Geospatial ML: Feature Engineering

Geospatial Machine Learning

Many needs in Environmental Data Analytics can be effectively addressed using **statistical/geostatistics methods**. However, for more advanced needs, especially when dealing with large, complex datasets or integrating various data sources, **Geospatial Machine Learning (Geospatial ML)** becomes essential.

Geospatial ML refers to the application of machine learning techniques to data that contains geographic components (e.g., latitude and longitude). The data can take various forms, including tabular data, images (e.g., satellite or aerial imagery), point clouds, raster data, or vector data.

Geospatial ML excels at uncovering complex, non-linear relationships that geostatistical methods may not capture. Its ability to handle high-dimensional data and leverage rich geospatial features makes it ideal for tackling sophisticated environmental challenges beyond the capabilities of traditional geostatistics.

Geospatial ML employs many of the same **concepts, methods, and tools** as conventional ML, but it involves additional considerations in **feature engineering**, **model selection**, **model training**, and **interpreting model results** due to the spatial nature of the data. Among the various subfields of conventional ML, **spatial ML** and **image ML** share more commonalities with geospatial ML, as they also deal with spatial relationships and structured data. Consequently, models like **Convolutional Neural Networks (CNNs)**, which are commonly used in image ML, are frequently applied in geospatial ML for tasks like processing satellite imagery, land use classification, and object detection. Although geospatial ML involves more complex geographic factors such as coordinate systems, spatial resolution, and real-world distances.

Feature Engineering in Geospatial ML

Feature engineering is the process of selecting, modifying, or creating new features (predictors or variables - i.e., the data that the model uses to learn and make predictions) from raw data to improve a machine learning model's performance. Feature engineering essentially means selecting the best inputs for a machine learning model.

Feature engineering is essential because selecting the most appropriate inputs isn't always intuitive, and there may be numerous potential features that could influence the

prediction. Feature engineering helps identify and retain the most relevant features, reducing input dimensionality and making training more efficient. Furthermore, raw data often contains irrelevant or redundant information, and feature engineering aims to eliminate these elements, which not only speeds up training but also reduces the risk of overfitting, ultimately improving the model's accuracy and reliability.

For example, in predicting wildfire risk using remote sensing maps, there is a wide range of potential inputs, such as vegetation type, soil moisture, temperature, wind speed, humidity, topography, and historical fire occurrence. Each of these could potentially influence wildfire risk, but not all of them may be equally informative, and some might even introduce noise if left unfiltered. Through feature engineering, we can focus on the most impactful features and prepare them in ways that make them more useful to the model. For instance, we might select vegetation type, soil moisture, and temperature as key predictors after analyzing their correlation with past fire events. We can also create interaction features, like combining temperature and humidity to capture their combined effect on fire risk. Additionally, we might compress highly detailed topographic data into simpler elevation or slope variables, which the model can more easily interpret. This careful feature selection and transformation process tailors the inputs, ensuring that only the most relevant and refined information is fed into the model, making wildfire prediction both more efficient and accurate.

Historically, feature engineering was a **manual** task where data scientists relied heavily on domain expertise to select and transform features that would enhance model performance. However, with the advent of deep neural networks, models have gained the ability to **automatically** learn relevant features from raw data. These models can capture complex patterns and representations directly, without manual intervention—a process known as **feature learning**, where the model discovers and refines the most useful features on its own. This capability has been transformative, particularly for high-dimensional and unstructured data like images, text, and audio.

However, there are still limitations: deep neural networks require vast amounts of data and computational resources to learn effectively, and they may not automatically capture certain **domain-specific** nuances. For many applications, particularly those with smaller datasets or unique domain requirements, manual feature engineering remains important, and it’s likely to stay essential for achieving optimal results in various contexts including environmental data analytics.

Spatial Feature Engineering

Spatial feature engineering is a specialized subset of feature engineering focused on creating features from data with spatial or geographical components. It emphasizes leveraging spatial relationships—such as proximity, neighborhood effects, or spatial

autocorrelation—while using location-specific tools and techniques to extract meaningful insights for location-dependent applications

At its core, spatial feature engineering is the process of developing additional information from raw data using *geographic knowledge*. This distilling of information can occur *between* datasets, where geography is used to link information in separate datasets together; or *within* datasets, where geography can be used to augment the information available for one sample by borrowing from nearby ones.

For cases where geography connects *different* datasets, we adopt the term "**Map Matching**"; while we use the mirroring concept of "**Map Synthesis**" describing the use of geographical structure to derive new features from a given dataset. Technically speaking, some of the methods we review are similar across these two cases, or even the same; however, they can be applied in the context of "matching" or "synthesis".

Key steps and techniques in feature engineering, which are common to all kinds of ML contexts, including environmental data analytics, include:

Feature Selection

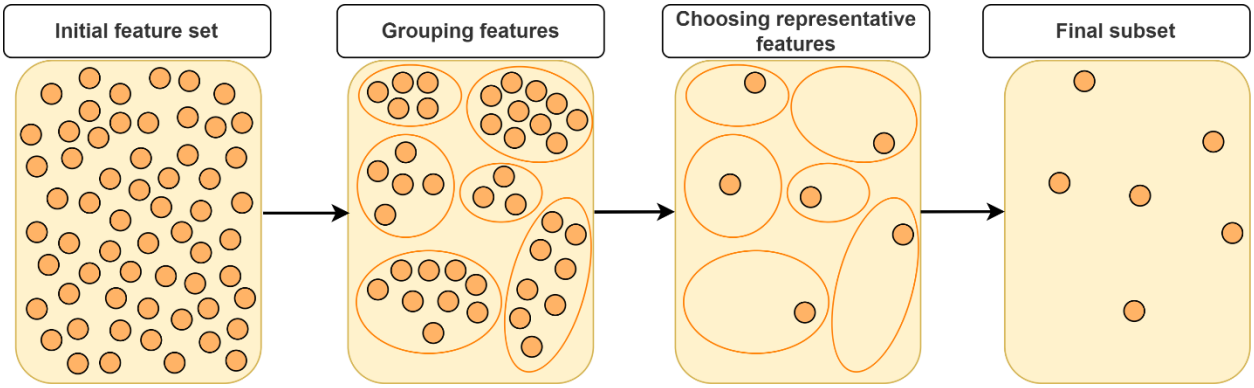
Feature selection is the process of choosing the most relevant inputs (features) for a model while discarding those that may be redundant, irrelevant, or noisy, helping the model focus on high-quality data for better performance.

For example, in remote sensing or geospatial data, we may have multiple bands (e.g., red, green, blue, near-infrared) and additional data layers like elevation, soil moisture, and land cover type. Suppose the goal is to predict vegetation health. Using feature selection, we might find that near-infrared and red bands, combined with soil moisture, are highly predictive of vegetation health, while the blue band or elevation data has little effect. By keeping only these relevant features, the model can focus on the most impactful data, reducing noise and potentially speeding up processing. There are a wide range of methods for feature selection, here we focus on two popular ones:

1) Cluster-based Feature Selection:

Cluster-based Feature Selection is a technique used to reduce feature redundancy by grouping similar features into clusters and then selecting representative features from each group. This process begins by measuring the similarity between features, often using correlation or mutual information, and then clustering them using algorithms like hierarchical clustering or k-means. Within each cluster, a representative feature is chosen—typically the feature that is most central, has the highest variance, or shows the strongest relationship with the target variable. By selecting one feature from each

cluster, this method ensures that the final subset retains the diversity of information across clusters while eliminating redundant features. This approach is especially useful in datasets where high correlations between features can lead to overfitting or inefficient model training.



Cluster-based Feature Selection

2) Recursive Feature Elimination

Recursive Feature Elimination (RFE) is a feature selection method that iteratively removes the least important features to find an optimal subset for model performance. It begins by training a model on the full set of features. The importance or contribution of each feature is then assessed, typically using importance scores derived from the model. The least important feature(s) are removed, and the model is retrained on the reduced feature set. This process repeats recursively until a specified number of features remain or model performance no longer improves. RFE is effective in selecting a compact and relevant feature subset by systematically eliminating features that contribute less to the model's predictive accuracy.

An **importance score** is a metric that quantifies the contribution or influence of each feature in a model. It reflects how much a feature improves the model's predictive accuracy and is commonly used in feature selection to rank and prioritize features. In deep neural networks (DNNs), calculating importance scores is more complex due to the model's non-linear structure and interactions between features across multiple layers.

One common method to compute importance scores in DNNs is **Gradient-Based Analysis**. By computing the gradient of the output with respect to each input feature, we can assess the sensitivity of the model's predictions to changes in each feature. High

gradients indicate that small changes in a feature have a strong effect on the output, suggesting higher importance.

Feature Combination

Feature combination is the process of creating new features by combining existing ones to capture more complex relationships in the data, often revealing patterns that individual features alone cannot show.

For example, in remote sensing or geospatial data, suppose we want to predict wildfire risk. Individual features like temperature, humidity, and wind speed each contribute to fire risk, but combining these into new features can make this risk clearer. We might create a new feature, like "dryness index" by combining temperature and humidity, or "fire spread potential" by combining wind speed with the dryness index. These combined features help the model understand complex environmental factors that increase fire risk, potentially improving predictive accuracy.

Methods for feature combination include the following among others:

- **Multiplying or Dividing Features (e.g., Ratios)**: Ratios can reveal relationships between environmental factors. For instance, dividing total annual rainfall by the number of rainy days provides an average rainfall per rainy day, which can indicate rainfall intensity. Similarly, dividing pollutant concentration by population density gives pollutant exposure per capita, providing insight into air quality relative to the local population.
- **Adding or Subtracting Values**: Adding or subtracting environmental values can yield aggregate or difference metrics. For example, adding daytime and nighttime temperatures produces a combined temperature measure, useful for assessing average thermal conditions. Subtracting the concentration of a pollutant in two consecutive years shows the change in pollution levels, helping track improvements or deteriorations in air quality over time.
- **Creating Interaction Terms (e.g., Polynomial Features)**: Interaction terms, like polynomial features, capture non-linear relationships in environmental data. For example, creating a feature by squaring wind speed (wind speed^2) can reflect its impact on erosion potential, as stronger winds have a disproportionate effect. Similarly, combining features, like multiplying temperature and humidity, can help model factors like heat index, a measure of perceived heat that combines temperature and moisture effects.

Dimensionality Reduction

Dimensionality reduction is a process that reduces the number of input features by transforming high-dimensional data into a lower-dimensional form while retaining as much essential information as possible. Unlike feature selection, which directly removes less relevant features, dimensionality reduction transforms the data into a new space, often using mathematical techniques to capture the most significant patterns.

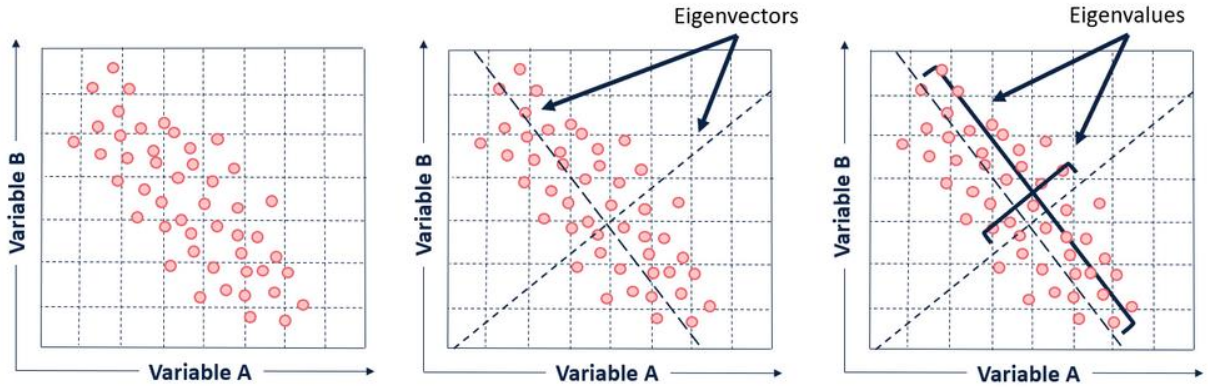
Principal Component Analysis (PCA) is a popular dimensionality reduction technique commonly used in data analysis and machine learning to simplify large datasets by transforming them into a smaller set of **uncorrelated variables**, known as **principal components**. The goal of PCA is *to capture as much variance in the data as possible with fewer features*, making the data easier to visualize and analyze without losing significant information.

PCA is popular in geospatial feature engineering. In geospatial contexts, data often includes many correlated features collected over extensive areas and time periods, making PCA a valuable tool for reducing complexity while retaining essential spatial patterns and relationships. A common example of PCA use cases is Multispectral and Hyperspectral Imaging. In remote sensing, PCA is frequently applied to satellite or aerial imagery, which often includes multiple spectral bands. By reducing the number of bands to a few principal components, PCA simplifies the data, making it easier to analyze without losing critical information

This is **how PCA works:**

1. **Standardize the Data**: PCA starts by making sure that each feature in the data is on a similar scale. For example, if we have features like temperature (in degrees) and altitude (in meters), we don't want the large numbers in altitude to dominate the results. To prevent this, we transform each feature so that it has a mean of zero and a standard deviation of one. This allows all features to contribute equally to the analysis.
2. **Compute the Covariance Matrix**: Next, we calculate the **covariance matrix**, which helps us understand how different features vary in relation to each other. The covariance matrix is a table that shows the relationships between all pairs of features, indicating whether they change together in a similar (positive covariance) or opposite (negative covariance) way. This step is crucial for identifying patterns and correlations within the dataset.
3. **Calculate Eigenvectors and Eigenvalues**: Once we have the covariance matrix, we compute its **eigenvectors** and **eigenvalues**. Eigenvectors represent the **directions** of maximum variance in the data, essentially showing the most important patterns. Eigenvalues tell us how much **variance** each eigenvector

explains. So, an eigenvector with a high eigenvalue represents a major trend in the data, while one with a low eigenvalue represents a less important trend.



Eigenvectors and eigenvalues in Principal Component Analysis (PCA).

4. **Select Principal Components**: After ranking the eigenvectors by their eigenvalues from highest to lowest, we choose the top principal components—those with the largest eigenvalues. These components capture the most important patterns in the data and contain the bulk of the information. Usually, we select enough components to explain a significant portion (e.g., 95%) of the total variance, which allows us to reduce dimensions while keeping most of the data's meaningful structure.
5. **Transform the Data**: Finally, we project the original data onto the selected principal components. This step creates a new dataset where each data point is now represented by its values along these principal components, instead of the original features. This transformed data has fewer dimensions but still captures the essential information, making it easier to analyze and visualize.

Feature Extraction

Feature extraction is the process of creating new features by transforming raw data into informative representations, often in cases where raw data is high-dimensional or complex (e.g., images, text, or audio). It involves generating meaningful features that summarize or capture essential information from the data, which can improve model performance. Unlike feature selection, which simply chooses existing features, feature extraction focuses on creating entirely new representations from the raw data.

In spatial feature engineering two types of feature extraction techniques are common:

Spatial summary feature which aggregates information from a spatial area to produce a single metric for a location. For example, taking the average income within a city block or the total vegetation cover in a specific radius around a point creates spatial summary features. These features help capture the overall characteristics of an area or its immediate surroundings.

Key categories of spatial summary features include:

- **Zonal Statistics Features**: These summarize values within defined spatial zones, such as mean, median, standard deviation, or sum. For example, average elevation, mean rainfall, or population density over a defined area. Zonal statistics are broadly used in fields like urban planning, hydrology, and ecology.
- **Spatial Density Features**: These features capture the density of objects or events within a specified area, like population density or the density of trees in a forest. Density features are applicable in fields like ecology, public health (e.g., disease spread), and urban studies.
- **Spatial Autocorrelation Features**: Measures like Moran's I capture how similar values are spatially clustered or dispersed. These features help in identifying patterns, such as clustering of high-pollution areas, and are widely used in environmental studies, economics, and crime analysis.
- **Edge and Boundary Features**: These focus on the properties of boundaries or transitions between different spatial areas, such as land cover edges or coastlines. Boundary features are useful in studying ecological zones, urban-rural boundaries, and habitat fragmentation across fields like geography, environmental science, and conservation.
- **Texture features** are a popular type of spatial summary feature in remote sensing. In remote sensing, texture features summarize the spatial patterns of pixel values within a specified area, capturing characteristics like roughness, smoothness, or variability. This aligns with the concept of spatial summary features, as texture features aggregate information from a defined spatial area to provide a single descriptive metric that represents the surface's overall texture within that region. The resulting texture features. for example, enhance the ability to differentiate between land cover types (e.g., forests, urban areas, water bodies).

A **proximity feature**, on the other hand, measures the distance or closeness of a location to specific landmarks or points of interest, such as the distance from a home to the nearest school, hospital, or public transportation hub. This type of feature directly assesses the accessibility or spatial relationship of a point to key locations.

The primary difference between spatial summary features and proximity features lies in their focus: spatial summary features describe the properties of an area around a point, while proximity features measure the closeness of a point to specific, targeted locations. Both types are valuable in spatial feature engineering but serve distinct analytical purposes.

Key categories of proximity features include:

- **Distance-Based Features**: These measure the straight-line (Euclidean) or network distance from one point to another. This concept is widely used to capture accessibility in fields like urban planning (distance to amenities), public health (distance to hospitals), and ecology (distance to water sources or habitats). The core idea of measuring distance between points is generalizable across contexts and data types.
- **K-Nearest Neighbor Features**: Instead of a single distance, this feature type identifies the closest *k* locations or objects (e.g., nearest cities, nearest pollutant sources) and can summarize distances or properties of these neighbors. Used in EDA for pollution analysis, socio-economic studies, and biodiversity assessments, K-nearest neighbor features capture immediate spatial relationships and influence in diverse applications.
- **Buffer-Based Features**: Buffering creates a zone around a point or area to calculate or summarize attributes within a specific proximity. This is commonly applied in environmental studies (e.g., buffers around pollution sources), real estate (amenities within a certain radius), and retail (customer density in store buffers). Buffer-based features provide insights into surrounding conditions and influence in various fields.
- **Gravity Model-Based Features**: These features measure the "attractiveness" or influence of one location over another based on distance and feature properties (e.g., population, size). Widely used in transportation, retail site selection, and ecology, gravity-based features capture both distance and influence, making them adaptable to different fields.

Feature Transformation

Transformation is the process of applying mathematical functions to features to make data easier for models to interpret, often by reducing skewness, handling outliers, or standardizing feature scales. Common transformations include **logarithmic transformations** (to reduce the impact of large values), **square root transformations** (to stabilize variance), and **scaling or normalization** (to put all features on a similar scale).

For instance, in remote sensing data, suppose we have a feature representing vegetation cover with a wide range of values. If most of the data is concentrated in lower values but a few areas have very high vegetation density, a log transformation can reduce this skewed distribution, making patterns more uniform and easier for the model to understand. Similarly, standardizing features like elevation or reflectance values across spectral bands can help the model treat each input fairly, improving performance.

Coordinate Encoding and Normalization

Coordinate encoding and normalization transform raw latitude and longitude values into features that are more interpretable and effective for machine learning models. Since longitude values repeat every 360 degrees and latitude has physical constraints (bounded between -90 and 90 degrees), these geographic coordinates often need special handling to maintain their spatial relationships. Here are some key methods

1) Cyclic Encoding (Sinusoidal or Trigonometric Encoding)

Latitude and longitude are circular in nature; for example, the longitude of -180 degrees is equivalent to +180 degrees, meaning there's a wraparound effect. Directly using these values in a model can lead to incorrect distance and direction relationships because 180 and -180, though numerically distant, are geographically adjacent.

- **Solution**: Transform latitude and longitude into cyclical features using sine and cosine transformations. This encoding helps maintain the periodic or wraparound nature of these coordinates, which can be particularly important for capturing spatial patterns or proximities.

- **Latitude Encoding**:

$$\text{latitude\_sin} = \sin\left(\frac{\pi \times \text{latitude}}{180}\right)$$

$$\text{latitude\_cos} = \cos\left(\frac{\pi \times \text{latitude}}{180}\right)$$

- **Longitude Encoding**:

$$\text{longitude\_sin} = \sin\left(\frac{\pi \times \text{longitude}}{180}\right)$$

$$\text{longitude\_cos} = \cos\left(\frac{\pi \times \text{longitude}}{180}\right)$$

In cyclic encoding, each coordinate (latitude and longitude) is transformed into two separate values, one for the sine (_sin) and one for the cosine (_cos).

2) Scaling and Normalization

Latitude and longitude are on different ranges (latitude between -90 and 90, longitude between -180 and 180), which can lead to scaling issues in models sensitive to feature magnitude (e.g., distance-based methods like K-nearest neighbors).

- **Solution**: Normalize both latitude and longitude to a consistent scale, typically between 0 and 1, or standardize them to have zero mean and unit variance. This step ensures that both coordinates contribute comparably to model predictions and that no single coordinate overwhelms the distance metric.

For example:

- **Min-Max Scaling**:

$$\text{latitude\_scaled} = \frac{\text{latitude} + 90}{180}$$

$$\text{longitude\_scaled} = \frac{\text{longitude} + 180}{360}$$

Encoding Categorical Variables

Encoding categorical features is the process of converting categorical data (e.g., labels or categories) into numerical values that a machine learning model can understand.

Common methods include **one-hot encoding** (creating binary columns for each category, where 1 indicates presence and 0 indicates absence) and **label encoding** (assigning a unique integer to each category).

For example, in geospatial or remote sensing data, suppose we have a feature representing land cover type with categories like "forest," "urban," and "water." Using one-hot encoding, we would create three new features—one for each type—so that each area is marked as either forest, urban, or water. This approach allows the model to work with categorical data more effectively, treating land cover as a distinct factor while still handling it in numerical form.

Feature Binning

Feature binning is the process of dividing continuous data into discrete intervals, or "bins," to simplify analysis and help models detect broader patterns within ranges rather than exact values. This technique can be particularly useful for managing outliers or creating categorical levels from numerical data.

For example, in geospatial or remote sensing data, suppose we have a feature for elevation that ranges from 0 to 3000 meters. Instead of using exact elevation values, we could create bins like "low" (0–1000m), "medium" (1000–2000m), and "high" (2000–3000m). By grouping elevation this way, the model can recognize general elevation patterns across regions, which may simplify its understanding and improve predictions when elevation only matters within these broader categories.