Logistic facility identification from longitudinal spatiotemporal data[☆].

Dirk J. De Beer[a,1], Johan W. Joubert[a,b,2,*]

[a] *Centre for Transport Development, Industrial & Systems Engineering, University of Pretoria, South Africa.*
[b] *KU Leuven, Institute for Mobility, Celestijnenlaan 300, Bus 2422, Leuven 3001, Belgium*

Abstract

Vehicle telemetry data is becoming more ubiquitous with increasingly sensorised vehicles, but making sense of the vehicles' purpose remains challenging without additional context. Clustering the vehicle activity data and identifying the underlying facilities where the activities occur reveals much insight, particularly for logistics planning. Unfortunately, current research typically only looks at a single point in time. This paper contributes by matching geospatial patterns, each representing a facility where trucks perform activities over multiple periods. The contribution is a necessary first step in studying how urban freight movement and its underlying inter-firm networks of connectivity change over time. We demonstrate how to overcome three challenges. Firstly, the complexity of identifying facilities from non-regular geometric polygons. Secondly, the challenge associated with the scale of comparing more than 200 000 facilities on a month-to-month basis over a multi-year period. Finally, overcoming the computational challenge of the workflow and getting the required performance on a consumer-grade laptop. The paper evaluates various machine learning algorithms, highlighting a support vector machine that outperforms more popular deep learning and neural network alternatives, with a mean average accuracy of 96.9%.

Keywords: Geospatial feature engineering, machine learning, shape comparison, multiperiod lineages, geospatial clustering

1. Introduction

As cities keep expanding, so does the density of consumption and waste produced. Heavy goods vehicles are best suited for both the supply of goods and services and the collection and removal of waste. While transporting goods and service personnel, these heavy goods vehicles frequently stop to perform activities. The locations where these vehicles perform stops suggest some economic or value-adding activity.

Furthermore, the movement of vehicles between facilities is a helpful proxy for business relations between the associated firms (Joubert & Axhausen, 2013). Therefore, identifying these

[☆]This paper has been published and should be cited as follows: De Beer, D.J., Joubert, J.W. (2024). Logistic facility identification from longitudinal spatiotemporal data. *Computers, Environment and Urban Systems*, **114**, 102182. DOI: 10.1016/j.compenvurbsys.2024.102182.

[*]Corresponding author

Email addresses: de.beer.dj@gmail.com (Dirk J. De Beer), johan.joubert@kuleuven.be (Johan W. Joubert)

[1]Tel: +27 82 851 3419

[2]Tel: +32 472 57 62 64

facility locations accurately is vital to better understanding the firms and their relationships with one another. Gathering detailed information about commercial vehicles' movement and activity schedules takes a lot of work, especially at the urban scale.

Using ubiquitous large-scale global navigation satellite system (GNSS) telemetry data, it is possible to derive both facility locations, a proxy for actual firm locations, and transport relationships between firms. It is only a proxy because the actual locations where vehicles stop differ each time, and the limited accuracy of the GNSS means the actual stop position is also not precise. The reader should note that with *stops*, we imply a trip-end (Sharman & Roorda, 2011) that typically involves the vehicle's ignition switched off, and not stops associated during travel, for example, at traffic intersections. To overcome the challenge of imprecise data, we can identify facilities by clustering vehicle stop locations for a given period. In the context of this paper, each single period is referred to as an *epoch* in the remainder of the paper. Each facility is then represented as a geospatial polygon, a concave hull of the associated vehicle stop locations.

But activities and freight relationships change over time. Since the clustering results are used as a proxy for facility location, the same facility might be represented by a differently shaped polygon in different epochs. An outstanding issue is the accurate identification of facilities between subsequent epochs, referred to as *inter-epoch* changes. This translates into the problem of large-scale matching of complex and irregular geospatial polygons in epoch t to those in epochs $t+1$, $t+2$, etcetera. This challenge is depicted in Figure 1. When density-based clustering is performed

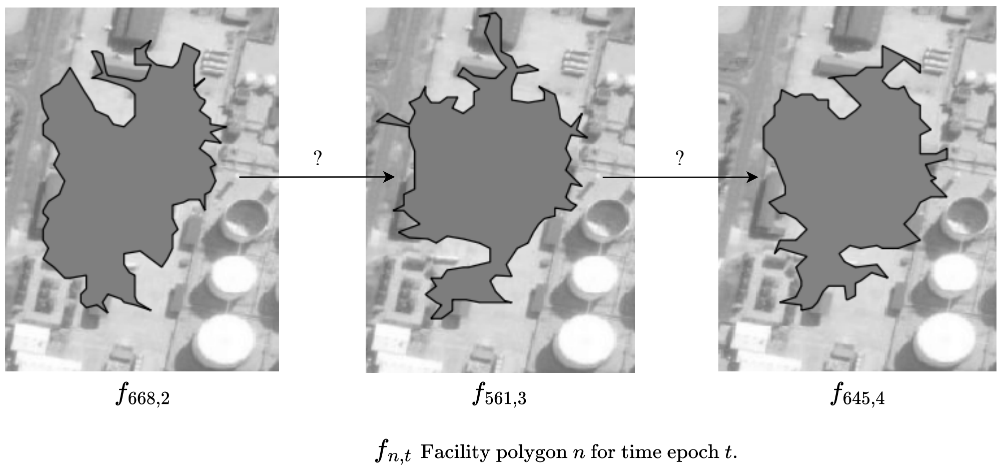


Figure 1: Example of inter-epoch facility polygons from sample data. Clustered telemetry data results in different polygons in each epoch.

to identify the facilities from the telemetry data, each epoch's data is clustered independently. After the density-based clustering, a unique facility number is assigned arbitrarily to each cluster as an identifier. As a result, the facility numbers in one epoch have no relationship to the facility numbers in any other epoch; if they have, it is entirely coincidental. Concerning Figure 1, one can ask, *"Is the polygon representing facility 668 in epoch 2 the same facility as the polygon representing facility 561 in epoch 3?"* Similarly, *"Is the polygon representing facility 561 in epoch 3 the same facility as the polygon representing facility 645 in epoch 4?"* While this might seem trivial to the human mind to resolve, automating the task on a large-scale data set using machine learning is more challenging.

If we can track the facilities over time, we can also track how their connections with other facilities, a proxy for business intensity, change over time. Accurate inter-epoch facility identification allows for the creation of a linked time series of facility networks with one network per epoch, enabling the investigation of:

- the *birth* of new facilities (first occurrence in time series);
- the *death* of facilities (last presence);
- the merging of smaller facilities into larger facilities;
- the split of larger facilities into smaller facilities;
- the age of facilities (firm longevity); and
- the facility relationships as they change over time (inter-firm connectivity).

In this work, we report on resolving the inter-epoch facility identification question. We describe the use of a metric space approach to geospatial feature engineering. This extracts machine learning-friendly features from the geospatial polygons in each epoch. Employing the engineered features, the paper evaluates several machine learning models for accuracy and potential use in matching intra-epoch polygons. The most accurate machine learning model, a support vector machine (SVM), is then deployed to identify intra-epoch facilities in a sample dataset. From this, we derived some preliminary insights on applying this approach to study the evolution of facilities over time.

This paper is structured as follows. Section 2 reviews related and ongoing work on facility identification, geospatial polygon similarity, and machine learning. Section 3 describes the data, feature engineering methodology, and data tagging for machine learning. The section also introduces the City of Cape Town study area. Section 4 reports on the experimental results. It also addresses a broader context of ongoing research and briefly touches on potential future work on facility lineage research. The paper concludes with some final remarks and future work in Section 5.

2. Literature review

This review addresses four particular themes. Firstly, identifying facility locations is an essential aspect of understanding logistic movements. Secondly, the review positions the geospatial shape matching within the broader body of knowledge. Thirdly, the review looks at feature engineering as an essential step in machine learning for shape matching. Finally, we also briefly touch on computational complexity.

2.1. Identifying facility locations

Establishing where commercial or freight vehicles perform their activities remains a challenge. Conducting detailed travel diaries through surveys is expensive and labour-intensive. In comparing different methodologies to detect and quantify logistics sprawl — the migration of facilities dealing with freight outwards, away from the economic centres — Trent & Joubert (2022) reflect on the challenges in establishing a unified approach in what is considered a *facility*. In more developed contexts, detailed land-use data is available, but these data typically fail to reflect the intensity of freight movement. A representative freight-specific survey is available only on rare occasions, like in Japan's Tokyo Metropolitan Freight Surveys (2003 and 2013). To protect privacy, the data is then disseminated at an aggregate level. Researchers have turned to another source of data: location traces collected through GNSS or other location service providers. In the case of commercial and freight vehicles, the telemetry data is frequently collected through onboard units. Such spatial data is becoming ubiquitous but brings its own challenges because the spatial breadcrumbsit leaves tell us something about their movement between facilities and little to nothing about the facilities where the actual activities occur. This section deals with the branch of literature concerned with using vehicular movement data to understand the activities and associated locations where freight vehicles conduct their business.

The use of density-based clustering to derive location information from geospatial point data is well established (Ansari et al., 2021; Shi & Pun-Cheng, 2019; Malzer & Baum, 2020; Nurmi, 2009). This allows for identifying places of interest such as a coffee shop from, e.g., a collection of geospatially co-located payment transaction data points (Zhou et al., 2004). From a transport perspective, clustering trip stop locations from transport geospatial data enable the identification of facilities and potential places of transport-related economic activity (Joubert & Axhausen, 2011). Sharman & Roorda (2011) and Cich et al. (2016) both extract trips and stops from continuous GNSS traces. Without additional data, vehicle stop locations suggest being involved in an activity, such as loading or unloading, which has economic value. Facilities result from activity clustering for a given period (epoch). Linking these facilities to one another allows for creating a network structure describing the transport relationships between facilities, albeit only for that particular epoch (Joubert & Axhausen, 2013; Joubert & Meintjes, 2016). This facility network allows for using complex network analysis in studying the facility relationship structure (Háznagy et al., 2015; Newman, 2018). For instance, a facility's connectivity to other facilities might be used as a proxy for its importance in the transport network: the higher the connectivity, the more important the facility. It is thus clear that there is a significant dependence on accurate underlying facility identification or accurate activity clustering. Recent work addressed this aspect through large-scale evolutionary programming to optimise activity clustering accuracy (De Beer & Joubert, 2022). As a result, location identification can be efficiently automated.

Given a sequence of facility networks, with one network per epoch, what remains outstanding is the accurate linking of facility polygons between networks in subsequent epochs. This translates into the accurate large-scale matching or linking of geospatial polygons *between epochs*, with similarity influenced by both polygon shape and location (as illustrated in Figure 1). Solving this will allow us to answer a question like *"How does the movement of urban freight vehicles change over time?"*

2.2. Shape matching

The ability to identify and compare complex shapes informs several fields. In copyright protection, identifying features are generated from digital images and compared to potential copyright infringements of said images (Daoui et al., 2021). In archaeology, pottery shapes and tool morphology are classified and compared (Wang et al., 2018; Cardillo, 2010). Plant morphology is used in botany to compare cultivars (Kupe et al., 2021). Medicine compares and categorises skeletal structures such as human mandibles (Schmittbuhl et al., 2001). Astronomy is interested in the categorisation and comparison of distant galaxies based on galaxy shape, sometimes in an invariant manner to shape rotation, scale, and translation (Cecotti, 2018; Tramacere et al., 2016). In various human behaviour-related fields, posture analysis is used, for example, to identify sign language symbols (Espejel-Cabrera et al., 2021; Fernando & Wijayanayake, 2020) and dance postures (Anami & Bhandage, 2019). Computer vision processing, underscoring recent advances in visual machine learning, requires detecting, identifying and comparing shapes and objects in visual scenes (Peters, 2017).

In geospatial transport analysis, the optimisation of intra-epoch facility identification compares polygons manually defined by humans to polygons resulting from automated clustering (De Beer & Joubert, 2022). While manual human input and insight may be considered repeatable (Joubert & Meintjes, 2015), they are slow and expensive. In response, efforts are invested in automating the matching process. Hausdorff distance measures are used in logistics facility analysis to compare and identify logistics facility polygons over time (Trent et al., 2020). Geospatial geometry classification employs machine learning to identify building classes based on the comparison between vector polygons (Veer et al., 2018) and shape similarity to classify aerial entities in geographical vector data (Fu et al., 2018).

2.3. *Feature engineering*

Most cases reported in the previous subsection use some form of feature extraction or engineering (Domingos, 2012; LeCun et al., 2015). Feature extraction is also called *location-encoding* in geospatial processing (Mai et al., 2022). Feature engineering transforms complex shape information into smaller sets of, sometimes fixed, well-defined features that can be used to classify and compare shapes. Well-engineered features also render the classification and comparison problem more friendly to machine learning approaches (Zheng & Casari, 2018). Three types of features are common in the literature:

- geometric features;
- elliptical Fourier descriptors (EFDs) (Kuhl & Giardina, 1982); and
- Hu moments (Hu, 1962).

De Beer & Joubert (2022) and Veer et al. (2018) extract intuitive geometric features such as polygon circumference, area, centroid location, and the number of polygon vertices. The prior work utilises these simplistic features to calculate the *delta* (difference) between feature values of two polygons, such as the distance between centroids and the difference between circumferences. These deltas are then input for penalty calculation as a similarity measure. The latter work uses the basic features as an additional input into machine learning models.

In their work, Veer et al. (2018) and Wang et al. (2018) extract EFDs. Along with geometric features, these are used as input into machine learning or for further analysis. EFD analysis disassembles a complex closed curve into a sequence of sine or cosine functions of increasing frequencies (Godefroy et al., 2012). For each $i \in \{1..n\}$, with n the number of harmonics, or level of required accuracy, four coefficients, a_i, b_i, c_i and d_i are derived that affect the sine or cosine functions. These coefficients are called EFDs. Summing the sine or cosine functions leads to a complex ellipse approximating the original closed curve. As such, EFDs describe the original closed curve and are used in shape comparisons (refer to Figure 2). When one deals with complex polygons, as in our paper, the number of EFDs can become prohibitively large.

In the classification and comparison of galaxies (Tramacere et al., 2016) and aerial entities in geographic vector data (Fu et al., 2018), the authors of both contributions extract Hu moments (Hu, 1962) for use as features. Briefly, Hu moments are geometric moments invariant to translation, scaling and rotation. Regardless of the size of the geometric shape under investigation, there are always only seven Hu moments for the shape. For example, in Figure 3, the Hu moments of a square image for the letter 'K' and those of another image for the same letter but which is rotated,

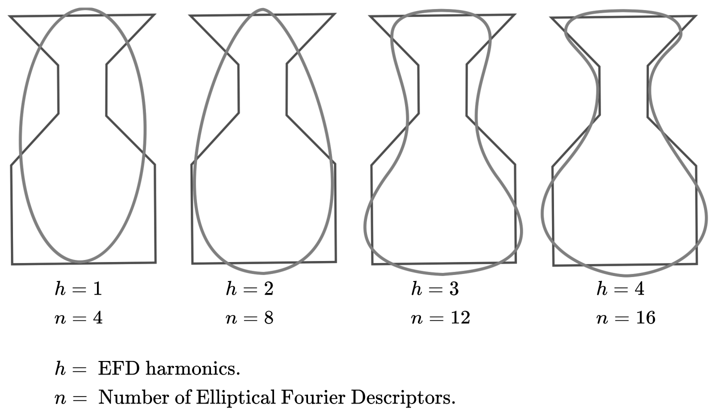


Figure 2: Elliptic Fourier reconstruction of a complex polygon (Adapted from Kuhl & Giardina (1982) and Veer et al. (2018)).

scaled and shifted to the right-hand side of the image square, will be very similar, approaching equality. But the Hu moments for an image of the letter 'T' would be significantly different (orders of magnitude) from the Hu moments for the image of the letter 'K'. Hu moments are therefore used to compare and classify images, polygons and contours.

A well-established practice for using Hu moments with vector images is to rasterise the vector image into a two-dimensional raster image up to a required level of accuracy. In support of their argument for using EFDs, Veer et al. (2018) state that this rasterisation approach leads to a loss of information. In their choice against using EFDs in favour of Hu moments, the present authors will address this concern later in this paper. For a detailed description of Hu moments, refer to Appendix A.1.

Many authors combine one or more of the different types of features. For example, Tramacere et al. (2016) utilise Hu moments, geometric features and morphological features while Veer et al. (2018) employs geometric features with EFDs as part of their analysis.

Once feature engineering is completed, most authors utilise some form of machine learning for the classification task. A detailed description of all these machine-learning approaches lies outside the scope of this paper. Veer et al. (2018) evaluate several shallow and deep learning models across three geometry classification tasks:

1. Predicting the number of inhabitants compared the national median based on neighbourhood geometry.
2. Using a building's contour polygon to predict a building class.
3. Predicting an archaeological feature type using its geometry.

The models evaluated were:

- k-nearest neighbour classifier (kNN);
- Logistic regression;
- Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel;

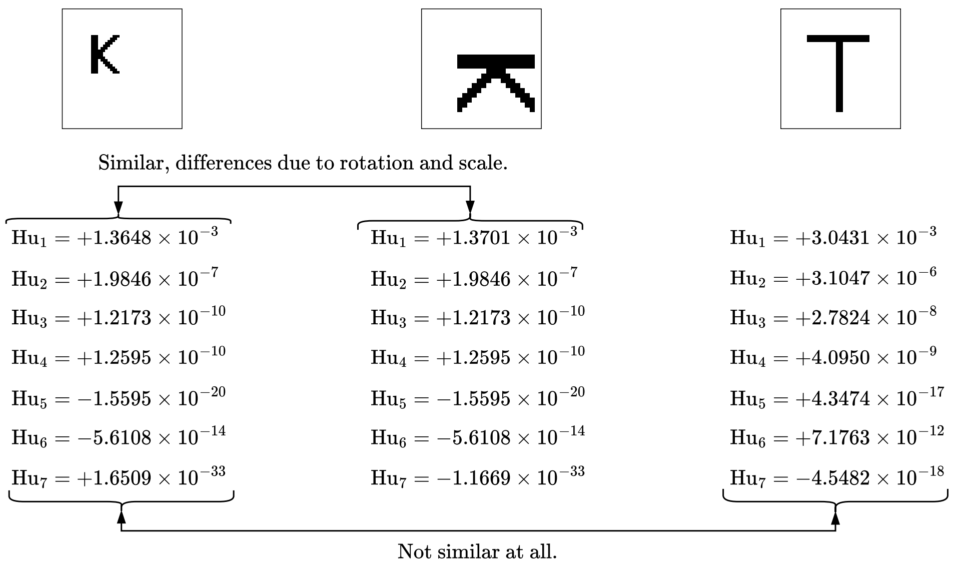

Similar, differences due to rotation and scale.
$\mathrm{Hu}_1 = +1.3648 \times 10^{-3}$
$\mathrm{Hu}_2 = +1.9846 \times 10^{-7}$
$\mathrm{Hu}_3 = +1.2173 \times 10^{-10}$
$\mathrm{Hu}_4 = +1.2595 \times 10^{-10}$
$\mathrm{Hu}_5 = -1.5595 \times 10^{-20}$
$\mathrm{Hu}_6 = -5.6108 \times 10^{-14}$
$\mathrm{Hu}_7 = +1.6509 \times 10^{-33}$
$\mathrm{Hu}_1 = +1.3701 \times 10^{-3}$
$\mathrm{Hu}_2 = +1.9846 \times 10^{-7}$
$\mathrm{Hu}_3 = +1.2173 \times 10^{-10}$
$\mathrm{Hu}_4 = +1.2595 \times 10^{-10}$
$\mathrm{Hu}_5 = -1.5595 \times 10^{-20}$
$\mathrm{Hu}_6 = -5.6108 \times 10^{-14}$
$\mathrm{Hu}_7 = -1.1669 \times 10^{-33}$
$\mathrm{Hu}_1 = +3.0431 \times 10^{-3}$
$\mathrm{Hu}_2 = +3.1047 \times 10^{-6}$
$\mathrm{Hu}_3 = +2.7824 \times 10^{-8}$
$\mathrm{Hu}_4 = +4.0950 \times 10^{-9}$
$\mathrm{Hu}_5 = +4.3474 \times 10^{-17}$
$\mathrm{Hu}_6 = +7.1763 \times 10^{-12}$
$\mathrm{Hu}_7 = -4.5482 \times 10^{-18}$
Not similar at all.

Figure 3: Hu moments for three different images. (Source: Authors.)

- Decision tree classifier;
- 7-layer Convolutional neural network (CNN); and
- Long short term memory (LSTM) Recurrent neural network (RNN).

The reported accuracy on the three tasks across all models evaluated falls within $[0.608, 0.683]$, $[0.328, 0.408]$ and $[0.555, 0.624]$, respectively. Tramacere et al. (2016) utilise two ensemble classification models in their galaxy morphology classification task:

- Random forest; and
- Gradient boosting.

2.4. A brief look at computing complexity

Processing complexity with big data is especially acute in the case of geospatial data sets. This due to the need to process data in the context of space and time (Li, 2020), the need to address complex dataset architectures (Zhao et al., 2016), the interaction of data (Poux et al., 2016) and potential data segmentation (Ruan & Liu, 2020). A matter made worse when having to compute machine learning models from the data, in many instance requiring special procession equipment such as GPUs (Teri et al., 2022). To overcome this complexity, a well established approach is through sampling, to experiment and evaluate model applicability, the embedding of spatial properties, apply potential parameter optimisation and implementing any additional algorithm enhancements deemed relevant (Du et al., 2020; De Beer & Joubert, 2022; Mai et al., 2022; Zheng & Casari, 2018). After which the models are then employed in a scaled-up production environment.

3. Methodology

3.1. Data, study area and initial preparation

The primary data source for this paper is derived from a comprehensive commercial vehicle trip telemetry dataset for South Africa from January 2010 to May 2014. More than 10 000 commercial vehicles subscribing to the *Digicore Fleet Management* services, using onboard tracking devices, travelled across South and Southern Africa. As described by Joubert & Axhausen (2013), the telemetry dataset was processed to derive a total of 50 862 385 activities which were then optimally clustered (De Beer & Joubert, 2022) into 504 198 unique facilities. The data's potential selection bias and representativeness are described in more detail in Joubert & Axhausen (2011). The methodology presented in this paper remains valid across different scales of data representation.

Switching off a vehicle indicated the start of an activity and vice versa. Consecutive activities were grouped into activity chains. Facilities were extracted by clustering the activities into facility clusters based on density and proximity.

Aggregating the trips of all vehicles over time results in a network of connectivity.

In previous research, facility clustering was done on a per-month basis. In the context of this current paper, each month is referred to as an *epoch*. This resulted in 53 networks, one per epoch, with descriptive statistics over all epochs provided in Table 1. The resulting sequence of facility networks represents a time series of facility transport relationships. Each facility network is recorded in two datasets: facilities (i.e. vertices) and links between facilities (i.e. edges). Of primary concern for the work reported in this paper were the facilities datasets, which provided the following data items:

Table 1: Descriptive statistics for facility clustering and network building process ($N_{\text{epochs}} = 53$).

Item	μ	σ	**min**	q_{25}	q_{50}	q_{75}	**max**
Activities	959 667.6	180 300.5	195 389	883 008	990 877	1 085 937	1 215 748
Facilities	9 512.2	1 441.7	2 982	8 968	9 813	10 395	11 515

- `Id` : The unique (intra-epoch) id of the facility assigned during clustering;
- `Birth` : The epoch, $e \in \{0, 1, \ldots, 52\}$, in which the facility originated as a result of clustering;
- `Long`, `Lat` : The longitude and latitude polygon centroid coordinates;
- `Count` : The number of activities clustered together;
- `Circum` : The facility polygon circumference in m;
- `Area` : The facility polygon area in m^2;
- `NumPts` : The number of polygon points (complexity of polygon); and
- `Polygon` : The sequence of longitude and latitude points making up the facility polygon.

To obtain an id for facilities that is unique across epochs, we combined the `Id` field with the `Birth` field. For example $f_{123,9}$ is facility 123 that originated in epoch 9.

Along the time dimension, the dataset was reduced to include only the 5 epochs, from 2 (March 2010) to 6 (July 2010), both inclusive. The choice of months is arbitrary and reflects a period that is considered an average business period. The extended Cape Town area, South Africa, was selected along the spatial dimension. All facilities not located within this study area were filtered out. The remaining facility data for the different facility networks were then merged into a single facility dataset with the above fields, with `Birth` used to determine the epoch each facility belongs to.

To facilitate sampling, we decided to bin the data spatially. Utilising the open source H3 library from Uber (Uber Engineering Team, 2022), the study area was divided into hexagonal bins of H3 resolution 8. This resulted in 6 997 bins with a reported average edge length of 461.4 m, the closest resolution to an intuitive edge length of 500 m. Using hexagons is practical since circles cannot be used to tessellate the surface without gaps or overlaps. Hexagonal binning allows us to tile the surface with minimum bin perimeter/area ratios as close to that provided by circles whilst reducing bias resulting from edge effects (Hales, 2001; Birch et al., 2007). This is depicted in Figure 4.

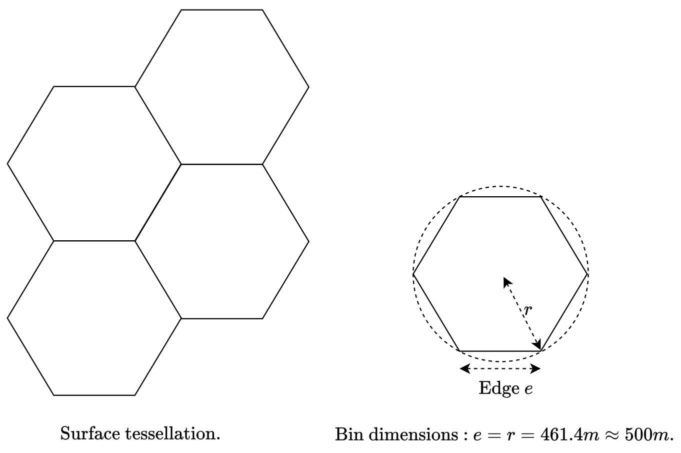


Figure 4: Tessellation of the surface and dimensions of hexagon bins. (Source: Authors)

Facilities were allocated to the hexagonal bins based on whether the facility's centroid fell within a bin.

We applied a kernel density estimate (KDE) of facilities in the study area. Informed by the selection bias present in the original vehicle data[3], our reasoning for this was that KDE would enable better estimation of the actual facility density. Next, we applied two-step sampling. As the first step, using uniform sampling weighted by the estimated facility density, we obtained 20 initial bins. We ordered the highest to lowest density bins and selected the top 10 bins with the highest density, $\mathbf{B} = \{b_0, b_1, ..., b_9\}$. This two-step approach balanced fair-weighted sampling with a slight bias towards increased facility density. This was done in anticipation of the planned application of machine learning later to maximise the number of training cases available in the sample. Refer to Tables 2 and 3 for descriptive statistics of $\mathbf{B}$.

Figure 5 shows the study area with the underlying density map, superimposed bins and 10 selected sample bins. Figure 6 provides an enlarged view of sample bin b_1 with the outlines of all

[3]Although acknowledging sampling measures taken to reduce bias in collating the original longitudinal vehicle telemetry dataset, we remain cognisant of the single source of the data and the potential for selection bias this might represent. For this paper, where the focus is matching polygonal shapes over time, the bias has no effect.

Table 2: Descriptive statistics for sampled facility counts per epoch (Epoch $\in [2, 6]$).

Item	***Epoch***$_2$	***Epoch***$_3$	***Epoch***$_4$	***Epoch***$_5$	***Epoch***$_6$	**Total**
Facilities per epoch	168	140	151	155	173	787

Table 3: Descriptive statistics for sampled pan-epoch facility counts over sampled bins ($N_{\text{bins}} = 10$).

Item	μ	σ	**min**	q_{25}	q_{50}	q_{75}	**max**
Facilities per bin	78.7	24.9	34.0	62.0	80.0	95.3	117.0

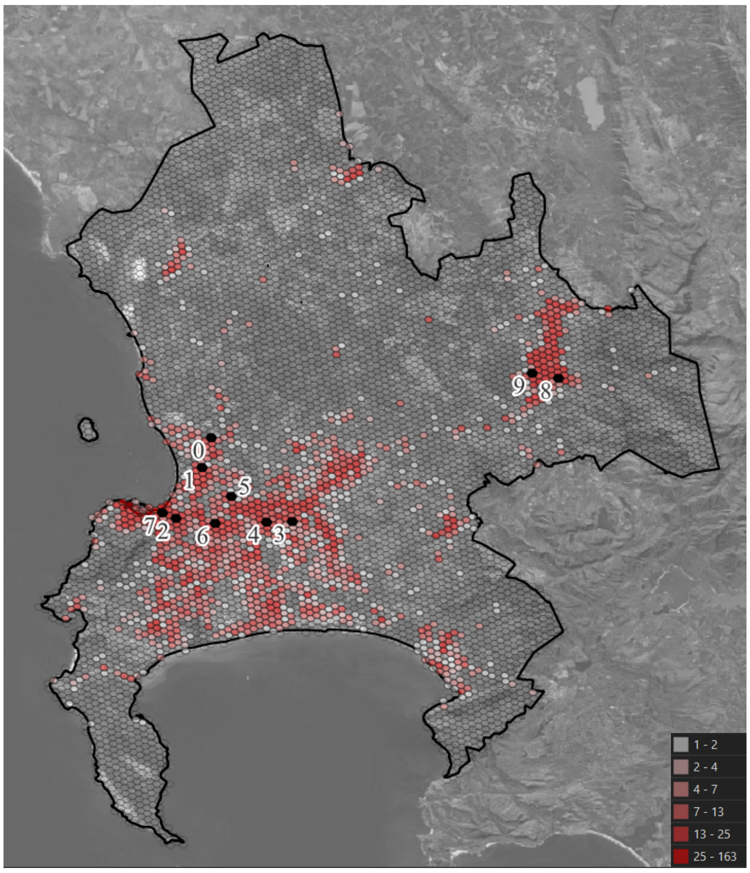


Figure 5: Extended Cape Town study area with density map, superimposed bins and sample bins. (Source: Authors)

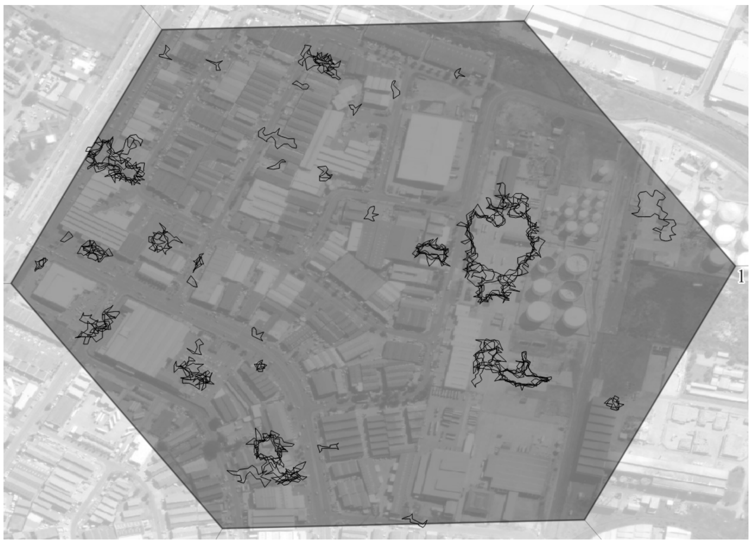


Figure 6: Enlarged view of sample bin b_1 with facility polygon outlines for epochs 2 to 6. (Source: Authors)

facility polygons for epochs 2 to 6, stacked on top of one another to illustrate the potential link between facilities over multiple epochs.

3.2. Data tagging

As per prior work presented in Section 2, we opted to utilise supervised machine learning to assist with the inter-epoch comparison of facility polygons. For this, we required accurately tagged data to train and evaluate machine learning models. To tag the data, we compared all facilities in an epoch to those in a previous epoch to find polygon pairs representing the same facility. Each comparison between two facility polygons, $f_{x,t}$ in epoch t and $f_{y,t-1}$ in epoch $t-1$, is referred to as a training *case*. For $t \in [3, 6]$, the range of cases covers all inter-epoch comparisons for epochs 2 to 6. Training cases are either *positive* or *negative*. In a positive case, $f_{x,t}$ represents the same facility in the current epoch t as that described by facility $f_{y,t-1}$ in the previous epoch $t-1$. In this instance, we refer to $f_{y,t-1}$ as the *parent* of $f_{x,t}$ and, inversely, to $f_{x,t}$ as the *child* of $f_{y,t-1}$. In a negative case, $f_{x,t}$ in the current epoch t is deemed to not represent the same facility as $f_{y,t-1}$ in the previous epoch $t-1$.

To obtain cases, we only compared facilities within a sample hexagon bin, $b_i \in \mathbf{B}$, to facilities in the same bin over all the sample bins. The average number of facilities in each bin over 5 epochs is $\mu = 78.8$ (Table 3). For any two epochs, t and $t-1$, the average number of per-bin cases is $78.8 \times 78.8 \approx 6\,209$. Over 4 inter-epoch comparisons for the 5 epochs and 10 sample bins, this represents an average total of 248 360 potential cases. To reduce the number of possible cases, we investigated the distribution of facility polygon size to obtain a sensible cut-off distance for inclusion. Utilising minimal rotated bounding boxes (Chan & Tan, 2001) for the facility polygons and the maximum axis lengths of these bounding boxes, we experimentally obtained a cut-off of $150m$. Excluding all cases where the closest points on the polygon edges of the two facilities are further apart than this cut-off distance resulted in a total of 2 122 cases.

To then tag each case as positive or negative, the QGIS (QGIS Development Team, 2019) open-source geospatial processing platform was used. An expert manually tagged all cases, comparing the two geospatial facilities in QGIS for each case (similar to what was illustrated in Figure 1).

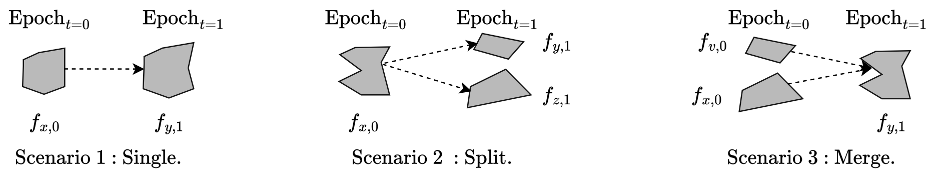


Figure 7: Different scenarios for positive cases. Each grey polygon, denoted with $f_{n,t}$, refers to facility n in epoch t. (Source: Authors)

The use of human experts to tag geospatial data is well established (Zhou et al., 2004; De Beer & Joubert, 2022). Although manual tagging of data remains subjective (Bruce & Wiebe, 1999), repeatability and reproducibility of the approach have been illustrated by Joubert & Meintjes (2015).

The above tagging approach resulted in 460 positive and 1 662 negative cases. Per example, a positive case might be where facility f_{123} in epoch 5 was identified as being the same facility as f_{456} in epoch 4. A negative case for the same facility f_{123} in epoch 5 might be where it was deemed by the expert not to be the same facility as f_{890} also in epoch 4.

Although the tagging approach is based on a case basis, comparing only a single facility from epoch t to a single facility in epoch $t-1$, doing so over all the facilities in both epochs, this approach does capture a range of inter-epoch facility dynamics. Figure 7 highlights combinations of the following three scenarios for positive cases that may (and have) arisen as part of tagging.

In scenario 1, facility $f_{y,1}$ in epoch $t=1$ could only be matched to a single facility $f_{x,0}$ in epoch $t=0$. As such, this results in a single positive case. In scenario 2, both facility $f_{y,1}$ and $f_{z,1}$ in epoch $t=1$ could be matched with a facility $f_{x,0}$ in epoch $t=0$. This represents the case where a facility polygon seemingly splits into two (or more) smaller facilities in the next epoch. This will result in *two* (or more) positive cases, one for each “split line”. Scenario 3 is the inverse of scenario 2, where two (or more) facilities merge into a single facility. This scenario also results in *two* (or more) positive cases, one for each “merge line”.

These scenarios later play a role in deriving the *lineages* for facilities.

3.3. Feature engineering

To prepare our tagged data set for machine learning, we conducted feature engineering (Zheng & Casari, 2018) and, in line with the earlier literature review, we use geometric and Hu moments as feature types. We further distinguish between two groups of features, *primary features* and *delta features*. Primary features are derived utilising one or more of the three main types of features above. Delta features are derived via further processing of primary features.

For *primary features*, we settled on combining geometric features and Hu moments. The former is chosen to build on our ongoing work with geometric features. The latter given reported levels of accuracy achieved (Tramacere et al., 2016) and that there are always only 7 Hu moments for each polygon, regardless of complexity. This allows for more deterministic computational needs and fixed *a priori* complexity in comparing two facility polygons.

We decided against using EFDs due to the potential varying and large number of EFDs required to adequately fit some of the facility polygons in our population. Some polygons in our population require harmonics of 8 and up, resulting in $8 \times 4 = 32$ features per polygon.

Extending concepts from our work on optimisation (De Beer & Joubert, 2022), we also derive *delta features* from the *primary features*. We represent feature i, of polygon p by $\phi_{i,p}$. Given two

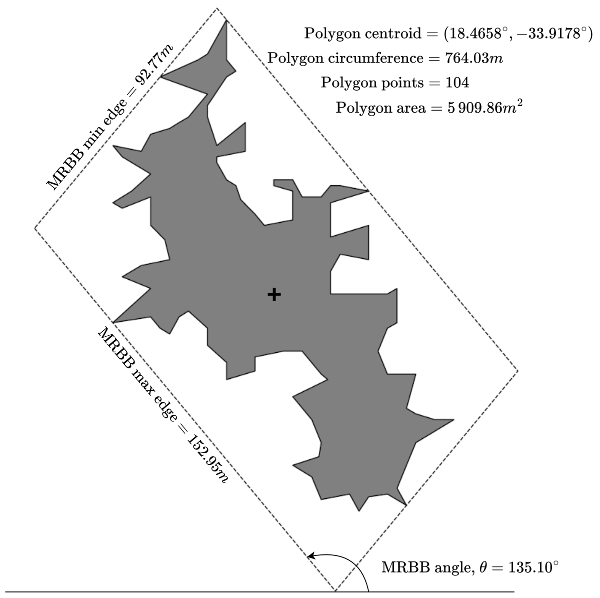

Polygon centroid = (18.4658°, −33.9178°)
Polygon circumference = 764.03m
Polygon points = 104
Polygon area = 5 909.86m²
MRBB min edge = 92.77m
MRBB max edge = 152.95m
MRBB angle, θ = 135.10°

Figure 8: Features for facility $f_{1\,094,5}$. (Source: Authors)

polygons, x and y, we experimentally settled on calculating delta features as:

$$\delta_{i,(x,y)} = |\phi_{i,x} - \phi_{i,y}| \geq 0 \tag{1}$$

Intuitively, delta features represents the *distance between* two feature values, such that $\sum_i \delta_{i,(x,y)}$ indicates the *overall difference* between x and y, and where $\sum_i \delta_{i,(x,y)} = 0 \implies x = y$.

We first obtained the primary features for each unique facility polygon in the tagged dataset. The following geometric features were either calculated or obtained from the input facility network data:

- centroid latitude;
- centroid longitude;
- polygon area (m^2);
- polygon circumference (m^2);
- polygon points count;
- minimum rotated bounding box (MRBB) angle ($\theta \in [0°, 180°]$);
- MRBB minimum edge length (m^2); and
- MRBB maximum edge length (m^2).

Of the above, the angle of the minimum rotated bounding box (Chan & Tan, 2001) requires clarification. As implemented, this is the angle, in degrees, between the horizontal line and the edge with the maximum length of the minimum rotated bounding box. In Figure 8, we illustrate the geometric primary features obtained for facility $f_{1\,094,5}$.

We then calculated the 7 Hu moments for each polygon. To do this, we first had to rasterise the vector polygon for each facility. Rasterisation (Popescu & Rosen, 2006) is a well-established process

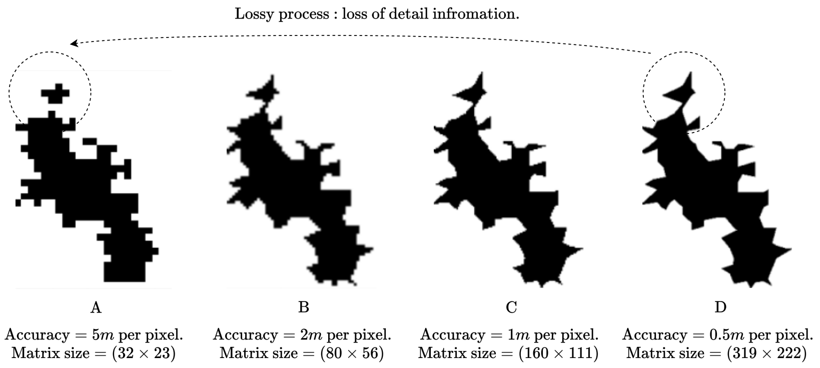

Lossy process : loss of detail infromation.
A
Accuracy = $5m$ per pixel.
Matrix size = (32 × 23)
B
Accuracy = $2m$ per pixel.
Matrix size = (80 × 56)
C
Accuracy = $1m$ per pixel.
Matrix size = (160 × 111)
D
Accuracy = $0.5m$ per pixel.
Matrix size = (319 × 222)

Figure 9: Visually scaled rasterisation at different accuracy levels for facility $f_{1\,094,5}$.
(Source: Authors using OpenCV (OpenCV team, 2022)).

whereby a vector image is converted into a two-dimensional raster image of required accuracy. For rasterisation, we used the highly optimised and well-established OpenCV library (OpenCV team, 2022). Veer et al. (2018) state that rasterisation is a lossy process in that there is a reduction in information and, thus, accuracy when converting a vector to a raster. This is indeed true when using a coarse resolution for the raster. We see this instead as an engineering trade-off between preferred accuracy and ease of use. Figure 9 illustrates this for facility $f_{1\,094,5}$ (visually scaled). The lossy nature of the process is illustrated when comparing raster image A to raster image D. At lower accuracy, raster image A depicts two polygons, which is not the case when looking at the higher accuracy raster image D. This represents a loss in information. Experimentally, we settled on an accuracy of 1m per pixel for computational purposes. Our facility polygons are only black-and-white, with 0 or 1 per pixel. This allowed us to use bit arrays to store rasterisation results. Each of our polygons fits into a maximum area of 228 pixels × 228 pixels = 51 984 pixels. At 1 bit per pixel, this represents a maximum storage requirement of 6.35 KB per polygon, or 4.88 MB for all 787 polygons.

The set of 15 primary features per polygon p is thus represented as $\phi_{i,p}, i \in [1, 15]$ (where the square brackets imply a range of values, both extremes included). For geometric features, $i \in [1, 8]$ and for Hu moments, $i \in [9, 15]$.

Using the primary features, we subsequently calculated the *delta features* for each case. Given a case that compares polygon x and y, the delta features for that case, $\delta_{j,(x,y)}, j \in [1, 14]$, were calculated using equation 1 with $\phi_{i,x}$ and $\phi_{i,y}$ as input. Instead of calculating a delta feature between the centroid longitudes and separately the centroid latitudes of two polygons, we calculated the actual geographic distance between the two centroids in m. This was done due to the more intuitive concept of geographic distance when compared to separate delta values for the longitudes and latitudes. As such, there are only 14 delta features. We appreciate that this geographic distance calculation deviates somewhat from the other feature delta calculations. For software and data structure design purposes, we decided not to treat it differently and simply use the calculated feature as feature 14 with the same indexing approach.

On a per-case basis comparing two polygons, we thus ended up with 15 × 2 = 30 primary features and 14 delta features, for a total of 44 features to consider during the next research stage.

Table 4: ML models selected for evaluation ($N_{\text{models}} = 11$).

Model	**Id**	**Non-Default Configuration**
Convolutionary Neural Network	CNN	100 training epochs & Figure A.18
Fully Connected Neural Network	FNN	100 training epochs & Figure A.18
Logistic Regression	LOR	Max iterations 200
Nearest Centroid	NEC	-
Linear Discriminant Analysis	LDA	-
Quadratic Discriminant Analysis	QDA	-
Support Vector Machine	SVMlin	Linear kernel
Support Vector Machine	SVMrad	Radial kernel
Random Forrest Classifier	RFC	20 estimators
Ada Boost Classifier	ABC	20 estimators
k-Nearest Neighbours	kNN	5 neighbours

4. Results and discussion

4.1. Experimental configuration

We had three objectives with our experimental setup:

- to solve for inter-epoch geospatial polygon comparison (Section 1);
- to evaluate effectiveness of *delta features* vs. *primary features* (Section 3.3); and
- to compare relevant ML approaches in achieving the above.

The first experimental objective relates to the main aim of this paper: can we link the underlying facilities, each represented by a polygon, with one another over time? The second experimental objective relates to the second aim of the paper: finding machine learning-friendly features for geospatial polygons. Finally, the third experimental objective relates to the third aim of the paper: evaluating different machine-learning algorithms' ability to achieve our overall goal of linking facilities over time.

We started by identifying a range of ML models to evaluate. Note that a comprehensive discussion of the different ML approaches referenced within this paper lies outside the scope of the work reported. To select models, we let ourselves be guided by related work (Veer et al., 2018; Tramacere et al., 2016). Consequently, we settled on 11 different ML models. Wherever possible, we used the default configuration provided by the underlying library for each model instance for the given model type. We also tagged each model with a short label for easier reference. Table 4 lists the models selected, labels, and all non-default configurations for each. Table A.12 in Appendix A.2 provides relevant references for all 11 models. In the spirit of repeatability, the interested reader is referred to Appendix A.3 for a more detailed configuration of the convolutionary neural network (CNN) and fully connected neural network (FNN) models.

We then grouped these models into two sets:

- neural networks,
  $\mathbf{M}_{\text{net}} \leftarrow \{\text{CNN, FNN}\}$; and

- non-neural networks,
  $\mathbf{M}_{\text{classic}} \leftarrow \{\text{LOR, NEC, LDA, QDA, SVMlin, SVMrad, RFC, ABC, KNN}\}$.

For $\mathbf{M}_{\text{net}}$ models we utilised the TensorFlow2 (Abadi et al., 2016) library. And for models in $\mathbf{M}_{\text{classic}}$ the SciKitLearn (Pedregosa et al., 2011) library was used. The full set of models evaluated, $\mathbf{M}$, is thus defined as $\mathbf{M} \leftarrow \mathbf{M}_{\text{net}} \cup \mathbf{M}_{\text{classic}}$.

Let the set of all polygons in the tagged dataset be defined as $\mathbf{P}$. For all cases in the tagged dataset comparing polygon x to polygon y, $x \in \mathbf{P}$ and $y \in \mathbf{P}$, the set of primary features are given by $\mathbf{D}_{\text{primary}} \leftarrow \{\phi_{i,x}, \phi_{i,y} | i = 1 \text{ to } 15, x \in \mathbf{P}, y \in \mathbf{P}\}$, with $\phi_{i,x}$ and $\phi_{i,y}$ as defined in Section 3.3. While the delta features are given by $\mathbf{D}_{\text{delta}} \leftarrow \{\delta_{i,(x,y)} | i = 1 \text{ to } 14, x \in \mathbf{P}, y \in \mathbf{P}\}$, with $\delta_{i,(x,y)}$ also as defined in Section 3.3. The full set of training data used during evaluation, $\mathbf{D}$, is defined as $\mathbf{D} \leftarrow \mathbf{D}_{\text{primary}} \cup \mathbf{D}_{\text{delta}}$

To evaluate all $m \in \mathbf{M}$ for $\mathbf{D}$, we defined function `Evaluate`(), as described in more detail in Appendix A.4. `Evaluate`() is derived from the well-established sample-train-test ML pattern supported by SciKitLearn (Pedregosa et al., 2011). It allows for training a set of models, evaluating models during training (i.e., in-sample evaluation), and evaluating best-trained models against unseen data (i.e., out-of-sample evaluation).

On completion, function `Evaluate`() returns *holdBest* containing the overall best instance of $m \in \mathbf{M}_{\text{in}}$ based on accuracy against a hold-out dataset over n_{hold} rounds, as well as all in-training and post-training evaluation scores for analysis.

4.2. Experimental Results

All experimentation and analysis have been done within a Jupyter Notebook environment (Kluyver et al., 2016) using the Python (Van Rossum & Drake, 2009) programming language. A consumer-grade laptop with an Intel i7-1185G7 (at 3GHz) CPU with eight logical cores and 32 GB of RAM was used as a processing platform.

To evaluate the complete set of $m \in \mathbf{M}, \mathbf{M} \leftarrow \mathbf{M}_{\text{net}} \cup \mathbf{M}_{\text{classic}}$ over both $\mathbf{D}_{\text{primary}}$ and $\mathbf{D}_{\text{delta}}$, `Evaluate`() was executed four times with relevant parameter sets as per Table 5. The values for

Table 5: Parameter sets of `Evaluate`() overall ML models selected ($N_{\text{paramSets}} = 4$).

No.	$\mathbf{M}_{\text{in}}$	$\mathbf{D}_{\text{in}}$	n_{hold}	n_{round}	n_{epochsNN}
1	$\mathbf{M}_{\text{net}}$	$\mathbf{D}_{\text{primary}}$	20	4	100
2	$\mathbf{M}_{\text{net}}$	$\mathbf{D}_{\text{delta}}$	20	4	100
3	$\mathbf{M}_{\text{classic}}$	$\mathbf{D}_{\text{primary}}$	20	10	-
4	$\mathbf{M}_{\text{classic}}$	$\mathbf{D}_{\text{delta}}$	20	10	-

n_{hold}, n_{round} and n_{epochsNN} were chosen to provide adequate training and evaluation whilst bearing in mind potential computational limitations. In reference to Algorithm 1, for each execution of `Evaluate`(), the inner loop executed $n_{\text{hold}} \times n_{\text{round}}$ times and the outer loop n_{holds} times. As such, for each $m \in \mathbf{M}_{\text{net}}$, a total of $4 \times 20 = 80$ instances of each m have been trained up to 100 training epochs (Abadi et al., 2016) and evaluated against in-training data, with the best 20 (highest accuracy score) evaluated against hold-out datasets. Similarly, for each $m \in \mathbf{M}_{\text{classic}}$, a total of $10 \times 20 = 200$ instances of each m have been trained and evaluated against in-training data, with the best 20 evaluated against hold-out datasets.

Tables 6 and 7 present our results for hold-out evaluation across all $m \in \mathbf{M}$. To assist with comparison of results, we grouped the results for $\mathbf{M}_{\text{net}}$ together and the results for $\mathbf{M}_{\text{classic}}$ together. For each m the following values are presented:

- the accuracy mean, μ_{primary}, and standard deviation, σ_{primary}, for training with $\mathbf{D}_{\text{primary}}$;
- the accuracy mean, μ_{delta}, and standard deviation, σ_{delta}, for training with $\mathbf{D}_{\text{delta}}$; and
- the difference between above items, $\mu_{\text{diff}} \leftarrow \mu_{\text{delta}} - \mu_{\text{primary}}$ and $\sigma_{\text{diff}} \leftarrow \sigma_{\text{delta}} - \sigma_{\text{primary}}$.

For pragmatic reasons informed by encouraging results using $\mathbf{D}_{\text{delta}}$, the fact that $\mathbf{D}_{\text{delta}}$ is derived from $\mathbf{D}_{\text{primary}}$ and the simpler experimental configuration by only evaluating these two options, the evaluation of $\mathbf{M}$ over a combined $\mathbf{D}_{\text{delta}}$ and $\mathbf{D}_{\text{primary}}$ has not been included in this work.

Table 6: Evaluation results of evaluation sets 1 and 2 ($N_{\text{models}} = 2$).

Id	μ_{primary}	σ_{primary}	μ_{delta}	σ_{delta}	μ_{diff}	σ_{diff}
CNN	0.778	0.020	0.955	0.011	0.177	-0.008
FNN	0.782	0.019	0.969	0.008	0.187	-0.010
Average	0.780	0.019	0.962	0.010	0.182	-0.009

Table 7: Evaluation results of evaluation sets 3 and 4 ($N_{\text{models}} = 9$). The shading of the cells reflect how close a value came to the best value: the darker a cell, the better it performed.

Id	μ_{primary}	σ_{primary}	μ_{delta}	σ_{delta}	μ_{diff}	σ_{diff}
LOR	0.788	0.015	0.968	0.007	0.180	-0.008
NEC	0.649	0.014	0.834	0.022	0.185	0.007
LDA	0.783	0.014	0.951	0.006	0.168	-0.008
QDA	0.916	0.011	0.950	0.011	0.034	-0.000
SVMlin	0.787	0.015	0.969	0.007	0.181	-0.008
SVMrad	0.800	0.016	0.958	0.008	0.158	-0.008
RFC	0.760	0.023	0.956	0.015	0.196	-0.007
ABC	0.752	0.041	0.962	0.009	0.210	-0.032
kNN	0.774	0.014	0.939	0.006	0.164	-0.008
Average	0.779	0.018	0.943	0.010	0.164	-0.008

The model with the highest accuracy, highlighted in bold, is SVMlin trained with $\mathbf{D}_{\text{delta}}$, which forms part of parameter set no. 4, $\mathbf{M}_{\text{classic}}$ with $\mathbf{D}_{\text{delta}}$. Narrowing our scope, we then performed a 5-fold cross-validation (Kohavi, 1995; Pedregosa et al., 2011) over the tagged data for parameter set no. 4. These results, with the highest average accuracy again highlighted in bold, are presented in Table 8. Again, the model with the highest average accuracy score reported is SVMlin with $\mathbf{D}_{\text{delta}}$. With the cross-validation results reinforcing our findings thus far, we again narrowed our scope and used SVMlin with $\mathbf{D}_{\text{delta}}$ to classify the tagged data for further analysis. Table 9 provides this classification's resulting confusion matrix (Kohavi & Provost, 1998). In this table, TP stands for

Table 8: 5-fold cross validation results of accuracy for parameter set 4 ($\mathbf{M}_{\text{classic}}$ with $\mathbf{D}_{\text{delta}}$) (Source: Our data with Kohavi (1995); Pedregosa et al. (2011)).

Id	$\boldsymbol{\mu}$	$\boldsymbol{\sigma}$
LOR	0.969	0.004
NEC	0.820	0.020
LDA	0.948	0.005
QDA	0.948	0.006
SVMlin	**0.970**	0.004
SVMrad	0.960	0.006
RFC	0.964	0.004
ABC	0.960	0.009
kNN	0.946	0.006
Average	0.943	0.007

Table 9: Confusion matrix for SVMlin trained with $\mathbf{D}_{\text{delta}}$ (Source: Our results applying Kohavi & Provost (1998)).

SVMlin +$\mathbf{D}_{\text{delta}}$	**Classified: Not Same**	**Classified: Same**	
Actual: Not Same	$TN = 1\,622$	$FP = 40$	1 662
Actual: Same	$FN = 15$	$TP = 445$	460
	1 637	485	2 122

true positive, TN for *true negative*, FP for *false positive* and FN for *false negative*. Table 10 presents the classification report for this classification as generated using SciKitLearn's library (Pedregosa et al., 2011).

Table 10: Classification report for SVMlin with $\mathbf{D}_{\text{delta}}$ (Source: Our results applying Pedregosa et al. (2011)).

SVMlin +$\mathbf{D}_{\text{delta}}$	**Precision**	**Recall**	**f1-score**	**Support**
Class 0 : Not Same	0.991	0.976	0.983	1 662
Class 1 : Same	0.918	0.967	0.942	460
Accuracy			0.974	2 122
Macro avg	0.954	0.972	0.963	2 122
Weighted avg	0.975	0.974	0.974	2 122

From the tagged dataset in Section 3.2 with 2 122 cases, $|\mathbf{D}_{\text{in}}| = |\mathbf{D}_{\text{primary}}| = |\mathbf{D}_{\text{delta}}| = 2\,122$. Note that $\mathbf{D}_{\text{primary}}$ and $\mathbf{D}_{\text{delta}}$ only differ in the selection of features (*columns*) for the same number of cases (*rows*). Algorithm 1 stipulates $|\mathbf{D}_{\text{hold}}| = 0.3 \times |\mathbf{D}_{\text{in}}| = 0.3 \times 2\,122 = 636.6 \approx 637$, which is the number of tagged cases in the hold-out dataset for evaluation of best models from the inner loop. While $|\mathbf{D}_{\text{round}}| = |\mathbf{D}_{\text{in}}| - |\mathbf{D}_{\text{hold}}| = 1\,485$ is the number of cases passed into the inner loop for training and (in-sample) evaluation. Inside the inner loop, after sampling, $|\mathbf{X}_{\text{test}}| = 0.3 \times |\mathbf{D}_{\text{round}}| = 0.3 \times 1\,485 = 445.5 \approx 446$, which is the number of cases in the (in-training) test set. While $|\mathbf{X}_{\text{train}}| = |\mathbf{D}_{\text{round}}| - |\mathbf{X}_{\text{test}}| = 1\,039$ is the number of cases used for training each time.

Over the 4 parameter sets, the recorded execution times for `Evaluate`() are presented in Table 11.

Table 11: Execution times for `Evaluate`() over all ML models selected ($N_{\text{paramSets}} = 4$).

No.	$\mathbf{M}_{\text{in}}$	$\mathbf{D}_{\text{in}}$	**Execution Time**
1	$\mathbf{M}_{\text{net}}$	$\mathbf{D}_{\text{primary}}$	16 min 48 sec
2	$\mathbf{M}_{\text{net}}$	$\mathbf{D}_{\text{delta}}$	12 min 21 sec
3	$\mathbf{M}_{\text{classic}}$	$\mathbf{D}_{\text{primary}}$	2 min 31 sec
4	$\mathbf{M}_{\text{classic}}$	$\mathbf{D}_{\text{delta}}$	1 min 5 sec

4.3. Discussion

Our results are promising, given the primary objective of our work, which is to solve for inter-epoch polygon comparison adequately. As reported in Tables 6 and 7, the highest average accuracy score against hold-out data, $\mu = 0.969$ ($\sigma = 0.007$), was achieved by SVMlin, trained with $\mathbf{D}_{\text{delta}}$. The full range of average accuracy scores achieved for all $m \in \mathbf{M}$ over $\mathbf{D} \leftarrow \mathbf{D}_{\text{primary}} \cup \mathbf{D}_{\text{delta}}$ is $\mu \in [0.6493, 0.9686]$ ($\sigma \in [0.0059, 0.0408]$). A total of 4 models achieved average accuracy scores of $\mu \geq 0.96$, with accompanying variance $\sigma \leq 0.01$. A 5-fold cross-validation (Kohavi, 1995) for all $m \in \mathbf{M}_{\text{classic}}$ trained with $\mathbf{D}_{\text{delta}}$ reinforced our findings. As per Table 8, cross-validation resulted in accuracy of $\mu \in [0.8204, 0.9703]$ ($\sigma \in [0.0035, 0.0198]$), which is similar to μ_{delta} and σ_{delta} values in Table 7. This time a total of 6 models achieved an average accuracy score of $m \geq 0.96$, with accompanying variance $\sigma \leq 0.01$. The model with the highest average accuracy was again reported as SVMlin, with $\mu = 0.9703$ ($\sigma = 0.0038$). We then proceeded with the SVMlin model trained with $\mathbf{D}_{\text{delta}}$ and classified the tagged dataset for further analysis. As per confusion matrix (Kohavi & Provost, 1998) in Table 9, this resulted in an accuracy of 0.9741. From the classification report (Pedregosa et al., 2011) in Table 10, the average precision is 0.9542, and the average recall is 0.9717. The balanced results for precision and recall are better at 0.9749 and 0.9741, respectively.

Looking at the results from a $\mathbf{D}_{\text{primary}}$ vs $\mathbf{D}_{\text{delta}}$ perspective tells an insightful story. Overall $m \in \mathbf{M}$, our results indicate that training with $\mathbf{D}_{\text{delta}}$ leads to higher average accuracy scores. For 10 models there were an increase of $\mu_{\text{diff}} \in [0.158, 0.210]$. The remaining model, QDA, still reported an increase of $\mu_{\text{diff}} = 0.034$. Future work will investigate the nature of this difference. For now, we surmise that $\mathbf{D}_{\text{delta}}$ solves particular challenges *a priori* which the ML models trained with $\mathbf{D}_{\text{primary}}$ have to overcome as part of training. This possibly includes pairing related features and deriving the correct relation between paired features. For instance, $\delta_{\text{area},(x,y)}$ already 'paired' $\phi_{\text{area},x}$ with $\phi_{\text{area},y}$ and already related the two features via a distance calculation, $\delta_{\text{area},(x,y)} \leftarrow |\phi_{\text{area},x} - \phi_{\text{area},y}|$, the difference in areas of x and y. Training with $\mathbf{D}_{\text{primary}}$ places the burden on the ML model to first pair and then calculate the differences between paired features.

From a $\mathbf{M}_{\text{net}}$ versus $\mathbf{M}_{\text{classic}}$ perspective, our results indicate similar average accuracy results for both. With similarly higher results for training with $\mathbf{D}_{\text{delta}}$ compared to training with $\mathbf{D}_{\text{primary}}$.

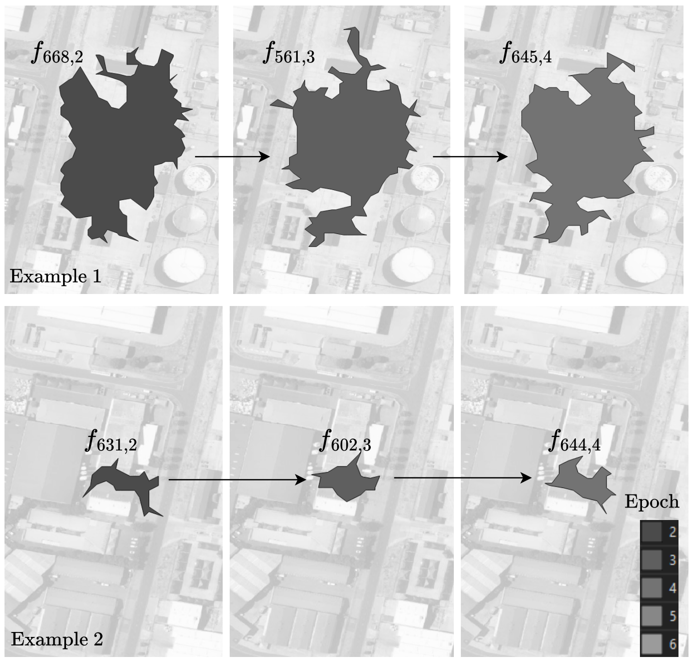


Figure 10: Two examples of true positive cases. (Source: Authors)

Although the best overall model, SVMlin, came from $\mathbf{M}_{\text{classic}}$, its average accuracy only surpassed that of the second highest overall model, FNN, which came from $\mathbf{M}_{\text{net}}$, by 0.0001, or 0.01%. As per summary lines in Tables 6 and 7, the averages of μ_{delta} for each of $\mathbf{M}_{\text{net}}$ and $\mathbf{M}_{\text{classic}}$ differs by $0.962 - 0.943 = 0.019 \leq 0.02$.

Given these results, the polygon comparison problem can be adequately addressed with non-neural Network and non-deep Learning models such as those in $\mathbf{M}_{\text{classic}}$. An influencing factor for this outcome might very well be the relatively small number of features engineered and records sampled. As part of further research, comparing our presented approach and employing Deep Learning models on the non-clustered underlying noisy GPS trail data might provide further insight into this.

Furthermore, although we did not conduct an extensive processing study, our results (Table 11) indicate that training of $\mathbf{M}_{\text{net}}$, with $|\mathbf{M}_{\text{net}}| = 2$ took between $8\times$ and $10\times$ longer than $\mathbf{M}_{\text{classic}}$, with $|\mathbf{M}_{\text{classic}}| = 9$.

Next, we looked at the classification results of the tagged data using the best model: SVMlin trained with $\mathbf{D}_{\text{delta}}$. Table 9 presents specific instances of TP, TN, FP and FN cases. Figure 10 presents 2 examples of true positive outcomes across 3 epochs, for a total of 4 (epochs 2 to 3 and 3 to 4 in each example) correctly classified cases. In example 1, facility $f_{668,2}$ from epoch 2 was correctly identified as the same as facility $f_{561,3}$ in epoch 3. The same for $f_{561,3}$ in epoch 3 and $f_{645,4}$ in epoch 4. Example 2 follows the same logic. As can be seen from both examples, successful matching appears to be resistant to variation in polygon morphology. Given our selected feature set, this is within our expectations (Section 3.3). Furthermore, both examples show a tendency towards polygon overlap to some extent. Further investigation of the larger TP set confirmed at least some overlap or touching of polygons. Remaining mindful of the potential for overfitting, a possible future improvement could be the addition of an overlap feature (De Beer & Joubert,

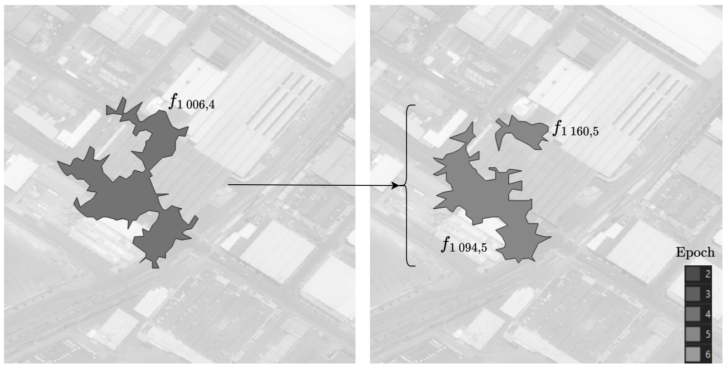


Figure 11: Example of a true positive case with a split. (Source: Authors)

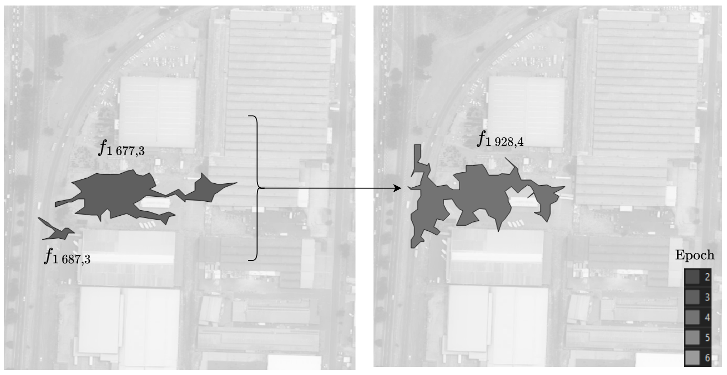


Figure 12: Example of a true positive case with a merge. (Source: Authors)

2022).

Regarding Section 3.2 and Figure 7, these two examples are scenario 1, 'single' matching instances. In all 4 cases, each facility in epoch t matches only one in epoch $t-1$. Figure 11 presents an instance of scenario 2, a 'split' matching. Facility $f_{1\,006,4}$ in epoch 4 was correctly matched to both facilities $f_{1\,160,5}$ and $f_{1\,094,5}$ in epoch 5. This represents a split of $f_{1\,006,4}$ into 2 facilities in the next epoch. Figure 12 presents an instance of scenario 3, a 'merge' matching. Facilities $f_{1\,677,3}$ and $f_{1\,687,3}$ in epoch 3 were both matched to facility $f_{1\,928,4}$ in epoch 4. This represents a merge of $f_{1\,677,3}$ and $f_{1\,687,3}$ into a single facility in the next epoch.

Figure 13 presents 4 examples of true negative outcomes. In example 1, facility $f_{1\,013,2}$ from epoch 2 has been correctly identified as not the same as facility $f_{1\,078,3}$ in epoch 3. The same logic applies for examples $2, 3$ and 4. All 4 examples indicate no overlap of polygons. On investigation of the full TN set, not overlapping was found to be a strong indicator of negative outcome. This reinforces the potential benefit of an overlap feature as per the TP discussion above.

Figure 14 presents 4 examples of false-negative outcomes; true matches are incorrectly classified as not matching. The first observation is that in all 4 examples, the two facilities overlap to some extent, in line with the TP cases described above. This confirms the potential benefit of using an overlap feature for future improvement.

Figure 15 provides 4 examples of false positives, cases tagged as not matching but classified as matched. Again, the observation holds that facilities of true negative cases mostly do not overlap. Only in example 4 do facilities $f_{1\,266,4}$ and $f_{1\,0883}$ overlap.

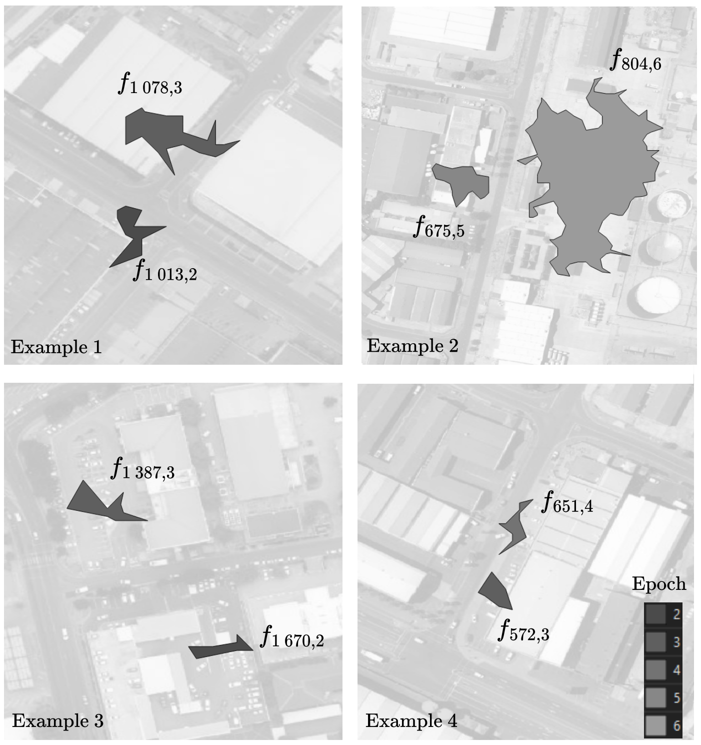


Figure 13: Four examples of true negative cases. (Source: Authors)

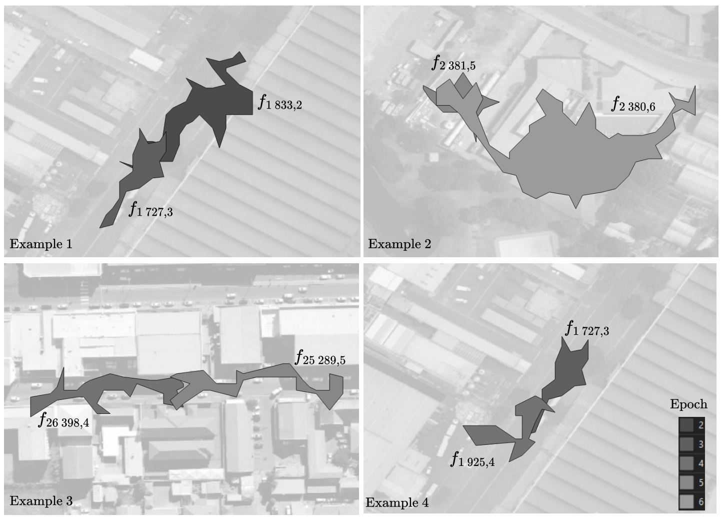


Figure 14: Four examples of false negative cases. (Source: Authors)

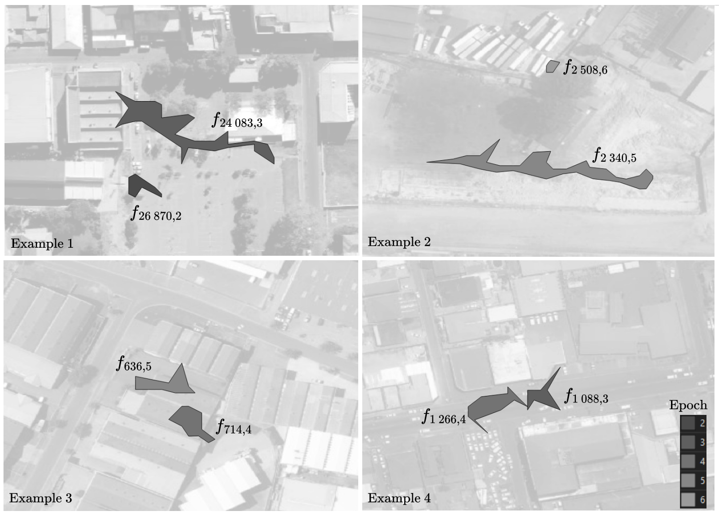


Figure 15: Four examples of false positive cases. (Source: Authors)

In attempting to interpret the above false positive and false negative cases, we acknowledge that the accuracy of our results depends on the data tagging process described in Section 3.2. Furthermore, our current experimental approach does not utilise contextual information, e.g. geospatial land use data. These two aspects represent potential areas of further investigation to improve the accuracy and the interpretation of results.

Accurately matching facility polygons inter-epoch enables us to derive what we call *facility lineages* or *lines*. Recall that facilities are nodes in a complex facility network per epoch. Facility lines, in effect, link these complex networks together and enable us to study potential change over time of both the embedded facilities and the network structures concerning these lines.

Although this facility lineage work primarily resides outside the scope of the work reported here, briefly consider the following as an indication of capability and expected results. Figure 16 serves as an example. In this figure, there is one facility line over 3 epochs. This line was obtained

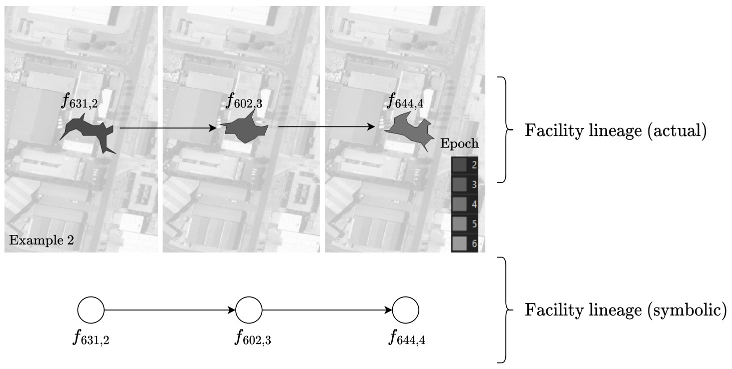


Figure 16: Concepts of facility lineage. (Source: Authors)

by linking relevant true positive cases from classification. In this instance, facility $f_{631,2}$ in epoch 2 was matched to facility $f_{602,3}$ in epoch 3, which in turn was matched to facility $f_{644,4}$ in epoch 4. From this line, we can derive some features for further analysis. For example, in any epoch, we can determine how long a line will still exist in the data by looking at the delta between the current and largest epoch values in the line. To determine how long a line has been present in the data in any epoch, in effect, the age of the line, we look at the delta between the current and smallest epoch in the line.

Considering the line in the context of linking a sequence of complex facility networks can address several network-related questions. For example, Figure 17, which extends the symbolic example in Figure 16 of an actual line, illustrates how one can now investigate the evolution of network attributes such as in-degree (deg^-) and out-degree (deg^+) connectivity (Newman, 2018), or more advanced metrics such as the vulnerability of the networks (Viljoen & Joubert, 2016, 2018), or the evolution of microcomunities (Viljoen & Joubert, 2019).

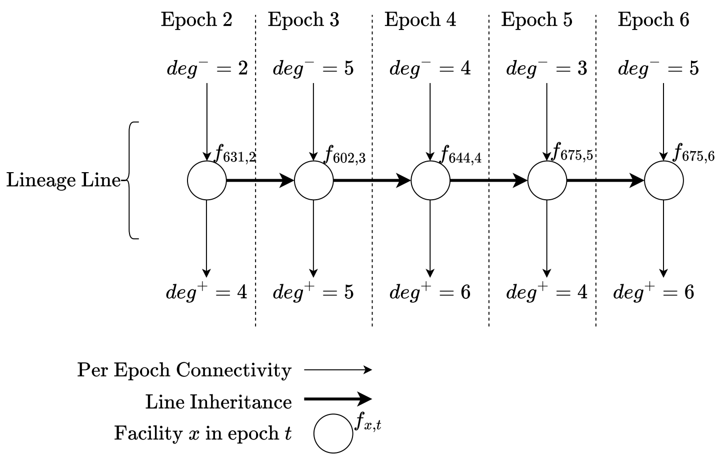


Figure 17: Studying facility and complex network evolution over time. (Source: Authors)

These are the number of directional links into and from a facility, indicative of transport economic relationships with other facilities. From this diagram, we can, for instance, see that over the 5 epochs, the facilities in the line mostly remain more active in an outward-focussed manner than an inward-focussed manner; this is due to the consistently higher out-degree over the epochs.

5. Conclusion

Over the last decade, research has made good progress in studying connectivity and vulnerability of freight movement using network theory, albeit mainly within a single epoch. This paper contributes to the body of knowledge by providing a rigorous and highly accurate methodology for identifying facilities from ubiquitous GNSS traces, global positioning system (GPS) specifically in this case, over multiple periods. To accomplish this, we addressed the underlying problem of accurately matching two-dimensional geospatial polygons associated with facilities. A non-neural network, non-deep learning SVM with a linear kernel model (Boser et al., 1992) achieved the best results with a mean average accuracy of $\mu = 0.969$ ($\sigma = 0.007$) on a consumer-grade computing platform. The paper also introduced the concept of delta features, which contributed 0.181 to this accuracy score.

A couple of immediate next steps remain. Firstly, expand the current multi-epoch approach so that consecutive epochs overlap. For example, say a single epoch represents a four-week month, then the first epoch is calculated from data spanning weeks 1 through 4. The next epoch, also representing a month, should span weeks 2 to 5 instead of the current non-overlapping approach, where the second epoch covers weeks 5 through 8. Smoother and overlapping transitions may yield improvements in ML accuracy. Overlapping epochs can represent rolling horizons in practice, a concept well-established in logistics planning.

Secondly, the question remains about why the delta features result in higher accuracy. Investigating this phenomenon may provide future insight into network structure dynamics. Finally, another area to consider is moving away from purely anonymous geospatial features and combining the enriched data with more contextual characteristics such as land-use data.

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous systems. *arXiv preprint arXiv:1603.04467*, . doi:10.48550/ARXIV.1603.04467.

Anami, B. S., & Bhandage, V. A. (2019). Combined Hu moments, orientation knowledge, and grid intersections feature based identification of Bharatanatyam mudra images. *Pattern Analysis and Applications*, *22*, 1439–1454. doi:10.1007/s10044-018-0715-2.

Ansari, M. Y., Mainuddin, Ahmad, A., & Bhushan, G. (2021). Spatiotemporal trajectory clustering: A clustering algorithm for spatiotemporal data. *Expert Systems with Applications*, *178*, 115048. doi:0.1016/j.eswa.2021.115048.

Bebis, G., & Georgiopoulos, M. (1994). Feed-forward neural networks. *IEEE Potentials*, *13*, 27–31. doi:10.1109/45.329294.

Birch, C. P. D., Oom, S. P., & Beecham, J. A. (2007). Rectangular and hexagonal grids used for observation, experiment and simulation in ecology. *Ecological Modelling*, *206*, 347–359. doi:10.1016/j.ecolmodel.2007.03.041.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144–152). doi:10.1145/130385.130401.

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5–32. doi:10.1023/A:1010933404324.

Bruce, R. F., & Wiebe, J. M. (1999). Recognizing subjectivity: a case study in manual tagging. *Natural Language Engineering*, *5*, 187–205. doi:10.1017/S1351324999002181.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). Api design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, . doi:10.48550/ARXIV.1309.0238.

Cardillo, M. (2010). Some applications of geometric morphometrics to archaeology. In A. M. Elewa (Ed.), *Morphometrics for Nonmorphometricians* (pp. 325–341). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-540-95853-6_15.

Cecotti, H. (2018). Rotation invariant descriptors for galaxy morphological classification. *arXiv preprint arXiv:1812.04706*, . doi:10.48550/ARXIV.1812.04706.

Chan, C. K., & Tan, S. T. (2001). Determination of the minimum bounding box of an arbitrary solid: an iterative approach. *Computers and Structures*, *79*, 1433–1449. doi:10.1016/S0045-7949(01)00046-3.

Cich, G., Knapen, L., Bellemans, T., Janssens, D., & Wets, G. (2016). Threshold settings for TRIP/STOP detection in GPS traces. *Journal of Ambient Intelligence and Humanized Computing*, *7*, 395–413.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, *13*, 21–27. doi:10.1109/TIT.1967.1053964.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, *20*, 215–232. doi:10.1111/j.2517-6161.1958.tb00292.x.

Daoui, A., Karmouni, H., Sayyouri, M., Qjidaa, H., Maaroufi, M., & Alami, B. (2021). New robust method for image copyright protection using histogram features and sine cosine algorithm. *Expert Systems with Applications*, *177*, 114978. doi:10.1016/j.eswa.2021.114978.

De Beer, D. J., & Joubert, J. W. (2022). Evolutionary optimisation of large-scale activity clustering with increased automation. *Computers and Operations Research*, *146C*, article 105925. doi:10.1016/j.cor.2022.105925.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*, 78–87. doi:10.1145/2347736.2347755.

Du, P., Bai, X., Tan, K., Xue, Z., Samat, A., Xia, J., Li, E., Su, H., & Liu, W. (2020). Discovering personal gazetteers: An interactive clustering approach. *Journal of Geovisualization and Spatial Analysis*, *4(13)*. doi:10.1007/s41651-020-00048-5.

Espejel-Cabrera, J., Cervantes, J., García-Lamont, F., Ruiz Castilla, J. S., & D. Jalili, L. (2021). Mexican sign language segmentation using color based neuronal networks to detect the individual skin color. *Expert Systems with Applications*, *183*, 115295. doi:10.1016/j.eswa.2021.115295.

Fernando, M., & Wijayanayake, J. (2020). Novel approach to use Hu moments with image processing techniques for real time sign language communication. *arXiv preprint arXiv:2007.09859*, . doi:10.48550/ARXIV.2007.09859.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*, 119–139. doi:10.1006/jcss.1997.1504.

Fu, Z., Fan, L., Yu, Z., & Zhou, K. (2018). A moment-based shape similarity measurement for areal entities in geographical vector data. *ISPRS International Journal of Geo-Information*, *7*. doi:10.3390/ijgi7060208.

Godefroy, J. E., Bornert, F., Gros, C. I., & Constantinesco, A. (2012). Elliptical fourier descriptors for contours in three dimensions: A new tool for morphometrical analysis in biology. *Comptes Rendus Biologies*, *335*, 205–213. doi:10.1016/j.crvi.2011.12.004.

Hales, T. (2001). The honeycomb conjecture. *Discrete and Computational Geometry.*, *25*, 1–22. doi:10.1007/s004540010071.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction.* volume 2. Springer. doi:10.1007/978-0-387-21606-5.

Háznagy, A., Fi, I., London, A., & Nemeth, T. (2015). Complex network analysis of public transportation networks: A comprehensive study. In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)* (pp. 371–378). doi:10.1109/MTITS.2015.7223282.

Hu, M.-K. (1962). Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, *8*, 179–187. doi:10.1109/TIT.1962.1057692.

Joubert, J., & Axhausen, K. (2011). Inferring commercial vehicle activities in Gauteng, South Africa. *Journal of Transport Geography*, *19*, 115–124. doi:10.1016/j.jtrangeo.2009.11.005.

Joubert, J. W., & Axhausen, K. W. (2013). A complex network approach to understand commercial vehicle movement. *Transportation*, *40*, 729–750. doi:10.1007/s11116-012-9439-0.

Joubert, J. W., & Meintjes, S. (2015). Repeatability & reproducibility: Implications of using GPS data for freight activity chains. *Transportation Research Part B: Methodological*, *76*, 81–92. doi:10.1016/j.trb.2015.03.007.

Joubert, J. W., & Meintjes, S. (2016). Freight activity chain generation using complex networks of connectivity. *Transportation Research Procedia*, *12*, 425–435. doi:10.1016/j.trpro.2016.02.078.

Kim, K. S., Choi, H. H., Moon, C. S., & Mun, C. W. (2011). Comparison of k-nearest neighbor, quadratic discriminant and linear discriminant analysis in classification of electromyogram signals based on the wrist-motion directions. *Current Applied Physics*, *11*, 740–745. doi:10.1016/j.cap.2010.11.051.

Kluyver, T., Ragan-Kelley, B., Perez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides, & B. Schmidt (Eds.), *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (pp. 87 – 90). Amsterdam: IOS Press. doi:10.3233/978-1-61499-649-1-87.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2* IJCAI'95 (pp. 1137–1143). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. doi:10.5555/1643031.1643047.

Kohavi, R., & Provost, F. (1998). Guest editors' introduction: On applied research in machine learning. *Machine Learning*, *30*, 127–132. doi:10.1023/A:1007442505281.

Kuhl, F. P., & Giardina, C. R. (1982). Elliptic fourier features of a closed contour. *Computer Graphics and Image Processing*, *18*, 236–258. doi:10.1016/0146-664X(82)90034-X.

Kupe, M., Saync, B., Demir, B., Ercisli, S., Baron, M., & Sochor, J. (2021). Morphological characteristics of grapevine cultivars and closed contour analysis with elliptic fourier descriptors. *Plants*, *10*. doi:10.3390/plants10071350.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444. doi:10.1038/nature14539.

Li, Z. (2020). Geospatial big data handling with high performance computing: current approaches and future directions. In *High Performance Computing for Geospatial Applications* (pp. 53–76). Springer.

Lones, M. A. (2021). How to avoid machine learning pitfalls: a guide for academic researchers. *arXiv preprint arXiv:2108.02497*, . doi:10.48550/ARXIV.2108.02497. arXiv:2108.02497.

Mai, G., Janowicz, K., Hu, Y., Gao, S., Yan, B., Zhu, R., Cai, L., & Lao, N. (2022). A review of location encoding for GeoAI: methods and applications. *International Journal of Geographical Information Science*, *36*, 639–673. doi:10.1080/13658816.2021.2004602.

Malzer, C., & Baum, M. (2020). A hybrid approach to hierarchical density-based cluster selection. *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, . doi:10.1109/mfi49285.2020.9235263.

Newman, M. (2018). *Networks*. Oxford University Press. URL: https://global.oup.com/academic/product/networks-9780198805090 https://global.oup.com/academic/product/networks-9780198805090.

Nurmi, P. (2009). *Identifying Meaningful Places*. Ph.D. thesis University of Helsinki. http://urn.fi/URN:ISBN:978-952-10-5790-8.

OpenCV team (2022). *Open Source Computer Vision Library*. URL: https://github.com/opencv.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html. Software available from https://scikit-learn.org/.

Peters, J. F. (2017). *Foundations of Computer Vision*. Springer International Publishing. doi:10.1007/978-3-319-52483-2.

Popescu, V., & Rosen, P. (2006). Forward rasterization. *ACM Transactions on Graphics (TOG)*, *25*, 375–411. doi:10.1145/1138450.1138460.

Poux, F., Hallot, P., Neuville, R., & Billen, R. . (2016). Smart point cloud: Definition and remaining challenges. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *4(2W1)*, 119–127. doi:10.5194/isprs-annals-IV-2-W1-119-2016.

QGIS Development Team (2019). *QGIS Geographic Information System.*. Open Source Geospatial Foundation Project Oregon, United States. URL: https://qgis.org/en/site/ software available from https://qgis.org/en/site/.

Ruan, X., & Liu, B. (2020). Review of 3d point cloud data segmentation methods. *International Journal of Advanced Network, Monitoring and Controls*, *5(1)*, 66–71. doi:10.21307/ijanmc-2020-010.

Schmittbuhl, M., Minor, J. M. L., Taroni, F., & Mangin, P. (2001). Sexual dimorphism of the human mandible: demonstration by elliptical fourier analysis. *International Journal of Legal Medicine*, *115*, 100,101. doi:10.1007/s004140100219.

Sharman, B. W., & Roorda, M. J. (2011). Analysis of freight global positioning system data: clustering approach for identifying trip destinations. *Transportation Research Record*, *2246*, 83–91.

Shi, Z., & Pun-Cheng, L. S. (2019). Spatiotemporal data clustering: A survey of methods. *International Journal of Geo-Information*, *8:3*. doi:https://doi.org/10.3390/ijgi8030112. https://doi.org/10.3390/ijgi8030112.

Teri, S. S., Musliman, I. A., & Rahman, A. A. (2022). Gpu utilization in geoprocessing big ggodata: A review. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLVI-4/W3-2021*, 295–304. doi:10.5194/isprs-archives-XLVI-4-W3-2021-295-2022.

Tramacere, A., Paraficz, D., Dubath, P., Kneib, J., & Courbin, F. (2016). Asterism: application of topometric clustering algorithms in automatic galaxy detection and classification. *Monthly Notices of the Royal Astronomical Society*, *463*, 2939–2957. doi:10.1093/mnras/stw2103.

Trent, N. M., & Joubert, J. W. (2022). Logistics sprawl and the change in freight transport activity: A comparison of three measurement methodologies. *Journal of Transport Geography*, *101*, 103350. doi:10.1016/j.jtrangeo.2022.103350.

Trent, N. M., Joubert, J. W., Gidofalvi, G., & Kordnejad, B. (2020). A matching algorithm to study the evolution of logistics facilities extracted from gps traces. *Transportation Research Procedia*, *46*, 237–244. doi:10.1016/j.trpro.2020.03.186.

Uber Engineering Team (2022). *H3: A Hexagonal Hierarchical Geospatial Indexing System.*. Uber Uber, United States. URL: https://github.com/uber/h3 software available at https://github.com/uber/h3.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace. Software available at https://www.python.org/.

Veer, R. v. t., Bloem, P., & Folmer, E. (2018). Deep learning for classification tasks on geospatial vector polygons. *arXiv preprint arXiv:1806.03857*, . doi:10.48550/ARXIV.1806.03857.

Viljoen, N. M., & Joubert, J. W. (2016). The vulnerability of the global container shipping network to targeted link disruption. *Physica A*, *462*, 396–409. doi:10.1016/j.physa.2016.06.111.

Viljoen, N. M., & Joubert, J. W. (2018). The road most travelled: The impact of urban road infrastructure on supply chain network vulnerability. *Networks and Spatial Economics*, *18*, 85–113. doi:10.1007/s11067-017-9370-1.

Viljoen, N. M., & Joubert, J. W. (2019). Supply chain micro-communities in urban areas. *Journal of Transport Geography*, *74*, 211–222. doi:10.1016/j.jtrangeo.2018.11.011.

Wang, J., Qian, W., & Liu, H. (2018). Shape analysis of pottery using elliptic fourier descriptor and 3d scanning. In

F. Rebelo, & M. Soares (Eds.), *Advances in Ergonomics in Design* (pp. 62–70). Springer International Publishing. doi:10.1007/978-3-319-60582-1_7.

Zhao, L., Chen, L., Ranjan, R., Choo, K., & He, J. (2016). Geographical information system parallelization for spatial big data processing: a review. *Cluster Computing*, *19(1)*, 139–152. doi:10.1007/s10586-015-0512-2.

Zheng, A., & Casari, A. (2018). *Feature engineering for machine learning: principles and techniques for data scientists*. O'Reilly Media, Inc. https://www.oreilly.com/library/view/feature-engineering-for/9781491953235/.

Zhou, C., Frankowski, D., Ludford, P., Shekhar, S., & Terveen, L. (2004). Discovering personal gazetteers: An interactive clustering approach. *GIS '04 Proceedings of the 12th annual ACM international workshop on Geographic information systems*, *76*, 266–273. doi:10.1145/1032222.1032261.

Appendix A. Additional information

Appendix A.1. Hu Moments

This section provides additional conceptual background information on Hu moments. It has been adapted from Hu (1962) and Tramacere et al. (2016).

Hu moments are geometric moments invariant to rotation, scaling and translation. In the simplest terms, a geometric moment is some form of weighted average over a given data structure. In the calculation of Hu moments, data entries are weighted by the content of the data structure (*values*) as well as the location of the data in the data structure (*value location*).

Formally, the geometric moment of order $(p+q)$ for a $M \times N$ two-dimensional distribution of points (x_i, y_j) with $x_i \in \mathbb{Z}$ and $y_j \in \mathbb{Z}$ is defined by (A.1).

$$m_{p,q} = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (x_i)^p (y_j)^q \tag{A.1}$$

In the case of a two-dimensional digital image with the same parameters and with pixel intensity given by $f(x_i, y_j)$, (A.1) becomes (A.2).

$$m_{p,q} = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (x_i)^p (y_j)^q f(x_i, y_j) \tag{A.2}$$

Note that for the work reported in this paper, pixel intensity is either 0 or 1 for strictly black and white images, thus $f(x_i, y_j) \in \{0, 1\}$. The centroid is given by $(\bar{x}, \bar{y})$ with

$$\bar{x} = \frac{m_{1,0}}{m_{0,0}} \tag{A.3}$$

and

$$\bar{y} = \frac{m_{0,1}}{m_{0,0}} \tag{A.4}$$

respectively. The central moment is defined by (A.5).

$$\mu_{p,q} = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (x_i - \bar{x})^p (y_j - \bar{y})^q \tag{A.5}$$

For the two-dimensional image case the central moment is defined by (A.6).

$$\mu_{p,q} = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(x_i - \bar{x})^p (y_j - \bar{y})^q f(x_i, y_j) \tag{A.6}$$

From the above, normalised central moments are obtained by (A.7).

$$\eta_{p,q} = \frac{\mu_{p,q}}{\mu_{0,0}^{1+\frac{p+q}{2}}} \tag{A.7}$$

Building on the above, Hu (1962) developed the following moments that are invariant under scaling, translation and rotation:

$$Hu_1 = \eta_{2,0} + \eta_{0,2} \tag{A.8a}$$
$$Hu_2 = (\eta_{2,0} - \eta_{0,2})^2 + 4(\eta_{1,1})^2 \tag{A.8b}$$
$$Hu_3 = (\eta_{3,0} - 3\eta_{1,2})^2 + 3(\eta_{0,3} - 3\eta_{2,1})^2 \tag{A.8c}$$
$$Hu_4 = (\eta_{3,0} + \eta_{1,2})^2 + (\eta_{0,3} + \eta_{2,1})^2 \tag{A.8d}$$
$$\begin{aligned}Hu_5 = {} & (\eta_{3,0} - 3\eta_{1,2})(\eta_{3,0} + \eta_{1,2})[(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{0,3} + \eta_{2,1})^2] \\ & + (3\eta_{2,1} - \eta_{0,3})(\eta_{0,3} + \eta_{2,1})[3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{0,3} + \eta_{2,1})^2]\end{aligned} \tag{A.8e}$$
$$Hu_6 = (\eta_{2,0} - \eta_{0,2})[(\eta_{3,0} + \eta_{1,2})^2 - 7(\eta_{0,3} + \eta_{2,1})^2] + 4\eta_{1,1}(\eta_{3,0} + \eta_{1,2})(\eta_{0,3} + \eta_{2,1}) \tag{A.8f}$$
$$\begin{aligned}Hu_7 = {} & (3\eta_{2,1} - \eta_{0,3})(\eta_{3,0} + \eta_{1,2})[(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{0,3} + \eta_{2,1})^2] \\ & + (\eta_{3,0} - 3\eta_{1,2})(\eta_{0,3} + \eta_{2,1})[3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{0,3} + \eta_{2,1})^2]\end{aligned} \tag{A.8g}$$

Appendix A.2. Selected Machine Learning Model References

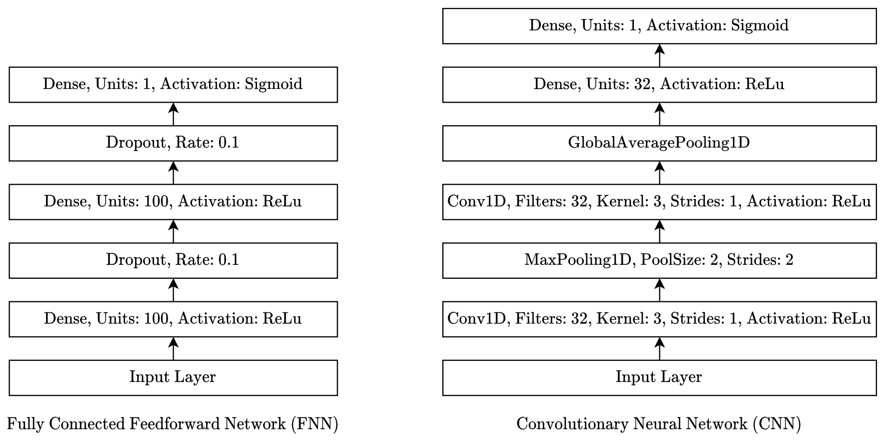
This section provides references for machine learning models selected for evaluation.

Table A.12: ML models selected for evaluation ($N_{\text{models}} = 11$).

Model	**Id**	**Reference**
Convolutional Neural Network	CNN	LeCun et al. (2015)
Fully Connected Neural Network	FNN	Bebis & Georgiopoulos (1994)
Logistic Regression	LOR	Cox (1958)
Nearest Centroid	NEC	Hastie et al. (2009)
Linear Discriminant Analysis	LDA	Kim et al. (2011)
Quadratic Discriminant Analysis	QDA	Kim et al. (2011)
Support Vector Machine	SVMlin	Boser et al. (1992)
Support Vector Machine	SVMrad	Boser et al. (1992)
Random Forrest Classifier	RFC	Breiman (2001)
Ada Boost Classifier	ABC	Freund & Schapire (1997)
k-Nearest Neighbours	kNN	Cover & Hart (1967)

Appendix A.3. Detailed configuration of the neural networks

Even though our features differed from those of Veer et al. (2018), we opted for a CNN configuration similar to theirs. Through subsequent experimentation, we settled on the configuration depicted in Figure A.18. For the FNN, we selected a fully connected feed-forward network, also depicted in Figure A.18.



Dense, Units: 1, Activation: Sigmoid
Dropout, Rate: 0.1
Dense, Units: 100, Activation: ReLu
Dropout, Rate: 0.1
Dense, Units: 100, Activation: ReLu
Input Layer
Fully Connected Feedforward Network (FNN)
Dense, Units: 1, Activation: Sigmoid
Dense, Units: 32, Activation: ReLu
GlobalAveragePooling1D
Conv1D, Filters: 32, Kernel: 3, Strides: 1, Activation: ReLu
MaxPooling1D, PoolSize: 2, Strides: 2
Conv1D, Filters: 32, Kernel: 3, Strides: 1, Activation: ReLu
Input Layer
Convolutionary Neural Network (CNN)

Figure A.18: Configuration of neural network models FNN and CNN. (Source: Authors)

Appendix A.4. Pseudocode and evaluation explanation

To evaluate all $m \in \mathbf{M}$ for $\mathbf{D}$, we defined function `Evaluate`(), as described in pseudocode in Algorithm 1. `Evaluate`() is derived from the well-established sample-train-test ML pattern supported by SciKitLearn (Pedregosa et al., 2011). It allows for the training of a set of models, the evaluation of models during training (i.e. in-sample evaluation), as well as the evaluation of best-trained models against unseen data (i.e. out-of-sample evaluation). The algorithm takes as input 5 parameters:

- The set of models, either neural networks or classic models, $\mathbf{M}_{\text{in}} \in \{\mathbf{M}_{\text{net}}, \mathbf{M}_{\text{classic}}\}$;
- Tagged dataset (i.e. cases) with either the primary or delta features, $\mathbf{D}_{\text{in}} \in \{\mathbf{D}_{\text{primary}}, \mathbf{D}_{\text{delta}}\}$;
- Number of repetitions to *evaluate* against hold-out unseen data, n_{hold};
- Number of repetitions to *train and evaluate* against training data, n_{round}; and
- Number of neural network training epochs, n_{epochsNN} for training of all $m \in \mathbf{M}_{\text{net}}$.

The decision to separately evaluate $\mathbf{M}_{\text{net}}$ and $\mathbf{M}_{\text{classic}}$ is informed by the availability of a 'number of epochs' parameter with neural network training (Abadi et al., 2016). This is the number of training cycles TensorFlow will execute during the training of the model in an attempt to improve its accuracy. Setting this parameter, n_{epochsNN}, to 100, we were able to reduce the number of inner loop executions, n_{round}, for evaluation of $\mathbf{M}_{\text{net}}$.

`Evaluate`() returns as output a data structure containing the best instance of each model $m \in \mathbf{M}_{\text{in}}$, along with all evaluation scores collected during and post-training for each m. The best instance of m is deemed as each m with the highest average accuracy score obtained over several post-training evaluations against hold-out datasets.

Algorithm 1 Evaluation function *Evaluate*

```
1: function EVALUATE($\mathbf{M}_{\text{in}}, \mathbf{D}_{\text{in}}, n_{\text{hold}}, n_{\text{round}}, n_{\text{epochsNN}}$)    ▷ Evaluate $\forall m \in \mathbf{M}_{\text{in}}$ with $\mathbf{D}_{\text{in}}$
2:     $holdBest \leftarrow \{\}$    ▷ New key:value dictionary
3:     for $i = 1$ to $n_{\text{hold}}$ do    ▷ Hold-out evaluation loop
4:         $\mathbf{D}_{\text{hold}} \leftarrow sample(\mathbf{D}_{\text{in}}, 0.3)$    ▷ Randomly sample 0.3 of cases
5:         $\mathbf{D}_{\text{round}} \leftarrow \mathbf{D}_{\text{in}} - \mathbf{D}_{\text{hold}}$    ▷ Remaining 0.7 of cases
6:         $roundBest \leftarrow \{\}$    ▷ New key:value dictionary
7:         for $j = 1$ to $n_{\text{round}}$ do    ▷ Inner training loop
8:             $\mathbf{X}_{\text{test}} \leftarrow sample(\mathbf{D}_{\text{round}}, 0.3)$    ▷ Randomly sample 0.3 of cases
9:             $\mathbf{X}_{\text{train}} \leftarrow \mathbf{D}_{\text{round}} - \mathbf{X}_{\text{test}}$    ▷ Remaining 0.7 of cases
10:            $\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{test}} \leftarrow scale(\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{test}})$    ▷ Scale training and testing feature data
11:            for $m \in \mathbf{M}_{\text{in}}$ do    ▷ For each ML model type
12:                $m \leftarrow newInstance(m)$    ▷ New instance of ML model $m$
13:                if $m \in \mathbf{M}_{\text{net}}$ then
14:                    $m \leftarrow train(m, \mathbf{X}_{\text{train}}, n_{\text{epochsNN}})$    ▷ Fit ML model $m \in \mathbf{M}_{\text{net}}$ to training data
15:                else
16:                    $m \leftarrow train(m, \mathbf{X}_{\text{train}})$    ▷ Fit ML model $m \in \mathbf{M}_{\text{classic}}$ to training data
17:                end if
18:                $s \leftarrow test(m, \mathbf{X}_{\text{test}})$    ▷ Test ML model $m$ with training data
19:                $roundBest \leftarrow update(roundBest, m, s)$    ▷ Update roundBest with $m$ and scores $s$
20:            end for
21:        end for
22:        $\mathbf{D}_{\text{hold}} \leftarrow scale(\mathbf{D}_{\text{hold}})$    ▷ Scale hold-out data
23:        for $m \in roundBest$ do    ▷ For each best-of-round model instance
24:            $p \leftarrow predict(m, \mathbf{D}_{\text{hold}})$    ▷ Use $m$ to predict unseen hold-out cases
25:            $s \leftarrow evaluate(p, \mathbf{D}_{\text{hold}})$    ▷ Evaluate predictions $p$ against hold-out data
26:            $holdBest \leftarrow update(holdBest, m, s)$    ▷ Update holdBest with $m$ and scores $s$
27:        end for
28:    end for
29:    return $holdBest$    ▷ Return evaluation results with best case models
30: end function
```

The algorithm consists of two loops, an outer *hold* loop and an inner *round* loop. Briefly, the inner loop trains and evaluates (in-sample) all $m \in \mathbf{M}_{\text{in}}$. While the outer loop evaluates all trained $m \in \mathbf{M}_{\text{in}}$ from the inner loop against an out-of-sample hold-out dataset.

As the first step, a data structure *holdBest* is initialised, storing hold-out accuracy scores and the best overall models during processing.

For each cycle in the outer loop, the algorithm first samples (without replacement) 30% of $\mathbf{D}_{\text{in}}$ as a hold-out unseen dataset, $\mathbf{D}_{\text{hold}}$. The remaining 70% of $\mathbf{D}_{\text{in}}$, $\mathbf{D}_{\text{round}} \leftarrow \mathbf{D}_{\text{in}} - \mathbf{D}_{\text{hold}}$, is deemed training data. Along with a record-keeping data structure *roundBest*, $\mathbf{D}_{\text{round}}$ is passed into the inner loop for model training and (in-sample) evaluation.

The inner loop starts by sampling (without replacement) 30% of $\mathbf{D}_{\text{round}}$ as in-training test set, $\mathbf{X}_{\text{test}}$. The remaining 70% of $\mathbf{D}_{\text{round}}$, $\mathbf{X}_{\text{train}} \leftarrow \mathbf{D}_{\text{round}} - \mathbf{X}_{\text{test}}$, is deemed training data for the current cycle.

Next, both $\mathbf{X}_{\text{train}}$ and $\mathbf{X}_{\text{test}}$ are scaled. Machine learning models generally benefit from scaling of training data (Pedregosa et al., 2011), with both ML libraries used in our work advising for the use of scaling (Buitinck et al., 2013; Abadi et al., 2016). Two types of scaling are of relevance. Normalisation, that scales values into a range, typically $x_{\text{scaled}} \in [0.0, 1.0]$, and standardisation, that scales values to have $\mu_{\text{scaled}} = 0.0$ and $\sigma_{\text{scaled}} = 1.0$. We experimentally settled on standardisation for $m \in \mathbf{M}_{\text{classic}}$ (SciKitLearn's `StandardScaler`) and normalisation for $m \in \mathbf{M}_{\text{net}}$ (SciKitLearn's `MinMaxScaler`). To avoid data leakage (Lones, 2021), the scaling configuration is obtained by only using $\mathbf{X}_{\text{train}}$, but applied to both $\mathbf{X}_{\text{train}}$ and $\mathbf{X}_{\text{test}}$ once configured.

Once the data has been scaled, the algorithm iterates over all $m \in \mathbf{M}_{\text{in}}$. For each m, a new instance is obtained per default configuration for the underlying library and non-default configuration per Table 4 and Figure A.18. Each m is then trained (m's `fit()` or equivalent function as per relevant library) with the scaled $\mathbf{X}_{\text{train}}$ and evaluated (m's `evaluate()` or equivalent function as per relevant library) against the scaled $\mathbf{X}_{\text{test}}$ for accuracy. All accuracy scores of m are then added to *roundBest*. If the current m's accuracy score is higher than the best instance of m stored in *roundBest* from a previous cycle, it is replaced with the current m.

The inner loop then repeats the above up to n_{round} times. Each time *roundBest* is updated with all accuracy scores and the best instance of each m. At the end of the inner loop, *roundBest* contains the best overall instance for this training round of each $m \in \mathbf{M}_{\text{in}}$ as well as all accuracy scores accumulated.

The outer loop continues processing. As first step, $\mathbf{D}_{\text{hold}}$ is scaled, with scaling only informed by $\mathbf{D}_{\text{hold}}$ itself to avoid data leakage (Lones, 2021). The type of scaling applied is the same as in the inner loop, either normalisation for $\mathbf{M}_{\text{net}}$ or standardisation for $\mathbf{M}_{\text{classic}}$.

For each $m \in$ *roundBest*, which represents the best m over the previous n_{round} inner rounds, m is used with the scaled $\mathbf{D}_{\text{hold}}$ to predict the case outcomes in $\mathbf{D}_{\text{hold}}$. The predictions are then compared to the actual case outcomes as per tagged data and evaluated for accuracy. *holdBest* is then updated with the hold-out accuracy scores for m. If the current m scored better than the previously stored best m, it is replaced with the current m.

The outer loop will then repeat the above up to n_{hold} times, evaluating trained models against a hold-out dataset that was not used for training or in-training evaluation.