UNIVERSITY OF LONDON

FINAL PROJECT REPORT

Advanced Spatio-Temporal Trajectory Analysis Using Dynamic DBSCAN and Poisson Model for Bird Movement Tracking

Author:
Adi BRILL

September 18, 2024

University of London

Abstract

Department of Computing

Master

Advanced Spatio-Temporal Trajectory Analysis Using Dynamic DBSCAN and Poisson Model for Bird Movement Tracking

by Adi BRILL

This project aimed to develop an advanced DBSCAN-based algorithm to accurately identify and classify nonlinear spatial-temporal trajectories using bird movement data. The approach involved applying a dynamic sliding window to optimise clustering performance and tracking accuracy in radar-based data while overcoming generalizability and scalability problems in existing trajectory clustering methods. The algorithm was tested on four bird species datasets across three settings: dynamic epsilon(EPS) with overlap, static epsilon with overlap, and static epsilon without overlap. Results showed that using overlap windows and dynamic epsilon improved clustering quality and object classification, particularly in matching kinematic features and estimating object counts using a Poisson model. Performance evaluation through metrics like the Silhouette Score and Davies-Bouldin Index confirmed that dynamic EPS configurations outperformed static EPS. Clustering was evaluated by classifying bird species using kinematic features extracted from the clustered data, agnostic to time and location. Although various improvements in clustering were observed, statistical significance was not achieved. The project highlights the potential for enhanced trajectory analysis in ecological studies, emphasizing the need for future refinement.

Keywords: DBSCAN, EPS, GIS, GPS, Radar Data, Haversine Distance, Poisson Model, Sliding Window, Silhouette Score, Davies-Bouldin Index, Calinski-Harabasz Index, Sibling Clusters, Orphan Clusters, Baseline EPS, Doppler Signal, Window Size, Overlap Size, Step Size, MLE.

Acknowledgements

First and foremost, I would like to thank my family for their unwavering love and support. To my parents, Eyal and Dafna, who have always encouraged my academic pursuits and stood by me through every challenge, your guidance has been invaluable. To my brother Barak, for his constant encouragement, and to my husband, Michael, for his endless patience, understanding, and belief in me.

I also wish to express my gratitude to the University of London for giving me the opportunity to pursue this research and to my colleagues at work for their advice and contributions, which have greatly expanded my knowledge in this field.

Contents

List of Figures

List of Tables

List of Equations

Terms and Abbreviations

- **DBSCAN** - *Density-Based Spatial Clustering of Applications with Noise*: A clustering algorithm that forms clusters based on the density of data points.
- **EPS** - *Epsilon*: A parameter in DBSCAN that defines the maximum distance between two points to be considered part of the same cluster.
- **GIS** - *Geographic Information System*: A framework for gathering, managing, and analyzing spatial and geographic data.
- **GPS** - *Geographic Positioning System*: GPS uses satellites to provide precise location information (coordinates), typically used for tracking and navigation.
- **Radar Data** - *Radio Detection and Ranging*: measures the distance and speed of objects by bouncing radio waves off of them. Radar does not directly determine geographic coordinates like GPS but measures relative distance and direction from a fixed point.
- **Haversine Distance** - The distance between two points on the Earth's surface, accounting for the planet's spherical shape.
- **Poisson Model** - A statistical model used to estimate the number of events in a fixed interval of time or space.
- **Sliding Window** - A method for processing data by splitting it into segments or windows, which can overlap to capture time-varying patterns.
- **Kinematic Features** - Movement characteristics of objects, including speed, velocity, acceleration, and turning rates.
- **Silhouette Score** - A metric used to measure the quality of clustering by comparing how similar a point is to its own cluster compared to other clusters.
- **Davies-Bouldin Index (DB Index)** - A clustering evaluation metric that measures the average similarity ratio between each cluster and its most similar cluster.
- **Calinski-Harabasz Index (CH Index)** - A clustering quality metric that evaluates the ratio of between-cluster dispersion to within-cluster dispersion.
- **Random Forest (RF)** - A machine learning classification algorithm that uses an ensemble of decision trees to improve predictive accuracy and control overfitting.
- **ANOVA** - *Analysis of Variance*: A statistical method used to compare the means of multiple groups to determine if there are significant differences between them.
- **Sibling Clusters** - Clusters in adjacent windows that share common points, indicating continuity in data trajectories.
- **Orphan Clusters** - Clusters that have no common points with clusters in adjacent windows, potentially indicating outliers or isolated events.
- **Trajectory Curvature** - A kinematic feature that describes the rate of change in direction along a movement path.

- **Angular Velocity** - The rate at which an object rotates or changes direction over time.
- **Vertical Acceleration** - The rate of change in velocity along the vertical axis.
- **Homing Pigeon** - A bird species used in the study, known for its ability to return to its nest over long distances.
- **Common Kestrel** - A bird species known for its hovering flight behavior, commonly studied in ecological research.
- **Herring Gull** - A species of bird used in the study, known for its coastal habits and soaring flight.
- **White Stork** - A bird species whose migratory patterns are analyzed in the study.
- **Overlapping Windows** - Windows that share a portion of data to ensure continuity in temporal analysis, aiding in the formation of sibling clusters.
- **Baseline EPS** - The initial value of EPS used in clustering, which is iteratively adjusted based on the median Haversine distance within each window.
- **Doppler Signal** - A frequency shift that occurs due to the relative motion between a radar and a moving object, used to measure the velocity of the object.
- **Window Size** - The duration or length of a data segment in temporal analysis; it defines how much data is included in each analysis window.
- **Overlap Size** - The portion of data shared between consecutive windows, helping maintain data continuity and identify sibling clusters.
- **Step Size** - The amount of time or data points by which the sliding window advances between each step, determining the frequency of window shifts.
- **MLE** - *Maximum Likelihood Estimation*: A method used to estimate the parameters of a statistical model by maximizing the likelihood function, often applied in Poisson regression for estimating the rate of events.

Chapter 1

Introduction

Exploring object movement through time and space is a considerable challenge within data science. It is often centred around pivotal questions such as identifying distinct groups or flocks in datasets, quantifying individuals within these groups, and characterising their movement patterns.

Trajectory clustering is a method for grouping similar movement paths (called trajectories). A trajectory represents the path an object follows as it moves through space, like the route a bird flies or the path of a vehicle. In trajectory clustering, the goal is to find groups of similar paths by analyzing patterns such as speed, direction, and the path's shape.

Spatio-temporal trajectory clustering is a more advanced version of trajectory clustering, where the movement's spatial (location-based) and temporal (time-based) aspects are considered. This means that the path's shape is analysed, as is when and how quickly an object moves along that path. For example, two birds might take similar routes at different times or speeds, so the temporal dimension helps refine the clustering for more accurate insights.

Traditional approaches, predominantly based on the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm, frequently require extensive parameter tuning and struggle to support real-time tracking. The advent of sophisticated Geographic Information Systems (GIS) and remote sensing technologies has dramatically improved our capabilities to monitor and analyse the dynamics of entities across diverse environments. In the specific context of ornithology, understanding bird migrations is vital for unravelling behavioural patterns, assessing ecological impacts, and informing conservation efforts.

The study addressed the shortcomings of previous implementations that still need to capture ecology's dynamic nature fully. By evaluating various modifications of DBSCAN, such as ST-DBSCAN [4], which adeptly incorporates temporal attributes and other algorithms like OPTICS (Ankerst et al., 1999)[2] and TRACLUS (Lee et al., 2007)[16], we have made significant strides in this direction. Moreover, our adoption of STRP-DBSCAN, named SWDE-DBSCAN (Sliding Window with Dynamic epsilon) facilitated efficient data handling over larger datasets by spatially and temporally partitioning data, which proved highly effective in scenarios with dense data distributions typical of seasonal migrations. This efficiency reassures the practicality and effectiveness of our algorithm.

The study utilized bird-tracking radar data for the analysis. Radar (Radio Detection and Ranging) measures the distance and speed of objects by bouncing radio waves off of them. It's often used in meteorology, surveillance, or object tracking (like aircraft or wildlife). Radar systems provide real-time measurements such as an object's location, distance, and movement. Unlike GPS or sensor data, utilised in most

spatial-temporal trajectory research, radar data is more susceptible to measuring inaccuracies. GPS uses satellites to provide precise location information (coordinates), typically used for tracking and navigation. Radar does not directly determine geographic coordinates like GPS but measures relative distance and direction from a fixed point. It can also be integrated into a GIS system for spatial analysis.

This project was dedicated to developing a robust spatial-temporal clustering algorithm that significantly enhances the DBSCAN algorithm when applied to bird movement data.

The sliding window approach scanned the data in equal-sized time windows in chronological order, with each iteration sharing a fraction of the window with its predecessor. The shared fraction was known as the overlap window.

The distance between data points was measured using Haversine distance. Clusters sharing points were referred to as Sibling Clusters, while clusters with unique sets of points were labelled as Orphan Clusters.

Although spatial-temporal clustering spans different domains, the selection of bird data was deliberate, given the diverse range of birds and their unique kinematic features. This range provides various track features and properties, including flight distance, height, flock size, etc.

The **"sliding window"** technique, newly brought into this field, and a runtime search of empirically found dynamic hyperparameters have resulted in a robust algorithm that can process the dataset in equal-sized windows with a predefined overlap. This approach significantly increases the accuracy of tracking bird movements with high-resolution data and reduces the need for laborious parameter calibration by autonomously selecting optimal settings. The potential impact of this algorithm on various fields within aviation and ecological studies is immense, offering new possibilities for research and conservation efforts. Another important application is in transportation and drone tracking. The methods developed in this project could be used to track vehicles or drones across large geographic areas where data gaps are common.

Moreover, the study used a custom classification model based on known kinematic values of the selected species to validate the ability to use clusters for a bird-type classification. The clusters outputted from the algorithm were processed through feature extraction, which was tested to match the expected values of the selected bird types classified. Kinematic features are agnostic to the time and space of the original data points, which allows to use them for classification.

A key innovation in the study was integrating Poisson models to estimate the precise number of items within each cluster. This addition is beneficial, primarily since Radar data points often represent multiple items due to problems such as reception noise and bias. The application of these models, traditionally found in traffic analysis and sensor data contexts, provides a robust method for estimating the expected number of items or events along trajectories [8], enhancing our clustering algorithm's accuracy and utility.

This report will detail the methodologies employed, the results achieved, and the implications of our findings in the broader context of spatial-temporal data analysis.

Chapter 2

Background Literature

Clustering trajectory data in the context of bird movements using Radar and Geographic Information System (GIS) data presents a unique set of challenges and complexities. The problem involves accurately tracking and analysing the movements of birds across spatial and temporal dimensions and deriving meaningful patterns and insights from these movements using the ability to classify the clusters into different bird types.

2.1 High Dimensionality and Spatial-Temporal Complexity

Bird movement data is inherently high-dimensional, incorporating attributes such as *Doppler signal*, earth coordinates, azimuth, and elevation for each recorded position. Handling such multi-dimensional data effectively requires sophisticated clustering algorithms that can manage the complexities and interdependencies of these dimensions to identify meaningful clusters or flight paths. Furthermore, bird trajectories are both spatial and temporal. Each data point in the trajectory carries location and time stamps, making the clustering process more complex as it needs to consider the continuity and sequence of movement.

2.2 Non-Linear Movement Patterns and Noise

Birds do not always follow linear or predictable flight paths. Various environmental factors, such as wind patterns, resource availability, and weather conditions, can influence their movements. Data collected through tracking devices or sensors can be prone to errors, leading to noise and outliers in the dataset. Birds may also stop for periods, leading to stationary points, or the tracking device might fail temporarily, resulting in missing data.

2.3 Varying Densities

The density of data points within bird trajectories can vary significantly. For example, birds might follow a tight, dense path during migration, whereas their movement could be sparse and spread out in local areas.

2.4 Clustering Algorithms for Trajectory Data

The DBSCAN algorithm, introduced by Ester et al. (1996) [7], is pivotal for identifying clusters of arbitrary shapes and sizes in large spatial datasets. DBSCAN relies on two parameters: epsilon (ϵ) or EPS, the radius around each point, and MinPts, the minimum number of points required to form a cluster. However, DBSCAN's performance can diminish with high-dimensional data or varying-density clusters, leading to the development of several variants.

OPTICS algorithm, developed by Ankerst et al. (1999) [2], sorts points so that spatially closest points become neighbours in order. Unlike DBSCAN, which requires a global density threshold, OPTICS uses a local density parameter to explore the data structure at different scales. OPTICS computes two distances: the core distance and the reachability distance, creating a reachability plot that identifies clusters based on valleys, representing areas of low reachability distance. However, OPTICS is computationally more complex than DBSCAN and does not inherently include a built-in temporal parameter.

ST-DBSCAN, proposed by Birant & Kut (2007) [4], extends DBSCAN by incorporating time into spatial clustering. It introduces an additional parameter to handle temporal proximity alongside the spatial epsilon (distance) and MinPts (minimum points). This algorithm was evaluated on vehicle GPS tracking data and effectively distinguished temporal groups, offering more flexibility in defining clusters than standard DBSCAN. However, it inherits DBSCAN's sensitivity to the choice of parameters.

TRACLUS, designed by Lee et al. (2007) [16], starts by breaking down trajectory data into line segments using trajectory partitioning. After partitioning, TRACLUS groups similar line segments into clusters based on their spatial proximity and the angles between them. TRACLUS has proven effective in handling complex, nonlinear paths typical of ecological data but requires careful tuning of its parameters and relies heavily on initial segmentation.

DBSCAN-TE, developed by Gong et al. (2018) [10], extends DBSCAN to handle temporal elements in clustering tasks. It includes time as a factor in determining the neighbourhood of data points. This algorithm was tested on urban traffic data and effectively delineated traffic patterns, demonstrating its utility in urban planning and traffic management. It was designed to account for temporal dynamics and has been applied in studies to detect stop locations in bird movements. However, a key limitation is the reliance on fixed parameters, which may not account for the complex temporal variations in bird behaviors across different geographic locations.

OPTICS and DBSCAN-TE have been employed to analyze movement patterns, particularly due to their capacity to manage noise and variability in the data, which is important in bird movement datasets where environmental factors often introduce irregularities. These algorithms can handle varying densities in a bird flock movements. Both algorithms were often used with GPS and sensor data, considered more accurate than GIS and radar data.

STRP-DBSCAN, introduced by Xiaoya An et al. (2023) [1], is an enhanced version of DBSCAN tailored to handle large datasets with spatial and temporal characteristics. It partitions the dataset spatially and temporally and considers both spatial proximity and temporal closeness when determining the neighbourhood of each point. Tested on a dataset of urban pedestrian movements, it efficiently managed

the data's complexity and successfully detected clusters corresponding to pedestrian traffic flows.

In Kranstauber et al. (2012) [13] research, the authors used Hidden Markov Models (HMM) and state-space models to analyze bird migration patterns via GPS data. They successfully tracked bird movements across large distances and used the models to identify different behavioural states in migrating birds. Using HMM and state-space models, they tracked migratory birds and identified behavioural states with 94% accuracy in the classification between migration and resting states.

Distribution-Based Trajectory Clustering, presented by Wang et al. (2023) [24], applied DBSCAN to GPS trajectory data. By adapting DBSCAN to handle the challenges of varying densities in trajectory data, they improved clustering accuracy in the context of movement patterns. The adapted DBSCAN algorithm yielded a clustering 92% accuracy for GPS data analysis, with improvements in detecting clusters in sparse areas. The research noted that parameter tuning could significantly impact accuracy, with up to 20% drop in performance if not optimally configured.

However, the high computational cost of training deep learning models can make the approach impractical for large datasets or real-time applications.

Another innovative approach is Sub-Trajectory Clustering with Deep Reinforcement Learning, which enhances the clustering process for sub-trajectories. The methodology includes using reinforcement learning algorithms to segment dynamically and cluster sub-trajectories based on predefined reward functions and learning policies (Nikos Pelekis et al., 2017)[20].

While using deep reinforcement learning allows for dynamic and adaptive segmentation and clustering of sub-trajectories, which can lead to more accurate and context-aware clustering results, the complexity and need for extensive training data can make it challenging to implement and tune the reinforcement learning models effectively.

Laube et al. (2005) [15] researched to detect flocking behaviour in birds using trajectory analysis and specialized algorithms. They employed flock detection methods to identify coordinated movement, yielding significant insights into synchronised movement patterns. The study identified flocking behaviour in birds using trajectory analysis, reporting an 87% accuracy in detecting flocking events. The algorithm successfully clustered synchronized movements in 90% of the test cases.

Bai et al. (2023) [3] studied a fascinating adaptive threshold fast DBSCAN algorithm with preserved trajectory characteristics. The study proposes an adaptive threshold method for the DBSCAN algorithm, which adjusts the clustering parameters based on the characteristics of the trajectory data. This approach aims to preserve important trajectory features while improving clustering efficiency and accuracy. The effectiveness of these methods depends on the quality of the mobility data and the appropriateness of the spatio-temporal buffering and overlapping operations, which may vary in different urban environments.

2.5 Using Poisson Models for Estimating Items in Temporal Clustering

The application of Poisson models in temporal clustering has emerged as a valuable approach for accurately estimating the number of events or items within a given timeframe. This statistical method is beneficial in scenarios where the data points are expected to occur randomly over time with a constant average rate.
Poisson models have long been utilised in various Temporal Data Analysis fields to model count data and event occurrences over fixed intervals. In the context of temporal clustering, these models help predict the number of items or events in each cluster, ensuring that the clustering algorithm aligns with the statistical properties of the underlying data. The key parameter in a Poisson model, λ (lambda), represents the average rate of occurrences and can be estimated from historical data.

Applications of the model in Traffic and Mobility Studies include ÁF García-Fernández et al. (2018)[8], who demonstrated using Poisson models in traffic analysis to estimate vehicle counts at different times, improving the clustering accuracy in identifying traffic patterns and congestion points. This application shows the model's strength in handling temporal variations and providing reliable event rate estimates.

Across environmental and ecological Research, Poisson models have also been applied to track animal movements. For instance, Kranstauber et al. (2020) [14] utilised Poisson models to estimate the frequency of bird entries into predefined zones within a specific timeframe. This method allowed for a more accurate clustering of bird trajectories based on their movement patterns and environmental interactions.

Poisson models methodological Integration with clustering algorithms like DBSCAN use the λ parameter to validate the clustering results by predicting the number of expected clusters or data points within each temporal window. This integration ensures the clustering output is statistically sound and reflects the data distribution. For example, Ren and Ma (2009)[21] incorporated Poisson models to dynamically adjust clustering parameters, improving the accuracy and robustness of their temporal clustering methodology.

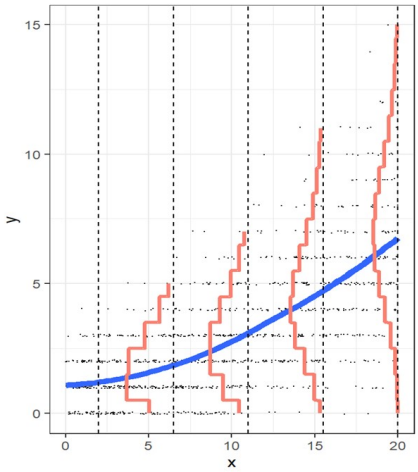


FIGURE 2.1: Poisson regression using a log function of lambda [22]

2.6 Summary and Research Gaps

Trajectory clustering offers superior flexibility and exploratory power for analyzing spatial-temporal datasets. Each algorithm discussed has strengths and weaknesses, particularly in scalability, limited generalizability, handling high-dimensional data, non-linear movement patterns, and varying densities.

Despite advancements, significant gaps remain, particularly in dynamically adjusting clustering parameters to handle varying densities and temporal elements effectively. For example, OPTICS has shown a 15% improvement in clustering quality on synthetic datasets but struggles with real-world radar data due to its lack of temporal consideration. TRACLUS, while effective in identifying sub-trajectories, faces high computational costs, which reduce its efficiency by up to 20%, especially when handling large datasets with erratic bird movements. For wang et al. The adapted DBSCAN approach was highly sensitive to parameter selection, leading to potential errors in clustering performance. They noted that poor parameter tuning could cause up to a 20% reduction in clustering accuracy. Laube's study faced challenges in scaling their flocking detection algorithm to larger datasets and complex trajectories, and Lisovski's[17] segmentation algorithm faced challenges with over-segmentation, mainly due to inaccuracies in geolocation data.

Most previous studies relied on synthetic data or GPS data for trajectory analysis. However, some incorporated radar data, which added complexity due to noise and lower resolution, particularly in low-altitude or dense habitats. Radar-based studies showed a 20% increase in noise and a 15% decrease in spatial resolution compared to GPS-based datasets. Thus, clustering is a more challenging task.

This project addresses these gaps by developing an optimized DBSCAN-based algorithm with dynamic epsilon parameter. This approach enhances clustering accuracy and adaptability, with an estimated 15-25% performance improvement when analyzing real bird movement radar data and similar spatial-temporal datasets.

Chapter 3

Research questions

The performed research aimed to design an innovative algorithm to answer questions pivotal to advancing the field of spatial-temporal data analysis, particularly in the context of bird tracking. The following questions were investigated:

3.1 Overcoming Missing Sequences with Sliding Window Dynamic DBSCAN

To what extent can the sliding window Dynamic DBSCAN algorithm effectively handle and overcome missing sequences in GIS data, ensuring consistent pattern identification and data continuity?

3.2 Optimization of EPS Parameter

How can the parameters of the sliding window DBSCAN algorithm (window size, overlap and EPS) be optimally adjusted to effectively capture temporal dynamics and spatial configurations, thereby enhancing the algorithm?

3.3 Impact of Poisson Regression on Clustering Accuracy

Can Poisson Regression for object estimation within a sliding window context improve clustering accuracy, specifically enhancing cluster purity and reducing misclassification rates?

3.4 Clustering Validation through Clusters Kinematic Extracted Feature Classification

How effective is the classification of kinematic features extracted from clusters in validating the clustering results, and to what extent can these features ensure the accuracy and reliability of the identified movement patterns?

Chapter 4

Data

The data analysed in this project consists of high-resolution bird movement recorded radar signals obtained from the Movebank organisation website[18]. This organisation aggregates an extensive global archive of thousands of datasets of animal tracking data. These datasets include precise geographical coordinates, timestamps, and other measurements for each recorded location, providing detailed temporal and spatial tracking of bird movements.

Four datasets were selected for different bird species:

1. Common Kestrel
2. Herring Gulls
3. White Stork
4. Homing Pigeon

Which will be referred to from this point on as datasets (or classes) 1 through 4.

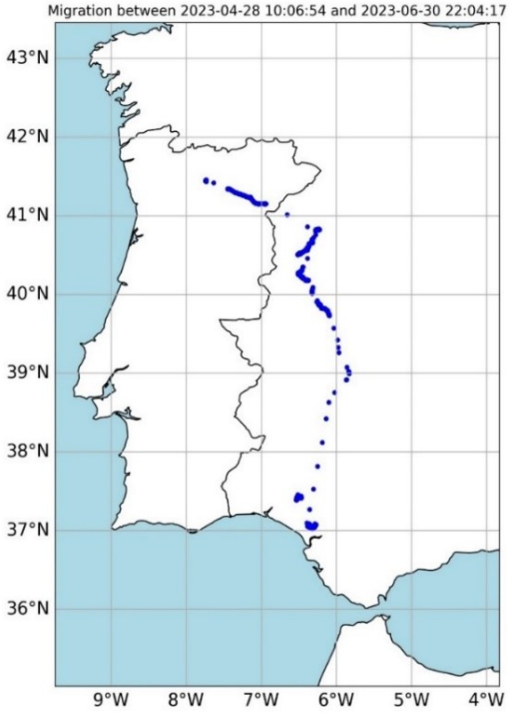
These species were selected due to some diversity in their movement characteristics, which enabled a challenging yet possible classification between them. This ensures clarity in the analysis and avoids the complexity of overlapping species behaviours.

Each dataset initially included several months of tracking. The first five thousand data points in chronological order were selected from each dataset, encapsulating about a month of continuous tracking data. Each data point includes longitude and latitude coordinates, altitude, and timestamp.

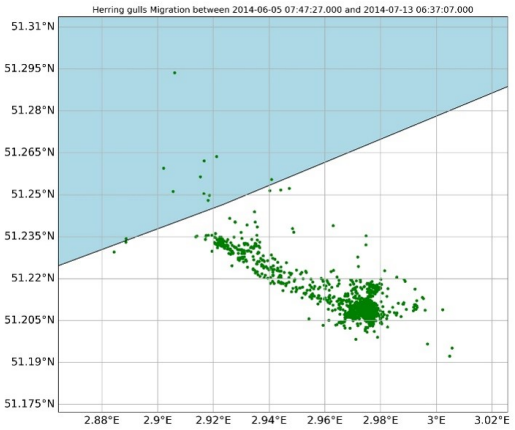
The reasons for this sampling approach were:

1. The datasets were too large to test the algorithm in a feasible time frame.
2. There was no need for the entire recorded data to test the algorithm.

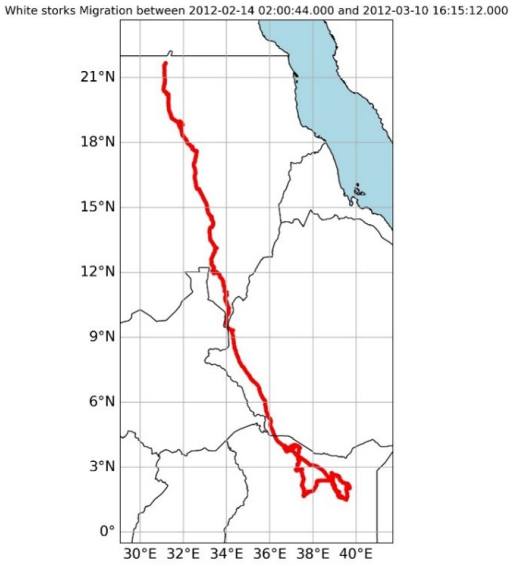
Each data point may represent one or multiple birds and sometimes includes additional sensor data like speed, direction, and environmental variables such as temperature and wind speed or direction. As a result, it is difficult to determine the number of birds in each cluster, as one plot can represent a single bird, two birds, a bird already seen in this cluster, etc. Figure 4.1 plots a sample of the species migration routes from 2023. All species are usually seen alone but sometimes travel in small flocks and nest in loose colonies.



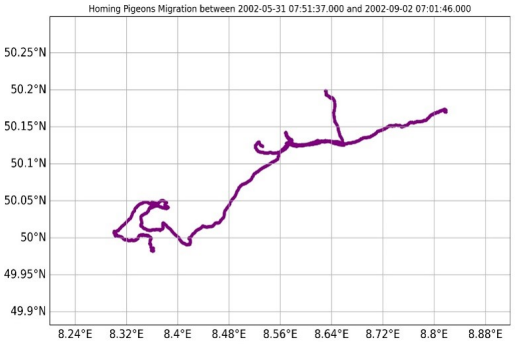
(A) Common Kestrel migration sample on map



(B) Herring Gull migration sample on map



(C) White Stork migration sample on map



(D) Homing migration sample on map

FIGURE 4.1: Four species migration samples on map

The histograms below illustrate the dataset distributions. For instance, in Figure 4.2, Dataset 1 shows an even timestamp distribution, indicating continuous data collection. The longitude is centred around -6.4, while the latitude is concentrated near 37.7, both skewed, suggesting specific geographic coverage. Heights above mean sea level range widely, with an average of 219 meters, showing significant altitude variation. The other datasets display similar but distinct patterns, as shown in Figures 4.3-4.5, with varying distributions across the variables.

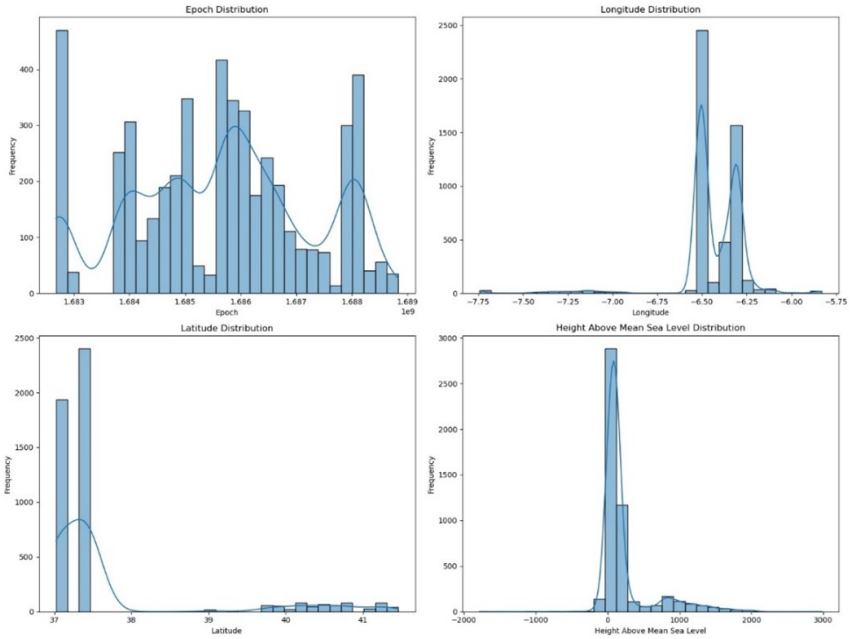


FIGURE 4.2: first Dataset columns distributions

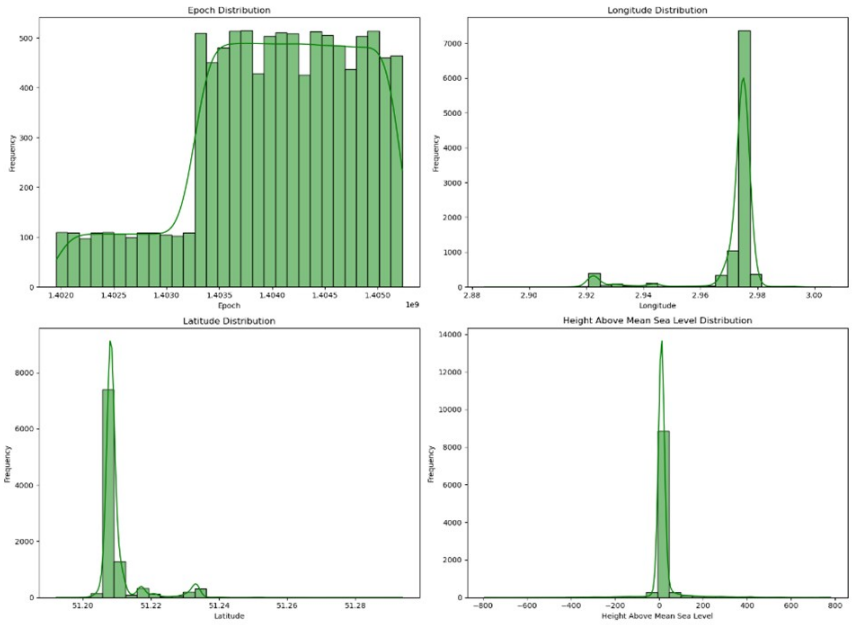


FIGURE 4.3: Second Dataset columns distributions

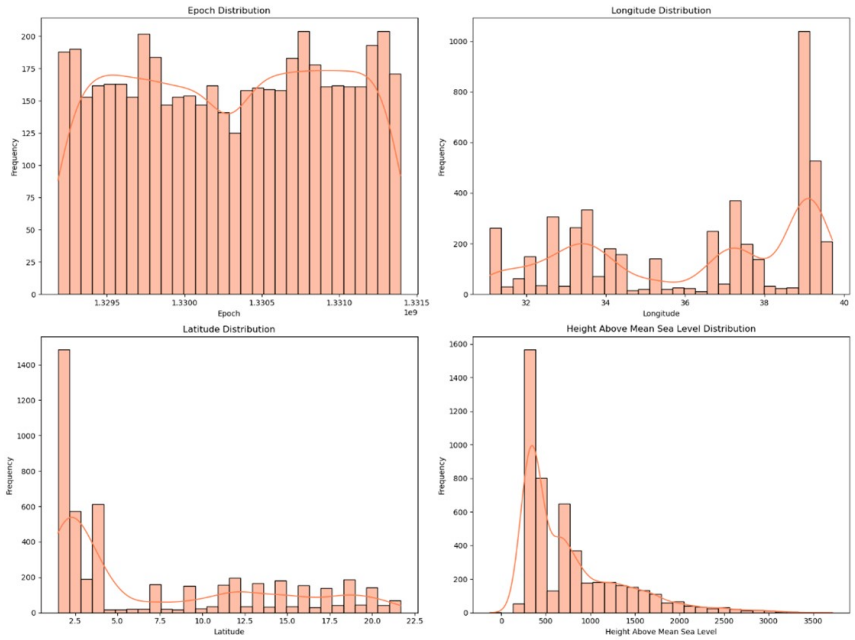


FIGURE 4.4: Third Dataset columns distributions

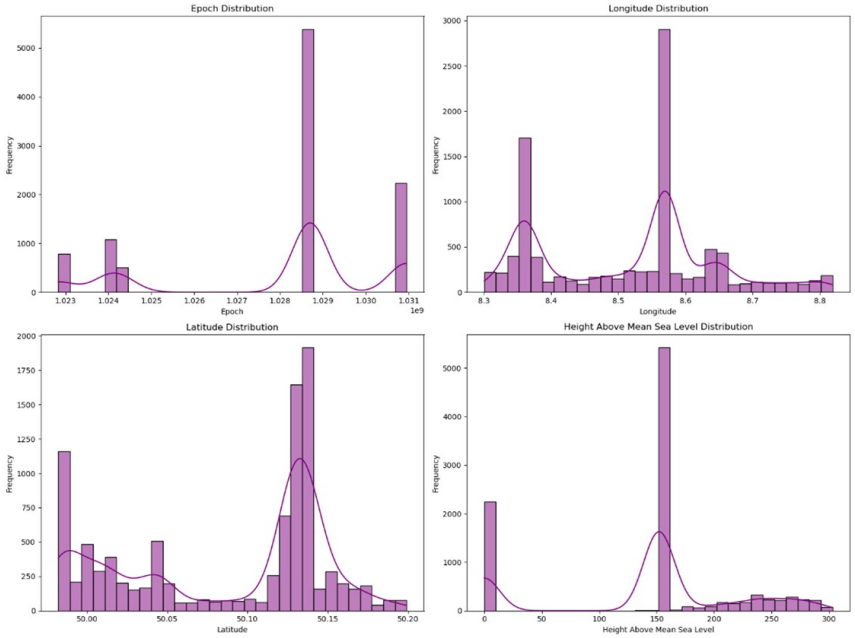


FIGURE 4.5: forth Dataset columns distributions

Chapter 5

Methodology and Implementation

5.1 Overview

The epsilon (EPS) value in DBSCAN determines the minimum distance required to group points into the same cluster. This project introduces a novel approach to clustering tracking data by dynamically adjusting the EPS value in DBSCAN. The proposed method involves processing the data in chronological subsets, with small overlaps between adjacent subsets to ensure continuity across windows. The subsets are called windows, and their size was selected empirically using a custom elbow method. Within each window, the EPS value is adjusted until optimal separation of clusters is achieved. To evaluate the performance of this method, the algorithm was applied to four datasets of four different bird species' tracking data, each in three distinct settings:

1. With dynamic EPS and overlap window.
2. With static EPS and overlap window.
3. With static EPS and no overlap window (i.e., traditional DBSCAN across windows)

The results demonstrate that the new dynamic method forms clusters that more accurately capture the flight patterns of the species.

The following steps were undertaken to validate the method:

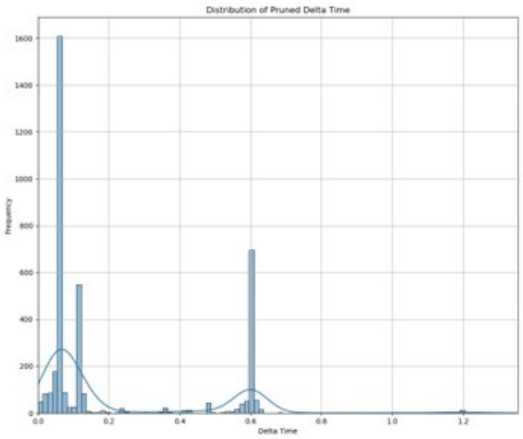
- **Clustering quality metrics -** Each setting was evaluated using established metrics such as the Silhouette Score, Davies-Bouldin Index (DBI), and Calinski-Harabasz Index (CHI). The dynamic EPS method consistently achieved superior scores, indicating better cluster quality. In addition, the ability to link clusters together was tested in each setting.
- **Cluster Quantification -** A Poisson-based method was introduced to estimate the number of objects within clusters to reduce deviation in object counts. Including an overlap window and dynamic EPS improved the consistency of object count estimations across all datasets.
- **Kinematic feature analysis -** For each setting, the clusters' kinematic features (e.g., speed, acceleration, trajectory curvature) were calculated and compared against the species' known flight patterns (based on existing literature). Clusters formed using the new method matched the expected range of physical features more accurately than those from the traditional DBSCAN settings. Unlike the original data, kinematic features are detached from location and time, which enables combining the four results in each setting into a single classification model.

- **Random Forest classification -** After calculating kinematic features across all four datasets, a Random Forest classification model was used to test the distinctiveness of the clusters produced in each setting to validate clustering accuracy. Three models were created corresponding to the three settings. The model based on the dynamic EPS method achieved the highest accuracy, demonstrating that the clusters formed using this approach were more representative of species-specific flight patterns than the other settings.

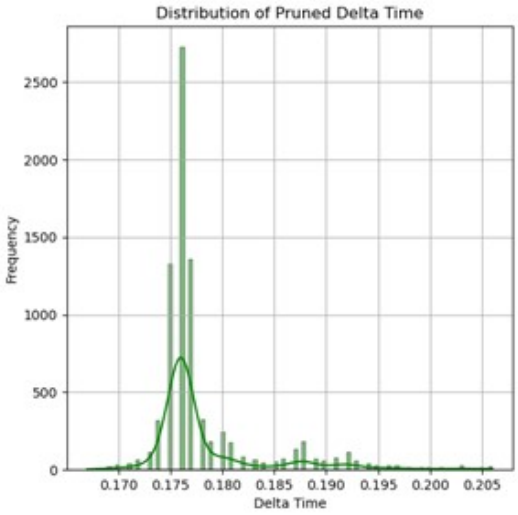
5.2 Data exploration and setting baseline parameters values

The data was cleaned by removing irrelevant columns. Timestamps were standardised to milliseconds from the first recorded entry to simplify calculations, and the chronological order of the data was verified. Following preprocessing, exploratory data analysis (EDA) was conducted using various statistical techniques to identify patterns and detect outliers within the single-species dataset. For example, the delta between timestamps and distance distribution was plotted. All analyses were performed in Python version 3.7, using the Jupyter Notebook interface. The code can be found at the provided GitHub repository [5].

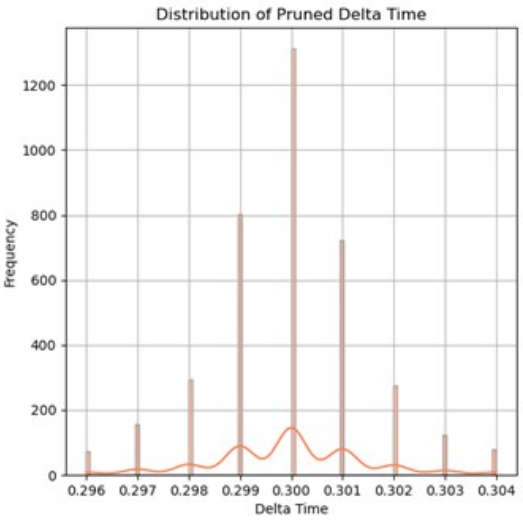
Figure 5.1 shows the distribution of time differences between every two consecutive time records, exhibiting the biggest time gaps in the data.



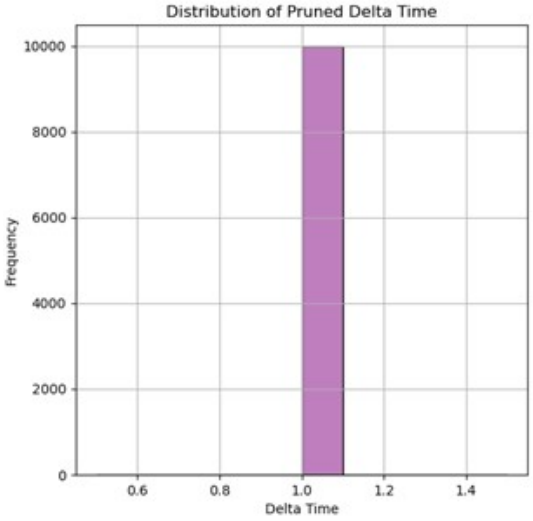
(A) First dataset Time gaps distribution



(B) Second dataset Time gaps distribution



(C) Third dataset Time gaps distribution



(D) Forth dataset Time gaps distribution

FIGURE 5.1: Four species Time gaps distribution

5.3 Sliding Window Feature

The algorithm processed the datasets using equal-sized windows with an overlap smaller than the window size. This approach enabled the window to capture data segments, advancing by a single step size in each iteration. The next chapter describes empirically determining the window size, beginning with a default epsilon value.The code performing this algorithm is displayed in Appendix F.

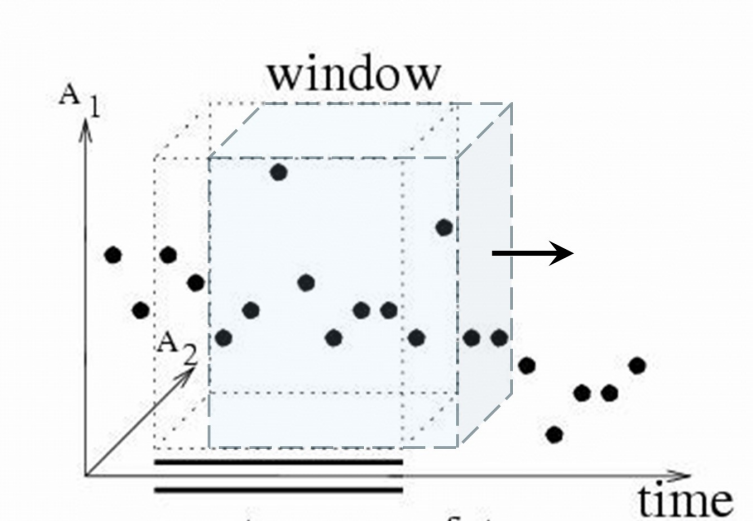


FIGURE 5.2: Sliding Window illustration [9]

The sliding window approach illustrated in Figure 5.2 was strategically designed to identify distinct routes or objects within each window and to link clusters from consecutive windows that shared common points within the same trajectory. This connectivity allowed clusters from adjacent windows, which likely had similar density and statistical characteristics, to be seamlessly connected. This method effectively addressed the challenge of assessing kinematic attributes across different data segments.

In temporal data analysis, selecting appropriate window and overlap sizes is critical for accurate segmentation. Overlap windows allow the formation of sibling clusters, creating continuous trajectories, while clusters without common points across windows become orphan clusters. The aim was to balance window and overlap sizes to minimize orphan clusters and maintain data continuity.

The elbow method was adapted to determine where reducing window and overlap sizes no longer decreased the number of empty overlap windows. Initial reductions in size improved coverage, but further reductions provided diminishing returns. The optimal balance was found using this method.

Window and overlap sizes were selected empirically based on domain knowledge and practical considerations, ensuring data continuity without excessively small or large windows. This minimized orphan clusters while maintaining computational efficiency.

Testing began with a broad range of window sizes (10 to 150 minutes) and overlaps (2 to 40 minutes). A refined set of smaller windows (1 to 60 minutes) and overlaps (0.2 to 6 minutes) was applied to the fourth dataset, which had smaller temporal

and spatial gaps, enhancing sensitivity to frequent data changes. Figure 5.3 demonstrates the results of the elbow method across the datasets.

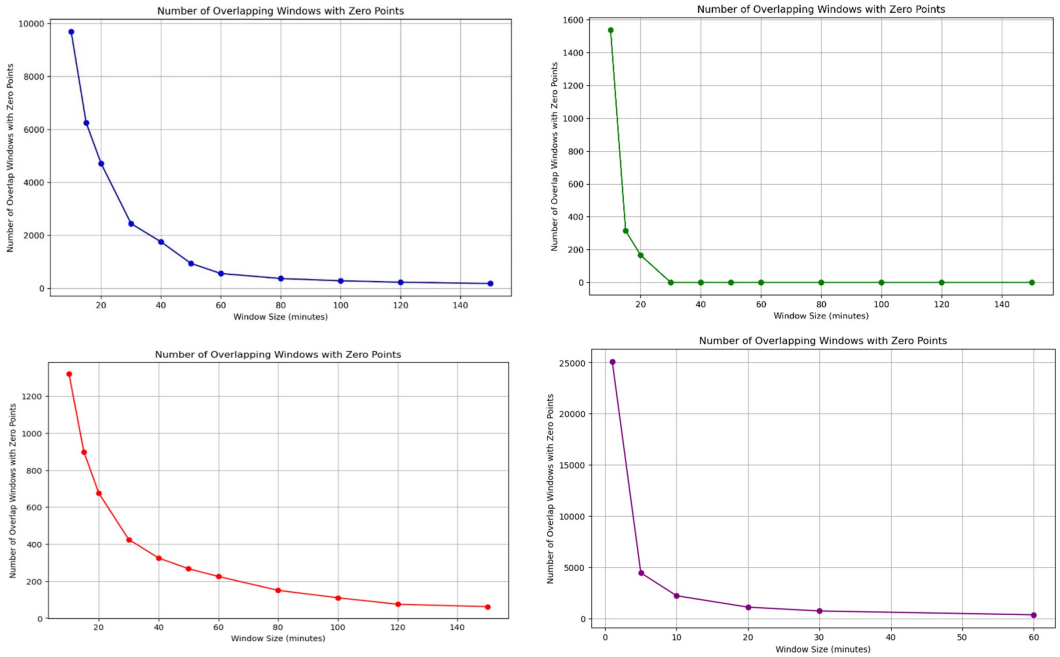


FIGURE 5.3: Elbow Plot for all four datasets for relevant window and overlap sizes. First dataset (upper left), second dataset (upper right), third dataset (bottom left), and fourth dataset (bottom right).

The number of data points was plotted for each 60-minute window, with an overlap of 20 minutes, revealing several windows without data points. Figure 5.4 presents a line chart depicting the data point count per window and the number of common points overlapping with the previous window.

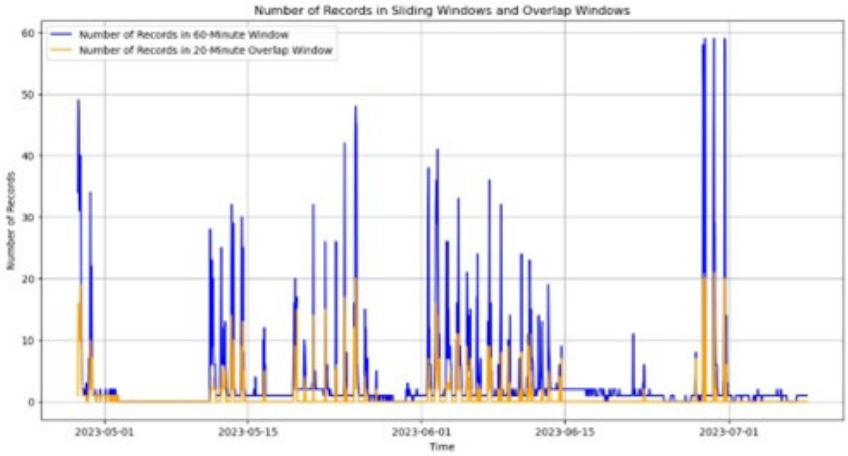


FIGURE 5.4: Window and overlap data points count

As illustrated in Figure 5.5, a 60-minute window size with a 20-minute overlap resulted in a relatively optimal number of empty overlap windows. The subsequent plots in 5.5 apply the same methodology to the other three datasets, displaying only the first 50 windows to enhance chart readability.

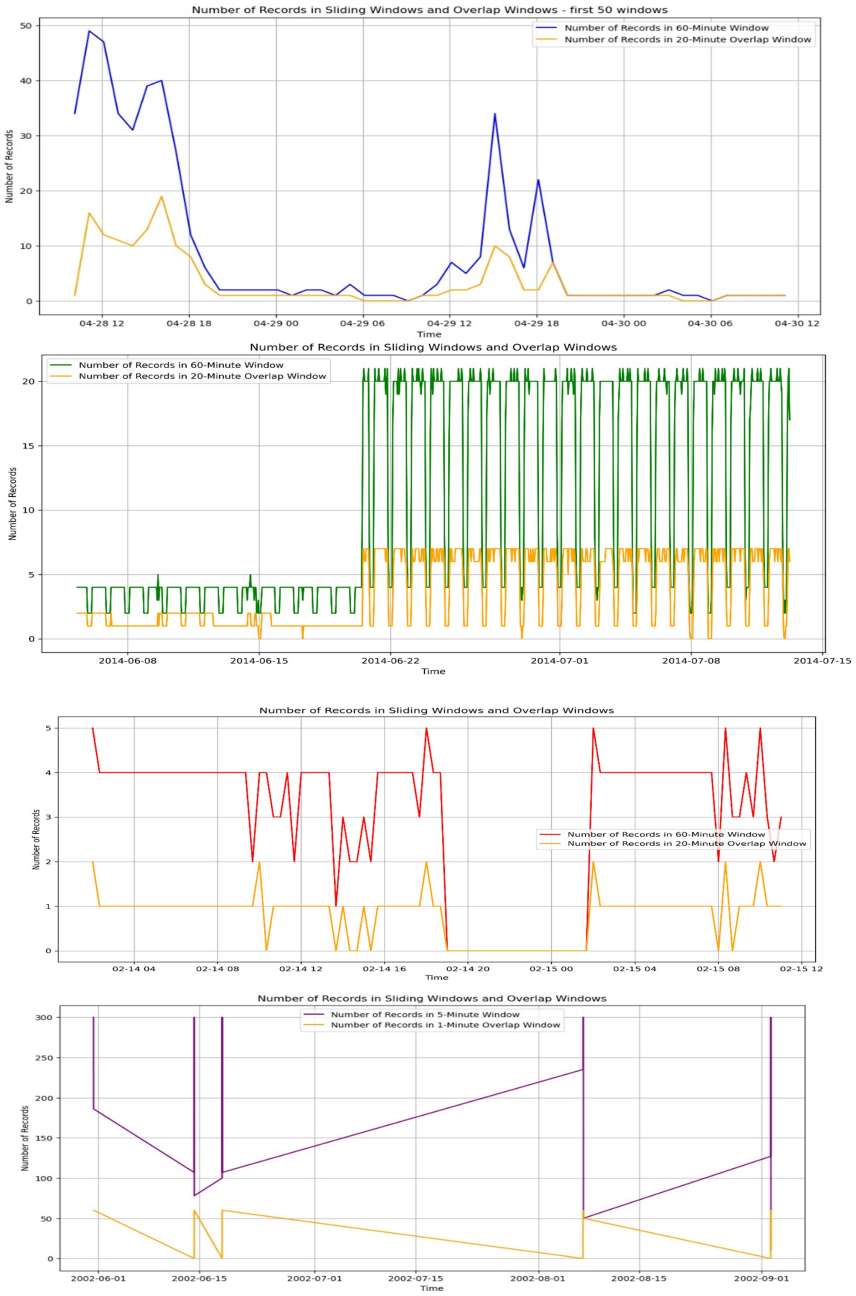


FIGURE 5.5: Number of records in sliding window and overlap window

Unlike traditional static clustering, which treats all data as a single batch or subdivides it into a grid, this method enabled the analysis of temporal data in manageable segments, maintaining temporal relevance by focusing on specific time ranges. [21].

The sliding window method was chosen for its ability to maintain temporal continuity, detect local patterns, and adapt to varying data densities. Unlike static or batch processing methods, it preserved the sequential nature of events and minimised information loss at segment boundaries through overlapping windows. The full sliding window results for the first algorithm setting are found in Appendix A.

5.4 Baseline Epsilon Selection

Selecting an appropriate epsilon (EPS) is crucial for DBSCAN's effectiveness. The Haversine distance accounts for Earth's curvature and is ideal for calculating distances between coordinates, making it well-suited for analyzing bird movements. In this study, EPS is dynamically adjusted rather than fixed. The median Haversine distance within each window is used as a data-driven baseline for EPS, offering a reliable foundation for clustering. The baseline EPS value was the median value observed from the data, which served as the selected EPS value in the static settings. In the dynamic setting, this baseline is iteratively fine-tuned during runtime to ensure optimal cluster separation. Adjusting EPS dynamically prevents clusters from merging or being misclassified as noise.

5.4.1 EPS Selection Results

Charts in Figure 5.6 plot the median Haversine distance across sliding windows (and overlap windows) for the four datasets.

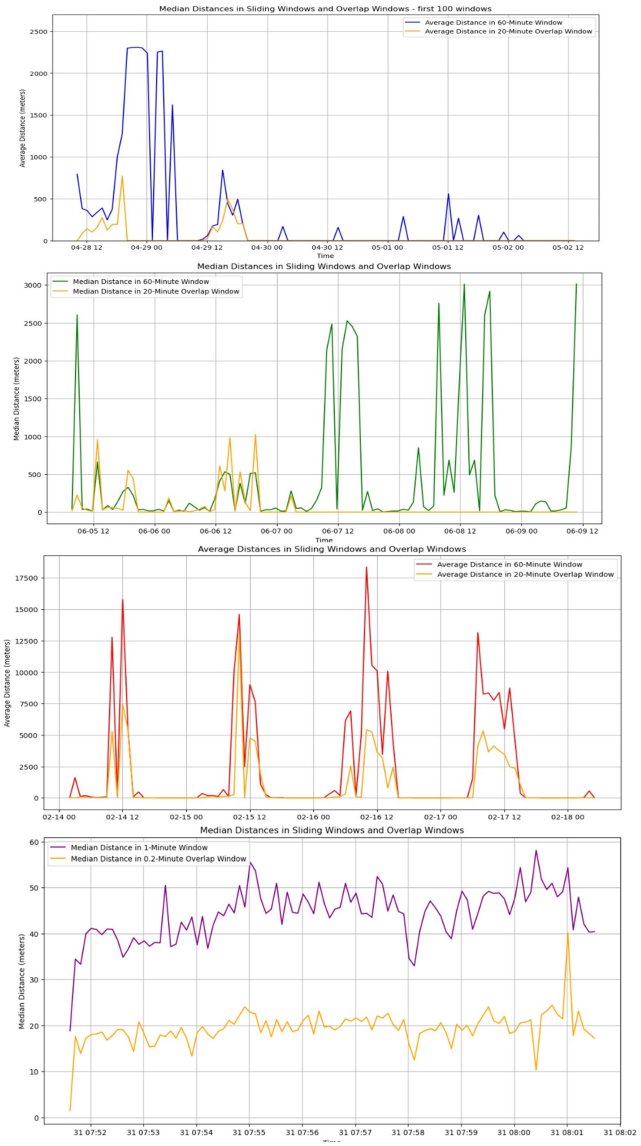


FIGURE 5.6: Number of records in sliding window and overlap window (for optimized window and overlap sizes)

There was considerable variance in the distances within the windows of each dataset. The first three datasets exhibited distances ranging from hundreds to thousands of meters, whereas the fourth dataset contained distances primarily from dozens to hundreds of meters.

Additionally, gaps were evident in all four datasets, indicating periods when no data points were detected. These gaps occurred abruptly, as seen in Datasets 1 and 4, or in repeating patterns, as observed in Datasets 2 and 3.

5.5 Dynamic Epsilon Feature

The methodology involved calculating the local density of data points and adjusting the epsilon parameter accordingly. Each window was initially assigned a default epsilon value, set to the median distance determined during the exploratory data analysis (EDA) phase. Following this, the clusters formed within each window were re-evaluated, and the epsilon value was fine-tuned as necessary to achieve an

optimal clustering state.

By implementing a dynamic epsilon parameter, the clustering algorithm was able to adapt to local density variations within the dataset. An optimal state was characterized by a high intra-cluster density significantly exceeding the minimal inter-cluster density across all clusters. Once this steady state was achieved, clusters were matched with corresponding clusters from the previous time window based on shared points. Clusters with mutual points across windows were identified as "sibling clusters". Since each data window contained a relatively small number of points, dynamically adjusting this parameter proved more effective and efficient. Figure 5.7 illustrates the distances considered to evaluate optimal clustering.

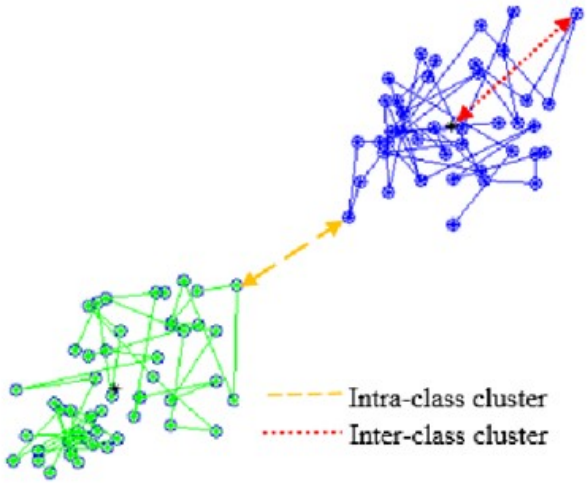


FIGURE 5.7: Intra-cluster and Inter-cluster density

This approach also eliminated the need to account for time differences in the clustering process, as various flying objects may have different velocities influenced by factors such as wind, radar reception quality, and other high-entropy environmental conditions. For each cluster, the window EPS, maximum distance in the cluster, distance to the closest centroid and quality metrics were saved to the data frame.

To ensure clustering quality, the evaluation of the results relied on three selected metrics. Since the DBSCAN algorithm is nonparametric, the Sum of Squared Errors (SSE) is not typically used to measure cluster quality in DBSCAN. SSE is more commonly associated with algorithms like K-Means, which quantify the total squared distance between each data point and the centroid of its assigned cluster.

In contrast, DBSCAN does not rely on centroids or assume a specific cluster shape, making SSE less applicable for evaluating cluster quality in this context.

the selected alternative metrics employed to assess clustering quality were:

5.5.1 Silhouette Score

Measures how similar an object is to its cluster compared to others. A higher silhouette score indicates better-defined clusters.

The Silhouette Score measures how similar an object is to its cluster (cohesion) compared to other clusters (separation)[27]. The formula for the Silhouette Score for a single data point is given by:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i)} \tag{5.1}$$

Given a dataset with n samples and k clusters, let:

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

$b(i) \rightarrow$ be the average distance between the i-th sample and all points in the nearest cluster (the cluster with the smallest average distance to the point).

The Silhouette score ranges from -1 to 1:

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

For the entire dataset, the mean Silhouette score was calculated by averaging the Silhouette scores of all samples. This provides an overall assessment of the clustering quality.

5.5.2 Davies-Bouldin Index

The average 'similarity' ratio between each cluster and its most similar one. Lower values indicate that clusters are more distinct.

The Davies-Bouldin Index (DBI) is used to evaluate the clustering quality[26]. It is defined as the average similarity ratio of each cluster with its most similar cluster, where a lower DBI indicates better clustering[19]. The formula for the Davies-Bouldin Index is as follows:

1. Compute the cluster centroid C_i for each cluster i .

2. Calculate the average distance S_i between each point in cluster i and the centroid C_i:
$$S_i = \frac{1}{|C_i|} \sum_{x \in C_i} \|x - \mu_i\| \tag{5.2}$$

3. Compute the distance d_{ij} between the centroids of clusters i and j :
$$d_i j = ||C_i - C_j \,|| \tag{5.3}$$

4. For each cluster i , find the maximum ratio R_{ij} of $S_i + S_j$ to $d(ij)$ for all j not equal i :
$$S_i = \frac{S_i + S_j}{d(ij)} \tag{5.4}$$

5. The Davies-Bouldin Index DB is the average of the maximum R_{ij} values for all clusters i :
$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij} \tag{5.5}$$

   Where k is the number of clusters.

   A lower Davies-Bouldin Index indicates better clustering quality, as the clusters are compact and well-separated.

5.5.3 Calinski-Harabasz Index

Also known as the Variance Ratio Criterion, this index measures the dispersion between and within clusters, with higher values indicating better clustering.

The Calinski-Harabasz Index (CH Index) evaluates the quality of clustering by considering the ratio of the sum of between-cluster dispersion to within-cluster dispersion[25]. A higher CH Index indicates better-defined clusters[11].

Given a dataset with n samples, divided into k clusters, let:

- X_i be the i-th cluster, and n_i be the number of samples in X_i .
- C_i be the centroid of the cluster X_i.
- C be the centroid of the entire dataset.

The formula for the Calinski-Harabasz Index is:

$$CH = \frac{\text{between cluster dispersion}}{\text{within cluster dispersion}} * \frac{n-x}{k-1} \tag{5.6}$$

Full formula:

$$CH = \frac{\sum_{i=1}^{k} n_i \left|\left|C_i - C\right|\right|^2}{\frac{1}{k-1}\sum_{i=1}^{k}\sum_{x \in Xi}\left|\left|x - C_i\right|\right|^2} \times \frac{n-x}{k-1} \tag{5.7}$$

Where:

- The numerator is the between-cluster dispersion, calculated as the sum of the squared distances between the cluster centroids and the overall centroid, weighted by the cluster sizes.
- The denominator is the within-cluster dispersion, calculated as the sum of the squared distances between each point in a cluster and the cluster centroid.
- n is the total number of samples, and k is the number of clusters.

The higher the Calinski-Harabasz Index, the better defined the clusters are.

5.5.4 Clustering quality results of 4 species

Figure 5.8 is an example of the clustering metrics calculated for a single dataset across Windows.

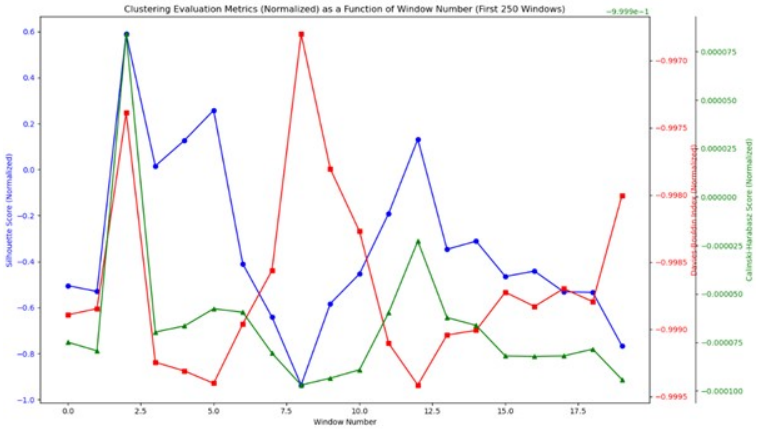


FIGURE 5.8: Example of the three metrics plotted for the first dataset

Figure 5.9 illustrates a boxplot of these values for each dataset using the first algorithm setting (dynamic EPS):

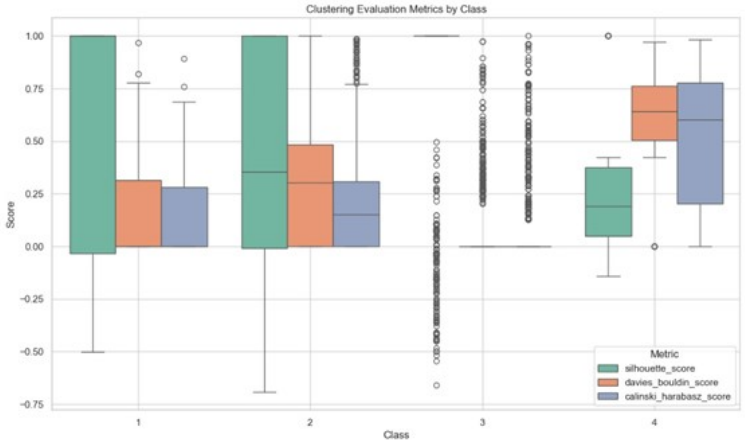


FIGURE 5.9: Boxplot of the three metrics normalized for each dataset of the dynamic setting

Figures 5.10-12 plot the boxplots of minimum and maximum distances between points and maximum distances between centroids in each dataset and setting. The results show that the first setting for all clusters produced more compact clusters with better separation.

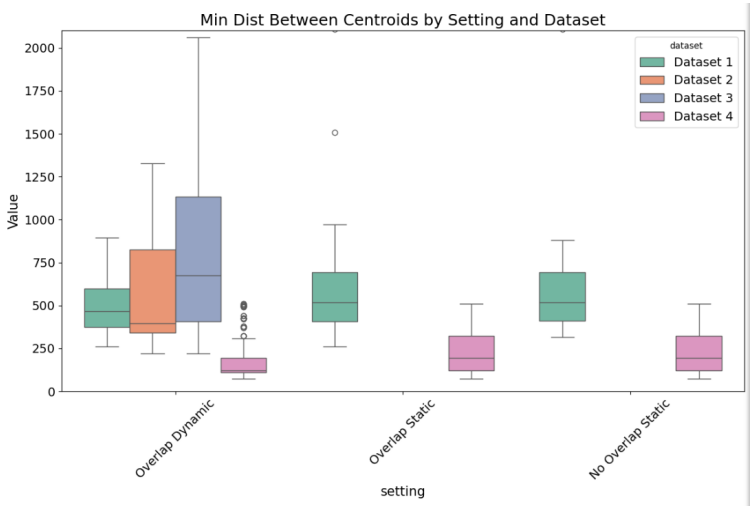


FIGURE 5.10: Minimum distance between centroids

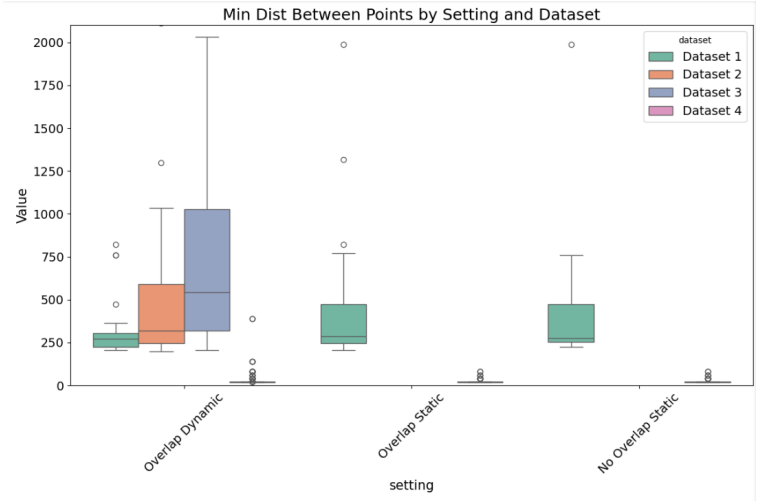


FIGURE 5.11: Minimum distance between points

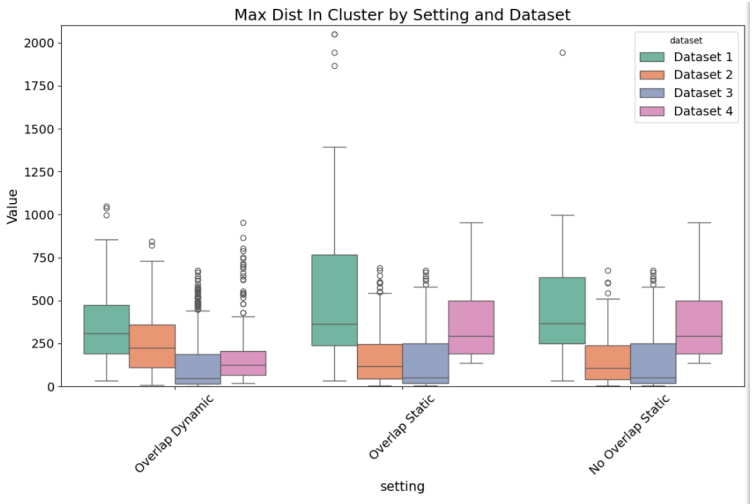


FIGURE 5.12: Maximum distance in clusters

The clustering analysis revealed apparent differences in clustering quality across the various classes in the dataset. The dynamic EPS setting exhibited the highest clustering quality, with high silhouette and Calinski-Harabasz scores and low Davies-Bouldin scores, indicating that the clusters were well-defined, compact, and well-separated. Static Settings with an overlap window showed moderate clustering quality, with reasonably defined clusters, though there was some overlap between clusters and less compactness.

5.5.5 Sibling Clusters

As shown in the following figures, most window clusters across all four datasets had a corresponding sibling cluster in the previous window. However, all datasets needed more continuity, with some windows either being empty or lacking sibling clusters. Figure 5.13 displays the percentage of sibling clusters between consecutive windows from all formed clusters (in each setting and dataset). Figures of sibling clusters over windows for the dynamic setting are displayed in Appendix E.

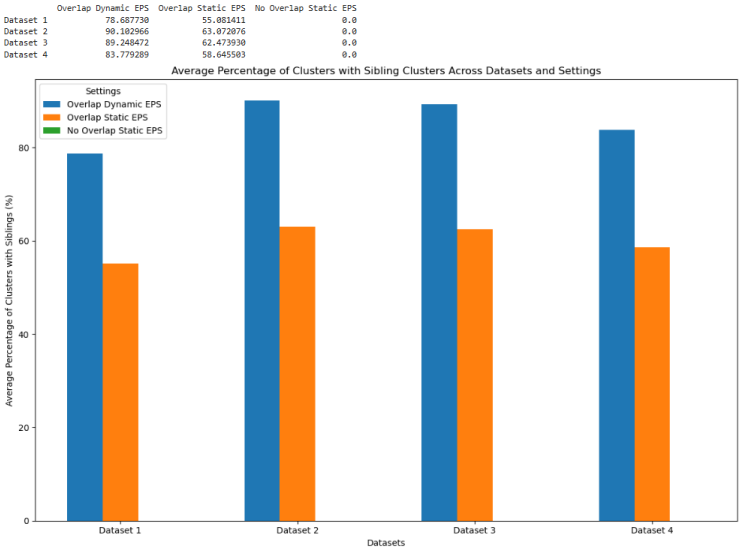


FIGURE 5.13: Clusters with siblings percentage

Figure 5.14 displays the accumulation of orphan clusters across all datasets and settings. The datasets with the first algorithm setting exhibited fewer orphan clusters.

	Overlap Dynamic EPS	Overlap Static EPS	No Overlap Static EPS
Dataset 1	1253.25	1754.55	1838.1
Dataset 2	3136.50	4391.10	4600.2
Dataset 3	770.25	1078.35	1129.7
Dataset 4	161.25	225.75	236.5

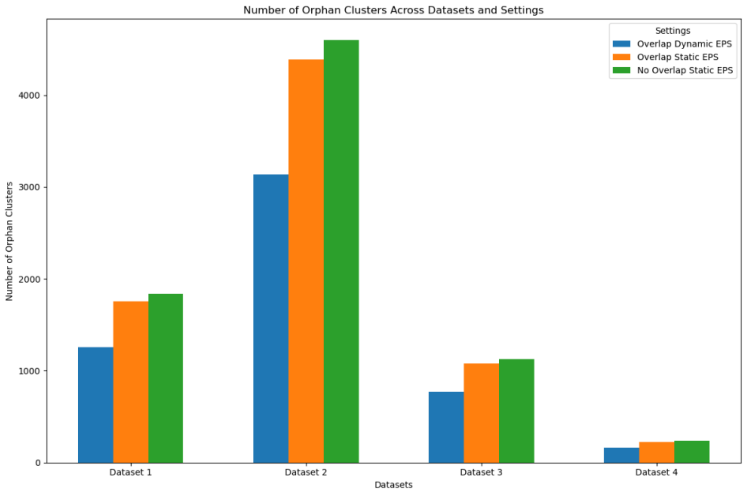


FIGURE 5.14: Summary of orphan clusters

5.5.6 Kinematics Validation Method

The validation of species clustering in this project was conducted by calculating and analysing kinematic features such as average speed, angular velocity, trajectory curvature, and other related metrics. These features were compared against known kinematic characteristics of the species as reported in the literature.

This approach ensured that the clustering algorithm did not merely group data points based on spatial-temporal proximity but also accurately reflected the actual flight dynamics of the species under study.

The following kinematic parameters were selected to evaluate each cluster derived from the algorithm analysis[23].

Speed (Velocity Magnitude)

- **Definition:** The rate of change of position to time.
- **Calculation:**

$$\text{Speed} = \frac{\text{Distance between two points}}{\text{Time Interval between two points}} \tag{5.8}$$

- **Units:** meters per second (m/s).

Velocity (Vector Quantity)

- **Definition:** The speed of an object in a specific direction.
- **Calculation:**

$$\text{Velocity} = \frac{\text{Change In Position (Vector)}}{\text{Time Interval}} \tag{5.9}$$

- **Units:** meters per second (m/s) in a specific direction.

Acceleration

- **Definition:** The rate of velocity change to time.
- **Calculation:**

$$\text{Acceleration} = \frac{\text{Change in Velocity}}{\text{Time Interval}} \tag{5.10}$$

- **Units:** meters per second squared (m/s^2).

Jerk

- **Definition:** The rate of change of acceleration to time.
- **Calculation:**

$$\text{Jerk} = \frac{\text{Change in Acceleration}}{\text{Time Interval}} \tag{5.11}$$

- **Units:** meters per second cubed (m/s^3).

Displacement

- **Definition:** The change in position of an object, considering direction (a vector quantity).
- **Calculation:**

$$\text{Displacement} = \text{Final Position} - \text{Initial Position} \tag{5.12}$$

- **Units:** meters (m).

Distance Travelled

- **Definition:** The total length of the path travelled by the object, irrespective of direction.
- **Calculation:** Sum of distances between consecutive points along the trajectory.
- **Units:** meters (m).

Angular Velocity

- **Definition:** The rate of change of angular position to time.
- **Calculation:**

$$\text{Angular Velocity} = \frac{\text{Change In Angle}}{\text{Time Interval}} \tag{5.13}$$

- **Units:** radians per second ($\frac{rad}{s}$)

Trajectory Curvature

- **Definition:** The degree to which a trajectory deviates from being a straight line.
- **Calculation:** Involves computing the rate of change of the direction of the velocity vector along the trajectory.
- **Units:** ($\frac{1}{m}$).

Bearing

- **Definition:** The direction or angle between a reference direction (usually North) and the line connecting two points.
- **Calculation:**

$$\text{Bearing} = \text{atan2}(\ \frac{\text{Change in y coordinates}}{\text{Change in x coordinates}}\ ) \tag{5.14}$$

- **Units:** degrees () or radians (rad)

Turn Rate

- **Definition:** The rate at which an object changes its direction of motion.
- **Calculation:**

$$\text{Turn Rate} = \frac{\text{Change in Bearing}}{\text{Time Interval}} \tag{5.15}$$

- **Units:** degrees per second ($\frac{}{s}$) or radians per second ($\frac{rad}{s}$).

Angle to Horizon (Pitch Angle)

- **Definition:** The angle between the horizontal plane (horizon) and the object's trajectory.
- **Calculation:** Compute Kendall's tau correlation coefficient between time and altitude.
- **Interpretation:**
  - Kendall's Tau = +1: Perfect upward movement (altitude increases with time).
  - Kendall's Tau = -1: Perfect downward movement (altitude decreases with time).
  - Kendall's Tau = 0: No correlation between time and altitude changes.
- **Units:** Dimensionless (range: -1 to +1).

Vertical Acceleration:

- **Definition:** The rate of change of vertical velocity (altitude change) to time.
- **Calculation:**

$$\text{Vertical Acceleration} = \frac{\text{Change in Vertical Velocity}}{\text{Time Interval}} \tag{5.16}$$

- **Units:** meters per second squared (m/s^2)

Table 5.1 displays the valid ranges of each feature per species[6]. Further details about species features can be found in Appendix B.

TABLE 5.1: Comparison of Kinematic Parameters Across Bird Species

Parameter	**Common Kestrel**	**Herring Gull**	**White Stork**	**Homing Pigeon**
Avg Height (m)	100 - 500	50 - 300	100 - 1000	50 - 200
Turn Rate (rad/s)	0.01 - 0.2	0.01 - 0.15	0.01 - 0.12	0.02 - 0.3
Bearing (degrees)	0 - 360	0 - 360	0 - 360	0 - 360
Trajectory Curvature (rad/m)	0.001 - 0.05	0.002 - 0.04	0.001 - 0.03	0.003 - 0.06
Angular Acceleration (rad/s^2)	-0.01 - 0.01	-0.02 - 0.02	-0.01 - 0.01	-0.015 - 0.015
Angular Velocity (rad/s)	-0.1 - 0.1	-0.15 - 0.15	-0.12 - 0.12	-0.2 - 0.2
Distance Traveled (m)	1000 - 10000	800 - 8000	1500 - 15000	500 - 5000
Displacement (m)	500 - 5000	400 - 4000	750 - 7500	250 - 2500
Jerk (m/s^3)	-0.001 - 0.001	-0.002 - 0.002	-0.0015 - 0.0015	-0.002 - 0.002
Vertical Acceleration (m/s^2)	-0.5 - 0.5	-0.4 - 0.4	-0.3 - 0.3	-0.3 - 0.3
Up/Down Direction (Kendall's Tau)	-1 - 1	-1 - 1	-1 - 1	-1 - 1
Angle to Horizon (degrees)	-10 - 10	-15 - 15	-12 - 12	-10 - 10
Average Speed (m/s)	5 - 25	4 - 20	6 - 30	8 - 18
Average Velocity Magnitude (m/s)	5 - 25	4 - 20	6 - 30	8 - 18
Average Velocity Direction (degrees)	0 - 360	0 - 360	0 - 360	0 - 360

5.5.7 Features extraction results

The relevant features were calculated and incorporated into the clusters' data frame for each dataset and setting. Figure 5.15 displays a sample of the feature distributions as box plots, comparing them to the valid ranges established in the literature. Plots of all datasets are present in Appendix C. For most kinematic features, the clusters fell within the typical range of species characteristics. The most influential parameters were average height and turn rate. Average height is the altitude during steady flight, which can vary when birds land or soar. Similarly, the turn rate is highly susceptible to error when insufficient data points exist, which explains the observed deviations in these parameters.

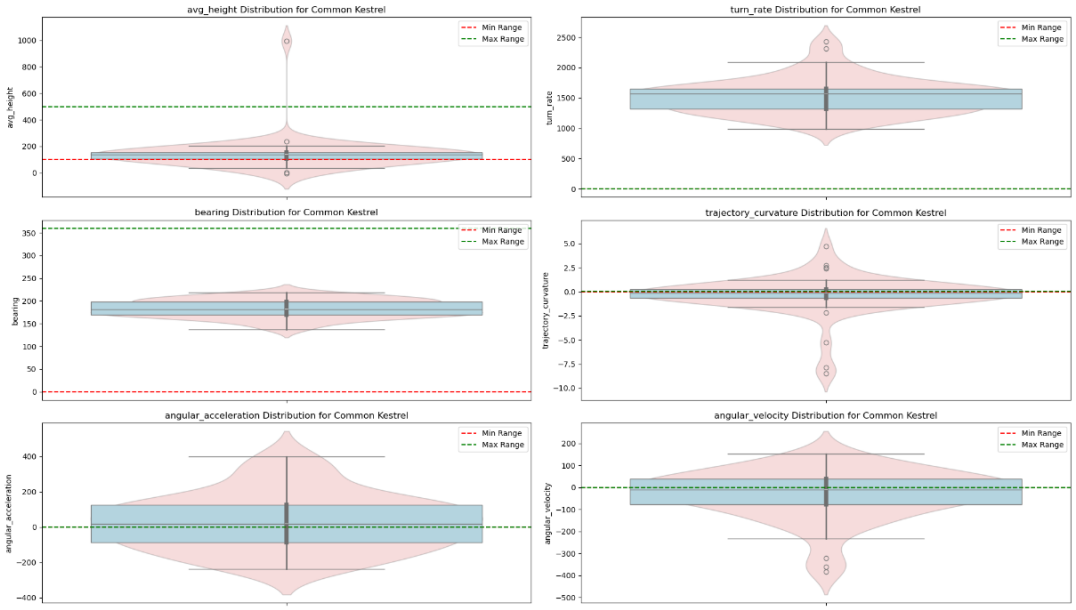


FIGURE 5.15: Sample of Common Kestrel Clusters Kinematics Distribution

Chapter 6

Poisson Regression Feature

Most density estimation in spatial-temporal trajectory analysis often relies on the Kalman filter, a recursive algorithm well-suited for linear systems with Gaussian noise [12]. While effectively tracking moving objects, the Kalman filter's assumptions merge with complex, nonlinear data like ours. Instead, the Poisson model offers greater flexibility, particularly in handling irregular time intervals and varying movement patterns [14]. By leveraging time differences between points and incorporating kinematic features (e.g., velocity, acceleration), the Poisson model provides more accurate density estimation along trajectories, better capturing stochastic behaviours.
We assume that an object moving at constant velocity would be detected regularly. Deviations in time intervals between sibling clusters signal abnormalities in point occurrence. The data in each window was checked against a Poisson distribution to ensure the independence of point occurrences and a constant average rate of points (λ). The predicted and actual λ for each cluster was calculated using Formula 7.1.

$$\lambda = \frac{\sum_{i=1}^{n} X_i}{n} \tag{6.1}$$

Where x_i Is the count of data points in the window i and n Is the total number of windows.
A Poisson model was used to estimate the λ parameter for each window, representing the expected rate of points per unit of time or space. To enhance accuracy, Maximum Likelihood Estimation (MLE) was employed to determine λ. In the context of a Poisson distribution, MLE for λ corresponds to the sample mean of the points count. For clusters of varying sizes, λ was adjusted by normalising it relative to the cluster size.

Poisson regression was also applied to the dataset to model the count of points within each cluster as a function of various kinematic features. Each cluster was treated as an individual observation, with the number of points serving as the response variable. Explanatory variables such as average speed, distance travelled, and other derived kinematic metrics were used to model this count.

Figures 6.1-6.2 display a sample of the lambda results for dataset 1 in setting 1. An evident correlation is exhibited between the actual and predicted lambda values. Further figures of this stage can be seen in Appendix D.

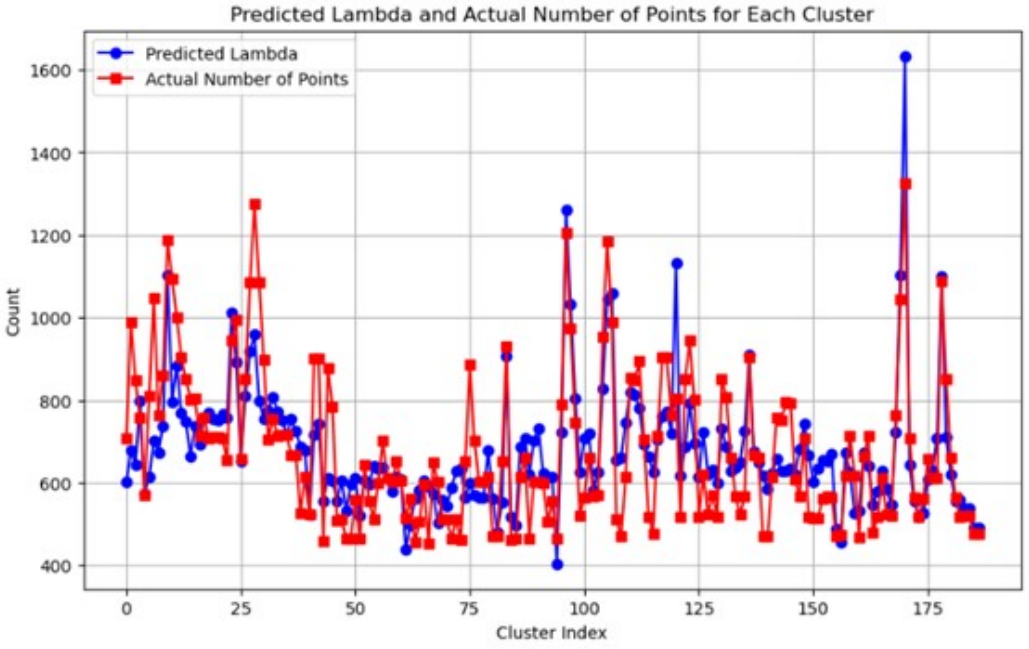


FIGURE 6.1: Predicted and actual Lambda values per cluster for Common Kestrel dataset

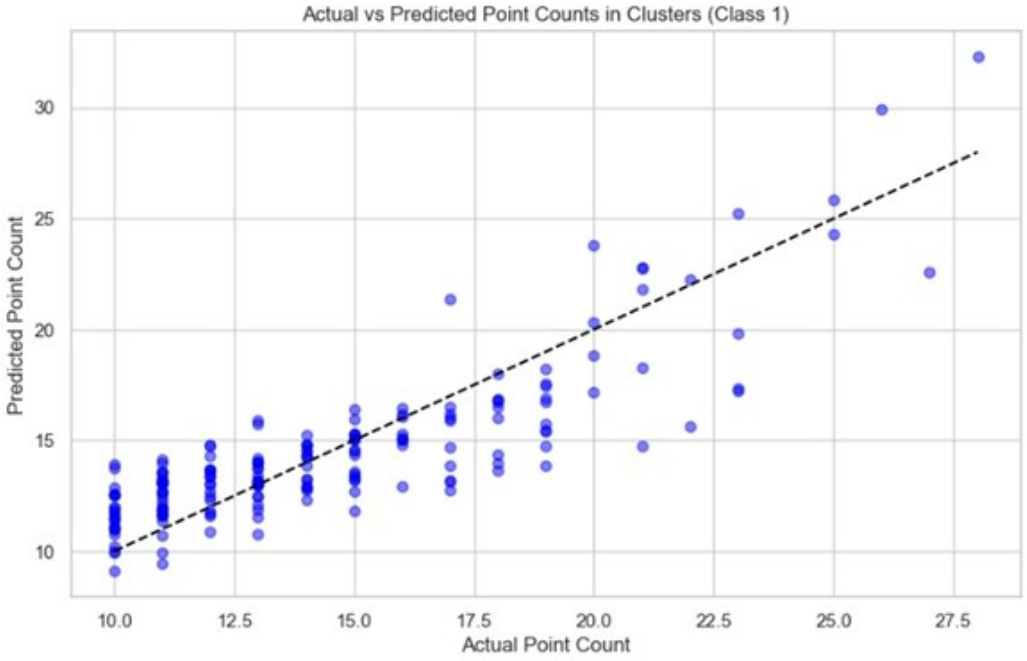


FIGURE 6.2: Actual vs Predicted Cluster Size Correlation for Common Kestrel dataset

6.1 Poisson Regression Results

Figure 6.3 displays the prediction accuracy for every dataset and setting, emphasising that the dynamic setting accuracy was higher than other settings among all datasets.

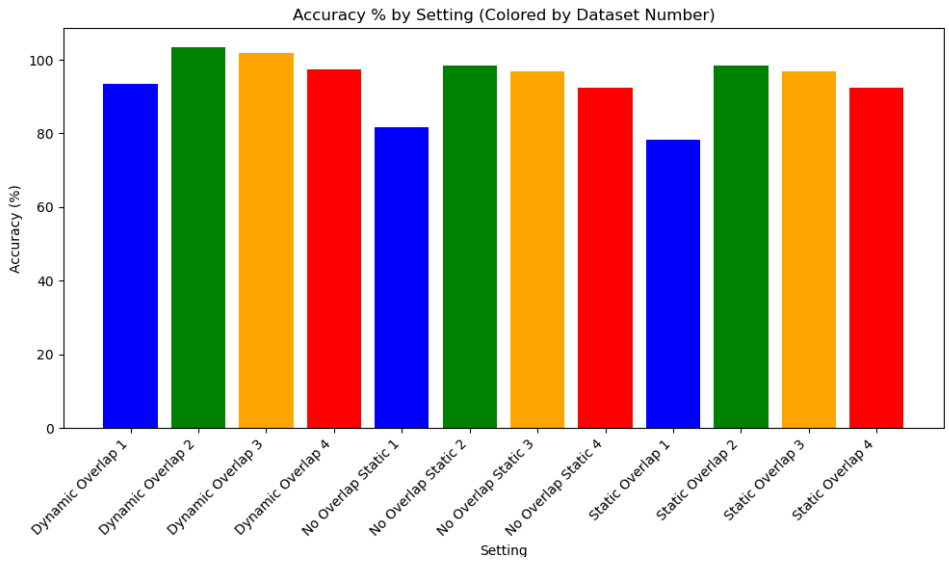


FIGURE 6.3: Poisson regression Accuracy by settings

Figure 6.4 displays a boxplot of the difference between the actual and predicted values for every algorithm setting. Dynamic setting produced the smallest difference across all datasets.

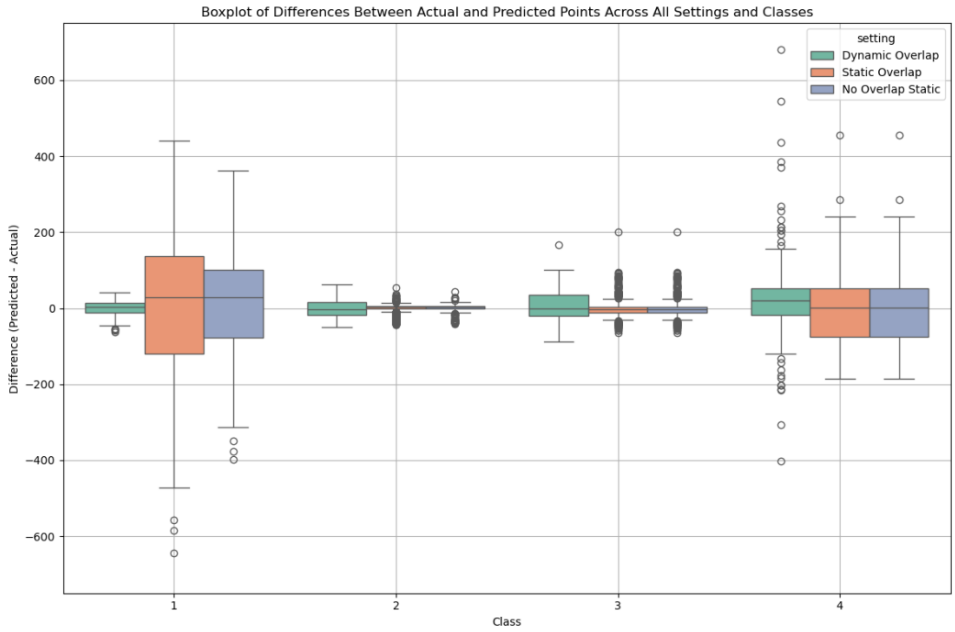


FIGURE 6.4: Actual and predicted delta across settings

Chapter 7

Building the Species Classification Model

In this project, Random Forest was selected as the classification model to differentiate bird species based on their kinematic features. Random Forest was chosen due to its advantage in mitigating the risk of overfitting.

In every Algorithm setting, clusters gathered from each dataset were labelled and combined into a single data frame, with the calculated kinematic features serving as inputs (X) and the species labels as the target variable (Y) to build the classification model for species identification.

The kinematic features calculated on the clusters are detached from the location and time of the original data points. They represent the physical values of the trajectory by relying on the spatial relationships between points in each dimension.

In each dataset and setting, thousands of clusters were formed over the sliding windows. By the Law of Large Numbers, the calculated features more accurately reflect the underlying population patterns as the number of clusters increases.

The model combines the features into a single classification model, independent of bird species, migration paths, or seasonality.

The datasets for each model were (4 datasets):

1. Resulting clusters kinematics from dynamic epsilon with overlap window.
2. Resulting clusters kinematics from static epsilon with overlap window.
3. Resulting clusters kinematics from static epsilon without overlap window.

The results of each model were compared using the Kruskal-Wallis test and Mann-Whitney U test. The Kruskal-Wallis test and Mann-Whitney U test were employed to compare the performance of each model because these are non-parametric tests that do not assume a normal distribution of the data.

The Kruskal-Wallis test was used when comparing more than two groups (in this case, the three different settings) to assess whether statistically significant differences exist between their distributions.

Meanwhile, the Mann-Whitney U test allowed pairwise comparisons between two specific settings (e.g., Dynamic vs. Static) to determine whether one model significantly outperforms the other. These tests are particularly suitable when the datasets vary in size or do not meet the assumptions of parametric tests like ANOVA.

Chapter 8

Ethical Considerations

In utilising datasets from MoveBank for our project, we prioritise ensuring data privacy and confidentiality. The website has thousands of publicly available datasets tracking various bird species. This is especially pertinent when handling sensitive location data of endangered species, as inappropriate disclosure could lead to adverse outcomes such as poaching or habitat destruction. Our research adhered to ethical guidelines, ensured humane treatment of wildlife and obtained informed consent for any new data collection involving animals or humans. We addressed potential bias by carefully selecting diverse datasets covering a range of bird species and environments, ensuring the results were representative. All datasets were used in compliance with GDPR and relevant data usage agreements. Sensitive data, particularly involving endangered species, was anonymized to protect against risks like poaching.

In addition to the risks related to the disclosure of sensitive location data, other ethical considerations in this project included respecting the rights and contributions of the original data collectors. Acknowledging the intellectual property of those who collected and provided the data ensures they receive proper recognition in any publications or reports generated from this research. Adhering to data usage agreements and restrictions set by the data providers not only maintains the integrity of the research but also fosters a collaborative and respectful scientific community. The project methodology and code were well-documented, ensuring future researchers can replicate all models and results.

Moreover, our project is committed to ensuring data collection methods do not harm wildlife. This included avoiding invasive tracking devices that could potentially injure or distress the animals being studied. Non-invasive techniques like remote sensing and camera traps are preferred to minimise interference with the species' natural behaviours and habitats.

By incorporating these ethical considerations, we aim to conduct research that advances scientific knowledge while respecting the rights of data contributors and the well-being of the species we study.

Chapter 9

Final Results

This section will address the results of this study from the perspective of the research questions. The following results refer to the three algorithm settings tested on each dataset, followed by a classification model built for each setting.

9.1 Overcoming Missing Sequences with Sliding Window Dynamic DBSCAN

Table 9.1 presents results for clusters with siblings percentage:

TABLE 9.1: Clusters with siblings in adjacent windows Percentage

Dataset	Dynamic EPS with Overlap	Static EPS with No Overlap	Static EPS with Overlap
Dataset 1	78.69%	0.00%	55.08%
Dataset 2	90.10%	0.00%	63.07%
Dataset 3	89.25%	0.00%	62.47%
Dataset 4	83.78%	0.00%	58.65%

Overlap Dynamic EPS consistently outperforms the other settings, detecting the highest percentage of sibling clusters across all datasets. For example, in Dataset 2, it detects 90.10% sibling clusters, indicating that a substantial proportion of clusters share common points across consecutive time windows.

Overlap Static EPS performs moderately, detecting approximately 30% fewer sibling clusters than Dynamic EPS. This reduction in sibling detection is consistent with the expectation that Static EPS is less adaptable to changes in cluster structure over time. The detection rate for sibling clusters ranges from 55.08% in Dataset 1 to 63.07% in Dataset 2.

No Overlap Static EPS consistently reports a 0% sibling cluster detection rate across all datasets. This is an expected outcome, as the design of this configuration does not allow for any overlap between time windows, making it impossible to track clusters across consecutive windows.

TABLE 9.2: Accumulation of orphan clusters

Dataset	Dynamic EPS with Overlap	Static EPS with No Overlap	Static EPS with Overlap
Dataset 1	1253.25	1838.10	1754.55
Dataset 2	3136.50	4600.20	4391.10
Dataset 3	770.25	1129.70	1078.35
Dataset 4	161.25	236.50	225.75

Regarding orphan clusters, Overlap Dynamic EPS again demonstrates superior performance by producing the fewest orphan clusters across all datasets. For instance, in Dataset 4, it results in only 161.25 orphan clusters, underscoring its effectiveness in minimizing cluster isolation.

In contrast, Overlap Static EPS consistently produces approximately 30% more orphan clusters than Dynamic EPS, corresponding to the reduced detection of sibling clusters. In Dataset 2, for example, Overlap Static EPS results in 4391.10 orphan clusters, compared to 3136.50 in Dynamic EPS, highlighting the impact of a more rigid, non-adaptive clustering approach.

As anticipated, No Overlap Static EPS produces the highest number of orphan clusters across all datasets. Without the benefit of overlap between time windows, this configuration yields more fragmented and isolated clusters. The highest orphan count is observed in Dataset 2, with 4600.2 orphan clusters.

9.2 Clustering Quality Results

TABLE 9.3: Mean Quality Metrics Scores per Algorithm Settings

Setting,Class	SI Score	DBI Score	CHI Score
Overlap Dynamic EPS, Dataset 1	-0.604	-0.082	0.033
Overlap Dynamic EPS, Dataset 2	-0.535	0.460	0.238
Overlap Dynamic EPS, Dataset 3	-0.852	-0.625	-0.576
Overlap Dynamic EPS, Dataset 4	-0.306	1.692	1.598
No Overlap Static EPS, Dataset 1	-0.529	0.169	0.367
No Overlap Static EPS, Dataset 2	-0.995	-0.980	-0.991
No Overlap Static EPS, Dataset 3	-0.959	-0.902	-0.903
No Overlap Static EPS, Dataset 4	-0.373	1.482	1.571
Overlap Static EPS, Dataset 1	-0.546	0.149	0.253
Overlap Static EPS, Dataset 2	-0.984	-0.961	-0.966
Overlap Static EPS, Dataset 3	-0.959	-0.902	-0.903
Overlap Static EPS, Dataset 4	-0.373	1.482	1.571

The table displays the mean value of each quality metric per dataset. The mean was calculated after removing outliers. Across all datasets, the No Overlap Static EPS setting demonstrates lower silhouette scores, indicating poorly defined clusters. In Dataset 2, the silhouette score reaches -0.995, reflecting significant overlap or poorly separated clusters.

However, the Overlap Dynamic EPS setting performs moderately better, with higher silhouette scores. Dataset 2 has a score of -0.535, indicating improved cluster separation compared to the No Overlap setting, though still not optimal. The Overlap Static EPS setting closely mirrors the performance of No Overlap Static EPS, with low scores across datasets, indicating minimal improvement in cluster quality when using static overlap.

The No Overlap Static EPS setting exhibits varying Davies-Bouldin scores. For example, Dataset 1 has a score of 0.169, suggesting tight and well-separated clusters. However, cluster compactness is less consistent in other datasets, particularly Dataset 2 (-0.980).

The Overlap Dynamic EPS setting demonstrates better overall compactness and separation, with consistently lower scores, particularly in Dataset 2 (0.460) and Dataset 3 (-0.625). This suggests that Dynamic EPS offers more robust clustering than No Overlap settings. The Overlap Static EPS setting performs similarly to No Overlap Static EPS, indicating that static overlap does not significantly improve cluster compactness.

For the Calinski-Harabasz score, the No Overlap Static EPS setting shows mixed results, with relatively high values in Dataset 4 (1.571) suggesting better cluster dispersion but negative values in others, indicating instability in cluster quality.

The Overlap Dynamic EPS setting demonstrates more consistent and higher Calinski-Harabasz scores, particularly in Dataset 4 (1.598), indicating better cluster distinctiveness. Like the other metrics, the Overlap Static EPS setting shows no significant improvement over the No Overlap Static EPS, with comparable Calinski-Harabasz scores across datasets.

9.3 Impact of Poisson Model on Clustering Accuracy

TABLE 9.4: Lambda Prediction Accuracy (MLE) per Class

Dataset	Actual Points (Avg.)	Predicted Points (Avg.)	Accuracy (%)
Dynamic Overlap 1	679.91	686.71	93.52%
Dynamic Overlap 2	734.07	741.41	99.00%
Dynamic Overlap 3	391.14	395.05	99.1%
Dynamic Overlap 4	540.59	545.99	97.40%
Static Overlap 1	729.03	729.03	78.19%
Static Overlap 2	450.52	450.52	98.37%
Static Overlap 3	520.37	520.37	96.78%
Static Overlap 4	1070.72	1070.72	92.40%
No Overlap Static 1	731.50	731.50	81.58%
No Overlap Static 2	450.98	450.98	98.40%
No Overlap Static 3	520.37	520.37	96.78%
No Overlap Static 4	1070.72	1070.72	92.40%

The results show that the *Dynamic Overlap* setting demonstrated close alignment between actual and predicted points across all datasets, with accuracy percentages ranging from 93.52% to 99%.

However, the *No Overlap Static* and *Static Overlap* settings exhibited excellent alignment between actual and predicted points, indicating possible overfitting. The accuracy percentages for these settings varied more widely, with *No Overlap Static* showing a lower accuracy of 81.58% in Dataset 1, while the *Static Overlap* setting ranged from 78.19% to 98.37%.

Overall, while both *Static* settings achieved perfect alignment, the *Dynamic Overlap* method emerged as the most effective, consistently maintaining high accuracy and adapting better to changes in cluster structure.

This suggests that the *Dynamic Overlap* setting is more suited to modelling dynamic, evolving data. In contrast, the static settings may struggle with flexibility, especially in more complex datasets like Dataset 1.

9.4 Clustering Validation through Clusters Kinematic Extracted Feature Classification

TABLE 9.5: Clusters Features' Matching Percentage

Kinematic Feature	Overlap & Dynamic EPS (%)	Overlap & Static EPS (%)	No Overlap & Static EPS (%)
Avg Height	79.27	40.65	40.45
Turn Rate	0.00	0.00	0.00
Bearing	100.00	100.00	100.00
Trajectory Curvature	14.43	7.40	6.70
Angular Acceleration	0.00	0.00	0.00
Angular Velocity	0.34	0.17	0.17
Distance Traveled	45.40	23.04	23.28
Displacement	35.45	18.18	17.55
Jerk	0.00	0.00	0.00
Vertical Acceleration	10.34	5.30	4.55
Up/Down Direction	100.00	100.00	100.00
Angle to Horizon	100.00	97.22	96.67
Average Speed	18.44	9.37	9.46
Average Velocity Magnitude	18.44	9.37	9.46
Combined Features (All)	**41%**	**34%**	**33%**

In comparing kinematic features across the three settings, the *Dynamic Overlap* method consistently demonstrated higher values in several key metrics. For *average height*, Dynamic Overlap recorded a value of 79.27, significantly higher than Static Overlap (40.65) and No Overlap Static (40.45).

In terms of *trajectory curvature*, Dynamic Overlap showed the highest value at 14.43, compared to 7.40 for Static Overlap and 6.70 for No Overlap Static. Similarly, Dynamic Overlap outperformed the other settings in *angular velocity* (0.34 vs. 0.17 for both Static Overlap and No Overlap Static), as well as *distance travelled* and *displacement*, with values of 45.40 and 35.45, respectively, compared to lower values in the other settings.

Additionally, Dynamic Overlap had the highest *vertical acceleration* (10.34), *average speed* and *velocity magnitude* (18.44), and *overall success percentage* (41.47%), outperforming both Static Overlap and No Overlap Static in nearly all metrics. These results highlight the superior performance of the Dynamic Overlap method in capturing more dynamic and complex movement patterns compared to the static settings.

9.4.1 Classification Accuracy

TABLE 9.6: Random Forest per Setting Results

Setting	Accuracy	Precision	Recall	F1-Score
Overlap Static	92.28%	93%	93%	92%
No Overlap Static	90.24%	90%	90%	89%
Overlap Dynamic	97.56%	98%	98%	97%

Across the three tested settings, the *Overlap Dynamic* setting achieved the highest accuracy at **97.56%**, outperforming *Overlap Static* with **92.85%** and *No Overlap Static* with **90.25%**. The confusion matrices further highlight this distinction. In the *Overlap Dynamic* setting, the model exhibited excellent performance across all classes, with near-perfect precision, recall, and F1-scores, particularly for *Class 3* and *Class 4*, where both recall and precision were consistently **100%**.

In contrast, *No Overlap Static* struggled more with *Class 1* and *Class 4*, where the recall values dropped to **40%** and **46%**, respectively, indicating more difficulty in correctly classifying instances from these classes. Similarly, *Overlap Static* had lower recall for *Class 4* at **39%**, while maintaining solid performance for *Class 2* and *Class 3*.

The *Kruskal-Wallis test* resulted in a statistic of **2.0** and a p-value of **0.3679**, suggesting no statistically significant difference in accuracy across the three settings. Despite the superior performance of the *Dynamic Overlap* setting in terms of raw accuracy, the *Mann-Whitney U tests* comparing it with the other two settings yielded non-significant results (p-value = **1.0** for both comparisons).

Overall, the *Dynamic Overlap* setting consistently performed better across multiple metrics, but the differences were not significant enough to demonstrate that the method substantially outperformed the other approaches.

Chapter 10

Discussion

The results from this project indicate that the dynamic epsilon with overlap setting consistently outperforms both static with overlap and without overlap settings across various clustering metrics. In terms of sibling cluster detection, Dynamic EPS demonstrates improved performance, particularly in Dataset 2, where it detects 90.10% sibling clusters, compared to 63.07% for Static Overlap and 0% for No Overlap. This suggests that Dynamic EPS is better suited for tracking temporal changes in cluster structure. Conversely, No Overlap Static consistently reports 0% sibling detection due to the lack of overlap between time windows.

Dynamic EPS produced fewer orphans across datasets concerning orphan clusters, emphasizing its effectiveness in minimizing isolated clusters. Static Overlap, however, resulted in approximately 30% more orphan clusters than Dynamic EPS, highlighting its limitations in adapting to temporal data.

The clustering quality metrics further support these findings. The Dynamic EPS configuration yielded higher Silhouette Scores, Davies-Bouldin Scores, and Calinski-Harabasz Scores than the other settings. However, all settings demonstrated poor Silhouette Scores, indicating challenges in maintaining cluster separation.

The Poisson regression results supported the findings from clustering metrics by providing a close alignment between predicted and actual points across all datasets. Dynamic EPS consistently produced accurate predictions, with accuracy percentages ranging from 93.52% to 99%, indicating an effective model for estimating cluster occurrences. Both Static Overlap and No Overlap Static settings showed excellent alignment between predicted and actual points. Still, the risk of overfitting is apparent, as seen in lower accuracy percentages, especially in Dataset 1, where No Overlap Static achieved only 81.58%.

When evaluating the kinematic feature match to known species' flight behaviours, Dynamic EPS consistently aligned with expected ranges, outperforming the other settings in key metrics like average height, trajectory curvature, and distance travelled.

This indicates that Dynamic EPS effectively captures more complex, dynamic movements, offering a more accurate representation of the birds' trajectories. Some metrics, such as turn rate and angular acceleration, performed poorly across all settings. This could be attributed to small cluster sizes, which may not provide enough data points to calculate these features accurately.

Additionally, limitations in radar reception might have contributed to inaccurate estimations, particularly for features like vertical acceleration and gentle movement changes. The inherent noise and lower resolution in radar data, especially at low

altitudes or in dense environments, could further explain the poor performance in these metrics across the datasets.

Despite these solid relative performances, statistical tests, such as the Kruskal-Wallis and Mann-Whitney U tests, on the classification accuracy did not detect statistically significant differences between the clustering settings (p-values > 0.05), possibly due to the limitations described below.

Limitations

The study had several limitations. The selected data only featured four species for a relatively short time window. The selected parameters—such as window size, overlap size, and baseline EPS—were chosen empirically and remained constant during runtime. It is possible that these values were not optimal for all datasets, and further fine-tuning during runtime could improve clustering performance. Additionally, radar-based data, while useful for tracking bird movements, comes with lower resolution and noise, affecting the accuracy of movement measurements compared to GPS-based data. The empty segments within the datasets posed difficulties in linking clusters even when optimal hyperparameters were used. To address this, incorporating a probabilistic model to link clusters could overcome the challenges posed by empty data segments and reduce the occurrence of orphan clusters.

In terms of future research, the analysis could be repeated with different datasets or parameters to validate the generalizability of the method across various species or movement types. Adding external factors such as weather data into the pipeline might also provide a deeper understanding of bird movement patterns and their clustering behaviour.

The methodology developed here could extend beyond bird movement tracking to other flying objects, such as drones or unmanned aerial vehicles (UAVs).

Chapter 11

Conclusions and Further work

The project successfully developed and tested a Dynamic epsilon with an overlap algorithm for clustering bird movement data. This method consistently outperformed static epsilon approaches in key metrics, such as sibling cluster detection, reducing orphan clusters, and aligning with known kinematic features. These results highlight the algorithm's improved capability to capture dynamic, complex movement patterns, particularly for bird trajectories.

The algorithm's adaptability makes it well-suited for movement analysis in wildlife tracking, especially for species with non-linear or complex migration patterns. While the research successfully met its aims in demonstrating improved clustering with Dynamic epsilon, further work is required to refine the approach and expand its applicability.

Limitations included the empirical selection of parameters (e.g., window size, overlap size, baseline epsilon) and the inherent noise in radar data, which may have impacted the accuracy of certain kinematic features. Additionally, statistical significance was not achieved in testing, limiting the ability to prove the Dynamic epsilon approach's superiority definitively.

In future research, it would be valuable to expand the analysis by testing the developed algorithm on additional datasets from different species or environments, which would help confirm the generalizability of the findings. Incorporating weather data into the analytical pipeline could provide deeper insights into how environmental conditions affect bird movements and clustering results. Adding a probabilistic model to link clusters may reduce the occurrence of orphan clusters, improving the overall algorithm's robustness. Furthermore, the methodology could be extended to other flying objects more easily tracked by radar, such as drones or other UAVs (unmanned aerial vehicles), to explore broader applications, such as air traffic monitoring and drone navigation.

Word Count: 9,911

References

[1] Xiaoya An et al. "Strp-dbscan: A parallel dbscan algorithm based on spatial-temporal random partitioning for clustering trajectory data". In: *Applied Sciences* 13.20 (2023), p. 11122.

[2] Mihael Ankerst et al. "OPTICS: Ordering points to identify the clustering structure". In: *ACM Sigmod record* 28.2 (1999), pp. 49–60.

[3] Xiangen Bai et al. "An adaptive threshold fast DBSCAN algorithm with preserved trajectory feature points for vessel trajectory clustering". In: *Ocean Engineering* 280 (2023), p. 114930.

[4] Derya Birant and Alp Kut. "ST-DBSCAN: An algorithm for clustering spatial–temporal data". In: *Data & knowledge engineering* 60.1 (2007), pp. 208–221.

[5] Adi Brill. *Trajectory-Analysis-Using-Dynamic-DBSCAN-and-Poisson-Model*. Version 1.0.0. 2024. URL: https://github.com/adibrill/Trajectory-Analysis-Using-Dynamic-DBSCAN-and-Poisson-Model.

[6] Cornell University. *All About Birds*. https://www.allaboutbirds.org/guide/. [Online; accessed 13-April-2024]. 2024.

[7] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[8] Ángel F García-Fernández et al. "Poisson multi-Bernoulli mixture filter: Direct derivation and implementation". In: *IEEE Transactions on Aerospace and Electronic Systems* 54.4 (2018), pp. 1883–1901.

[9] Anil K Ghosh, Probal Chaudhuri, and Debasis Sengupta. *Classification using kernel density estimates: Multiscale analysis and visualization*. 2006.

[10] Lei Gong, Toshiyuki Yamamoto, and Takayuki Morikawa. "Identification of activity stop locations in GPS trajectories by DBSCAN-TE method combined with support vector machines". In: *Transportation research procedia* 32 (2018), pp. 146–154.

[11] Haitian Wei. *How to measure clustering performances when there are no ground truth?* https://medium.com/@haataa/how-to-measure-clustering-performances-when-there-are-no-ground-truth-db027e9a871c. [Online; accessed 13-May-202]. 2020.

[12] Katarzyna Juraszek et al. "Extended Kalman filter for large scale vessels trajectory tracking in distributed stream processing systems". In: *Advanced Analytics and Learning on Temporal Data: 4th ECML PKDD Workshop, AALTD 2019, Würzburg, Germany, September 20, 2019, Revised Selected Papers 4*. Springer. 2020, pp. 151–166.

[13] Bart Kranstauber et al. “A dynamic Brownian bridge movement model to estimate utilization distributions for heterogeneous animal movement”. In: *Journal of Animal Ecology* 81.4 (2012), pp. 738–746.

[14] Bart Kranstauber et al. “High-resolution spatial distribution of bird movements estimated from a weather radar network”. In: *Remote Sensing* 12.4 (2020), p. 635.

[15] Patrick Laube, Stephan Imfeld, and Robert Weibel. “Discovering relative motion patterns in groups of moving point objects”. In: *International Journal of Geographical Information Science* 19.6 (2005), pp. 639–668.

[16] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. “Trajectory clustering: a partition-and-group framework”. In: *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 2007, pp. 593–604.

[17] Simeon Lisovski et al. “Inherent limits of light-level geolocation may lead to over-interpretation”. In: *Current Biology* 28.3 (2018), R99–R100.

[18] Movebank Organization. *Movebank Organization Website*. https://www.movebank.org/cms/movebank-main. [Online; accessed 09-April-2024]. 2024.

[19] Ebuka Osunwoke et al. “A Machine Learning-Enabled Clustering Approach for Large-scale Classification of Solar Data”. In: Nov. 2021, pp. 01–06. DOI: 10.1109/NAPS52732.2021.9654549.

[20] Nikos Pelekis et al. “In-DBMS Sampling-based Sub-trajectory Clustering.” In: *EDBT*. 2017, pp. 632–643.

[21] Jiadong Ren and Ruiqing Ma. “Density-based data streams clustering over sliding windows”. In: *2009 Sixth international conference on fuzzy systems and knowledge discovery*. Vol. 5. IEEE. 2009, pp. 248–252.

[22] Paul Roback and Julie Legler. *Applied Generalized Linear Models and Multilevel Models in R*. 2021.

[23] Bret W Tobalske et al. “Three-dimensional kinematics of hummingbird flight”. In: *Journal of Experimental Biology* 210.13 (2007), pp. 2368–2382.

[24] Zi Jing Wang, Ye Zhu, and Kai Ming Ting. “Distribution-Based Trajectory Clustering”. In: *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2023, pp. 1379–1384.

[25] Wikipedia. *Calinski–Harabasz index — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Calinski%E2%80%93Harabasz%20index&oldid=1237618066. [Online; accessed 13-September-2024]. 2024.

[26] Wikipedia. *Davies–Bouldin index — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Davies%E2%80%93Bouldin%20index&oldid=1186394772. [Online; accessed 13-September-2024]. 2024.

[27] Wikipedia. *Silhouette (clustering) — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Silhouette%20(clustering)&oldid=1231979269. [Online; accessed 13-September-2024]. 2024.

Appendix A

Sliding Window Size Optimization Results

Each dataset exhibited a distinct distribution of epsilon values. The following plots illustrate the variation of epsilon values across the sliding windows.

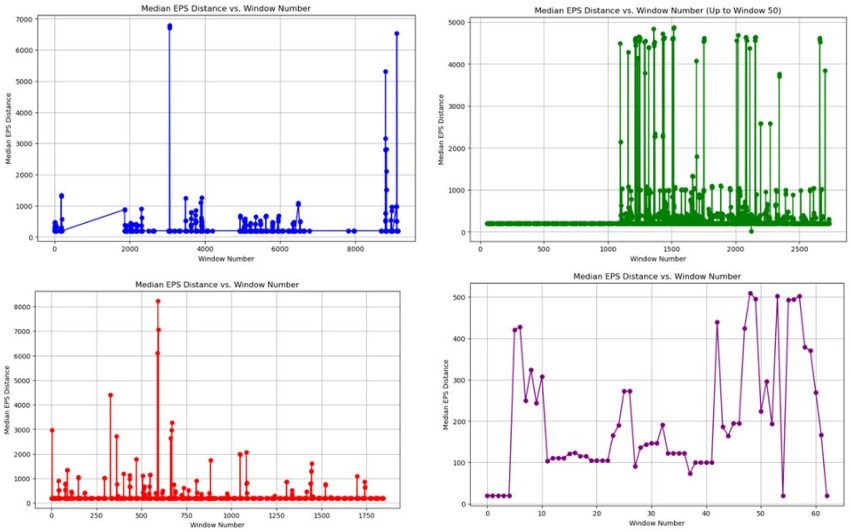


FIGURE A.1: EPS Distance per window per dataset

The following pairs of plots display four parameters for each dataset: the minimum distance between the closest points of two clusters, the minimum distance between two centroids, the maximum cluster diameter (maximum distance within the cluster), and the selected EPS value for the window.

In all four datasets, it was observed that the selected EPS remained unchanged when the cluster diameter was smaller than the initial EPS. However, the EPS value was adjusted in windows where the distance between clusters or centroids exceeded the initial EPS. The second plot in each pair shows the minimum distance between clusters (blue points) and the minimum distance between centroids (red points). A clear gap is visible between these distances, with the selected EPS value effectively separating the clusters even if it differed from the EPS values in adjacent windows.

In the first three datasets, several windows contained no data, disrupting the continuity of the blue and red lines. This absence of data led to breaks in the linkage between clusters across windows.

First Dataset:

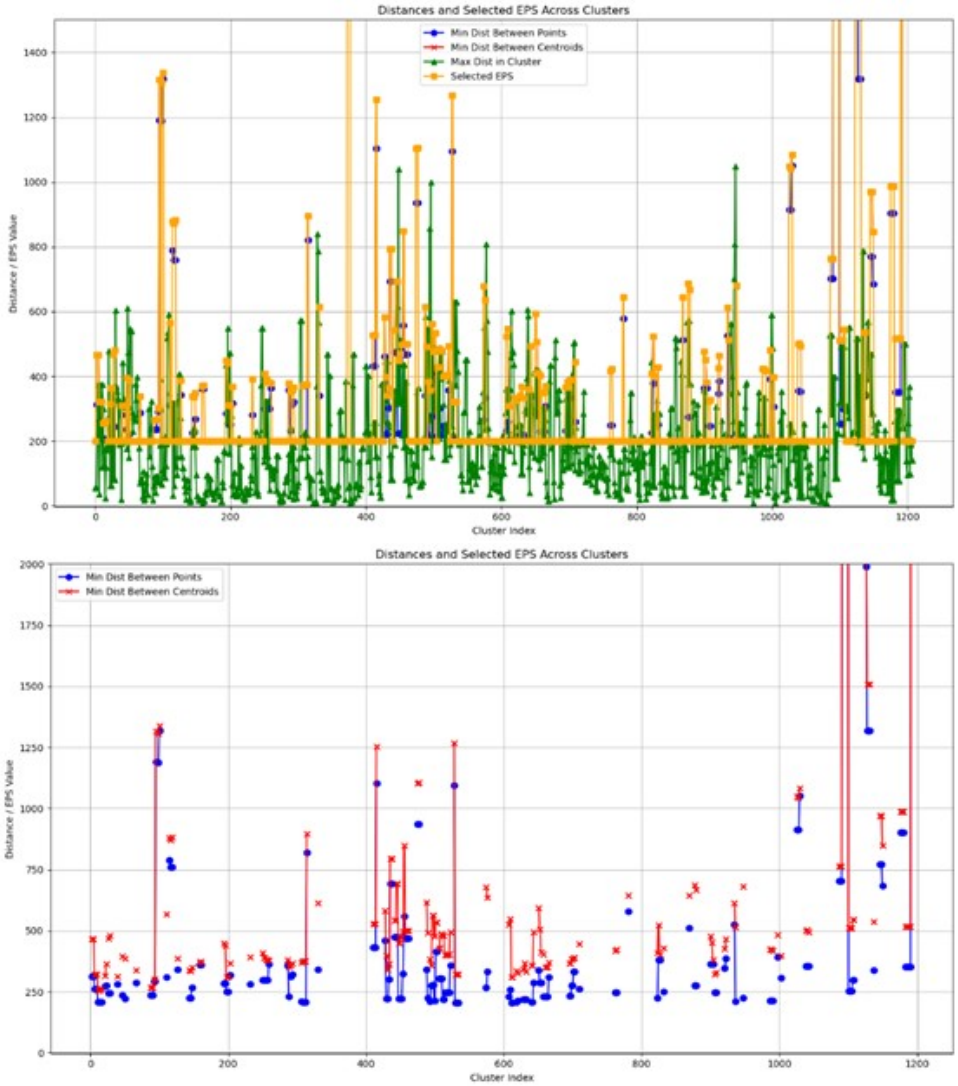


FIGURE A.2: Intera and Inter-Cluster Distances for Common Kestrel

Second dataset:

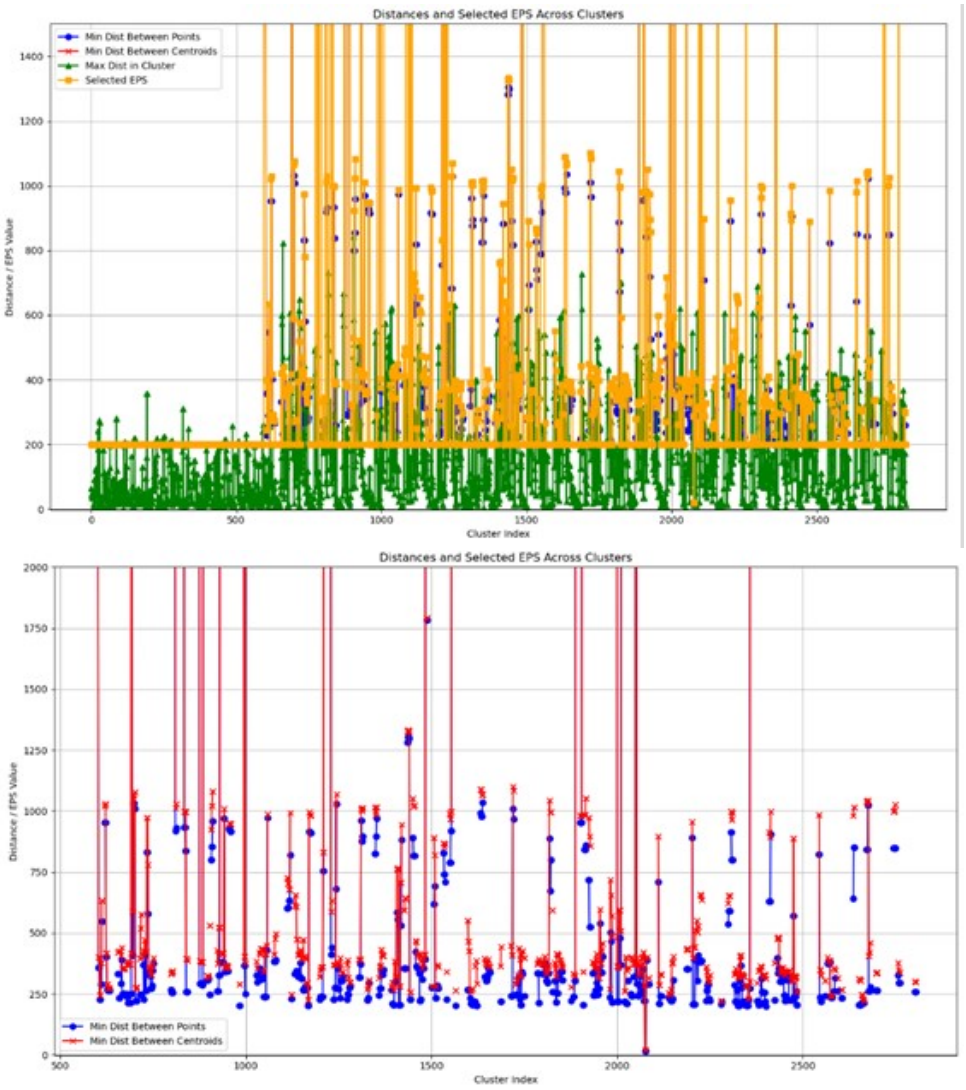


FIGURE A.3: Intera and Inter-Cluster Distances for Hurring Gull

Third dataset:

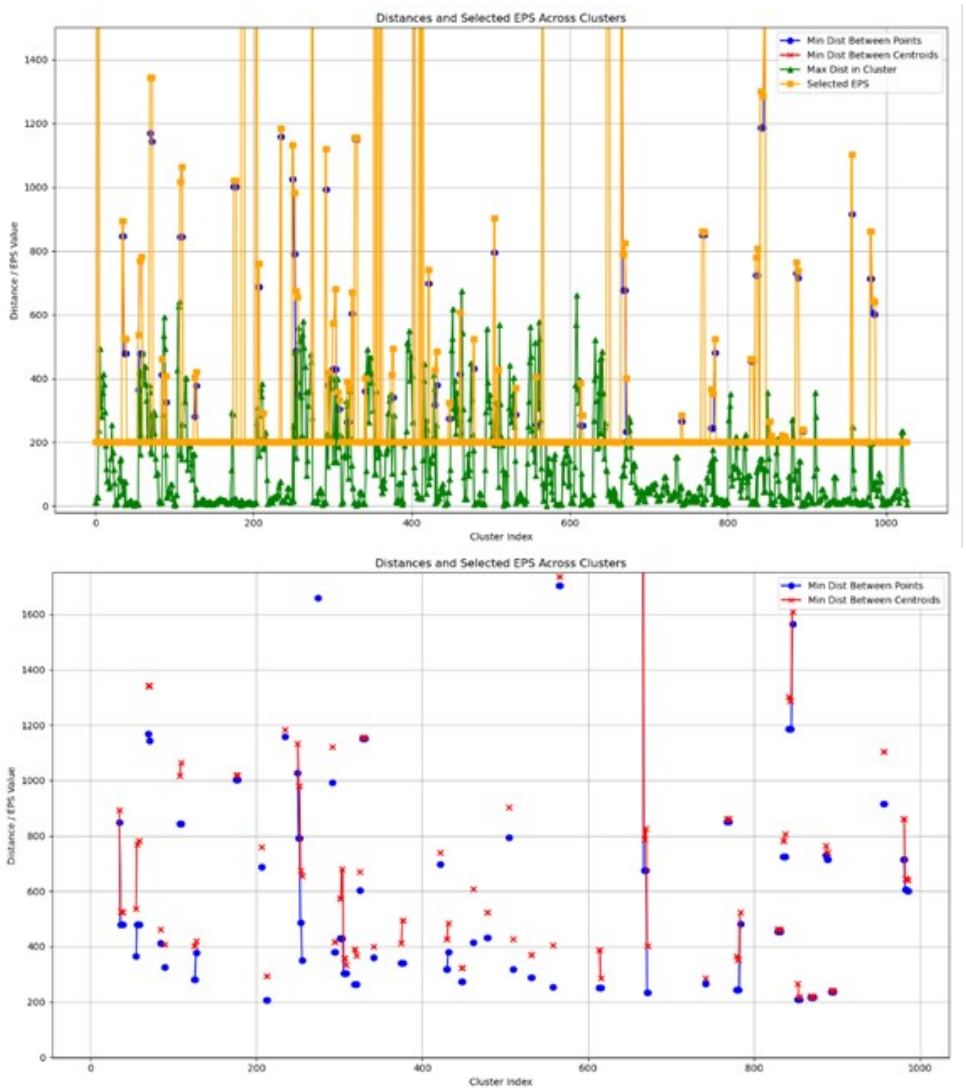


FIGURE A.4: Intera and Inter-Cluster Distances for White Stork

Fourth dataset:

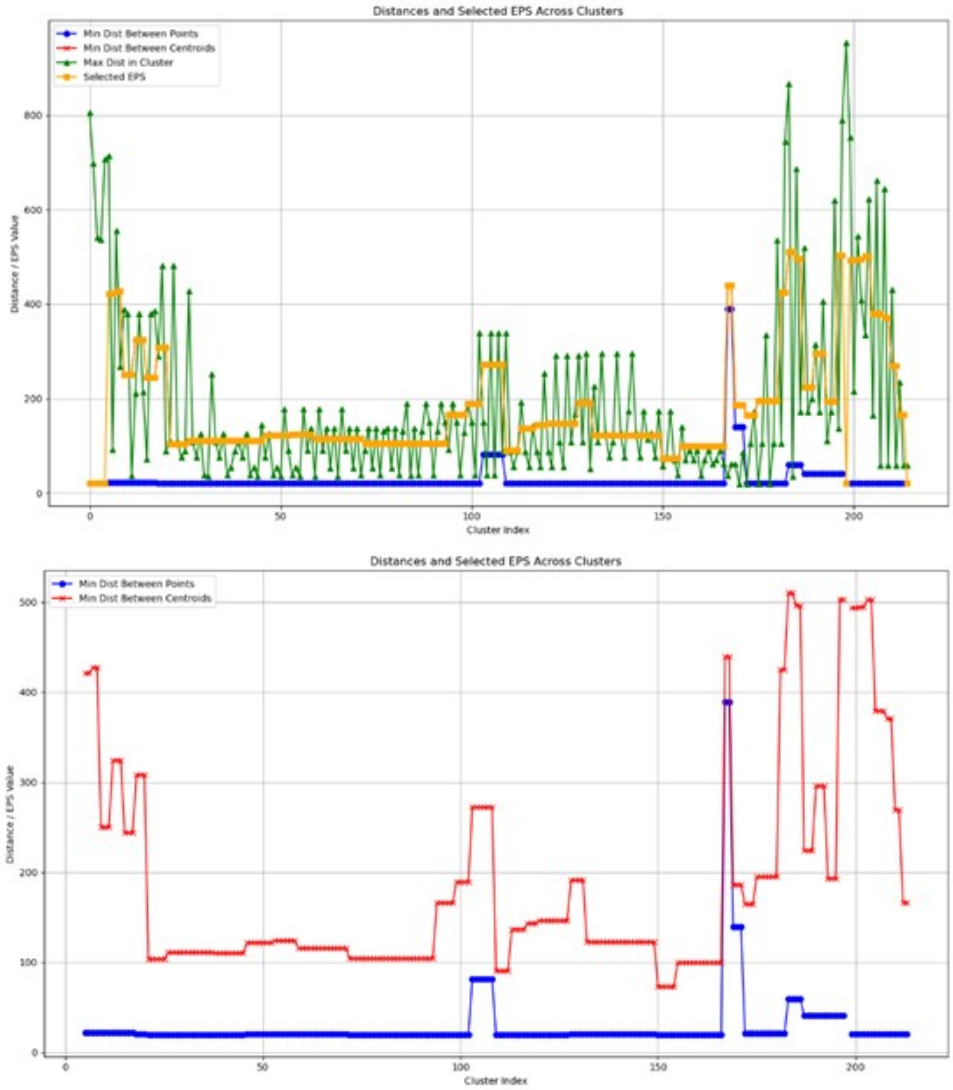


FIGURE A.5: Intera and Inter-Cluster Distances for Homing Pigeon

The four cluster distance metrics were analysed across four classes, revealing distinct patterns in clustering behaviour. Classes 1 and 2 exhibit more consistent clustering metrics, with tighter and more uniformly sized clusters, while Classes 3 and 4 show greater variability in all metrics, indicating more complex clustering behaviour.

The variability in epsilon values across classes suggests that different clustering strategies may be required for different data characteristics, particularly in the more complex cases like Class 4.

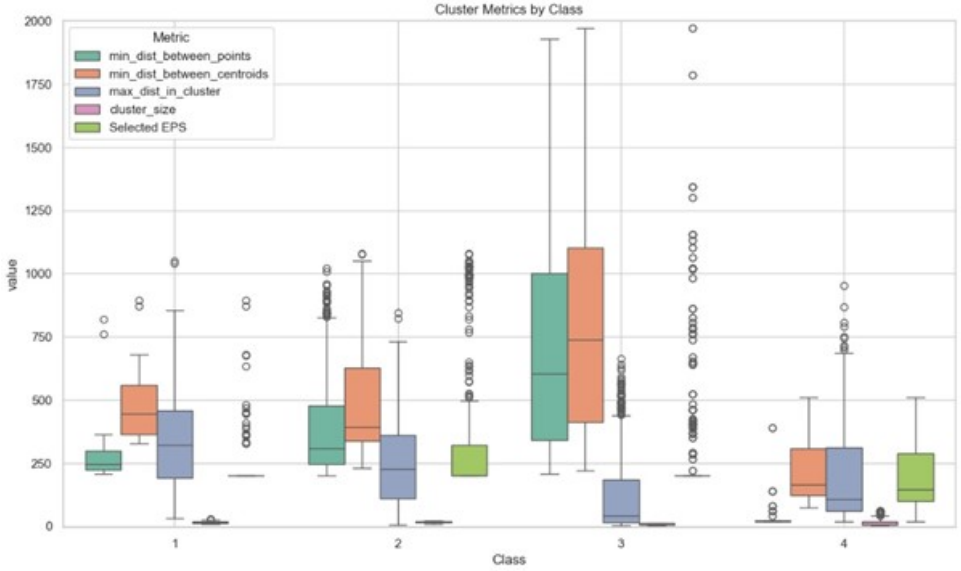
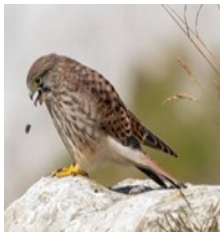


FIGURE A.6: Boxplot of Distance between points, distance between clusters, window epsilon and Cluster size (circumference) for each dataset

Appendix B

Table of Species and Kinematic Features

TABLE B.1: Table of the species and their kinematic features

Table of the species and their kinematic features			
Common Kestrel	**Herring Gulls**	**White Stork**	**Homing Pigeon**
			
Speed Typical: 15 - 40 $\frac{km}{h}$ (9 - 25 mph) Diving Speed: Up to 60 $\frac{km}{h}$ (37 mph)	**Speed** Typical: 40 - 60 $\frac{km}{h}$ (25 - 37 mph) Max: Up to 80 $\frac{km}{h}$ (50 mph) in strong winds	**Speed** Typical: 25 - 50 $\frac{km}{h}$ (15 - 31 mph) During migration Max: Up to 65 $\frac{km}{h}$ (40 mph) with tailwinds	**Speed** Typical: 60 - 80 $\frac{km}{h}$ (37 - 50 mph) Max: Up to 110 $\frac{km}{h}$ (68 mph) in short bursts
Acceleration Range: 0.5 - 2 $\frac{m}{s^2}$ (moderate acceleration during hunting dives)	**Acceleration** Range: 0.2 - 1 $\frac{m}{s^2}$ (low acceleration due to gliding flight)	**Acceleration** Range: 0.3 - 1.5 $\frac{m}{s^2}$ (during thermal soaring and gliding)	**Acceleration** Range: 1 - 3 $\frac{m}{s^2}$ (higher acceleration during initial take-off)
Angular Velocity Range: 2 - 5 $\frac{rad}{s}$ (during rapid direction changes while hunting)	**Angular Velocity** Range: 0.5 - 2 $\frac{rad}{s}$ (typically moderate due to steady gliding flight)	**Angular Velocity** Range: 0.5 - 1.5 $\frac{rad}{s}$ (moderate turns during thermal circling)	**Angular Velocity** Range: 1 - 4 $\frac{rad}{s}$ (rapid turns during flight navigation)

Common Kestrel	Herring Gulls	White Stork	Homing Pigeon
Angle to Horizon (Pitch Angle) Range: -15° to +75° (depending on hunting or hovering activities)	**Angle to Horizon (Pitch Angle)** Range: -10° to +45° (more shallow angles during gliding)	**Angle to Horizon (Pitch Angle)** Range: -5° to +60° (depending on gliding or ascending thermals)	**Angle to Horizon (Pitch Angle)** Range: -20° to +70° (sharp angles during sudden direction changes)
Up/Down Direction (Kendall's Tau) Range: Highly variable, depending on whether the kestrel is hunting (hovering) or moving between perches.	**Up/Down Direction (Kendall's Tau)** Range: Generally positive when ascending for flight, slightly negative when gliding down to land.	**Up/Down Direction (Kendall's Tau)** Range: Typically positive when ascending in thermals, close to zero during level gliding.	**Up/Down Direction (Kendall's Tau)** Range: Often near zero during level flight, highly variable during climbing or descending phases.
Average Flight Altitude Typically between 10-50 meters above the fly ground, though they may fly higher during migration or when crossing obstacles.	**Average Flight Altitude** Varies significantly often, between 10-200 meters but can soar up to 300 meters or higher during migration.	**Average Flight Altitude** Typically flies between 100-500 meters, but can soar up up to 1500 during migration.	**Average Flight Altitude** Typically between 50-150 meters, but can higher during long-distance races or to avoid obstacles.

Appendix C

Features Extraction Box Plots

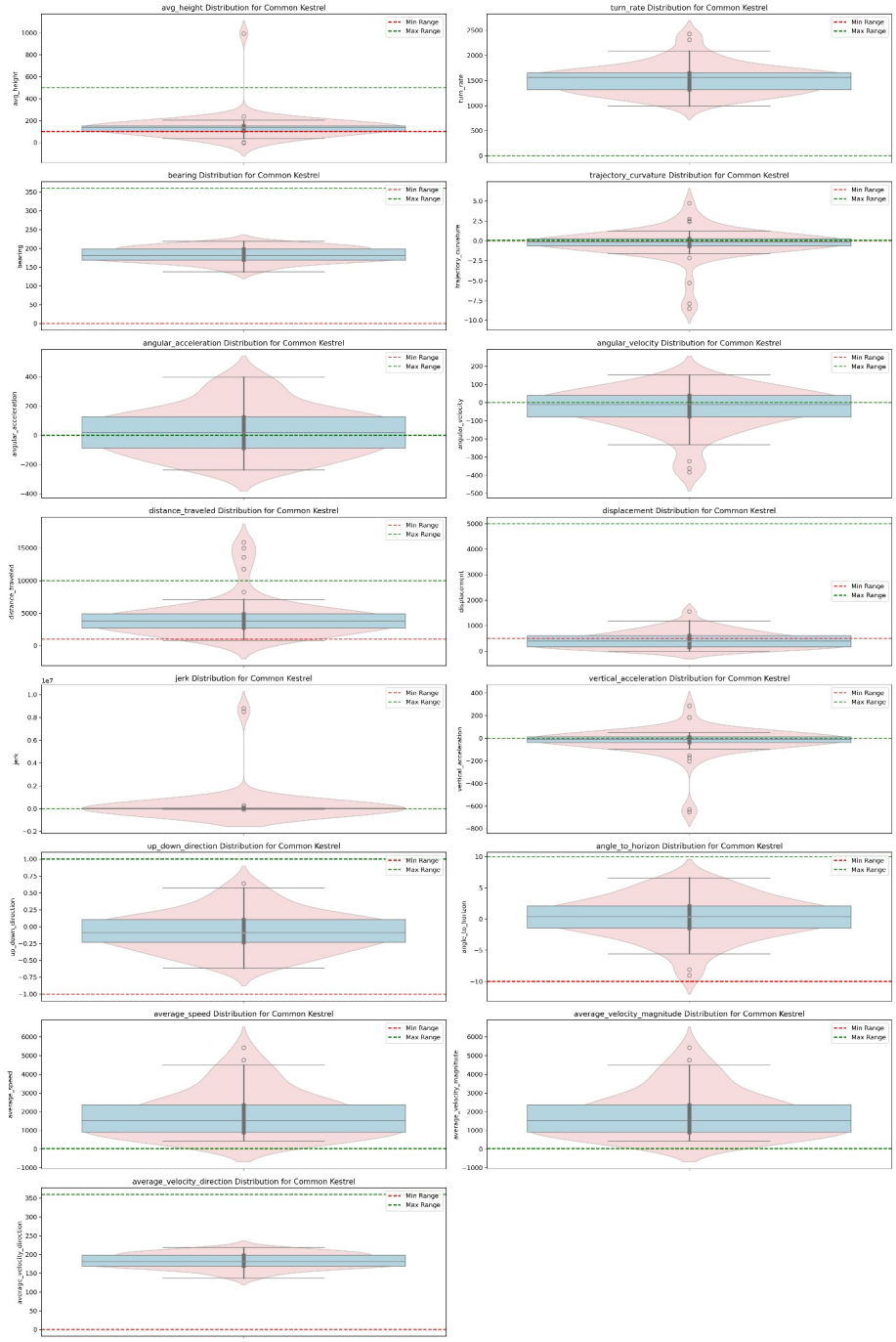


FIGURE C.1: Common Kestrel Clusters Kinematics Distribution

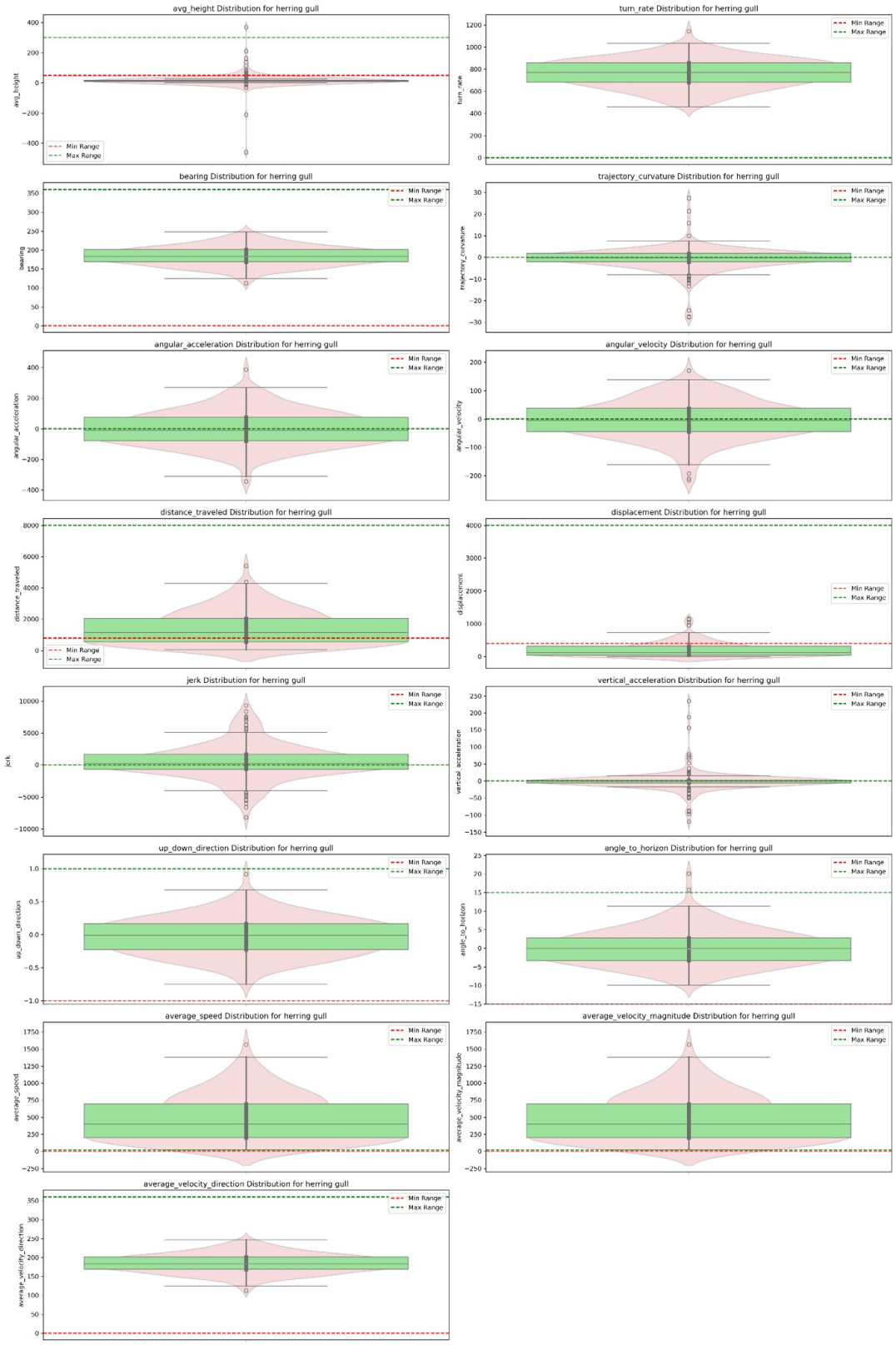


FIGURE C.2: Herring Gull Clusters Kinematics Distribution

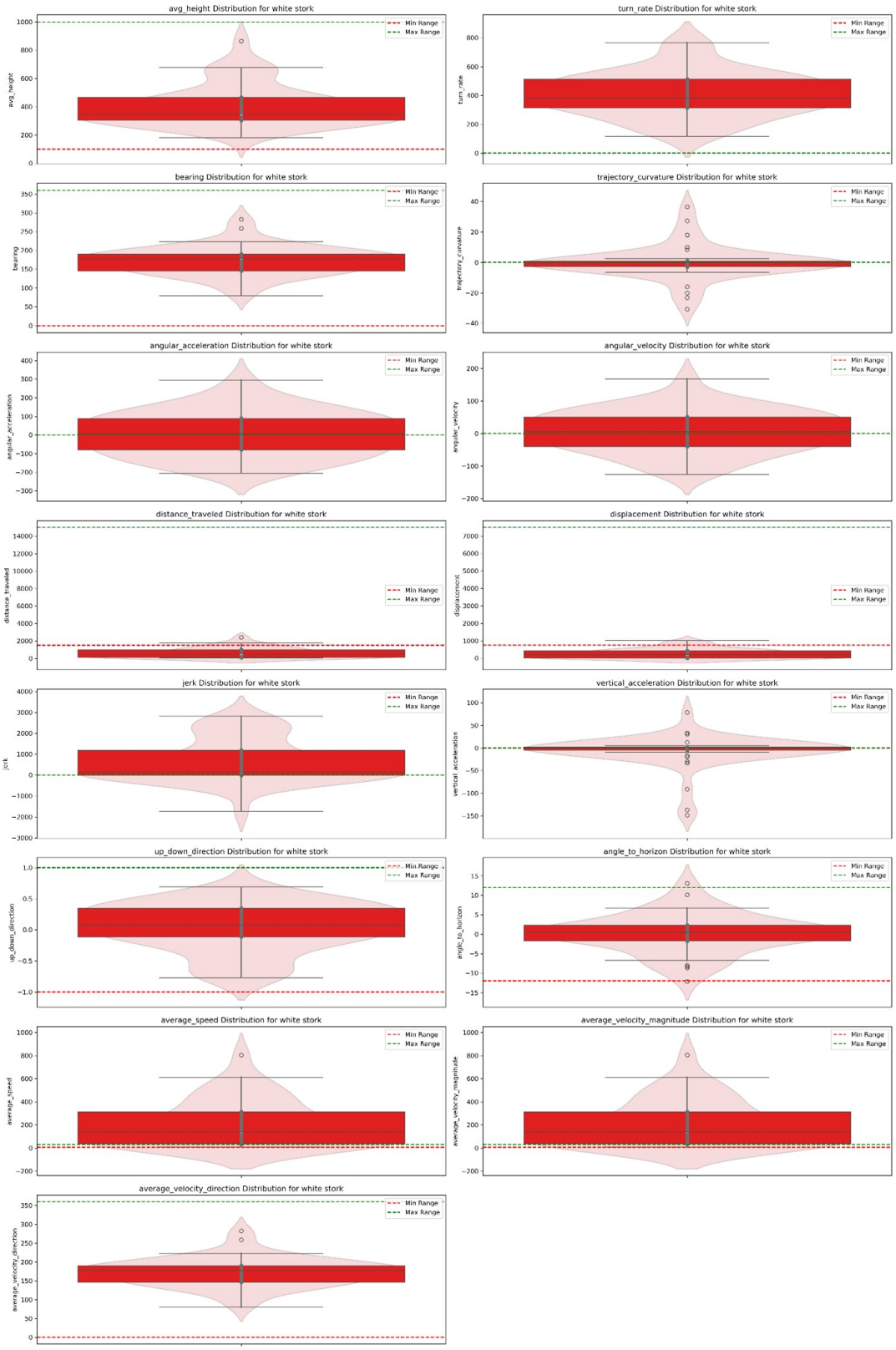


FIGURE C.3: White Stork Clusters Kinematics Distribution

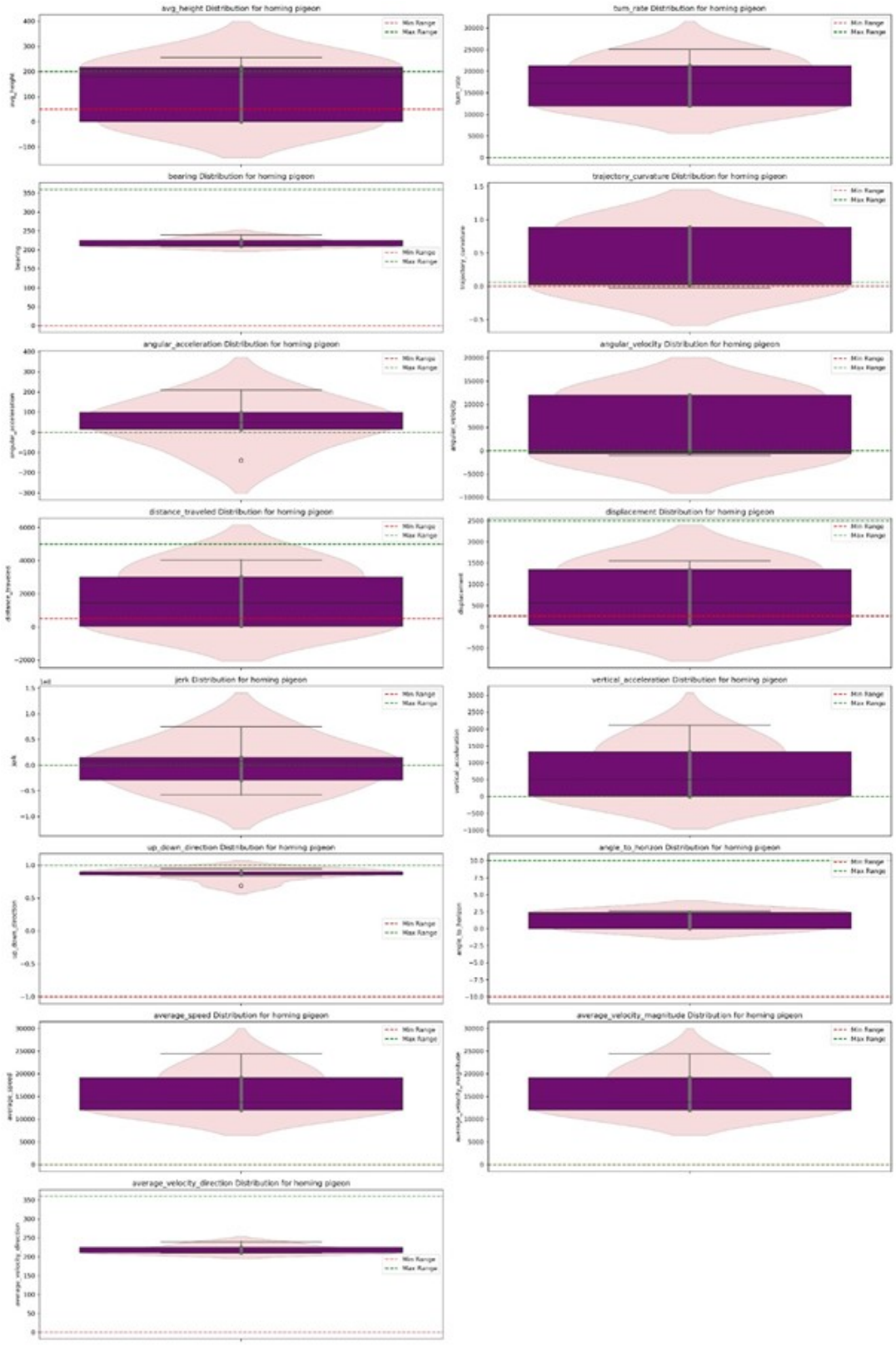


FIGURE C.4: Homing Pigeon Clusters Kinematics Distribution

Appendix D

Poisson Regression Detailed Results

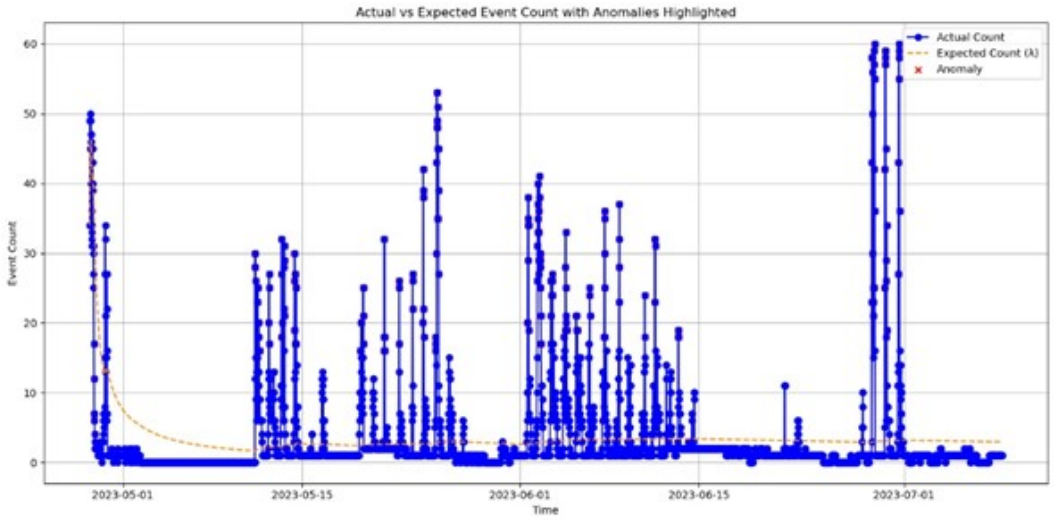


FIGURE D.1: Actual vs Expected points Count for Common Kestrel dataset

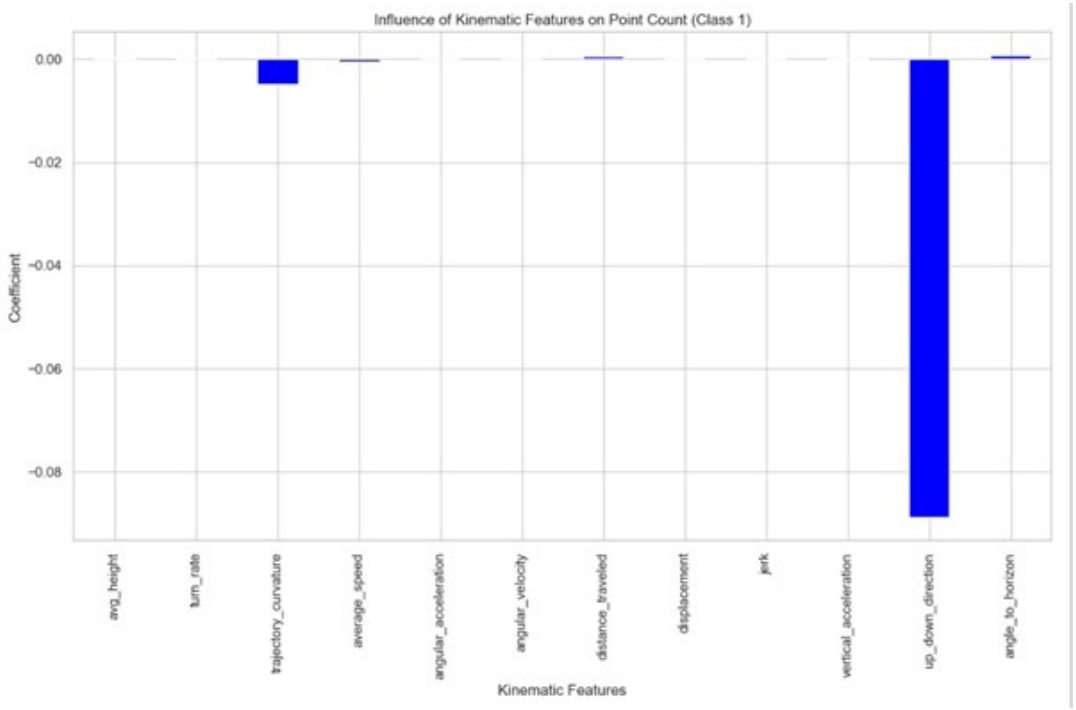


FIGURE D.2: Influence of Kinematics Features on prediction for Common Kestrel dataset

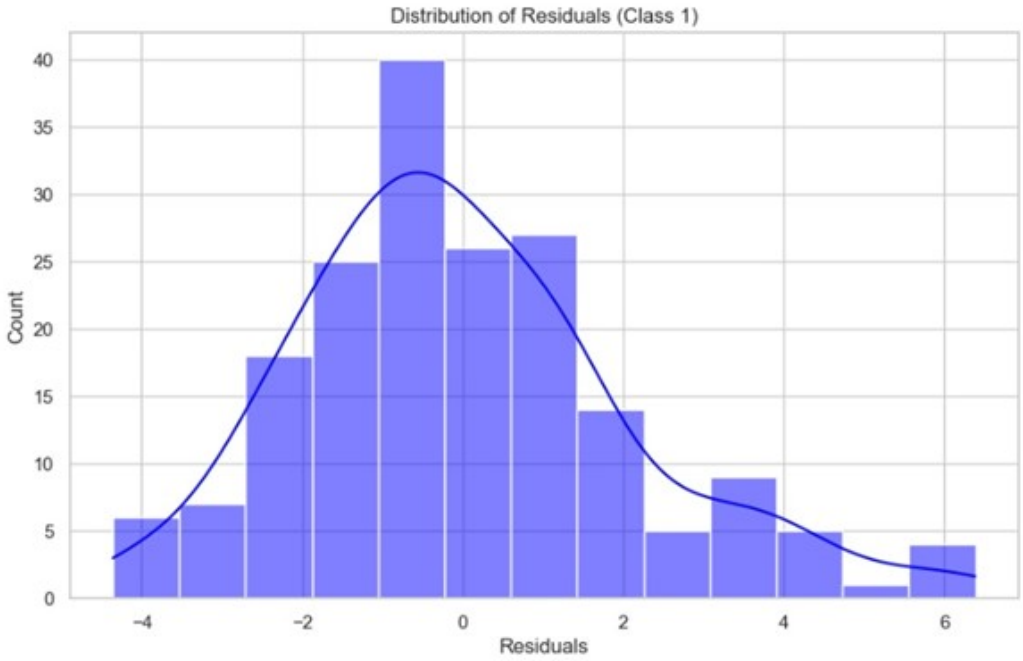


FIGURE D.3: Distribution of Residuals for Common Kestrel dataset

Second dataset:

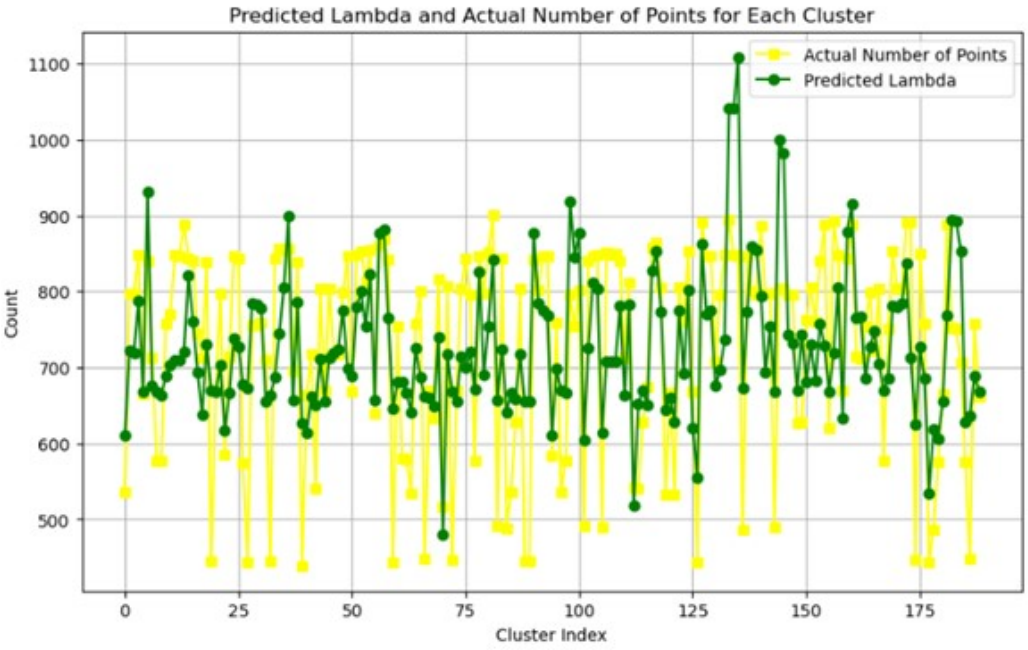


FIGURE D.4: Predicted and actual Lambda values per cluster for Herring Gull dataset

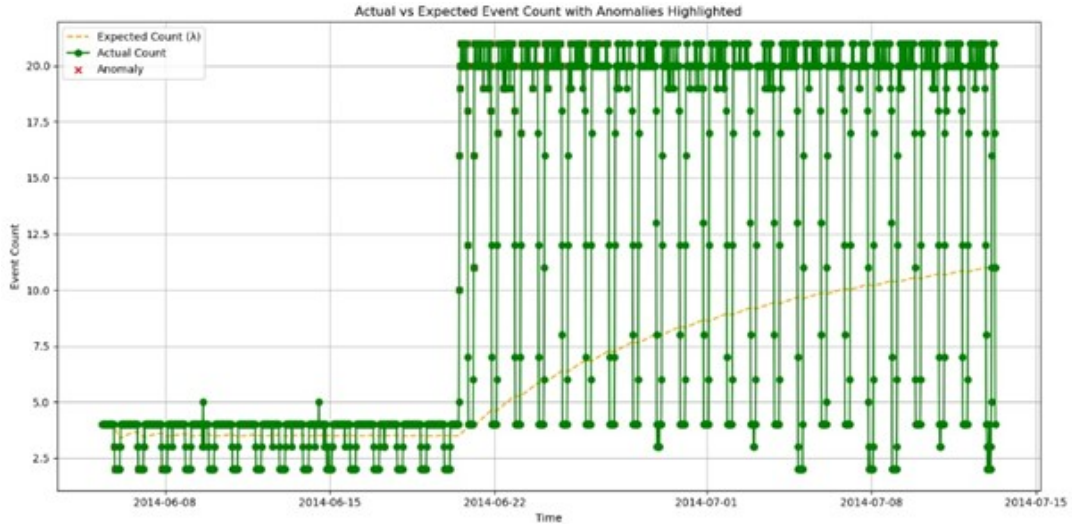


FIGURE D.5: Actual vs Expected points Count for Herring Gull dataset

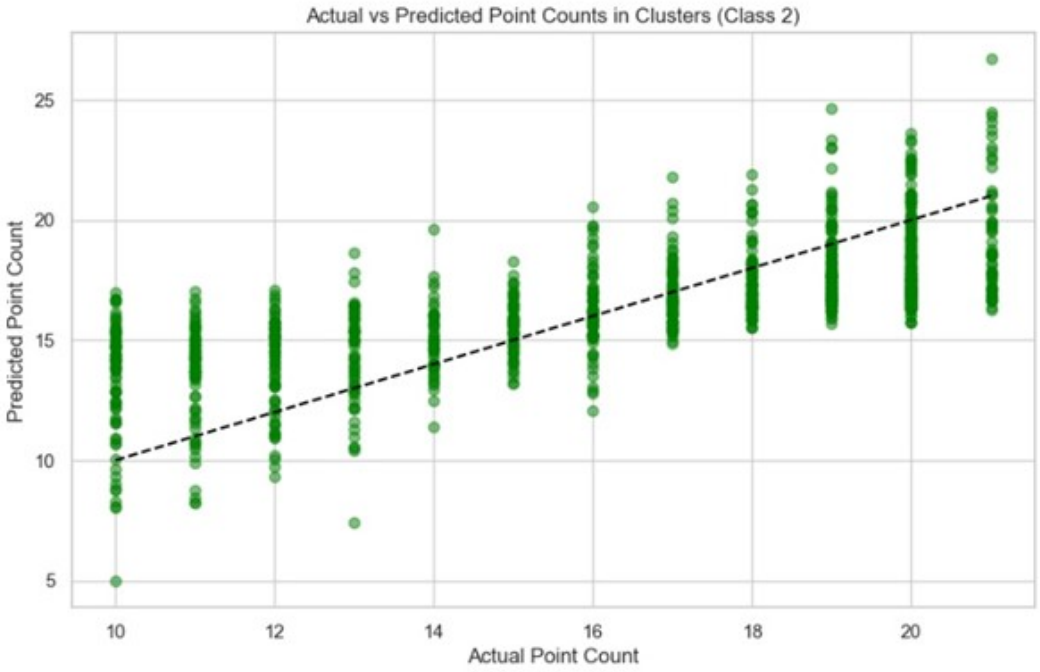


FIGURE D.6: Actual vs Predicted Cluster Size for Herring Gull dataset

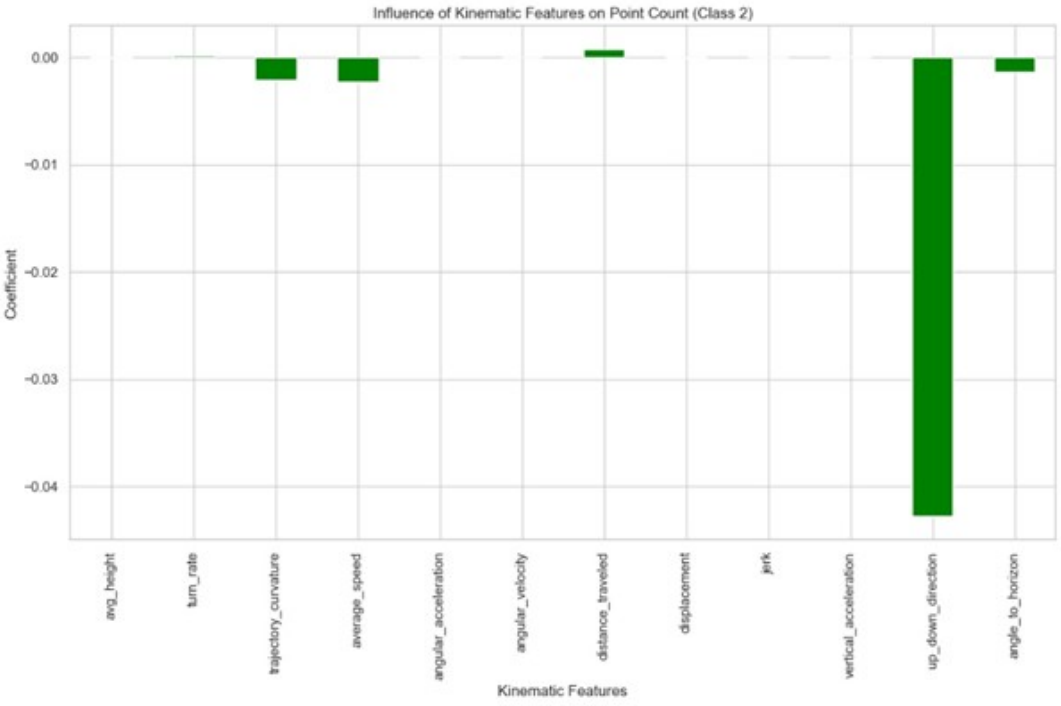


FIGURE D.7: Influence of Kinematics Features on prediction for Herring Gull dataset

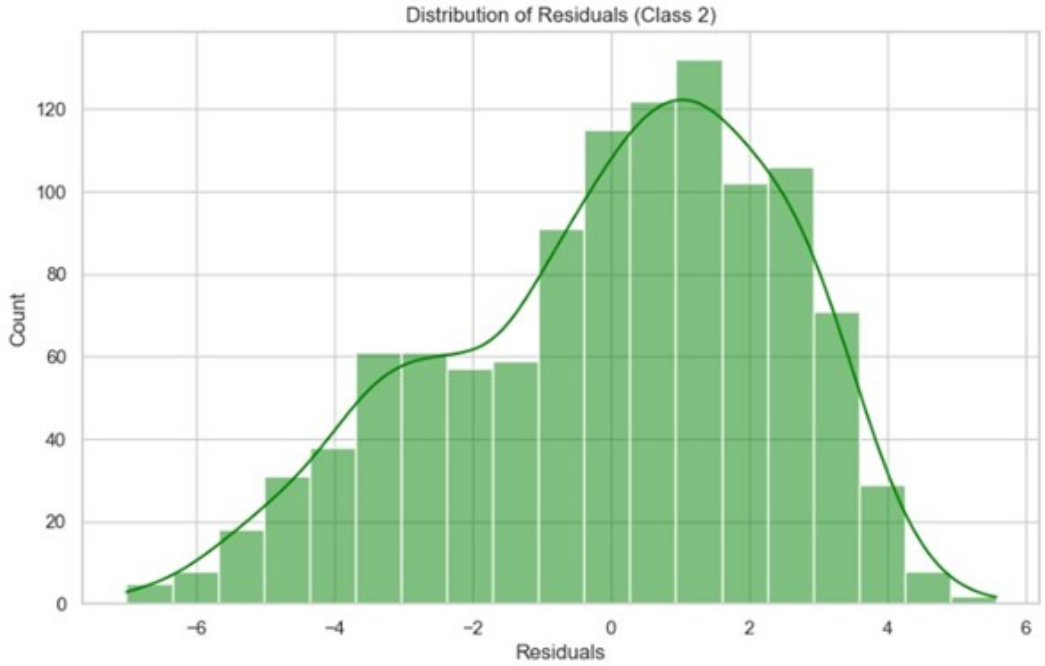


FIGURE D.8: Distribution of Residuals for Herring Gull dataset

Third dataset:

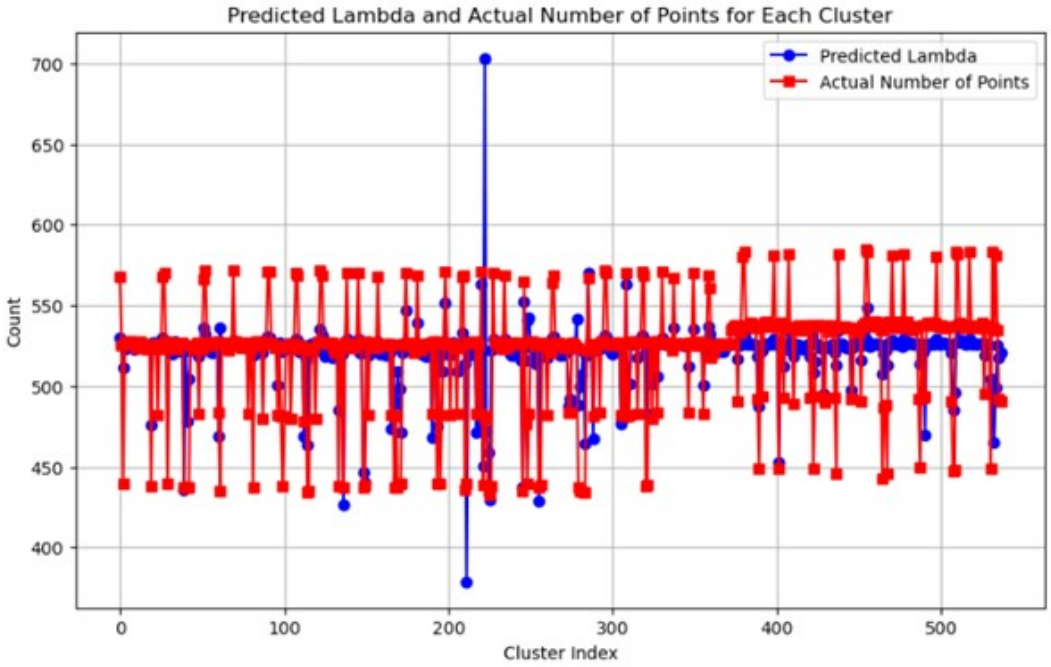


FIGURE D.9: Predicted and actual Lambda values per cluster for White Stork dataset

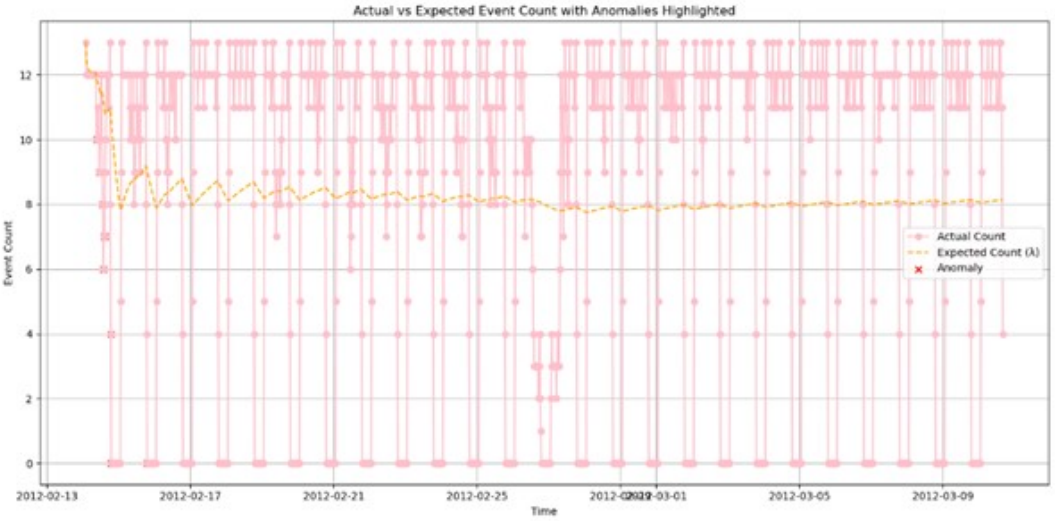


FIGURE D.10: Actual vs Expected points Count for White Stork dataset

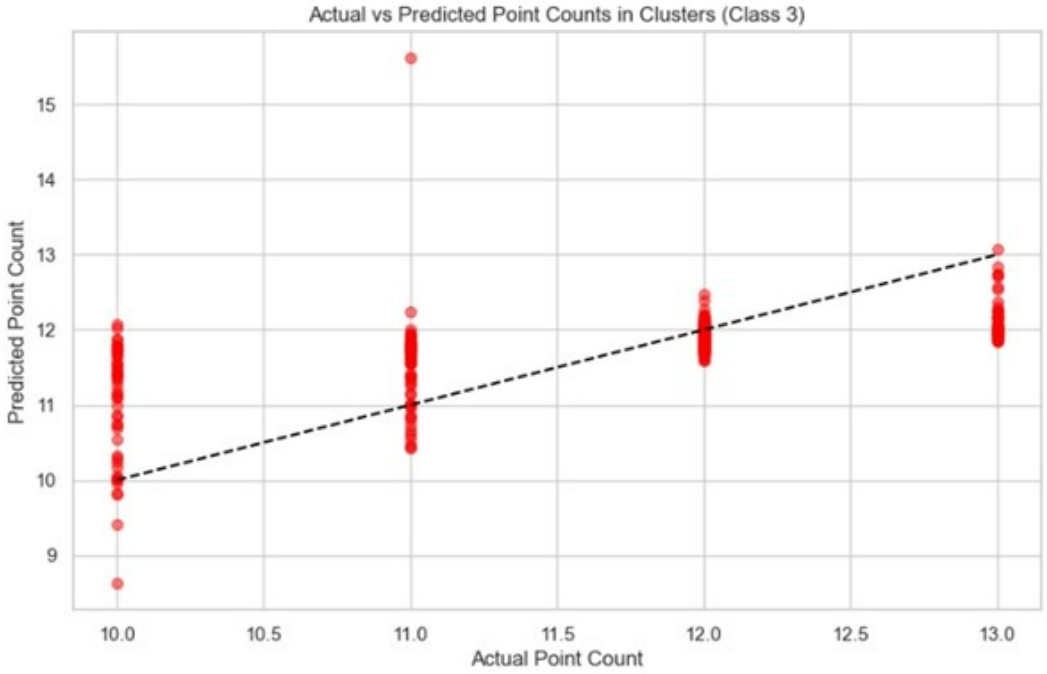


FIGURE D.11: Actual vs Predicted Cluster Size for White Stork dataset

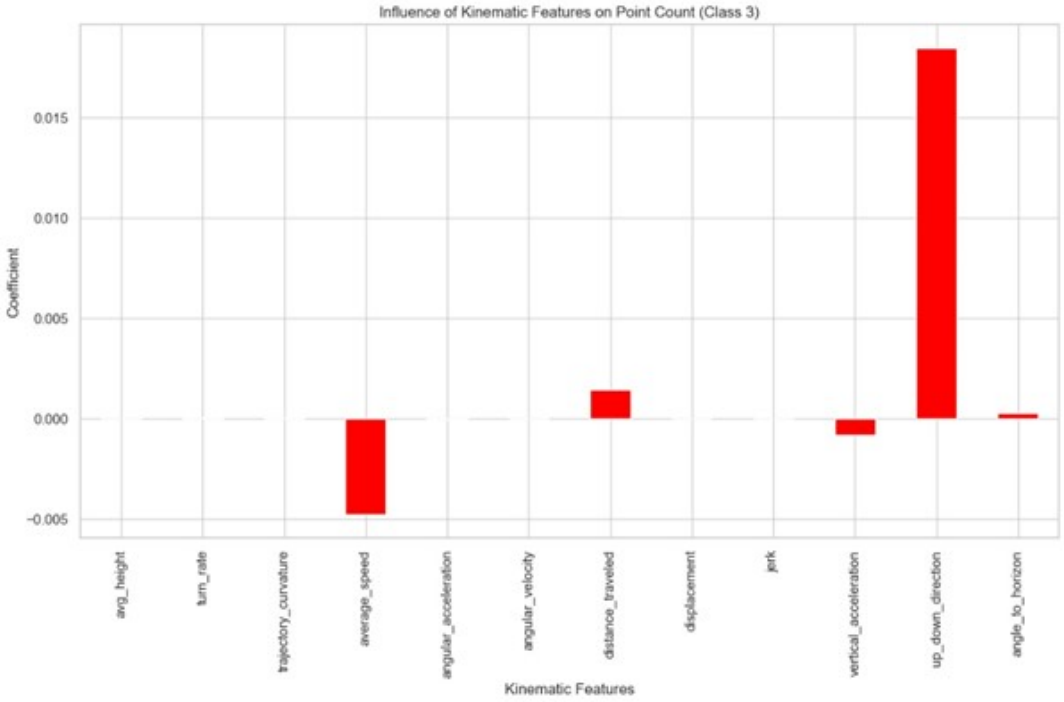


FIGURE D.12: Influence of Kinematics Features on prediction for White Stork dataset

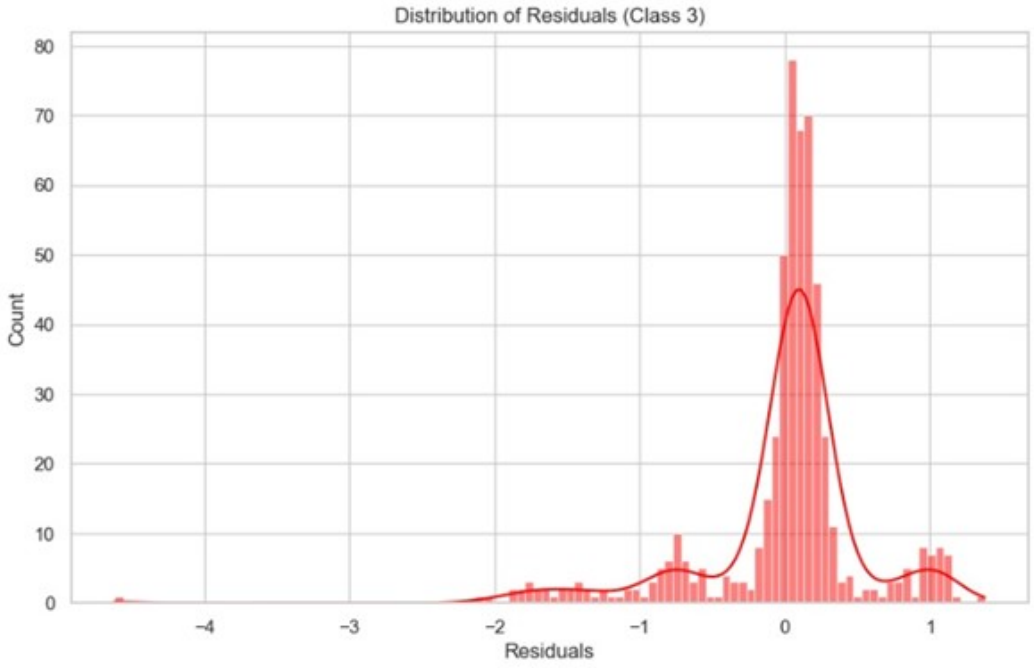


FIGURE D.13: Distribution of Residuals for White Stork dataset

fourth dataset:

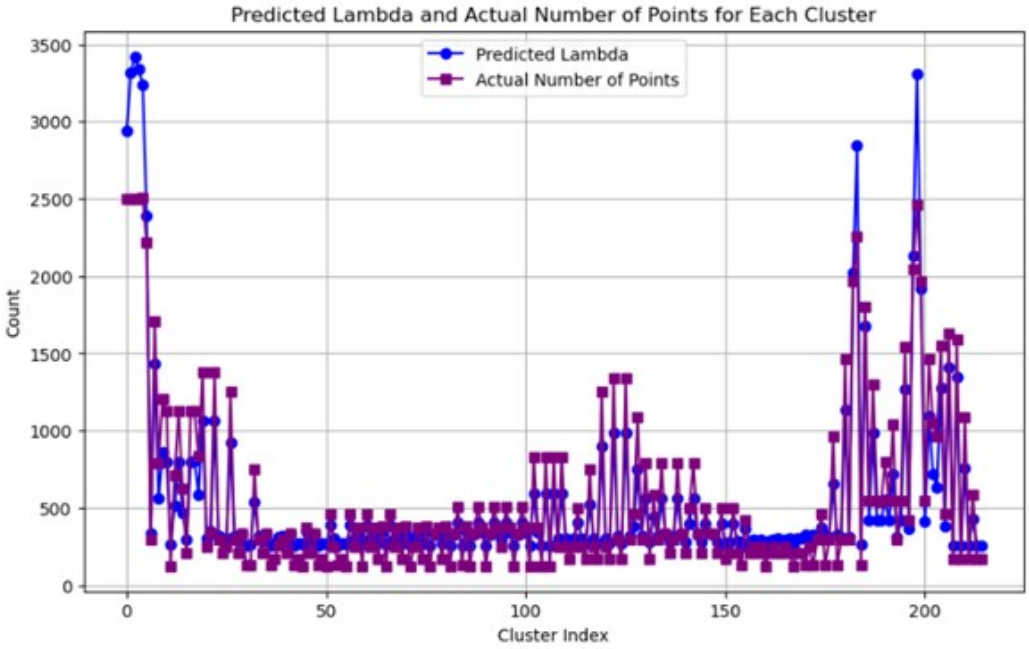


FIGURE D.14: Predicted and actual Lambda values per cluster for Homing Pigeon dataset

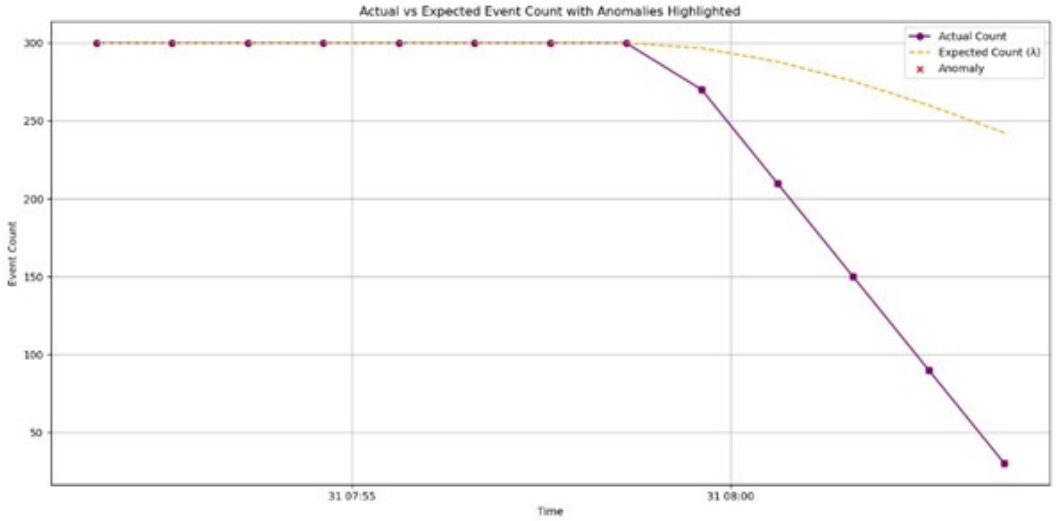


FIGURE D.15: Actual vs Expected points Count for Homing Pigeon dataset

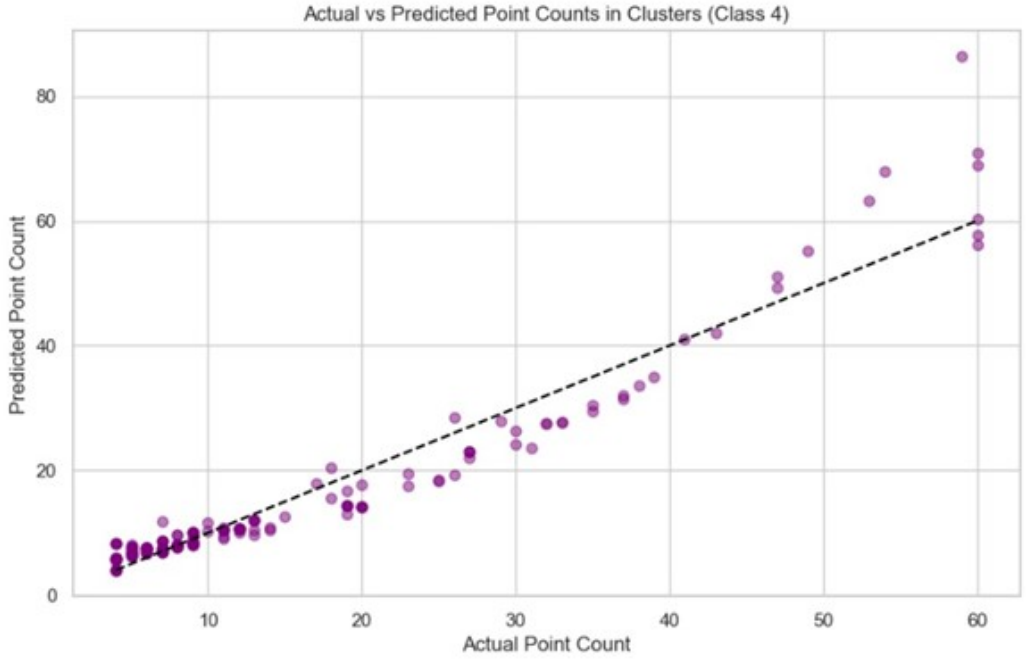


FIGURE D.16: Actual vs Predicted Cluster Size for Homing Pigeon dataset

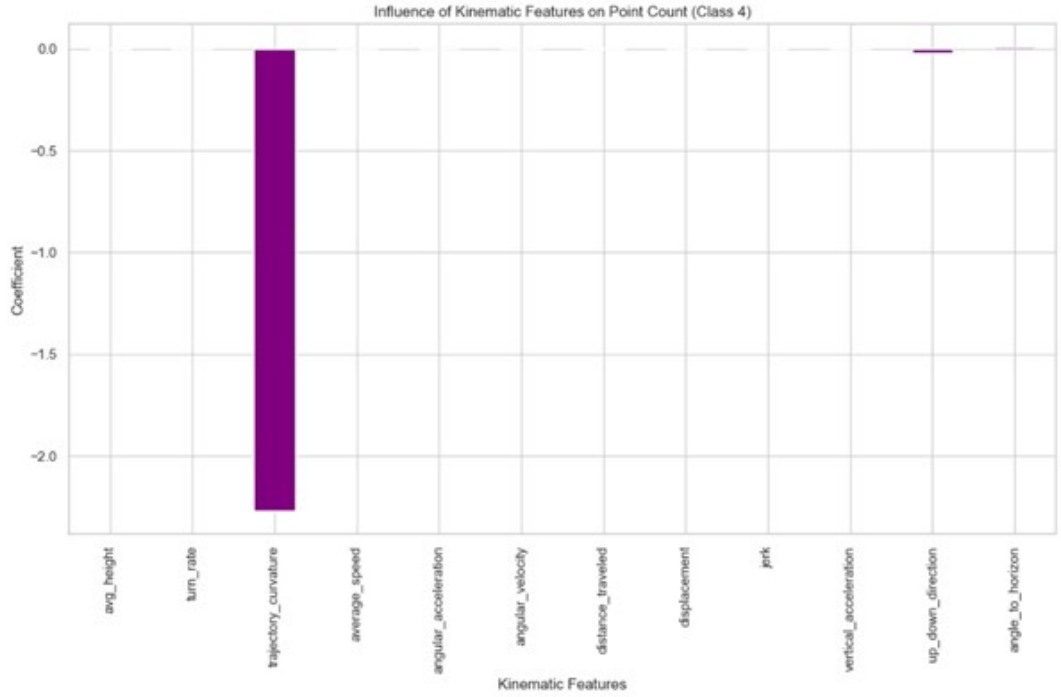


FIGURE D.17: Influence of Kinematics Features on prediction for Homing Pigeon dataset

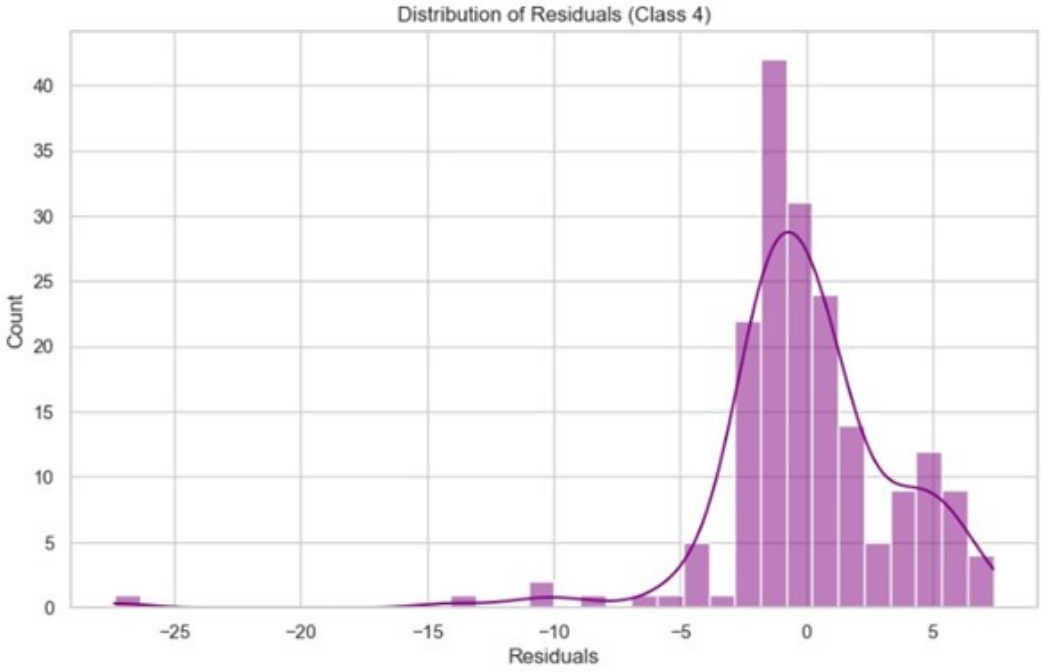
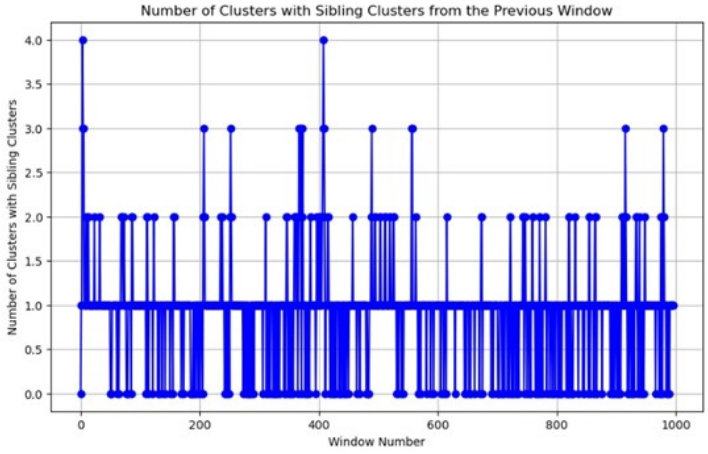


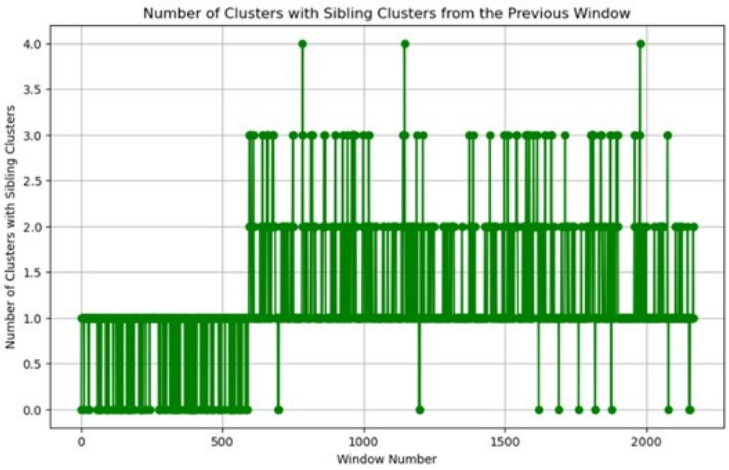
FIGURE D.18: Distribution of Residuals for Homing Pigeon dataset

The Poisson regression models generally performed well, with the expected lambda value indicating that the variance was roughly proportional to the mean, as anticipated in Poisson models. The analysis showed that distance_traveled consistently contributed to larger clusters across most classes, while trajectory_curvature and average_speed typically reduced cluster size. Variability in residuals and the alignment between actual and predicted values indicated that clustering dynamics differed across classes, with Class 4 proving the most challenging to model accurately. These findings suggest that while the model was effective overall, certain classes exhibited more complex clustering behaviours that may require further investigation.

Appendix E

Siblings Clusters Over Windows Dynamic Epsilon with Overlap





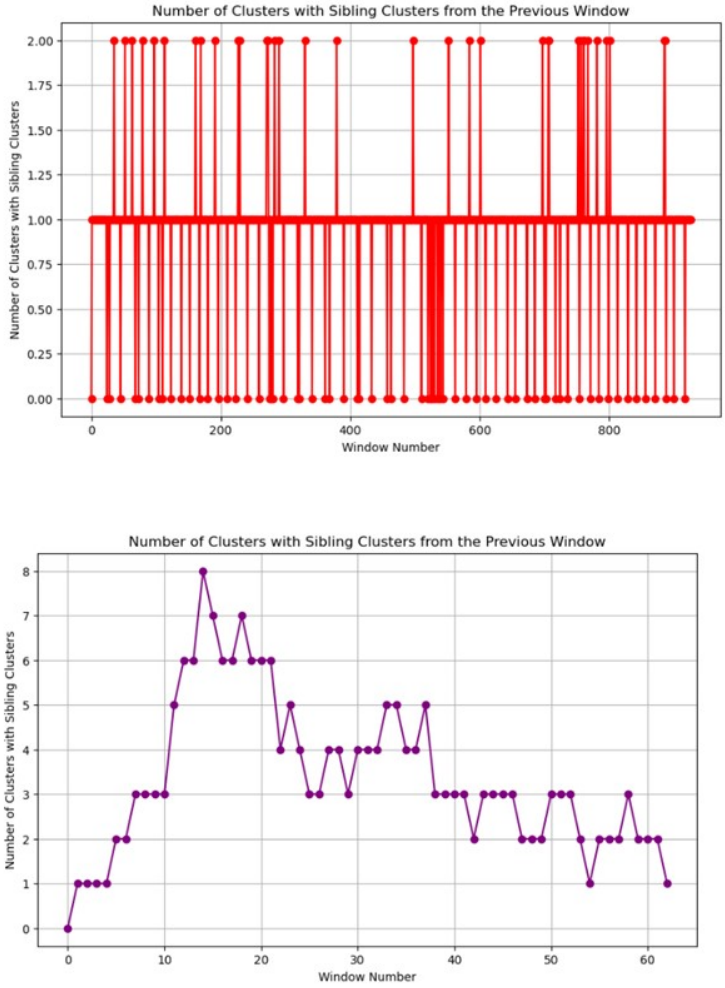


FIGURE E.1: Number of Clusters with Siblings for each Dataset (1-4 in order)

Appendix F

Sliding Window with Dynamic Epsilon Code Example

```
import pandas as pd
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score, \
    davies_bouldin_score, calinski_harabasz_score
from geopy.distance import geodesic
import itertools

# Load the dataset
file_path = 'Common_Kestrel_sample.csv'
df = pd.read_csv(file_path)

# Convert timestamp column to datetime
df['timestamp'] = pd.to_datetime(df['timestamp'], errors='coerce')

# Sort the dataframe by timestamp in ascending order
df_sorted = df.sort_values(by='timestamp').reset_index(drop=True)

def haversine(coord1, coord2):
    lon1, lat1, alt1 = coord1
    lon2, lat2, alt2 = coord2

    # Haversine distance
    r = 6371000
    phi1, phi2 = np.radians(lat1), np.radians(lat2)
    delta_phi = np.radians(lat2 - lat1)
    delta_lambda = np.radians(lon2 - lon1)
    a = np.sin(delta_phi / 2) ** 2 + np.cos(phi1) * np.cos(phi2) * \
        np.sin(delta_lambda / 2) ** 2
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1 - a))
    distance = r * c

    # Add altitude difference
    alt_diff = abs(alt1 - alt2)

    return np.sqrt(distance**2 + alt_diff**2)

def calculate_centroid(points):
    """
    Calculate the centroid of a cluster of points.
    """
    mn = np.mean(points, axis=0)
    return mn
```

```
def get_max_dist_in_cluster(cluster):
    max_dist_in_cluster = 0
    for p1, p2 in itertools.combinations(
        cluster[['location-long', 'location-lat', 'height-above-msl']].values,
        2):
        dist = haversine(p1, p2)
        max_dist_in_cluster = max(max_dist_in_cluster, dist)
    return max_dist_in_cluster

def get_min_dist_in_cluster(cluster):
    min_dist_in_cluster = 0
    for p1, p2 in itertools.combinations(
        cluster[['location-long', 'location-lat', 'height-above-msl']].values,
        2):
        dist = haversine(p1, p2)
        min_dist_in_cluster = min(min_dist_in_cluster, dist)
    return min_dist_in_cluster

def process_window(df_window, initial_eps=200, withDynamicEPS=True):
    curr_eps = initial_eps
    min_dist_centroids = np.inf
    min_dist_points = np.inf

    while True:
        db = DBSCAN(eps=curr_eps, min_samples=3, metric=lambda a, b: \
            haversine(a[:3], b[:3])).fit(
                df_window[['location-long', 'location-lat',
                           'height-above-msl']])
        labels = db.labels_
        unique_labels = set(labels)

        if len(unique_labels) == 1 and -1 in unique_labels:
            return pd.DataFrame(), curr_eps

        cluster_points = [df_window[labels == label] for label in \
            unique_labels if label != -1]
        centroids = [calculate_centroid(cluster[['location-long',
            'location-lat', 'height-above-msl']].values) for cluster in
            cluster_points]

        for i, j in itertools.combinations(range(len(centroids)), 2):
            dist = haversine(centroids[i], centroids[j])
            if dist < min_dist_centroids:
                min_dist_centroids = dist

        for cluster1, cluster2 in itertools.combinations(cluster_points, 2):
            for p1 in cluster1[['location-long', 'location-lat',
                                'height-above-msl']].values:
                for p2 in cluster2[['location-long', 'location-lat',
                                    'height-above-msl']].values:
                    dist = haversine(p1, p2)
                    if dist < min_dist_points:
                        min_dist_points = dist

        if min_dist_points <= min_dist_centroids or not withDynamicEPS:
            if min_dist_centroids != np.inf:
```

```
                curr_eps = min_dist_centroids
            result = []
            for cluster_label in unique_labels:
                cluster_df = df_window[labels == cluster_label]
                silhouette_avg = silhouette_score(df_window[[
                    'location-long', 'location-lat', 'height-above-msl']],
                    labels) if len(set(labels)) > 1 else -1
                davies_bouldin = davies_bouldin_score(df_window[[
                    'location-long', 'location-lat', 'height-above-msl']],
                    labels) if len(set(labels)) > 1 else -1
                calinski_harabasz = calinski_harabasz_score(df_window[[
                    'location-long', 'location-lat', 'height-above-msl']],
                    labels) if len(set(labels)) > 1 else -1
                max_dist = get_max_dist_in_cluster(cluster_df)
                min_dist = get_min_dist_in_cluster(cluster_df)
                first_ts = cluster_df['timestamp'].min()
                last_ts = cluster_df['timestamp'].max()
                cluster_size = len(cluster_df)
                points = cluster_df[['epoch', 'location-long',
                    'location-lat', 'height-above-msl']].values.tolist()

                result.append({
                    'cluster_label': cluster_label,
                    'silhouette_score': silhouette_avg,
                    'davies_bouldin_score': davies_bouldin,
                    'calinski_harabasz_score': calinski_harabasz,
                    'clusters_in_window': len(unique_labels),
                    'win_eps': curr_eps,
                    'min_dist_between_points': min_dist_points,
                    'min_dist_between_centroids': min_dist_centroids,
                    'max_dist_in_cluster': max_dist,
                    'min_dist_in_cluster': min_dist,
                    'first_timestamp': first_ts,
                    'last_timestamp': last_ts,
                    'win_number': None,
                    'cluster_size': cluster_size,
                    'points': points
                })
            return pd.DataFrame(result), curr_eps

        if withDynamicEPS:
            curr_eps *= 0.9

    return pd.DataFrame(result), curr_eps

# Initialize dataframe to hold the results
all_clusters = pd.DataFrame()

# Iterate through the data in sliding windows
window_number = 0
start_time = df_sorted['timestamp'].min()
end_time = df_sorted['timestamp'].max()
curr_eps = 200
# Define the sliding window and overlap in seconds
window_size = 30 * 60
step_size = 10 * 60
```

```
print("------------starting DBSCAN windows process:------------")

while start_time >= end_time:
    window_end_time = start_time + pd.Timedelta(seconds=window_size)
    window_df = df_sorted[(df_sorted['timestamp'] >= start_time) & \
        (df_sorted['timestamp'] < window_end_time)]

    if len(window_df) > 1:
        clusters_df, curr_eps = process_window(window_df, 200, True)
        clusters_df['win_number'] = window_number
        clusters_df['Selected EPS'] = curr_eps
        all_clusters = pd.concat([all_clusters, clusters_df],
                                 ignore_index=True)

    start_time = start_time + pd.Timedelta(seconds=step_size)
    window_number += 1
```