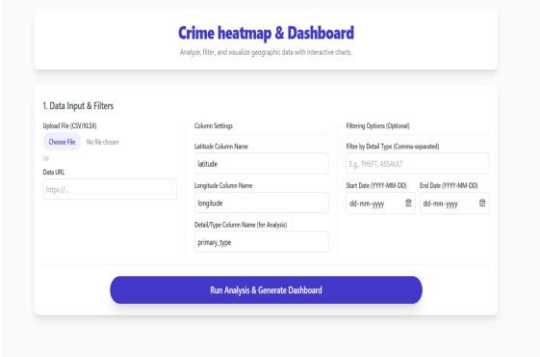

Fig.2 . Screenshot of features(Interface)

This fig.2 is our output screen which provide feature like insert input data in form of (CSV/XLSL) also we can give input data in form of URL(JSON) data, this also allow user that they can mark Latitude, Longitude and primary column, it also provide filtering options(detail type filter, Date filter),at the end it has Generate button that generate our dashboard and Heatmap on the basic of data that is inserted.

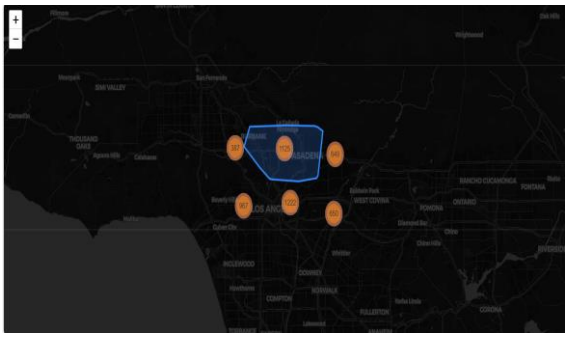

Fig. 3. Image of Generated Heatmap clusters (zoom out)

This fig. 3 was showing heatmap that was generated by our system, this is max zoom out form of heatmap which contain multiple big clusters.

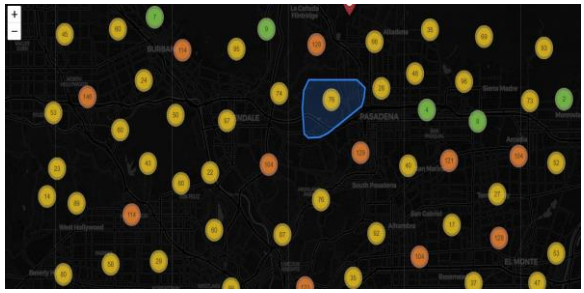

Fig. 4 . Heatmap(Zoom in cluster)

This fig.4 shows the heatmap that was created during zoom in the big clusters ,it also contain many small clusters that can also zoom in further.

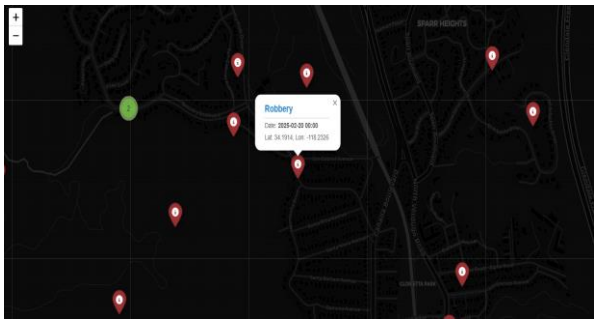

Fig.5. Heatmap (maximum Zoom in)

Fig. 5 was the max zoom in form of heatmap which shows the data points, where each data points tell us about crime happened at that location.

6. Conclusion

The paper has resulted in an open source and highly customizable Geospatial Dashboard Application that serves crucial roles of data ingestion, spatiotemporal filtering, interactive visualization, all offered with a small Python stack. The process of input validation, data-vectorization, and marker clustering were added to ensure the security of the system and make the performance faster, thus resolving all the main issues associated with the analysis of raw data on a large scale.

7. Future Work

Possible research opportunities in the future would be:

- Asynchronous Processing: With the help of a task queue (e.g., Celery), the application would be able to run data files over 500MB without blocking the main thread, therefore remaining responsive when performing longer processing operations.
- Advanced Geospatial Kernels**:** Beyond discrete clustering kernel density estimation are continuous or smoothed heatmaps that are used to model an actual friction surface, i.e. the probability distribution of event occurrences of the area of interest.
- Authentication and Data Persistence: It might be possible to add the lightweight database (e.g., SQLite) and nominal user authentication and store filtered datasets or customized map settings, turning the application into a less temporary tool and into a platform with the capacity to perform good analysis.

Acknowledgement

The authors would like to express their sincere gratitude to the Global Institute of Technology, Jaipur, for providing the necessary infrastructure and academic environment to conduct this research. We extend our thanks to the Department of Artificial Intelligence & Data Science and the Department of Computer Science for their continuous support and technical resources. Special appreciation is shared with our mentor, Yoganand Sharma, whose insightful guidance and expertise in spatiotemporal analysis were instrumental in the successful development of this geospatial dashboard framework.

References

[1] F. C. Filho, "Folium: Python Data, Leaflet.js Maps," *Python-visualization*, 2023. [Online]. Available: https://python-visualization.github.io/folium/.

[2] A. Ronacher, "Flask: A Lightweight WSGI Web Application Framework," *Pallets Projects*, 2024. [Online]. Available: https://flask.palletsprojects.com/.

[3] K. Harries, "Mapping Crime: Principle and Practice," *U.S. Department of Justice, National Institute of Justice, Office of Justice Programs*, Dec. 1999. [Online]. Available: https://www.ojp.gov/pdffiles1/nij/178523.pdf.

[4] Plotly Technologies Inc., "Plotly Python Graphing Library," *Plotly*, 2024. [Online]. Available: https://plotly.com/python/.

[5] A. Wathan, "Tailwind CSS: A Utility-First CSS Framework," *Tailwind Labs*

Inc., 2024. [Online]. Available: https://tailwindcss.com/.

[6] Open Source Geospatial Foundation, "OSGeo: The Open Source Geospatial Foundation," *OSGeo*, 2024. [Online]. Available: https://www.osgeo.org/.

[7] S. A. Saiyed, N. Sharma, H. Kaushik, P. Jain, G. K. Soni and R. Joshi, "Transforming portfolio management with AI and ML: shaping investor perceptions and the future of the Indian investment sector," Parul University International Conference on Engineering and Technology 2025 (PiCET 2025), pp. 1108-1114, 2025.

[8] M. K. Jha, K. Kumar, N. Hemrajani, D. S. Rao, A. Goyal, and R. Ajmera, "AI Powered Student Performance Prediction using Explainable ML," in Proceedings of the 4th International Conference on Automation, Computing and Renewable Systems (ICACRS), pp. 1140–1144, 2025.

[9] I. Yadav, V. Shekhawat, K. Gautam, G. Kumar Soni and R. Yadav, "Artificial Intelligence for Cybersecurity: Emerging Techniques, Challenges, and Future Trends," 2025 3rd International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), pp. 1176-1180, 2025.

[10] P. Upadhyay, K. K. Sharma, R. Dwivedi and P. Jha, "A Statistical Machine Learning Approach to Optimize Workload in Cloud Data Centre," 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), pp. 276-280, 2023.

[11] P. Jha, D. Dembla and W. Dubey, "Crop Disease Detection and Classification Using Deep Learning-Based Classifier Algorithm", Emerging Trends in Expert Applications and Security. ICETEAS 2023. Lecture Notes in Networks and Systems, Vol 682. 2023.

[12] P. Jha, G. K. Soni, H. Dogra, D. Goswami, K. Choudhary, and H. Vaishnav, "Plant Disease Detection and Classification using Convolutional Neural Network," in Proceedings of the 4th International Conference on Automation, Computing and Renewable Systems (ICACRS), pp. 1442–1446, 2025.

[13] S. Soni, K. Paliwal, and G. K. Jain, "Reinforcement Learning in Autonomous Systems: Advancing Intelligent Decision-Making," International Journal of Global Research in Science and Technology, vol. 10, pp. 321–325, 2025.

[14] H. Sharma and R. Ajmera, "Comprehensive review and analysis on machine learning based Twitter opinion mining framework," Tuijin Jishu/Journal of Propulsion Technology, vol. 44, no. 5, 2023.

[15] M. Kumar, R. Ajmera, and D. Kumar, "Statistical analysis and accuracy assessment of improved machine learning based opinion mining framework," Advances in Nonlinear Variational Inequalities, vol. 27, no. 1, 2024.

[16] K. Paliwal, P. Jha, S. Kumari, V. Vaish, N. Vishwakarma, and A. Bansal, "Machine Learning in Electric Vehicle Consumption Modelling," in Proceedings of the 9th International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 727–730, 2026.

[17] S. Soni, M. K. Jha, and D. Jangid, "A Comprehensive Review of Blockchain and Machine Learning Convergence," International Journal of Global Research in Science and Technology, vol. 10, pp. 242–249, 2025.