# Enhancing Communication between Ad-Hoc Vehicular Networks and UAVs using Federated Machine Learning

Mohammad Saim Khan, Bibek Paudyal

July 19, 2024

**Abstract**

This study explores the potential of federated learning to enhance communication between ad-hoc vehicular networks (VANETs) and Unmanned Aerial Vehicles (UAVs) for collaborative traffic management. We investigate the use of various distributed machine learning techniques, including k-means, single linkage agglomerative clustering, mean shift, DBSCAN, Gaussian Mixture Model (GMM), OPTICS, and BIRCH, to identify optimal communication strategies for improved network efficiency.

Simulations were conducted using the Open Street Map of Berlin and the SUMO traffic simulator with 50 vehicles for 20,000 seconds. The results suggest that federated learning can achieve model updates while preserving data privacy by exchanging model parameters instead of raw data. However, inconsistencies may arise due to diverse accuracies at the vehicle (edge) level.

The study identified GMM as a promising clustering algorithm when the optimal number of clusters is known. BIRCH was found to be effective for high-dimensional data with a large number of schemas, while Principal Component Analysis (PCA) can be used for dimensionality reduction to minimize computational costs.

Future work highlights the limitations of SUMO for simulating non-stationary nodes like UAVs. Utilizing platforms like Unity with integrated communication protocols for UAVs and vehicles is proposed for future research to analyze data transfer rates and communication speed. This data can then be leveraged for reinforcement learning to establish a robust communication network between UAVs and VANETs.

## 1 Introduction

Intelligent transportation systems (ITS) are transforming traffic management by enabling cooperative interactions between vehicles and infrastructure. Ad-hoc vehicular networks (VANETs) play a key role in ITS, allowing vehicles to communicate directly and share information. Unmanned Aerial Vehicles (UAVs), with their maneuverability and mobility, offer exciting possibilities for enhancing data collection and communication within VANETs.

However, establishing efficient communication between these dynamic networks remains a challenge. Traditional centralized learning approaches, where data is collected from vehicles and sent to a central server for model training, can be impractical due to privacy concerns and high communication overhead.

This paper explores federated learning, a distributed machine learning technique, as a potential solution for communication between VANETs and UAVs. Federated learning allows participating entities to collaboratively train a model without sharing raw data. This approach can address privacy concerns and communication costs while enabling collaborative intelligence for improved traffic management.

This work focuses on optimizing communication strategies within the network by investigating the effectiveness of various clustering algorithms in conjunction with federated learning. We present a simulation-based study using real-world map data and evaluate the performance of different algorithms for efficient communication and data exchange. The findings pave the way for the development of a communication framework between VANETs and UAVs that fosters collaboration and optimizes traffic management in smart cities.

# 2 Overview of Clustering, Clustering Techniques, and Federated Learning

This section provides a brief overview of clustering, different clustering techniques, and federated learning, highlighting their relevance to communication between ad-hoc vehicular networks (VANETs) and Unmanned Aerial Vehicles (UAVs).

## 2.1 Clustering:

Clustering is an unsupervised machine learning technique that aims to group data points into subsets (clusters) based on their similarities. These clusters represent groups of data points with similar characteristics, helping to identify patterns and structures within the data. Clustering plays a crucial role in optimizing communication by:

Data partitioning: By grouping vehicles with similar traffic patterns or locations, data exchange within the network can be focused on relevant information for each cluster.

Dimensionality reduction: Clustering can be used to group similar data points, effectively reducing the dimensionality of the data transmitted between vehicles and UAVs, leading to lower communication overhead.

## 2.2 Types of Clustering Techniques:

This research investigates the effectiveness of various clustering algorithms for communication optimization:

K-means clustering: A popular technique that groups data points into a predefined number of clusters based on their distance to a central point (centroid).

$d(x_i, c_j) = \sqrt{\sum_{k=1}^{D}(x_{ik} - c_{jk})^2}$

where: d$(x_i, c_j)$ represents the distance between data point $x_i$, and centroid $c_j$.

D represents the dimensionality of the data (number of features).

$x_{ik}$ represents the k-th feature value of data point $x_i$

$c_{jk}$ represents the k-th feature value of data point $c_j$.

$c_j^{(t+1)} = \frac{1}{|C_j|}\sum_{x_i \in C_j} x_i$

$c_j^{(t+1)}$ represents the updated centroid for cluster j in iteration (t+1).

$|C_j|$ represents the number of data points assigned to cluster j.

$x_i$ represents a data point belonging to cluster j.

Hierarchical clustering: This approach builds a hierarchy of clusters by iteratively merging or splitting data points based on their similarity. Two main types exist: agglomerative (bottom-up) and divisive (top-down).

Density-based clustering: These algorithms identify clusters based on areas of high data density, separating them from sparse regions. Examples include DBSCAN (Density-Based Spatial Clustering of Applications with Noise).

$N_eps(x_i) = \{x_j \in X | d(x_i, x_j) \leq Eps\}$

Here: $N_eps(x_i)$ represents the neighborhood of data point $x_i$ with radius Eps.

X represents the entire dataset.

$d(x_i, x_j)$ represents the distance between data points $x_i$ and $x_j$.

Core Points: A data point is considered a core point if its neighborhood has at least MinPts points. This can be written as:

$core_points(X) = \{x_i \in X | |N_{eps}(x_i)| \geq MinPts\}$

$core_points(X)$ represents the set of core points in dataset X.

Model-based clustering: This approach assumes the data follows a specific statistical model and groups data points based on their probability of belonging to a particular cluster. An example is the Gaussian Mixture Model (GMM), which assumes the data is generated by a mixture of Gaussian distributions.

## 2.3 Federated Learning:

Federated learning is a distributed machine learning technique where participating devices (vehicles in this case) collaboratively train a model without sharing their raw data directly. Instead, each device trains a local model on its own data and shares only the model updates with a central server. This approach offers several advantages:

Privacy preservation: Vehicles retain control of their raw data, mitigating privacy concerns.

Reduced communication overhead: Only model updates, typically smaller in size than raw data, are transmitted, improving network efficiency.

Scalability: Federated learning can handle large-scale networks with numerous vehicles without overwhelming a central server.

By combining clustering techniques with federated learning, this research aims to optimize communication within a network of VANETs and UAVs for collaborative traffic management.

## 2.4 Global View Architecture

The global view architecture utilizes a federated learning approach to facilitate communication between ad-hoc vehicular networks (VANETs) and Unmanned Aerial Vehicles (UAVs). This architecture aims to achieve collaborative intelligence for optimized traffic management in a privacy-preserving manner.

Here's a breakdown of the key components and functionalities:

Data Collection:

VANETs: Vehicles within the network collect data relevant to traffic conditions, such as speed, location, and density. This data can be captured by onboard sensors. UAVs: UAVs can gather data from a broader range of the network due to their mobility. They can collect similar traffic-related data as VANETs, potentially including real-time visual information. Data Clustering:

The collected data undergoes clustering using algorithms like k-means, DBSCAN, or GMM. This helps group vehicles and UAVs with similar traffic patterns or locations, enabling targeted communication and reducing data redundancy. Federated Learning:

Instead of raw data exchange, each vehicle trains a local model on its own data. This local model captures the knowledge learned from the vehicle's surrounding traffic. Only the model updates, which are significantly smaller than raw data, are transmitted to a central server or a designated UAV within the cluster. The central server or UAV aggregates the received model updates and utilizes them to update a global traffic management model. The updated global model is then broadcasted back to the vehicles and UAVs within the cluster, allowing them to improve their local models. Communication Protocols:

The architecture relies on communication protocols designed for VANETs and UAV communication, enabling efficient data exchange between vehicles and UAVs, as well as communication between UAVs and the central server. Benefits:

Privacy Preservation: By only exchanging model updates, the architecture protects the raw traffic data collected by individual vehicles. Reduced Communication Overhead: Transferring smaller model updates significantly reduces communication load compared to transmitting raw data. Scalability: The federated learning approach can accommodate a large number of vehicles and UAVs within the network. Collaborative Intelligence: The global traffic management model continuously improves through collaborative learning, leading to optimized traffic flow and congestion mitigation. Limitations:

Central Server Dependency: A central server or designated UAV is required to aggregate and distribute model updates, creating a single point of potential failure. Real-time Communication Challenges: Ensuring real-time communication between dynamic network entities like UAVs can be challenging. Future Work:

The limitations mentioned above can be addressed by exploring alternative architectures that minimize reliance on a central server and incorporate real-time communication strategies for UAVs. Reinforcement learning techniques can be leveraged to further optimize communication and network performance based on real-time traffic data. Overall, the federated learning-based global view architecture offers a promising approach for communication between VANETs and UAVs, facilitating collaborative traffic management while preserving data privacy and reducing communication overhead.

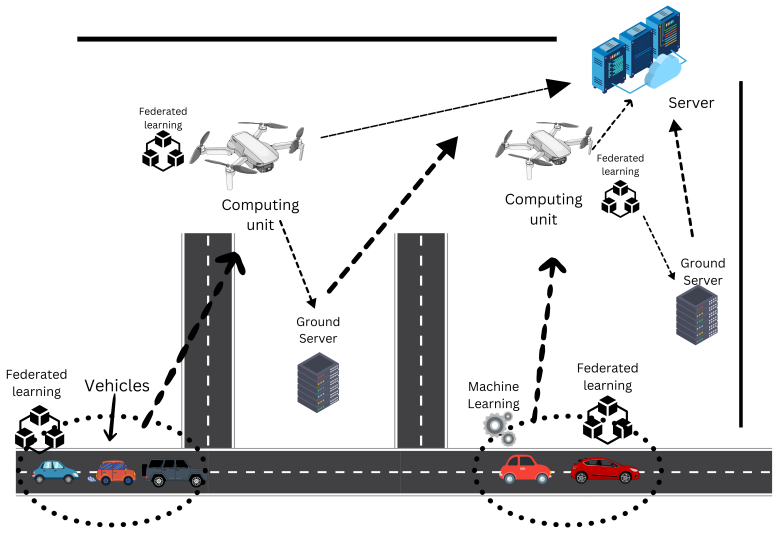


Figure 1: Global View Architecture

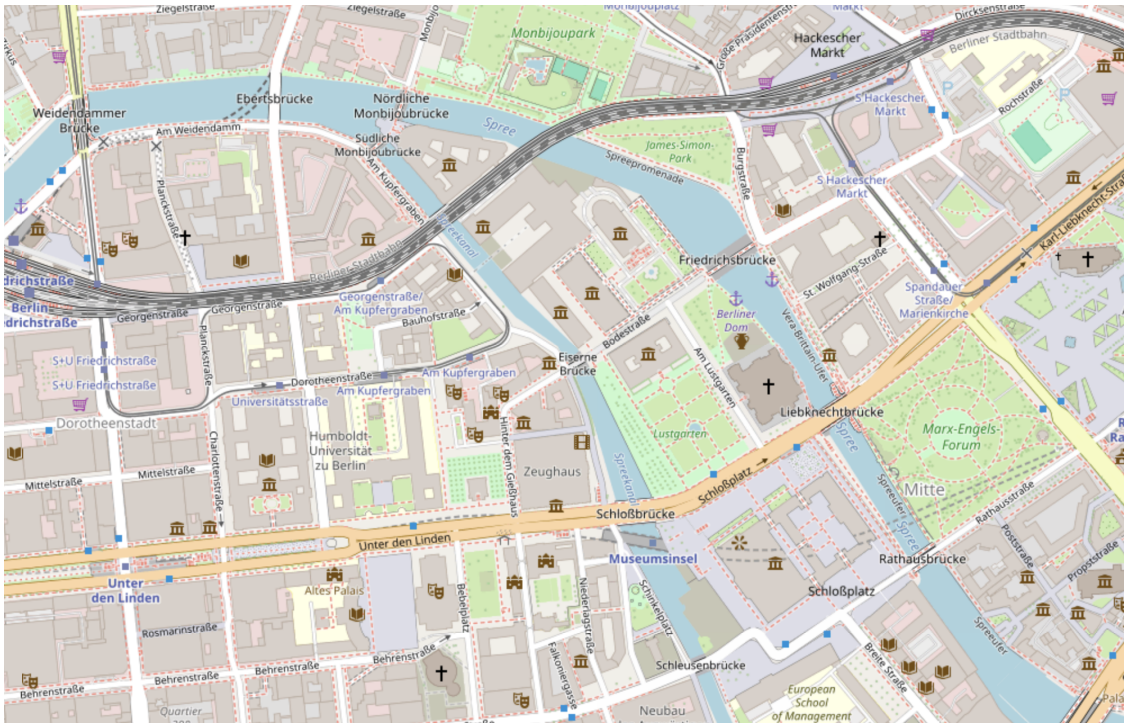


Figure 2: Open Street Map of Berlin

Figure 3: SUMO Simulation - Extracted Map

## 2.5 Open Street Map and SUMO Simulations

The Open Street Map (OSM) is providing a valuable source of data for simulating communication strategies between Ad-hoc Vehicular Networks (VANETs) and Unmanned Aerial Vehicles (UAVs). Here's a breakdown of its significance:

What is Open Street Map (OSM)?

OSM is a collaborative project that functions as a freely editable map of the entire world. Unlike traditional maps owned by corporations, OSM relies on contributions from volunteers to collect and maintain map data. This collaborative approach ensures the data is constantly updated and reflects real-world changes.

Benefits of OSM in our Project:

Realistic Traffic Representation: OSM provides detailed maps of cities, including streets, intersections, traffic signs, and even building outlines. This data can be used to create realistic traffic simulations for evaluating your communication strategies. In our case, we are using the map of Berlin as an example. Scalability: Since OSM covers entire cities, we simulate large-scale VANET deployments with numerous vehicles, which offers a more comprehensive evaluation. Data Accuracy: The collaborative nature of OSM promotes continuous updates, leading to generally accurate data for simulations. Utilizing OSM Data:

Data Extraction: OSM offers various tools and APIs that allow us to extract relevant data from the map. This data can include road network information, geographical coordinates of intersections, and traffic sign locations. Processing Tools: We use SUMO (Simulation of Urban MObility) tool to convert the extracted OSM data into a format suitable for traffic simulations.

By leveraging OSM data, we benefits from a realistic and scalable simulation environment to evaluate the effectiveness of federated learning for communication between VANETs and UAVs in traffic management.

## 2.6 SUMO Simulation

What is Sumo?

SUMO, standing for Simulation of Urban MObility, is an open-source, microscopic traffic simulation package designed to model the behavior of individual vehicles and pedestrians within an urban environment. It offers a comprehensive suite of features for creating realistic traffic scenarios, simulating

different transportation modes, and analyzing traffic flow dynamics.

The image is an extracted map of Berlin generated from Simulation of Urban MObility (SUMO), a traffic simulation software. Here is an explanation of the key elements in the map:

1. Road Network: The black lines represent the road network in Berlin. They show the various streets and highways, capturing the connectivity and layout of the city's transportation infrastructure.

2. Highlighted Route: The red segment indicates a high-traffic zone.

3. Intersections and Nodes: The points where the black lines intersect represent intersections or nodes within the traffic network. These are crucial points in traffic simulations for studying congestion, traffic light control, and route planning.

4. Geographical Layout: The overall layout corresponds to the actual geographical distribution of roads in Berlin. The map reflects the urban planning and street arrangement of the city, aiding in realistic traffic simulation.

In the context of SUMO, this map could be used to simulate various traffic scenarios, analyze traffic flow, test different traffic management strategies, or study the impact of infrastructural changes.

## 2.7 Extracted Parameters for Traffic Management with Federated Learning

This section details the parameters extracted from vehicle data used in the proposed federated learning approach for traffic management. These parameters can be categorized into two main groups:

**1. Parameters for Clustering:**

These parameters are used to group vehicles with similar traffic patterns for efficient communication and data aggregation.

time: Represents the timestamp associated with the vehicle's state. This allows for analysis of traffic dynamics over time.

id (integer): A unique identifier for each vehicle, essential for tracking individual behavior within the network.

x, y: The vehicle's position coordinates, enabling spatial analysis and identification of traffic patterns in specific locations.

angle (degrees): The vehicle's orientation angle, potentially useful for understanding traffic flow and lane changes.

speed: The instantaneous speed of the vehicle, crucial for congestion detection and traffic flow optimization.

pos: The vehicle's position along its route, indicating its progress and potential bottlenecks.

lane: The lane identifier the vehicle is currently occupying, important for lane-based traffic management strategies.

slope: The slope of the road at the vehicle's position, which can influence speed and fuel consumption.

By analyzing these parameters, clustering algorithms like k-means, DBSCAN, or GMM can group vehicles with similar spatial and temporal traffic patterns. This enables targeted communication within clusters, reducing data redundancy and improving communication efficiency.

**2. Parameters for Data Processing and Federated Learning:**

These parameters provide a richer dataset for training and evaluating the federated learning model for traffic management.

id (integer): Same as above, used for unique vehicle identification throughout the data processing pipeline.

eclass: The emission class of the vehicle, relevant for analyzing the overall environmental impact of traffic.

CO2, CO, HC, NOx, PMx:Emission levels of various pollutants from the vehicle, crucial for developing environmentally conscious traffic management strategies.

fuel: The amount of fuel consumed by the vehicle, valuable for optimizing fuel efficiency and reducing overall fuel usage.

electricity: For electric vehicles, the amount of electricity consumed, helpful for understanding electric vehicle usage patterns and optimizing charging infrastructure.

noise: The noise level generated by the vehicle, relevant for noise pollution mitigation strategies.

route: The specific route the vehicle is following, aiding in understanding traffic flow between destinations.

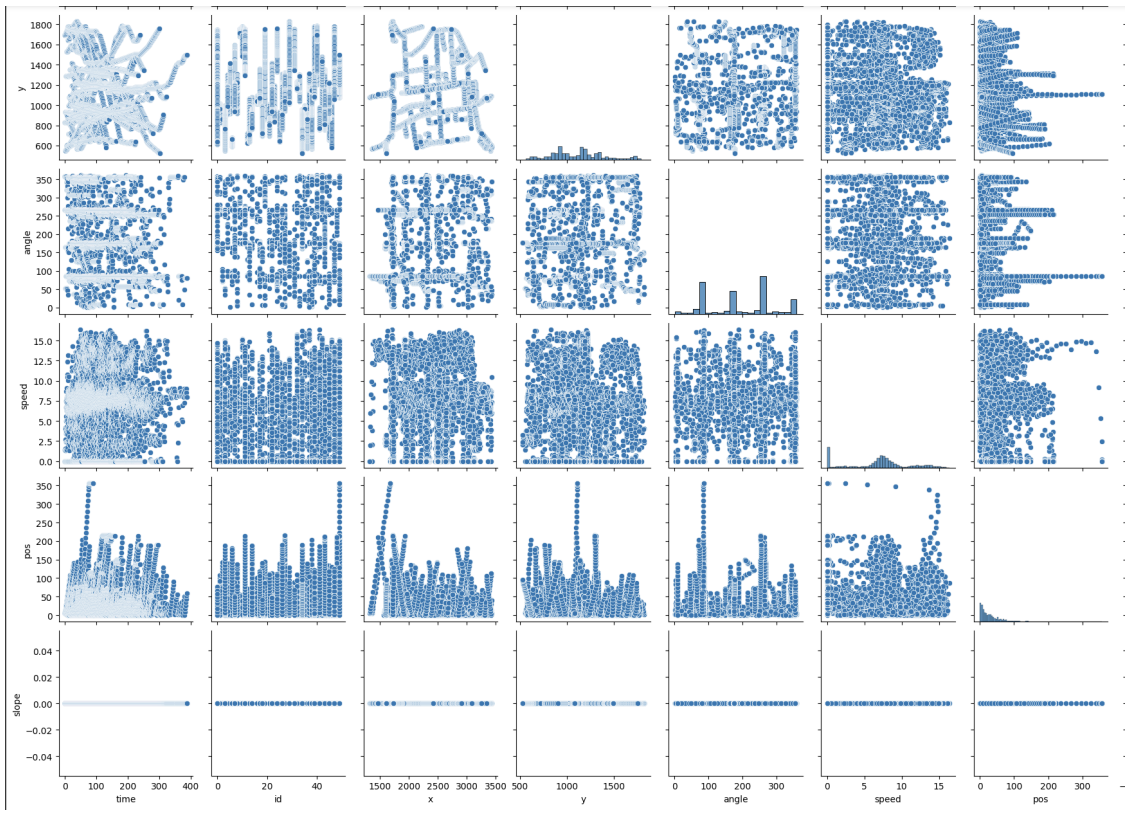


Figure 4: Outlining relation between vehicular schemas for clustering - Pair Plot Matrix 1

type: The type of vehicle (car, bus, truck, etc.), important for tailoring traffic management strategies to different vehicle classes.

waiting:The time spent by the vehicle waiting at intersections or due to congestion.

lane, pos, speed, angle, x, y: Same as above, providing detailed information for model training and evaluation.

These combined parameters offer a comprehensive dataset for federated learning. Each vehicle trains a local model based on its own data, focusing on the most relevant parameters for its specific traffic situation. Only the model updates, significantly smaller than raw data, are transmitted for federated learning, preserving privacy and reducing communication overhead.

This approach allows for collaborative learning and continuous improvement of the global traffic management model, leading to optimized traffic flow, reduced congestion, and minimized environmental impact.

## 2.8 Outlining relation between vehicular schemas for clustering

The images show pair plot matrices used for visualizing relationships between multiple variables in a dataset, aiding in clustering and understanding data patterns. Each matrix contains scatter plots and histograms illustrating the relationships between different vehicular data parameters.

**Explanation of the Pair Plot Matrices:**

1. Pair Plot Matrix 1:

- Variables: The axes are labeled with variables such as ‘time‘, ‘id‘, ‘x‘, ‘y‘, ‘angle‘, ‘speed‘, ‘pos‘, and ‘slope‘.

- Scatter Plots: The scatter plots show the relationship between each pair of variables. For example, the plot at the intersection of ‘time‘ and ‘speed‘ shows how speed varies with time.

- Histograms: Diagonal plots are histograms showing the distribution of individual variables. For instance, the histogram on the diagonal for ‘speed‘ shows how speeds are distributed across the dataset.

- Patterns and Clusters: By examining these plots, one can identify patterns, correlations, or clusters. For instance, tightly clustered points indicates a strong correlation, while dispersed points

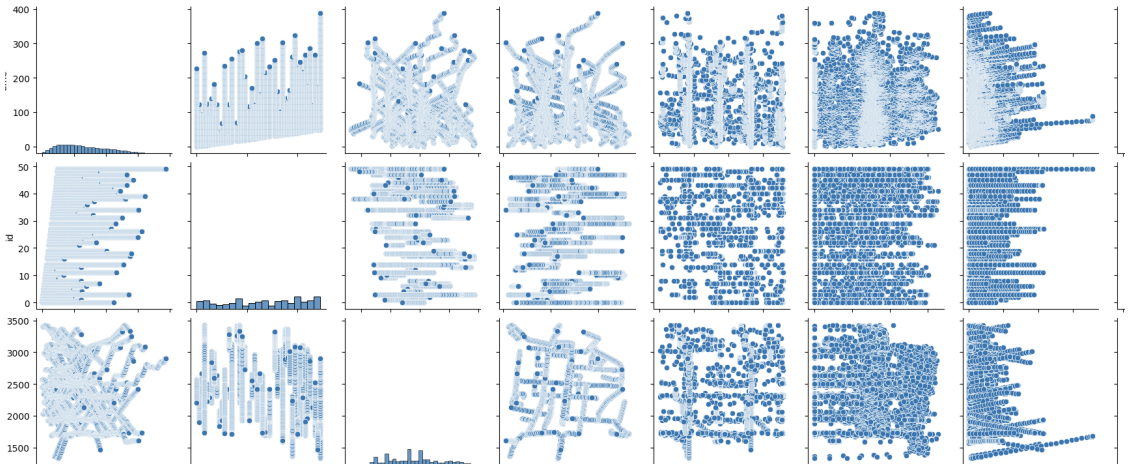

Figure 5: Outlining relation between vehicular schemas for clustering - Pair Plot Matrix 2

suggest a weaker relationship.

2. Pair Plot Matrix 2:

- Similar to the first matrix, this matrix includes additional variables or a different dataset, indicated by the different scales and plot distributions.

- Scatter Plots: Show pairwise relationships among variables.

- Histograms: On the diagonals, providing the distribution of individual variables.

- Correlation and Clustering: Useful for observing correlations and potential clustering among variables. For example, a linear pattern in the scatter plots indicates a strong correlation between variables.

**Key Observations:**

- Correlations: Strong correlations can be observed where the points form a clear pattern (e.g., linear or curved). This can help identify which variables are most closely related.

- Distributions: Histograms help understand the distribution of each variable, showing whether they are normally distributed, skewed, etc.

- Clusters: Clustering of points in scatter plots can indicate groupings or clusters within the data, useful for identifying patterns or behaviors in vehicular data.

**Applications:**

1. Traffic Analysis: Understanding how different factors such as time, speed, and position correlate can help in traffic management and planning.

2. Anomaly Detection: Identifying outliers or unusual patterns in the scatter plots can help detect anomalies in vehicular movement or behavior.

3. Data Clustering: Useful for clustering algorithms to group similar data points, aiding in tasks such as route optimization and traffic prediction.

In summary, these pair plot matrices are powerful tools for visualizing the relationships between multiple variables in vehicular data, helping to uncover patterns, correlations, and clusters that are essential for traffic analysis and clustering applications.

## 2.9 Data Preprocessing:

**Data Preprocessing for UAVs in Vehicular Ad-hoc Networks**

Data preprocessing is a crucial step in any data-driven project, especially when dealing with UAVs (Unmanned Aerial Vehicles) in vehicular ad-hoc networks (VANETs). The primary goal of data preprocessing is to convert raw data into a clean and usable format, ensuring that subsequent analysis, modeling, and machine learning tasks yield accurate and reliable results. Here, we will discuss the data preprocessing steps in detail, focusing on the parameters like: time, id, x-coordinates, y-coordinates, angle, speed, POV (Point of View), lane, slope, route, noise, etc.

**1. Data Collection**

Data is collected from various sensors and communication devices in the vehicular ad-hoc network. This raw data may include:

- Time: Timestamp of data collection.
- ID: Unique identifier for each UAV.
- x-coordinates and y-coordinates: Positional data of the UAV.
- Angle: Orientation of the UAV.
- Speed: Velocity of the UAV.
- POV (Point of View): Perspective data, possibly indicating the field of view or camera angle.
- Lane: The lane in which the UAV or vehicle is operating.
- Slope: The gradient or incline of the path.
- Route: The designated path or waypoints for the UAV.
- Noise: Measurement errors or external interference affecting sensor data.

**2. Data Cleaning**

Objective: Identify and correct errors, handle missing values, and reduce noise in the data.

Steps:

- Identify and Correct Errors: Detect anomalies or erroneous data points using statistical methods (e.g., Z-score, IQR) and domain knowledge.
- Example: If a speed value is abnormally high or negative, it could be flagged as an error.
- Handle Missing Values: Missing data can be handled through:
- Interpolation: Estimate missing values using surrounding data points.
- Linear Interpolation: Simple method using linear equations to estimate missing values.
- Polynomial Interpolation: Higher-order polynomials for more complex data trends.
- Imputation: Use statistical methods to fill in missing values.
- Mean/Median Imputation: Replace missing values with the mean or median of the column.
- K-Nearest Neighbors (KNN) Imputation: Use the average of the nearest neighbors' values.
- Noise Reduction: Filter out noise using techniques such as:
- Smoothing: Apply moving average or exponential smoothing to reduce short-term fluctuations.
- Outlier Detection and Removal: Identify and remove outliers that do not conform to the expected data distribution.

**3. Data Normalization/Standardization**

Objective: Scale the data to a standard range, ensuring that all features contribute equally to the analysis and modeling processes.

Steps:

- Normalization: Rescale data to a range of [0, 1] or [-1, 1].
- Min-Max Scaling: $X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$
- Example: Transform the speed values so they all fall within the range of 0 to 1.
- Standardization: Transform data to have a mean of 0 and a standard deviation of 1.
- Z-score Standardization: $X' = \frac{X - \mu}{\sigma}$
- Example: Apply Z-score standardization to positional data (x, y coordinates) to ensure they have a common scale.

**4. Data Transformation (if necessary)**

Objective: Convert data into a suitable format or structure for analysis and modeling.

Steps:

- Feature Engineering: Create new features from existing data to capture more information.
- Example: Calculate the distance traveled using x and y coordinates.
- Encoding Categorical Variables: Convert categorical data into numerical format.
- Example: One-hot encoding for the 'route' variable if it is categorical.

**5. Data Integration**

Objective: Combine data from different sources or sensors into a cohesive dataset.

Steps:

- Merge Data: Integrate data from multiple UAVs or sensors into a unified dataset.
- Example: Combine positional data with speed and angle data for each UAV.
- Synchronization: Ensure that data collected at different times or from different sources are aligned correctly.
- Example: Synchronize data based on the timestamp to ensure consistency.

**6. Data Validation**

Objective: Ensure the preprocessed data is accurate, consistent, and ready for analysis.

Steps:

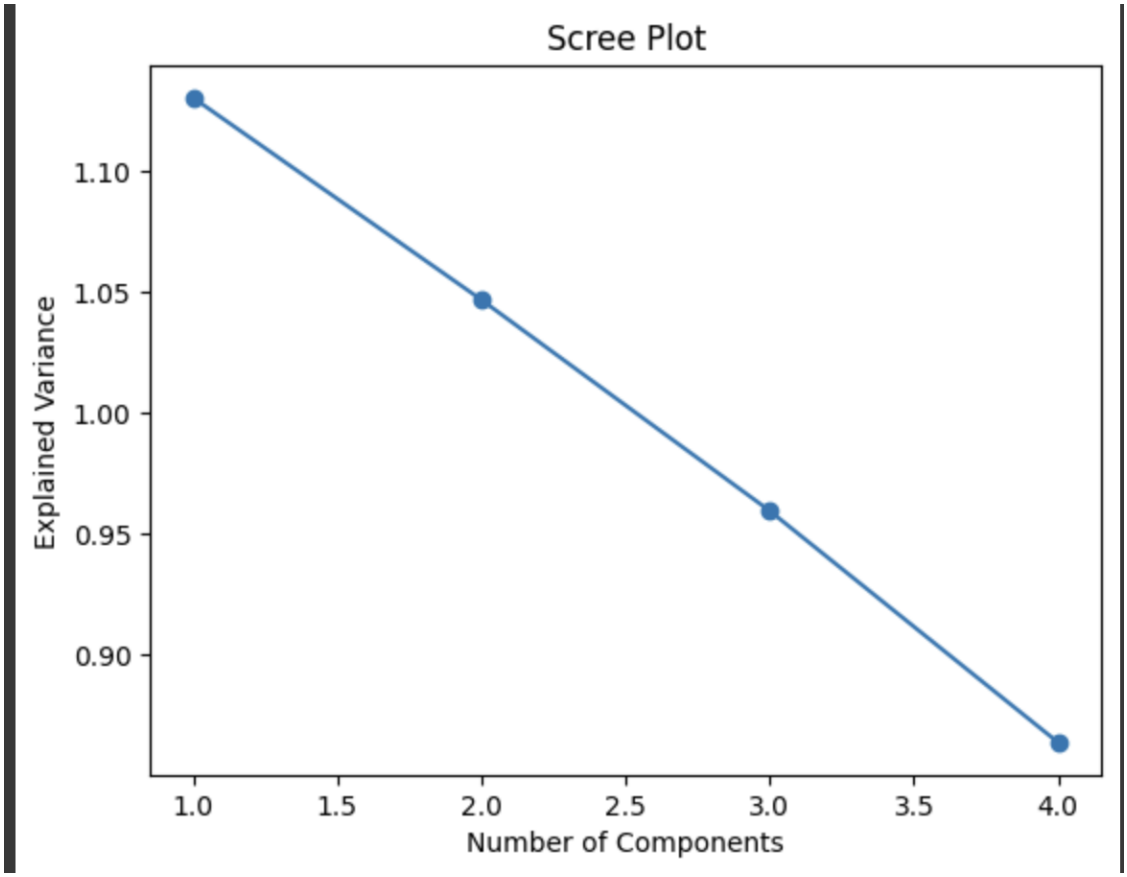


Figure 6:

- Consistency Checks: Verify that data values are logically consistent.

- Example: Ensure that speed values are always non-negative.

- Data Quality Assessment: Assess the quality of the preprocessed data using metrics such as completeness, accuracy, and reliability.

**7. Principal Component Analysis (PCA)** Objective: Reduce the dimensionality of the dataset while retaining most of the variance, simplifying the dataset for analysis and modeling.

Steps: - Standardization: Ensure all features are on the same scale before applying PCA.

- Example: Standardize variables such as x, y coordinates, speed, and angle.

- Covariance Matrix Computation: Calculate the covariance matrix to understand how variables vary with respect to each other.

- Example: Compute the covariance matrix for standardized features.

- Eigen Decomposition: Perform eigen decomposition on the covariance matrix to obtain eigenvalues and eigenvectors.

- Example: Extract eigenvectors representing the principal components and eigenvalues indicating the amount of variance explained by each component.

- Selecting Principal Components: Choose the top principal components that capture the most variance.

- Example: Select the first few principal components that explain a significant portion (e.g., 95

- Transforming Data: Project the original data onto the selected principal components to obtain a reduced-dimension dataset.

- Example: Transform the data into a new feature space defined by the principal components.

- Selecting Principal Components: Choose the top principal components that capture the most variance.

The screen plot, as shown in the image, indicates that at component 3, approximately 80

- Transforming Data: Project the original data onto the selected principal components to obtain a reduced-dimension dataset.

- Example: Transform the data into a new feature space defined by the principal components.

**Conclusion**

Effective data preprocessing, including data cleaning, normalization, transformation, integration, validation, and dimensionality reduction (PCA), is essential for accurate analysis and modeling in UAVs operating within vehicular ad-hoc networks. The screen plot provided illustrates that selecting three principal components preserves around 80

# 3 Algorithms

## 3.1 K-means Algorithm:

The K-Means algorithm is a popular clustering method used in unsupervised machine learning. It partitions a dataset into K distinct, non-overlapping clusters based on the features of the data points. The goal of K-Means is to minimize the within-cluster sum of squares (also known as inertia) and maximize the between-cluster variance. Here's a detailed explanation of the K-Means algorithm and the interpretation of the elbow method for determining the optimal number of clusters (K).

**Steps of the K-Means Algorithm**

1. Initialization: - Randomly select K initial centroids from the dataset. These centroids can be chosen randomly or using specific initialization methods like K-Means++.

2. Assignment Step: - Assign each data point to the nearest centroid. This creates K clusters, where each data point belongs to the cluster with the closest centroid. The distance metric commonly used is Euclidean distance.

3. Update Step: - Recalculate the centroids as the mean of all data points assigned to each cluster.

4. Convergence: - Repeat the assignment and update steps until the centroids no longer change significantly or a maximum number of iterations is reached. Convergence implies that the clusters are stable and no longer changing significantly.

**Objective Function**

The objective of K-Means is to minimize the sum of squared distances between each data point and its corresponding centroid. Mathematically, this can be expressed as:

$$J = \sum_{i=1}^{K} \sum_{j=1}^{N} \|x_j^{(i)} - \mu_i\|^2$$

where $x_j^{(i)}$ is a data point belonging to cluster $i$, $\mu_i$ is the centroid of cluster $i$, and $N$ is the number of data points.

**Elbow Method for Optimal K**

The elbow method is a heuristic used to determine the optimal number of clusters for K-Means clustering. It involves running the K-Means algorithm for a range of values of K and plotting the within-cluster sum of squares (inertia) against the number of clusters. The point at which the plot forms an "elbow" is considered the optimal K.

Steps for the Elbow Method

1. Run K-Means: Execute the K-Means algorithm for a range of K values (e.g., 1 to 20).

2. Calculate Inertia: For each K value, calculate the within-cluster sum of squares (inertia).

3. Plot Inertia vs. K: Plot the inertia values against the corresponding K values.

4. Identify the Elbow Point: The optimal number of clusters is typically at the "elbow point" where the inertia starts to decrease at a slower rate.

**Interpretation of the Elbow Plot**

The elbow plot shows the inertia (within-cluster sum of squares) on the y-axis and the number of clusters (K) on the x-axis.

From the plot, we observe that the inertia decreases significantly as the number of clusters increases, up to a certain point. After this point, the decrease in inertia slows down, forming an elbow-like shape. The optimal K is generally located at this elbow, where adding more clusters doesn't significantly reduce the inertia.

In your elbow plot:

- The inertia decreases steeply from 2 to 5 clusters. - The rate of decrease slows down after K = 5. - The elbow appears to be around K = 5, suggesting that 5 clusters may be optimal for our data.

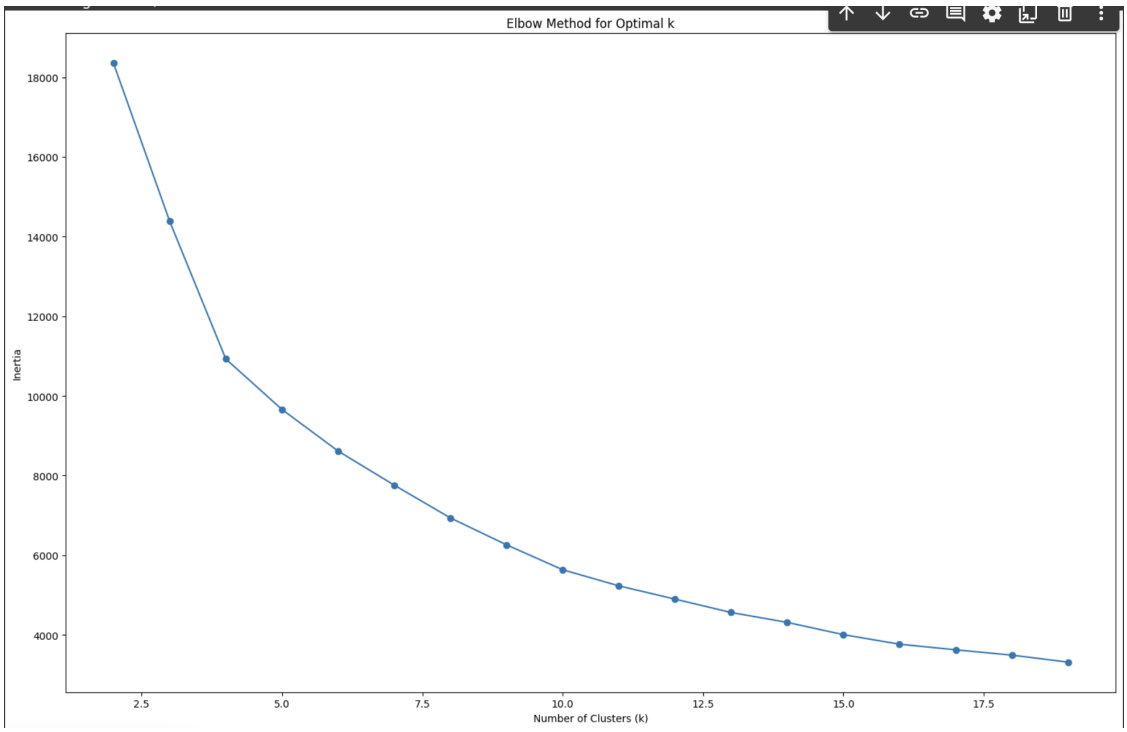


Figure 7: Elbow Method for Optimal K

**Conclusion**

The K-Means algorithm is a powerful clustering technique for partitioning data into distinct groups based on similarity. The elbow method helps determine the optimal number of clusters by identifying the point where increasing the number of clusters yields diminishing returns in terms of reducing inertia. In our case, the elbow plot suggests that using 5 clusters provides a good balance between minimizing inertia and avoiding overfitting with too many clusters.

By effectively clustering the data, we improves the efficiency and accuracy of the communication protocols between UAVs and vehicular ad-hoc networks (VANETs). The clustering helps in organizing the data more effectively, allowing the federated machine learning models to learn more accurate patterns and make better predictions or decisions.

## 3.2 Evaluating K-Means Clustering with Silhouette Score and Davies-Bouldin Index

Evaluating the performance of clustering algorithms like K-Means is crucial to ensure that the clusters formed are meaningful and useful. Two common metrics for evaluating clustering performance are the Silhouette Score and the Davies-Bouldin Index. Let's delve into these metrics and interpret the results from the provided image.

**Silhouette Score**

The Silhouette Score measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation). It ranges from -1 to 1, where:

- A score close to 1 indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.
- A score around 0 indicates that the object is on or very close to the decision boundary between two neighboring clusters.
- A negative score indicates that the object might have been assigned to the wrong cluster.

Mathematically, the Silhouette Score for a single sample $i$ is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

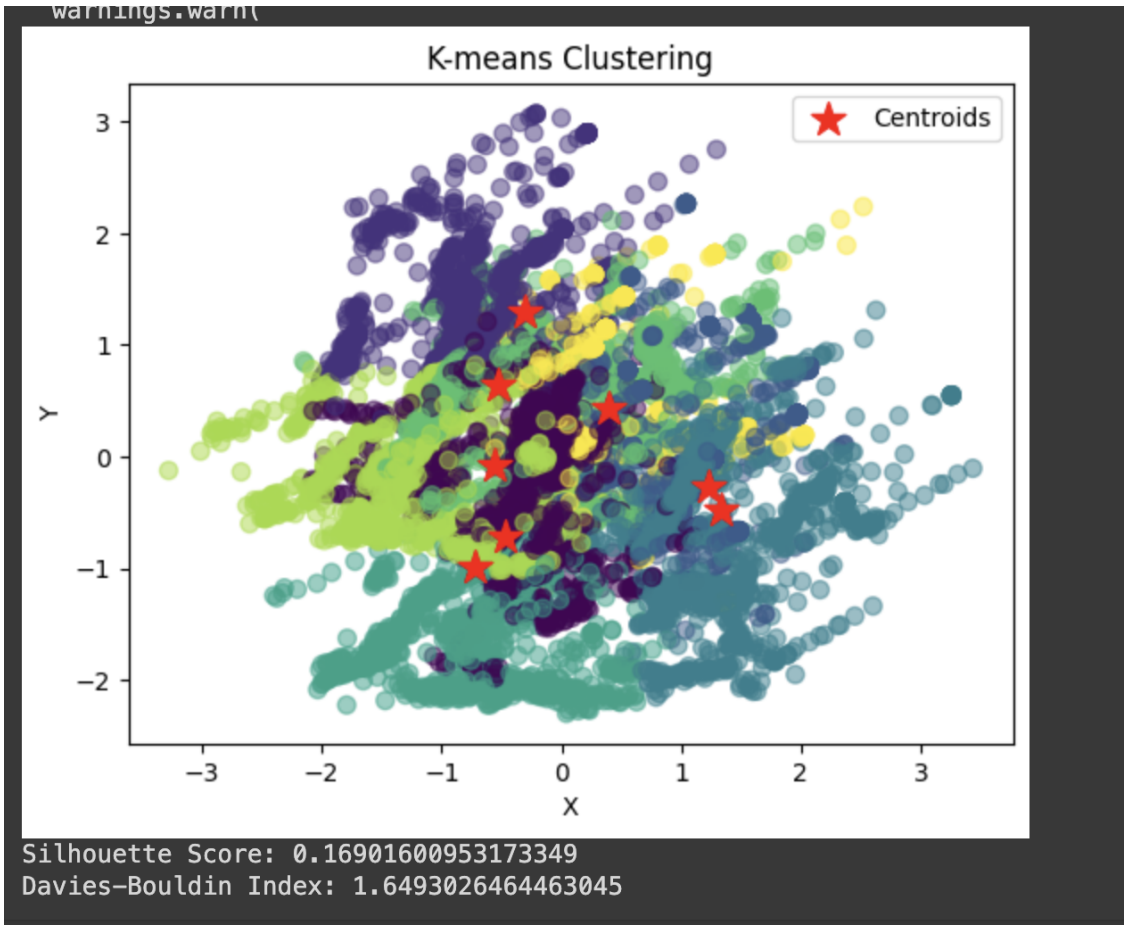


Figure 8: Silhouette Score and Davies-Bouldin Index - K-means

where:

- $a(i)$ is the average distance between the sample $i$ and all other points in the same cluster.

- $b(i)$ is the minimum average distance from the sample $i$ to all points in any other cluster (of which $i$ is not a member).

The overall Silhouette Score is the mean of $s(i)$ over all samples.

In your K-Means clustering result, the Silhouette Score is 0.1690, which is relatively low. This suggests that there is considerable overlap between the clusters, indicating that the clustering structure is not very strong or well-defined.

**Davies-Bouldin Index**

The Davies-Bouldin Index (DBI) is another metric to evaluate clustering algorithms. It is defined as the average similarity ratio of each cluster with its most similar cluster. Lower values of DBI indicate better clustering performance. The DBI for a set of clusters is calculated as follows:

$$DBI = \frac{1}{K} \sum_{i=1}^{K} \max_{j \neq i} \left( \frac{\sigma_i + \sigma_j}{d_{ij}} \right)$$

where: - $K$ is the number of clusters.

- $\sigma_i$ is the average distance between each point in cluster $i$ and the centroid of cluster $i$.

- $d_{ij}$ is the distance between the centroids of clusters $i$ and $j$.

The Davies-Bouldin Index for your K-Means clustering result is 1.6493. Lower values of DBI represent better partitioning, and while there's no strict threshold for good clustering, lower values generally indicate better-defined clusters. The value of 1.6493 suggests that the clusters are reasonably well-separated but there is room for improvement.

**Interpretation of the Results**

The plot shows the clusters formed by the K-Means algorithm with the centroids marked. Based on the silhouette score and Davies-Bouldin index: - The Silhouette Score of 0.1690 indicates that the clusters are not very distinct and there may be substantial overlap between them.

- The Davies-Bouldin Index of 1.6493 suggests that the clusters are moderately well-separated but could be improved for better clustering performance.

**Conclusion**

In the context of our project, "Enhancing Communication between Ad-Hoc Vehicular Networks and UAVs using Federated Machine Learning," these clustering evaluations are vital. Well-defined clusters can significantly improve the efficiency of communication protocols and the effectiveness of machine learning models. The current metrics indicate that while the clustering is moderately successful, there is potential for improvement.

Possible ways to enhance clustering performance include: - Trying different values for K and re-evaluating the clusters.

- Using more advanced clustering techniques like Gaussian Mixture Models (GMM) or DBSCAN.

- Feature engineering to create more informative features that can help in better distinguishing between clusters.

By iterating on these steps, we can improve the clustering performance, leading to more effective and efficient communication strategies in our network.

### 3.3 Mean-Shift Clustering Algorithm

## References