PROCEEDINGS, 50th Workshop on Geothermal Reservoir Engineering
Stanford University, Stanford, California, February 10-12, 2025
SGP-TR-229

GeoCore: An Efficient and Scalable Framework to Optimize Geospatial Machine Learning

Ognjen Grujic[1,2], Thomas Hossler[3], Jacob Lipscomb[4], Rachel A. Morrison[5], and Connor M. Smith[6]

Zanskar Geothermal & Minerals Inc., USA

connor@zanskar.us

Keywords: Machine Learning, Geothermal, Play Fairway, Great Basin, Modeling, Geospatial, Exploration, INGENIOUS, Python, Codebase.

ABSTRACT

Machine learning is used extensively in many geospatial applications, including geothermal and mineral resource exploration. The scale on which these models are applied can be quite significant, especially when we consider the need for predictions on very fine grids and across large regions. Developing machine learning (ML) models requires many iterations, experimentation with various model parameters, and management of training and validation sets. In geospatial ML it is also important to consider the proper spatial separation between training, test, and validation sets, all of which significantly increase the computational requirements. In this work, we present an open-source library called GeoCore that enables efficient development of geospatial ML models with an H3 grid indexing of a wide range of resolutions. GeoCore enables modelers to work on top of any spatial database (PostgreSQL, Snowflake, BigQuery) with data indexed with H3 grid blocks, automatic feature caching, and MLflow for experiment tracking. GeoCore includes a dynamic registry system for utilizing various machine learning models, spatial cross-validation techniques, and utilities such as statistical fold plots and automatic buffering with user specified validation sets. Internally, we leverage GeoCore with public and proprietary geospatial data to produce large scale geothermal play fairway analysis models. We will demonstrate the entire modeling process with one of our models using public geospatial data from the INnovative Geothermal Exploration through Novel Investigations Of Undiscovered Systems (INGENIOUS) project.

1. INTRODUCTION

Geospatial machine learning (ML) is a rapidly evolving field, and over the last decade there has been a proliferation of studies (e.g., Gunderson et al., 2019, Okaroafor et al., 2021) with applications ranging from resource exploration to environmental monitoring. Despite this growth, many academic researchers face barriers in its practical application due to operational challenges and limited awareness of scalable solutions for geospatial ML. These barriers include:

1. Scalable storage and computation: Geospatial data can be massive and multidimensional, making storage and processing a bottleneck. Without workflows that mitigate this risk and scalable infrastructure, researchers often struggle with slow computations or data fragmentation across local systems.
2. Spatial indexing and analysis: The nature of geospatial data presents important considerations when applying ML techniques. Spatial operations, such as aggregations or distance calculations, can be computationally intensive. Furthermore, spatial correlation adds unique challenges to cross-validation and training/testing splits in machine learning.
3. Experiment tracking and reproducibility: Managing ML experiments requires tools that ensure reproducibility and systematic organization. As data sets and experiments grow in complexity, this becomes more challenging.
4. Interactive visualization: Geospatial model features can be multi-dimensional with complex spatial variability, and model outputs can be similarly complex. Gaining insight from ML models often involves time consuming map-based visualizations of features and model outputs. This slows the speed at which researchers can iterate on models and communicate their results.

GeoCore[7] is an open-source framework designed to address these challenges and streamline testing geospatial ML modeling. It leverages current best practices to enable reproducibility, efficiency, scalability, and flexibility in geospatial ML modeling.

---

[1] Authors are listed in alphabetical order with individual contributions cited in footnotes.

[2] Research, code implementation, and primary mentorship

[3] Research and code implementation

[4] Code adoption and code maintenance

[5] Code adoption, code maintenance, and secondary mentorship

[6] Corresponding author, primary manuscript author, code adoption, and code maintenance

[7] Zanskar Geothermal & Minerals. GeoCore. GitHub. https://github.com/Zanskar-Geothermal/GeoCore

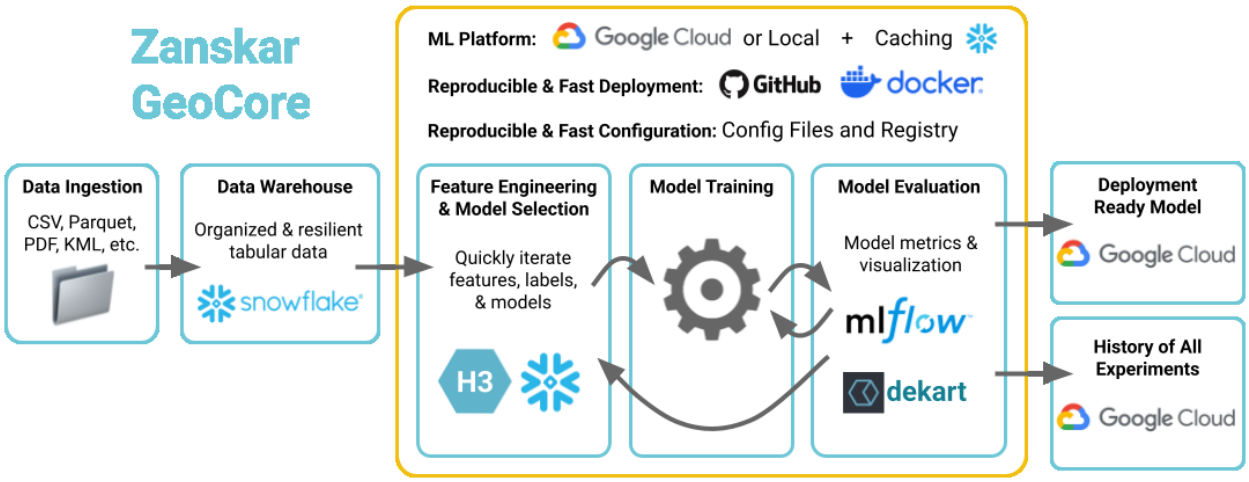

Figure 1: GeoCore overview. The figure illustrates an experiment workflow and the key components utilized for handling data preparation, modeling, visualization, and storage.

GeoCore provides a solution to these challenges by combining the following tools:

1. Scalable storage and computation: GeoCore relies on a SQL database/warehouse for data storage, feature engineering, and feature caching. Modern data warehouses like Snowflake enable cost-effective data storage and the provisioning of computational resources on demand making them very attractive for high resolution geospatial ML applications. GeoCore is deployed with a Docker container configuration and integrated with a geospatial database (such as Snowflake). The backbone of GeoCore is its dependency management via poetry, and Docker runtime environment, which enables seamless cross platform model development/training. Both local as well as remote model training is possible with GeoCore depending on the computational requirements at hand. Because GeoCore delegates feature engineering—the most computationally expensive aspect of geospatial ML model development—to a data warehouse, it is often feasible to train geospatial ML models locally on a modern laptop. (At Zanskar, we ran high resolution geospatial models with ~4M grid blocks, on M1-max machines).

2. Spatial indexing and analysis: GeoCore harnesses Uber's H3 (Uber, 2018) geospatial indexing system to systematically integrate a wide variety of geospatial data types, while geospatial functions available in Snowflake streamline complex data transformations. A variety of spatial cross-validation approaches are implemented to handle testing and training with spatial correlation. GeoCore's reliance on Uber's grid is critical since it enables efficient neighborhood aggregation and searching.

3. Experiment tracking and reproducibility: A modeling component registry system (fvcore) that maps names to objects provides a flexible framework for structuring code hierarchies and managing ML experiment design. This is particularly useful for large-scale or complex ML workflows. Meanwhile, the MLflow platform (Zaharia et al., 2018) facilitates experiment tracking, metadata organization, model management, and deployment, making it easier to ensure reproducibility and streamline workflows.

4. Interactive visualization: Uber's H3 grid is integrated into mainstream open-source geospatial data visualization libraries like the open-source platform Dekart (Bilonenko, V., 2024). Dekart seamlessly connects with big data warehouses enabling quick visualization of H3 grid indexed data. This is particularly useful in the iterative geospatial ML model development and feature engineering. Namely, GeoCore defines all ML features using SQL and indexes them on the H3 grid. By having Dekart in the mix we are able to prototype and visualize all our features in a scalable manner before we feed them into our feature store. Moreover, GeoCore ML model output is also indexed by the H3 grid, enabling us to plot the final model predictions alongside the generated (and cached) features and raw data, hence significantly reducing the model iteration process.

In this paper, we detail the structure of GeoCore, then we showcase a play fairway analysis (PFA) study to demonstrate its application. PFA is a method commonly used in geothermal exploration to assess and rank areas based on their resource potential. PFA integrates diverse datasets, such as geological, geophysical, and geochemical information, into geospatial models to identify regions with the highest likelihood of geothermal resources (e.g., Faulds et al., 2015a, 2015b; Shervais et al., 2016; Forson et al., 2016; Lautze et al., 2017; Siler et al., 2017; Wannamaker et al., 2017; Craig et al., 2021; Smith et al., 2023; Hart-Wagoner et al., 2024).

2. GEOCORE CODEBASE

Managing experiment configurations is a key challenge in most ML workflows. The GeoCore codebase is designed as a framework system for tracking data, models, experiment designs, geospatial tools, plots, and other common geospatial ML features. It also offers a structured and hierarchical design that optimizes ML workflows, promotes best practices, and ensures consistency. The hierarchical

structure minimizes duplication, defines parameters consistently, and supports automated workflows to enhance modularity and traceability.

GeoCore is leveraging a modular design with component registration via Meta's fvcore library (Meta AI Research, n.d.). fvcore enables us to write every component (e.g.,. cross validator, model, feature set) independently from the rest of the code, and then reference the new component in a yaml config file. The library automatically performs parameter validation without the need for extra coding, and it seamlessly integrates with the rest of the components defined in the config file workflow. GeoCore's software architecture is split into several modules, each managed by a separate fvcore registry.

Within a central directory called **modeling** are key subdirectories, including:

- **actions:** Functions for training and applying models
- **config:** Default ML experiment configuration parameters are set in this centralized location
- **cross_validators:** Classes implementing custom cross-validation
- **datasets:** Classes defining custom features and labels
- **models:** Classes implementing various supervised and unsupervised ML models

The directories cross_validators, datasets, and models include:

- **Base File:** A single base.py file which contains an abstract base class to reduce redundant code
- **Class Files:** Classes defining different cross validators, datasets, and models. Each class is in a separate file. This enables modularity and flexibility, as each class can contain off the shelf tools or can be customized.

Adjacent to the modeling directory is the **experiment_configs** directory, where parameter settings for individual experiments can be set. These parameters define which model features, labels, model types, and cross-validations schemes will be used in model training and application. The user can then easily iterate on various datasets, models, and cross-validation schemes by simply updating or creating a new experiment config file. Underpinning the entire structure is the registry, which uses Meta's CfgNode class to share config parameters throughout the framework.

This structure allows for easy implementation and flexible tuning of any data type, model architecture, cross validator, dataset, and other elements one might wish to incorporate. For example, if a user wants to compare a random forest model to an XGBoost model, the user can set up a class for each model within the modeling/model directory and register it with fvcore. Similarly, if we want to experiment with a new set of features, we define a sql code that computes the feature on the h3 grid and register the new feature with fvcore**.** Then it is a simple matter of running the experiment once with the first model, updating the experiment config to point to the new model, and re-running to compare the two approaches.

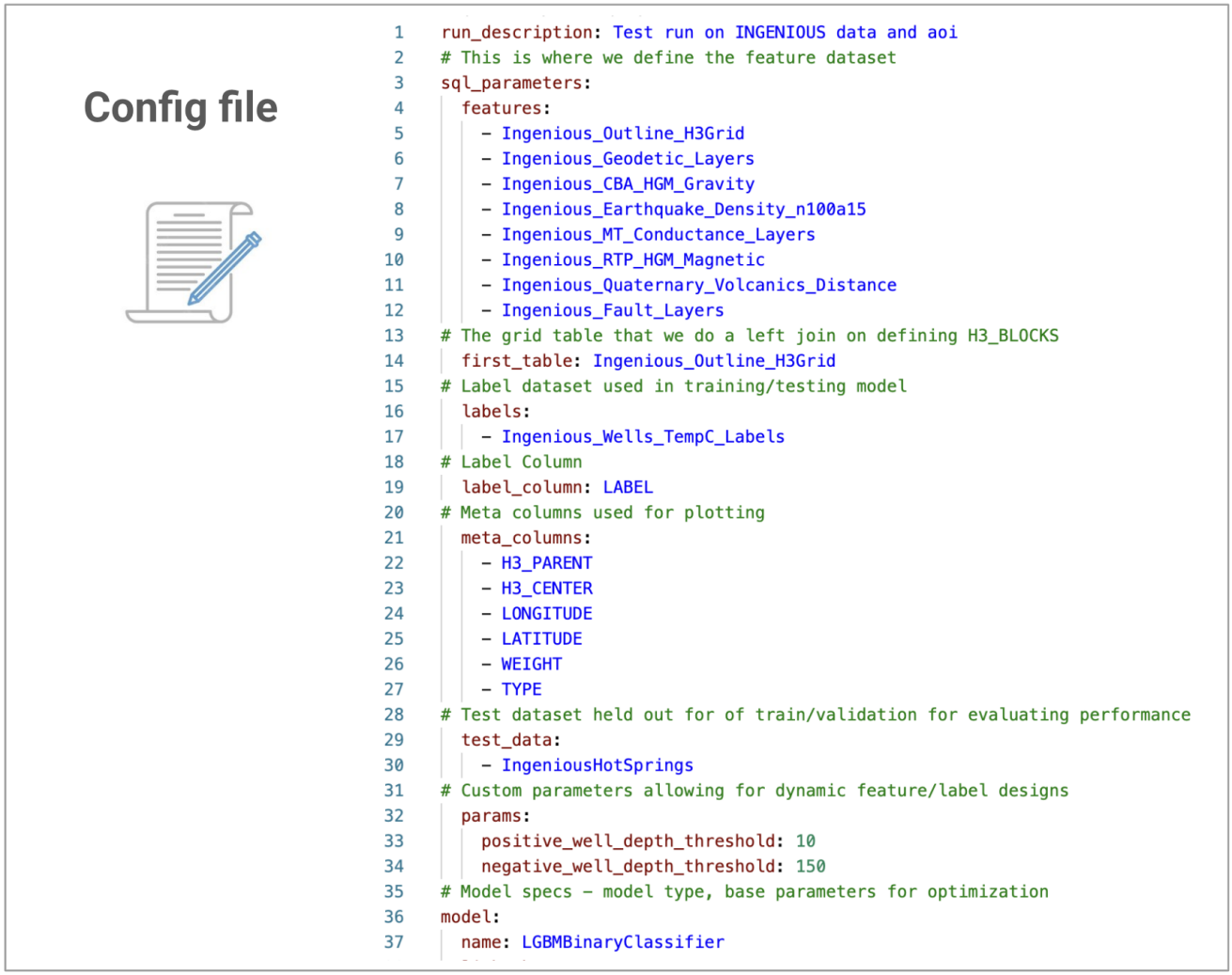

Figure 2: Example portion of a YAML file for experiment design. The full file specifies the experiment design for features, labels, parameters, models, and cross-validation.

2.1 Experiment Design

Experiment configuration files in YAML format control data and model inputs for experiments run with the codebase. YAML files are a great way to store and manage configuration settings for machine learning models. They help manage paths, model parameters, and other configurations, making it easier to experiment with different configurations and maintain code reusability. All experiments are based on the design of these files. This design enhances reusability and simplifies experimentation with various configurations. An example of the configuration for the modeling is shown in Fig. 2.

In the study described below, we utilized a LightGBM decision tree model. LightGBM, a gradient boosting framework based on decision trees, is well-suited for geospatial ML due to its ability to handle large datasets, model non-linear relationships, and provide interpretable outputs through feature importance metrics. Details into the design are found in the codebase modeling directory. Data and cross-validation choices are outlined below and further explained in the codebase.

Running experiments with these configurations is streamlined using command-line decorators and custom flags, leveraging the flexibility of Poetry. This approach mitigates common problems like dependency drift and environment mismanagement, providing a reliable foundation for ML workflows. The bash command:

poetry run train -c <PATH_TO_CONFIG> -e <EXPERIMENT_NAME> (1)

initiates the following workflow as illustrated in Fig. 1:

1. **Data Retrieval:** Fetches data and, if the features are not already pre-computed and cached locally, runs feature engineering using Snowflake.
2. **Experiment Logging:** Creates and logs an experiment instance in MLflow, enabling traceability and reproducibility.

3. **Model Training:** Conducts training using cross-validation and hyperparameter optimization to fine-tune model performance.
4. **Application Set Inference:** Performs inference on application datasets not included in training, validation, or testing.
5. **Test Set Inference:** Evaluates model performance using predefined test datasets, such as known geothermal systems.
6. **Result Logging:** Logs all metrics, models, and outputs, storing them locally and/or in cloud repositories like Google Cloud or Snowflake.

This workflow automates key steps, reducing manual intervention while ensuring consistency across experiments. Metrics, datasets, and trained models are easily accessible through MLflow and Dekart, providing transparency and facilitating collaborative analysis.

2.2 Best Practices

The codebase is designed to enable rapid experimentation and iteration, even with large datasets. With the dataset utilized in this study (discussed below), within minutes, the workflow can retrieve data, generate features, define training and test splits, calculate distance matrices, optimize a model, and produce results. To maximize efficiency in both performance and collaboration, the following practices are integral:

- **Configuration Management:** Centralized control of hierarchical parameters, inspired by YACS, ensures modularity and reusability across workflows.
- **Dependency Management:** Poetry ensures efficient and conflict-free handling of dependencies and virtual environments.
- **Reproducibility and Scalability:** Docker packages the codebase into portable containers, with CPU or GPU-enabled support for large datasets and computationally intensive tasks.
- **Git and CI/CD Integration:** The codebase can work with streamlined Git workflows and GitHub Actions to automate testing and deployment within containerized environments, ensuring reproducibility and reducing development time.
- **Scalable Data Management and Experiment Tracking:** Leveraging efficient data caching and Snowflake for computational scalability and integrating MLflow ensures smooth data retrieval, storage, and workflow automation.

By combining these best practices with a hierarchical and modular design we provide flexibility, scalability, and an emphasis on reproducibility in geospatial ML workflows.

2.3 Data Processing and Storage

Effective geospatial data processing for ML efforts commonly requires proper indexing of diverse datasets, such as points, lines, polygons, and rasters to a grid format. A cornerstone of GeoCore is the use of H3 spatial indexing, which provides a hierarchical, hexagonal grid system. The H3 spatial indexing is a global grid system that exists at multiple resolutions (e.g., from coarse scales like resolution 1 with 122 cells worldwide to finer scales such as resolution 15 with billions of cells). Unlike geometries, spatial indexes are referenced by a short ID string. This makes them far smaller to store and faster to process (e.g., Li et al., 2020). These strings have nested resolutions, enabling seamless multi-scale analysis. Furthermore, the hexagonal system is more convenient for any spatial transformations because cells have uniform distance between each neighbor unlike square grid cells.

Once point, line, polygon, and raster data are indexed into H3 indexes, they can be easily augmented with feature engineering and joined with other datasets on the H3 index. In our workflow to streamline data processing, we often leverage Snowflake's native geospatial functions, many of which support H3 indexing and geospatial transformations including calculating intersections, distances, neighborhood-level features, etc. In our analysis with the INGENIOUS dataset, we demonstrate several preprocessing steps, and grid sampling examples to index data onto H3 cells.

The storage and data management infrastructure are designed to support scalable and efficient workflows. Data storage is organized across three key components:

- **Snowflake** acts as the primary data repository, compressing data and enabling seamless two-way access to tables. Snowflake supports rapid iteration during feature engineering while its scalable compute warehouses efficiently handle spatial operations such as joins and intersections. These operations take approximately 10–30 seconds per million rows, enabling quick processing of large datasets.
- **Google Cloud** provides scalable storage for raw datasets, intermediate processing outputs, and results. Buckets store tables, plots, model artifacts, and experiment configurations, enabling easy organization and retrieval. Tools like Dekart and MLflow can be hosted on Google Cloud, facilitating collaborative map visualizations and experiment tracking.
- **Local Caching and hosting** accelerate experiments by storing frequently used feature sets and application datasets locally. This minimizes redundant table-building steps during iterative experiments, saving time and computational resources. Additionally, local connections to Dekart and MLflow allow for flexibility when working in offline or constrained environments, ensuring workflows remain accessible and efficient regardless of infrastructure constraints.

This infrastructure supports flexible workflows that can scale based on the size and complexity of the exploration region, ensuring seamless performance regardless of the data volume. Shared cloud resources enable team members to collaborate on model development, visualization, and analysis in a controlled and secure environment.

2.4 User Interface

Visualization plays a crucial role in interpreting the results of ML models in geothermal exploration. Without proper channels for result interface users may experience issues with loss of version details, difficulty in comparing results, and challenges in collaborating and improving on past efforts. To evaluate our data, we focus on two main aspects: a map viewer for visualizing play fairway models and an experiment log for tracking data, plots, metrics, and experiment designs.

With MLflow we can log model parameters, artifacts, plots, and models, keeping our experiment workflow organized and traceable. MLflow plays a critical role in tracking the life cycle of modeling efforts. Some things we log for in each experiment include the experiment configuration YAML, artifacts of plots and metrics, the saved model, tables of results, and maps (Dekart). This approach allows our team to audit the entire modeling process, identify and compare the performance of models, and understand the impact of hyperparameter tuning on model performance (Fig. 3). We can also host profile files (utilizing the cProfile python library) to track and visualize function performance of any actions we run to identify bottlenecks and guide optimization.

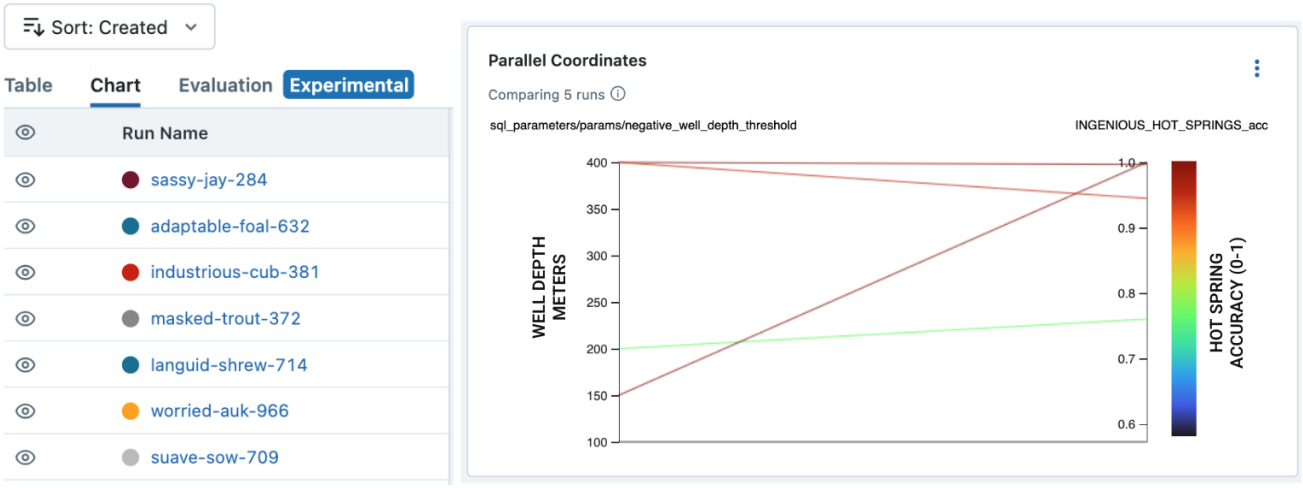

Figure 3: Example from the MLflow interface comparing experiment runs with different metrics. The parameters being adjusted in this instance is the cut off for negative sites well depth (meters) for each experiment (represented as lines on the plot) versus the accuracy of a model in predicting hot springs.

Visualizing the maps of our results, training and test data plots, each feature, etc., we can leverage the integration of our H3 grid data with Uber's open-source viewer kepler.gl through the Dekart interface. Dekart is connected directly to our Snowflake data storage and allows us to create maps from SQL queries and share them instantly and securely with live updates and multi-user editing. Another convenient aspect is the built-in compatibility with viewing H3 layers, allowing for quick and efficient rendering of relatively large datasets.

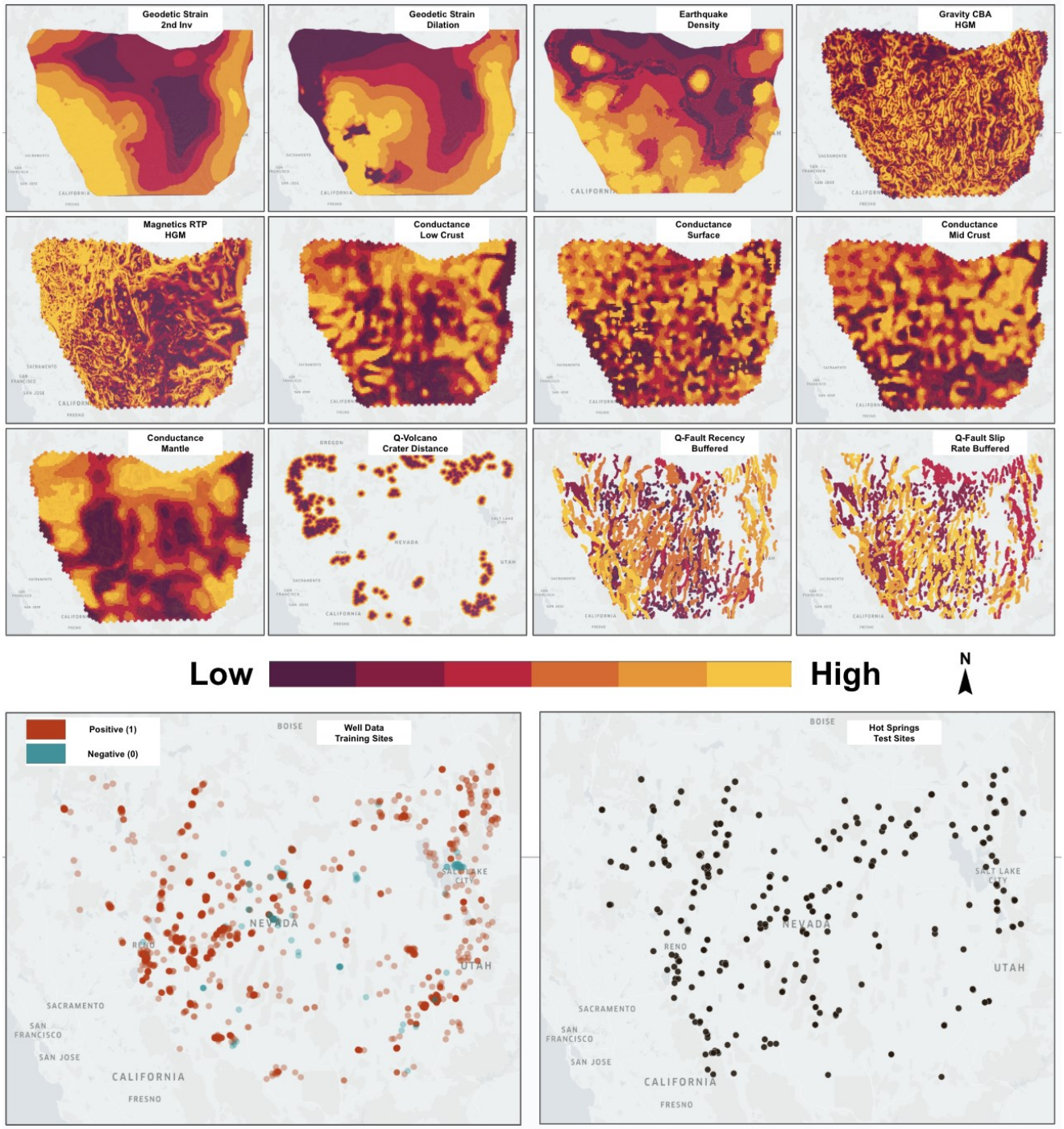

FIGURE 4: DEKART MAPS OF DATASETS (SAMPLED TO H3) USED IN STUDY. SOME OF WHICH TRANSFERRED DIRECTLY FROM RASTERS TO THE H3 GRID. SOME TRANSFORMED FROM POINT OR LINE DATA INTO H3 CELLS USING SNOWFLAKE. TEXT BELOW PROVIDES A REFERENCE TO EACH DATASET DISPLAYED. DATA IS READILY AVAILABLE ON THE GEOTHERMAL DATA REPOSITORY (AYLING ET AL., 2023).

3. CASE STUDY PFA WITH INGENIOUS DATA

We chose to utilize geospatial data from the INGENIOUS project (Ayling et al., 2023) to perform a PFA study predicting geothermal favorability in the Great Basin region of the western United States. The INGENIOUS study within the Great Basin region has curated publicly available datasets that are well-suited for machine learning efforts, making it an ideal dataset to demonstrate the application of GeoCore.

3.1 INGENIOUS Data Preparation

In this analysis we selected and organized a subset of the INGENIOUS data into a feature, label, and test data set (Fig. 4), using many of the same features identified by Hart-Wagoner et al., 2024, through weights of evidence and logistic regression analysis, based on their relevance to known geothermal sites. The data was sampled using an H3 grid resolution of 8, which has an average edge length of 0.531 km. In total we utilized 12 feature sets and a full grid size of 679,826 cells.

During the data preparation stage, we leveraged Snowflake's native functions to transform features from point and line data into continuous spatial variables. We utilized the H3_GRID_DISK to perform k-ring distance transformations with faults and quaternary volcanic centers where a k-ring distance of 1 equates to ~1 km. Furthermore, YAML-based configurations enabled dynamic customization of parameters (e.g., distance thresholds, depth cutoffs) for each feature in YAML experiment files. For raster datasets, QGIS was used to sample values and join them onto the H3 grid. Point and line geometries were converted to H3 cells using Snowflake tools such as H3_COVERAGE_STRINGS and H3_LATLNG_TO_CELL_STRING, ensuring spatial alignment and consistency across datasets. The entire process, including data retrieval, feature transformations, and joining datasets onto an H3 grid, was completed in a matter of minutes. This efficiency allows for rapid iteration and experimentation of different parameters and features, improving not only the speed of model development but also the quality of the model itself. A breakdown of the features, labels, and test data design is shown below.

Features:

- Geodetic strain (2nd inverse and dilation)
- Earthquake density (N=100, α=0.15, see Hart-Wagoner et al., 2024)
- Gravity (complete Bouguer anomaly - horizontal gradient magnitude)
- Magnetics (reduced to pole - horizontal gradient magnitude)
- Conductance (low-crust, surface, mid-crust, mantle)
- Quaternary Volcanic Center Distance (20 km distance)
- Quaternary Fault Recency (10 km distance)
- Quaternary Fault Slip Rate (10 km distance)

Labels:

- Positive Labels: Wells with a thermal class of 'Hot' or 'Warm' **AND** a depth ≥ 10 m (6009 wells)
- Negative Labels: Wells with a thermal class of 'Cold' AND a depth ≥ 400 m (148 wells)

Test Data:

- 309 Hot springs locations with a temperature ≥ 40 °C

To utilize the training data in this study we performed a cross-validation split into a training and validation set. Cross-validation is a critical and commonly overlooked aspect of geospatial ML efforts. Cross-validation addresses the risks of bias from spatial autocorrelation and heterogeneity, which are prevalent in geospatial datasets. It also supports optimizing a model to account for spatial variance and provides valuable spatial context for model performance (e.g., performance in different regions).

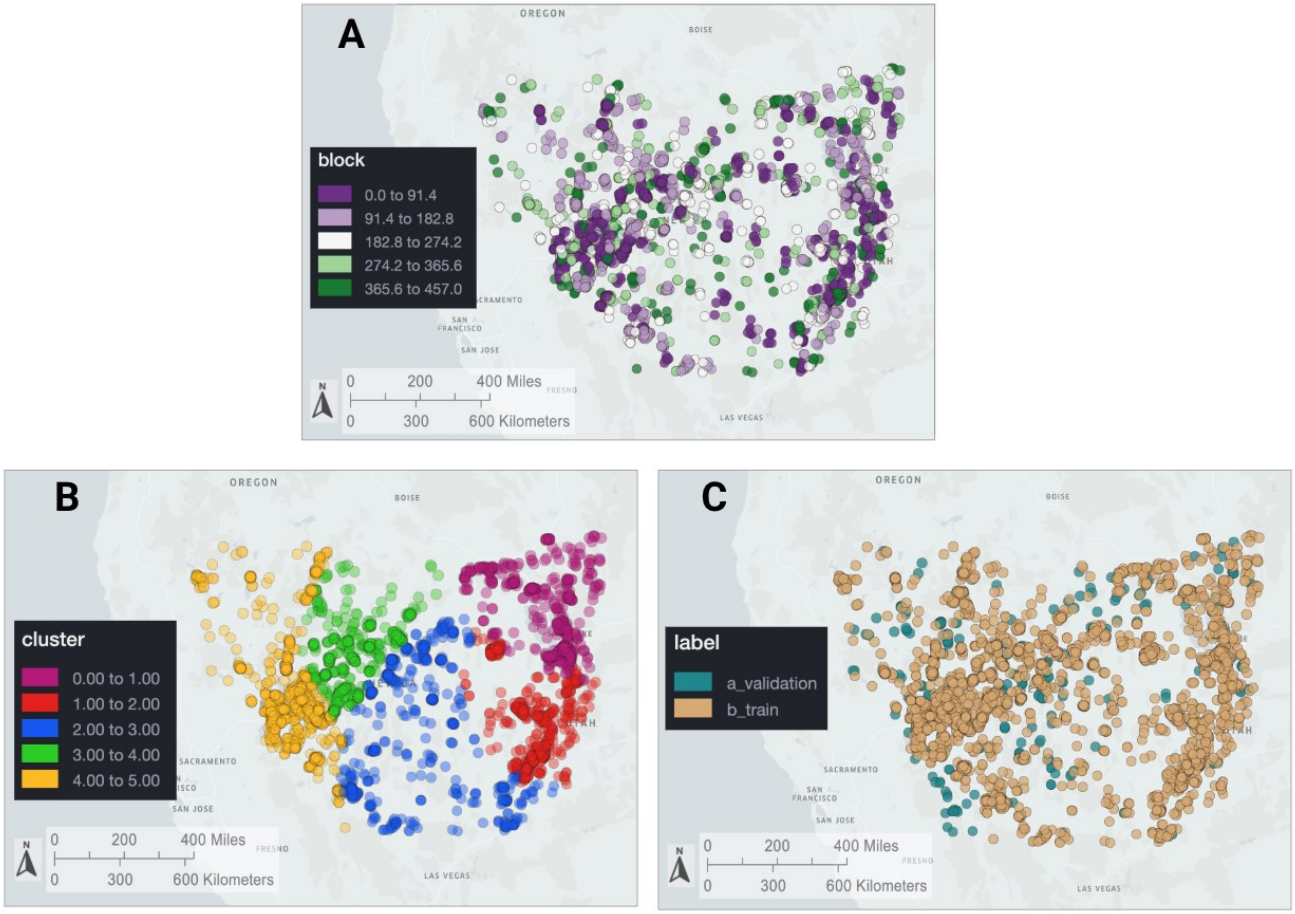

Figure 5: Spatial partitioning of training data using BlockCV. The top panel (A) shows points grouped into blocks using Agglomerative Clustering with a clustering distance of 15 km. Each block (color-coded on a purple to green scale) represents a spatially cohesive grouping of nearby points. The block IDs are unique identifiers for these spatial groups, which serve as the building blocks for creating clusters. The bottom left panel (B) shows the final 5 clusters formed by combining nearby blocks into a balanced number of points in each fold for cross-validation. The final splitting of well validation and training data are shown in the bottom panel (C).

The GeoCore library offers several useful classes of cross-validation methods common for geospatial modeling. Here, we demonstrate the use of BlockCV, a clustering-based cross-validation method based on Spatial+ cross-validation (Wang, W. et al., 2023), to maintain spatial independence between training and validation datasets. This ensures a realistic estimate of model performance by reducing overfitting. The BlockCV method involves:

- **Block Formation**: Data points were grouped into spatially cohesive blocks using Agglomerative Clustering with a clustering distance of 15 km (Fig. 5A). Each block represents nearby points and serves as the foundation for forming clusters.
- **Cluster Formation**: Blocks were merged into larger, spatially balanced clusters to form cross-validation folds (Fig. 5B). This process balanced fold sizes while maintaining spatial separation.

This methodology mitigates data leakage risks while aligning with the spatial nature of the dataset, ensuring validation results better reflect real-world scenarios.

3.2 Modeling Results

The results of the PFA are presented in map (Fig. 6) form along with select plots output through the MLflow workflow. The application set results map includes prediction scores and hot spring points used for testing. The workflow with test data ensures that no training points overlap the same cells as these test points, so any prediction applied becomes a ground truth test of performance. In this example, the model achieved a 94.5% success rate in predicting hot springs within the top 50th percentile of geothermal favorability scores. In terms of spatial patterns in the predictions, high favorability scores were observed in western Nevada and along Utah's Wasatch Front, regions that host many operating geothermal power plants. In contrast, areas in eastern Nevada, southwestern Idaho, and far western Utah exhibited relatively lower favorability scores, which are regions that host much fewer operating geothermal power plants. Shapley

results (Fig. 7) indicate which features were used to support model predictions. Some of the key features from Shapley analysis with the LightGBM classification model include the lower crust and mantle conductance, fault slip rate, geodetic strain, and earthquake density. These each have a magnitude greater than 0.5 indicating a notable contribution to the model's prediction of positive and negative samples during the training stage.

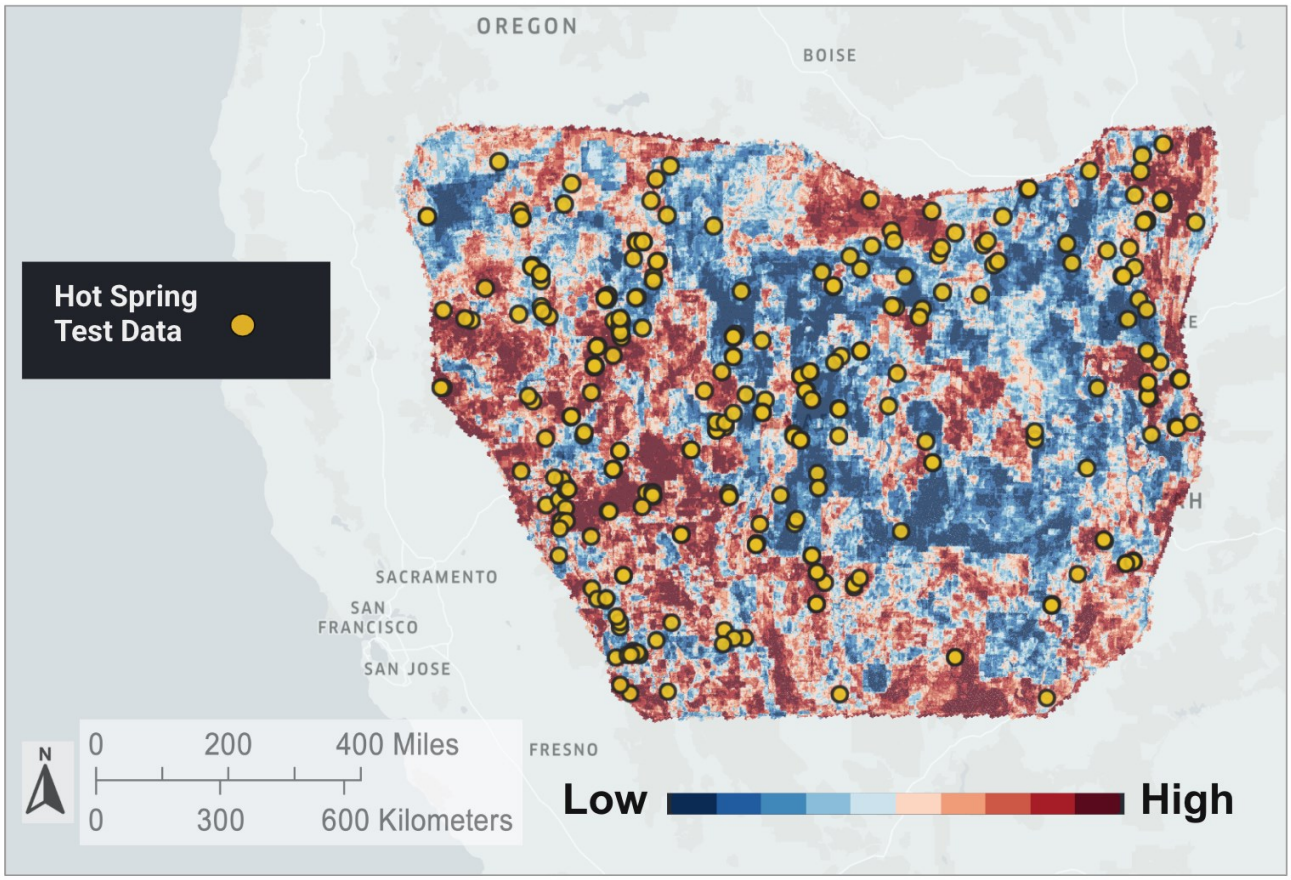

Figure 6: Dekart PFA prediction map using INGENIOUS data.

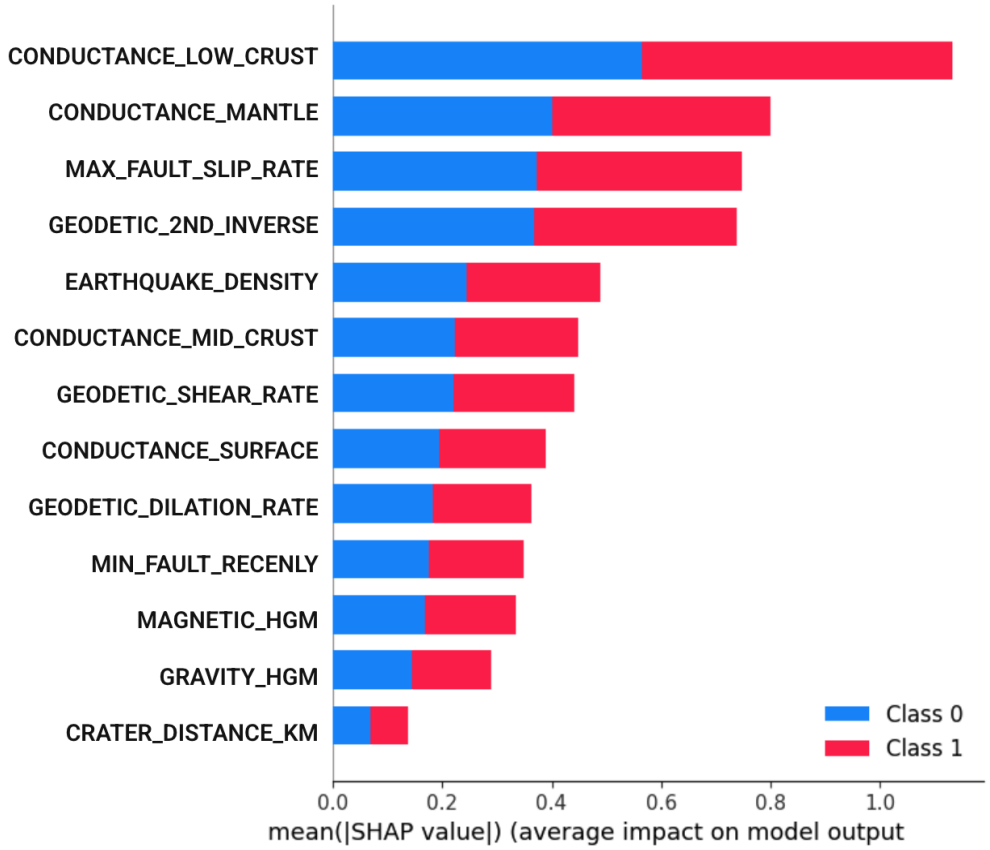

Figure 7: Shapley feature importance plot logged in MLflow from the LightGBM classifier. Bar widths depict feature importances for classes 0 and 1. The sum of colored bar widths indicates overall feature importance for the model based on average Shapley additive explanations magnitude across all training data point locations.

Despite strong performance in predicting hot springs, the model exhibited variability across validation folds (Fig. 8), particularly in folds 1 and 5 (yellow and purple clusters in Fig. 5B). These folds demonstrated lower predictive performance, likely due to sparse data or unique geological characteristics within those regions. Furthermore, the imbalanced distribution of positive and negative training samples across all folds added complexity to the modeling process. Given the challenges of characterizing 'negative' geothermal favorability, exploring alternative labeling approaches or augmenting datasets with additional negative examples may enhance model performance.

Advanced cross-validation techniques, such as BlockCV, ensured spatial independence between training and validation datasets, mitigating overfitting and enhancing generalizability. However, the variability in fold-specific performance underscores the need for continued refinement. Incorporating domain-specific insights and leveraging additional geological datasets could address fold-specific challenges and improve the model's robustness. The results underscore the importance of rigorous validation, balanced datasets, and informed feature design in advancing geospatial machine learning for geothermal exploration. GeoCore's flexibility also supports regression modeling, enabling continuous predictions of geothermal potential rather than binary classification. This capability opens avenues for more nuanced and granular analyses, particularly in resource assessment and exploration prioritization. By refining training datasets and adopting alternative validation approaches, future iterations of this model are expected to deliver significant performance improvements.

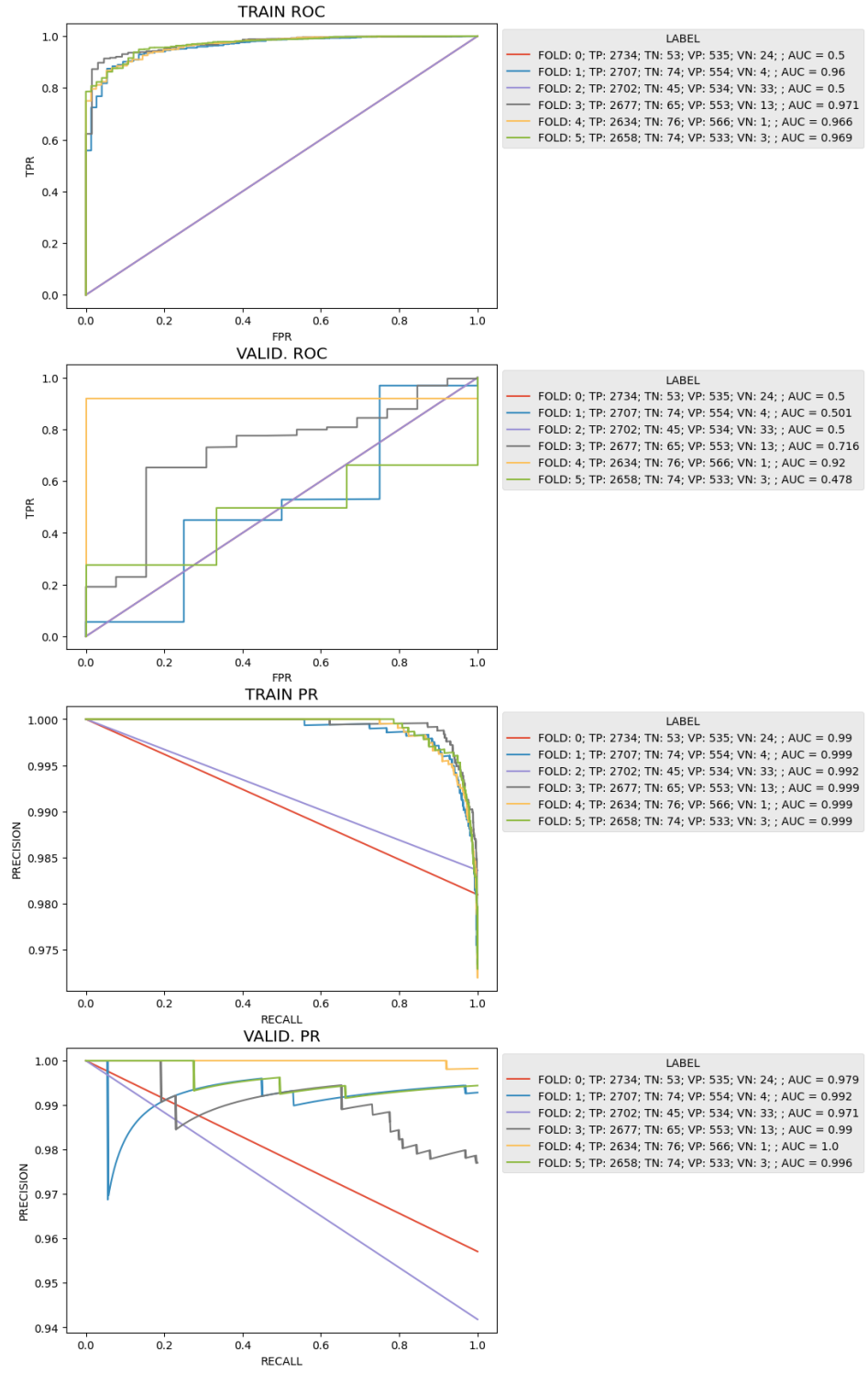

Figure 7: Example results saved to MLflow of ROC curves and precision/recall curves (ranging from zero to one). Both metrics are often summarized by its integral, the area under the curve (AUC). The closer the AUC is to one, the better the predictive performance of the model with test data (e.g., Davis, 2006). The ROC considers both the true positive rate and false positive rate according to different probability threshold values. Precision is the proportion of true positives among predicted positives, while recall is the proportion of true positives among actual positives.

4. CONCLUSION

This study demonstrates the utility of the GeoCore framework in addressing key challenges inherent to geospatial machine learning, such as spatial autocorrelation, experiment tracking, computational scalability, and efficient data processing. By leveraging modular design principles and integrating best practices, GeoCore provides a robust foundation for scalable, reproducible, and efficient workflows.

Using publicly available geothermal data, we highlighted how GeoCore facilitates seamless integration of diverse geospatial datasets using H3 spatial indexing, ensures rigorous spatial validation through advanced cross-validation techniques, and enables efficient model development via experiment tracking and automated workflows. The framework's flexibility not only streamlined the modeling process but also allowed us to generate actionable insights.

Geocore's scalability and versatility make it well-suited for broader geospatial applications, including mineral exploration, environmental modeling, and hazard assessment. Its emphasis on modularity, reproducibility, and domain-specific adaptability ensures that researchers and practitioners can tailor the framework to their unique needs. While variability in validation performance underscores the complexity of geospatial ML, the insights from this study provide a pathway for refining models and improving their robustness.

REFERENCES

Ayling, B.F., Faulds, J., Morales, R.A., Koehler, R., Kreemer, C., Mlawsky, E., Coolbaugh, M., Micander, R., dePolo, C., Kraal, K., Wagoner, N., Siler, D., DeAngelo, J., Glen, J., Peacock, J., Batir, J., Gentry, E., Berti, C., Lifton, Z., Clark, A., Kirby, S., Hardwick, C., & Kleber, E. (2023). INGENIOUS - Great Basin Regional Dataset Compilation. United States. https://dx.doi.org/10.15121/1881483

Bilonenko, V. (2024). Dekart: Open-source backend for Kepler.gl [Computer software]. https://dekart.xyz/

Craig, J.W., Faulds, J.E., Hinz, N.H., Earney, T.E., Schermerhorn, W.D., Siler, D.L., Glen, J.M., Peacock, J., Coolbaugh, M.F., & Deoreo, S.B. (2021). Discovery and analysis of a blind geothermal system in southeastern Gabbs Valley, western Nevada, USA. *Geothermics, 97*, 18. https://doi.org/10.1016/j.geothermics.2021.102177

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning*, 233–240.

Faulds, J.E., Hinz, N.H., Coolbaugh, M.F., Shevenell, L.A., Siler, D.L., dePolo, C.M., Hammond, W.C., Kreemer, C., Oppliger, G., Wannamaker, P., Queen, J.H., & Visser, C. (2015a). Integrated geologic and geophysical approach for establishing geothermal play fairways and discovering blind geothermal systems in the Great Basin region, western USA. *Final submitted report to the Department of Energy (DE-EE0006731)*, 106.

Faulds, J.E., Hinz, N.H., Coolbaugh, M.F., Shevenell, L.A., Siler, D.L., dePolo, C.M., Hammond, W.C., Kreemer, C., Oppliger, G., Wannamaker, P.E., Queen, J.H., & Visser, C.F. (2015b). Integrated geologic and geophysical approach for establishing geothermal play fairways and discovering blind geothermal systems in the Great Basin region, western USA. *Geothermal Resources Council Transactions, 39*, 691–700.

Forson, C., Czajkowski, J.L., Norman, D.K., Swyer, M.W., Cladouhos, T.T., & Davatzes, N. (2016). Summary of phase 1 and plans for phase 2 of the Washington state geothermal play fairway analysis. *Geothermal Resources Council Transactions, 40*, 541–550.

Google. (n.d.). Google Cloud. *Google Cloud Documentation*. https://cloud.google.com/

Gunderson, K.L., Holmes, R.C., & Loisel, J. (2019). Recent digital technology trends in geoscience teaching and practice. *GSA Today, 30*(1), 39–41. https://rock.geosociety.org/net/gsatoday/groundwork/G404GW/GSATG404GW.pdf

Hart-Wagoner, M. (2024). Preliminary regional play fairway workflow for the Great Basin Region, USA. In *Proceedings, 49th Workshop on Geothermal Reservoir Engineering*, Stanford University, Stanford, California, February 12–14, 2024, SGP-TR-227.

Lautze, N., Thomas, D., Hinz, N., Apuzen-Ito, G., Frazer, N., & Waller, D. (2017). Play fairway analysis of geothermal resources across the State of Hawaii. *Geothermics, 70*, 376–392.

Li, M., & Stefanakis, E. (2020). Geospatial operations of discrete global grid systems—a comparison with traditional GIS. *Journal of Geovisualization and Spatial Analysis, 4*(2), 26. https://doi.org/10.1007/s41651-020-00066-3

Meta AI Research. (n.d.). fvcore library. GitHub. https://github.com/facebookresearch/fvcore

Okoroafor, E.R., Smith, C.M., Ochie, K.I., Nwosu, C.J., Gudmundsdottir, H., & Aljubran, M.J. (2022). Machine learning in subsurface geothermal energy: Two decades in review. *Geothermics*, 102, 102401.

Shervais, J.S., Glen, J.M., Nielson, D., Garg, S., Dobson, P., Gasperikova, E., Sonnenthal, E., Visser, C., Liberty, L.M., DeAngelo, J., Siler, D., Varriale, J., & Evans, J.P. (2016). Geothermal play fairway analysis of the Snake River Plain Phase 1. In *Proceedings, 41st Workshop on Geothermal Reservoir Engineering*, Stanford University, 7. SGP-TR-209.

Siler, D.L., Zhang, Y., Spycher, N.F., Dobson, P.F., McClain, J.S., & Gasperikova, E. (2017). Play-fairway analysis for geothermal resources and exploration risk in the Modoc Plateau region. *Geothermics, 69*, 15–33.

Smith, C.M., Faulds, J., Brown, S.R., Coolbaugh, M., DeAngelo, J., Glen, J.M., Burns, E.R., Siler, D.L., Treitel, S., Mlawsky, E., Fehler, M.C., Gu, C., & Ayling, B. (2023). Exploratory analysis of machine learning techniques in the Nevada geothermal play fairway analysis. *Geothermics*. https://doi.org/10.1016/j.geothermics.2023.102693

Snowflake Inc. (n.d.). Snowflake Platform. *Snowflake Documentation*. https://www.snowflake.com/

Uber. (2018). H3: Uber's hexagonal hierarchical spatial index. *Uber Engineering Blog*. https://www.uber.com/blog/h3/

Wang, W., Khodadadzadeh, M., & Zurita-Milla, R. (2023). Spatial+: A new cross-validation method to evaluate geospatial machine learning models. *International Journal of Applied Earth Observation and Geoinformation*. https://doi.org/10.1016/j.jag.2023.103364

Wannamaker, P.E., Moore, J.N., Pankow, K.L., Simmons, S.F., Nash, G.D., Maris, V., Trow, A., & Hardwick, C.L. (2017). Phase II of play fairway analysis for the eastern Great Basin extensional regime, Utah: Status of indications. *Geothermal Resources Council Transactions, 41*, 2368–2382.

Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, M., Konwinski, A., Murchie, P., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., & Xin, R. (2018). Accelerating the machine learning lifecycle with MLflow. *Databricks Blog*. https://databricks.com/mlflow