GeoGenIE: Geographic Genetic Inference Engine

Geolocation predicting from SNPs using deep learning

Drs. Bradley T. Martin, Tyler K. Chafin, Zach D. Zbinden, Marlis R. Douglas, and Michael E. Douglas

Version: 1.0.0

2024-06-05

Contents

List of Figures

GeoGenIE: Geolocation Predictions from SNPs using Deep Learning

Introduction

GeoGenIE (Geographic-Genetic Inference Engine) is a comprehensive software tool designed to predict geographic coordinates (longitude and latitude) from genetic SNP data. GeoGenIE utilizes deep learning models and offers several advanced features such as automated model parameter searches (via Optuna (Akiba et al. 2019)), outlier detection to remove translocated individuals (based on the GGOutlieR method (Chang et al. 2023)), handling of imbalanced sampling using a custom oversampling algorithm adapted from SMOTE (Lemaître, Nogueira, and Aridas 2017), and weighting the loss function by inverse sampling densities. The software is user-friendly and provides extensive visualizations and metrics for evaluating predictions, making it a robust and accurate solution for geographic-genetic inference.

GeoGenIE is designed for researchers in population genetics and molecular ecology, providing a powerful tool to infer geographic origins of individuals based on their genetic data. The software employs state-of-the-art deep learning techniques to handle complex genetic data and generate precise geographic predictions. However, researchers often face challenges due to imbalanced sampling, where certain geographic regions are overrepresented while others are underrepresented in the data. GeoGenIE addresses these challenges through several innovative algorithms.

GeoGenIE incorporates an outlier detection algorithm adapted from GGOutlieR to identify and remove individuals that have been translocated. This is crucial for ensuring the accuracy

of geographic predictions by eliminating samples that could introduce bias. Additionally, GeoGenIE implements a weighted loss function using PyTorch, where inverse sample weights are used to focus the loss function more heavily on areas with lower sample densities. This approach helps in balancing the influence of samples from different regions, improving the robustness of the model.

Furthermore, GeoGenIE employs a regression-based synthetic oversampling method adapted from SMOTE. This method uses a genotype interpolation algorithm based on Mendelian inheritance to generate synthetic samples in underrepresented regions, thereby balancing the dataset. These advanced algorithms collectively enable GeoGenIE to provide reliable geographic predictions even in the presence of sampling imbalances.

Deep Learning Model Architecture

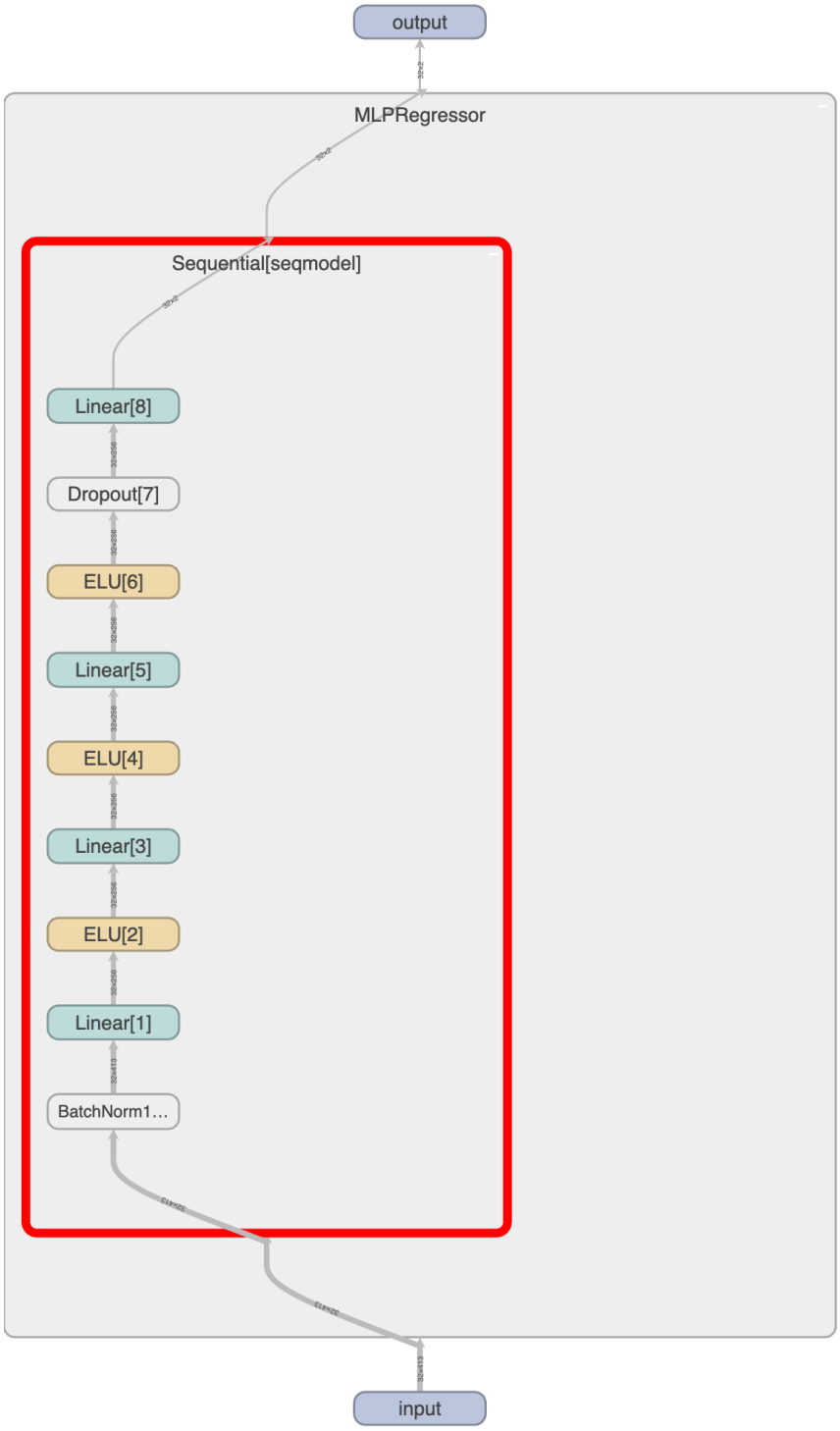

Figure 1: GeoGenIE model architecture diagram.

GeoGenIE was written in PyTorch. Below is the deep learning model architecture (Figure 1)), as adapted from the original Locator architecture (Battey, Ralph, and Kern 2020). GeoGenIE allows lots of flexibility in the architecture, with each hidden layer either being constant or reduced by a factor with the `--factor` option. The model also includes batch normalization and dropout layers to reduce overfitting and facilitate better cross-batch training.

Installation

To install GeoGenIE, it is recommended to use a virtual environment or conda environment. From the root project directory, enter the following command:

```
pip install .
```

GeoGenIE will be hosted on PyPI in the future for easier installation.

Dependencies

The following packages will be installed when running `pip install .`:

Here is the list of dependencies:

- python >= 3.11
- geopandas
- geopy
- imblearn
- jenkspy
- kaleido
- kneed
- matplotlib
- numba
- numpy
- optuna
- pandas
- plotly
- pynndescent
- pysam
- pyyaml
- requests
- scikit-learn
- scipy
- seaborn
- statsmodels
- torch
- xgboost

Usage

Running GeoGenIE

GeoGenIE can be run with individual command-line arguments, but using a YAML config file is recommended. See `config_files/config.yaml` for a template YAML file. Assuming GeoGenIE is installed in your environment, you can run it like this:

```
geogenie --config config_files/config.yaml
```

Command-line Options

You can see all the command-line options by running the help flag:

```
geogenie -h
```

Note: If you don't want to use the configuration file, you can specify each argument individually on the command line. For example:

```
geogenie --vcf <path/to/vcf_file.vcf.gz> --sample_data
   <path/to/coordinates_file.tsv> <other arguments>
```

We do recommend using the configuration file, however, as it enables reproducible runs and also promotes ease-of-use when performing multiple runs.

Configuration File

You can set all the options for input files, model parameters, etc., in the `config_files/config.yaml` file. Using a configuration file allows tracking of parameters across multiple runs and ensures better reproducibility.

- Python `None` values are represented by `null` (without quotes).
- Python `True` values are represented by `true` (all lowercase, no quotes).
- Python `False` values are represented by `false` (all lowercase, no quotes).

You can also leave comments with `# my_comment`. The arguments can be in any order in the `config_files/config.yaml` file.

Running the Software

```
geogenie --config config_files/config.yaml
```

Required Input Files

- VCF file
  - Specified with `--vcf <path/to/vcf_file.vcf.gz>`.
- Coordinates file
  - Specified with `--sample_data <path/to/coordinates_file.tsv>`.
  - Tab-delimited file with three columns:

    * "sampleID"
    * "x" (longitude)
    * "y" (latitude)
  - Coordinates ("x" and "y") should be in decimal degree format.
  - Unknown coordinates for which predictions should be made should contain the string "nan" in the "x" and "y" columns.
- Known Coordinates file
  - Specified with `--known_sample_data <path/to/coordinates_file.tsv>`.
  - Tab-delimited file with three columns:
    * "sampleID"
    * "x" (longitude)
    * "y" (latitude)
  - Known coordinates file provides recorded localities for per-sample bootstrapped output plots.

Algorithms for Sampling Imbalances

GeoGenIE employs several advanced algorithms to accommodate sampling imbalances, ensuring robust and accurate geographic predictions:

- **Outlier Detection Algorithm**: Adapted from GGOutlieR, this algorithm identifies and removes translocated individuals that deviate significantly from expected geographic and/or genetic patterns.
- **Weighted Loss Function**: Implemented with PyTorch, this function uses inverse sample weights to focus the loss function (RMSE, Huber, or MAE) more heavily on areas with lower sample densities, balancing the influence of samples from different regions.
- **Synthetic Oversampling Method**: Adapted from SMOTE, this regression-based method uses a genotype interpolation algorithm based on Mendelian inheritance to generate synthetic samples in underrepresented regions, balancing the dataset.

GeoGenIE Features and Settings

Data Input and Preprocessing

GeoGenIE supports various options for data input and preprocessing:

- `--min_mac`: Minimum minor allele count (MAC) to retain SNPs. Default: 2.
  - **Description**: This setting filters out SNPs with a minor allele count below the specified threshold. Increasing the MAC helps to remove rare variants, which can reduce noise and improve the signal for common genetic patterns.
  - **Tip**: Increasing the MAC can help to filter out rare variants and reduce noise in the data. However, be wary of overfiltering the data so as not to lose inherent data patterns and information that the model can learn from.
  - **Importance**: **High**

- `--max_SNPs`: Maximum number of SNPs to randomly subset. Default: None (Use all SNPs).
  - **Description**: This setting limits the number of SNPs used in the analysis to reduce computational load. Subsetting the SNPs can speed up processing but may result in loss of some genetic information.
  - **Tip**: Use this option to reduce computational load by limiting the number of SNPs used in the analysis. However, note that removing too many SNPs could reduce the amount of information that the model can learn from.
  - **Importance**: **Medium**

Model Configuration

Configure the deep learning model with the following options:

- `--dropout_prop`: Dropout rate to reduce overfitting. Default: 0.25.
  - **Description**: Dropout is a regularization technique that helps prevent overfitting by randomly setting a fraction of input units to zero during training. This encourages the network to learn more robust features that generalize well to new data.
  - **Tip**: Adjust this rate to balance model complexity and generalization.
  - **Importance**: **High**
- `--nlayers`: Number of hidden layers to include in the neural network. Default: 10.
  - **Description**: The number of hidden layers in the neural network. More layers can capture more complex patterns but also increase the risk of overfitting.
  - **Tip**: Increase the number of layers for more complex models but beware of overfitting.
  - **Importance**: **Medium**
- `--width`: Number of neurons per layer. Default: 256.
  - **Description**: The number of neurons in each hidden layer. More neurons allow the network to learn more detailed patterns, but also increase computational complexity and the risk of overfitting.
  - **Tip**: Higher width values allow the model to learn more complex features but increase computation time and can also lead to overfitting.
  - **Importance**: **Medium**
- `--criterion`: Model loss criterion. Options: 'rmse', 'huber', 'drms'. Default

: 'rmse'.

- **Description**: The loss function used to evaluate the model's performance. RMSE (Root Mean Squared Error) is a standard measure of prediction accuracy. Huber loss is a combination of RMSE and MAE (Mean Absolute Error), which is less sensitive to outliers.
- **Tip**: Try 'rmse' first, but use 'huber' for a combination of mean squared error and mean absolute error to handle outliers.
- **Importance**: **Medium**

- `--load_best_params`: Load best parameters from previous Optuna search. The value specified here (if not None) should be the path to the JSON file with the best found parameters from a previous Optuna search. This file can be found in: `<output_dir>/optimize`. Default: None.
  - **Description**: This setting allows you to reuse the best parameters found in a previous parameter search, saving time and ensuring consistent model performance.
  - **Tip**: Use this option to save time by reusing optimized parameters without having to re-run the parameter search.
  - **Importance**: **Medium**
- `--use_gradient_boosting`: Use Gradient Boosting model instead of deep learning model. **NOTE**: This option is currently still in developmental stages. Default: False.
  - **Description**: This setting switches from using a deep learning model to a Gradient Boosting model, which can be more effective for smaller datasets or when the deep learning model overfits.
  - **Tip**: Gradient Boosting can be effective for smaller datasets or when deep learning models overfit.
  - **Importance**: **Low**
- `--dtype`: PyTorch data type. Options: 'float32', 'float64'. **NOTE**: Only 'float32' can be used with a GPU. Either 'float64' or 'float32' can be used if only using CPU threads. Default: 'float32'.
  - **Description**: The data type used for computations. 'float32' is standard for GPU computation due to its balance of precision and performance. 'float64' provides higher precision but requires more memory (i.e., RAM).
  - **Tip**: Use 'float64' for higher precision at the cost of increased memory usage.
  - **Importance**: **Medium**

Training Parameters

Define training parameters:

- `--max_epochs`: Maximum training epochs. Default: 5000.
  - **Description**: The maximum number of training cycles through the entire dataset. Setting this high allows the early stopping mechanism to control when training should halt.
  - **Tip**: Set this high and rely on the early stopping mechanism to prevent overfitting.
  - **Importance**: **High**
- `--learning_rate`: Learning rate for optimizer. Default: 0.001.
  - **Description**: The step size used by the optimizer to update model weights. Lower learning rates allow for finer adjustments, while higher rates speed up training but may overshoot optimal values.
  - **Tip**: Adjust this parameter to control the speed of learning; lower values provide finer adjustments.
  - **Importance**: **High**

- `--train_split`: Training data proportion. Default: 0.8.
  - **Description**: The proportion of the dataset to be used for training. The remainder is used for validation and testing.
  - **Tip**: Ensure the sum of `--train_split` and `--val_split` is 1.0.
  - **Importance**: **High**
- `--val_split`: Validation data proportion. Default: 0.2.
  - **Description**: The proportion of the dataset to be used for validation. Validation helps monitor model performance and avoid overfitting.
  - **Tip**: Use a larger validation split for more robust model evaluation.
  - **Importance**: **High**
- `--batch_size`: Training batch size. Default: 32.
  - **Description**: The number of samples processed before the model's weights are updated. Larger batch sizes provide more stable updates but require more memory.
  - **Tip**: Larger batch sizes can lead to more stable training but require more memory.
  - **Importance**: **Medium**
- `--early_stop_patience`: Epochs to wait after no improvement before stopping. Default: 48.
  - **Description**: The number of epochs with no improvement in validation performance before stopping training early. Helps prevent overfitting by halting training once performance plateaus.
  - **Tip**: Decrease this value to stop training earlier if no performance improvement has occurred for many iterations, leading to overfitting.
  - **Importance**: **Medium**
- `--l2_reg`: L2 regularization weight. Default: 0.0.
  - **Description**: A penalty added to the loss function to penalize large weights, helping to reduce overfitting.
  - **Tip**: Use L2 regularization to penalize large weights and reduce overfitting.
  - **Importance**: **Medium**
- `--do_bootstrap`: Enable bootstrapping. Default: False.
  - **Description**: Bootstrapping involves repeatedly resampling the dataset (i.e., the loci) with replacement to estimate confidence intervals for predictions.
  - **Tip**: Enable bootstrapping to estimate confidence intervals for predictions.
  - **Importance**: **Medium**
- `--nboots`: Number of bootstrap replicates. Default: 100.
  - **Description**: The number of bootstrap samples to generate. More replicates provide more accurate confidence intervals but increase computation time.
  - **Tip**: Increase this value for more accurate confidence intervals.
  - **Importance**: **Medium**

Geographic Density Sampler

Configure the geographic density sampler:

- `--use_weighted`: Use inverse-weighted probability sampling. Options: 'loss', 'none'. Default: 'none'.

  - **Description**: Applies weights to samples based on their inverse density, giving more importance to underrepresented regions during training.
  - **Tip**: Apply inverse weighting to address imbalanced sampling densities.
  - **Importance**: **High**
- `--oversample_method`: Synthetic oversampling/undersampling method. Options: 'kmeans', 'none'. Default: 'none'.
  - **Description**: Generates synthetic samples in underrepresented regions using clustering methods and our custom adaptation of a regression-based SMOTE algorithm that interpolates genotypes with a Mendelian inheritance pattern. This option will balance the dataset by creating synthetic samples where sampling densities are low.
  - **Tip**: Use 'kmeans' to create synthetic samples for underrepresented regions.
  - **Importance**: **High**
- `--oversample_neighbors`: Number of nearest neighbors for oversampling. Default: 5.
  - **Description**: Controls the number of nearest neighbors considered when generating synthetic samples. More neighbors can provide more diverse synthetic samples.
  - **Tip**: Adjust to control the number of synthetic samples generated.
  - **Importance**: **Medium**
- `--use_kmeans`: Use KMeans clustering in the sampler. Default: True.
  - **Description**: KMeans clustering helps to generate synthetic samples based on clusters, ensuring that synthetic samples are representative of the data structure.
  - **Tip**: Essential for generating synthetic samples based on clusters.
  - **Importance**: **High**
- `--n_bins`: Number of bins for synthetic resampling. Default: 8.
  - **Description**: Determines the granularity of the sampling density adjustments. More bins provide finer adjustments.
  - **Tip**: Increase for finer-grained sampling density adjustments.
  - **Importance**: **Medium**
- `--w_power`: Exponential power for inverse density weighting. Default: 1.0.
  - **Description**: Controls the strength of the weighting applied to sample densities. Higher values result in more aggressive weighting.
  - **Tip**: Adjust for more aggressive weighting of sampling density.
  - **Importance**: **Medium**
- `--max_clusters`: Maximum number of clusters for KMeans. Default: 10.
  - **Description**: Sets the upper limit for the number of clusters used in KMeans clustering. More clusters provide more detailed clustering.
  - **Tip**: Increase for more detailed clustering, but note that this will increase computation time.
  - **Importance**: **Medium**
- `--focus_regions`: Geographic regions of interest for sampling density weights.
  - **Description**: Specifies regions to prioritize during sampling. Helps focus the model on areas of interest by adjusting sampling weights.
  - **Tip**: Use to prioritize specific geographic areas in sampling. Should be a list of tuples with focus region bounding areas (min_lon, min_lat, max_lon, max_lat).

Multiple regions can be specified in the list.
  - **Importance**: **Low**
- `--normalize_sample_weights`: Normalize sample weights between 0 and 1. Default: False.
  - **Description**: Ensures that all sample weights are on a comparable scale, which can improve the stability of the training process.
  - **Tip**: Ensure all sample weights are on a comparable scale.
  - **Importance**: **Low**

Outlier Detection

GeoGenIE can remove outliers flagged as distant from nearby samples in spatial and genetic contexts:

- `--detect_outliers`: Enable outlier detection. Default: False.
  - **Description**: Detects and removes samples that deviate significantly from expected geographic and/or genetic patterns, improving model accuracy.
  - **Tip**: Enable to remove samples that deviate significantly from expected geographic and/or genetic patterns.
  - **Importance**: **High**
- `--min_nn_dist`: Minimum distance between nearest neighbors for outlier detection. Units are in

meters. Default: 1000.

- **Description**: Sets the threshold for considering samples as outliers based on their distance from nearest neighbors. Higher values detect more distant outliers.
- **Tip**: Increase to detect only very distant outliers. This setting is also useful to exclude neighbors that are in very close proximity to each other.
- **Importance**: **Medium**
- `--scale_factor`: Scale factor for geographic distance. Default: 100.
  - **Description**: Adjusts the scaling of geographic distances in outlier detection. Helps fine-tune the sensitivity of outlier detection.
  - **Tip**: Adjust to control the scaling of distances in outlier detection. Probably safer to not adjust this setting, unless you know what you are doing.
  - **Importance**: **Low**
- `--significance_level`: Significance level for p-values with Maximum Likelihood Estimation (MLE) gamma distribution to determine outliers. Default: 0.05.
  - **Description**: Sets the p-value threshold for identifying outliers. Lower values increase stringency, reducing the risk of false positives but potentially missing true outliers.
  - **Tip**: Lower values increase the stringency of outlier detection, at the risk of more false negatives.

  - **Importance**: **Medium**

- `--maxk`: Maximum number of nearest neighbors for outlier detection. The optimal K will be searched between K = 2 and K = 'maxk'. Used for KMeans clustering. Default: 50.
  - **Description**: Sets the range for the number of nearest neighbors considered in outlier detection. Higher values increase computational complexity but can improve accuracy.
  - **Tip**: Adjust to control the scope of neighbor comparisons.
  - **CAUTION**: Raising too high can increase computation time and resources.
  - **Importance**: **Medium**

Bootstrapping for Error Estimates

To obtain confidence intervals for locality predictions, enable bootstrapping with the `--do_bootstrap` boolean option. Bootstrapping is parallelized, and you can set the number of CPU threads with `--n_jobs <n_cpus>` or `--n_jobs -1` to use all available CPU threads.

Using bootstrapping generates additional plots showing confidence intervals for each sample, saved in `<output_dir>/plots/bootstrapped_sample_ci/<prefix>_bootstrap_ci_plot_<test/val/pr`

The file type for output plots can be specified with `--filetype "pdf"`, `--filetype "png"`, or `--filetype "jpg"`. The number of bootstrap replicates can be changed with `--nboots <integer>`.

Embedding Settings

GeoGenIE offers several embedding options for input features (i.e., loci):

- `--embedding_type`: Embedding to use with input SNP dataset. Supported options: 'pca', 'kernelpca', 'nmf', 'lle', 'mca', 'mds', 'polynomial', 'tsne', and 'none'. Default: 'none' (no embedding).
  - **Description**: Determines the type of embedding technique used to reduce the dimensionality of the input SNP data. Different techniques capture different aspects of the data.
  - **Tip**: Use 'pca' for principal component analysis to reduce dimensionality.
  - **Importance**: **High**
- `--n_components`: Number of components for 'pca' or 'tsne' embeddings. Default: None (Search for optimal components).
  - **Description**: Sets the number of components to retain in the embedding process. More components capture more variance but increase computational complexity.
  - **Tip**: Specifying a higher number of components can capture more variance but may also increase computational complexity.
  - **Importance**: **Medium**
- `--embedding_sensitivity`: Sensitivity setting for selecting the optimal number of components with 'mca' and 'pca'. Default: 1.0.

  - **Description**: Adjusts the sensitivity for determining the number of components to retain. Higher sensitivity captures more variance but can lead to overfitting.
  - **Tip**: Adjust this parameter to fine-tune the balance between overfitting and underfitting.
  - **Importance**: **Medium**
- `--tsne_perplexity`: Perplexity setting for T-SNE embedding. Default: 30.
  - **Description**: Controls the balance between local and global aspects of the data captured by T-SNE. Lower perplexity values focus more on local structure.
  - **Tip**: Lower perplexity values can capture finer details but may lead to more noise.
  - **Importance**: **Medium**
- `--polynomial_degree`: Polynomial degree for 'polynomial' embedding. Default: 2.
  - **Description**: Sets the degree of the polynomial used in the embedding process. Higher degrees increase complexity and computational load.
  - **CAUTION**: Higher degrees add complexity and computational overhead; increase with caution.
  - **Importance**: **Low**
- `--n_init`: Number of initialization runs for Multi-Dimensional Scaling embedding. Default: 4.
  - **Description**: Sets the number of times the embedding algorithm is run with different initializations. More runs provide more stable results.
  - **Tip**: More initialization runs can provide more stable results but increase computation time.
  - **Importance**: **Low**

Plot Settings

Set plotting parameters to customize the visualizations:

- `--show_plots`: Show in-line plots. Default: False.
  - **Description**: Controls whether plots are displayed interactively. Useful for Jupyter or Google Collab notebooks.
  - **Tip**: Enable for interactive environments like Jupyter or Google Collab notebooks.
  - **Importance**: **Low**
- `--fontsize`: Font size for plot labels, ticks, and titles. Default: 24.
  - **Description**: Sets the font size for all text in the plots, ensuring readability.
  - **Tip**: Adjust to improve readability of plots.
  - **Importance**: **Low**
- `--filetype`: File type for saving plots. Default: 'pdf'.
  - **Description**: Specifies the format for saving plots. 'pdf' is suitable for high-quality vector graphics, while 'png' and 'jpg' are image formats.
  - **Tip**: Use 'pdf' for high-quality vector graphics, or 'png' or 'jpg' for an image format.
  - **Importance**: **Low**
- `--plot_dpi`: DPI for image format plots. Has no effect on 'pdf' format. Default: 300.
  - **Description**: Sets the resolution for saved plots in image formats. Higher DPI

values produce clearer images.
  - **Tip**: Higher DPI produces clearer images but increases file size.
  - **Importance**: **Low**
- `--remove_splines`: Remove all axis splines from map plots. Default: False.
  - **Description**: Controls whether axis lines are removed from plots, creating a cleaner appearance.
  - **Tip**: Use for cleaner map visualizations.
  - **Importance**: **Low**
- `--shapefile`: URL or file path for shapefile used in plotting. Default: Continental USA County Lines basemap.
  - **Description**: Specifies the shapefile to use as a base map for geographic plots. Custom shapefiles can be used for different regions.
  - **Tip**: Specify a custom shapefile for regions outside the continental USA.
  - **Importance**: **Low**
- `--basemap_fips`: FIPS code for basemap. This will subset the basemap to just the appropriate region encoded by the FIPS code. For example, the Arkansas state FIPS code is '05'. Default: None (do not zoom to FIPS code on basemap).
  - **Description**: Subsets the basemap to focus on a specific region using FIPS codes. Useful for zooming into specific areas.
  - **Tip**: Use to focus plots on specific states or regions. E.g., the Arkansas FIPS code is "05".
  - **Importance**: **Low**
- `--highlight_basemap_counties`: Highlight specified counties on the base map in gray. Default: None (no highlighting).
  - **Description**: Highlights specified counties on the base map, enhancing visualization of areas of interest.
  - **Tip**: Highlight areas of interest for better visualization.
  - **Importance**: **Low**
- `--samples_to_plot`: Comma-separated sample IDs to plot when using bootstrapping. The provided samples will be plotted with confidence intervals as contours on top of the basemap. Default: None (make a plot for every sample).
  - **Description**: Specifies samples to plot, allowing focused visualization of particular cases.
  - **Tip**: Plot specific samples to focus on particular cases.
  - **Importance**: **Low**
- `--n_contour_levels`: Number of contour levels for the `<output_dir>/plots/<prefix>_geographic` plot. Default: 20.
  - **Description**: Sets the number of contour levels for error distribution plots. More levels provide more detailed contours.
  - **Tip**: Adjust for more detailed or simplified contour plots.
  - **Importance**: **Low**
- `--min_colorscale`: Minimum colorbar value for the `<output_dir>/plots/<prefix>_geographic_e` plot. Default: 0.
  - **Description**: Sets the minimum value for the color scale in error distribution plots. Ensures color scale matches data range.

  - **Tip**: Set to match the range of

your data.

- **Importance**: **Low**
- `--max_colorscale`: Maximum colorbar value for the `<output_dir>/plots/<prefix>_geographic_e` plot. Default: 300.
  - **Description**: Sets the maximum value for the color scale in error distribution plots. Ensures color scale matches data range.
  - **Tip**: Increase if your data range exceeds the default.
  - **Importance**: **Low**
- `--sample_point_scale`: Scale factor for sample point size on plots. Default: 2.
  - **Description**: Adjusts the size of sample points in plots for better visibility.
  - **Tip**: Adjust to ensure points are visible but not overwhelming.
  - **Importance**: **Low**
- `--bbox_buffer`: Buffer for the sampling bounding box on map visualizations. Default: 0.1.
  - **Description**: Adds a buffer around the sampling area in map visualizations, providing additional context.
  - **Tip**: Increase to add more context around the sampling area.
  - **Importance**: **Low**

Output and Miscellaneous

Configure output and other miscellaneous settings:

- `--prefix`: Output file prefix. Used for all output files. Default: 'output'.
  - **Description**: Sets a prefix for all output files, helping to organize results from different runs.
  - **Tip**: Use meaningful prefixes to organize output files from different GeoGenIE runs.
  - **Importance**: **High**
- `--output_dir`: Directory to store output files. Default: './output'.
  - **Description**: Specifies the directory for storing output files, keeping results organized.
  - **Tip**: Organize outputs by specifying a unique directory and/or unique prefix for each run.
  - **Importance**: **High**
- `--n_jobs`: Number of CPU jobs to use. Default: -1 (use all CPUs).
  - **Description**: Controls the number of CPU threads used for parallel processing.
  - **Tip**: Adjust to limit CPU usage if running multiple processes.
  - **Importance**: **High**

- `--gpu_number`: GPU number (an integer) for computation. Default: None (use CPU only).
  - **Description**: Specifies the GPU to use for computation, speeding up training for large datasets.
  - **Tip**: Use a GPU to speed up training for large datasets.
  - **Importance**: **Low**
- `--seed`: Random seed for reproducibility. Default: None.
  - **Description**: Sets a random seed to ensure reproducible results across runs.
  - **Tip**: Set a seed to ensure reproducible results.
  - **Importance**: **Low**
- `--sqldb`: SQLite3 database directory for Optuna optimization. Specifying a file path here allows Optuna parameter searches to be resumed. Leaving this value as None stores all Optuna searches in-memory and renders resumability disabled. Default: None.
  - **Description**: Stores Optuna optimization results in a SQLite3 database, enabling resumption of parameter searches.
  - **Tip**: Specify to save optimization results for future use and resuming Optuna optimization.
  - **Importance**: **Low**
- `--verbose`: Verbosity level for logging. Default: 1.
  - **Description**: Sets the level of detail for logging messages. Higher levels provide more detailed logs.
  - **Tip**: Increase to 2 get more detailed logging information, or decrease to 0 to reduce logging information.
  - **Importance**: **Low**

Output Files and File Structure

Outputs are saved to the directory specified by `--output_dir <my_output_dir>/<prefix_>_*`. The prefix is specified with `--prefix <prefix>`. The directory structure of `<output_dir>` includes:

- `benchmarking`: Execution times for model training and prediction, with one line per bootstrap replicate if using bootstrapping.
- `bootstrapped_sample_ci`: One plot per sample showing confidence intervals on a map.
- `bootstrap_metrics`: JSON files with statistical metrics for each bootstrap replicate, in `test` and `val` subdirectories.
- `bootstrap_predictions`: CSV files containing predictions for each bootstrap replicate, in `test`, `val`, and `unknown` subdirectories.
- `bootstrap_summaries`: Mean, median, and standard deviation representations of bootstrap replicates for the test, val, and unknown ("pred") datasets.
- `data`: Text files with sample IDs detected as outliers if `--detect_outliers` is enabled.
- `logfiles`: Logs with INFO, WARNING, and ERROR messages, including timestamps and GeoGenIE modules.
- `models`: Trained PyTorch models saved as ".pt" files, one per bootstrap if `--do_bootstrap` is enabled.

- `optimize`: Optuna results, including the best-found parameters as a JSON file.
- `plots`: All plots and visualizations, including model prediction error visualizations and the basemap shapefile specified with `--shapefile <url>`. Per-sample plots visualizing bootstrapped prediction error are saved in the `<output_dir>/plots/bootstrapped_sample_ci` subdirectory.

Warning: Re-running GeoGenIE with the same `output_dir` and `prefix` will overwrite all outputs except the Optuna SQL database.

Plot Descriptions

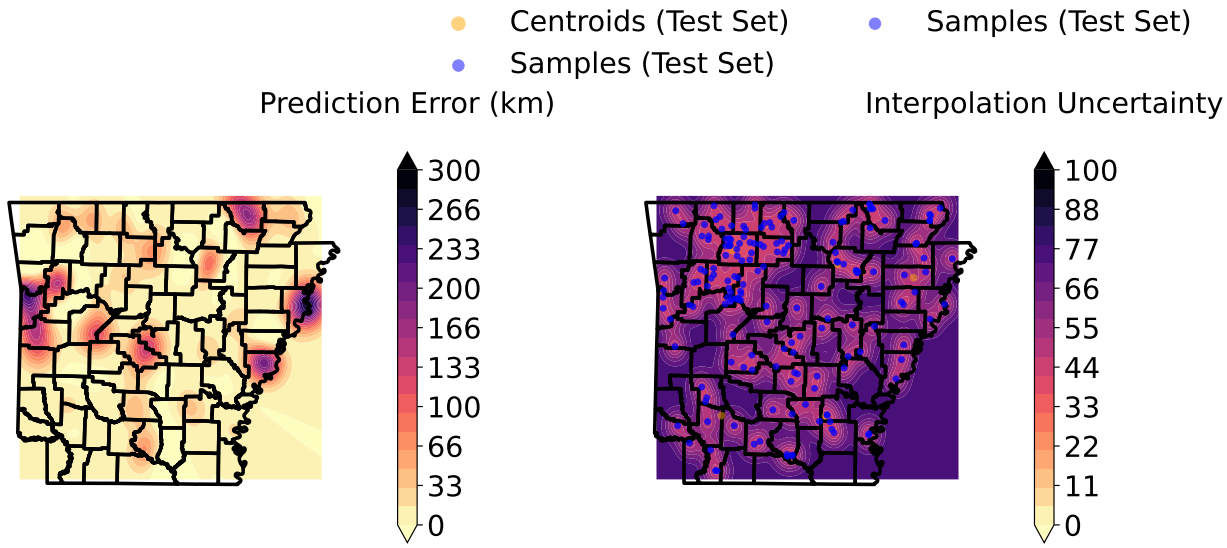

Figure 2: Geographic error distribution of the model predictions interpolated across the Arkansas landscape. Interpolated contour levels represent error magnitudes. Prediction error is Haversine distance between the predicted and recorded localities, in km. This hold-out test dataset was used to obtain realistic predition error estimates.

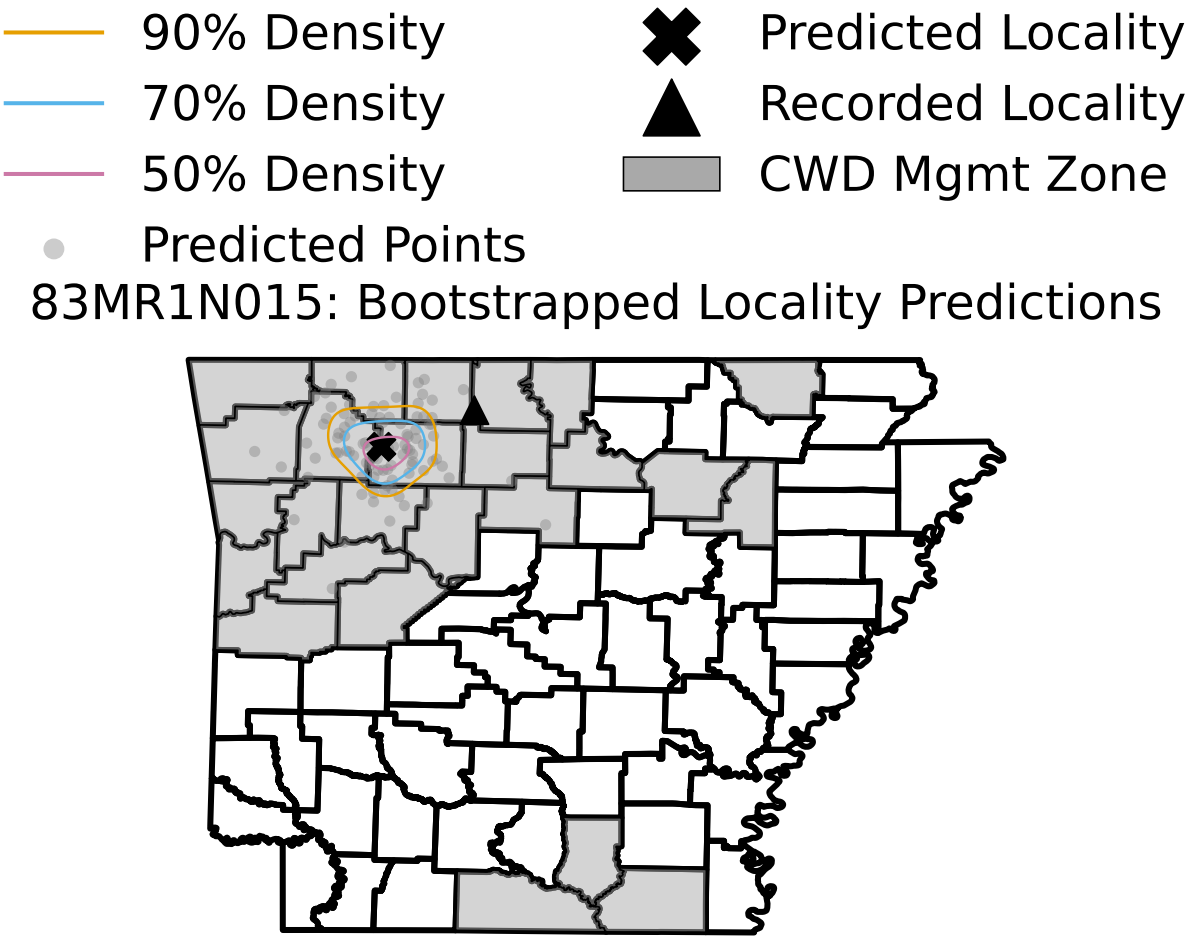

Figure 3: GeoGenIE bootstrap predictions (gray circles; N=100), with the geographic centroid of the bootstrap replicates being marked by **X** and the recorded locality as ▲. Orange, blue, and pink contours contain 90, 70, and 50 percent of the bootstrap replicates, respectively. This hold-out test dataset was used to obtain realistic predition error estimates.

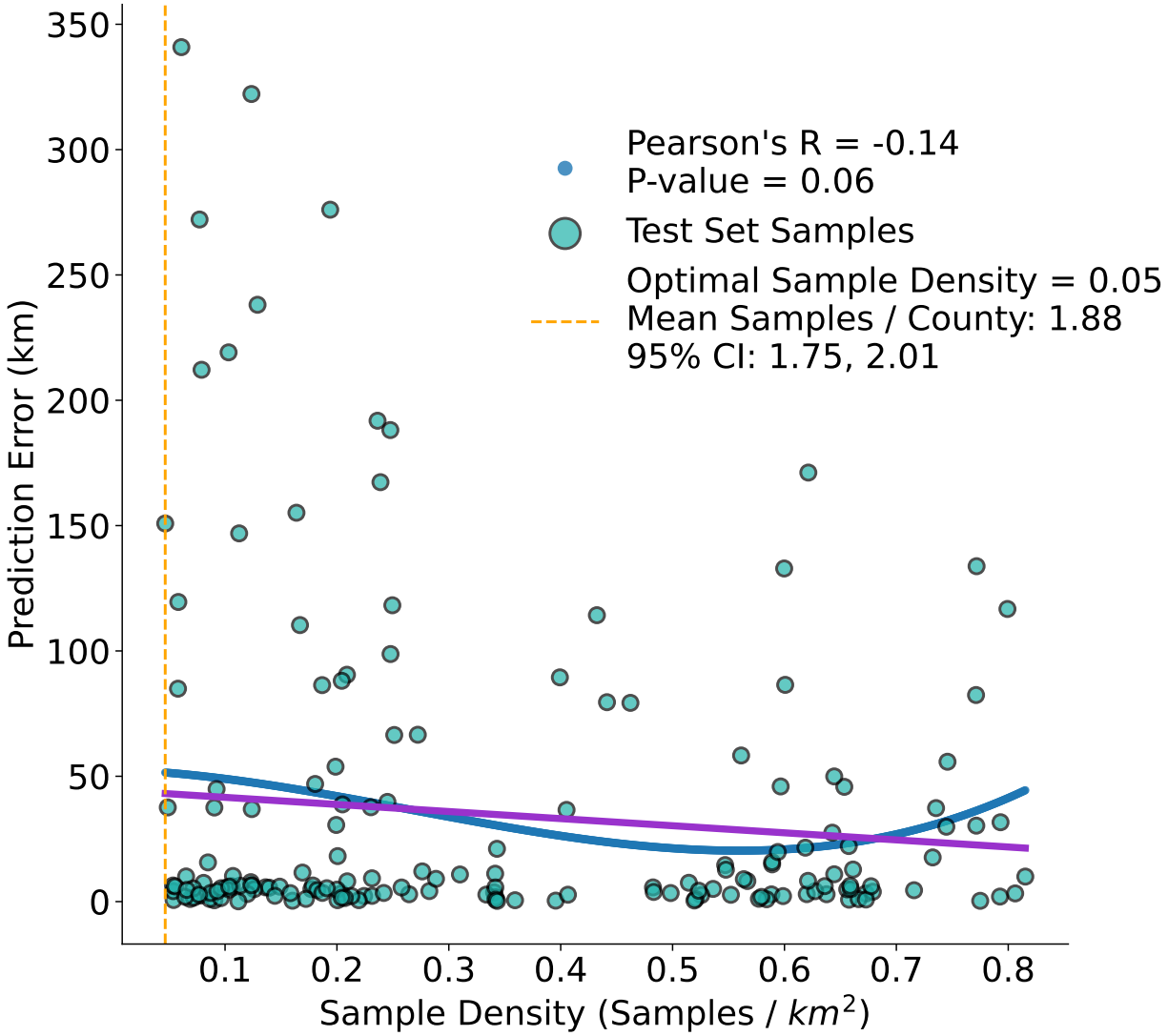

Figure 4: Linear and non-linear (3 *rd* order polynomial) regressions between sampling density (samples / km^2) and prediction error (km). Prediction error is the Haversine distance between the predicted and recorded localities. The orange dashed line represents optimal sampling density as the knee of the polynomial curve, beyond which sampling efforts may yield diminishing returns. This hold-out test dataset was used to obtain realistic predition error estimates.

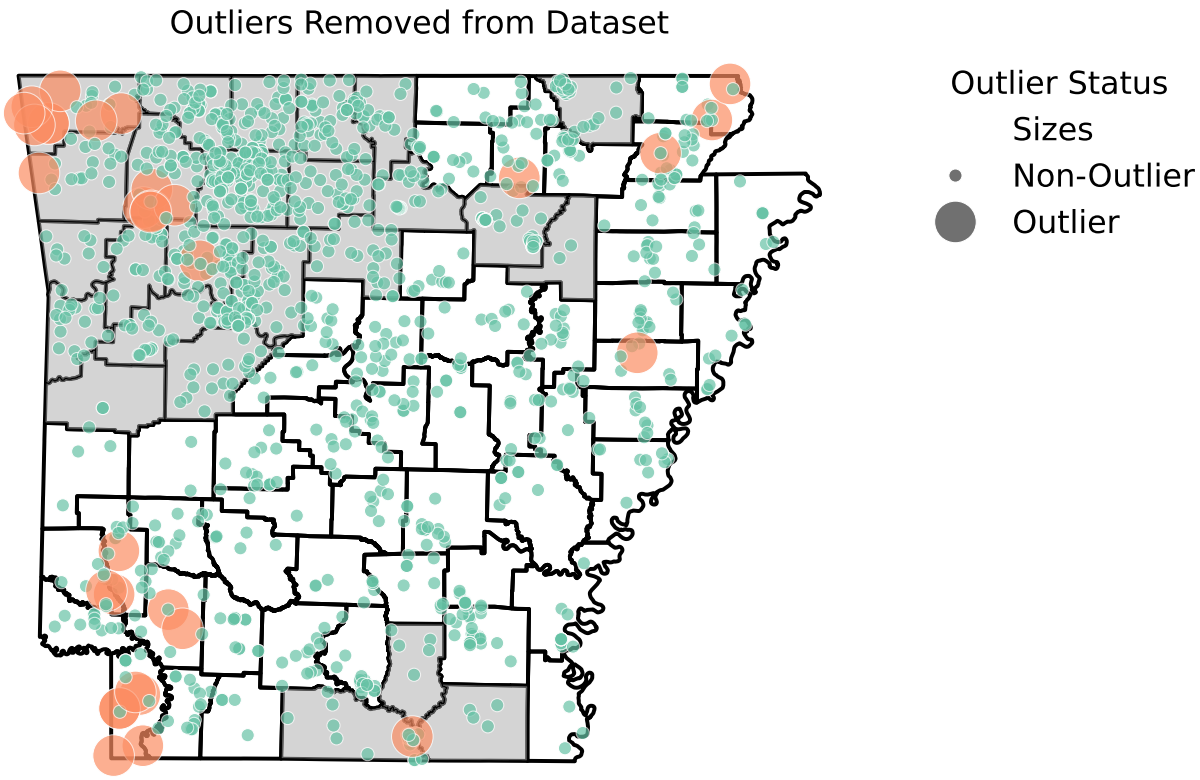

Figure 5: Map depicting sample outliers (large orange circles) removed from the training dataset by our algorithm adapted from GGOutlieR. Non-outliers are illustrated as the smaller green circles.

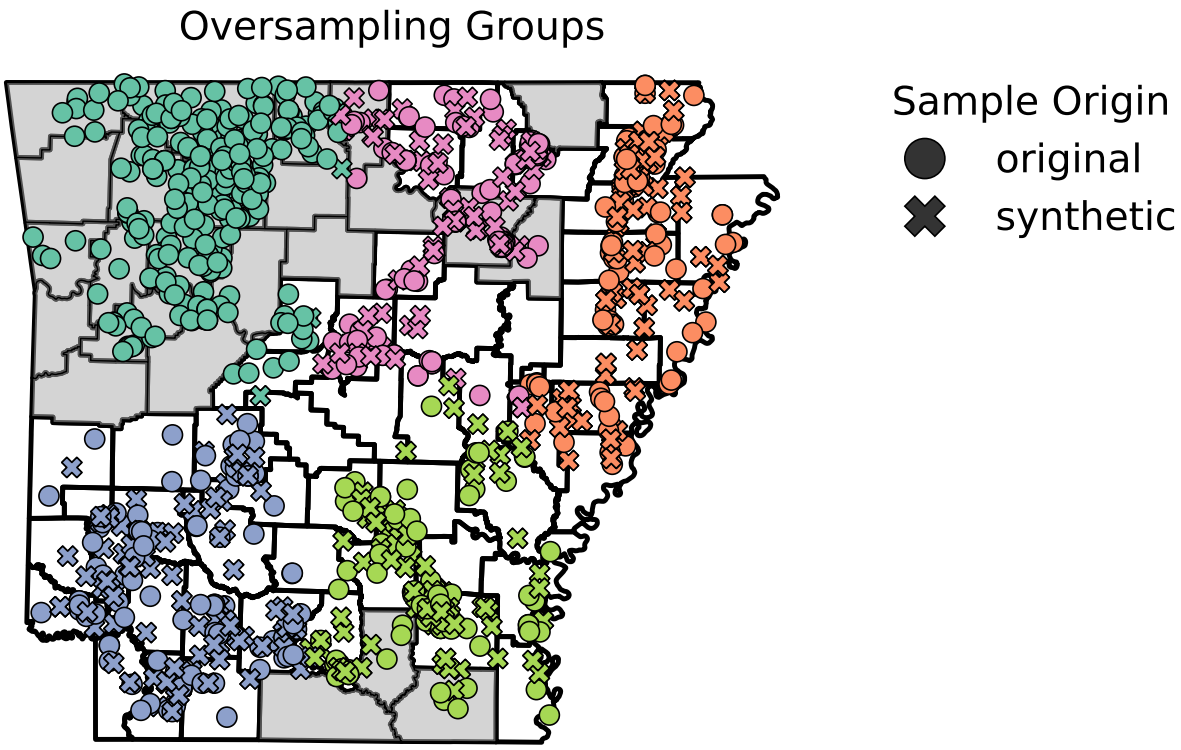

Figure 6: Training dataset samples, with "x" markers depicting synthetically created samples via our custom Mendelian inheritance interpolation method algorithm from a regression-based SMOTE method. Synthetic sample generation frequencies are inversely proportional to the sampling density (samples / km^2). Circles represent real samples that were not synthetically created.

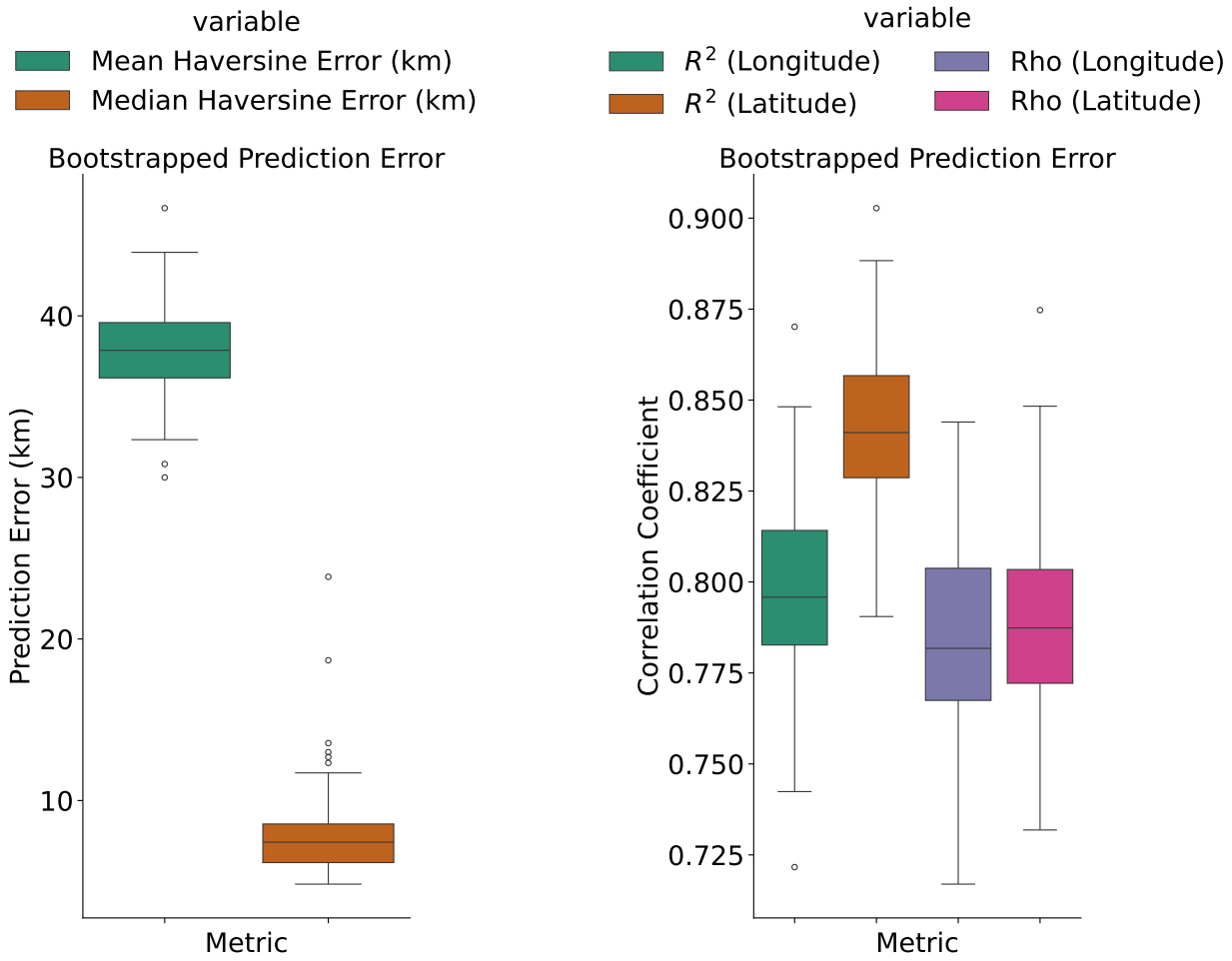

Figure 7: Boxplots, summarized across '–nboots' bootstrap replicates, showing (Left) the mean and median prediction error, represented as the Haversine distance between predicted and recorded localities (in Kilometers). (Right) Pearson's and Spearman's correlation coefficients depicting the correlation between the predicted and recorded localities. This hold-out test dataset was used to obtain realistic predition error estimates.

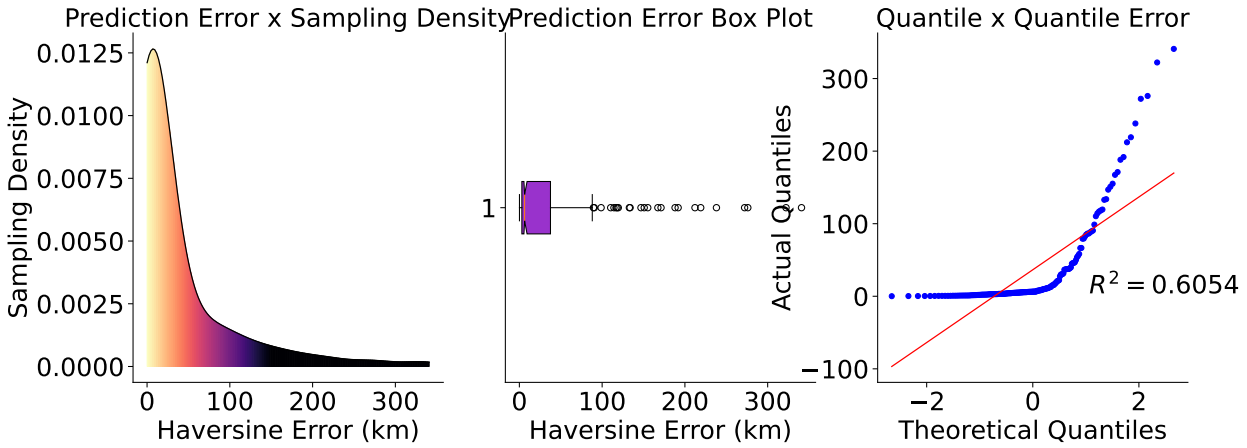

Figure 8: (Left) Area plot depicting prediction error (i.e., Haversine distance between predicted and recorded localities, in km) versus sampling density (samples / km^2). The color gradient corresponds to the geographic interpolation of prediction error in (Figure 2). (Middle) Boxplot showing the mean prediction error. (Right) Quantile X quantile regression plot of mean prediction error. This hold-out test dataset was used to obtain realistic predition error estimates.

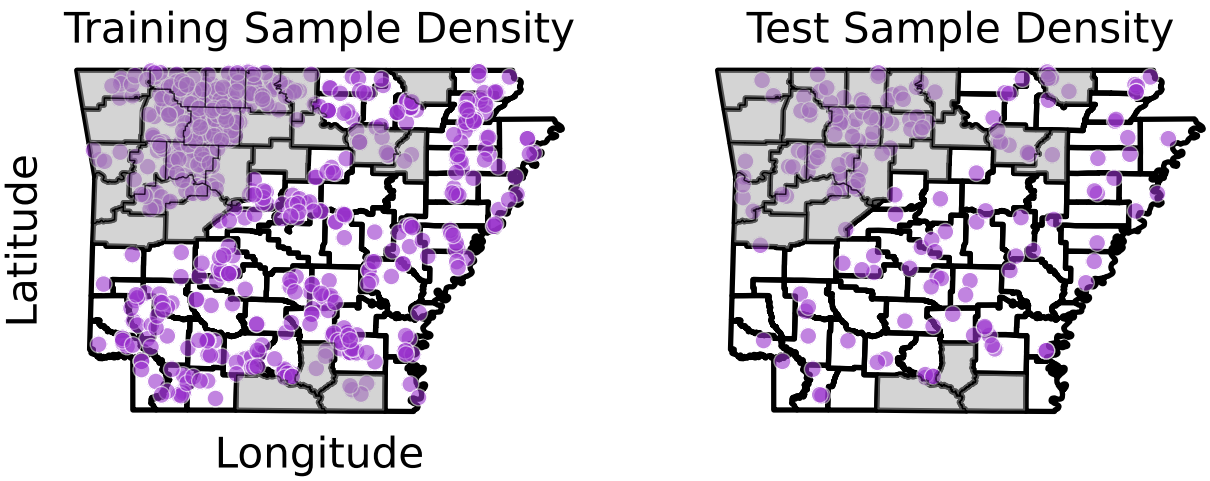

Figure 9: Samples (purple circles) selected for the training and test (i.e., hold-out) datasets and visualized on a map of Arkansas.

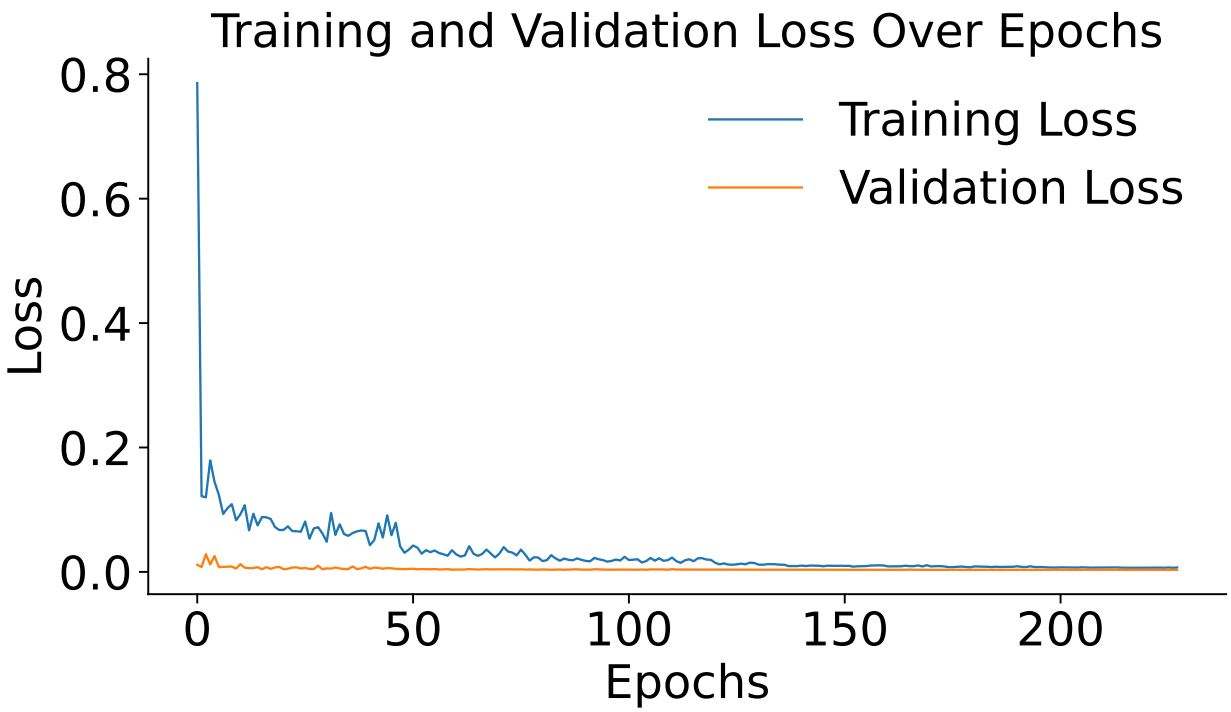

Figure 10: Training and validation loss over all epochs, visualizing the model's learning process and allowing diagnosis of potential overfitting or underfitting.

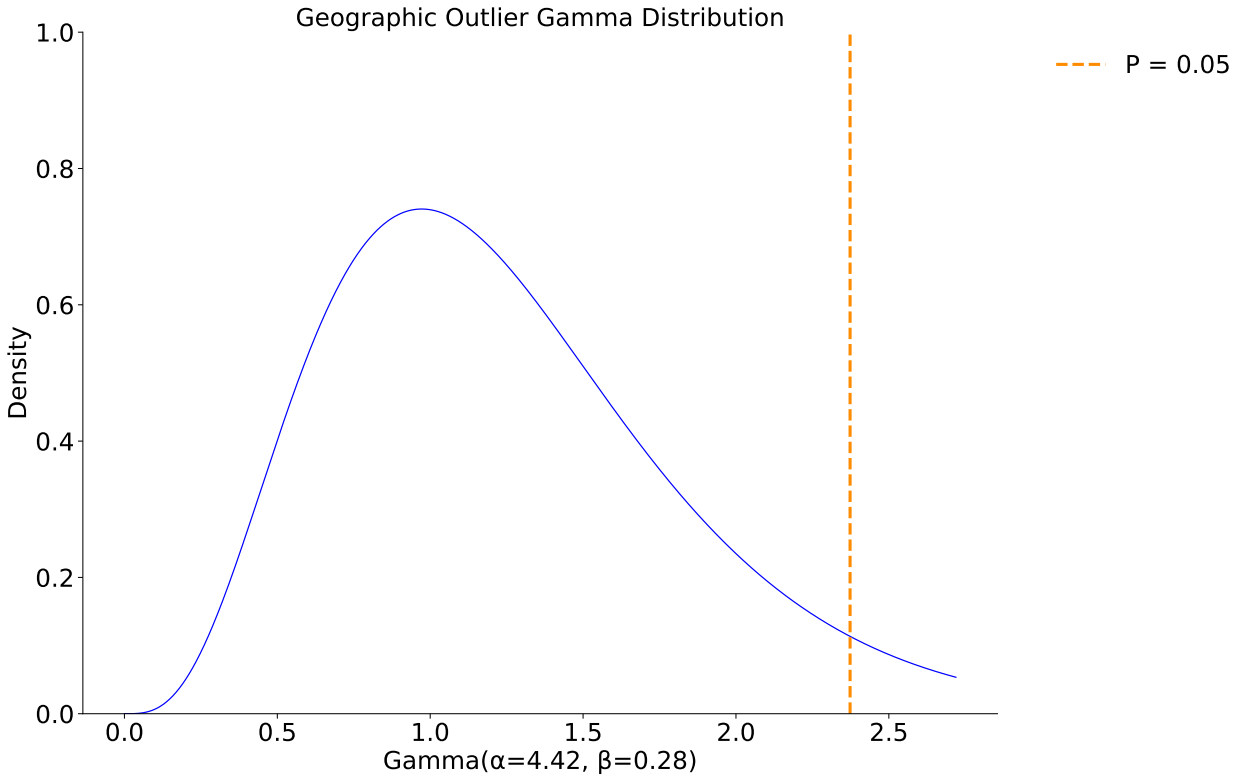

Figure 11: Geographic outlier gamma distribution used to identify the geographic outliers via our outlier removal algorithm adapted from GGOutlieR. The gamma distribution fit allows significant ($P < 0.05$) geographic outliers to be detected and removed from the training dataset.

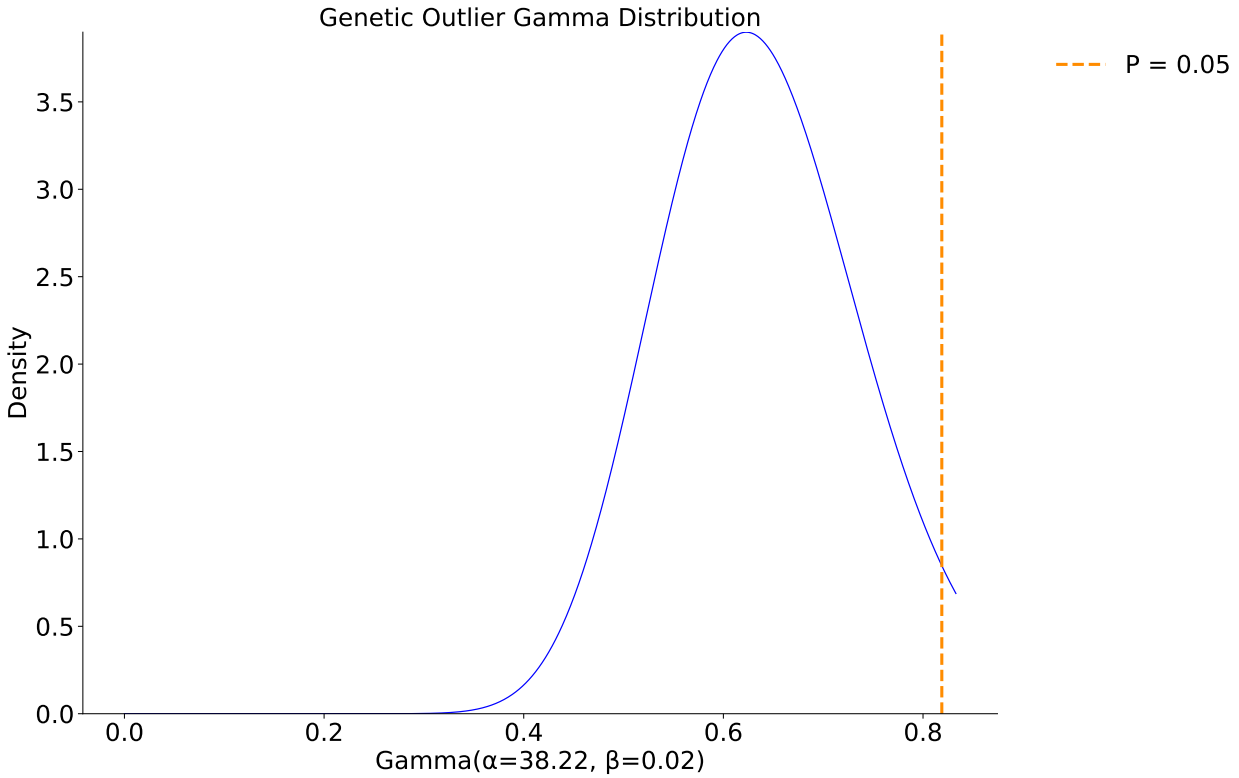

Figure 12: Geneetic outlier gamma distribution used to identify the genetic outliers via our outlier removal algorithm adapted from GGOutlieR. The gamma distribution fit allows significant ($P < 0.05$) genetic outliers to be detected and removed from the training dataset.

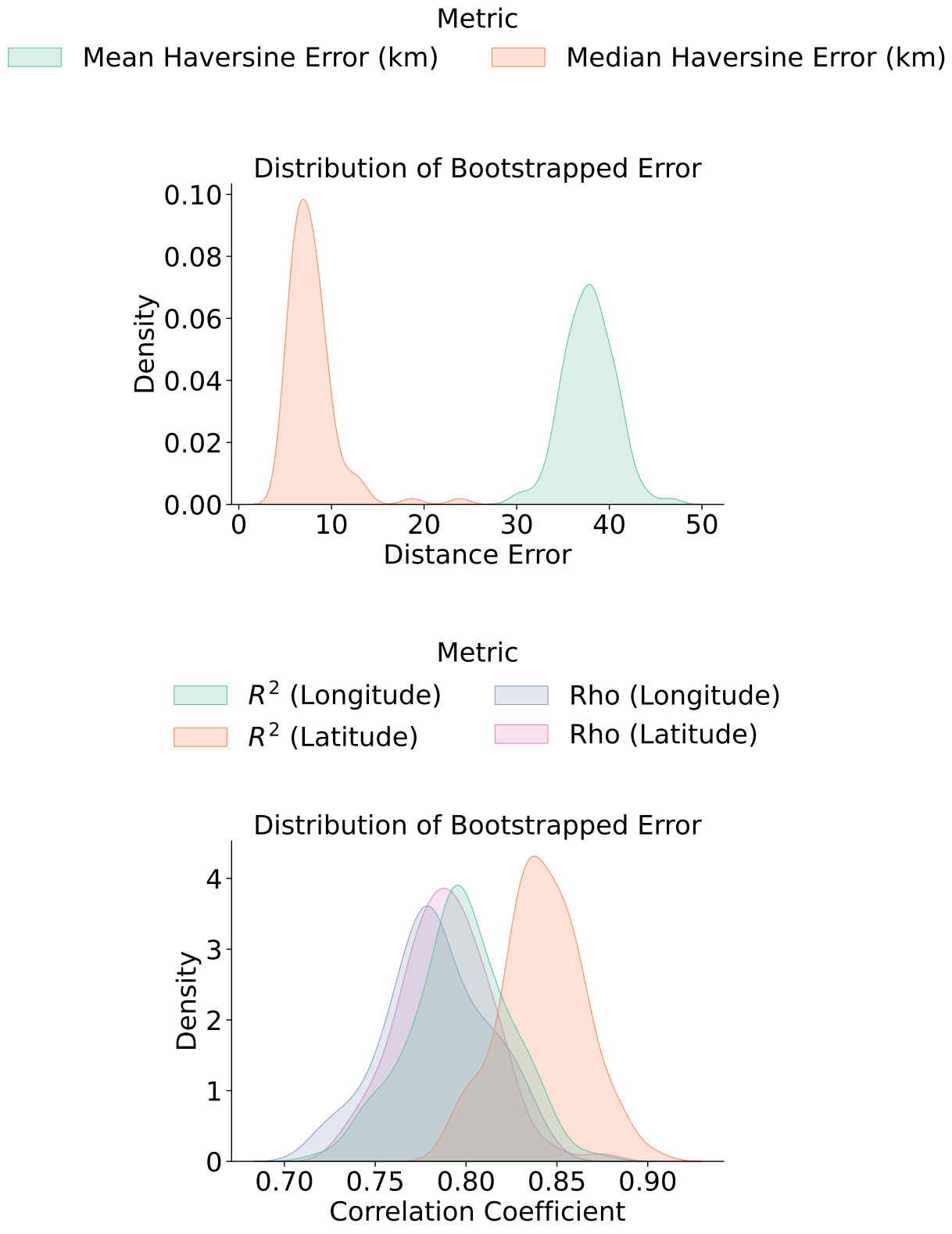

Figure 13: Distribution of prediction errors (i.e., Haversine distance between predicted and recorded localities, in km) across N=100 bootstrap replicates. It visualizes the variability and spread of prediction errors, providing insights into the model's robustness and consistency.

Metric Descriptions

Metric 1: Root Mean Squared Error (RMSE)

Description: RMSE measures the square root of the average squared differences between predicted and actual values. It provides a measure of prediction accuracy, with lower values indicating better performance.

Metric 2: Mean Absolute Error (MAE)

**Description

**: MAE calculates the average absolute differences between predicted and actual values. It is less sensitive to outliers compared to RMSE, providing a straightforward measure of prediction accuracy.

Metric 3: Huber Loss

Description: Huber loss combines RMSE and MAE, offering a balance between sensitivity to outliers and overall accuracy. It is particularly useful when dealing with datasets that contain outliers.

Metric 4: Mean Distance (mean_dist)

Description: The mean distance measures the average distance between predicted and actual geographic coordinates. Lower values indicate better predictive accuracy.

Metric 5: Median Distance (median_dist)

Description: The median distance represents the middle value of the distance distribution between predicted and actual geographic coordinates. It is less sensitive to extreme values compared to the mean distance.

Metric 6: Standard Deviation of Distance (stdev_dist)

Description: This metric measures the dispersion of distances between predicted and actual geographic coordinates. A lower standard deviation indicates more consistent prediction accuracy.

Metric 7: Kolmogorov-Smirnov Statistic (kolmogorov_smirnov)

Description: The Kolmogorov-Smirnov statistic quantifies the maximum difference between the empirical distribution functions of the predicted and actual distances. It assesses the goodness of fit between the two distributions.

Metric 8: Kolmogorov-Smirnov p-value (kolmogorov_smirnov_pval)

Description: This p-value indicates the statistical significance of the Kolmogorov-Smirnov test. Lower values suggest a significant difference between the distributions of predicted and actual distances.

Metric 9: Skewness (skewness)

Description: Skewness measures the asymmetry of the distance distribution between predicted and actual coordinates. Positive skewness indicates a longer tail on the right, while negative skewness indicates a longer tail on the left.

Metric 10: Spearman's Rank Correlation Coefficient (rho)

Description: Spearman's rho measures the strength and direction of the monotonic relationship between predicted and actual coordinates. Values close to 1 or -1 indicate a strong relationship.

Metric 11: Spearman's Rank Correlation p-value (rho_p)

Description: This p-value assesses the statistical significance of Spearman's rho. Lower values indicate a significant monotonic relationship between predicted and actual coordinates.

Metric 12: Spearman Correlation for Longitude (spearman_corr_longitude)

Description: This metric measures the Spearman correlation between predicted and actual longitude values. Higher values indicate a stronger correlation.

Metric 13: Spearman Correlation for Latitude (spearman_corr_latitude)

Description: This metric measures the Spearman correlation between predicted and actual latitude values. Higher values indicate a stronger correlation.

Metric 14: Spearman p-value for Longitude (spearman_pvalue_longitude)

Description: This p-value assesses the statistical significance of the Spearman correlation for longitude. Lower values indicate a significant correlation.

Metric 15: Spearman p-value for Latitude (spearman_pvalue_latitude)

Description: This p-value assesses the statistical significance of the Spearman correlation for latitude. Lower values indicate a significant correlation.

Metric 16: Pearson Correlation for Longitude (pearson_corr_longitude)

Description: This metric measures the Pearson correlation between predicted and actual longitude values. Higher values indicate a stronger linear relationship.

Metric 17: Pearson Correlation for Latitude (pearson_corr_latitude)

Description: This metric measures the Pearson correlation between predicted and actual latitude values. Higher values indicate a stronger linear relationship.

Metric 18: Pearson p-value for Longitude (pearson_pvalue_longitude)

Description: This p-value assesses the statistical significance of the Pearson correlation for longitude. Lower values indicate a significant linear relationship.

Metric 19: Pearson p-value for Latitude (pearson_pvalue_latitude)

Description: This p-value assesses the statistical significance of the Pearson correlation for latitude. Lower values indicate a significant linear relationship.

Metric 20: Mean Absolute Deviation Haversine (mad_haversine)

Description: This metric calculates the mean absolute deviation using the Haversine formula, which accounts for the curvature of the Earth. It measures the average absolute distance between predicted and actual coordinates.

Metric 21: Coefficient of Variation (coefficient_of_variation)

Description: The coefficient of variation is the ratio of the standard deviation to the mean distance. It provides a standardized measure of the dispersion of distances.

Metric 22: Interquartile Range (interquartile_range)

Description: The interquartile range measures the spread of the middle 50% of the distance distribution. It is calculated as the difference between the 75th and 25th percentiles.

Metric 23: 25th Percentile (percentile_25)

Description: This metric represents the 25th percentile of the distance distribution, indicating the value below which 25% of the distances fall.

Metric 24: 50th Percentile (percentile_50)

Description: This metric represents the 50th percentile (or median) of the distance distribution, indicating the middle value of the distances.

Metric 25: 75th Percentile (percentile_75)

Description: This metric represents the 75th percentile of the distance distribution, indicating the value below which 75% of the distances fall.

Metric 26: Percent Within 20km (percent_within_20km)

Description: This metric indicates the percentage of predicted coordinates that are within 20 kilometers of the actual coordinates. Higher values indicate better accuracy.

Metric 27: Percent Within 50km (percent_within_50km)

Description: This metric indicates the percentage of predicted coordinates that are within 50 kilometers of the actual coordinates. Higher values indicate better accuracy.

Metric 28: Percent Within 75km (percent_within_75km)

Description: This metric indicates the percentage of predicted coordinates that are within 75 kilometers of the actual coordinates. Higher values indicate better accuracy.

Metric 29: Mean Absolute Z-Score (mean_absolute_z_score)

Description: This metric measures the average absolute z-score of the distances between predicted and actual coordinates. It provides a standardized measure of how far the distances deviate from the mean in terms of standard deviations.

Glossary

Activation Function

Definition: A mathematical function applied to each neuron in a neural network to introduce non-linearity. Common activation functions include ReLU (Rectified Linear Unit), sigmoid, and tanh.

Backpropagation

Definition: A training algorithm used for neural networks, where the error is propagated backward through the network to update the weights. This process helps the network learn by minimizing the loss function.

Batch Normalization

Definition: A technique to improve the training of deep neural networks by normalizing the inputs of each layer. It helps to stabilize and speed up the training process.

Bootstrapping

Definition: A statistical method that involves resampling a dataset with replacement to create multiple simulated samples. This technique is used to estimate the variability of a statistic and to create confidence intervals for model predictions.

Confidence Intervals

Definition: A range of values derived from the sample data that is likely to contain the true value of an unknown population parameter. Confidence intervals provide a measure of the uncertainty associated with a sample estimate.

Convolutional Neural Network (CNN)

Definition: A type of deep learning model particularly effective for image and spatial data processing. CNNs use convolutional layers to automatically and adaptively learn spatial hierarchies of features.

Cross-Validation

Definition: A technique used to evaluate the performance of a machine learning model by dividing the data into several subsets and training/testing the model on different combinations of these subsets.

Dropout

Definition: A regularization technique used in neural networks to prevent overfitting. Dropout involves randomly setting a fraction of input units to zero at each update during training, which helps to prevent the network from becoming too reliant on specific nodes.

Early Stopping

Definition: A form of regularization used to prevent overfitting when training a machine learning model. Early stopping halts the training process if the model's performance on a validation set does not improve after a certain number of epochs.

Epoch

Definition: One complete pass through the entire training dataset. Training a machine learning model typically involves multiple epochs, with the model's weights being updated after each pass through the dataset.

Feedforward Neural Network

Definition: A type of artificial neural network where connections between the nodes do not form a cycle. It is the simplest form of neural networks.

Gradient Boosting

Definition: A machine learning technique used for regression and classification tasks, which builds an ensemble of weak prediction models (usually decision trees). Gradient boosting sequentially adds models to correct the errors of the combined ensemble.

Haversine Formula

Definition: A formula used to calculate the distance between two points on the surface of a sphere, given their latitude and longitude. The Haversine formula accounts for the curvature of the Earth, making it useful for geographic distance calculations.

Hyperparameter Optimization

Definition: The process of finding the best hyperparameters for a machine learning model. Hyperparameters are the parameters that are set before the learning process begins, such as learning rate, number of hidden layers, and dropout rate. Techniques such as grid search, random search, and Bayesian optimization are commonly used for hyperparameter tuning.

Imbalanced Sampling

Definition: A situation where some classes or categories are underrepresented or overrepresented in the training dataset. Imbalanced sampling can lead to biased models that perform poorly on the minority classes.

KMeans Clustering

Definition: A popular unsupervised learning algorithm used to partition a dataset into K clusters, where each data point belongs to the cluster with the nearest mean. KMeans clustering is used to group similar data points together.

Learning Rate

Definition: A hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. It determines the step size at each iteration while moving toward a minimum of the loss function.

Mean Absolute Error (MAE)

Definition: A metric used to measure the average absolute differences between predicted and actual values. MAE is less sensitive to outliers compared to other metrics like Root Mean Squared Error (RMSE).

Mendelian Inheritance

Definition: The basic principles of heredity established by Gregor Mendel, which describe how traits are passed from parents to offspring. Mendelian inheritance involves the segregation and independent assortment of alleles.

Minor Allele Count (MAC)

Definition: The count of the less common allele in a population. Setting a minimum MAC threshold helps to filter out rare variants that might introduce noise into the analysis.

Neural Network

Definition: A computational model inspired by the human brain, composed of layers of interconnected nodes (neurons). Neural networks are used for a variety of tasks, including classification, regression, and pattern recognition.

Optuna

Definition: A hyperparameter optimization framework designed to automatically find the best hyperparameters for machine learning models. Optuna uses techniques such as Bayesian optimization to efficiently search the hyperparameter space.

Detecting Outliers

Definition: The process of identifying and removing data points that deviate significantly from the rest of the dataset. Outliers can skew the results of analysis and lead to inaccurate predictions.

Overfitting

Definition: A modeling error in machine learning where the model learns the details and noise in the training data to the extent that it negatively impacts the model's performance on new data. Overfitting occurs when the model is too complex.

Principal Component Analysis (PCA)

Definition: A dimensionality reduction technique used to transform a large set of variables into a smaller set of uncorrelated variables called principal components. PCA is commonly used to simplify data, reduce noise, and visualize high-dimensional data.

Regularization

Definition: A technique used to prevent overfitting by adding a penalty to the loss function. Common forms of regularization include L1 (lasso) and L2 (ridge) regularization.

Root Mean Squared Error (RMSE)

Definition: A metric used to measure the square root of the average squared differences between predicted and actual values. RMSE provides a measure of prediction accuracy, with lower values indicating better performance.

Sampling Density

Definition: The concentration of samples in a given area. In geographic data, sampling density can vary across regions, leading to imbalanced datasets.

SMOTE (Synthetic Minority Over-sampling Technique)

Definition: A technique used to address imbalanced datasets by creating synthetic samples of the minority class. SMOTE generates new samples by interpolating between existing samples of the minority class.

Spearman's Rank Correlation Coefficient

Definition: A non-parametric measure of the strength and direction of the monotonic relationship between two variables. Spearman's rank correlation coefficient ranges from -1 to 1, with values close to 1 or -1 indicating a strong relationship.

Synthetic Oversampling

Definition: The process of generating synthetic data points to balance an imbalanced dataset. This technique helps to ensure that the model has sufficient data to learn from underrepresented classes or regions.

T-SNE (t-distributed Stochastic Neighbor Embedding)

Definition: A dimensionality reduction technique used for visualizing high-dimensional data. T-SNE transforms the data into a lower-dimensional space while preserving the relative distances between data points.

Underfitting

Definition: A modeling error in machine learning where the model is too simple to capture the underlying structure of the data. Underfitting occurs when the model has high bias and low variance.

Validation Split

Definition: The portion of the dataset set aside for evaluating the model's performance during training. The validation split helps to monitor the model's generalization ability and to detect overfitting.

Weighted Loss Function

Definition: A loss function that assigns different weights to different samples based on their importance. In GeoGenIE, the weighted loss function uses inverse sampling densities to focus more on areas with lower sample densities, balancing the influence of samples from different regions.

Xavier Initialization

Definition: A method of initializing the weights of a neural network to ensure that the variances of the input and output of each layer are equal. This helps to improve the convergence speed during training.

References

Akiba, Takuya, Shotaro Sano, Takeru Yanase, Toshihiko Ohta, and Masanori Koyama. 2019. "Optuna: A Next-Generation Hyperparameter Optimization Framework." In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–31. ACM.

Battey, C. J., P. L. Ralph, and A. D. Kern. 2020. "Predicting Geographic Location from Genetic Variation with Deep Neural Networks." *eLife* 9: e54507. https://doi.org/10.7554/eLife.54507.

Chang, Author et al. 2023. "GGoutlieR: An r Package to Identify and Visualize Unusual Geo-Genetic Patterns of Biological Samples." *Journal of Open Source Software* 8 (91): 5687. https://doi.org/10.21105/joss.05687.

Lemaître, Guillaume, Fernando Nogueira, and Christos K Aridas. 2017. "Imbalanced-Learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning." *Journal of Machine Learning Research* 18 (17): 1–5.