PyGeoModel: A Python Package for Integrating Intelligent Geographic Model Services for Urban Analysis in Jupyter

Peilong Ma[a,b,c], Min Chen[a,b,c,*], Dichen Liu[a,b,c], Wei Xie[a,b,c], Tianyu Sheng[a,b,c], Zaiyang Ma[a,b,c], Yongning Wen[a,b,c], Songshan Yue[a,b,c], Guonian Lv[a,b,c]

[a]*State Key Laboratory of Climate System Prediction and Risk Management, Nanjing Normal University, Nanjing, 210023, Jiangsu, China*
[b]*Key Laboratory of Virtual Geographic Environment (Ministry of Education of PRC), Nanjing Normal University, Nanjing, 210023, Jiangsu, China*
[c]*State Key Laboratory Cultivation Base of Geographical Environment Evolution, Nanjing, 210023, Jiangsu, China*

Abstract

Geographic modeling plays a critical role in addressing complex environmental and urban challenges, enabling researchers to simulate spatial processes across climate assessment, disaster prediction, and resource management. Jupyter has emerged as a widely adopted platform for modeling due to its capabilities in interactive computing, reproducible workflows, and collaborative research. However, a critical gap persists: Jupyter lacks seamless access to professional-level geographic models—sophisticated, domain-specific algorithms that encapsulate decades of expertise and are essential for rigorous urban system analysis. We have previously developed the Open Geographic Modeling and Simulation (OpenGMS) platform, which hosts over 3,000 validated models spanning urban hydrology, transportation analysis, and terrain modeling. Building upon this foundation, we introduce PyGeoModel, an open-source Python package that bridges Jupyter's interactive environment and OpenGMS's model repository. PyGeoModel delivers two key innovations: (1) a code-free graphical interface that enables direct model invocation, eliminating technical barriers for non-programmers, and (2) intelligent model services powered by large language models, including a context-aware model recommendation system and a knowledge-enhanced question-answering service for real-time model parameter guidance. We demonstrate PyGeoModel's effectiveness through a case study assessing urban rooftop photovoltaic potential in Xuanwu District, Nanjing, China. Using PyGeoModel's graphical interface and intelligent services, researchers complete the entire analysis within a single Jupyter notebook, from data loading and model selection to execution and visualization. What traditionally demands extensive manual configuration, specialized software, and deep domain knowledge is now accessible through intuitive interactions and AI-assisted guidance, advancing open urban data science and enabling interdisciplinary collaboration on urgent environmental challenges.

Keywords: Urban modeling, Geographic models, Jupyter Notebook, Large Language Model, OpenGMS, Reproducible Science

*Corresponding author: Min Chen. Email: chenmin0902@163.com

1. Introduction

Urban systems face unprecedented challenges in the era of rapid urbanization and climate change (Zhou, 2025), from managing stormwater runoff and air quality to optimizing energy infrastructure and transportation networks. Geographic modeling has become indispensable for evidence-based urban planning and policy-making (Goodchild, 2009), enabling researchers to simulate complex urban processes. By integrating sophisticated algorithms with diverse urban datasets spanning building footprints, mobility patterns, environmental sensors, and socioeconomic indicators, these models provide critical insights into climate change impacts (Sheng et al., 2025), natural disaster risks (Wang et al., 2023), and sustainable resource management (Lü et al., 2025). However, the technical complexity and fragmented toolchains of urban modeling workflows pose significant barriers to broader adoption, limiting the participation of interdisciplinary researchers.

Recent years have seen the rapid maturation of the Python geospatial ecosystem. Libraries such as GeoPandas and GDAL form the backbone of vector and raster data processing (Jüstel et al., 2022), alongside machine learning frameworks like TorchGeo. Jupyter plays a distinct role within this ecosystem by providing an interactive computing environment that seamlessly integrates these processing capabilities with visualization and narrative documentation (Wagemann et al., 2022). This integration significantly enhances workflow efficiency and scientific reproducibility (Pimentel et al., 2021). Leveraging Jupyter's unique widget infrastructure, the community has further developed tools that transform static scripts into interactive applications, such as geemap for observation data access (Wu, 2020), and leafmap or PyGMT for interactive mapping (Wu, 2021; Uieda et al., 2021). These tools demonstrate the platform's potential to streamline urban modeling workflows from data acquisition to visualization (Zhang et al., 2025).

Despite these advances, a critical gap persists within the Python computational ecosystem: the lack of seamless integration with urban analysis models. These sophisticated, validated models, such as the Storm Water Management Model (SWMM) (Gironás et al., 2010) for urban hydrology, the Integrated Valuation of Ecosystem Services (InVEST) (Nelson et al., 2009) for urban green space assessment, or specialized models for solar potential estimation, encapsulate decades of domain expertise and are essential for rigorous urban system analysis. These models often operate as standalone executables or rely on complex, non-standardized input formats, lacking native Python interfaces. Consequently, even within a unified environment like Jupyter, researchers are compelled to exit their computational workflows, navigate external platforms, manually transfer data, and execute models in separate interfaces. This fragmentation undermines the reproducibility and analytical coherence that Python-based workflows otherwise enable (Zhu et al., 2023).

This integration gap stems from two fundamental challenges. First, model availability: professional geographic models are distributed across institutional repositories with unique access protocols (Chen et al., 2020; Zhang et al., 2021), and many are computationally intensive, exceeding typical local computing capacities (Yang et al., 2017). Second, model usability: even when models are accessible via APIs, invoking them requires programming expertise to handle authentication, data serialization, parameter configuration, and result parsing (Morsy et al., 2017;

Ma et al., 2025a). These tasks demand domain-specific expertise and complex configurations (Chen et al., 2020; Longley and Chen, 2025), including parameters that require deep understanding of physical processes and local environmental conditions (Zhang et al., 2016). Moreover, researchers struggle to identify optimal models from vast repositories, selecting appropriate models demands understanding of model assumptions, input requirements, and regional applicability, expertise typically confined to domain specialists. These dual barriers of availability and usability limit participation from interdisciplinary researchers, constraining the broader impact of urban data science.

To address the availability challenge, we developed the Open Geographic Modeling and Simulation (OpenGMS) platform (Xu et al., 2024a) (`https://geomodeling.njnu.edu.cn`) in our prior work, which hosts over 3,000 validated models spanning urban hydrology, climatology, transportation, and terrain analysis. Through service-oriented architecture, OpenGMS transforms complex models into web-accessible services with standardized interfaces and distributed computing capabilities, enabling global access to professional models without local installation or configuration. This infrastructure establishes a foundation for model sharing and reuse, resolving the fragmentation and computational resource barriers that previously limited model accessibility.

Building upon the OpenGMS infrastructure, we introduce PyGeoModel, an open-source Python package designed to integrate geographic modeling services into the Jupyter environment. The package acts as a bridge for domain researchers and data scientists, enabling them to utilize specialized geographic models without mastering the specific implementation details of these models or deep domain expertise. To achieve this, PyGeoModel implements two primary capabilities: (1) a hybrid interface that enables model invocation via standard Python scripts, while providing an optional graphical interface for parameter configuration that can be converted into executable code; and (2) intelligent support services that deliver model recommendations and answer context-specific questions by analyzing notebook content—including research objectives, data characteristics, and spatial context. By connecting distributed model resources with this Python-native integration, PyGeoModel facilitates the rigorous application of urban modeling tools in interdisciplinary research.

2. Background

2.1. The Emergent Role of Jupyter in Urban Data Science

2.1.1. Core Advantages for Urban Analysis

Jupyter has emerged as the de facto standard for data-intensive scientific research (Perkel, 2018). Recent studies indicate a dramatic increase in Jupyter's usage within data science, with over 2.7 million public notebooks on GitHub by 2020 (Källén and Wrigstad, 2020) and a significant portion dedicated to urban data science applications (Gil et al., 2016; Söchting et al., 2025). As illustrated in Figure 1, Jupyter's widespread adoption is driven by four fundamental advantages—Interaction, Reproducibility, Collaboration, and Ecosystem—which collectively address longstanding challenges in urban data science.

- Interaction: Jupyter's interactive environment profoundly changes how researchers conduct research by enabling real-time experimentation with data and visualization (Choromański et al., 2023). Jupyter supports GUIs through widgets, enabling researchers to create sophisticated "Research GUIs" with minimal programming effort. These interfaces integrate scientific code with interactive elements—such as sliders, buttons, and menus—making advanced urban analyzes accessible to non-programmers and accelerating the iterative research process.

- Reproducibility: By combining code, data, documentation, and results in a single, shareable document, Jupyter enhances transparency and reproducibility (Samuel and Mietchen, 2024). For instance, a notebook built to assess urban solar potential can seamlessly bundle the Python scripts that calculate building sun exposure, the raw geospatial data of rooftops, interactive maps showing intermediate results, and the narrative text that documents the entire analytical methodology, ensuring that other researchers can replicate the study exactly as conducted, making Jupyter a preferred choice for publishing and sharing modeling workflows (Kluyver et al., 2016).

- Collaboration: Jupyter facilitates collaboration among researchers by enabling teams to work together. Notebooks can be hosted on cloud platforms, such as Binder or Google Colab, allowing remote researchers to edit, comment, and execute code collaboratively. Urban modeling often involves multidisciplinary collaboration, by providing a centralized platform for sharing and reviewing modeling process, Jupyter fosters interdisciplinary dialogue and accelerates the dissemination of research findings.

- Ecosystem: Jupyter's ecosystem offers researchers a comprehensive array of specialized tools and libraries. This ecosystem encompasses numerous libraries specifically designed for urban modeling, such as GeoPandas (Bossche et al., 2024) (for vector data processing), GDAL (Rouault et al., 2025) (for vector and raster operations) and xarray (Hoyer et al., 2024) (for multi-dimensional array analysis), which enable researchers to conduct urban modeling workflow within a single notebook.

2.1.2. Existing Toolchains and Established Workflows

The urban modeling community enables a relatively comprehensive modeling workflow through specialized tools, as systematically illustrated in Figure 2. We organize urban modeling across four sequential stages: (1) Data Acquisition and Processing is facilitated through interfaces like geemap (Wu, 2020) for accessing Google Earth Engine (Gorelick et al., 2017), and OSMnx (Boeing, 2017) for retrieving urban network data; (2) Statistical Modeling and Model Simulation demonstrates how TorchGeo (Stewart et al., 2022) and Hugging Face (Wolf et al., 2020) are employed for implementing machine learning algorithms; (3) Data Analysis and Visualization showcases functional implementations using PyL7VP (`https://li.antv.antgroup.com`) to transform model outputs into interactive visualizations; and (4) Collaboration and Notebook Sharing highlight publish mechanisms through nbviewer (`https:`

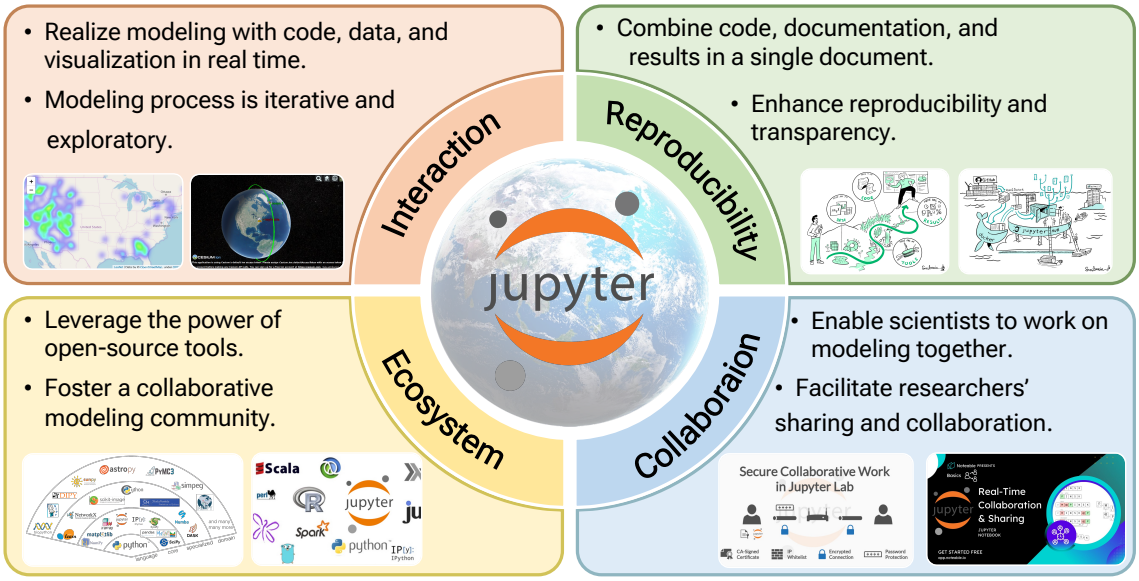

Figure 1: Jupyter's Advantages in Urban Data Science.

`//nbviewer.org`) and Binder (`https://mybinder.org`), researchers transform from private to shareable scientific products. These implemented toolchains demonstrate the concrete realization of Jupyter's theoretical potential, provide researchers with practical pathways to leverage Jupyter's capabilities across the complete modeling lifecycle

2.2. The Critical Integration Gap: Availability and Usability Challenges

Despite Jupyter's advancements in urban modeling workflows, a significant gap remains in integrating what constitutes the critical and resource-intensive step: professional geographic models. Lacking direct access, researchers must halt their analysis within the notebook, manually export their data into formats compatible with an external tool, execute the model in that separate environment, and then laboriously re-integrate the results back into Jupyter for continued analysis, undermining Jupyter's otherwise comprehensive environment for urban modeling. This gap stems from several challenges:

2.2.1. The Availability Challenge: Fragmentation and Computational Barriers

Professional geographical models are distributed across numerous institutional platforms (Chen et al., 2020), each with unique access protocols, execution environments, and documentation standards. This fragmentation forces researchers to navigate disparate systems outside Jupyter, manually transferring data between platforms and disrupting the unified workflow that Jupyter otherwise facilitates.

Many sophisticated geographical models demand substantial computational resources and specialized execution environments that exceed typical local computing capabilities. While Jupyter offers some distributed computing options, it lacks native support for orchestrating the high-performance computing infrastructure often required by professional geographical models.

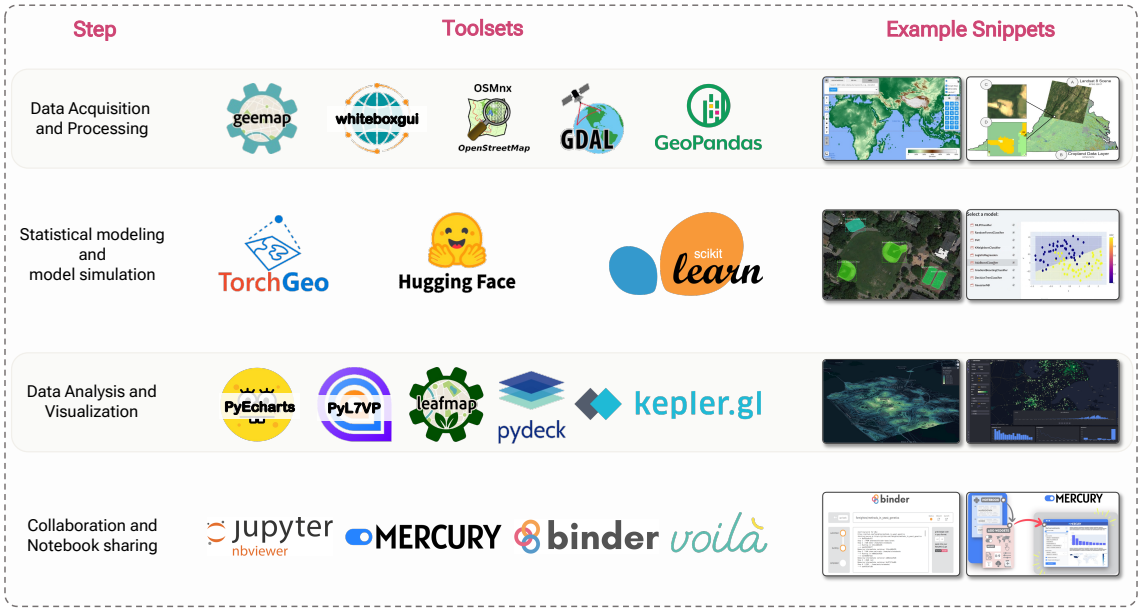

Figure 2: Urban modeling workflow and the corresponding tools in Jupyter. It illustrates the four-stage urban modeling workflow supported by Jupyter, as described in the text: Data Acquisition and Processing (e.g., geemap, GDAL), Statistical Modeling and Model Simulation (e.g., TorchGeo, scikit-learn), Data Analysis and Visualization (e.g., PyL7VP, leafmap), and Collaboration and Notebook Sharing (e.g., Binder, nbviewer). Example snippets demonstrate how these tools streamline fragmented workflows into a unified, interactive environment.

2.2.2. The Usability Challenge: Domain Expertise and Configuration Complexity

Unlike general-purpose data analysis tools or standard machine learning frameworks, professional geographical models typically incorporate sophisticated physics-based algorithms (Bhasme et al., 2022; Gurbuz et al., 2024), require complex setup procedures (Brookfield et al., 2023; Vivoni et al., 2011) and demand domain-specific expertise that extends beyond basic programming skills (Qin et al., 2025). This inherent complexity poses a significant usability challenge for researchers, and simultaneously complicates the task of creating standardized, user-friendly integrations for developers.

These gaps represent a significant barrier to realizing Jupyter's full potential as an integrated environment for urban modeling, as it forces researchers to fragment their workflows precisely at the stage where integrated analysis is most crucial.

2.3. OpenGMS: A Foundation for Solving Model Availability

OpenGMS is an open geographic modeling and simulation platform, designed to promote collaborative research by supporting the sharing of geographic resources such as models, tools, and data within the scientific community (Xu et al., 2024a). In particular, OpenGMS provides a robust environment where experts and users can upload, share, and access professional geographic models, as shown in Figure 3.

2.3.1. Model Sharing via Service-Oriented Architecture

OpenGMS employs a service-oriented architecture (SOA) to transform complex geoscientific models into web-distributed services, significantly enhancing their accessibility and reusability (Zhang et al., 2020). This mechanism addresses the challenges

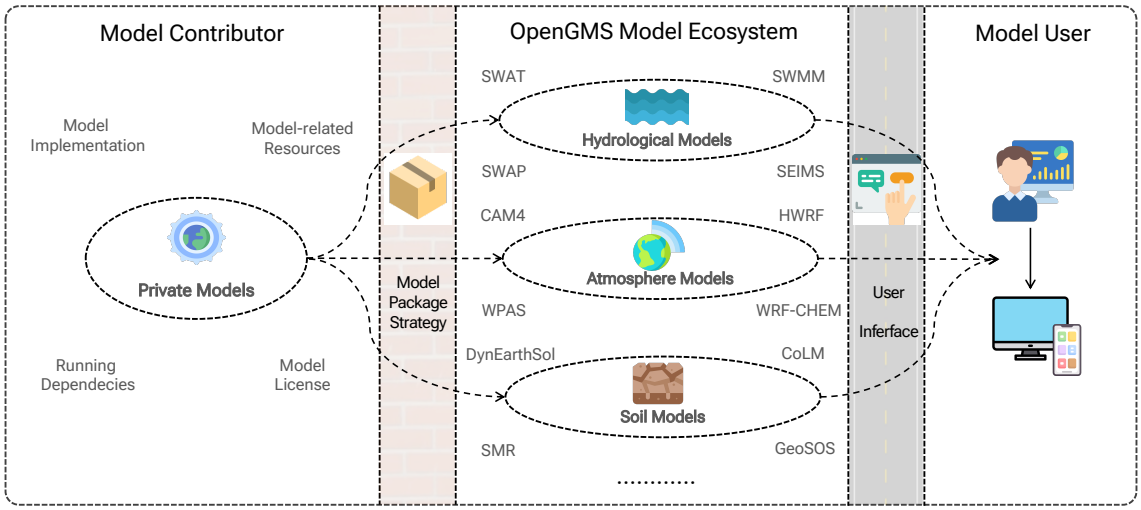

Figure 3: Model Sharing Process of OpenGMS. It shows how model contributors share on OpenGMS by packaging models with dependencies and licenses. The models are organized into categories like Hydrological, Atmospheric, and Soil Models, and made available to users worldwide through a user interface and distributed computing.

of resource fragmentation and technical barriers by enabling seamless integration and distributed computing across global networks.

The process begins with experts uploading their models through the OpenGMS portal, which bridges resource contributors and users. Experts encapsulate their models as services, packaging essential components such as source code, executable files, dependency libraries, and metadata into a standardized format (Zhang et al., 2019). The encapsulation leverages SOA principles, allowing models to be deployed as web services that can be invoked remotely via application programming interfaces. The platform's distributed computing framework further enhances this process by utilizing idle computing resources across the internet, enabling models to run on any connected computer worldwide without requiring users to download or configure complex local environments.

The upload and publication workflow are meticulously structured to ensure efficiency and adherence to FAIR principles (Findable, Accessible, Interoperable, Reusable) (Xu et al., 2024a). Experts first register their models on the OpenGMS portal, providing detailed metadata including the model's name, purpose, applicable domain, input/output specifications, and usage guidelines. OpenGMS then facilitates the transformation of these models into web services, generating comprehensive metadata to enhance their discoverability and interoperability. This structure organizes resources effectively and supports a classification system that categorizes models by application-focused (e.g., natural, human, or integrated perspectives) and method-focused criteria (e.g., data or process perspectives) (Xu et al., 2024b). After submission, these models will be available for global users to search, access, and invoke through a simple graphical interface.

2.3.2. The OpenGMS Repository and Community Ecosystem

OpenGMS has become a flourishing hub for geographic modeling and simulation (Bayer et al., 2021; Jiang et al., 2022), amassing contributions from experts across a

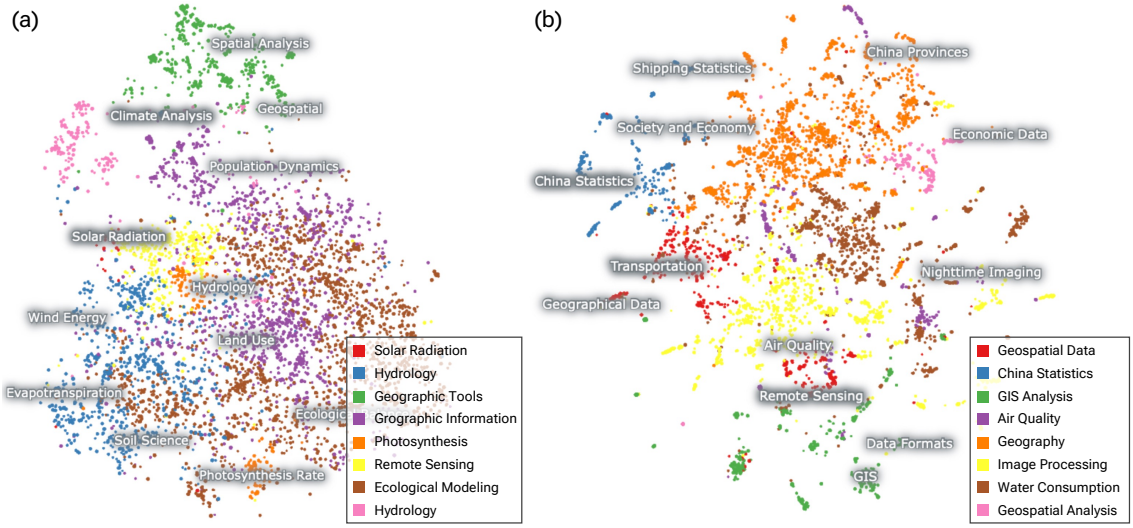

Figure 4: Semantic Maps of Resource Contexts on OpenGMS. (a) Model contexts, including themes like Climate Analysis and Hydrology; (b) Data contexts, covering areas such as Land Cover and Nighttime Imaging, highlighting the diverse themes and fields supported by OpenGMS.

wide range of domains, resulting in over 3,000 online models. It's now attracting a global community of researchers, practitioners, and institutions, fostering a vibrant ecosystem.

The diversity of model contributions is driven by a multidisciplinary network of experts who have enriched OpenGMS with specialized models, including prominent models like the Soil and Water Assessment Tool (SWAT) (Arnold et al., 2012), Storm Water Management Model (SWMM) (Gironás et al., 2010), Integrated Valuation of Ecosystem Services and Tradeoffs (InVEST) (Nelson et al., 2009), and Community Atmosphere Model (CAM) (Collins et al., 2006). OpenGMS's community ecosystem, comprising resource contributors—such as model developers, universities, and research institutes—and a diverse user base, including academics, policymakers, and environmental practitioners. OpenGMS's community ecosystem is further bolstered by its integration into broader scientific networks, such as CoMSES Net (Network for Computational Modeling in Social and Ecological Sciences) and the Open Modelling Foundation.

OpenGMS hosts a rich and diverse collection of geographic resources, encompassing both models and data, which cater to a wide array of research needs across natural, human, and integrated domains. As illustrated in the resource context semantic maps (Figure4 (a) and (b)), these resources span numerous themes and fields. Figure 4(a) illustrates the thematic landscape of the available models, with clusters representing key application areas like Climate Analysis, Hydrology, Wind Energy, Land Use, and Ecological Balance, enabling researchers to simulate processes such as evapotranspiration, population dynamics, and solar radiation impacts. Similarly, Figure 4(b) maps the subject matter of the platform's data, covering diverse fields from Shipping Statistics, Land Cover, and Nighttime Imaging to Air Quality and Economic Data, alongside specialized datasets like Historical Maps and Remote Sensing. This extensive coverage ensures that OpenGMS provides comprehensive support for interdisciplinary studies, from urban planning and environmental management to

climate change assessment, thereby facilitating collaborative and impactful research.

The vast repository of models and data on OpenGMS thus represents a major, yet largely untapped, resource for the geoscientific community. To unlock this potential and directly address the critical integration gap within Jupyter, we developed PyGeoModel—a tool designed to seamlessly connect these two powerful ecosystems.

3. PyGeoModel: A User-Centric Approach to Professional Urban Modeling in Jupyter

3.1. Overview

PyGeoModel realizes a user-centric approach for professional urban modeling in Jupyter. The package is designed to shift the researcher's focus from the complexities of technical implementation to the core of scientific inquiry. To achieve this, PyGeoModel is built upon two key components: (1) a code-free GUI that abstracts away the procedural steps of model configuration and execution, and (2) a suite of intelligent, LLM-powered services designed to lower the domain knowledge barrier. These services include a context-aware recommendation system to aid in model selection, and a knowledge-enhanced Q&A service to provide real-time guidance. This integrated, user-centric architecture streamlines the decision-making process, empowering researchers to engage with sophisticated models more efficiently.

As illustrated in Figure 5, PyGeoModel seamlessly integrates with Jupyter Notebook and the OpenGMS platform, facilitating the invocation of over 3000 validated geographic models. The system allows researchers to access, configure, and run these models through an intuitive GUI, supported by intelligent services such as model recommendation and knowledge-enhanced Q&A.

3.2. Graphical User Interface Architecture

PyGeoModel employs a three-tier architectural design that seamlessly integrates professional urban modeling capabilities within the Jupyter ecosystem. This architecture bridges the gap between Jupyter's interactive computational environment and the distributed OpenGMS backend infrastructure through a user-friendly interface paradigm.

The architecture comprises three interconnected layers that work in concert to deliver a cohesive modeling experience. At the presentation layer, the client-side GUI leverages Jupyter's native widget framework to provide an intuitive interface for model interaction. The ModelGUI class serves as the primary orchestrator, dynamically rendering interactive components including model selection dropdowns, file upload interfaces, parameter configuration panels, and execution controls through Jupyter's ipywidgets library. As demonstrated in Figure 6(a), this interface transforms complex modeling workflows into accessible point-and-click operations while maintaining full transparency of the underlying processes within the notebook environment. The architectural flow (Figure 6(b)) illustrates how the IPython kernel initializes PyGeoModel components and renders the interactive interface through ipywidgets, while the model selection and configuration interface (Figure 6(c)) showcases the practical implementation with features such as model browsing, parameter input, and intelligent assistance through the integrated Q&A system.

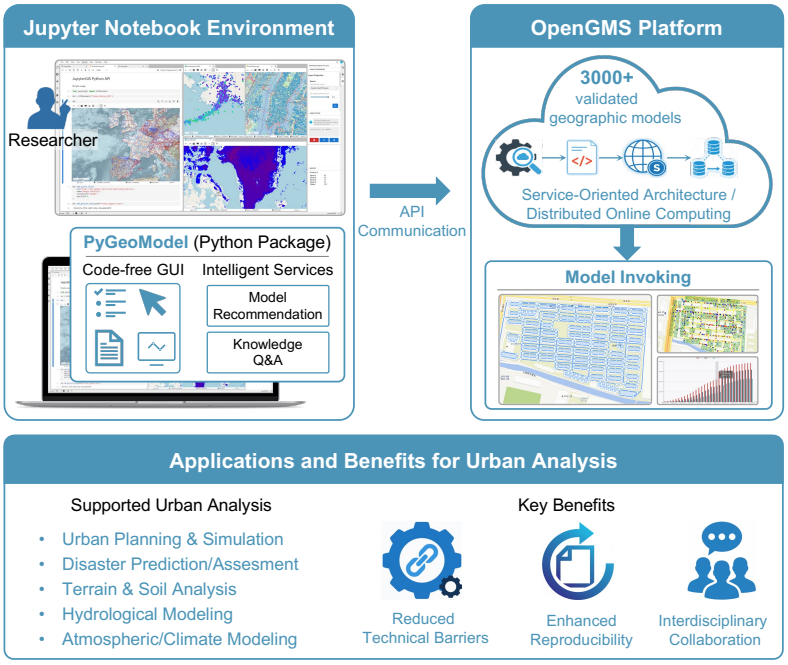

Figure 5: PyGeoModel and OpenGMS Integration Workflow for Urban Analysis. This figure shows the integration of PyGeoModel within the Jupyter Notebook environment and its connection to the OpenGMS platform. It highlights key functionalities and applications in urban analysis, including urban planning, disaster assessment, and environmental modeling.

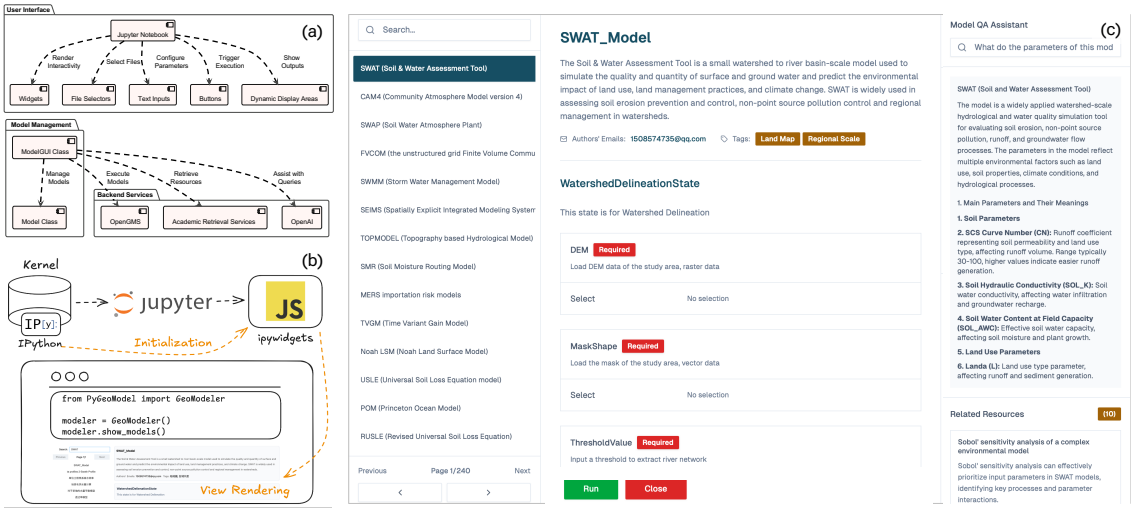

Figure 6: PyGeoModel Architecture for Urban Modeling in Jupyter. (a) User interface components and model management workflow showing the integration between GUI widgets and backend services; (b) Kernel initialization process illustrating how IPython kernel loads and renders PyGeoModel components through ipywidgets; (c) Model selection interface displaying SWAT model configuration.

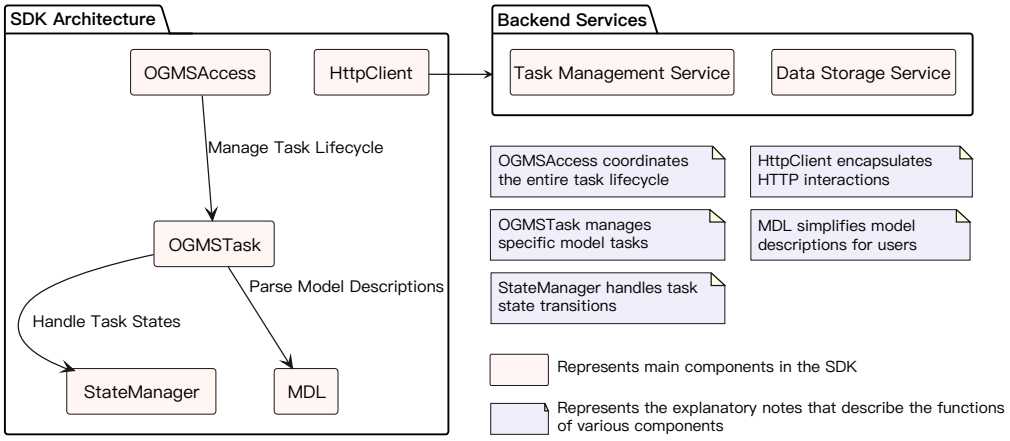

Figure 7: SDK Architecture for PyGeoModel's Model Execution in Jupyter. This figure illustrates the SDK supporting PyGeoModel's model invocation, as described in the text. Components like OGMSTask and StateManager manage task lifecycles and states, while MDL simplifies model descriptions for user interaction. OGMSAccess integrates with backend services (Task Management and Data Storage) to enable efficient model execution within Jupyter.

The intermediate layer consists of a Software Development Kit (SDK) that serves as the operational backbone connecting user interactions to backend services. This SDK architecture, detailed in Figure 7, encompasses five specialized components working in coordination: (1) OGMSAccess orchestrates the complete task lifecycle, managing the progression from model selection through result retrieval and ensuring seamless workflow continuity; (2) OGMSTask provides precise control over individual modeling operations, handling data uploads, execution monitoring, and status tracking with robust error handling capabilities; (3) HttpClient encapsulates all communication protocols with backend services, implementing secure and efficient data transfer mechanisms for task management and storage operations; (4) StateManager maintains persistent task state information, enabling reliable tracking of long-running computations and supporting workflow resumption capabilities; and (5) Model Description Language (MDL) parser translates complex model specifications into user-comprehensible configurations, abstracting technical implementation details while preserving modeling precision.

This layered architecture enables a streamlined yet powerful workflow by translating what are typically complex, code-based modeling requirements into a guided, graphical process. Once the user defines a task through this interface, the SDK transparently coordinates its execution across the OpenGMS distributed cloud infrastructure via standardized HTTP protocols. The resulting outputs are then seamlessly integrated back into the notebook environment, ready for immediate analysis and visualization. By distributing computational workloads to specialized backend resources, this design ensures exceptional scalability—supporting access to thousands of models without overwhelming local computing resources—while maintaining the responsive interactivity that characterizes an effective Jupyter workflow.

3.3. Context-aware intelligent model services powered by Large Language Models

The practical employment of professional geographical models relies on significant domain-specific expertise. It particularly impacts non-specialists attempting to

leverage these powerful analytical tools. Furthermore, with OpenGMS hosting over 3,000 models, researchers frequently encounter two critical bottlenecks in their workflows: first, navigating this vast selection to identify the optimal model that aligns with their specific research objectives, data characteristics, and regional contexts; and second, overcoming operational challenges in correctly parameterizing chosen models, managing input/output requirements, and interpreting results—tasks that typically require specialized knowledge often confined to domain experts or original model developers. These challenges significantly limit the broader adoption and impact of sophisticated urban modeling approaches across disciplines.

To address these critical limitations, we introduce context-aware intelligent model services powered by LLMs, seamlessly integrated within the PyGeoModel framework. These services harness natural language processing and reasoning capabilities to deliver two synergistic functionalities: (1) context-aware model recommendation that streamlines model selection by analyzing users' research contexts and automatically identifying optimal models from the extensive OpenGMS repository, and (2) knowledge-enhanced Q&A that provides real-time, domain-specific guidance for model configuration, parameter optimization, and result interpretation. This intelligent service architecture effectively bridges the complexity gap between sophisticated professional models and diverse user requirements, thereby democratizing access to advanced urban modeling capabilities while maintaining scientific rigor.

3.3.1. Dynamic Context Building Framework

To maximize the effectiveness of LLM in domain-specific applications, researchers must first solve a fundamental challenge: how to provide them with relevant, structured, and high-quality input. This has given rise to context engineering, a critical new field focused on the systematic design and optimization of the information provided to these models. PyGeoModel realizes this principle through a dynamic context-building framework specifically tailored for urban modeling workflows. This approach leverages in-context learning (Dong et al., 2022), a technique that enhances model performance by embedding rich, task-specific information directly within the prompts sent to the LLM. By actively constructing a context that understands the user's research intentions (Phua et al., 2024; Jin and Ma, 2024) and integrates domain-specific knowledge (Song et al., 2025), PyGeoModel's intelligent services can provide far more precise and relevant support.

Without a comprehensive understanding of the user's Jupyter notebook context—including ongoing analytical tasks, dataset characteristics, and research objectives—LLM responses may be too generic or misaligned with user needs. To address this challenge, we developed a dynamic context-building framework that integrates diverse elements of the workflow into a unified contextual profile. This approach systematically constructs the profile by synthesizing real-time information across three key dimensions:

- Modeling-History Context: This dimension is derived by parsing the content of Jupyter notebook, including markdown annotations, code cells, and outputs. Rather than treating the notebook's content as a static collection of keywords, the framework analyzes the ordered progression of these elements. This sequential analysis allows it to construct a timeline of the user's evolving

analytical objectives. For example, it can infer intent from narrative text and track the progression of data preprocessing steps, thus providing a dynamic understanding of the user's immediate analytical goals.

- Model-Repository Context: Sourced from the OpenGMS model repository, this dimension provides comprehensive metadata for over 3,000 models, including input/output specifications, domain classifications (e.g., hydrology, climatology), computational dependencies, and regional applicability. This information allows the framework to map user needs to the most suitable models within the repository, enhancing the precision of model recommendation. Beyond traditional keyword-based matching, this dimension implements semantic embedding techniques to map the functional capabilities and application domains of each model. This enables intelligent discovery through a conceptual match between user research contexts and model capabilities.

- Data-Repository Context: Encompassing metadata from datasets accessible to the user—including local files within the Jupyter's workspace or external resources linked to repositories like the Yangtze River Delta Science Data Center, National Earth System Data Center (`https://geodata.nnu.edu.cn`)—this dimension includes attributes such as geographic coverage, file size, temporal resolution, and structural format. The metadata enables the framework to assess data compatibility with available models, ensuring that recommendations account for practical constraints and research-specific requirements. This dimension implements a process of proactive data-model validation: it continuously evaluates the alignment between the user's data and a model's requirements across multiple dimensions, enabling the proactive identification of workflow bottlenecks and alternative solutions.

By aggregating these dimensions into a coherent contextual profile, this dynamic framework enhances the precision of PyGeoModel's intelligent services by embedding real-time, workflow-specific information into LLM prompts. For example, the model recommendation service utilizes the contextual profile to suggest the most suitable models from the OpenGMS repository, ensuring compatibility with the user's dataset and research objectives. Similarly, the Q&A service provides tailored guidance on parameter configuration and result interpretation, reducing technical barriers for users. By transforming fragmented workflows into a unified analytical process, this dynamic context framework enables LLMs to accurately understand the Jupyter Notebook context, thereby improving the effectiveness of intelligent services in PyGeoModel.

3.3.2. Context-Aware Model Recommendation: Improving Model Selection Efficiency

Navigating the vast repository of geographical models to find the optimal fit for a specific task can be a notable challenge for researchers. Selecting the right model requires careful consideration of research objectives, dataset properties, and regional applicability (Chen et al., 2021; Ma et al., 2022). Conventional methods for discovery, like manual browsing or keyword searching, face inherent limitations in this complex selection process. Manual browsing is often laborious and depends

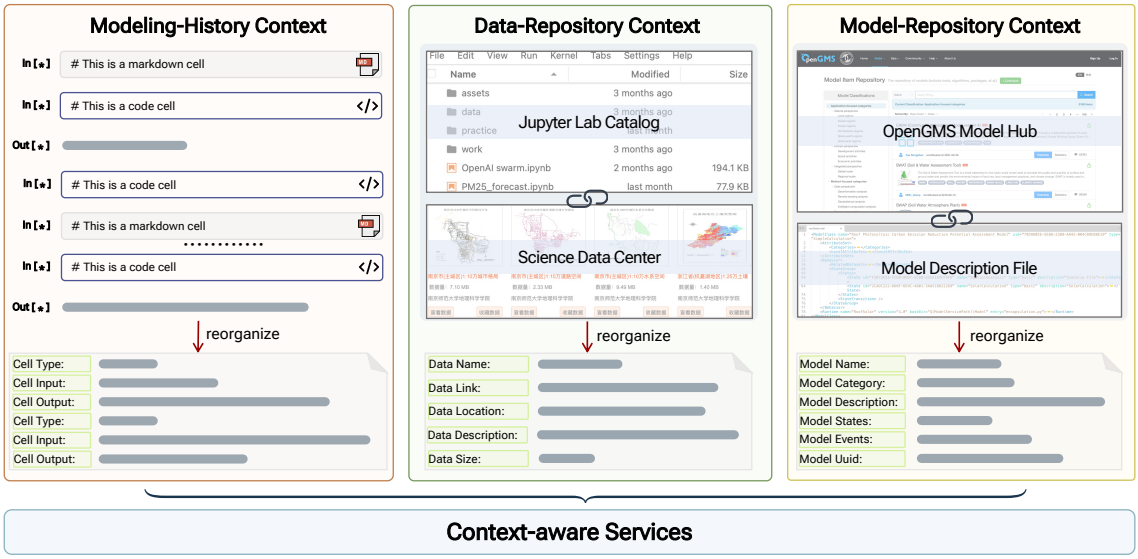

Figure 8: Dynamic Context-Building Framework for Context-Aware Intelligent Model Services in PyGeoModel. The framework integrates three dimensions—Modeling-History Context (extracted from Jupyter notebook cells), Data-Repository Context (sourced from Jupyter Lab and Science Data Center), and Model-Repository Context (derived from the OpenGMS Model Hub and model description files)—which are reorganized and synthesized into a unified context profile to support LLM-driven model recommendation and Q&A services.

heavily on the user's prior knowledge of model classification and applicability, making it easy to overlook suitable options. Keyword-based searches, while faster, often lack semantic understanding; they may return irrelevant results based on superficial term matching or fail to identify models whose descriptions don't perfectly align with the search query, overlooking critical aspects like data format compatibility, computational requirements, or the model's appropriateness for the specific stage of the user's analytical workflow. Neither approach effectively synthesizes the rich contextual information available — the user's ongoing analysis, available datasets, and the detailed specifications of potential models. To bridge this gap, PyGeoModel implements a context-aware model recommendation service that leverages the dynamic context-building framework and LLMs to streamline model selection, delivering tailored suggestions that enhance both efficiency and accuracy.

This service operates by harnessing the unified context profile constructed from the three dimensions outlined earlier: modeling-history, data-repository, and model-repository contexts. The process begins with the system analyzing the user's Jupyter activities to identify implicit or explicit modeling needs. For instance, a researcher preprocessing rooftop vector data and documenting an objective like "assessing urban photovoltaic potential" in markdown cells provides detectable signals of intent. These signals, together with dataset metadata (e.g., geographic scope, format) and model metadata from the OpenGMS repository (e.g., input requirements, domain tags), are integrated into an LLM-driven recommendation pipeline. The pipeline employs a multi-layered prompt structure, comprising: (1) a task definition layer that explicitly instructs the LLM to perform model recommendation; (2) a context injection layer that integrates current notebook content, data characteristics, and available model metadata; (3) an inference guiding layer that directs the LLM to evaluate candidates

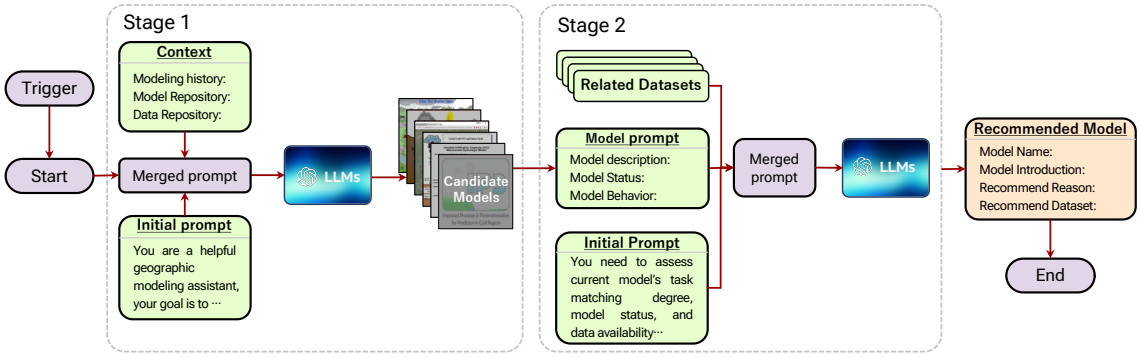

Figure 9: PyGeoModel's Two-Stage Strategy for Model Recommendation. Stage 1 merges context (modeling history, model/data repositories) with an initial prompt to generate candidate models using LLMs. Stage 2 refines the prompt with model details and datasets to deliver a final recommendation with reasoning.

based on multi-dimensional criteria, including data compatibility, computational complexity, and application scenario fit; and (4) an output formatting layer that standardizes the presentation of recommendation results.

The recommendation workflow, illustrated in Figure 9, unfolds through a two-stage strategy. In Stage 1, the dynamic context-building framework generates an initial context profile by parsing notebook content in real time. This profile is enriched with metadata from the data and model repositories, forming a comprehensive prompt. The LLM then assesses a shortlist of models, scoring them based on predefined criteria such as domain alignment, input/output compatibility, and contextual fit. In Stage 2, an iterative refinement step verifies each candidate's suitability by cross-referencing its metadata with the user's workflow specifics (such as regional applicability or preprocessing steps) to ensure practical applicability. The final output is presented in the output cell, showcasing a single, highly relevant model deemed most suitable based on the contextual analysis. This output takes the form of an interactive profile card for the recommended. This card features several key informational sections derived from the model's metadata and the contextual analysis: (1) Model Core Strengths, detailing its primary advantages for the specific context; (2) Recommendation Reason, justifying why this particular model aligns well with the user's inferred objectives and available data; (3) Application Scenario, suggesting practical research or operational contexts where the model can be effectively applied; and (4) Recommended Data Resources, specifying the required input datasets, including necessary local and potentially relevant datasets from integrated knowledge repository. Users can interact with elements within this profile, for instance, to follow links to view knowledge repository data or potentially to directly select and configure the recommended model for execution.

3.3.3. Knowledge-Enhanced Q&A: Supporting Geographical Modeling Expertise

After selecting an appropriate model, researchers often face challenges in understanding model mechanics, configuring parameters, or interpreting outputs due to limited domain-specific knowledge (Ma et al., 2025b). Much of this knowledge is available in published literature, model manuals, books, and online community forums, yet accessing and synthesizing it in real time remains a hurdle for users,

particularly non-experts. Rather than relying on external searches that often yield irrelevant results, PyGeoModel's knowledge-enhanced Q&A service provides tailored, authoritative guidance directly within Jupyter. By leveraging the dynamic context-building framework and LLMs, this service refines user queries to align with their actual needs, ensuring relevant and actionable support within Jupyter.

The service integrates two types of knowledge sources to address diverse user requirements, as illustrated in Figure 10. The first source taps into third-party knowledge repository, such as Elicit (`https://elicit.com`), Consensus (`https://consensus.app`), and Mosaic (`https://xiangqian.tech`), which specialize in thesis-based Q&A. These platforms have curated extensive databases of academic papers, enabling the retrieval of answers and related literature based on user queries. They can provide answers grounded in peer-reviewed research, along with citations to original papers, providing scientific insights (Whitfield and Hofmann, 2023).

While academic papers focus on theoretical frameworks, methodologies, and high-level insights, they frequently lack the fine-grained details about specific model parameters or practical configurations critical for effective application (Puetz et al., 2024). To address this gap, we developed a self-built knowledge repository comprising modeling manuals, related books, and community forum discussions specific to the geographical model repository. This knowledge repository captures detailed, model-specific information, such as parameter descriptions, typical value ranges, and usage scenarios—that complements the broader context from academic sources (Xu et al., 2024b). The retrieval process employs a RAG pipeline (Lewis et al., 2020), where content is segmented, vectorized using embedding techniques, and ranked for relevance through semantic matching, ensuring precise information retrieval.

The Q&A workflow seamlessly integrates these sources. Upon receiving a user query, PyGeoModel first leverages the dynamic context-building framework to refine it, drawing on the unified context profile (modeling-history, data-repository, and model-repository contexts) to clarify intent and align with the user's workflow. The refined query is then sent in parallel to both the third-party services and the self-built knowledge repository. The LLM synthesizes retrieved content from both third-party repositories (academic summaries with citations) and the self-built knowledge base (detailed parameter guidance). The LLM evaluates the relevance of each response based on the user's context and query specifics, selecting the most suitable answer or blending insights when appropriate. The final response will combine a theoretical explanation from a paper with practical configuration advice from the model manual, delivering both depth and applicability.

4. Case Study: Assessing Urban Rooftop Photovoltaic Potential

This case study demonstrates how PyGeoModel, integrated within Jupyter, streamlines the workflow for assessing the rooftop PV potential of Xuanwu District, Nanjing City, China. By leveraging graphic model invoking services, intelligent model services, this experiment showcases PyGeoModel's ability to simplify professional urban modeling, reduce technical barriers, and enhance reproducibility. The scenario follows a researcher equipped with Xuanwu District rooftop vector data, aiming to evaluate the district's PV potential distribution and total power generation, all within a unified, interactive notebook.

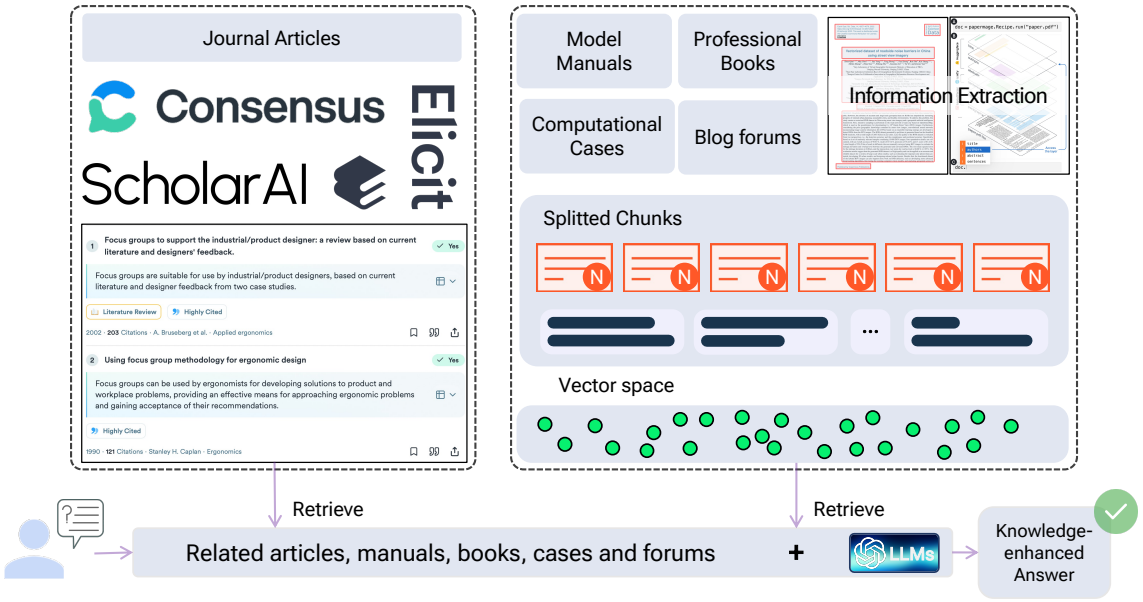

Figure 10: Knowledge-Enhanced Q&A Workflow in PyGeoModel. It retrieves academic answers directly via third-party APIs (e.g., Elicit, Consensus for journal), while model-specific details from a self-built knowledge repository (e.g., model manuals, computational cases) are split into chunks, vectorized, and processed. Both sources are then synthesized by LLMs to deliver a tailored, knowledge-enhanced answer.

The workflow encompasses six key stages: (1) background and objective definition using Jupyter's literate programming, (2) data loading, (3) exploratory data analysis, (4) model invocation via PyGeoModel, (5) result analysis, visualization, and (6) notebook publication through Binder. The tools used in this case study are listed in Table 1.

Table 1: Tools and Their Functions in the Case Study. This table lists the tools used, including their versions and specific roles in supporting background definition, data analysis, model invocation, visualization, and notebook sharing.

Tool	Version	Function in the case
folium	v0.12.0	Map visualization
PyGeoModel	v0.1.4	Model recommendation, interpretation, and invocation; Data recommendation
Geopandas	v1.0.1	Load and process data
Plotly	v6.0.1	Statistical data visualization
Binder	/	Notebook Release and Sharing

4.1. Experiment Design

The experiment simulates a realistic urban modeling research scenario where a researcher seeks to assess rooftop PV potential for urban energy planning. Nanjing, a major economic hub in eastern China with a rooftop area of approximately 330.36 km^2, serves as an ideal case due to its high energy demand and favorable solar conditions. The researcher begins with rooftop vector data of Xuanwu District, reflecting a common starting point in urban modeling studies. The design leverages

PyGeoModel's capabilities to bypass the labor-intensive computations typical of traditional PV potential assessments, such as those described by Zhong et al. (Zhong et al., 2021), while integrating seamlessly with Jupyter's ecosystem for analysis and sharing.

Traditional PV potential assessments, as exemplified by Zhong et al., involve a multi-stage, computationally intensive pipeline that demands significant expertise and resources. Solar radiation modeling integrates Copernicus Atmosphere Monitoring Service (CAMS) data with weather correction factors (e.g., monthly atmospheric transmittance and diffusion ratios), requiring complex equations and iterative computations to estimate per-rooftop potential. Power generation and validation steps involve additional manual data processing and statistical analysis, often using disparate tools like ArcGIS, Python, or MATLAB. Traditional workflow spans multiple software environments, necessitates proficiency in GIS, programming, and PV modeling, and consumes substantial time.

In contrast, PyGeoModel streamlines this process by encapsulating the entire multi-stage methodology of Zhong et al. into a single model named the 'Roof Photovoltaic Carbon Emission Reduction Potential Assessment Model', which is accessible through an interactive, graphical interface within Jupyter. The user's role is reduced to two primary actions: uploading their rooftop vector data and configuring key parameters (such as the assessment time period) via the GUI. PyGeoModel then handles the entire subsequent computational pipeline—CAMS data integration, solar radiation modeling, potential estimation, and attribute assignment. This automation reduces the workflow from dozens of manual steps to a few GUI interactions, requiring no coding or specialized hardware. By integrating intelligent services (e.g., model recommendation and parameter guidance), PyGeoModel further alleviates the technical burden, making professional-grade PV assessment accessible to users without advanced GIS or programming skills.

4.2. Implementation Details

The experiment was conducted entirely within a Jupyter Notebook to showcase the end-to-end integration of PyGeoModel, hosted on the widely accessible Google Colab cloud platform to ensure transparency and reproducibility. As illustrated in Figure 11, this all-in-one workflow is structured into six sequential stages, which are detailed below.

Step 1: Background Documentation. The Jupyter Notebook commences with markdown cells that delineate the research context: Nanjing, with a population of 8.44 million, faces increasing energy demands amidst China's transition to renewable energy. This study aims to quantify Xuanwu district's rooftop PV potential distribution and evaluate its power generation using existing vector data. This documentation establishes the foundation for the subsequent technical workflow, utilizing Jupyter's literate programming features to enhance clarity and readability, embedding the 'why' alongside the 'how'.

Step 2: Data Loading. Xuanwu District rooftop vector data, approximately 6 MB in shapefile format, are imported into the Jupyter Notebook using GeoPandas. This step ensures the rooftop dataset is seamlessly integrated into the computational environment for further analysis, a common starting point for many urban modeling workflows.

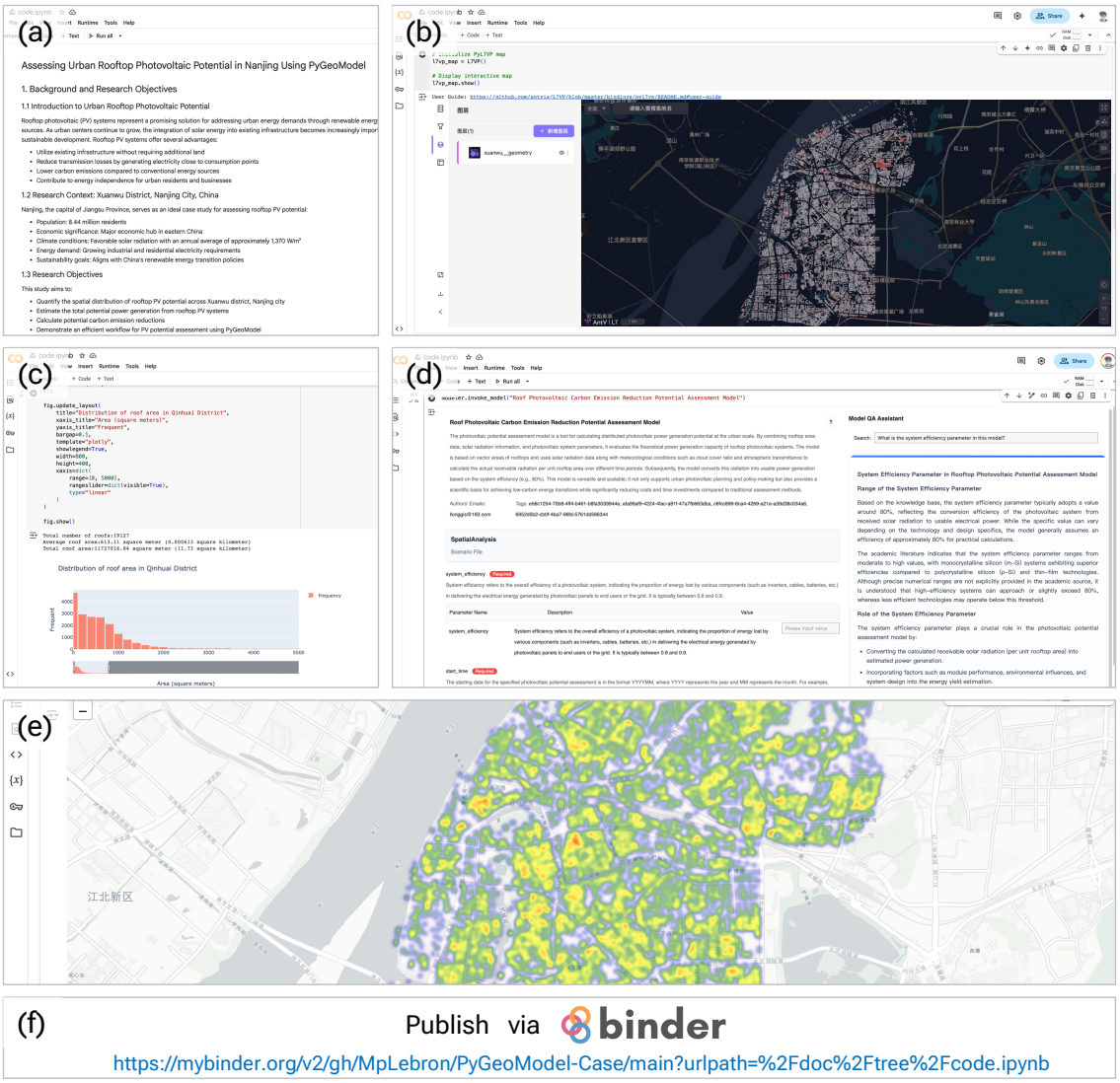

Figure 11: The all-in-one Workflow and Results of Assessing Rooftop Photovoltaic Potential. (a) Background documentation in the Jupyter notebook, outlining the research context and objectives for assessing rooftop photovoltaic (PV) potential in Nanjing, China. (b) Interactive map visualizing the spatial distribution of rooftops in Xuanwu District, generated using Folium during exploratory data analysis (EDA). (c) Statistical analysis of rooftop areas in Xuanwu District, showing a histogram of rooftop sizes (in square meters) with an average area of 615 m^2, created using Matplotlib. (d) PyGeoModel GUI interface for model invocation, displaying the "Roof Photovoltaic Carbon Emission Reduction Potential Assessment Model" model selection, parameter configuration (e.g., PV conversion efficiency, solar radiation intensity), and execution options. (e) Heat map of PV potential distribution across Nanjing's rooftops, generated using Folium, with color gradients indicating potential in kWh/m^2 (yellow for high potential, purple for low). (f) Binder publication link for the notebook, enabling global access and replication of the experiment.

Step 3: Exploratory Data Analysis. Exploratory Data Analysis is performed to examine the characteristics of the loaded dataset. Basic statistical metrics are computed, revealing 19,128 rooftops with an average area of 615 m^2. The spatial distribution of rooftops is visualized using Folium, producing an interactive map embedded within the notebook. Markdown cells document the analysis outcomes, noting that the central urban zone exhibits a higher rooftop density, indicating a potentially greater PV capacity. This interactive exploration helps build understanding before formal modeling.

Step 4: Model Invocation. PyGeoModel is initialized through the invoke_model() function, which renders an interactive interface within the Jupyter Notebook. The suggest_model() function is then employed to analyze the notebook's context, including markdown references to "PV potential" and the rooftop dataset type, subsequently recommending the "Roof Photovoltaic Carbon Emission Reduction Potential Assessment Model", pre-encapsulated based on the methodology of Zhong.

The model requires an additional parameter named system_efficiency. This crucial parameter represents the overall efficiency of the PV system in converting solar energy into usable electricity delivered to the end-user, accounting for losses from components such as inverters, cables, and batteries. Through PyGeoModel's knowledge-enhanced Q&A functionality, a query is posed: "What is an appropriate value for system efficiency?". PyGeoModel retrieves information from its knowledge base, providing the response: "System efficiency accounts for energy losses during DC-to-AC conversion, transmission, and due to component degradation. For typical rooftop installations, this value ranges from 0.8 (80%) to 0.9 (90%). A standard, conservative value of 0.8 is recommended for this type of potential assessment." This context-specific guidance, obtained directly through the Q&A service, saves the user the effort of researching component standards, performance ratios, or other system derating factors. Based on this recommendation, the value 0.8 is input via the GUI, validated by the ModelGUI class, and the model is executed using the "Run" button. Utilizing OpenGMS's distributed computing infrastructure, the task completes in approximately 2 minutes, with results downloaded to the user's local storage.

Step 5: Result Analysis and Visualization. The resulting dataset includes PV potential attributes (in kWh/m^2) for each rooftop. An interactive heat map, generated using Folium, visualizes the spatial distribution of PV potential. Markdown cells provide interpretive insights, stating that suburban industrial rooftops exhibit the greatest potential due to their larger areas and fewer obstructions, directly linking the visual output back to the research questions.

Step 6: Publication via Binder. To facilitate maximum reproducibility and promote open science principles, the notebook is uploaded to a GitHub repository and linked to Binder. Binder creates a fully executable environment directly from the repository in the cloud, bundling the code, necessary dependencies (like PyGeoModel, GeoPandas, etc.), and the notebook itself. This allows any researcher, educator, student, or policymaker globally to re-run the entire analysis exactly as performed with just a web browser and a single click, ensuring transparency and verification of the results.

4.3. Results and Discussion

The experiment yields a comprehensive PV potential assessment for Xuanwu District, Nanjing City. The estimated annual power generation is 2,365.28 GWh. The workflow, completed in under 20 minutes, contrasts sharply with traditional methods, which require hours of manual computation and data integration. The GUI eliminates coding requirements, enabling non-programmers to perform professional-grade analysis. The intelligent services, model recommendation and Q&A, enhance decision-making, while distributed computing offloads resource demands. Binder publication further distinguishes this approach, offering a reproducible, cloud-based alternative to static research outputs.

This case study underscores PyGeoModel's transformative potential in urban modeling. By integrating distributed model services into Jupyter, it empowers researchers to tackle complex problems with minimal technical overhead. The seamless workflow, from data preprocessing to result sharing, embodies the all-in-one modeling paradigm, reducing fragmentation and enhancing efficiency. The experiment's success in replicating findings with fewer resources demonstrates its reliability for professional applications, while its accessibility broadens participation to policymakers, educators, and novices. The Binder-hosted notebook invites global collaboration, aligning with open science principles and reinforcing PyGeoModel's role in advancing reproducible urban modeling research.

5. Conclusion

This study introduced PyGeoModel, an open-source Python package that addresses critical model accessibility challenges in urban modeling by seamlessly integrating over 3,000 professional models into Jupyter's interactive computing environment. Building upon the OpenGMS platform's solution to model availability, PyGeoModel tackles the usability challenge through a dual-component architecture: a code-free graphical interface that eliminates programming barriers for non-technical users, and intelligent LLM-driven services that provide context-aware model recommendations and real-time parameter guidance. By democratizing access to professional urban modeling capabilities while maintaining complete analytical transparency and reproducibility, PyGeoModel advances open urban data science and enables broader participation from interdisciplinary researchers, practitioners, and stakeholders.

The Nanjing rooftop photovoltaic potential assessment demonstrates PyGeoModel's practical impact. What traditionally required dozens of manual steps across multiple platforms was condensed into a single, interactive notebook that maintains complete analytical transparency and reproducibility. The context-aware model recommendation service guides users toward appropriate models based on research context, while the knowledge-enhanced Q&A service provides critical parameter guidance that typically requires deep domain expertise. By eliminating technical barriers that fragment workflows across disparate platforms, PyGeoModel enables interdisciplinary teams to focus on analytical insights rather than software navigation complexities.

Despite these contributions, several limitations warrant acknowledgment. PyGeoModel's current implementation relies on the OpenGMS platform's availability and

network connectivity, which may pose challenges for users in regions with limited internet access or during platform maintenance periods. Additionally, while the LLM-driven services significantly lower technical barriers, they inherit the inherent limitations of large language models, including potential hallucinations in model recommendations and parameter guidance. These issues underscore the need for ongoing refinement of the intelligent service components. Future work will focus on enhancing the reliability of LLM-driven services through implementing confidence scoring mechanisms for model recommendations and parameter suggestions. Such validation frameworks would provide users with transparent uncertainty estimates, enabling more informed decision-making when selecting and configuring models.

As urban system challenges intensify and demand increasingly sophisticated analytical responses, tools like PyGeoModel become essential for fostering the collaborative, transparent, and reproducible research practices that effective urban stewardship requires. This work represents a step toward more democratic and impactful urban modeling, where sophisticated analytical capabilities extend beyond traditional disciplinary boundaries to address urgent urban challenges.

CRediT authorship contribution statement

Peilong Ma: Writing – review & editing, Writing – original draft, Software, Methodology, Formal analysis, Conceptualization. Min Chen: Writing – review & editing, Supervision, Methodology, Funding acquisition. Dichen Liu: Writing – review & editing, Resources, Data curation, Conceptualization. Wei Xie: Writing – review & editing, Resources. Tianyu Sheng: Writing – review & editing, Formal analysis. Zaiyang Ma: Writing – review & editing, Formal analysis. Yongning Wen: Supervision, Funding acquisition. Songshan Yue: Supervision, Funding acquisition. Guonian Lv: Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We appreciate the detailed suggestions and comments from the editor and anonymous reviewers. We express heartfelt thanks to the other members of the OpenGMS team. This work was supported by the National Natural Science Foundation (NSF) of China (Grant No.42325107 and Grant No.42401574).

Code and Data availability

The source code of the package is available through the GitHub repository: https://github.com/MpLebron/PyGeoModel, and data will be made available on request.

References

Arnold, J.G., Moriasi, D.N., Gassman, P.W., Abbaspour, K.C., White, M.J., Srinivasan, R., Santhi, C., Harmel, R., Van Griensven, A., Van Liew, M.W., et al., 2012. Swat: Model use, calibration, and validation. Transactions of the ASABE 55, 1491–1508.

Bayer, T., Ames, D.P., Cleveland, T.G., 2021. Design and development of a web-based epanet model catalogue and execution environment. Annals of GIS 27, 247–260.

Bhasme, P., Vagadiya, J., Bhatia, U., 2022. Enhancing predictive skills in physically-consistent way: Physics informed machine learning for hydrological processes. Journal of Hydrology 615, 128618.

Boeing, G., 2017. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. Computers, environment and urban systems 65, 126–139.

Brookfield, A.E., Ajami, H., Carroll, R., Tague, C., Sullivan, P., Condon, L., 2023. Recent advances in integrated hydrologic models: Integration of new domains. Journal of Hydrology 620, 129515.

Chen, H.x., Tao, F., Ma, P.l., Gao, L.n., Zhou, T., 2021. Applicability evaluation of several spatial clustering methods in spatiotemporal data mining of floating car trajectory. ISPRS International Journal of Geo-Information 10, 161.

Chen, M., Voinov, A., Ames, D.P., Kettner, A.J., Goodall, J.L., Jakeman, A.J., Barton, M.C., Harpham, Q., Cuddy, S.M., DeLuca, C., et al., 2020. Position paper: Open web-distributed integrated geographic modelling and simulation to enable broader participation and applications. Earth-Science Reviews 207, 103223.

Choromański, K., Gotlib, D., Łobodecki, J., 2023. Jupyter technology as a cartographer's work environment: A case study, in: Proceedings of the ICA, Copernicus Publications Göttingen, Germany. p. 4.

Collins, W.D., Rasch, P.J., Boville, B.A., Hack, J.J., McCaa, J.R., Williamson, D.L., Briegleb, B.P., Bitz, C.M., Lin, S.J., Zhang, M., 2006. The formulation and atmospheric simulation of the community atmosphere model version 3 (cam3). Journal of Climate 19, 2144–2161.

Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Liu, T., et al., 2022. A survey on in-context learning. arXiv preprint arXiv:2301.00234 .

Gil, Y., David, C.H., Demir, I., Essawy, B.T., Fulweiler, R.W., Goodall, J.L., Karlstrom, L., Lee, H., Mills, H.J., Oh, J.H., et al., 2016. Toward the geoscience paper of the future: Best practices for documenting and sharing research from data to software to provenance. Earth and Space Science 3, 388–415.

Gironás, J., Roesner, L.A., Rossman, L.A., Davis, J., 2010. A new applications manual for the storm water management model(swmm). Environmental Modelling & Software 25, 813–814.

Goodchild, M.F., 2009. Geographic information systems and science: today and tomorrow. Annals of GIS 15, 3–9.

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., Moore, R., 2017. Google earth engine: Planetary-scale geospatial analysis for everyone. Remote sensing of Environment 202, 18–27.

Gurbuz, F., Mudireddy, A., Mantilla, R., Xiao, S., 2024. Using a physics-based hydrological model and storm transposition to investigate machine-learning algorithms for streamflow prediction. Journal of Hydrology 628, 130504.

Jiang, J., Pang, T., Zhang, F., Men, Y., Yadav, H., Zheng, Y., Chen, M., Xu, H., Zheng, T., Wang, P., 2022. Pathway to encapsulate the surface water quality model and its applications as cloud computing services and integration with edss for managing urban water environments. Environmental Modelling & Software 148, 105280.

Jin, Y., Ma, J., 2024. Large language model as parking planning agent in the context of mixed period of autonomous vehicles and human-driven vehicles. Sustainable Cities and Society 117, 105940.

Jüstel, A., Correira, A.E., Pischke, M., de la Varga, M., Wellmann, F., 2022. Gemgis-spatial data processing for geomodeling. Journal of Open Source Software 7, 3709.

Källén, M., Wrigstad, T., 2020. Jupyter notebooks on github: characteristics and code clones. arXiv preprint arXiv:2007.10146 .

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., et al., 2016. Jupyter notebooks–a publishing format for reproducible computational workflows, in: Positioning and power in academic publishing: Players, agents and agendas. IOS press, pp. 87–90.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al., 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems 33, 9459–9474.

Longley, P., Chen, M., 2025. Smart data, information geographies and intelligent data services. Information Geography 1, 100013.

Lü, G., Xiong, J., Wu, M., Yuan, L., Li, J., Chen, M., Yu, Z., Zhou, L., Yue, S., Zhang, X., et al., 2025. Geographic information discipline development: Demands, strategy and challenges. Information Geography 1, 100012.

Ma, P., Chen, M., Zhang, S., Zhu, Z., Qian, Z., Ma, Z., Zhang, F., Li, W., Yue, S., Wen, Y., 2025a. Facilitating sensitivity analysis of hydrological models through knowledge-driven configuration and distributed online model services. Journal of Hydrology , 133406.

Ma, P., Tao, F., Gao, L., Leng, S., Yang, K., Zhou, T., 2022. Retrieval of fine-grained pm2. 5 spatiotemporal resolution based on multiple machine learning models. Remote Sensing 14, 599.

Ma, Z., Li, H., Zhang, K., Wang, J., Yue, S., Wen, Y., Lü, G., Chen, M., 2025b. Knowledge co-creation during urban simulation computation to enable broader participation. Sustainable Cities and Society 118, 105994.

Morsy, M.M., Goodall, J.L., Castronova, A.M., Dash, P., Merwade, V., Sadler, J.M., Rajib, M.A., Horsburgh, J.S., Tarboton, D.G., 2017. Design of a metadata framework for environmental models with an example hydrologic application in hydroshare. Environmental Modelling & Software 93, 13–28.

Nelson, E., Mendoza, G., Regetz, J., Polasky, S., Tallis, H., Cameron, D., Chan, K.M., Daily, G.C., Goldstein, J., Kareiva, P.M., et al., 2009. Modeling multiple ecosystem services, biodiversity conservation, commodity production, and tradeoffs at landscape scales. Frontiers in Ecology and the Environment 7, 4–11.

Phua, S.Z., Hofmeister, M., Tsai, Y.K., Peppard, O., Lee, K.F., Courtney, S., Mosbach, S., Akroyd, J., Kraft, M., 2024. Fostering urban resilience and accessibility in cities: A dynamic knowledge graph approach. Sustainable Cities and Society 113, 105708.

Pimentel, J.F., Murta, L., Braganholo, V., Freire, J., 2021. Understanding and improving the quality and reproducibility of jupyter notebooks. Empirical Software Engineering 26, 65.

Puetz, S.J., Condie, K.C., Sundell, K., Roberts, N.M., Spencer, C.J., Boulila, S., Cheng, Q., 2024. The replication crisis and its relevance to earth science studies: Case studies and recommendations. Geoscience Frontiers 15, 101821.

Qin, C.Z., Zhu, L.J., Zhu, A.X., 2025. Position paper: Domain knowledge-driven intelligentization of watershed modeling and scenario analysis for precise watershed management. Information Geography , 100016.

Samuel, S., Mietchen, D., 2024. Computational reproducibility of jupyter notebooks from biomedical publications. GigaScience 13, giad113.

Sheng, T., Zhang, Z., Qian, Z., Ma, P., Xie, W., Zeng, Y., Zhang, K., Sun, Z., Yu, J., Chen, M., 2025. Examining urban agglomeration heat island with explainable ai: An enhanced consideration of anthropogenic heat emissions. Urban Climate 59, 102251.

Söchting, M., Scheuermann, G., Montero, D., Mahecha, M.D., 2025. Interactive earth system data cube visualization in jupyter notebooks. Big Earth Data , 1–15.

Song, Z., Yan, B., Liu, Y., Fang, M., Li, M., Yan, R., Chen, X., 2025. Injecting domain-specific knowledge into large language models: A comprehensive survey. arXiv preprint arXiv:2502.10708 .

Stewart, A.J., Robinson, C., Corley, I.A., Ortiz, A., Ferres, J.M.L., Banerjee, A., 2022. Torchgeo: deep learning with geospatial data, in: Proceedings of the 30th international conference on advances in geographic information systems, pp. 1–12.

Uieda, L., Tian, D., Leong, W.J., Toney, L., Schlitzer, W., Grund, M., Newton, D., Ziebarth, M., Jones, M., Wessel, P., 2021. Pygmt: A python interface for the generic mapping tools .

Vivoni, E.R., Mascaro, G., Mniszewski, S., Fasel, P., Springer, E.P., Ivanov, V.Y., Bras, R.L., 2011. Real-world hydrologic assessment of a fully-distributed hydrological model in a parallel computing environment. Journal of Hydrology 409, 483–496.

Wagemann, J., Fierli, F., Mantovani, S., Siemen, S., Seeger, B., Bendix, J., 2022. Five guiding principles to make jupyter notebooks fit for earth observation data education. Remote Sensing 14, 3359.

Wang, H., Xu, S., Xu, H., Wu, Z., Wang, T., Ma, C., 2023. Rapid prediction of urban flood based on disaster-breeding environment clustering and bayesian optimized deep learning model in the coastal city. Sustainable Cities and Society 99, 104898.

Whitfield, S., Hofmann, M.A., 2023. Elicit: Ai literature review research assistant. Public Services Quarterly 19, 201–207.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al., 2020. Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations, pp. 38–45.

Wu, Q., 2020. geemap: A python package for interactive mapping with google earth engine. Journal of Open Source Software 5, 2305.

Wu, Q., 2021. Leafmap: A python package for interactive mapping and geospatial analysis with minimal coding in a jupyter environment. Journal of Open Source Software 6, 3414.

Xu, K., Chen, M., Yue, S., Zhang, F., Wang, J., Wen, Y., Lü, G., 2024a. The portal of opengms: Bridging the contributors and users of geographic simulation resources. Environmental Modelling & Software 180, 106142.

Xu, K., Yue, S., Chen, Q., Wang, J., Zhang, F., Wang, Y., Ma, P., Wen, Y., Chen, M., Lü, G., 2024b. Construction of an open knowledge framework for geoscientific models. Transactions in GIS 28, 154–175.

Yang, C., Yu, M., Hu, F., Jiang, Y., Li, Y., 2017. Utilizing cloud computing to address big geospatial data challenges. Computers, environment and urban systems 61, 120–128.

Zhang, C., Chen, M., Li, R., Fang, C., Lin, H., 2016. What's going on about geo-process modeling in virtual geographic environments (vges). Ecological Modelling 319, 147–154.

Zhang, F., Chen, M., Ames, D.P., Shen, C., Yue, S., Wen, Y., Lü, G., 2019. Design and development of a service-oriented wrapper system for sharing and reusing distributed geoanalysis models on the web. Environmental modelling & software 111, 498–509.

Zhang, F., Chen, M., Kettner, A.J., Ames, D.P., Harpham, Q., Yue, S., Wen, Y., Lü, G., 2021. Interoperability engine design for model sharing and reuse among openmi, bmi and opengms-is model standards. Environmental Modelling & Software 144, 105164.

Zhang, F., Chen, M., Yue, S., Wen, Y., Lü, G., Li, F., 2020. Service-oriented interface design for open distributed environmental simulations. Environmental Research 191, 110225.

Zhang, J., Clairmont, C., Que, X., Li, W., Chen, W., Li, C., Ma, X., 2025. Streamlining geoscience data analysis with an llm-driven workflow. Applied Computing and Geosciences 25, 100218.

Zhong, T., Zhang, Z., Chen, M., Zhang, K., Zhou, Z., Zhu, R., Wang, Y., Lü, G., Yan, J., 2021. A city-scale estimation of rooftop solar photovoltaic potential based on deep learning. Applied Energy 298, 117132.

Zhou, C., 2025. Exploring future gis visions in the era of the scientific and technological revolution. Information Geography 1, 100007.

Zhu, Z., Chen, M., Qian, Z., Li, H., Wu, K., Ma, Z., Wen, Y., Yue, S., Lü, G., 2023. Documentation strategy for facilitating the reproducibility of geo-simulation experiments. Environmental Modelling & Software 163, 105687.