A Machine Learning-Aware Data Re-partitioning Framework for Spatial Datasets

Kanchan Chowdhury
Arizona State University
Tempe, USA
kchowdh1@asu.edu

Venkata Vamsikrishna Meduri
Arizona State University
Tempe, USA
vmeduri@asu.edu

Mohamed Sarwat
Arizona State University
Tempe, USA
msarwat@asu.edu

***Abstract*—Spatial datasets are used extensively to train machine learning (ML) models for applications such as spatial regression, classification, clustering, and deep learning. Most of the real-world spatial datasets are often too large, and many spatial ML algorithms represent the geographical region as a grid consisting of several spatial cells. If the granularity of the grid is too fine, that results in a large number of grid cells leading to long training time and high memory consumption issues during the model training. To alleviate this problem, we propose a machine learning-aware spatial data re-partitioning framework that substantially reduces the granularity of the spatial grid. Our spatial data re-partitioning approach combines fine-grained, adjacent spatial cells from a grid into coarser cells prior to training an ML model. During this re-partitioning phase, we keep the information loss within a user-defined threshold without significantly degrading the accuracy of the ML model. According to the empirical evaluation performed on several real-world datasets, the best results achieved by our spatial re-partitioning framework show that we can reduce the data volume and training time by up to 81%, while keeping the difference in prediction or classification error below 5% as compared to a model that is trained on the original input dataset, for most of the ML applications. Our re-partitioned framework also outperforms the state-of-the-art data reduction baselines by 2% to 20% w.r.t. prediction and classification errors.**

***Index Terms*—Spatial Machine Learning, Spatial Adjacency, Spatial Data, Data Partitioning**

I. Introduction

Spatial datasets are prominently used to train and test machine learning (ML) models in the context of various applications [1]–[4]. For example, a data scientist who might want to analyze and compare the housing prices across several regions within a country can make use of spatial machine learning (spatial ML) in the form of a regression model, which can help predict the housing prices in various regions. It is important that such models are learned from several training samples (spatial regions and their corresponding housing prices in this example) to yield a competent prediction quality.

Although large-scale training data is beneficial in learning spatial ML models of high accuracy, it creates bottlenecks such as long training time and high memory consumption [5], [6]. The training time may range from several hours to even days, thereby incurring a long waiting time for the data scientists who need to analyze the prediction results. For example, it takes 14 days to train 90 epochs of ResNet-50 model with the ImageNet-1k dataset on Nvidia M40 GPU [5]. In the case of our prior housing example, if we want to learn a spatial ML model of high quality, it needs to be trained on a spatial grid of fine-grained cells corresponding to smaller partitions of geographical space. This is because a competent spatial ML model needs to differentiate between the localities of the housing regions at a high granularity to make accurate predictions. To empirically validate our claim of long training time, we trained a support vector regression model on the Washington home sales dataset [7], which consists of $\approx 100K$ spatial cells. Our experiment took more than one day to finish the training (details are in Section IV).

For the purpose of reducing the volume of spatial datasets, spatial sampling techniques [8]–[12] have been used effectively in the literature with various applications. Although sampling techniques are effective in reducing the size of the training data, they cannot capture the property of spatial autocorrelation, which is a significant drawback. Spatial autocorrelation essentially refers to the similarity in the attribute values among the closely located regions. For example, closely located regions may have similar housing prices (details of spatial autocorrelation are in Section II). Spatial ML applications exploit this autocorrelation property by the use of the neighborhood information among geolocations. The selection of a subset of samples by sampling techniques breaks this adjacency information. For example, if a cell in a grid has four adjacent cells, the sampling technique might pick the cell without picking most of its adjacent cells affecting the adjacency information in the adjacency matrix resulting in the poor predictive performance of the spatial ML models.

Another category of works, known as spatially constrained clustering or regionalization techniques [13]–[18], aggregate the polygons in a spatial dataset into a set of adjacent regions satisfying various user constraints. These techniques perform the aggregation in two phases: initialization phase and region growing phase. If the goal is to cluster n polygons into p regions, the initialization phase initializes p regions randomly with p polygons. Later, in the region growing phase, each of these p regions is grown by assigning adjacent unassigned polygons to the corresponding region. Regionalization techniques are applied to cluster zones into several groups of zones where p is much less than n [19]–[22]. To the best of our knowledge, no previous studies applied regionalization techniques to training datasets for the purpose of reducing the

training time of spatial ML models. Clustering and regionalization techniques suffer from the following disadvantages. i) Users need to pre-specify the number of regions or clusters which can be unintuitive and infeasible in the case of large training datasets. ii) Each region consists of a set of polygons with an arbitrary shape which makes the computation of the adjacency matrix cumbersome. iii) Each polygon is always assigned to one of the adjacent regions, although a single polygon might need to be considered as a separate region if it differs significantly from adjacent regions. iv) These techniques most often end up in suboptimal solutions because of their sensitivity to the selection of initial regions [23].

To alleviate these concerns, we propose a re-partitioning framework for spatial datasets by utilizing the idea of spatially constrained clustering or regionalization which reduces the number of spatial partitions in the dataset while maintaining the total loss of information within a user-defined threshold. To the best of our knowledge, this is the first work to attempt attenuating the long training time and high memory consumption issues of training a spatial ML model by adopting a spatial data re-partitioning approach. Our spatial data re-partitioning approach works by - 1) iteratively merging the adjacent cells into cell-groups such that the cells within a group have similar attribute values, 2) and keeping the information loss that is incurred by our re-partitioning framework under a user-specified threshold. While the merging property only allows highly similar, as well as proximal cells to be grouped together, the loss constraint ensures that the downstream spatial ML algorithm trained upon these re-partitioned datasets does not suffer from loss in prediction quality.

We call our framework ML-aware because it overcomes the limitations of existing data reduction techniques in supporting the training of spatial ML models. Our re-partitioned dataset can capture spatial autocorrelation unlike existing sampling techniques because the neighborhood relationship among cell-groups can be represented by an adjacency list which we describe in Section III-B. It has the following advantages over the regionalization techniques. i) We do not need the specification of the number of regions. Instead, the user is asked to provide a numerical loss threshold between 0 and 1, which is the target dissimilarity between the re-partitioned and the original spatial grid. This marks an intuitive trade-off as low loss thresholds result in high quality (low dissimilarity) at the expense of long training times, whereas high thresholds can reduce the training time while also yielding low quality or high dissimilarity. ii) Our merged cell-groups always maintain a rectangular shape which enables a concise mapping between cell-groups and their constituent cells and an easy representation of the cell-groups and the corresponding adjacency matrix.

Fast computation of adjacency speeds up training spatial ML models such as spatially constrained hierarchical clustering, which need to calculate the adjacency in every iteration of the algorithm. Another advantage of maintaining rectangular cell-groups is that spatial ML models such as spatial kriging require the coordinates of spatial objects to be a part of the feature vector. Rectangular cell-groups facilitate the creation of feature vectors because a rectangle can be represented by a fixed number of vertices. iii) Our hierarchical merging algorithm does not merge cells with high dissimilarity to the nearby cells. It should be noted that dimensionality reduction techniques [24]–[26] are orthogonal to our approach because they focus on reducing the #attributes in the dataset and not the #data instances or #spatial grid cells in the context of spatial ML. Since our spatial re-partitioning framework does not reduce the #attributes in a dataset, it can be used in conjunction with dimensionality reduction for an enhanced reduction in model training time.

We use conservatively low thresholds of information loss in our experiments which preserves the high accuracy of an ML model while also reducing the data volume significantly. Our experiments with various low thresholds of information loss study the trade-off between training time and accuracy. Our empirical evaluation shows that our spatial re-partitioning framework reduces the training time by up to 81% without significantly increasing prediction error and outperforms the prediction quality of the state-of-the-art baselines by 2% to 20% for a variety of spatial ML models.

Following are our high-level contributions:

- To the best of our knowledge, this is the first paper to adopt a data re-partitioning approach to reduce the long training time and memory usage of spatial ML models.
- We propose an ML-aware spatial data re-partitioning framework, which maintains the spatial adjacency in order to prevent the loss of spatial autocorrelation. It reduces the data volume while losing information within a threshold.
- We evaluate our proposed framework on four real-world datasets and compare it against three state-of-the-art baselines. The experimental results show that our framework can reduce training time and memory usage of spatial ML models by up to 81% and 65%, respectively, while keeping the loss of accuracy below 5%.
- Our framework outperforms baselines by up to 20% in terms of prediction and classification errors.

II. Background

We first discuss a few necessary terms before proceeding with the details of our proposed framework.

Spatial Cell: - We represent the geographical space as a two-dimensional $m \times n$ grid by dividing the latitudes and longitudes into 'm' and 'n' equi-sized intervals, respectively. Each resulting rectangular unit is termed as a spatial cell. *The input grid fed to the re-partitioning framework in Figure 1 shows the cellular structure of a spatial dataset consisting of* 5×5 *cells where* lat_1, lon_1 *is an example cell.*

Univariate and Multivariate Datasets: - If the schema of a dataset consists of only one attribute, it is called a univariate dataset. A multivariate dataset schema consists of two or more attributes. Each univariate grid cell is represented by a uni-dimensional feature vector (FV), whereas a multivariate grid cell is represented by a p-dimensional FV where p is #attributes and each feature dimension is the attribute value. The feature vector (FV) of a spatial cell is derived by applying

aggregation operators such as AVG on the FVs of the data instances mapped to the cell.

Cell-Group: - A group of cells is termed as a cell-group if and only if every cell in the group is adjacent to at least one more cell in the same group. *In the input grid dataset shown in Figure 1,* $\{(lat_1, lon_1)$*,* (lat_1, lon_2)*,* (lat_2, lon_1)*,* $(lat_2, lon_2)\}$ *is an example cell-group.*

Attribute Normalized Data: - A dataset is deemed to be attribute normalized if the instance values of each attribute in the dataset lie in the range [0, 1]. For an example dataset consisting of 2 attributes and 3 instances: (10, 15), (20, 20), and (30, 10), its attribute normalized variant would be (0.33, 0.75), (0.67, 1.0), and (1.0, 0.5).

Attribute Variation Between Cells: - Attribute variation between two cells in a grid dataset is the total pair-wise absolute difference between the attribute values of the corresponding cells, averaged across the #attributes. Variation between cells i and j, $Variation_{ij}$, is defined as:

$$Variation_{ij} = \frac{1}{p} \sum_{k=1}^{p} |d_i(k) - d_j(k)| \tag{1}$$

where p and $d_i(k)$ denote #attributes and value of attribute k for cell i, respectively.

Local Loss of Cell-Groups: - If a cell-group cg consists of t cells $(cell_1, cell_2, .., cell_t)$, local loss of cg for attribute k, denoted by $Loss_{cg}(k)$, is defined as:

$$Loss_{cg}(k) = \frac{1}{t} \sum_{i=1}^{t} |d_i(k) - cg(k)| \tag{2}$$

where $d_i(k)$ and $cg(k)$ denote the value of attribute k for cell i and cell-group cg, respectively. The attribute values of the cell-groups are obtained by applying our spatial re-partitioning framework or a baseline approach upon the input spatial grid.

Information Loss: - We abbreviate the information loss between the original and the re-partitioned dataset as IFL. We define IFL in terms of mean absolute percentage error defined in 2D-STR [27]. Given a spatial grid dataset d and its re-partitioned form $\bar{d}$, mathematically, IFL is represented as:

$$IFL(d, \bar{d}) = \frac{1}{n * m} \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{|d_i(j) - \bar{d}_i(j)|}{d_i(j)} \tag{3}$$

where n denotes the total number of spatial cells in the original dataset, m is #attributes in the dataset, i refers to a single cell having a valid (not null) feature vector in the input dataset, $d_i(j)$ is the value of attribute j in the feature vector of cell i, $\bar{d}_j$ is the representative value of attribute j of cell i in the re-partitioned dataset.

Spatial Autocorrelation: - Spatial autocorrelation denotes the degree to which the attribute values (such as housing price) of a set of geographical locations (which can be cells or cell-groups) are similar to each other. It can be defined by statistical measures such as Moran's I [28] and Geary's C [29]. Moran's I is frequently used in the literature and is represented as:

$$I = \frac{N}{\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_i (x_i - \bar{x})^2} \tag{4}$$

where N refers to #geographical locations, w_{ij} is 1 if locations i and j are adjacent to each other and 0 otherwise, x is the attribute of interest, and $\bar{x}$ is the mean of x.

Problem Statement: - Given an $m \times n$ grid dataset d (univariate or multivariate) consisting of $m * n$ spatial cells and a user-specified threshold θ of information loss (*IFL*), we return a re-partitioned dataset $\bar{d}$ of t cell-groups such that $t < m * n$, and t is minimized while information loss $IFL(d, \bar{d}) \leq \theta$.

III. Our Solution

In this section, we elaborate on our machine learning-aware spatial data re-partitioning algorithm. The input to the re-partitioning algorithm is the raw spatial grid dataset, and the output of the algorithm is a new and re-partitioned version of the input spatial dataset.

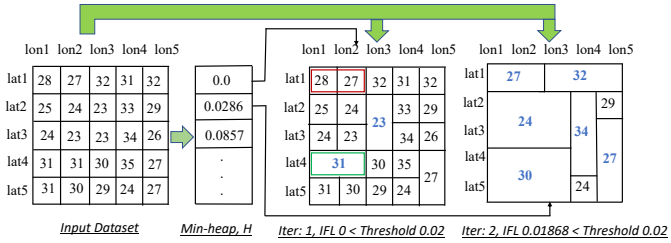


Fig. 1: Example of spatial univariate data re-partitioning

Example 1. *Figure 1 depicts an example of how our re-partitioning framework can re-partition a spatial grid dataset. The input spatial grid in our example consists of* 5×5 *(25) cells, which is converted to a dataset of* 8 *cell-groups after our re-partitioning framework is applied for two iterations. Although our proposed re-partitioning framework works for both univariate and multivariate datasets, we use a univariate dataset of only one attribute for ease of explanation.*

This section elaborates on our re-partitioning methodology at first and then, detail how to use the re-partitioned dataset to train spatial ML models. Finally, we discuss how a naïve variant of our re-partitioning solution can be designed to produce homogeneous partitions and state its pros and cons.

A. Re-partitioning Methodology

Spatial data re-partitioning is an iterative algorithm based on grouping the cells together such that the attribute variation (see Equation 1) among the cells within a group is kept to a minimum. The extent of variation among the cells can be adjusted based on a min-adjacent variation (details will be discussed in Section III-A1) that is pre-computed. We require that all the cells within a group have their cumulative attribute difference under the pre-computed min-adjacent variation. We get several smaller partitions of high granularity if the min-adjacent variation is low and fewer large partitions of low granularity if the min-adjacent variation is high. In each iteration of our re-partitioning algorithm, we pick a different min-adjacent variation that is higher than the variation we chose in the previous iteration. This gives us increasingly relaxed partitions of larger sizes with more iterations.

Figure 2 depicts the system architecture of our re-partitioning framework. Our system picks an updated value of the min-adjacent variation in each iteration. Subsequently, *Cell-Group Extractor* finds the rectangular groups of adjacent cells where the variation among the cells within the group is at most the value of the min-adjacent variation. Once we retrieve all such rectangular cell-groups, each group of cells acts as a single cell in the new dataset, the details of which are discussed in Section III-A2. After that, *Feature Allocator* creates feature vectors for the cell-groups which is elaborated in Section III-A3. Finally, we calculate the information loss between the input dataset and the new re-partitioned dataset, which will be discussed in section III-A4. We proceed to the next iteration if the information loss is less than a user-specified threshold and exit otherwise.

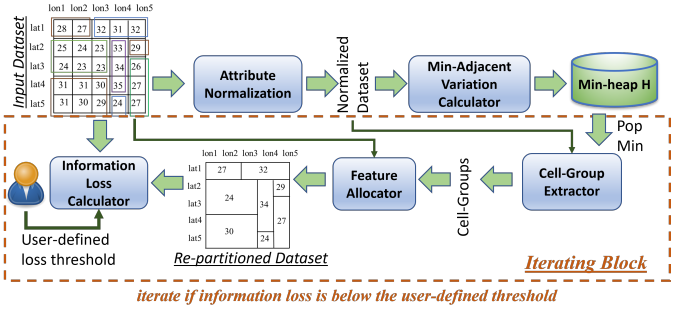


Fig. 2: System architecture of the proposed framework

1) Min-Adjacent Variation Calculator: As we have discussed before, spatial cells are binned to a cell group if the attribute variation among them is below the min-adjacent variation. Our framework pre-computes the variation values between all possible pairs of adjacent cells and stores them in a min-adjacent variation heap data structure (min-heap), H. The input dataset is converted to an attribute normalized form before computing the variation values. This is because, in a multivariate dataset, the ranges of some of the attributes might differ significantly from those of other attributes. If we use unnormalized attribute values, the variation will be dominated by attributes having high ranges of values. We pre-compute H exactly once at the beginning, and in each re-partitioning iteration, we pop the root node from H and use it as the updated min-adjacent variation ($minAdjacentVariation$).

Example 2. *For the input dataset shown in Figure 1, the value of* $minAdjacentVariation$ *in the first iteration is 0 because it is the least possible variation among all pairs of adjacent cells (variation between normalized values of (lat3, lon2) and (lat3, lon3)). In the second iteration,* $minAdjacentVariation$ *will be* 0.02857143*, which is the second-least variation and happens to be the variation between normalized values of pairs (lat1, lon1) and (lat1, lon2).*

2) Cell-Group Extractor: Similarly to min-adjacent variation calculator, attribute-normalized form of the input dataset is sent as input to this cell-group extractor. After calculating $minAdjacentVariation$, our next target is to find all groups of adjacent cells where all adjacent pairs of cells within the same group have a variation $\leq minAdjacentVariation$. All possible pairs of cells within a cell-group need not maintain the $minAdjacentVariation$, but all possible adjacent pairs of cells should maintain this variation. If a cell does not have any other adjacent cell with variation $\leq minAdjacentVariation$, that cell alone forms a cell-group. A cell with null feature vector forms a cell-group with other adjacent null cells.

Although our aim is to minimize the number of partitions that can be formed with the $minAdjacentVariation$, the optimal solution to this problem has to enumerate an exponential search space. Therefore, we propose a greedy heuristic which can use any possible cell within the grid as the starting point. Without loss of generality, we use the top-left corner of the spatial grid as the starting point. The algorithm for extracting all cell-groups having $minAdjacentVariation$ is shown in Algorithm 1. This algorithm returns two mappings: cell index and cell-group index. While cell-index maps a cell to its corresponding cell-group, cell-group index stores for a cell-group the positions of first row, last row, first column, and last column of all cells forming the cell-group.

Algorithm 1: Extracting cell-groups

input : Original dataset ($gridData$) and $minAdjacentVariation$
output: Mapping from cell-groups to cells $gIndex$ and from cells to cell-groups $cIndex$

Let $dataNorm$ be the attribute normalized $gridData$;
$gIndex \leftarrow \emptyset$;
Let $cIndex$ and $visitedCell$ be 2 dimensional lists of all zeros with size equal to input grid;
for *each cell in* $dataNorm$ **do**
  Let $vCount$ be number of vertically adjacent cells with $variation \leq minAdjacentVariation$;
  Let $hCount$ be the number of horizontally adjacent cells with $variation \leq minAdjacentVariation$;
  Let $rCount$ be the number of horizontally and vertically adjacent cells forming a rectangle with $variation \leq minAdjacentVariation$;
  $maxCount \leftarrow max(vCount, hCount, rCount)$;
  for *each cell* (i, j) *counted for maxCount* **do**
    $visitedCell(i, j) \leftarrow 1$;
    $cIndex(i, j) \leftarrow length(gIndex)$;
  end
  Let $rBeg$, $rEnd$, $cBeg$, and $cEnd$ be the first row, last row, first column, and last column positions respectively of all cells counted for $maxCount$;
  $gIndex \leftarrow gIndex \cup (rBeg, rEnd, cBeg, cEnd)$;
end

Example 3. *In Figure 1, assume that* 0.02857143 *is the value of* $minAdjacentVariation$ *in the current iteration (iteration 2). From cell (lat2, lon1), we can move 3 steps horizontally and 2 steps vertically without exceeding* $minAdjacentVariation$*. Therefore, values of* $hCount$ *and* $vCount$ *in Algorithm 1 are 3 and 2, respectively. We can also form a rectangle of 3 columns and 2 rows from (lat2, lon1) in which all possible adjacent pairs of cells have a variation* $\leq$ 0.02857143*. So,* $rCount$ *is 6, which is higher than* $hCount$ *and* $vCount$*. Therefore, these 6 cells can form a cell-group.*

3) Feature Allocator: After re-partitioning the spatial cells into various cell-groups, our next task is to calculate the feature vector for each cell-group. A cell-group consisting of cells having null feature vectors is also assigned a null feature vector. Each dimension in the feature vector represents a distinct attribute in the multivariate representation, and it is

assigned a representative attribute value for all the merged cells within the entire cell-group. For each feature in the feature set, users need to define a parameter, known as aggregation type, in this step. Aggregation type takes two values: *summation* and *average*. As an example, if the feature is the count of criminal cases, summing up the counts for constituent cells forms the count for a cell-group. On the other hand, in the case of average housing prices, averaging the prices of constituent cells should be ideal. For the first case (*summation* is the aggregation type), we directly sum up the attribute values of constituent cells to form the cell-group feature. In the latter case, when *average* is the aggregation type, we noticed that choosing the average attribute value of all cells does not always minimize the local loss. Sometimes, choosing the most frequently occurring attribute value among all the cells as the representative cell-group value could minimize the local loss. Therefore, we select the best of these two options in the case of *average* aggregation type. The average value of an attribute is rounded to the nearest integer in the case of an integer-typed attribute. Although we use attribute-normalized form of the input dataset for earlier steps, this step works on the original input dataset. Algorithm 2 shows how to allocate feature vectors to all cell-groups of a re-partitioned dataset.

Algorithm 2: Allocating features to cell-groups

```
input : Original dataset gridData, mapping from cell-groups to cells
        gIndex, and feature aggregation types aggType
output: Feature vectors of cell-groups newData
for each cell-group cg in gIndex do
    for each feature k in gridData do
        if aggType(k) == summation then
            Let gFeature be the sum of feature k of all cells under cg;
            newData(cg, k) ← gFeature;
            continue;
        end
        Let A be the average of feature k of all cells under cg;
        Let B be the value of k at maximum number of cells under cg;
        lossA ← loss_cg(k) assuming newData(cg, k) ← A;
        lossB ← loss_cg(k) assuming newData(cg, k) ← B;
        if lossA ≤ lossB then
            newData(cg, k) ← A;
        else
            newData(cg, k) ← B;
        end
    end
end
```

Example 4. *Considering the same cell-group discussed in Example 3 with aggregation type average, the average attribute value of the group, A (in Algorithm 2), is* 23.67*, which is rounded to* 24 *since the attribute value is of integer type. The value of B for the cell-group is* 23*, as it is the most frequent value. Since $lossA$ and $lossB$ in Algorithm 2 are equal (*4*), the attribute value of the cell-group becomes* 24*. Allocating representative attribute values in a similar fashion to all the cell-groups in this example will result in the re-partitioned grid shown in iteration 2 of Figure 1.*

4) Information Loss Calculator: After finding the feature vectors of all cell-groups, we calculate the information loss (*IFL*) between the input dataset and re-partitioned dataset in the current iteration using the formula of *IFL* defined in Equation 3. The formula defined as *IFL* stands for mean absolute percentage error. Although other forms of information loss exist, such as normalized root mean square error, we opt for mean absolute percentage since we aim at making our re-partitioning framework sensitive to errors of individual instances relative to their corresponding original values. When calculating *IFL* using Equation 3, we need to retrieve the representative attribute values of input cells in the re-partitioned dataset. We take into account the aggregation type of each attribute used in the feature allocation step to calculate a representative attribute value. Each cell in the input dataset is mapped to a cell-group in the re-partitioned dataset. The attribute value of a cell-group should be divided by the number the cells within the cell-group if *summation* is the aggregation type. Otherwise, the attribute value of a cell-group is directly used as representative values of constituent cells.

Example 5. *In Figure 1, IFL is* 0 *after the first iteration and* 0.01868 *after the second iteration based on Equation 3.*

B. Training with Re-partitioned Dataset

A spatial grid dataset is formed such that all data objects that map to a cell are aggregated to produce the feature vector of the corresponding cell. Therefore, when a spatial grid dataset is used to train a spatial ML model, each cell is treated as an individual data object (instance) in the training dataset. There are two essential steps in training a spatial ML model with re-partitioned dataset - a) training data preparation and b) actual model training. The preparation step comprises feature vector creation and adjacency matrix computation out of the re-partitioned grid. The ML models consume the feature vectors of the re-partitioned data and the adjacency matrix as input. Our effort in this paper is focused on spatial data re-partitioning and training data preparation. The actual training of ML models is done out-of-the-box using PySAL [30], Pyinterpolate [31], and scikit-learn [32] libraries. While the feature vector creation is straightforward for regression, clustering, and classification models, in the case of spatial kriging, the feature vectors consist of the coordinates of the vertices of cell-groups along with the attribute values. Among the regression models, geographically weighted regression takes the centroids of cell-groups as part of the feature vectors.

Spatial ML systems such as PySAL encode the adjacency information with two dictionaries: one for neighbors list and another for weights associated with each neighbor. Our re-partitioning framework supports the creation of such adjacency lists along with weights. Since the input spatial cells are merged into cell-groups in our re-partitioned grid, creation of the adjacency list requires some non-trivial effort. We identify neighboring cell-groups based on the boundary cells within a cell-group. Since our new partitions are invariably shaped as rectangles, the neighboring cell-groups can be identified from the cells adjacent to the left-most, right-most, top-most and bottom-most cells acting as boundaries of a cell-group. Algorithm 3 contains the pseudocode to retrieve the adjacency list of cell-groups from the re-partitioned dataset. It is a binary adjacency list, where weight is 1 for each neighbor included in the neighbor list of a cell-group and 0 otherwise.

In order to train spatial ML models, we split each dataset into two parts - a) training data (80%) and b) test data (20%). We train each spatial ML model separately upon the original datasets and the reduced datasets obtained via our re-partitioning framework and all other data reduction baselines. For a fair comparison, we use the same hyperparameters to train each spatial model consistently regardless of whether the underlying spatial grid is prepared out of the original data or the reduced data. Hyperparameters used for various spatial ML models are reported in Table I under Section IV-A4.

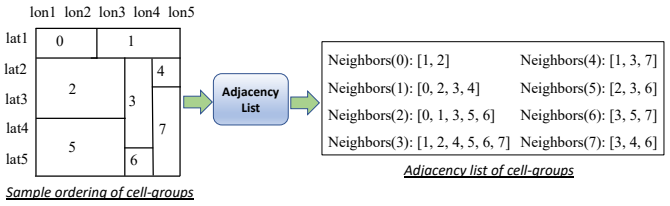


Fig. 3: An illustration of adjacency list for cell-groups

Algorithm 3: Retrieving adjacency list of cell-groups

input : Mapping between cell-groups and cells: $gIndex$, $cIndex$
output: Neighbors of cell-groups, $neighbors$
Let $neighbors$ be an empty dictionary;
for *each cell-group cg in gIndex* **do**
  Let $rBeg$, $rEnd$, $cBeg$, and $cEnd$ be the first, second, third, and fourth element of cg, respectively;
  $nList \leftarrow \emptyset$;
  for *each column c between cBeg and cEnd* **do**
    if *(rBeg - 1, c) not in nList* **then**
      $nList \leftarrow nList \cup cIndex(rBeg - 1, c)$;
    end
    if *(rEnd + 1, c) not in nList* **then**
      $nList \leftarrow nList \cup cIndex(rEnd + 1, c)$;
    end
  end
  for *each row r between rBeg and rEnd* **do**
    if *(r, cBeg - 1) not in nList* **then**
      $nList \leftarrow nList \cup cIndex(r, cBeg - 1)$;
    end
    if *(r, cEnd + 1) not in nList* **then**
      $nList \leftarrow nList \cup cIndex(r, cEnd + 1)$;
    end
  end
  $neighbors(cg) \leftarrow nList$;
end

Example 6. *Figure 3 shows the adjacency list for the re-partitioned dataset obtained in the second iteration of Figure 1. Consider cell-group* 1*, the neighbors of which are cell-groups* 0*,* 2*,* 3*, and* 4*.*

C. Mapping from Re-partitioned Cell-Groups to Input Cells

Our re-partitioned framework produces a compact grid with cell-groups which can be used to train spatial ML models. In the context of applications such as regression, the spatial ML models predict the values of attributes whose values are not always present in the original dataset. For example, housing prices in the home sales dataset [7] can be predicted from other existing attributes such as build year, renovation year, size of living area, etc. Note that the predicted housing price is for the cell-groups, and the user may want to know the housing price for the individual data instances or cells. In such a scenario, we need to re-construct the attribute values for cells based on the predicted attribute values of the cell-groups.

We store the mapping between the cell-group and its constituent cells in a hash map to enable a constant-time lookup. In order to enable the re-construction of attribute values for the constituent cells in a cell-group, we first need to retrieve the cells belonging to a cell-group, and then transform the cell-group features into features of constituent cells. This transformation depends on the aggregation type used during the feature allocation step of re-partitioning process. For a particular attribute in the feature set, if the aggregation type provided by the user is *average* during the feature allocation step, the value of the corresponding attribute for a cell-group can be directly assigned to all cells belonging to that cell-group. If *summation* is the aggregation type during feature allocation, corresponding value of a cell-group should be divided by the number the cells within the cell-group.

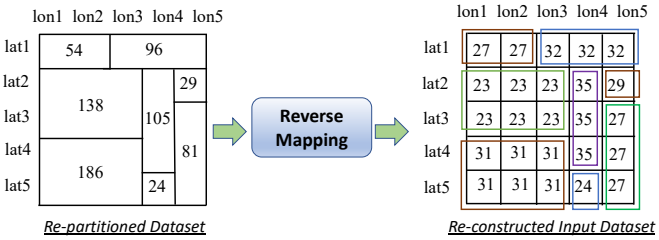


Fig. 4: Sample re-construction of cells from cell-groups

Example 7. *Figure 4 shows an example of how attribute values of all cells can be re-constructed from the cell-groups in a sample re-partitioned dataset. The sample dataset is univariate, and we assume that the aggregation type for the attribute is summation. The cell-group consisting of cells (lat1, lon1) and (lat1, lon2) has an attribute value of 54. Therefore, the attribute value of each of its constituent cells should be 27 as there are 2 cells in that cell-group.*

D. Homogeneous Re-partitioning Solution

Our proposed re-partitioning framework produces rectangular cell-groups of heterogeneous sizes, which require us to maintain the mapping between the input cells and the re-partitioned cell-groups. We implement a naïve variant of our re-partitioning approach as a baseline, which merges adjacent rows and adjacent columns to arrive at a coarse-grained grid with a pre-specified target resolution. We start with the least possible granularity of merging two adjacent rows and columns to obtain a coarse-grained grid, and we incrementally increase the value of the merged cells to 3, 4, .., in the subsequent iterations as long as the information loss does not exceed the pre-specified threshold.

Although this approach produces homogeneously-sized cell-groups, it suffers from the following disadvantages: i) the size of the dataset reduces with geometric progression after every iteration resulting in a high increase in information loss and ii) some of the cells in adjacent rows or columns might differ significantly from each other, as they are merged regardless of their similarity or dissimilarity.

IV. Experimental Evaluation

In this section, we report the experimental results to show the effectiveness of our re-partitioning framework which is publicly available [33]. Following are the insights we try to obtain empirically:

1) What is the data reduction w.r.t. the #spatial cells that we can achieve using our re-partitioning framework? We perform this evaluation in Section IV-B.
2) What are the savings in training time and memory consumption that can be obtained by training a spatial ML model upon the re-partitioned dataset instead of the original spatial grid dataset? See IV-C for details.
3) What is the penalty incurred w.r.t. increase in prediction error if a spatial ML model is trained on re-partitioned dataset? We experimentally evaluate this in Section IV-D.
4) We evaluate the homogeneous re-partitioning variant (proposed in Section III-D) in Section IV-E.

A. Experimental Setup

Not to rely on a high configuration setup, all experiments are conducted on an Ubuntu Linux machine with 48 CPU cores (Intel Xeon CPU E5-2687 3.00GHz) and 32GB memory size.

1) Evaluation Metrics:

- **Spatial cell reduction:** also termed as *data volume reduction*, denotes the number of spatial cells in the reduced re-partitioned dataset, relative to the original dataset.
- **Cell reduction time:** indicates the total time consumed by the re-partitioning algorithm until convergence.
- **Standard error (SE) of regression:** indicates the average distance of the ground truth from the regression line. It is also termed as residual standard error. It is one of two goodness-of-fit measures for spatial regression analysis [34].
- **Pseudo r-squared (R^2):** another widely used goodness-of-fit measure for spatial regression analysis. Denoting actual observation, predicted value, mean of observations, and the number of observations as y_i, $\hat{y}_i$, $\bar{y}$, and n respectively, pseudo r-squared [35] is defined as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{5}$$

Besides the aforementioned evaluation metrics, we measure the mean absolute error (MAE) and root mean square error (RMSE) for spatial regression and kriging. We evaluate the goodness of multi-class classification models using weighted F1-score, which is the weighted mean of class-wise F1-scores where weights are the class probabilities [36]. To evaluate the benefits of the re-partitioned dataset, we measure the training time and memory consumption of spatial ML models.

2) Dataset Preparation: Four real-world datasets are used to evaluate the efficiency of our re-partitioning framework which include New York City (NYC) taxi trip dataset [37], Washington King county home sales dataset [7], Chicago abandoned vehicles dataset [38], and NYC earning dataset [39]. We prepare a univariate dataset out of the Chicago abandoned vehicles by counting the #service requests in each spatial cell, whereas, for the home sales dataset, we prepare a multivariate dataset consisting of seven attributes: price of the home, #bedrooms, #bathrooms, size of living area, size of parking lot, build year, and renovation year. We calculate these attribute values by averaging all sales records in each cell. For the NYC taxi trip dataset, we prepare a univariate dataset with the #pickups in each spatial cell during a month and a multivariate dataset with total #pickups, total #passengers, summation of distances, and summation of taxi fares for all the rides in each cell. In the case of NYC earning dataset, the univariate dataset consists of total #jobs in each cell while the multivariate dataset has five attributes: land area, water area, #jobs with earning $\leq \$1250/month$, #jobs with earning between $\$1251/month$ and $\$3333/month$, and #jobs with earning $\geq \$3333/month$.

Spatial ML models trained on fine-grained grids perform better than their coarse-grained counterparts in terms of prediction and classification accuracy. On the other hand, training spatial ML models with fine-grained grids suffers from long training time and high memory usage. Our framework enables spatial ML practitioners to train models on fine-grained grids while incurring less training time and memory consumption. In order to evaluate our framework upon grids of varying granularity w.r.t. reducing training time and memory usage, we use three grid granularities - a) $\approx 100k$ (315×318) cells, b) $\approx 78k$ (279×280) cells, and c) $\approx 36k$ (191×193) cells.

Empty cells in a grid, if any, can be tackled with data imputation methods [40]–[43]. Applying imputation as a pre-processing step presents a trade-off between accuracy and data reduction. In our framework, we currently do not incorporate imputation methods in the interest of preserving accuracy, and represent empty cells with null feature vectors. Our framework allows empty cells to be merged with other adjacent empty cells but not with the adjacent non-empty cells.

3) Baselines: We compare the performance of our re-partitioning framework with two state-of-the-art methods that are used widely in the literature to reduce data volume.

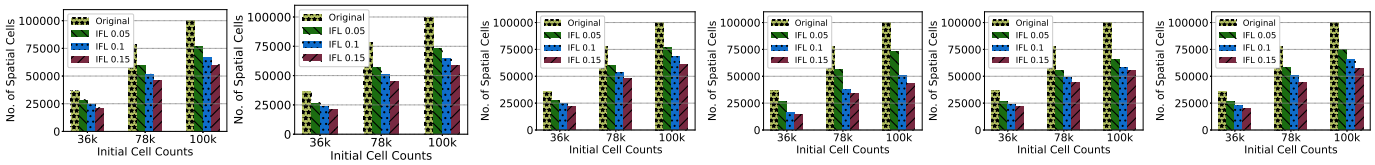
1) **Sampling:** We evaluate the effect of sampling on spatial training data by implementing the spatial sampling method proposed by Guo et al. [9].
2) **Regionalization:** We implement the optimized regionalization technique proposed by Biswas et al. [13] and use the regionalized data to train spatial ML models.
3) **Spatially Contiguous Clustering:** We implement the hierarchical clustering technique proposed by Kim et al. [15] and use the clustered data to train spatial ML models.

For a fair comparison, we set the number of target samples/regions/clusters for the baselines to the counts of cell-groups returned by our re-partitioning framework for various *IFL* thresholds.

4) Model Hyperparameters: Table I lists the values of the hyperparameters used to train the spatial ML models.

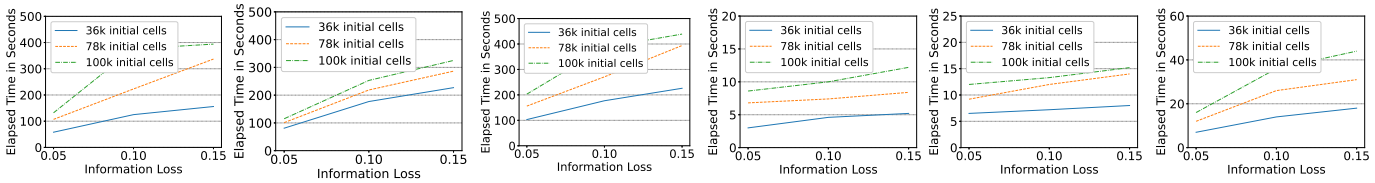
B. Evaluating Cell Reduction Performance

The number of cells in the grid before applying the re-partitioning algorithm is referred to as *initial cell count*. We



(a) Taxi trip multivariate (b) Home sales multivariate (c) Earning multivariate (d) Taxi trip univariate (e) Vehicles univariate (f) Earning univariate

Fig. 5: Evaluating cell reduction performance of re-partitioning algorithm with various values of information loss on all datasets



(a) Taxi trip multivariate (b) Home sales multivariate (c) Earning multivariate (d) Taxi Trip univariate (e) Vehicles univariate (f) Earning univariate

Fig. 6: Analyzing required time for running re-partitioning algorithm with various values of information loss on all datasets

TABLE I: Hyperparameters of spatial ML models

Model	Hyperparameters
Random Forest Regression	n_estimators: 225, max_depth: 7, min_samples_leaf: 20, criterion: *mse*
Support Vector Machine Regression	kernel: *rbf*, C: 15, gamma: 0.5, epsilon: 0.01
Geographically Weighted Regression	kernel: *gaussian*, criterion: *AICc*, fixed: *False*
Spatial Kriging	search_radius: 0.01, max_range: 0.32, number_of_neighbors: 8
Lag Regression	weight: *adjacency_list*, adjacency_type: *Binary*
Error Regression	weight: *adjacency_list*, adjacency_type: *Binary*
Gradient Boosting Classification	n_estimators: 200, max_depth: 5, min_samples_leaf: 12, loss: *deviance*
K-Nearest Neighbor Classification	leaf_size: 18, n_neighbors: 7

evaluate our re-partitioning algorithm for initial cell counts of approximately 36k, 78k, and 100k. We analyze the spatial cell reduction and time for three thresholds of user-specified information loss: 0.05, 0.1, and 0.15.

Figure 5 depicts the spatial cell reduction on all datasets with various thresholds of information loss. Our re-partitioning framework can reduce #spatial cells by around 30% with an information loss of only 0.05. Increasing the values of information loss (IFL) thresholds to 0.1 and 0.15 reduces #spatial cells by ∼37% and ∼42%, respectively. To quantify the effect of the % of spatial cell reduction, we also present the training time and memory usage in Section IV-C.

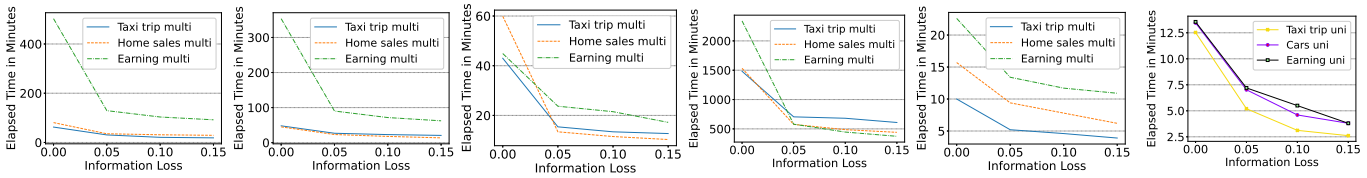
Upon observing the plots shown in Figure 5, we can state that the cell reduction achieved by our framework is unaffected by #attributes in the dataset because #cells reduced is approximately equal on both multivariate and univariate datasets. This can be attributed to two factors. Firstly, using the attribute-normalized form of input dataset during the computation of min-adjacent variation and cell-group extraction of the re-partitioning algorithm helps in treating all the attributes equally. Secondly, averaging the total IFL across various attributes in Equation 2 makes it agnostic to #attributes. Although #cells decreases as the IFL threshold increases, the rate of cell reduction keeps getting lower. It indicates that #cells cannot be reduced much after IFL threshold reaches a high value. Thus, the IFL threshold marks a trade-off between the training time and prediction error.

Figure 6 shows the cell reduction time with various IFL thresholds on both multivariate and univariate datasets. We can observe that the cell reduction time on the multivariate datasets varies from 50 to 390 seconds for various thresholds of IFL and initial cell counts. The cell reduction time increases with the increase in both IFL threshold and initial cell count. With a higher value of initial cell count, the re-partitioning algorithm needs to process a higher number of partitions. As the IFL threshold increases, the algorithm runs for more iterations resulting in higher elapsed time. In the case of univariate dataset, the elapsed time during re-partitioning is in the range of 2 to 15 seconds. Elapsed time is higher for the multivariate datasets as compared to the univariate datasets because, IFL, $minAdjVariation$, and other statistics need to be calculated for several attributes in the case of a multivariate dataset. Note that the price we pay in terms of cell reduction time is negligible as compared to the savings we obtain for model training time which will be presented in Section IV-C.

C. Evaluating Training Time and Memory Usage

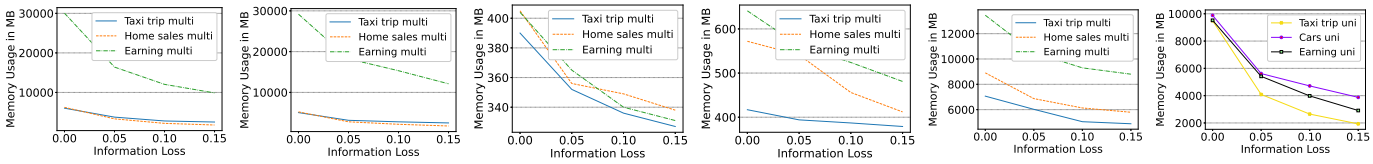
In this section, we analyze the performance of our framework in terms of reducing training time and memory usage when the re-partitioned dataset is applied to train four types of spatial ML models: regression, kriging, clustering, and classification. We need not compare the training time and memory usage against the baselines because our implementation of baselines, as discussed in Section IV-A3, retrieves exactly the same number of cell-groups as required by our re-partitioning framework. This results in similar training times and memory usage for all the approaches, including the baselines.

1) Evaluation with Regression Models: We evaluate the reduction of training time and memory usage for five spatial regression models: spatial lag, spatial error, geographically weighted, spatial support vector, and spatial random forest regression models. Since regression can only be applied to multivariate datasets, we use three multivariate datasets (taxi trip, home sales, and earning datasets) and their re-partitioned forms to train and test regression models. Also, we have three versions of each original dataset with different counts of initial cells as discussed in Section IV-B. We use the largest original datasets with $\approx 100k$ cells and their re-partitioned versions to train each model. We use the attributes taxi fare amount, home



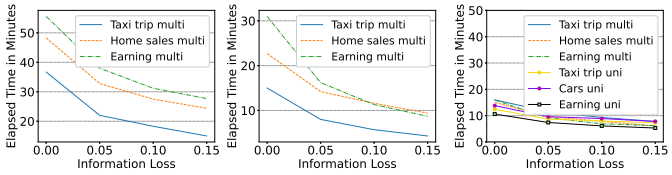
(a) Lag regression (b) Error regression (c) GWR (d) SVM regression (e) RF regression (f) Spatial kriging

Fig. 7: Performance evaluation of re-partitioning system in terms of training time reduction of regression and kriging models



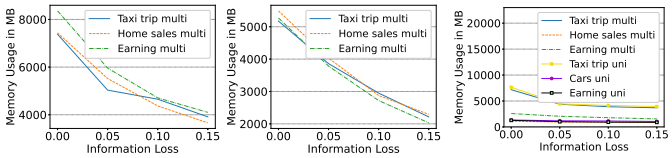
(a) Lag regression (b) Error regression (c) GWR (d) SVM regression (e) RF regression (f) Spatial kriging

Fig. 8: Performance evaluation of re-partitioning system in terms of memory usage reduction of regression and kriging models



(a) GB classifier (b) KNN classifier (c) Clustering

Fig. 9: Evaluating clustering and classification training time



(a) GB classifier (b) KNN classifier (c) Clustering

Fig. 10: Evaluating clustering and classification memory usage

prices, and #jobs with earning $\geq$ \$3333/*month* as target attributes for taxi trip, home sales, and earning datasets respectively, while treating other attributes as features. Training time and memory usage of spatial lag model, spatial error model, geographically weighted regression, support vector regression, and random forest regression have been reported in Figures 7 and 8. We compare the values of the evaluation metrics for the original dataset against the re-partitioned datasets with *IFL* thresholds of 0.05, 0.1, and 0.15.

Our experimental results show that the re-partitioned dataset with *IFL* threshold 0.05 can reduce the training time in the range [40%, 77%]. Training time reduction is the highest for the support vector regression model and the least for the random forest regression model. The performance gain in terms of training time is higher for models that usually take a long training time, such as support vector regression, geographically weighted regression, and spatial lag model, which is beneficial because training time is a concern for those models. In the case of memory usage reduction, the range varies from 9.5% to 47% for the same *IFL* threshold. We observe the highest savings for the spatial lag regression model, while geographically weighted regression model has the least savings. Similarly to the training time, memory savings are higher for models that consume high memory during training, such as spatial lag regression, spatial error regression, and random forest regression. In the case of geographically weighted regression and support vector regression models, although we see a reduction in training time, memory savings are not significant enough because the memory consumption on the original dataset itself is very low, which is less than 2GB. On the other hand, memory usage for the original datasets in the case of random forest regression and lag regression is around 18GB, where our re-partitioned dataset achieves the highest memory reduction.

Training time reduction for *IFL* thresholds of 0.1 and 0.15 are in the ranges [50%, 81%] and [58%, 84%], respectively. For these two thresholds, although we allow the loss of information to be 2$\times$ and 3$\times$ higher as compared to a threshold of 0.05, we do not achieve significant training time reduction as compared to the reduction we get for the 0.05 *IFL* threshold. We achieve memory reduction of up to 65% and 72% for IFL thresholds of 0.1 and 0.15, respectively. Similarly to training time, memory reduction for high thresholds is not significantly higher as compared to the low threshold. Therefore, the choice of *IFL* threshold should be a trade-off between training time or memory usage reduction and model quality which we discuss in Section IV-D. Re-partitioned datasets with high *IFL* thresholds follow a similar pattern as the re-partitioned datasets with low threshold w.r.t. one of our earlier observation that savings in training time and memory usage are higher for models that take long training time and consume high memory.

2) Evaluation with Classification Models: Besides regression, we also evaluate our framework upon two classification models: gradient boosting and k-nearest neighbor classification. To train these models, we convert the target variable of every multivariate dataset into a categorical variable by mapping its values into five distinct range bins representing multiple classes: low, low-medium, medium, medium-high, and high. Figures 9a, 9b, 10a, and 10b show that our framework achieves a consistent reduction rate in memory usage

TABLE II: Prediction errors of spatial regression and kriging models

Dataset			SE	R Squared
Taxi Trip	Original		112.2	0.92
	IFL = 0.05	Re-partitioning	**116.06**	**0.92**
		Sampling	122.0	0.86
		Regionalization	119.6	0.89
		Clustering	119.87	0.89
	IFL = 0.1	Re-partitioning	**117.59**	**0.917**
		Sampling	124.71	0.87
		Regionalization	122.3	0.88
		Clustering	123.09	0.87
	IFL = 0.15	Re-partitioning	**119.05**	**0.905**
		Sampling	128.27	0.85
		Regionalization	125.3	0.865
		Clustering	124.68	0.86
Home Sales	Original		276.25	0.81
	IFL = 0.05	Re-partitioning	**282.6**	**0.806**
		Sampling	298.14	0.759
		Regionalization	294.21	0.777
		Clustering	295.16	0.772
	IFL = 0.1	Re-partitioning	**287.5**	**0.80**
		Sampling	304.73	0.754
		Regionalization	299.86	0.763
		Clustering	299.35	0.76
	IFL = 0.15	Re-partitioning	**290.3**	**0.79**
		Sampling	309.49	0.737
		Regionalization	305.11	0.748
		Clustering	307.20	0.741
Earning	Original		140.11	0.85
	IFL = 0.05	Re-partitioning	**143.47**	**0.85**
		Sampling	160.68	0.79
		Regionalization	149.93	0.82
		Clustering	151.45	0.81
	IFL = 0.1	Re-partitioning	**145.57**	**0.84**
		Sampling	157.22	0.77
		Regionalization	151.83	0.80
		Clustering	151.19	0.80
	IFL = 0.15	Re-partitioning	**148.23**	**0.83**
		Sampling	161.57	0.75
		Regionalization	155.79	0.79
		Clustering	156.04	0.78

(a) Spatial Lag Regression

Dataset			SE	R Squared
Taxi Trip	Original		120	0.93
	IFL = 0.05	Re-partitioning	**121.11**	**0.924**
		Sampling	127.89	0.877
		Regionalization	125.71	0.887
		Clustering	125.48	0.88
	IFL = 0.1	Re-partitioning	**123.46**	**0.918**
		Sampling	132.35	0.86
		Regionalization	129.39	0.881
		Clustering	130.12	0.873
	IFL = 0.15	Re-partitioning	**126.62**	**0.90**
		Sampling	137.51	0.84
		Regionalization	132.82	0.868
		Clustering	134.48	0.859
Home Sales	Original		261.25	0.82
	IFL = 0.05	Re-partitioning	**267.63**	**0.81**
		Sampling	285.19	0.762
		Regionalization	277.53	0.776
		Clustering	277.92	0.773
	IFL = 0.1	Re-partitioning	**271.29**	**0.803**
		Sampling	292.45	0.76
		Regionalization	281.6	0.748
		Clustering	283.21	0.733
	IFL = 0.15	Re-partitioning	**277.73**	**0.774**
		Sampling	311.61	0.72
		Regionalization	289.95	0.737
		Clustering	289.32	0.74
Earning	Original		124.42	0.83
	IFL = 0.05	Re-partitioning	**128.25**	**0.82**
		Sampling	144.25	0.76
		Regionalization	134.28	0.79
		Clustering	134.84	0.78
	IFL = 0.1	Re-partitioning	**130.14**	**0.82**
		Sampling	144.46	0.76
		Regionalization	136.10	0.78
		Clustering	136.78	0.773
	IFL = 0.15	Re-partitioning	**133.51**	**0.80**
		Sampling	145.53	0.75
		Regionalization	140.45	0.76
		Clustering	142.11	0.757

(b) Spatial Error Regression

Dataset			MAE	RMSE
Taxi Trip	Original		131.5	112
	IFL = 0.05	Re-partitioning	**131.9**	**112.6**
		Sampling	141.13	117.23
		Regionalization	137.44	116.43
		Clustering	138.26	116.88
	IFL = 0.1	Re-partitioning	**132.6**	**113.2**
		Sampling	142.55	118.41
		Regionalization	138.96	117.08
		Clustering	139.18	117.74
	IFL = 0.15	Re-partitioning	**134.7**	**115.62**
		Sampling	147.73	124.08
		Regionalization	141.7	120.3
		Clustering	142.44	120.95
Home Sales	Original		286.31	292.246
	IFL = 0.05	Re-partitioning	**291.52**	**299.4**
		Sampling	305.76	312.78
		Regionalization	301.14	310.18
		Clustering	301.69	310.82
	IFL = 0.1	Re-partitioning	**295.5**	**301.5**
		Sampling	314.43	316.31
		Regionalization	306.43	313.47
		Clustering	308.15	315.28
	IFL = 0.15	Re-partitioning	**298.08**	**305.28**
		Sampling	317.93	322.38
		Regionalization	310.48	317.8
		Clustering	311.41	318.16
Earning	Original		50.55	113.39
	IFL = 0.05	Re-partitioning	**52.07**	**117.36**
		Sampling	56.34	128.51
		Regionalization	54.60	122.64
		Clustering	55.46	123.58
	IFL = 0.1	Re-partitioning	**52.67**	**118.60**
		Sampling	57.57	131.41
		Regionalization	54.81	123.82
		Clustering	55.13	124.21
	IFL = 0.15	Re-partitioning	**53.73**	**119.51**
		Sampling	57.87	132.06
		Regionalization	56.04	125.01
		Clustering	56.63	125.71

(c) Geographically Weighted Regression

Dataset			MAE	RMSE
Taxi Trip	Original		35.05	142.3
	IFL = 0.05	Re-partitioning	**36.41**	**145.58**
		Sampling	37.96	153.16
		Regionalization	37.56	150.21
		Clustering	37.70	150.93
	IFL = 0.1	Re-partitioning	**36.83**	**146.16**
		Sampling	38.56	154.3
		Regionalization	38.05	151.27
		Clustering	38.40	152.22
	IFL = 0.15	Re-partitioning	**37.70**	**148.50**
		Sampling	39.72	154.9
		Regionalization	39.02	154.14
		Clustering	39.29	154.45
Home Sales	Original		185.41	278.40
	IFL = 0.05	Re-partitioning	**189.37**	**285.35**
		Sampling	196.64	296.07
		Regionalization	195.62	294.60
		Clustering	195.86	295.10
	IFL = 0.1	Re-partitioning	**193.25**	**286.57**
		Sampling	202.91	297.48
		Regionalization	200.21	296.49
		Clustering	200.64	–296.80
	IFL = 0.15	Re-partitioning	**198.63**	**291.07**
		Sampling	216.85	302.5
		Regionalization	213.17	300.38
		Clustering	214.12	300.91
Earning	Original		76.45	211.37
	IFL = 0.05	Re-partitioning	**79.36**	**217.92**
		Sampling	85.31	233.17
		Regionalization	83.01	227.07
		Clustering	83.76	227.88
	IFL = 0.1	Re-partitioning	**79.81**	**221.43**
		Sampling	86.59	240.03
		Regionalization	83.06	230.73
		Clustering	83.62	232.47
	IFL = 0.15	Re-partitioning	**80.89**	**224.05**
		Sampling	88.33	240.85
		Regionalization	84.37	233.24
		Clustering	85.21	234.26

(d) Support Vector Regression

Dataset			MAE	RMSE
Taxi Trip	Original		4.02	19.40
	IFL = 0.05	Re-partitioning	**4.17**	**19.61**
		Sampling	4.49	20.6
		Regionalization	4.35	20.54
		Clustering	4.38	20.58
	IFL = 0.1	Re-partitioning	**4.21**	**19.94**
		Sampling	4.56	20.92
		Regionalization	4.42	20.79
		Clustering	4.42	20.82
	IFL = 0.15	Re-partitioning	**5.1**	**20.86**
		Sampling	5.51	21.97
		Regionalization	5.36	21.84
		Clustering	5.43	21.89
Home Sales	Original		57.46	185.617
	IFL = 0.05	Re-partitioning	**59.0**	**189.44**
		Sampling	63.16	196.74
		Regionalization	61.81	196.08
		Clustering	61.98	196.32
	IFL = 0.1	Re-partitioning	**60.13**	**193.36**
		Sampling	64.06	203.7
		Regionalization	63.17	200.51
		Clustering	63.56	201.08
	IFL = 0.15	Re-partitioning	**61.38**	**198.69**
		Sampling	64.11	207.85
		Regionalization	63.77	207.23
		Clustering	63.90	207.42
Earning	Original		8.12	44.81
	IFL = 0.05	Re-partitioning	**8.23**	**45.66**
		Sampling	9.22	49.77
		Regionalization	8.65	47.82
		Clustering	8.75	48.13
	IFL = 0.1	Re-partitioning	**8.40**	**46.33**
		Sampling	9.41	50.27
		Regionalization	8.86	48.97
		Clustering	9.05	49.23
	IFL = 0.15	Re-partitioning	**8.69**	**48.22**
		Sampling	9.91	54.0
		Regionalization	9.25	50.78
		Clustering	9.46	51.66

(e) Random Forest Regression

Dataset			MAE	RMSE
Taxi Trip	Original		177.4	203.54
	IFL = 0.05	Re-partitioning	**185.2**	**248.75**
		Sampling	194.65	269.89
		Regionalization	190.39	255.59
		Clustering	191.18	258.30
	IFL = 0.1	Re-partitioning	**188.14**	**253.76**
		Sampling	198.31	272.32
		Regionalization	193.03	260.24
		Clustering	194.25	264.10
	IFL = 0.15	Re-partitioning	**190.7**	**258.3**
		Sampling	197.73	267.26
		Regionalization	196.23	265.53
		Clustering	196.67	266.82
Vehicles	Original		7.31	11.05
	IFL = 0.05	Re-partitioning	**7.6**	**11.47**
		Sampling	8.22	12.89
		Regionalization	7.81	11.8
		Clustering	7.89	12.24
	IFL = 0.1	Re-partitioning	**7.79**	**11.79**
		Sampling	8.94	13.42
		Regionalization	8.02	12.1
		Clustering	8.32	12.53
	IFL = 0.15	Re-partitioning	**8.13**	**12.03**
		Sampling	10.38	13.57
		Regionalization	8.38	12.37
		Clustering	8.65	12.65
Earning	Original		124.03	206.87
	IFL = 0.05	Re-partitioning	**126.14**	**211.21**
		Sampling	134.97	226.21
		Regionalization	130.05	217.34
		Clustering	131.35	219.16
	IFL = 0.1	Re-partitioning	**128.25**	**212.66**
		Sampling	138.64	227.12
		Regionalization	136.71	219.89
		Clustering	137.09	220.55
	IFL = 0.15	Re-partitioning	**133.58**	**216.80**
		Sampling	144.93	229.59
		Regionalization	137.19	227.21
		Clustering	140.24	227.60

(f) Spatial Kriging

and training time for both the classification models.

3) Evaluation with Spatial Kriging: Kriging is a geostatistical method to estimate the value of a variable at an unobserved location from nearby observations based on the spatial arrangement of the observations. Figures 7f and 8f show the evaluation results with spatial kriging model. Since spatial kriging is usually done on a univariate dataset, we train the spatial kriging model with all univariate datasets. Training time reduction varies from 48% to 58%, while the reduction in memory usage varies from 43% to 57% with the 0.05 *IFL* threshold. With an *IFL* threshold of 0.1, the training time and memory usage are reduced by [57%, 75%] and [52%, 72%], respectively. Similarly to regression models, we do not observe significant savings in memory usage and training time for the *IFL* threshold 0.15 as compared to smaller thresholds.

4) Evaluation with Spatial Clustering: We evaluate our framework with spatial clustering on all multivariate and univariate datasets. Figures 9c and 10c show the clustering time and memory savings on spatially constrained hierarchical clustering. For the 0.05 *IFL* threshold, clustering time and

TABLE III: Weighted F1-score of classification models

Dataset			F1 Score
Taxi trip	Original		0.94
	IFL 0.05	Re-partitioning	**0.93**
		Sampling	0.85
		Regionalization	0.88
		Clustering	0.88
	IFL 0.1	Re-partitioning	**0.93**
		Sampling	0.86
		Regionalization	0.87
		Clustering	0.88
	IFL 0.15	Re-partitioning	**0.93**
		Sampling	0.85
		Regionalization	0.88
		Clustering	0.87
Home sales	Original		0.96
	IFL 0.05	Re-partitioning	**0.93**
		Sampling	0.81
		Regionalization	0.83
		Clustering	0.84
	IFL 0.1	Re-partitioning	**0.92**
		Sampling	0.78
		Regionalization	0.81
		Clustering	0.82
	IFL 0.15	Re-partitioning	**0.88**
		Sampling	0.75
		Regionalization	0.76
		Clustering	0.78
Earning	Original		0.97
	IFL 0.05	Re-partitioning	**0.95**
		Sampling	0.88
		Regionalization	0.9
		Clustering	0.91
	IFL 0.1	Re-partitioning	**0.94**
		Sampling	0.86
		Regionalization	0.88
		Clustering	0.89
	IFL 0.15	Re-partitioning	**0.91**
		Sampling	0.82
		Regionalization	0.84
		Clustering	0.84

(a) Gradient Boosting

Dataset			F1 Score
Taxi trip	Original		0.92
	IFL 0.05	Re-partitioning	**0.92**
		Sampling	0.82
		Regionalization	0.88
		Clustering	0.88
	IFL 0.1	Re-partitioning	**0.92**
		Sampling	0.80
		Regionalization	0.87
		Clustering	0.88
	IFL 0.15	Re-partitioning	**0.91**
		Sampling	0.79
		Regionalization	0.85
		Clustering	0.84
Home sales	Original		0.93
	IFL 0.05	Re-partitioning	**0.9**
		Sampling	0.8
		Regionalization	0.81
		Clustering	0.81
	IFL 0.1	Re-partitioning	**0.89**
		Sampling	0.78
		Regionalization	0.77
		Clustering	0.80
	IFL 0.15	Re-partitioning	**0.88**
		Sampling	0.74
		Regionalization	0.76
		Clustering	0.77
Earning	Original		0.75
	IFL 0.05	Re-partitioning	**0.73**
		Sampling	0.58
		Regionalization	0.63
		Clustering	0.62
	IFL 0.1	Re-partitioning	**0.72**
		Sampling	0.57
		Regionalization	0.59
		Clustering	0.60
	IFL 0.15	Re-partitioning	**0.69**
		Sampling	0.54
		Regionalization	0.57
		Clustering	0.59

(b) K Nearest Neighbor

TABLE IV: Comparing correctness of clustering

Dataset	Method	IFL = 0.05	IFL = 0.1	IFL = 0.15
Taxi trip multivariate	Re-partitioning	**99.48**	**99.46**	**99.14**
	Sampling	95.71	92.12	88.83
	Regionalization	97.39	96.97	96.91
	Clustering	97.06	96.80	96.58
Home sales multivariate	Re-partitioning	**96.86**	**96.17**	**95.37**
	Sampling	91.39	89.74	88.18
	Regionalization	93.66	93.16	91.32
	Clustering	93.49	92.72	91.54
Earnings multivariate	Re-partitioning	**98.33**	**97.76**	**97.14**
	Sampling	92.26	91.68	91.19
	Regionalization	94.48	93.75	93.17
	Clustering	94.74	94.10	93.37
Taxi trip univariate	Re-partitioning	**99.38**	**98.73**	**98.24**
	Sampling	94.33	90.73	90.09
	Regionalization	97.09	96.66	95.88
	Clustering	96.92	96.71	96.15
Vehicles univariate	Re-partitioning	**97.47**	**95.93**	**95.08**
	Sampling	92.29	89.95	87.13
	Regionalization	94.91	93.75	92.96
	Clustering	94.78	93.56	93.16
Earnings univariate	Re-partitioning	**98.76**	**98.27**	**97.81**
	Sampling	92.51	91.92	91.38
	Regionalization	94.79	94.16	93.40
	Clustering	94.96	94.45	93.83

memory usage reduction is in the range [28%, 35%] and [11%, 42%], respectively. Clustering time, memory consumption, and their reduction rate obtained using re-partititoning are low for the univariate datasets as compared to the multivariate datasets.

D. Evaluating Prediction and Classification Errors

We evaluate the prediction and classification quality of trained spatial models in terms of mean absolute error, root mean square error and weighted F1-score. In the case of spatial lag and error models, we report the pseudo r-squared and SE of regression which are common measures for these models. Prediction errors for all the regression models and the kriging model have been reported in Table II. We make two comparisons in this section: i) we measure the percent increase of prediction error when the re-partitioned dataset is used to train a model instead of the original dataset and ii) we compare the prediction errors for the re-partitioned framework against the baseline techniques.

1) Analyzing the increase in Errors: In this section, we compare the prediction errors upon the original dataset against those on the re-partitioned datasets with various *IFL* thresholds. Our results show that the difference between prediction error on the original datasets and re-partitioned datasets is always $\leq 4\%$ for an *IFL* threshold of 0.05 and $\leq 5\%$ for the 0.1 *IFL* threshold. Pseudo r-squared values for 0.05 and 0.1 *IFL* thresholds are very close to the pseudo r-squared for the original dataset, indicating the goodness of spatial lag and error models. Similarly to regression models, spatial kriging also maintains the difference in the prediction errors below 4% and 5% for the *IFL* thresholds 0.05 and 0.1, respectively.

On the contrary, differences in standard regression errors are more than 5% for the 0.15 *IFL* threshold. Therefore, we recommend that an IFL threshold of up to 0.1 should be selected so that training time and memory usage can be reduced considerably, while keeping the difference in prediction error within 5%. Similarly to spatial lag and error models, other regression models and spatial kriging also have more than 5% difference in MAE and RMSE when IFL $\geq$ 0.15.

Table IV reports the $\%$ of cells that are under the same clusters when clustering is applied to both the original and the re-partitioned (also sampled and regionalized) grids. Based on Table IV, we can say that the clustering accuracy is around 99% upon the taxi trip dataset for the 0.05 *IFL* threshold, and it is more than 98% for other thresholds. Clustering correctness varies from 95% to 97.5% on other datasets.

2) Comparing Prediction and Classification Errors With Baselines: In this section, we compare the prediction and classification errors for re-partitioned grids with the errors incurred by applying the baseline techniques. As mentioned before, note that we use the count of cell-groups obtained from the re-partitioning framework with a specific *IFL* threshold as the target number of samples, regions and clusters for sampling, regionalization and clustering respectively for a fair comparison. Upon observing the prediction errors, we can state that the re-partitioned grids obtained from our framework perform the best, and we outperform the regionalization, clustering, and sampling techniques by 3% to 14% for all regression models. It is due to the fact that spatial sampling techniques cannot capture spatial adjacency, while the regionalization and clustering techniques fail to maintain low information loss.

In the case of classification models, weighted F1-scores for the re-partitioned datasets are 5% to 20% higher than the F1-scores obtained by the baselines. Prediction errors of spatial kriging model for re-partitioned datasets are 2.5% to 8.5% lower than the datasets reduced by baselines. Similarly to regression models, sampling performs the worst in the case of spatial kriging because kriging techniques use spatial autocorrelation to interpolate the values in the spatial cells. Clustering correctness of re-partitioning framework is 2% to 10% better than the baselines. Sampling is agnostic to spatial contiguity and performs poorer than other baselines

that preserve autocorrelation.

E. Evaluating Homogeneous Re-partitioning

In this empirical study, we estimate the information loss (*IFL*) that occurs when we re-partition the input spatial grid to generate homogeneous cell-groups applying the method discussed in Section III-D. We report the *IFL* after running this approach for the first iteration in Table V, where we merge the least possible #adjacent rows or columns or both (which is 2) to arrive at a coarse-grained grid. The results show that the *IFL* values are very high (> 0.4), and exceed the largest possible *IFL* threshold of 0.15 that we use to evaluate our re-partitioning framework in the prior sub-sections. Due to the poor performance shown by homogeneous re-partitioning, it is clear that we need not run this approach for further iterations. This is because *IFL* values will only become worse in subsequent iterations.

TABLE V: Information loss for homogeneous grid

Dataset	Merging 2 rows	Merging 2 columns	Merging 2 rows & 2 columns
Taxi trip multivariate	0.538	0.497	0.641
Taxi trip univariate	0.428	0.467	0.534
Home sales multivariate	0.484	0.509	0.613
Vehicles univariate	0.416	0.405	0.476
Earnings multivariate	0.504	0.527	0.669
Earnings univariate	0.391	0.435	0.487

V. Related Work

Existing spatial data reduction techniques can be classified into four categories: data sampling, regionalization, dimensionality reduction, and data reduction by modeling.

1. Data Sampling - As discussed in earlier sections, sampling is an effective approach to reduce the volume of large datasets. Guo et al. [9] propose a technique to sample spatial map data such that sampled data instances maintain a proximal distance threshold while maximizing a similarity score. Tabula [10], [44] introduces a sampling technique allowing user-defined loss function where samples are pre-materialized. Other spatial sampling techniques include multi-level sampling [11] and sampling and aggregation [8]. The limitation of sampling approaches is that they are orthogonal to the goal of this paper, and they cannot capture the adjacency among the sample partitions drawn which turns out to be inapplicable to our target applications.

2. Regionalization - Regionalization clusters a large set of small polygons into a small set of large regions. Examples in this category include [13]–[18]. The number of clusters is given as a parameter, which is significantly smaller than the total number of polygons. These approaches start with initializing the regions with random cells and grow each region later by satisfying user constraints. No existing works applied these techniques to reduce the training data of spatial ML models. Our work is closely related to this category, with the exception that the number of target regions is not pre-specified. Instead, we expect the user to specify an information loss (IFL) threshold, which is an intuitive numerical value between 0 and 1. We utilize this to constrain our re-partitioning technique to produce a minimal set of target cell-groups, while not incurring a high loss. We also made the first attempt to adapt regionalization towards reducing the volume of input training data for spatial ML models.

3. Feature Selection and Dimensionality Reduction - Many current dataset reduction approaches concentrate on eliminating or preserving a subset of the original dataset's features. Feature selection approaches have been surveyed by prior works [45], [46]. On the contrary, dimensionality reduction techniques project the features of the original dataset into a new feature space with different dimensionality [24]–[26]. Although feature selection and dimensionality reduction cannot be applied towards reducing spatial cells, they can serve as complementary techniques to reduce the dimensionality of the feature vectors within the re-partitioned spatial cells in order to further enhance the reduction in ML model training time and memory usage.

4. Data Reduction by Modeling - Several data reduction techniques exploit statistical modeling to reduce the size of a dataset [47], [48]. They exploit statistical properties of data blocks to find similar blocks within streaming spatio-temporal data. 2D-STR [27] introduced a 2-dimensional spatio-temporal reduction method where spatio-temporal matrix of a dataset is partitioned into regions of similar instances, and each region is reduced to a model of its instances. This approach is targeted towards spatio-temporal traffic data processing tasks and aims to reduce the processing time by reducing the data volume. They do not address how to concisely represent the reduced spatio-temporal dataset for the purpose of feeding it to an ML model. Our work solves an orthogonal problem where the training data is not streaming and we exclusively focus on spatial data and not spatio-temporal data, the extension to which can be treated as possible future work.

VI. Conclusion

In this paper, we propose a machine learning-aware data re-partitioning framework for spatial datasets. Our proposed re-partitioning framework reduces the volume of a spatial grid dataset by reducing the number of cells in the spatial grid, which further helps in reducing the model training time and memory usage when the re-partitioned dataset is applied to train a spatial ML model. To evaluate the effectiveness of our proposed framework on diverse spatial ML applications, we perform training and testing of multiple spatial regression, spatial kriging, classification, and clustering models using both input datasets and re-partitioned datasets. Our empirical evaluations on four real-world datasets validate that our proposed re-partitioning framework can reduce the training time in the range of 43% to 81% for kriging, classification, and regression models with less than 5% difference in prediction error when the information loss is within a very low threshold (≤ 0.1). Our framework also outperforms the state-of-the-art baselines by 3% to 14% on regression models, 5% to 20% on classification models, and 2% to 10% for other spatial models in terms of prediction and classification errors. In the future, we plan to improve our work by extending support for categorical attributes, spatio-temporal datasets, and streaming scenarios.

REFERENCES

[1] I. Sabek and M. F. Mokbel, "Machine learning meets big spatial data," *Proc. VLDB Endow.*, vol. 12, no. 12, p. 1982–1985, Aug. 2019. [Online]. Available: https://doi.org/10.14778/3352063.3352115

[2] S. Shekhar, C.-T. Lu, and P. Zhang, "Detecting graph-based spatial outliers: Algorithms and applications (a summary of results)," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 371–376. [Online]. Available: https://doi.org/10.1145/502512.502567

[3] R. Frank, M. Ester, and A. Knobbe, "A multi-relational approach to spatial classification," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 309–318. [Online]. Available: https://doi.org/10.1145/1557019.1557058

[4] F. Qian, Q. He, and J. He, "Mining spatial co-location patterns with dynamic neighborhood constraint," in *Machine Learning and Knowledge Discovery in Databases*, W. Buntine, M. Grobelnik, D. Mladenić, and J. Shawe-Taylor, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 238–253.

[5] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "Imagenet training in minutes," in *Proceedings of the 47th International Conference on Parallel Processing*, ser. ICPP 2018. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3225058.3225069

[6] K. Chowdhury, A. Sharma, and A. D. Chandrasekar, "Evaluating deep learning in systemml using layer-wise adaptive rate scaling(lars) optimizer," *CoRR*, vol. abs/2102.03018, 2021. [Online]. Available: https://arxiv.org/abs/2102.03018

[7] "2014-15 home sales in king county, wa," https://geodacenter.github.io/data-and-lab/KingCounty-HouseSales2015/, accessed: 2021-02-25.

[8] L. Wang, R. Christensen, F. Li, and K. Yi, "Spatial online sampling and aggregation," *Proc. VLDB Endow.*, vol. 9, no. 3, p. 84–95, Nov. 2015. [Online]. Available: https://doi.org/10.14778/2850583.2850584

[9] T. Guo, K. Feng, G. Cong, and Z. Bao, "Efficient selection of geospatial data on maps for interactive and visualized exploration," in *Proceedings of the 2018 International Conference on Management of Data*, ser. SIGMOD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 567–582. [Online]. Available: https://doi.org/10.1145/3183713.3183738

[10] J. Yu and M. Sarwat, "Turbocharging geospatial visualization dashboards via a materialized sampling cube approach," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, 2020, pp. 1165–1176.

[11] E. Maduekwe and W. De Vries, "Random spatial and systematic random sampling approach to development survey data: Evidence from field application in malawi," *Sustainability*, vol. 11, p. 6899, 12 2019.

[12] Y. Park, M. Cafarella, and B. Mozafari, "Visualization-aware sampling for very large databases," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, 2016, pp. 755–766.

[13] S. Biswas, F. Chen, Z. Chen, C.-T. Lu, and N. Ramakrishnan, "Incorporating domain knowledge into memetic algorithms for solving spatial optimization problems," in *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 25–35. [Online]. Available: https://doi.org/10.1145/3397536.3422265

[14] S. Biswas, F. Chen, Z. Chen, A. Sistrunk, N. Self, C.-T. Lu, and N. Ramakrishnan, "Regal: A regionalization framework for school boundaries," in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 544–547. [Online]. Available: https://doi.org/10.1145/3347146.3359377

[15] K. Kim, "Spatial contiguity-constrained hierarchical clustering for traffic prediction in bike sharing systems," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2021.

[16] W. Li, R. L. Church, and M. F. Goodchild, "The p-compact-regions problem," *Geographical Analysis*, vol. 46, no. 3, pp. 250–273, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/gean.12038

[17] J. C. Duque, R. L. Church, and R. S. Middleton, "The p-regions problem," *Geographical Analysis*, vol. 43, no. 1, pp. 104–126, 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1538-4632.2010.00810.x

[18] J. C. Duque, R. Ramos, and J. Suriñach, "Supervised regionalization methods: A survey," *International Regional Science Review*, vol. 30, no. 3, pp. 195–220, 2007. [Online]. Available: https://doi.org/10.1177/0160017607301605

[19] Z. Chen, P. Cheng, L. Chen, X. Lin, and C. Shahabi, "Fair task assignment in spatial crowdsourcing," *Proc. VLDB Endow.*, vol. 13, no. 12, p. 2479–2492, Jul. 2020. [Online]. Available: https://doi.org/10.14778/3407790.3407839

[20] R. Benedetti, F. Piersimoni, G. Pignataro, and F. Vidoli, "Identification of spatially constrained homogeneous clusters of covid-19 transmission in italy," *Regional Science Policy & Practice*, vol. 12, no. 6, pp. 1169–1187, 2020. [Online]. Available: https://rsaiconnect.onlinelibrary.wiley.com/doi/abs/10.1111/rsp3.12371

[21] M. P. Armstrong, G. Rushton, R. Honey, B. T. Dalziel, P. Lolonis, S. De, and P. J. Densham, "Decision support for regionalization: A spatial decision support system for regionalizing service delivery systems," *Computers, Environment and Urban Systems*, vol. 15, no. 1, pp. 37–53, 1991. [Online]. Available: https://www.sciencedirect.com/science/article/pii/019897159190044E

[22] K. Spence Cheruvelil, P. Soranno, M. Bremigan, T. Wagner, and S. Martin, "Grouping lakes for water quality assessment and monitoring: The roles of regionalization and spatial scale," *Environmental management*, vol. 41, pp. 425–40, 04 2008.

[23] J. C. Duque, R. Ramos, and J. Suriñach, "Supervised regionalization methods: A survey," *International Regional Science Review*, vol. 30, no. 3, pp. 195–220, 2007. [Online]. Available: https://doi.org/10.1177/0160017607301605

[24] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. [Online]. Available: https://doi.org/10.1080/14786440109462720

[25] A. Martinez and A. Kak, "Pca versus lda," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

[26] J. Tenenbaum, V. Silva, and J. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science (New York, N.Y.)*, vol. 290, pp. 2319–23, 01 2001.

[27] L. Steadman, N. Griffiths, S. Jarvis, S. McRobbie, and C. Wallbank, "2d-str: Reducing spatio-temporal traffic datasets by partitioning and modelling," 01 2019, pp. 41–52.

[28] P. A. P. MORAN, "NOTES ON CONTINUOUS STOCHASTIC PHENOMENA," *Biometrika*, vol. 37, no. 1-2, pp. 17–23, 06 1950. [Online]. Available: https://doi.org/10.1093/biomet/37.1-2.17

[29] R. C. Geary, "The contiguity ratio and statistical mapping," *The Incorporated Statistician*, vol. 5, no. 3, pp. 115–141, 1954. [Online]. Available: https://rss.onlinelibrary.wiley.com/doi/abs/10.2307/2986645

[30] S. J. Rey and L. Anselin, *PySAL: A Python Library of Spatial Analytical Methods*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 175–193. [Online]. Available: https://doi.org/10.1007/978-3-642-03647-7_11

[31] S. Moliński, "Pyinterpolate: Spatial interpolation in python for point measurements and aggregated datasets," *Journal of Open Source Software*, vol. 7, no. 70, p. 2869, 2022. [Online]. Available: https://doi.org/10.21105/joss.02869

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[33] "Spatial data re-partitioning," https://github.com/kanchanchy/spatial-repartitioning-ml.

[34] "Standard error of the regression vs. r-squared," https://statisticsbyjim.com/regression/standard-error-regression-vs-r-squared/, accessed: 2021-04-22.

[35] "What are pseudo r-squareds?" https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faq-what-are-pseudo-r-squareds/, accessed: 2021-04-25.

[36] "Micro, macro & weighted averages of f1 score, clearly explained," https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f, accessed: 2022-02-08.

[37] "Tlc trip record data," https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, accessed: 2021-01-20.

[38] "Abandoned cars in chicago," https://geodacenter.github.io/data-and-lab/1-source-and-description/, accessed: 2021-03-05.

[39] "Block-level earnings in nyc (2002-14)," https://geodacenter.github.io/data-and-lab/LEHD_Data/, accessed: 2022-02-05.
[40] "Dealing with missing data," https://www.esri.com/about/newsroom/arcuser/dealing-with-missing-data/, accessed: 2022-02-04.
[41] Y. Gong, Z. Li, J. Zhang, W. Liu, B. Chen, and X. Dong, "A spatial missing value imputation method for multi-view urban statistical data," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 1310–1316, main track. [Online]. Available: https://doi.org/10.24963/ijcai.2020/182
[42] S. R. Kuppannagari, Y. Fu, C. M. Chueng, and V. K. Prasanna, "Spatio-temporal missing data imputation for smart power grids," in *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, ser. e-Energy '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 458–465. [Online]. Available: https://doi.org/10.1145/3447555.3466586
[43] P. Amitha, V. Binu, and B. Seena, "Estimation of missing values in aggregate level spatial data," *Clinical Epidemiology and Global Health*, vol. 9, pp. 304–309, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2213398420302232
[44] J. Yu, K. Chowdhury, and M. Sarwat, "Tabula in action: A sampling middleware for interactive geospatial visualization dashboards," *Proc. VLDB Endow.*, vol. 13, no. 12, p. 2925–2928, Aug. 2020. [Online]. Available: https://doi.org/10.14778/3415478.3415510
[45] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.
[46] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Comput. Surv.*, vol. 50, no. 6, Dec. 2017. [Online]. Available: https://doi.org/10.1145/3136625
[47] K. Wu, D. Lee, A. Sim, and J. Choi, "Statistical data reduction for streaming data," in *2017 New York Scientific Data Summit (NYSDS)*, 2017, pp. 1–6.
[48] S. Lakshminarasimhan, J. Jenkins, I. Arkatkar, Z. Gong, H. Kolla, S.-H. Ku, S. Ethier, J. Chen, C. S. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "Isabela-qa: Query-driven analytics with isabela-compressed extreme-scale scientific data," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '11. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2063384.2063425