Geospatial Systems Thesis

TOWARDS PREDICTIVE SITUATIONAL AWARENESS OF URBAN PEDESTRIAN FLOWS: AN ASSESSMENT OF DATA QUALITY ON PERFORMANCE METRICS

Carrow Morris-Wiltshire

August 2023

DECLARATION

"I hereby certify that this work is my own, except where otherwise acknowledge, and that it has not been submitted previously for a degree at this, or any other university".

[illegible]

Carrow Morris-Wiltshire

ACKNOWLEDGEMENTS

I would like to thank my supervisors; Prof. Stuart Barr, Prof. Phil James, and Keith Hermiston from DSTL, for their support, guidance and encouragement throughout this project.

I would also like to express my gratitude to the CDT team, particularly Prof. Jon Mills and the CDT manager Jamie Stogden for their continued support throughout the academic year.

This research was funded by the Defence Science and Technology Laboratory (DSTL) under project number RES/0539/7397.

ABSTRACT

The modern urban environment, a dynamic and ever-evolving system, calls for enhanced situational awareness to aid effective decision-making. This thesis focuses on how the quality of data obtained from IoT-enabled sensors influences our ability to monitor and predict urban pedestrian flows. By filling a critical gap in current research, this work seeks to improve our understanding of the behaviour and management of crowds in an urban context.

This thesis provides an exploration of anomaly prediction within pedestrian data and its interplay with data quality's influence on model precision. An investigation covering 1310 models states is undertaken. Key findings and narratives from distinct test objectives have been derived. The implications of this research extend to large-scale urban data repositories, such as urban observatories, marking a significant stride toward harnessing the potential of data-driven situational awareness.

The main findings in this study from the models that were tested is:that high data quality appears to be important for predicting anomalies but not creating models that make accurate prediction. There is a significant amout of futher work that needs to be done in order to understand the effect of data quality on anomaly detection, and recommendation for further work are discussed.

Contents

1 INTRODUCTION

1.1 Motivation

The modern urban environment, a dynamic and ever-evolving system, calls for enhanced situational awareness to aid effective decision-making. This thesis focuses on how the quality of data obtained from IoT-enabled sensors influences our ability to monitor and predict urban pedestrian flows. By filling a critical gap in current research, this work seeks to improve our understanding of the behaviour and management of crowds in an urban context.

Situational awareness, as defined by Endsley (1995), involves understanding and predicting the state of the environment through real-time data assimilation. This capability is vital for managing urban complexities and risks, as the effectiveness of interventions is intrinsically linked to understanding real-time situations. Supported by artificial intelligence (AI) and big data, data-driven decision-making has emerged as a critical tool for enhancing situational awareness in an urban context (MoD, 2022).

The formation of crowds is difficult to predict beyond the short term, and overcrowding events can have dramatic consequences, resulting in hundreds of injuries and fatalities each year (Sharma et al., 2023). As the world becomes more urbanised, and large-scale gatherings occur more frequently, monitoring pedestrian flows becomes increasingly important. IoT-enabled sensor networks offer enormous potential for the real-time monitoring of urban systems and potentially allow improved situational awareness of impending hazards and their impacts (Barr et al., 2020).

1.2 Background

1.2.1 Urban Modelling

The inception of urban modelling arose from the need to evaluate the impact of large-scale public investments on cities. Over the course of the 20th century, urban modelling has evolved significantly due to shifting paradigms. Driven by planning and policy requirements, and our increased understanding of human behaviour and the limits of computational reducibility, urban science has gradually transitioned from macro-statics (rooted in a mechanical worldview) to micro-dynamics (rooted in a complex systems worldview) (Wolfram, 2002, Batty, 2008). This shift in perspective to that of complex systems has introduced several new characteristics to urban modelling (Thurner et al., 2018), including:

- **Nonlinearity**: Slight changes can precipitate substantial effects, demonstrating that the relationship between cause and effect is not always proportional nor predictable.
- **Emergence**: The behaviour of the entire system cannot be predicted merely by understanding its components. Novel properties and behaviours can "emerge" at the system level.
- **Adaptation**: Complex systems have the capacity to learn from and adapt to their environment, making them dynamic and evolving rather than static.
- **Irreversibility**: Processes often have a specific direction and are irreversible — an egg, once scrambled, cannot be unscrambled.
- **Uncertainty**: It is impossible to predict the exact future state of a complex system; only probabilities of various outcomes can be determined.

Simulation methods like agent-based modelling (ABM) coupled with large compute and storage capabilities make it possible to model systems at an extremely fine level of detail that can capture some complex phenomena (Heppenstall et al., 2011, Alvarez Castro and Ford, 2021). The COVID-19 pandemic prompted a surge in research aimed at using ABMs to simulate interactions between individuals in cities (Hoertel et al., 2020, Cuevas, 2020, Silva et al., 2020, Hinch et al., 2021, Kerr et al., 2021, Alvarez Castro and Ford, 2021). Alvarez Castro and Ford (2021) for example, used an ABM to show quantitatively the effect of various mitigation strategies on COVID-19 transmission rates in university accommodation. Similarly, Cuevas (2020) used an ABM to simulate complex interactions between individuals to inform decision making for medical facilities.

However, a method for evaluating the results of such models remains elusive and is still an active area of research (Heppenstall et al., 2016, Kieu et al., 2022). Evaluation and validation of ABMs is crucial because these models primarily rely on historical data (Xie et al., 2020, Bai et al., 2021, Heppenstall et al., 2021, Tang and Malleson, 2022). Because human behaviour evolves as a result of new technology, models that rely solely on historical data diverge from reality at a rate related to the length of time since the data was recorded (Batty et al., 2012, Shi et al., 2021). One major difficulty identified by Malleson et al. (2019) lies in the mapping of real observations on to an ABM state space. The author concludes that future experiments should explore the spatio-temporal resolution of the aggregate pedestrian data required. This should include the number, and characteristics, of the sensors that would need to be installed to accurately simulate the target system.

Kieu et al, (2022) make progress in solving some of these issues using a time-series emulator for ‘real-time’ predictions on simulated data coupled with an ABM that models how pedestrians move through an environment. The authors acknowledge the lack of real data in evaluation, and that simulated data might not fully capture the dynamics of a target system compared to data derived from a real system.

1.2.2 The Role of Data Latency and Their Applications

Data latency refers to the length of time between the collection of data and the practical application of the results deriving from modelling/analysis procedures. Table 1 categorises the different levels of latency in data (Barlow, 2013), from machine-speed real-time to archive latency, and describes some of the possible applications in urban planning and response. These are important distinctions to make because the latency of data in ABMs is probably in the long-term planning category. Incorporating real and near-real-time data for ABMs unlocks better decision-making for applications requiring minimal latency.

Table 1 Data latency and example applications

Latency Level	Description and Example Applications
Immediate Response (Machine-speed Real-time)	Automated decision-making for rapid response such as traffic routing, congestion control, utility management. Minimal human intervention required.
Rapid Response (Near Real-time)	Predefined responses coordinated by humans for quick response planning e.g., short-term public transportation adjustments, law enforcement despatching.
Daily Planning (Short Latency)	Human-machine collaboration for decisions within a day such as daily route planning for waste management, minor infrastructure repair, park maintenance.
Weekly Planning (Medium Latency)	Automated suggestions with human review for decisions over a week such as neighbourhood resource allocation, parking regulations, public service planning.
Long-term Planning (Long Latency)	Human-led decisions aided by machines over weeks to months e.g., public space redesign, road construction planning, seasonal urban farming planning.
Extended Planning (Extended Latency)	Mainly human decision-making over years for urban development planning, zoning law changes, and public transportation network redesign.
Retrospective Analysis (Archive Latency)	No automation, long term historical data used for retrospective studies and long-term planning, like demographic changes, urban growth patterns.

1.2.3 Nowcasting and Data-Assimilation

Nowcasting and methods of data assimilation are two potential related methods that can be used to address the challenge of utilising real-time data within ABMs (Tang and Malleson, 2022, Ternes et al., 2022, Kieu et al., 2022, Heppenstall et al., 2021, Shi et al., 2021, Hunt et al., 2007). Nowcasting (a portmanteau of *now* and *forecasting*) is a process that involves making very short-term predictions about the current or immediate future state of a system, and is widely used in fields like meteorology and economics, that require rapid, real-time predictions (Giannone et al., 2008, Shi et al., 2017, Albani et al., 2022). Data assimilation is a statistical method that combines observational data with prior knowledge (usually from a predictive model), to estimate the current state and potentially the future evolution of a system. Data assimilation is a popular method for providing nowcast estimates as it can be used to fill in gaps where observational data is missing or sparse (Hunt et al., 2007, Wang et al., 2019, Shi et al., 2021).

1.2.4 The Application of Enhanced Situational Awareness in Crowd Control

Enhanced situational awareness arising from nowcasting and data-assimilation methods, applied to crowd control shows potential in risk management. Effective management of crowds can help prevent catastrophic incidents such as crowd crushes when critical density thresholds are surpassed (Haghani and Lovreglio, 2022), and support epidemic/pandemic response planning by limiting sudden outbreaks (Yadav et al., 2022). There are several concerns stemming from overcrowding events that can be mitigated through more effective management and response:

- **Physical injury –** high densities of people can lead to accidents and injuries, from minor ones like tripping and falling to major incidents like stampedes and trampling – resulting in the death of hundreds of people each year (Sharma et al., 2023).
- **Inadequate emergency response access –** it can be difficult for emergency personnel to respond effectively to incidents, where, in extreme cases, emergency exits may become blocked, or individuals may not be able to evacuate safely in emergencies such as fires or natural disasters (Akinwande et al., 2015).
- **Disease transmission** - rapid spread of infectious diseases which is particularly concerning during an epidemic/pandemic or outbreak situation (Yadav et al., 2022).
- **Social disruption** - increases in anti-social behaviour, crime, or violence, and the disturbance of local communities.
- **Infrastructure stress** - accelerated wear and tear of the built-environment occasionally leading to system failure e.g., public transport.

1.3 Practical Considerations

There are two underlying challenges to nowcasting. These are questions of *temporal dynamics*, or how the recorded values at a sensor changes over time; and *spatial interaction*, or how strongly sensor locations interact with one another (based on the sensor values) (Batty, 2013). The urban observatory data offers an opportunity to improve insight into these changes by using pedestrian flow data. The pedestrian flow data on the high street is collected at relatively frequent intervals and should not only offer insight into the temporal dynamics but open the door to future research that would give some indication of the strength of spatial interaction between different points in the city.

1.4 Summary

To encapsulate the complexity inherent in cities, it is necessary to merge our simulated model of the observed urban system with data that can calibrate and validate the simulation. Encouragingly, the past decade has witnessed a significant increase in the number of IoT devices, many of which generate 'big data'. *Big data* — characterised by the five Vs: *volume*, *velocity*, *variety, veracity and value* — is vital for real-time applications in complex environments such as pedestrian networks (Batty et al., 2012, Batty and Milton, 2021). The temporal frequency and depth of the Urban Observatory data offers an opportunity to investigate some of these complex temporal dynamics, but to capture some of the micro dynamics, methods for pattern recognition are required.

1.5 Aims and objectives

1.5.1 Primary research aim

The aim of this project is to build a roadmap for creating a model that detects the emergent behaviour of overcrowding events, through fast, incremental evaluation of data that is scalable and deployable for real-time applications. This would enable rapid identification, and targeted response to overcrowding events before they escalate into emergencies. The objective here is to close the OODA loop (Observe, Orient, Decide, React) and is enabled by the utilisation of real-time data (Boyd, 1996). However, leveraging real-time data for effective event detection and intervention is not without its challenges. The first of these challenges is understanding what quality of data is needed to make useful predictions which forms the primary research question for this thesis.

1.5.2 Research questions

How does the quality of IoT pedestrian data affect our ability to make predictions?

This is the main question that is addressed in this thesis. It is further broken down into three separate objectives.

1. *Can we detect anomalies without pre-classification and are there any patterns to the anomalies?*
2. *Does increasing the number of input features improve the performance of the model?*
3. *How does perturbing the input data affect the results of the model?*

1.5.3 Deliverables

1. A review of existing literature for the subject area.
2. A comprehensive exploratory data analysis report.
3. A methodology explaining the approach taken by this research
4. Recommendations and future work based on the results of the analysis.

2 LITERATURE REVIEW

In this section a variety of methods are discussed that relate to the research questions discussed in the previous section. This section will start with a brief overview of how pedestrian behaviour has been recorded and modelled in the past and introduce some of the data-driven approaches necessary for situational awareness. It will then look at how temporal patterns can be extracted from time series data and the different approaches that have been used to convert extracted patterns into predictions. Finally, it will discuss clustering and anomaly detection methods.

2.1 Modelling Pedestrian Behaviour

Helbing and Molnar (1995) first pioneered the ‘social force model’ approach that treats pedestrians as individual, intelligent agents who are influenced by a combination of physical forces and social factors. This work has provided the foundation for many later studies developing theoretical models that seek to describe pedestrian dynamics and behaviour. Over the last decade, as datasets on crowd behaviour are becoming readily available, many of these models which have previously not been validated can now be tested using real-world datasets (Draghici and Steen, 2019).

2.1.1 Data-Driven Approaches

Much of the academic literature around crowd behaviour focuses on measuring the dynamics of a crowd, from a video scene, or from application data using metrics like density and speed. For video sensors, detecting crowds, is largely a computer vision problem, and this modelling approach can detect crowding when certain speed/density thresholds have been crossed (Kretz, 2011, Wirz et al., 2013). Draghici and Steen (2019) talk about spatiotemporal features necessary for capturing crowd behaviour such as densities and movements . According to Still (2019) densities of up to 3 people per square metre with a flow of up to 82 people/metre/minute constitute a low risk crowd. Whilst being able to accurately calculate the risk of crowding in a given scene is useful, it is difficult to conceive how techniques that involve measuring the properties of a crowded video scene would allow for predictions of crowding events with an appropriate timeframe in which to coordinate a response. To generate predictions of when crowds may form, it is necessary to aggregate this data so patterns that lead to crowding events can be identified through spatiotemporal analysis on a wider scale (Hoang et al., 2016, Xie et al., 2020, Jiang et al., 2022). The challenge presented in this thesis is around understanding the *temporal* dynamics that give rise to crowding events (anomalous flow detection using forecasting) rather than trying to analyse the risk as one of the events *takes place*.

A range of data sources have been used to predict flow using aggregated data. Automated methods for collecting pedestrian flow data can be classified into infrastructure-based (network) and application-driven (devices) (Bamaqa et al., 2022). Application-driven data collection makes use of mobile on-board (visual and non-visual) sensing (e.g., GPS for location and accelerometer for motion) or software sensing (e.g., Twitter) to provide real-time updates of crowd conditions such as location, speed and direction (Felemban et al., 2020, Mohamed et al., 2019).

Many approaches to modelling crowding involve mobile phone datasets (Fu et al., 2021, Cecaj et al., 2020, Li et al., 2022). While mobile phone datasets provide a rich dataset to train a model on, it is not possible to receive this data in a real-time stream as there are many practical, legal and ethical problems standing in the way (Taylor, 2016). Whilst there are certain applications where real-time collection of location is feasible such as large organised events like pilgrimages where attendees might download an app that collects their location data (Felemban et al., 2020). In most other instances automated collection of real-time data largely is limited to stationary sensors. There are many approaches that use data fusion highlighted by (Li et al., 2021), which seek to combine application data with infrastructure-based data sources. However, there has been little research into how far infrastructure-based data sources can go in solving the prediction problem alone.

Pedestrian sensors are becoming more prevalent in cities (cameras using object detection algorithms to yield aggregated data) and there are an increasing number of public and private services providing centralised, data repositories that make this data accessible (CoM, 2023, UO, 2023, VivaCity, 2023). The depth of temporal data has passed the threshold required to train reasonably accurate machine learning models, and there is an increasing amount of research utilising this data to develop predictive models (Peppa et al., 2021, Asher et al., 2023). The advantage of data from pedestrian sensors is that it is relatively temporally complete, providing a rich source of information from which to extract the complex patterns and micro dynamics discussed in the previous section.

As research into the utilisation of pedestrian flow data is still relatively limited, the literature review will investigate research into *urban flow* predictions. This more general term encapsulates all agents that move about an urban area with autonomy (cars, bikes etc. but not bus, trains etc.). The temporal patterns of the these urban flows are subject to the same characteristics discussed in [BACKGROUND – URBAN MODELLING] (Shi et al., 2021).

2.1.2 Situational Awareness

Turning predictive models powered by real-time data into a valuable tool for decision making requires overcoming some other significant problems that lie at the heart of the situational awareness. As we have discussed, we can roughly categorise existing studies on pedestrian volume estimation into two main approaches: data-driven prediction and model-driven prediction (Li and Wu, 2021, Jiang et al., 2022). Data-driven prediction commonly relies on static sensors like CCTV and traffic cameras, producing data specific to these locations (Peppa et al., 2018, Ide et al., 2017, Chen et al., 2021). However, to generate a comprehensive and timely estimation of pedestrian volume i.e., *nowcasting*, a combination of data-driven and model-driven approaches will be necessary.

This combination introduces analytical complexities due to the inherent non-Gaussianity, high dimensionality, and nonlinearity in the data. For example, pedestrian flows can exhibit non-Gaussian distributions, variables such as time of day or weather conditions introduce high dimensionality, and complex interactions between pedestrians and their environment add nonlinearity. To tackle these complexities, approximation methods such as the extended Kalman filter, Gaussian sum approximations, grid-based filters, and Sequential Monte Carlo have been proposed. However, the first two methods often fail to capture all the critical statistical features of the processes, leading to suboptimal results (Doucet et al., 2001). While grid-based filters, which rely on deterministic numerical integration methods, can yield accurate results, they are challenging to implement and computationally demanding, particularly when dealing with a high number of variables or 'dimensions' (Doucet et al., 2001).

2.2 Temporal Pattern Extraction

Temporal pattern extraction refers to the process of identifying and analysing patterns or regularities in data over time. These patterns can provide valuable insights into the underlying processes that generate the data, enabling predictions about future events or explaining past occurrences (Chatfield, 2003). The extracted patterns can take many forms such as:

- **Trends** - increasing or decreasing values over time such as urban development, population growth, and changing habits.
- **Seasonality** - patterns that repeat at regular intervals such as holiday patterns or school/university term dates.
- **Cycles** - patterns that occur over irregular, often longer, periods (Hyndman and Athanasopoulos, 2018) such as city-wide festivals or sporting events.
- **Time series motifs** - recurrent sub-sequences in the time series data (Lonardi and Patel, 2002) such as a surge in pedestrians during lunch hours (12-2pm) and after work (5-7pm) on weekdays.
- **Time-delay embedding structures** - current data is seen to affect subsequent data for a short period (Kantz and Schreiber, 2004) for example, paydays or specific sale days might influence counts on subsequent days.

Understanding the temporal patterns in pedestrian data is a critical part of the data mining process needed to generate useful insight about risk. Lin et al. (2003) give a summary of the time-series data mining problems that are solved by pattern extraction (for more complete definitions of the following term see [GLOSSARY]):

- **Indexing**: Given a query time series Q, and some similarity/dissimilarity measure $D(Q,C)$, find the most similar time series in database DB.
- **Clustering**: Find natural groupings of the time series in database DB under some similarity/dissimilarity measure $D(Q,C)$.
- **Classification**: Given an unlabelled time series Q, assign it to one of two or more predefined classes.
- **Anomaly Detection**: Given a time series Q, and some model of "normal" behaviour, find all sections of Q which contain anomalies, or unexpected behaviour.

The tasks outlined by Lin et al. (2003) set the groundwork for understanding and extracting patterns from time-series data. Early prediction methods attempted to capture patterns through dimensionality reduction (Figure 1). While more computationally efficient, these methods may not fully capture more intricate patterns within the data.

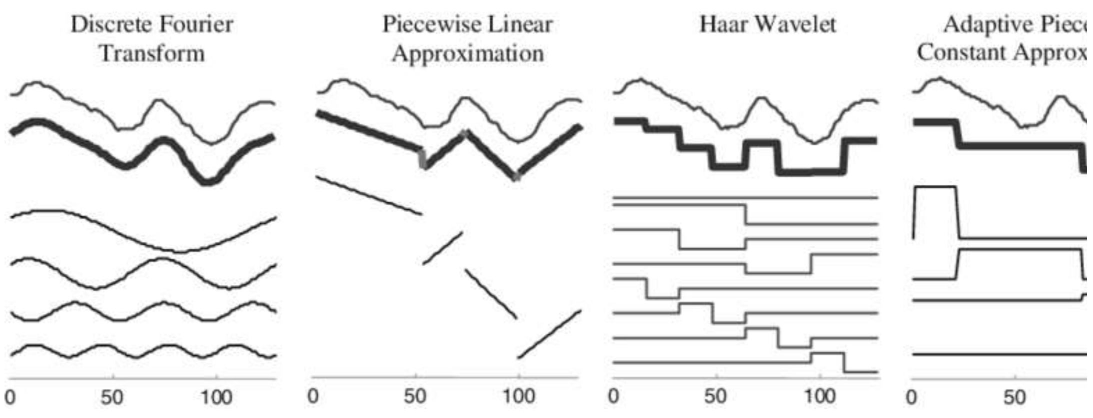


Figure 1 The most common representations for time series data (Lin et al., 2003)

With the growing complexity of time series data, especially in the realm of big data, the necessity for methods capable of handling higher dimensionality and complex patterns has risen. Consider pedestrian behaviour on a busy city street - dimensionality reduction techniques might capture the overall patterns of foot traffic - such as increased activity during lunch hours - but they might fail to capture more intricate patterns, such as the interplay between weather, organised events, and the day of the week, which could drastically influence pedestrian counts. These subtle, interdependent patterns often require the flexibility and nuance that machine learning models provide, and why they have started to supersede traditional approaches in building reliable predictive models (Längkvist et al., 2014, Wang et al., 2017).

However, it is not always the case, that bigger equals better, when it comes to modelling. Makridakis et al. (2018) find that for single-step forecasts, statistical models like ARIMA outperform machine learning models in terms of accuracy when tested on benchmark datasets. The authors lay out common pitfalls that result in poor forecasting performance and their research highlights the importance of baselining any non-linear black-box machine learning model against a linear and interpretable statistical one.

2.2.1 Classical Machine Learning Approaches

In recent years there has been a burgeoning field of research that seeks to apply machine learning methods (ML) to vehicular traffic prediction for which a multitude of approaches have been tested (Dai et al., 2020, Xie et al., 2020, Peppa et al., 2021). Classic ML approaches predict the traffic condition by learning the statistical regularity from historical traffic data. However, these approaches can sometimes ignore the temporal features of traffic data (Wu et al., 2004, Sun et al., 2006, Dai et al., 2020). This can occur for the following reasons:

- **Lack of temporal ordering -** Classic machine learning models lack an internal mechanism to remember and leverage past information (Bagnall et al., 2017).
- **Stationarity assumption** - Many real-world time series data exhibit non-stationary behaviours, such as trends and seasonality, which classic ML models might fail to handle appropriately (Hyndman and Athanasopoulos, 2018).
- **Lack of long-term dependency recognition -** An event that occurred ten time steps in the past may influence the current observation, but many models struggle to identify and learn from these relationships (Bengio et al., 1994).

- **Inadequate feature engineering** - Without careful feature engineering (which can be complex and time-consuming), classic ML models might fail to capture essential temporal features like seasonality, cyclical patterns, trends, etc (Guyon and Elisseeff, 2003).

Whilst there are some cases where a classical approach like Random Forest outperformed more complex models (Peppa et al., 2021), a more common approach is to use a more complex machine learning model like the Recurrent Neural Network (RNN). RNNs are considered the natural choice for dealing with time-series data as they can incorporate *hidden states*, which capture information about what has been seen so far in the data sequence. The 'memory' maintained by an RNN’s hidden state helps it capture these dynamics, making it capable of understanding temporal dependencies, such as the influence of past events on future ones (see [GLOSSARY]) (Sutton, 1988, Peddinti et al., 2015).

2.2.2 Deep Learning Approaches

Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) (see [GLOSSARY]) are both popular types of RNN as they include mechanisms to store and access information over longer periods of time. They address the vanishing or exploding gradient problem that basic RNNs are subject to. These problems occur because RNNs process sequence data iteratively (the current output is dependent on the previous computations). This means that the gradients, which are used to update the model's weights, tend to get exponentially smaller or larger as they are backpropagated through time (depending on whether the weight of the recurrent connections are less than or greater than one) (Pascanu et al., 2013). For vanishing gradients this means that the model ‘forgets’ long term dependencies and exploding gradients cause model instability and convergence failure. LSTMs use a memory cell that includes gates to control the flow of information into and out of the cell, which helps to mitigate the vanishing gradients problem and allows the network to learn longer-term dependencies (Hochreiter and

Schmidhuber, 1997). The exploding gradient problem can be mitigated by gradient clipping (limiting the maximum value of the gradient) (Bengio et al., 1994).

LSTMs are used extensively in urban flow prediction (Wu and Lin, 2019, Du et al., 2020, Jing et al., 2021). Dai et al. (2020) propose a framework for capturing both the temporal dynamics and the spatial complexity of traffic data using an LSTM/GCN architecture to make predictions. The results of their combined architecture (trained on both spatial and temporal features) outperformed the GCN (trained solely on spatial feature), the LSTM, and T-GCN (both trained solely on temporal features). Of the two models trained on the temporal features the LSTM performed the best. Fu et al. (2016) similarly compare the effectiveness of two models, LSTM and GRU for short-term traffic speed forecasting. The authors found that both networks outperformed the baseline ARIMA model in terms of MAE and MSE, but the simpler GRU architecture outperformed the LSTM. It is however, not clear how much data was used in the training process, only that the LSTM and GRU were trained for the same number of epochs. Some research utilises convolutional neural networks (CNNs) to capture the spatial features and GRU for the temporal aspects of traffic data. It found that despite the CNN's usefulness in Euclidean spaces, it fell short in capturing the spatial features of complex topological transport networks (Cao et al., 2017).

2.3 Anomaly Detection

A useful distinction between types of time-series anomalies can be achieved by categorising them from two perspectives: granularity and behavioural characteristics is worth noting (Blázquez-García et al., 2021, Choi et al., 2021, Cook et al., 2019):

From the granularity perspective:

- **Point Anomalies**: These are individual data points significantly deviating from others in the time series or their local neighbours in a particular time frame. Causes can include noise, sensor malfunctions, or short-term system disruptions.

- **Subsequence Anomalies**: These are sets of consecutive observations within the time series that deviate from expected patterns. However, each point within these subsequence's may be within the expected range when examined individually.
- **Sequence Anomalies**: In the context of multivariate input data, a sequence anomaly is a univariate time series that behaves significantly differently from others.

From the behavioural perspective:

- **Point Anomalies**: Like the point anomalies in granularity, these are observations or sequences that abruptly deviate from the normal state of the entire dataset.
- **Contextual Anomalies**: These are observations or sequences that may not globally deviate from the normal range but are unexpected when considering the given context.
- **Collective Anomalies**: These are sets of observations that show unique patterns relative to the rest of the data.

A recent survey on unsupervised ML discusses a unified perspective on applications for anomaly detection and clustering, both of which are typically unsupervised methods for mining data patterns (Liu et al., 2023).

Table 2 (Blázquez-García et al., 2021)

	Data used	→	Expected value	→	Point outliers
Estimation models	$[\boldsymbol{x_{t-k_1}}, \dots, \boldsymbol{x_t}, \dots, \boldsymbol{x_{t+k_2}}]$	→	$\hat{x}_t$	→	$\lvert x_t - \hat{x}_t \rvert > \tau$
Prediction models	$[x_{t-k}, \dots, x_{t-1}]$	→	$\hat{x}_t$	→	

Detection of point outliers in univariate time-series

Blázquez-García et al. (2021) discuss detection of outlier detection for streaming time series. The authors define techniques that apply to streaming data as those that are able to detect outliers by determining whether or not a new incoming datum is an outlier as soon as it arrives without having to wait for more data. They also highlight that although there are many techniques that can theoretically deal with streaming time-series, very few are able to adapt incrementally to the evolution of the stream. The simplest estimation models use basic statistics like the median or median absolute deviation (MAD) to obtain $\hat{x}_t$ using either the full series or grouping the data into equal length segments (Mehrang et al., 2015). Dani et al. (2015) use the mean of each segment to determine the expected value of the points within that segment, and an adaptive threshold $\tau_i = \alpha\sigma_i$,where α is a fixed value and σ_i is the standard deviation of segment i. Prediction models that retrain the model as the time-series evolves: Zhou et al. [2018a] fit an ARIMA model within a sliding window to compute the prediction interval, so the parameters are refitted each time that the window moves a step forward. This is a useful feature of anomaly detection as it ensures the model continuously adapts to new data.

2.3.1 Clustering Methods

Clustering is one of the most used unsupervised learning algorithms. The goal of clustering is to organise objects into homogeneous groups where the intra-group similarities are maximized and the inter-group similarities are minimised (Chen et al., 1996). Traditional clustering methods generally fall into five categories: partitioning, hierarchical, density-based, grid-based, and model-based methods (Jain, 2010). However, time-series data presents challenges to traditional clustering methods due to its sequential nature and possible dependence on time lags. High dimensionality, noise, and the temporal dependencies in the data can make clustering based on raw values problematic (Aghabozorgi et al., 2015).

Based on the literature, either dynamic time warping (DTW) or Mahalanobis seem to be sensible choices for a specific distance metric (Ding et al., 2008). DTW is a mapping of points between a pair of time series designed to minimise the pairwise Euclidean distance and is used for comparing sequences or time series data, especially when sequences can differ in length (Mei et al., 2016). Mahalanobis distance is a measure of the distance between a variable and a distribution which is calculated by a mean and the covariance matrix. It is used for determining the "distance" of an observation from a group of observations with multivariate normal distribution, taking into account the correlations of the variables (Sitaram et al., 2015). Both methods allow for multivariate time-series to be used and crucially do not require that lengths of all time-series are equal, which is unusual in real data (Liu et al., 2023).

There are two major design criteria in clustering methods, the clustering algorithm, and the distance metric. Javed et al. carry out a benchmark study using eight clustering algorithms (partitional, density-based and hierarchical) and three distance metrics, (Euclidean, dynamic time-warping and shape based). The authors concluded that it is necessary to test a pool of clustering algorithms during exploratory data analysis (EDA) as the accuracy of different algorithms is highly sensitive to the particular time-series dataset (Javed et al., 2020). It is worth emphasizing that in time-series clustering, the choice of distance metric can significantly influence the resulting clusters,

Belhadi et al. investigate a space-time series clustering algorithms, the state of the art, and their limitations. The conclusions drawn by the authors come down to four challenges that need to be solved (Belhadi et al., 2020):

- Run-time of algorithms
- Quality of performance for complex and big time series data
- Correlation between space-time series data
- Adaptation of advanced and specialised clustering techniques

2.4 Summary

1. Anomalies in time series can be categorised based on their granularity (point, subsequence, sequence) and behavior (point, contextual, collective).
2. Traditional clustering methods face challenges with time-series data due to issues like high dimensionality, noise, and temporal dependencies.
3. Classical machine learning approaches to traffic prediction can sometimes overlook the temporal features of traffic data due to reasons like lack of temporal ordering, assumption of stationarity, inadequate feature engineering, and inability to recognise long-term dependencies.
4. Despite their potential, deep learning models like LSTMs may not always outperform traditional statistical models, highlighting the need for benchmarking.
5. Classical machine learning approaches can sometimes struggle with recognizing long-term dependencies in data and may require complex feature engineering to account for temporal patterns.

3 METHODOLOGY

The methodology is split into four sections. The first section will cover how the project can be accessed, what software is used, and how it was implemented. The second section will cover the data sources used in this project. The third section will highlight the cleaning and preprocessing methods used to prepare the data for modelling. The final section will show the different modelling workflows and give a rationale for selecting each of the models and evaluation metrics, as well as a brief explanation of the model architecture and evaluation metrics.

3.1 Research Objectives

The research objectives for this project are broadly split into three parts. The first part is anomaly prediction, which aims to test a range wide range of trained models on their ability to detect anomalies in a test set. Two model architectures are used, a simple

LSTM and a linear model, with a large set of hyperparameters that applied to them to produce differently trained models. The state of the model is then saved and evaluated on another set of hyperparameters (more detail in 3.5).

For anomaly prediction requires having a trained model that can make predictions, and then a mechanism for flagging anomalies when are real observation is outside of the normal range of predictions. For this to be effective, the model needs to be able to make predictions with low error. The next test compares the effect of increasing the prediction horizon on the model error. For reasons that should be clear, a larger horizon means greater error in the predictions, but the purpose of this test is to find the largest horizon value that a model still makes predictions with low enough error. The final test covers data robustness and looks at the effect of the varying data completeness on the model's prediction error.

3.1.1 Anomaly prediction test

1. Develop a modelling workflow that identifies anomalies in the data using single-step univariate prediction validating on unseen data.
2. Generate additional input features (feature engineering) to assess how the model's performance changes for anomaly prediction.

3.1.2 Horizon test

3. Measure the change in prediction accuracy as the prediction horizon increases for univariate and multivariate models.

3.1.3 Data robustness test

4. Measure the change in prediction accuracy as the data completeness of the training data is reduced for univariate and multivariate models.

3.2 Implementation

The entire reproducible codebase of this project can be found on the projects GitHub and can be accessed by cloning the repo and recreating the virtual environment. Alternatively, the project can be run directly using docker. The project backend uses Python 3 and a number of additional libraries were installed using pip (a full list of dependencies is provided). The full exploratory data analysis report is also hosted here using jupyterbook.

3.2.1 Software and Tools Used

For the data preprocessing and exploratory analysis, a standard workflow using NumPy, Pandas and Matplotlib has been followed. For the development of supervised machine learning models PyTorch was chosen due to its performance, interpretability, and ubiquity for machine learning research projects (PapersWithCode, 2023, Paszke et al., 2019). Scikit-learn has been used for further model evaluation (Pedregosa et al., 2011) and SciPy has been used extensively for additional statistical methods (Virtanen et al., 2020). Docker has been used to package the analysis to ensure reproducibility on most machines.

3.2.2 Coding and Algorithms

The different workflows described above are implemented in the script `model_training_loop.py.` The training loop can be used for both univariate and multivariate workflows and any combination of hyperparameters. The outputs are then saved to a folder, named in accordance with the hyperparameters used in the test.

In addition to the training loop, two modules have been created in this project to assist in the analysis and execute the modelling process called `eda_helper.py` and `modelling_functions.py` respectively. These modules can be found on GitHub with docstrings explaining the purpose of each function. Some of the most important parts of the script are included in 9.1 of the appendix, including the `LSTMModel` and `LinearModel` PyTorch objects and a number of functions relating to the construction of the custom `DataLoader` objects.

3.2.3 Computational Resources

All of the processing has been carried out using an Intel Core i7-1270P, 2200 MHz, 12 Core(s), 16 Logical Processor(s) with 32GB of RAM. A full list of runtimes for training each model can be found on GitHub. The `DataLoader` objects used only a single core so there is an opportunity to decrease the runtime.

3.3 Data Collection

This subsection covers the primary data source which are two pedestrian sensors located on Northumberland Street in Newcastle upon Tyne, UK. It will provide a brief overview of how the sensors record data, the data provider, and acquisition procedures before providing a description of the dataset. It will then briefly cover the auxiliary data source which is the term dates of the two major universities in Newcastle upon Tyne.

3.3.1 Pedestrian Sensors

The primary dataset used in this project is a csv dataset from the pedestrian sensors operated by the Urban Observatory. The Urban Observatory is based in Newcastle upon Tyne led by the Newcastle University and holds an extensive collection of real-time and historic data, from a wide range of sensors detecting everything from pedestrians and traffic to weather and air pollution. The data is open and freely available.

The pedestrian sensor collects data using object detection on video footage that identifies and counts the heads of individuals in motion. An algorithm computes the bearing of each detected head object, categorising its movement into one of sixteen possible directional vectors derived from the eight cardinal and intercardinal directions (for example, from northwest to southwest or south to east).

3.3.2 Data Acquisition

The raw data in this section was collated by a member of the Urban Observatory as an initial test dataset for this project and is stored as a single csv file which is available on GitHub. The data is also available through the Urban Observatory REST API and custom data loaders have been built for accessing the data this way during this project (available on GitHub).

3.3.3 Sensor Data

The data from two pedestrian sensors have been used in this project. Both have been collecting data intermittently over a period of roughly 3 years at 15-minute intervals in the same location but observing different sides of the street (Figure 2 and Figure 3). The pattern of intermittency is unique to each sensor shown in Figure 7. Table 3 to Table 5 show the features and an overview of the dataset.

The *dt* field contains the timestamp for the period of flow aggregation which is 15 minutes in the raw data. The *value* field contains the total number of pedestrians detected over the 15 minute period, and the *dir* field contains the direction of movement. For the sensors used in this project (shown in Figure 2) only the four cardinalities in the diagram are recorded (northwest to southwest, southwest to northwest, northeast to southeast, southeast to northeast). The west facing sensor only collects the first two vectors and the west facing sensor collects the last two.

The observations for *veh_class*, *location*, and *category* in this case are always 'person', 'NclNorthumberlandStSavilleRowWest' or 'NclNorthumberlandStSavilleRowEast', and 'flow' respectively (Table 4).

Table 3 shows a description of the raw dataset. There are just under 80,000 observations for each sensor, out of an expected 93,006 for the given time period meaning the time series is about 85% complete (missing data gaps are shown in Figure 7).

Table 3 Features and data types

Feature	Data type
dt	Object
value	int64
veh_class	Object
dir	Object
location	Object
category	Object

Table 4 Randomly sampled records from the raw dataset

dt	value	veh_class	dir	location	category
2022-08-18 02:45:00	4	person	southwest_to_northwest	NclNorthumberlandStSavilleRowWest	flow
2022-06-11 20:15:00	9	person	northwest_to_southwest	NclNorthumberlandStSavilleRowWest	flow
2022-03-02 23:15:00	9	person	northeast_to_southeast	NclNorthumberlandStSavilleRowEast	flow

Table 5 Description of the raw dataset

Direction	Length	Length if complete	Completeness	Start datetime	End datetime
East	79522	93006	85.5%	2021-12-02 14:00:00	2023-03-31 23:45:00
West	79787	93006	85.8%	2021-12-02 14:00:00	2023-03-31 23:45:00

Figure 2 shows the approximate field of view for both the sensors and the movement trajectories (shown as solid arrows). Figure 3 shows a picture of the two sensors used in this project, which are located on a lamppost in the middle of the street. Figure 4 shows the a map of Northumberland Street and the connecting side streets. It is worth noting that the street lies in the middle of the central business district, and is limited to pedestrians use only (no bicycles or scooters).

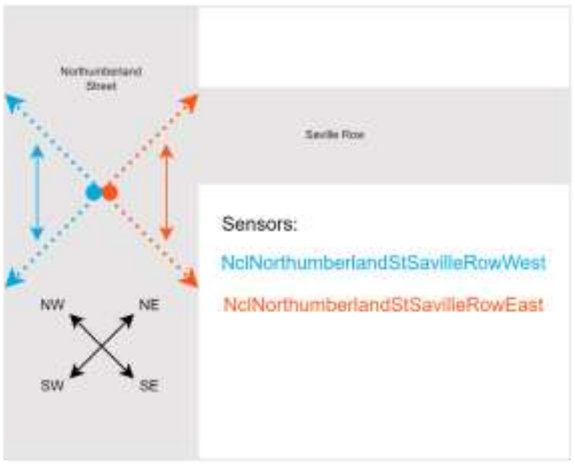

Northumberland Street
Saville Row
Sensors:
NclNorthumberlandStSavilleRowWest
NclNorthumberlandStSavilleRowEast
NW
NE
SW
SE

Figure 2 East and west facing sensor diagram

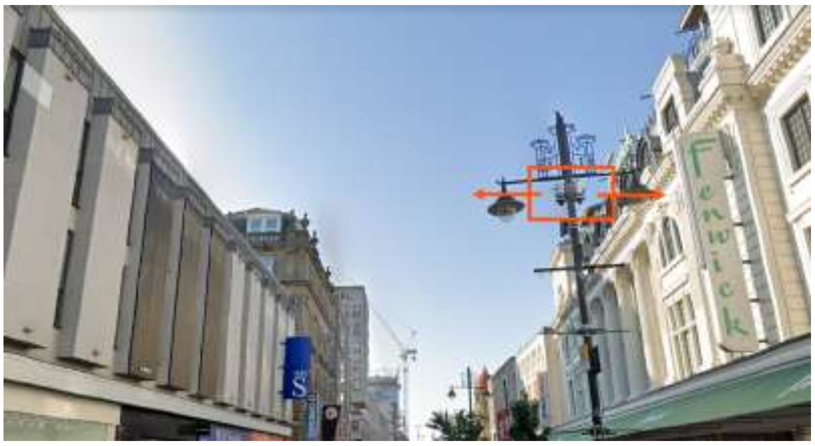


Figure 3 Pedestrian sensors on Northumberland Street

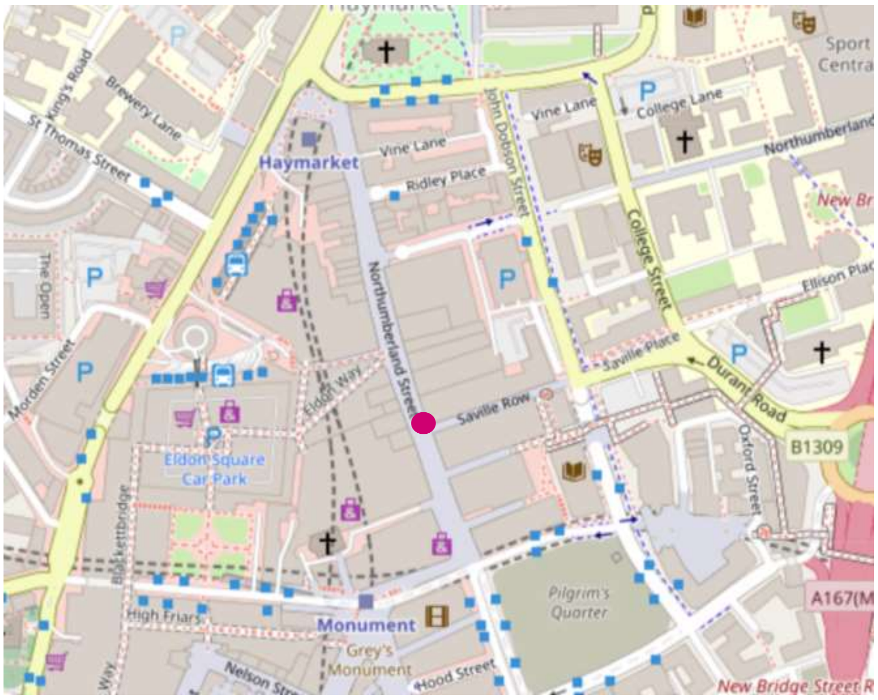


Figure 4 Map of Northumberland Street and the local area (pink circle is the sensors location)

3.3.4 Auxiliary Data Sources

Term time data from the Newcastle University and Northumbria University websites was extracted for use in the modelling process. This is for use in the multivariate modelling workflows (see section 3.4.5).

3.4 Data Preprocessing

The preprocessing subsection covers why cleaning and formatting is particularly important in time-series analysis. Then the full pre-processing workflow developed for this project is broken down step by step. This is followed by a more in-depth overview of the missing data handling, and the feature extraction and engineering that is used for the multivariate modelling process.

3.4.1 Requirements for Time Series Analysis

In time-series analysis it is important to use a continuous sequence of data for several reasons highlighted below. A continuous sequence in this context means data observations that are measured at successive points in time and at uniform intervals (15-minutes).

- **Temporal dependency** – there is an explicit order dependence between observations the structure of which provides an additional source of information (Chatfield, 2003).
- **Seasonality and trends –** repeating patterns and trends over time exist within most time series data and by training on a continuous sequence the model can learn these trends (frequencies can also be artificially added as input features to the model).
- **Autocorrelation –** time-series often exhibit autocorrelation, where a value is influenced by its predecessors, a feature which can only be captured in continuous data sequences.
- **Avoiding leakage –** using continuous sequences helps to ensure we are not using information from the future to predict the past which is important when considering train-test splits (trends in the data may be lost by evaluating the model using (test) data that predeceases the training data) (Hyndman and Athanasopoulos, 2018).
- **Model stability –** this refers to models' ability to generate consistent predictions over time and its sensitivity to slight changes in the input data and the models parameters. Stability relies on the model learning the trends rather than the noise in the data. Model stability can be assessed using cross-validation methods (Hyndman and Athanasopoulos, 2018).

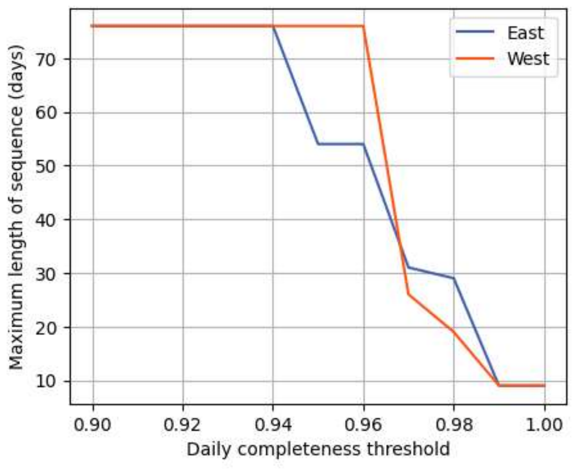
3.4.2 Preprocessing Workflow

The data is cleaned and formatted using the `preprocess_data()` function that can be found in 9.1.1 of the appendix. This function can implement any combination of the following cleaning and formatting steps depending on the workflow it is being used in:

1. Removing the directionality field *dir*. This function groups the data to a single bidirectional flow value on each timestamp. The original data contains two unidirectional flows with associated trajectories shown in Figure 2.
2. Selecting data that meets a daily completeness threshold. This function selects only observations that are made on a day that there at least a minimum threshold of observations for. This is discussed in more detail in the next section and the full docstring and function can be found in 9.1.2 of the appendix.
3. Finding the longest sequence of days that continuously meet the daily completeness threshold. This function performs a brute force search for the longest sequence, and times out when a longer sequence can't be found. The full docstring and function can be found in 9.1.3 of the appendix.
4. Adding auxiliary data (term dates) for multivariate analysis. This function reads in the auxiliary data and appends them to the sensor time series.
5. Adding engineered data (periodicity features) for multivariate analysis. This function creates the periodicity features and appends them to the timeseries. The periodicity features are calculated using a simple equation and which is explained in 9.1.4 of the appendix.
6. Scaling the value data field using a standard scaler from scikit learn. Using a standard scaler preserves the pattern of the data and allows for values outside of the range of the dataset. Any additional fields are normal (between 1 and -1).
7. Resampling the data to get average flow over a longer period, for example an hour or a day.

3.4.3 Missing Data Handling

Any days that do not contain enough data to be useful for training the models are removed. For most of the exploratory analysis a daily completeness threshold has been set at 100%. This is because most time-series analysis use continuous data sequences as machine learning models designed for learning time-series are particularly sensitive to gaps in the data. The function discussed in step three above then calculates the longest sequence of consecutive days in the data for at the chosen threshold. The effect of the threshold on the maximum sequence length that can be found is shown in the diagram in Figure 5 and Figure 6. At a 94% threshold we get a sequence length of 76 days for both datasets. Whereas at 100% threshold this drops to 9 days for both datasets. Figure 6 shows that at a 100% threshold <40% of the original data remains (>60% of the observations exist in days that do not have a complete set of observations). Whereas at 94% there is still >80% of the original data (<20% of the observations exist in days that do not have at least 94% of their observations). The pattern of this removal for three different thresholds is shown in Figure 7 - note the sudden jump in sparsity between 94% and 100% completeness.



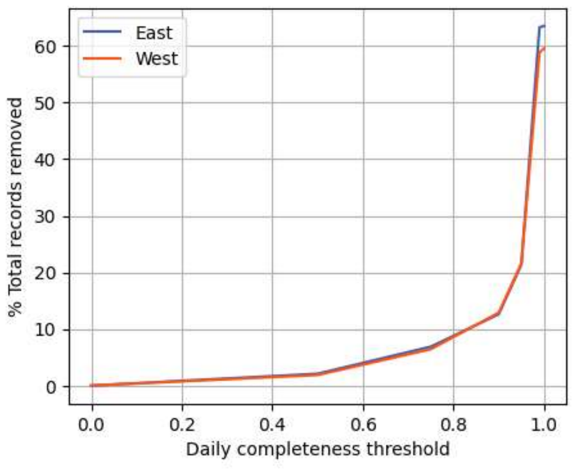
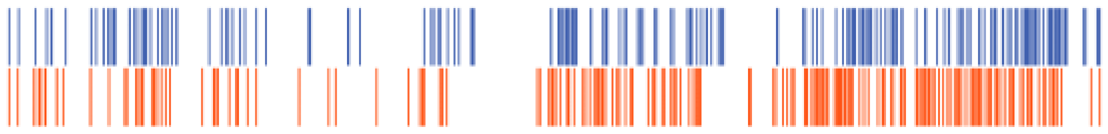


Figure 5 Effect of threshold on maximum sequence length

Figure 6 Effect of threshold on removal of total records



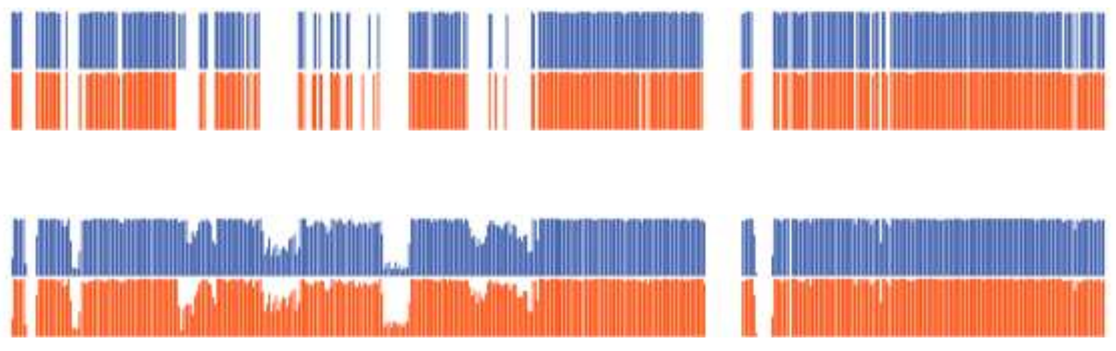


Figure 7 Total data at 100%, 94% and 0% thresholds (y axis – total records per day, x axis - date)

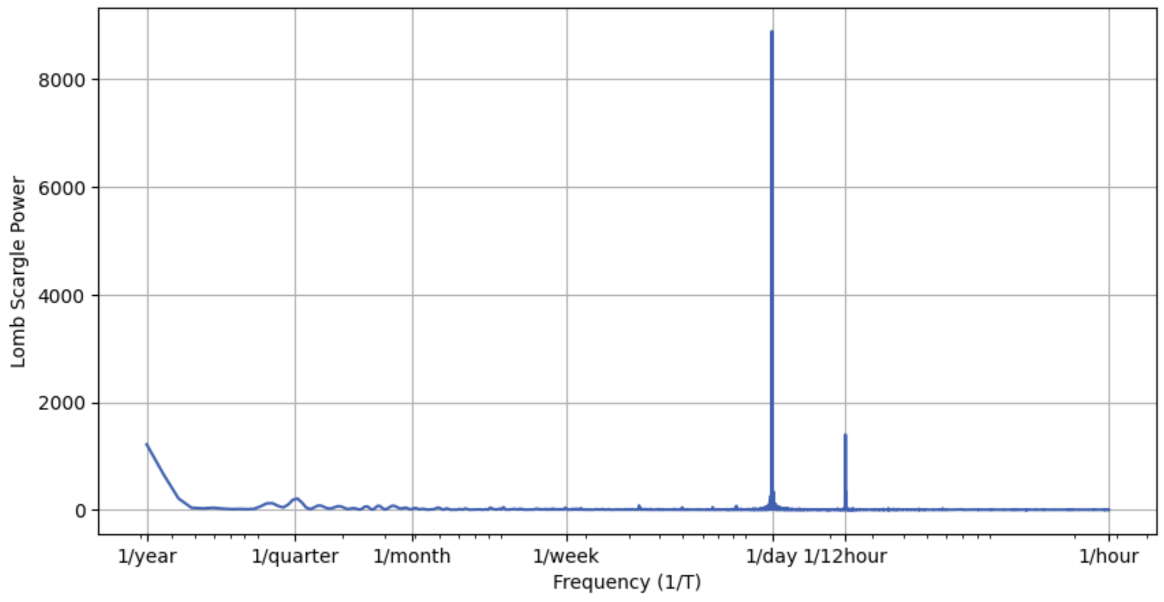
3.4.4 Feature Extraction

Feature extraction helps with data understanding and to identify patterns and features. The periodicity first needs to be extracted from the data so the most powerful frequencies can be identified. By creating periodicity features based on the most powerful signals they can be fed to the models to capture some of the patterns with less training. This is particularly important for linear models which have no capacity for storing temporal dependencies of data points. Though LSTMs are capable of capturing these patterns, including them as input features can help speed up model convergence. To extract the most powerful signals, a Fourier method will be used.

The Fourier Transform is a mathematical technique used in signal processing. It is a way of breaking down a complex signal into a set of simple sine waves of different frequencies. This allows the signal to be analysed in the frequency domain, which can reveal important features that are not readily apparent in the time domain (Bloomfield, 2004). One limitation of traditional Fourier methods, however, is that they assume the data is stationary - i.e., its properties do not change over time. This is not the case with the data used in this project and trends or changes in variance over time can be expected. The final criteria for method selection is that the data is incomplete, and for now, data imputation or interpolation is being avoided as this might introduce artificial patterns.

The Lomb-Scargle Periodogram (LSP) is a Fourier method which is implemented in the `scipy` library and computes a Fourier-like power spectrum estimator. LSP involves fitting a model to the data at each candidate frequency and selecting the frequency that maximizes the likelihood, which makes it more flexible and suitable for data that doesn't align neatly with whole cycles of sine waves. One of the key assumptions made by the Periodogram is that the noise is Gaussian and evenly distributed (VanderPlas, 2018).

The result of the analysis (Figure 8) using the LSP shows that the most powerful signals arose at the 12- and 24-hour periods. These are used to create additional input features in section 3.4.5. There are barely any perceptible differences in the LSP for both sensors except for the west having a slightly more powerful annual signal and slightly less powerful diurnal signal.



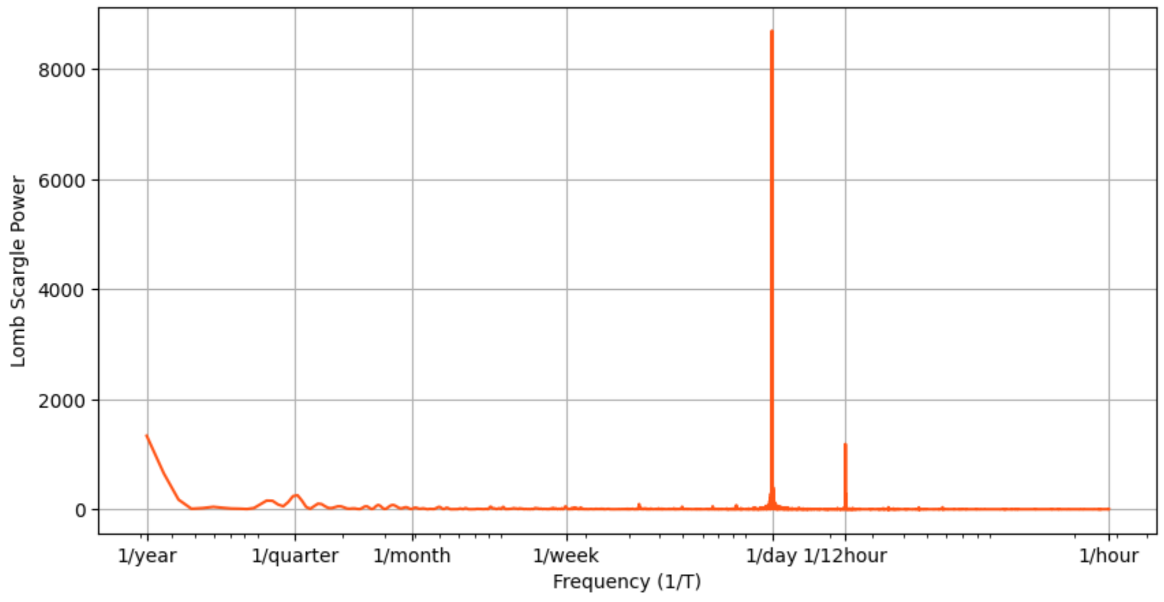


Figure 8 Lomb-Scargle Periodogram – East sensor (top) and West sensor (bottom)

3.4.5 Feature Engineering

Feature engineering can be useful to speed up model convergence, especially for models not capable of capturing temporal dependencies, like a linear model. Several features have been created for this purpose. Figure 9 shows an input window of length

100 with some of the engineered features. The periodicity features are calculated using the following formula

Equation 1 Periodicity

$$\mathrm{a} = \sin\left(\frac{2\pi * t_{x_a}}{24}\right)$$

where t_{x_a} is the frequency component of the datetime index for the period a (see 9.1.4 for further detail).

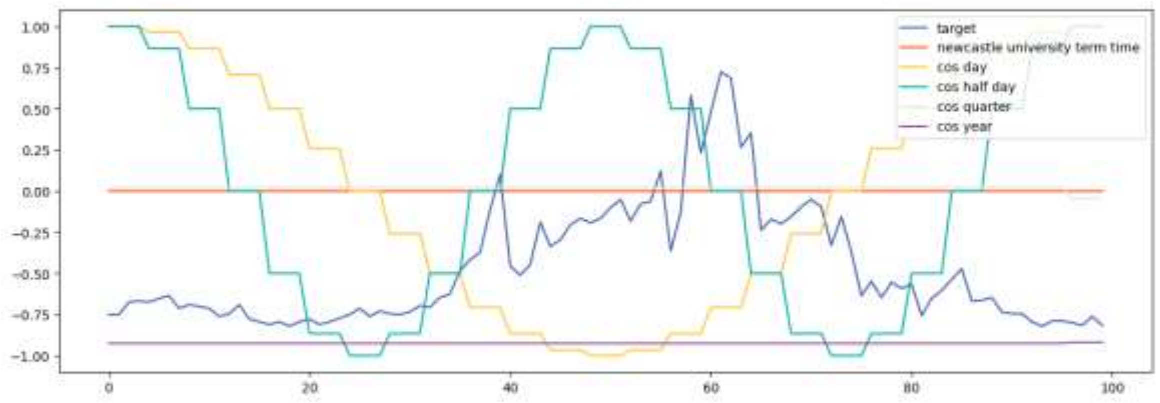


Figure 9 A selection of input features used in the modelling

Table 6 shows the maximum and minimum values for the scaled data. The data is scaled using the `StandardScaler` from the preprocessing module in the sci-kit learn library. It standardises a feature by removing the mean and scaling by the unit variance

Equation 2 Standard Scaler

$$z = (x - u)/s$$

where the mean and standard deviation of the series are u and s respectively, and x is each value in the series.

It can be seen in Table 6 the effect on the maximum and values for the data when the longest sequence at each completeness threshold has been selected. At 100% the range of values is smaller than at 94% as they contain 9 days worth of data and 72 days worth of data respectively.

Table 6 Max and min values of features after data preprocessing

	100%		94%	
name	max	min	Max	MIn
value	3.62	-0.83	3.64	-0.88
newcastle_term	0.00	0.00	1.00	0.00
northumbria_term	0.00	0.00	1.00	0.00
sin_day	1.00	-1.00	1.00	-1.00
cos_day	1.00	-1.00	1.00	-1.00
sin_half_day	1.00	-1.00	1.00	-1.00
cos_half_day	1.00	-1.00	1.00	-1.00
sin_quarter	1.00	0.86	1.00	-1.00
cos_quarter	0.02	-0.50	1.00	-1.00
sin_year	-0.38	-0.50	0.97	0.02
cos_year	-0.87	-0.93	1.00	0.26

Figure 10 shows the distribution values for the scaled *value* field.

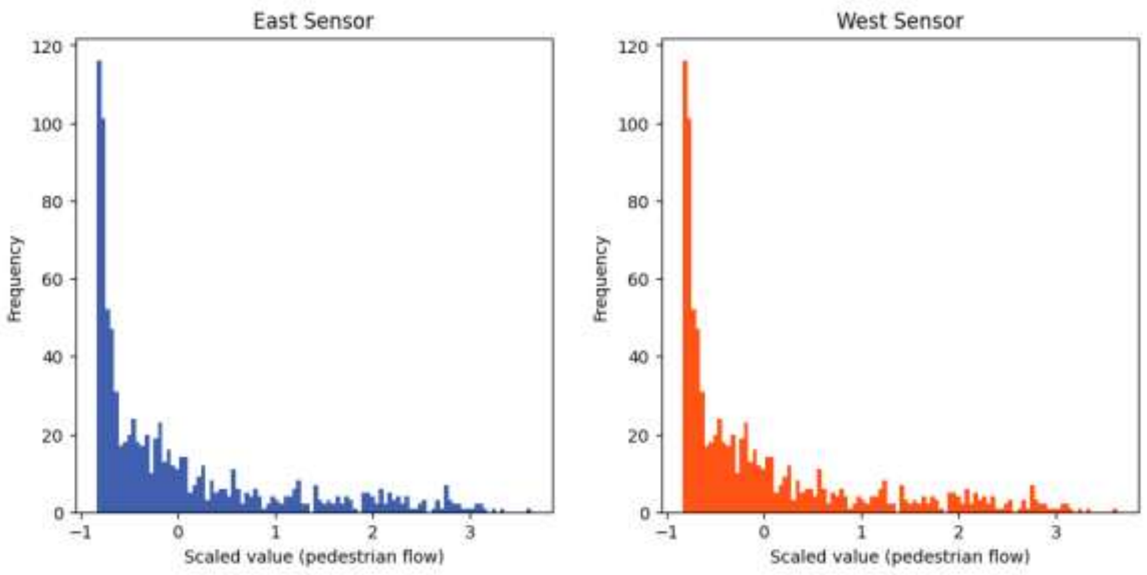


Figure 10 Distribution of standardised values

3.5 Modelling

This section will cover the modelling process, starting with the rationale for model selection and then providing an overview of the modelling and evaluation workflow (shown in Figure 14 and Figure 15 respectively).

3.5.1 Rationale for Model Selection

The linear model was chosen as it is the simplest conceivable model that is suitable for both univariate and multivariate problems. It offers a baseline performance to compare the LSTM against; and for the multivariate workflow, the weights assigned to each variable after training can be extracted. The decision to use both a univariate and multivariate workflow, is because univariate models are simpler to implement and faster to train but generally do not perform as well as multivariate equivalents. It is therefore useful to compare the results of with multivariate to see if the (potentially) increased performance is worth the extra training time.

The decision to use LSTMs is a result of the literature review. They are a special kind of RNN (see 8.2.5) that are often used in problems that involve sequences, including time-series data due to their architecture. Traditional RNNs suffer from the vanishing or exploding gradients problem, which makes them inefficient at learning long-term dependencies (see 2.2). LSTMs, however, are designed to mitigate this problem. They can remember and retrieve information over long sequences. LSTMs maintain a form of memory because they can remember or forget information in the sequence using their gates (input, forget, and output) which allows them to remember patterns spanning long time-periods (see Figure 11). They can be trained with standard backpropagation algorithms which enables end-to-end training of deep networks that include LSTMs.

However, LSTMs involve a more complex architecture and more parameters than traditional feed-forward networks or simpler RNNs. This complexity can increase the computational requirements (both in terms of memory and CPU/GPU time) and make LSTMs slower to train. They also often require large amounts of data to train effectively without overfitting, compared to simpler models. Additionally, like many other neural networks, LSTMs are "black boxes" and it can be difficult to interpret their internal state and understand exactly why they're making certain predictions. Future work may consider other models like the Transformer which show promising result for high-dimensional sequences where long-term dependencies are less important.

Figure 11 shows the different components within an LSTM cell and Figure 12 shows the architecture of the RNN. Each input x_t is passed through the LSTM cell which decides whether to update the model parameters for the next step (shown in Figure 12) or to leave it as the previous state.

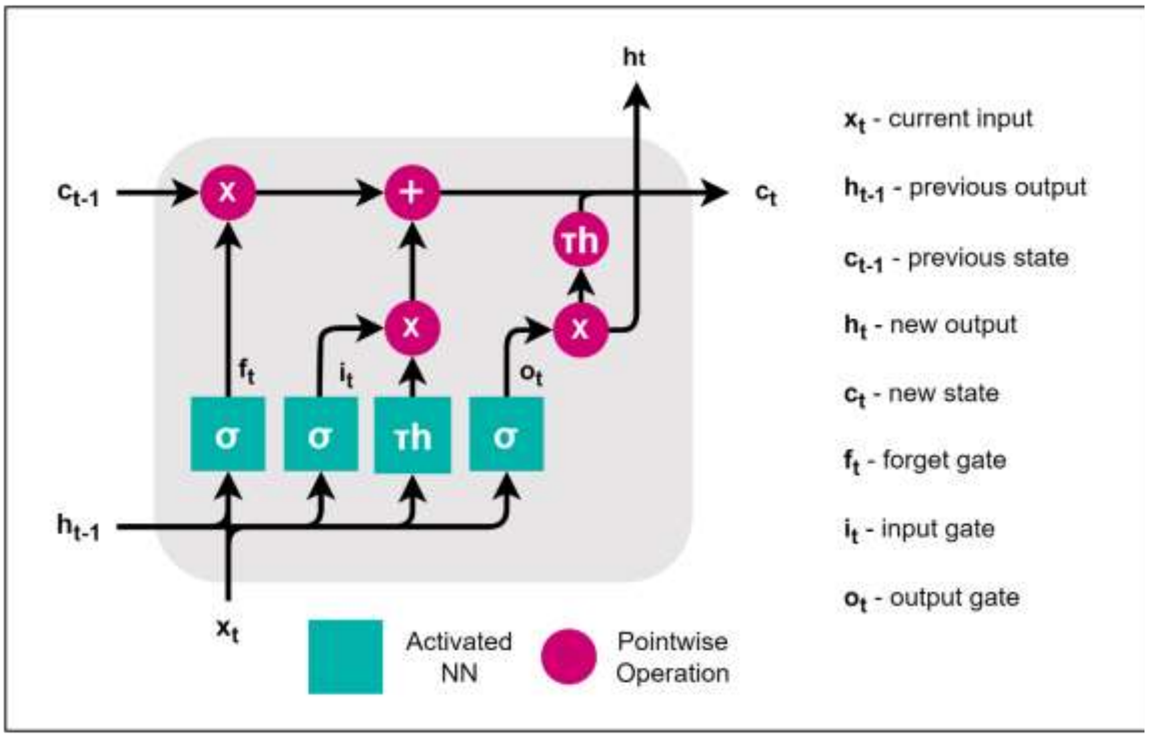


Figure 11 LSTM Cell

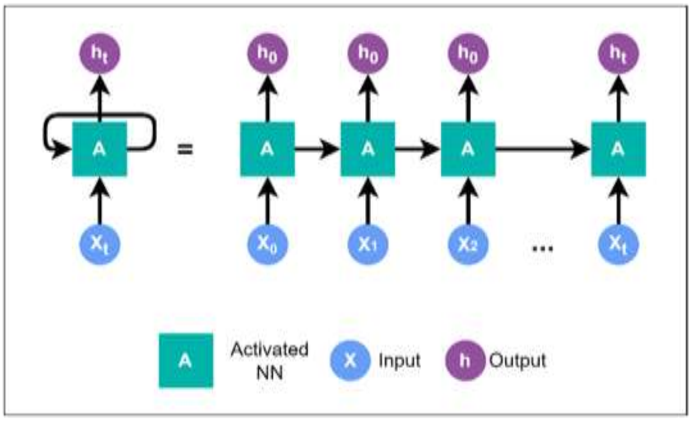


Figure 12 RNN Architecture

3.5.2 Overview

The linear and LSTM models both follow a univariate and multivariate workflow. This gives the following basic model types before hyperparameters are applied.

- Univariate linear
- Multivariate linear

- Univariate LSTM
- Multivariate LSTM

Table 7 provides a description of all of the hyperparameter used in the modelling. A test will be performed for every combination of numbers in this table, computing and storing a range metrics including the training loss for each epoch of the models, training summary statistics, the trained model state, linear model weights, and run time metrics. A total of 2,240 models have been trained and evaluated in this analysis, 320 of which were linear, and the rest LSTM. The reason for the difference in number of tests between models is because the linear model can only accept an input of length 1 (WINDOW_SIZE).

Table 7 Table of Hyperparameters

Hyperparameter	**Values**
COMPLETENESS_THRESHOLD	1.0, 0.98, 0.96, 0.94 or 1.0, 0.95, 0.90, 0.80
INPUT_INDICES	0 or [0,1,2,3,4,5,6,7,8,9,10]
TARGET_INDEX	0
WINDOW_SIZE	1 (linear) or 3, 6, 12, 24, 48, 96 (LSTM)
SEQUENCE_LENGTH	¼ , ½ , ¾ , 1 (proportion of the full sequence length used)
HORIZON	1, 3, 6, 12, 24
BATCH_SIZE	8 (for both linear and LSTM)
LEARNING_RATE	0.1 (with scheduler)
EPOCHS	20 (LSTM), 50 (Linear)
OPTIMISER	torch.optim.Adam
CRITERION	torch.nn.MSELoss()
ERROR_STD	list(range(2, 10.01, step=0.1))

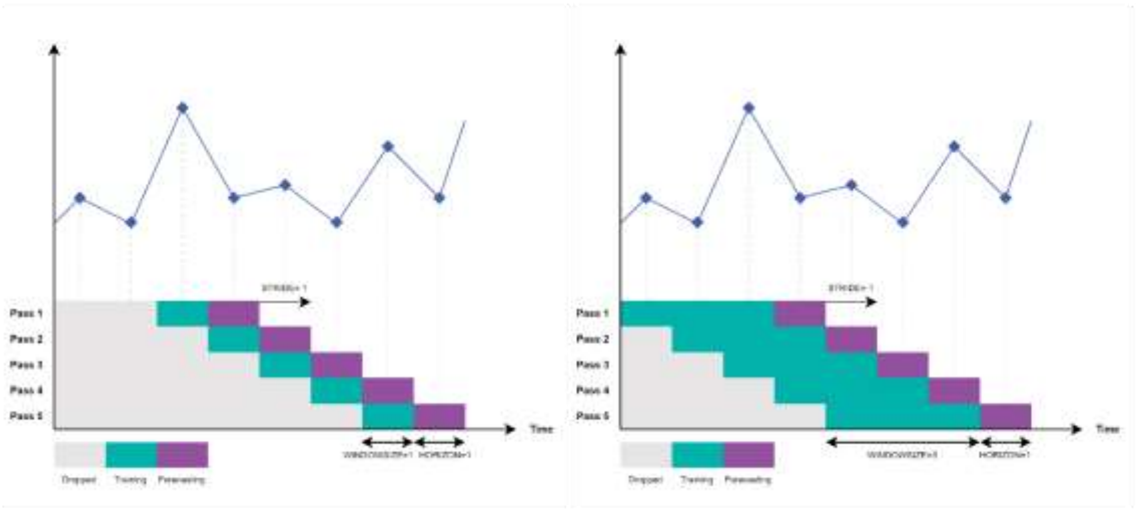
3.5.3 Model Description

Two types of models are used in the analysis. These are single-step univariate models, and single-step, multivariate models. Both model types are being tested in order to see what benefits feature engineering bring to the prediction capability. The step-size is dependent on the value of the of HORIZON which is an integer value denoting the number of 15-minute periods into the future that the model will predict. For univariate models the value of INPUT_INDICES and is equal to 0 which refers to the value field (pedestrian flow). For the multivariate models the value of INPUT_INDICES is list(range(0,11)) which translates to the value field and all of the engineered features described in 3.4.5 - these can also be seen in Figure 19. The TARGET_INDEX for both models is 0.

The data for both of the LSTM's use a train-test-validation split of 0.7, 0.2 and 0.1 respectively. The data is further split into windows which produce the train inputs, train targets, test inputs, test targets, validation inputs and validation target where the targets contain one value and the inputs contain a number of values equal to WINDOW_SIZE (Figure 14). The windows for each are then stacked into mini batches (equal to BATCH_SIZE) for training and testing. Figure 13 shows the effect of the windowing parameters on the structure of the data (WINDOW_SIZE, HORIZON, STRIDE).

3.5.4 Single-Step Models

The linear model uses a train-test split of 0.7 and 0.3 respectively. For this model the testing set is used validate the model as there is no testing step during the training of the model so the test set can be considered independent. As the model is linear, the weights for how it uses each of the engineered features can be accessed which gives us an indication of the utility of the features. Figure 14, Figure 19 and Figure 20 show how strongly each of the variables is weighted in a test run. When the model is trained on the target variable it dominates the weighting compared to when the target variable is excluded.



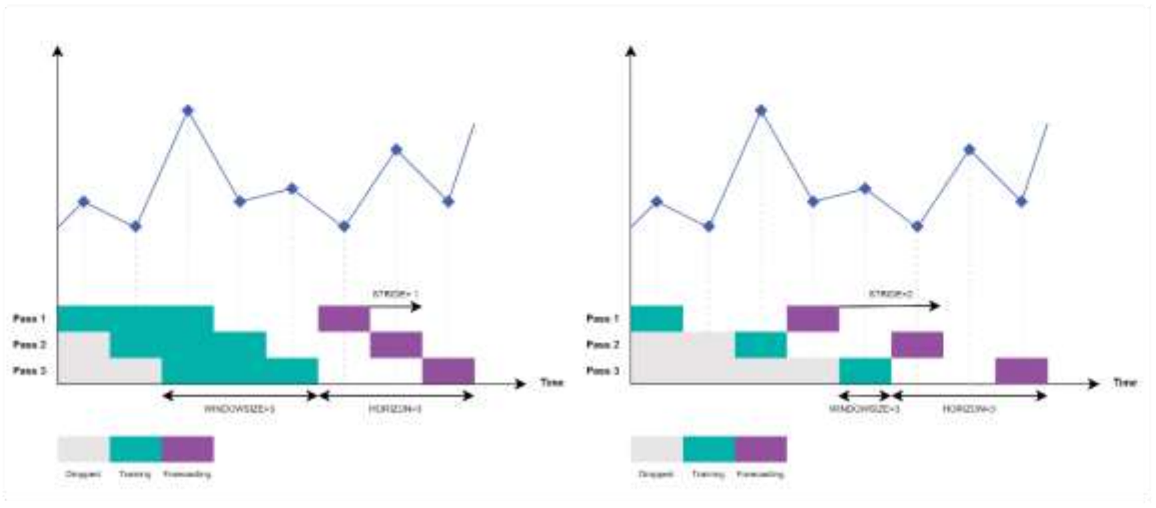


Figure 13 Sliding windows with different parameters

3.5.5 Model Training and Evaluation

Figure 14 shows the full workflow used to meet the objectives discussed in 3.1. The training workflow has been constructed to be versatile and adaptable, so different types of tests can be run with minimal overhead. This workflow describes the implementation of the `main()` function in the `model_training_loop.py` file found here. Figure 15 describes the evaluation workflow for which different parts of the workflow are implemented in the following Jupyter notebooks:

- Multivariate performance
- Multivariate anomaly detection
- Univariate performance
- Univariate anomaly detection

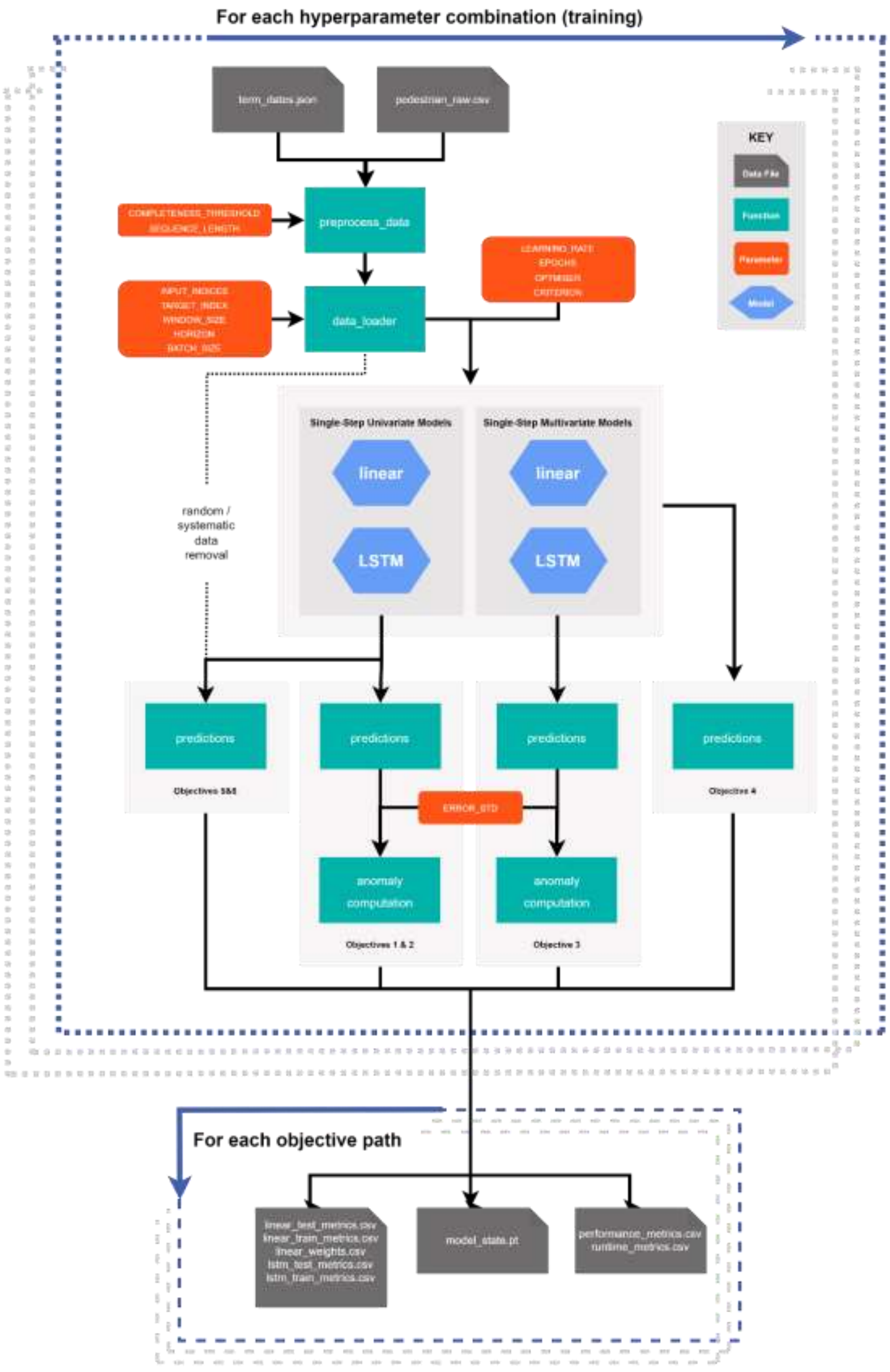


Figure 14 Overview of modelling workflow

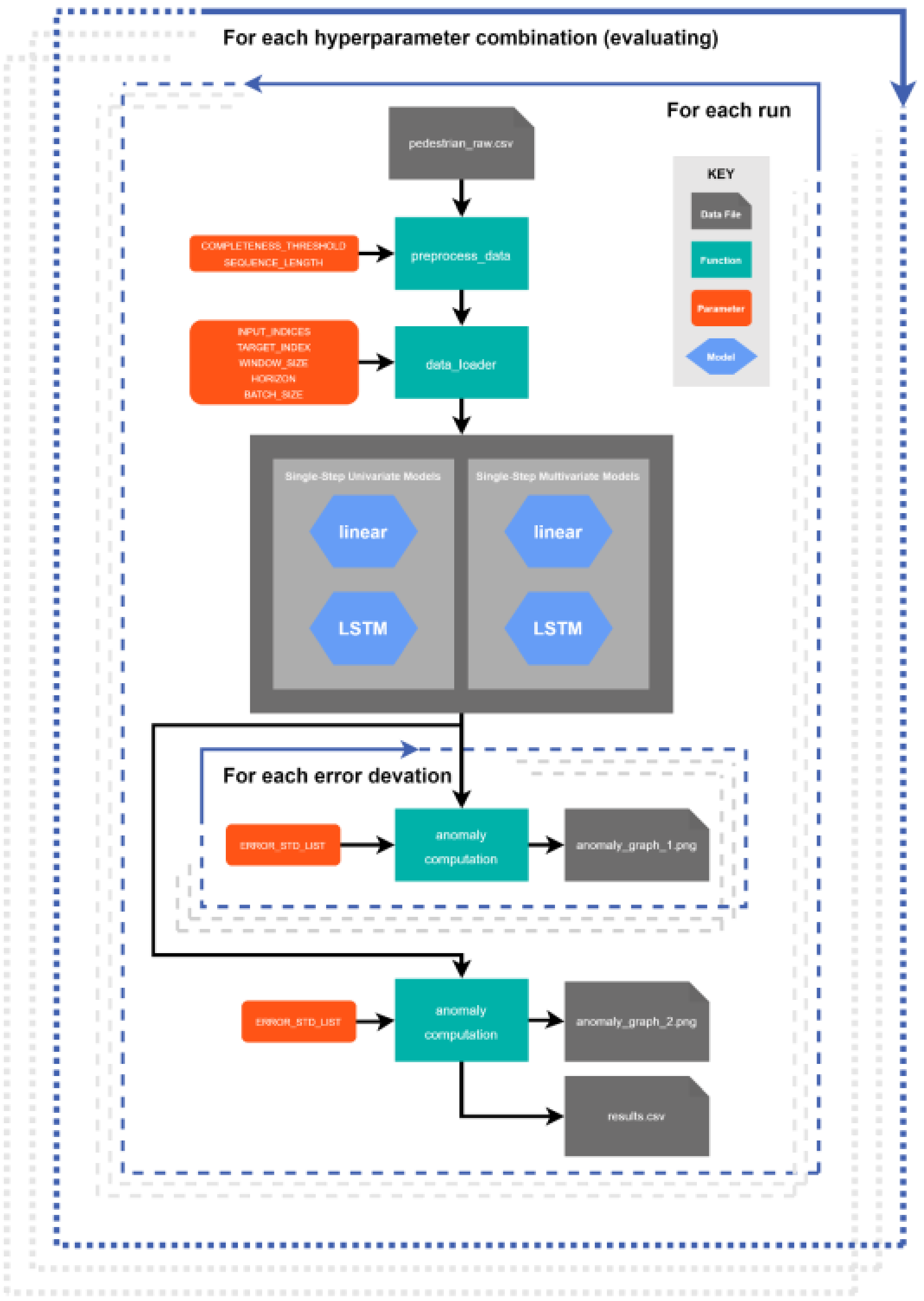


Figure 15 Overview of evaluation workflow

3.5.6 Anomaly Detection

The first three objectives involve anomaly detection which necessitate a separate workflow. The first task in anomaly detection is defining what is meant by an anomaly in this context. In time-series analysis they can be broadly categorises into two types - the first is outlier, which are values that are located outside the normal class. The second is is an anomalous behavior, which is a periodic collapsing phenomenon in time series. For this project there are two problems to investigate. The first is detecting data gaps that arise from incomplete data. When the COMPLETENESS_THRESHOLD parameter is set to 1 the model should not detect any anomalies of this type. But when gaps are introduced by reducing the COMPLETENESS_THRESHOLD, it is likely that the model will struggle make a prediction across the gap as this is equivalent to setting the HORIZON parameter to the length of the gap. Gaps of this type would be classed as a periodic collapsing phenomenon as the the sequence might jump from values expected at midday to values expected at midnight in the sequence.

Table 8 shows two consecutive days in December. The Monday sequence satisfies COMPLETENESS_THRESHOLD=1. The Tuesday sequence does not satisfy the threshold. On the Monday it is evident that the value for pedestrian count changes gradually at each subsequent 15-minute time period. On the Tuesday the changes are less gradual. It is not clear if the values we see in the second half of the Tuesday sequence are the sum of the recorded pedestrians since the previous recording or if they are coincidentally very high around the time that the sensor stopped recording properly. In the case that there is a genuine value of 2927 in the 15 minutes period before 16:00 then this value can be considered an outlier, otherwise it is an example of collapsing phenomena.

Table 8 Outliers

Monday 2021-12-27	Tuesday 2021-12-28

hour-minutes	value	hour-minutes	value
12:30	707	12:45	445
12:45	827	13:00	613
13:00	892	13:15	544
13:15	1050	14:00	2201
13:30	988	16:00	2927
13:45	1167	17:00	1867
14:00	926	19:00	685
14:15	948	22:00	132
14:30	988	23:00	52

Table two shows more clearly an example of periodic collapsing phenomena from two Saturdays in October 2022.

Table 9 Periodic collapsing phenomena

Saturday 2022-10-01		Saturday 2022-10-22	
hour-minutes	value	hour-minutes	value
10:00	217	10:00	183
10:15	248	10:15	160
10:30	273	10:30	1
10:45	358	10:45	3
11:00	286	11:00	1
11:15	393	11:15	2
11:30	305	11:30	1
11:45	295	11:45	1
12:00	256	12:30	4
12:15	217	13:15	428

In the case of Table 8 it is relatively easy identify genuine outliers like this using percentiles. However, identifying the patterns shown in Table 9 is more of a challenge. The reasons for can be seen in Figure 16. It would be conceivable to consider any value above 1000 people per 15-minutes an outlier. but as the flow values follow a Poisson distribution with lambda ~ 1, it is not possible to use the bottom percentiles. Instead, a method is proposed using the predictions from the trained models to calculate the errors for each data point and defining a threshold of standard deviation from the mean error:

$$\overrightarrow{errors} = \overrightarrow{prediction} - \overrightarrow{targets}$$

$$threshold = \overline{error} + \sigma_{error} * E_{\sigma}$$

$$\overrightarrow{anomalies} = \overrightarrow{errors} > threshold$$

Where E_{σ} is the number of standard deviations from the error chosen for the threshold. This value will have to be emprically derived.

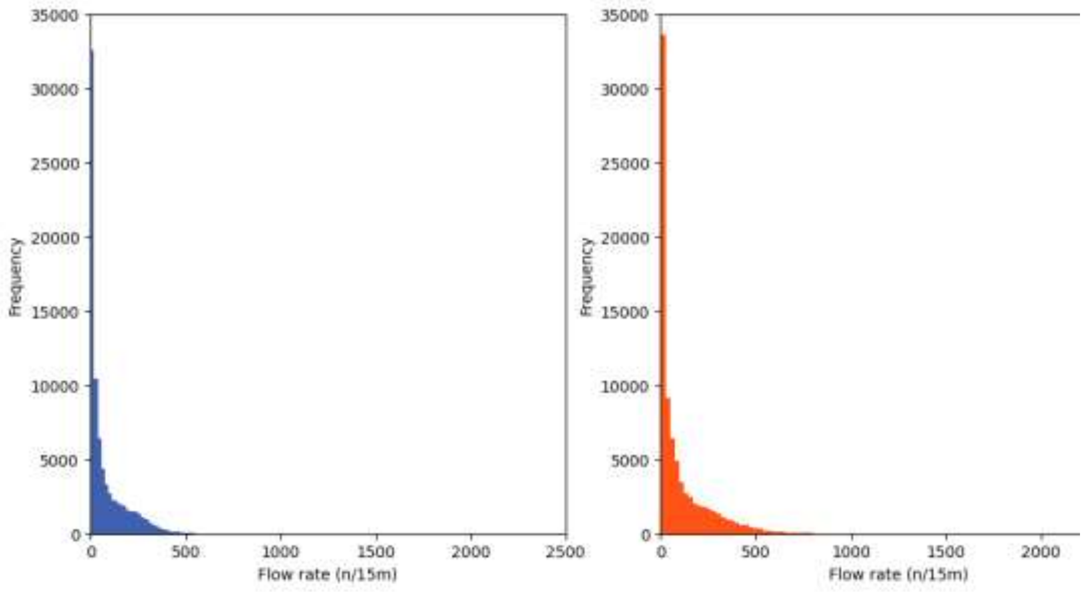


Figure 16 Histogram of flow rate for East and West sensors

3.5.7 Horizon Evaluation

For the evaluation on the effect of multivariate vs univariate modelling on the prediction accuracy of the horizon a variety of evaluation metrics will be compared. These can be seen in section 3.5.10.

3.5.8 Data robustness test

The data robustness test will involve comparing MAE of the data at different completeness thresholds.

3.5.9 Cross-validation

To ensure stability in the model a train-test-validation split of 70-20-10 has been used. Whilst it is not possible to randomly shuffle these for true cross-validation due to the nature of time series data, there are data sequences from different parts of the year that the model has not been trained, tested, or validated on. In some of the experiments we expose the model to these unseen sequences offering a cross-validation of sorts.

3.5.10 Model Evaluation

A variety of evaluation metrics have been calculated to assess the performance of the models.

Mean absolute error (MAE): this is the average absolute difference between the prediction and target values.

Mean squared error (MSE): similar to MAE but it squares the difference between prediction and target values before they are averaged, which mean larger errors are more noticeable than smaller errors.

Root mean squared error (RMSE): this is the square root of MSE, so the effect of each error on the average result is proportional to the size of the squared error yielding the same sensitivity to larger errors as MSE but can be more easily compared with MAE.

Coefficient of determination (R^2): measure the proportion of the variance for the prediction values that are explained by the input values, i.e., it provides a measure of how well future samples are likely to be predicted by the model.

4 RESULTS

The results section will cover the results derived from the methods for each of the objectives that are shown in 3.1. The results section is split into four subsections. The first subsection shows an overview of the univariate and multivariate approaches (for a more in-depth view see 3.5). The second subsection shows the results of the tests involving anomaly prediction (objectives 1, 2, & 3). The third subsection looks at the effect of prediction horizon on prediction accuracy (objective 4), and the fourth subsection covers tests concerning the robustness of the raw data (objective 5).

4.1 Workflow Overview

In the methodology, two models (LSTM and linear) were introduced, with each following a univariate and multivariate workflow.

4.1.1 Overview – Univariate Models

Figure 17 shows three example windows that the model is trained on. As this is a univariate model, the input is the pedestrian flow variable, and the target is the data-point a horizon length of time-steps further on from the end of the input sequence (in the case of Figure 17 six time-steps futher). This means that the model is trying to predict an hour and a half into the future, using the forty-eight previous observations.

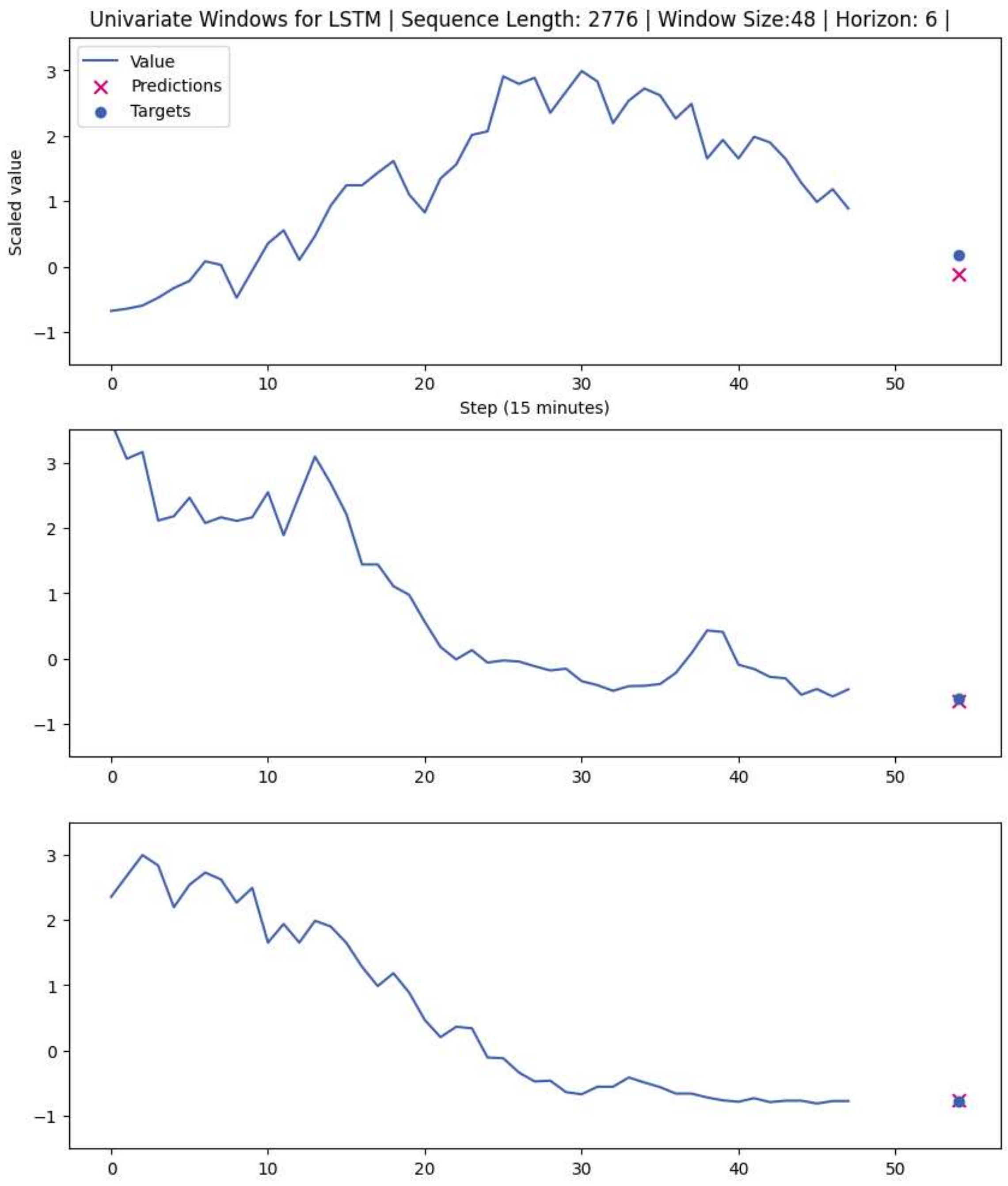


Figure 17 Univariate Input Window, Target, and Prediction

4.1.2 Overview – Multivariate Models

Here the input variables for the multivariate model can be seen for three randomly sampled windows (Figure 18). It is worth noting the sine functions calculate the values for each hour (4-time steps). In the future, although slightly more computationally expensive, it would be worth smoothing these to get a one to one relationship between sine value and timestep.

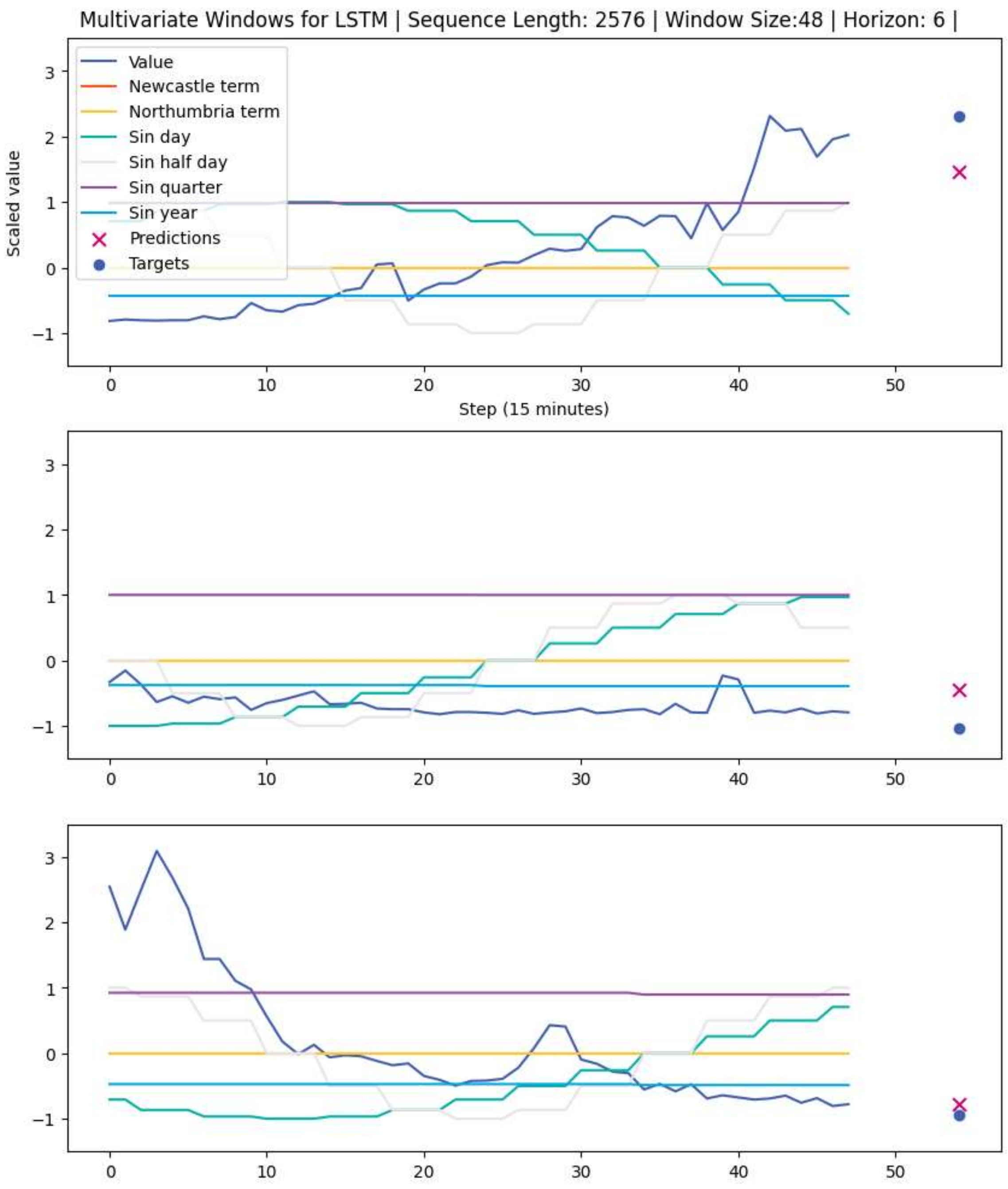


Figure 18 Multivariate Input Window, Target and Predictions

4.1.3 Linear Model Training

The models are trained for a number of epochs. Figure 19 and Figure 20 show the variable weightings of the linear model if it is trained for 1000 epochs. Figure 19 shows the weightings if all 11 inputs variables are used for predicting the target variable (pedestrian count) and it shows a strong weighting on the target variable Figure 20. If the target variable is excluded from the input variables, it shows the relative weightings of added variables. The weightings for university term times appear to be unstable however, and oscillate significantly between epochs. The weight values for the periodicity features seem stable through multiple epochs. It is interesting to note the amount of weight placed on the feature with a 1/quarter frequency, as this was not a powerful signal in the frequency analysis (see 3.4.4).

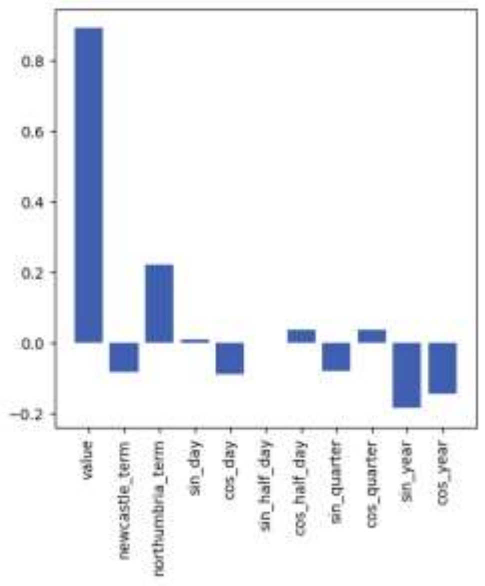


Figure 19 Linear model weights (target variable included)

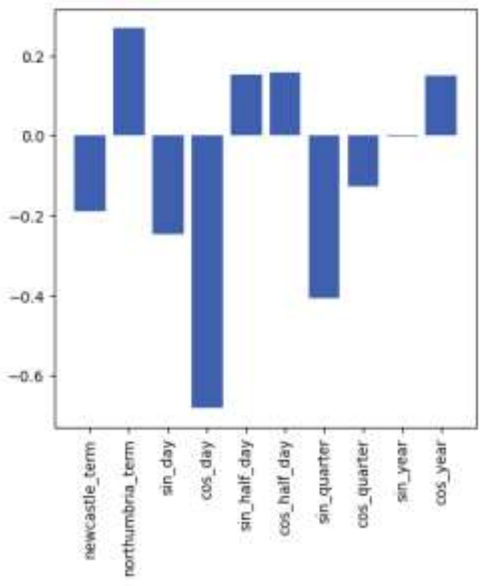


Figure 20 Linear model weights (target variable excluded)

4.2 Anomaly prediction test

The anomaly prediction results come from the evaluation workflow (shown in 3.5.5). There are three objectives for this test. The first objective is to identify anomalies using single-step univariate prediction. This objective is important to measure because the feature engineering required for multivariate models requires time and resources. The second objective follows the same process as the first but with the multivariate model (input features are increased from 1 to 11).

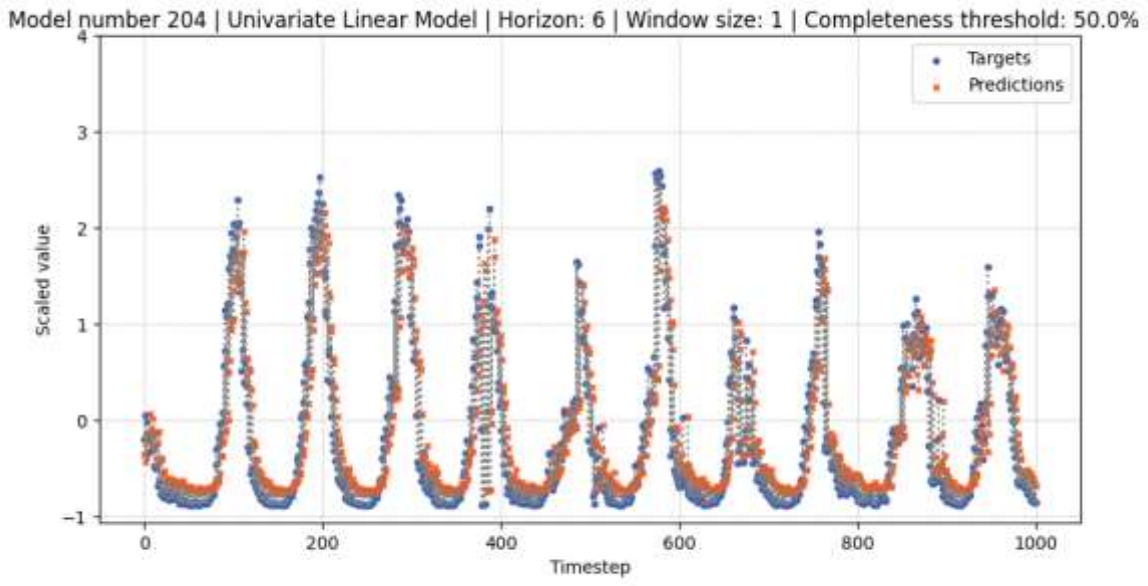
When testing the model on an unseen data an interesting pattern emerges. To demonstrate, six has been chosen as the horizon as this equates to an hour and a half, which for practical applications, like crowd control, seems like a reasonable compromise between horizon time needed to coordinate a response and prediction accuracy (which decreases as horizon time increases).

The models presented here, are validated on a data set that has been loaded with a completeness threshold of 50% (much lower than the threshold for training). For both objectives in this section, the window size for the LSTM models has been kept at 48 for consistency.

4.2.1 Test Objective 1

Objective 1: Create a modelling workflow that identifies anomalies in the data using single-step univariate prediction validating on unseen data.

Figure 21 shows the error between each prediction and target for a particular model and for a portion of the validation data. For the univariate models it seems that both the LSTM and linear models perform similarly.



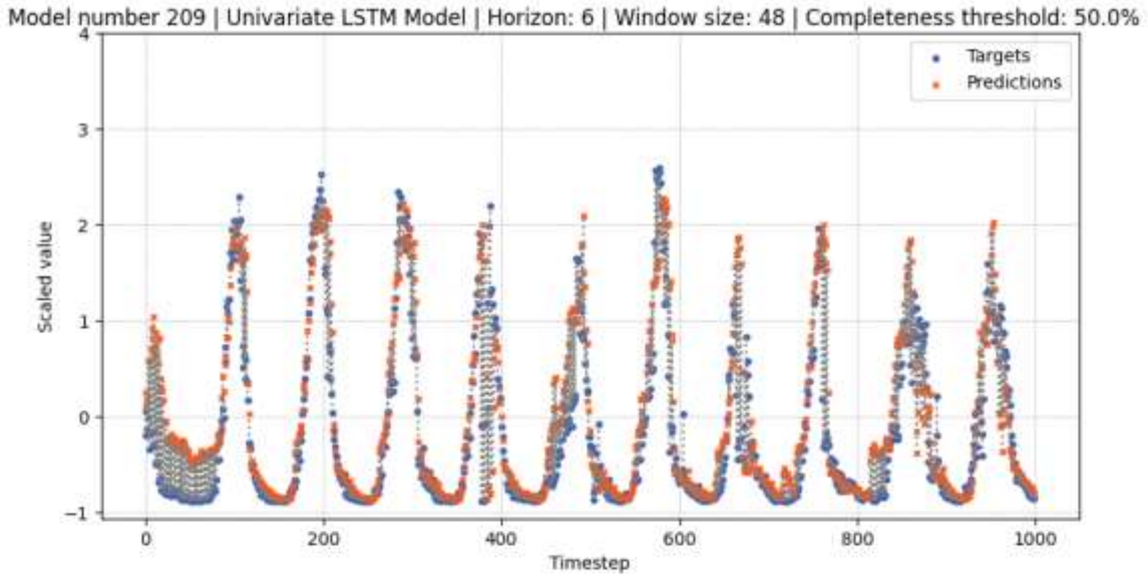


Figure 21 Targets v Predictions for a Segment of the Validation Data – Univariate

Figure 22 shows the target variables that have been identified as anomalies. It is evident that the linear model have not identified the anomalies correctly. It appears that the model once fed the first window sequence containing an anomalous data point, significantly reduces its subsequent prediction. This means that when when the real data returns to its expected behaviour, that there is lag equal to the horizon length of the model before it will stop predicting anomalies. It can be seen in Figure 22 that the three last anomalies are probably not actually anomalies, but because the model updated its prediction based on the most recent input window, all the data points are beyond the anomaly threshold of standard deviations from the mean error.

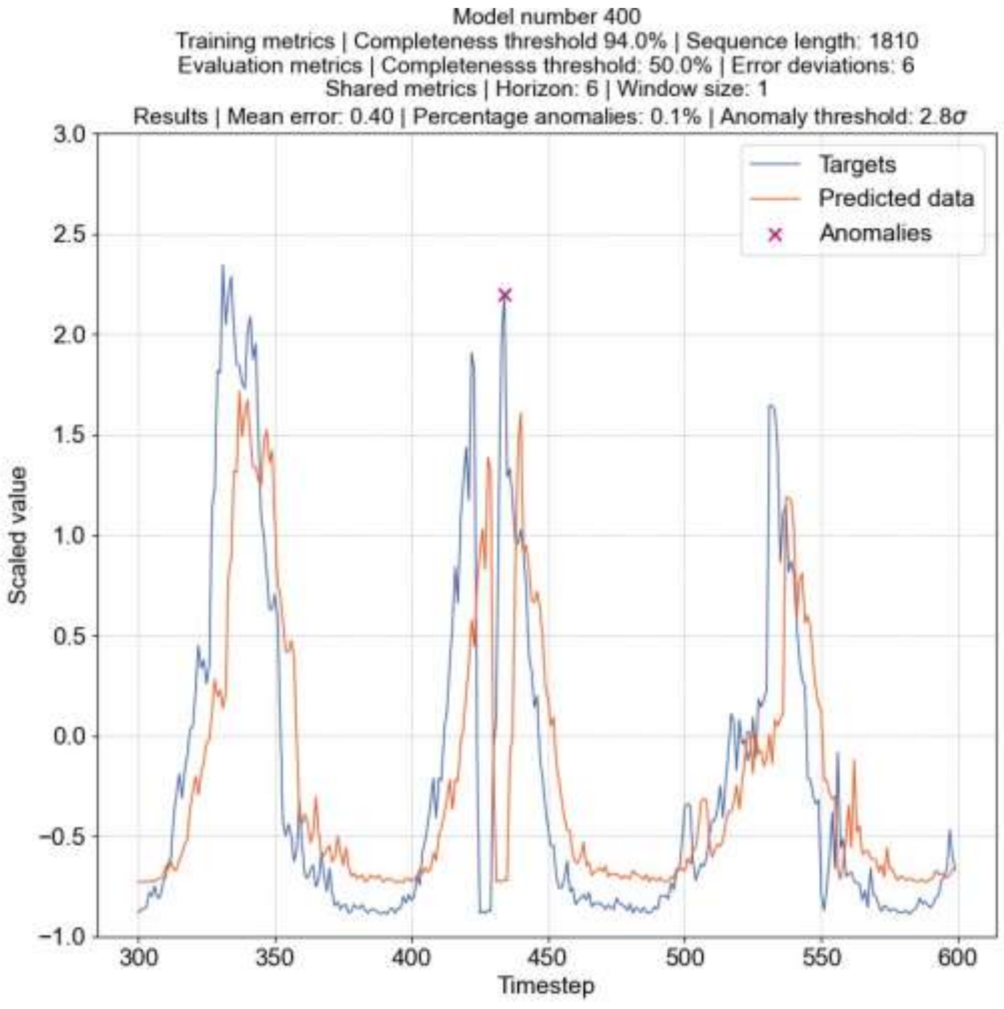


Figure 22 Sample Anomaly Predictions – Univariate Linear

Figure 23 shows the results of model 209 which used a completeness threshold of 98% during training. This model has correctly identified all the anomalies, but the model still experiences the same problem as above, when the values return to normal behaviour they are still being labelled as anomalies for a couple of time-steps. This suggests that the model is not generalising enough and instead appearing to be very reliant on the values in the previous input window.

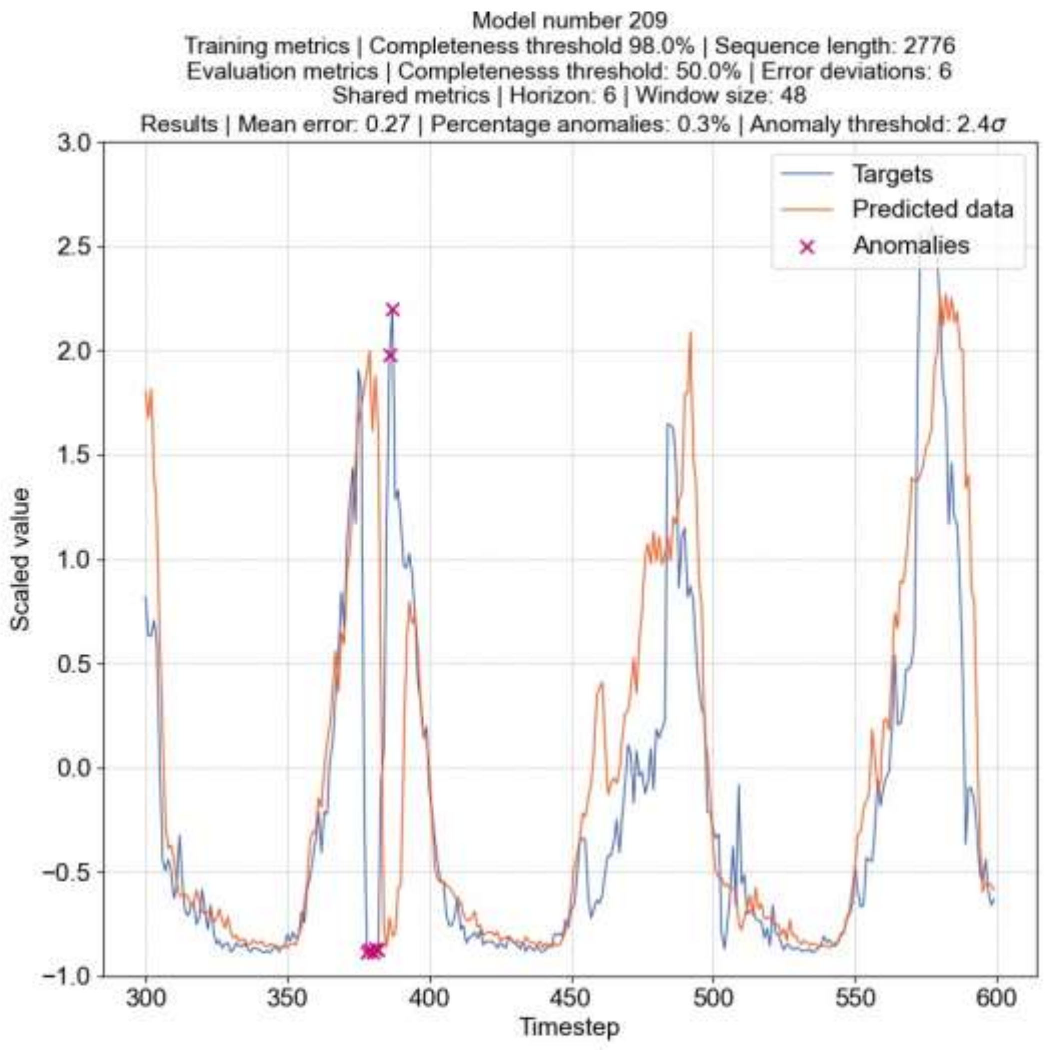


Figure 23 Sample Anomaly Predictions - Univariate LSTM

Figure 24 shows how the percentage of the targets identified as anomalies changes as the ERROR_DEV threshold changes. Both models follow a similar pattern.

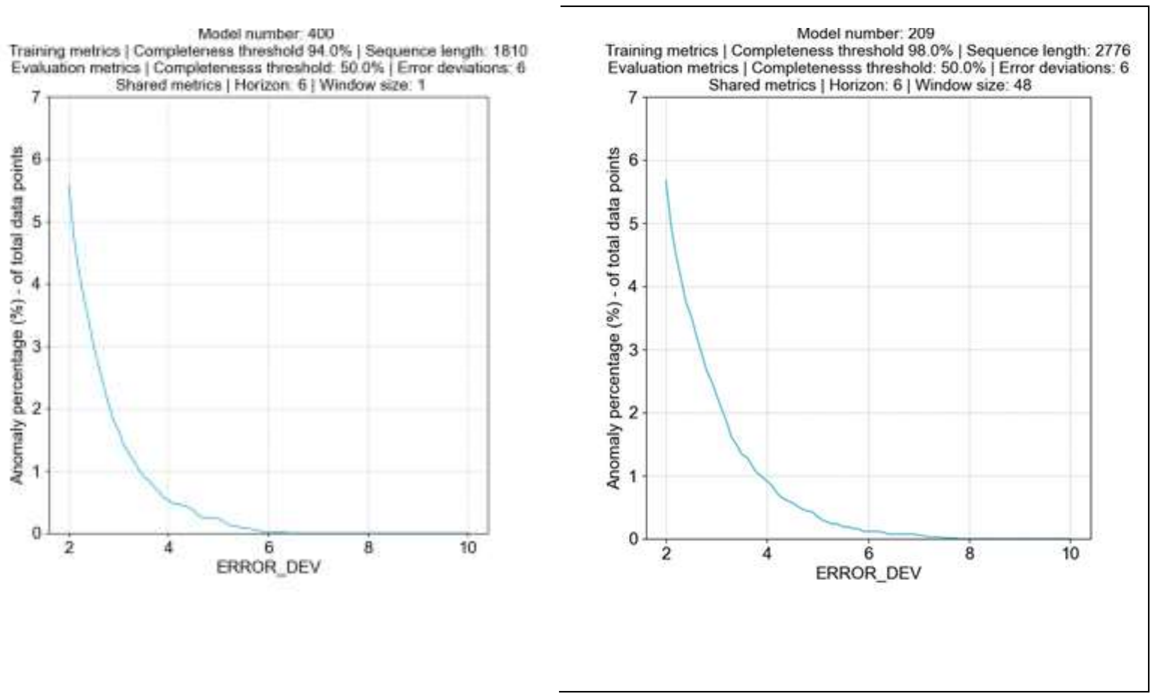
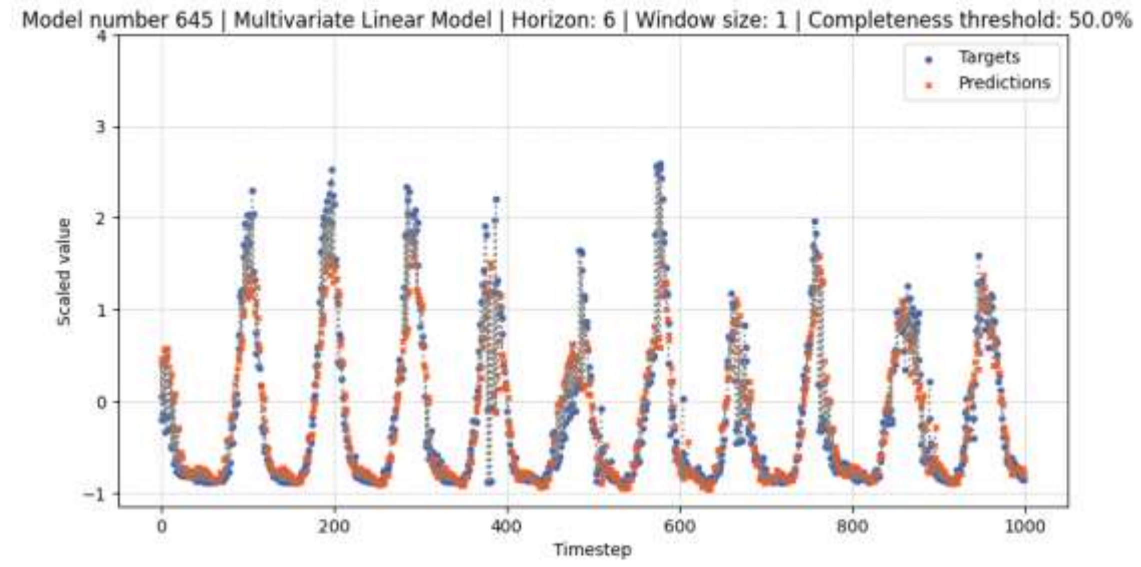


Figure 24 Percentage Anomalies Detected for Values of ERROR_DEV – Univariate

4.2.2 Test Objective 2

Objective 2: Create a modelling workflow that identifies anomalies in the data using single-step univariate prediction validating on unseen data.

Figure 25 shows the error between predictions and targets for both the multivariate LSTM and linear models for a section of the validation set. Both models perform much better than for the univariate workflow in term of MAE (see 9.2.1 and 9.2.2 of the appendix).



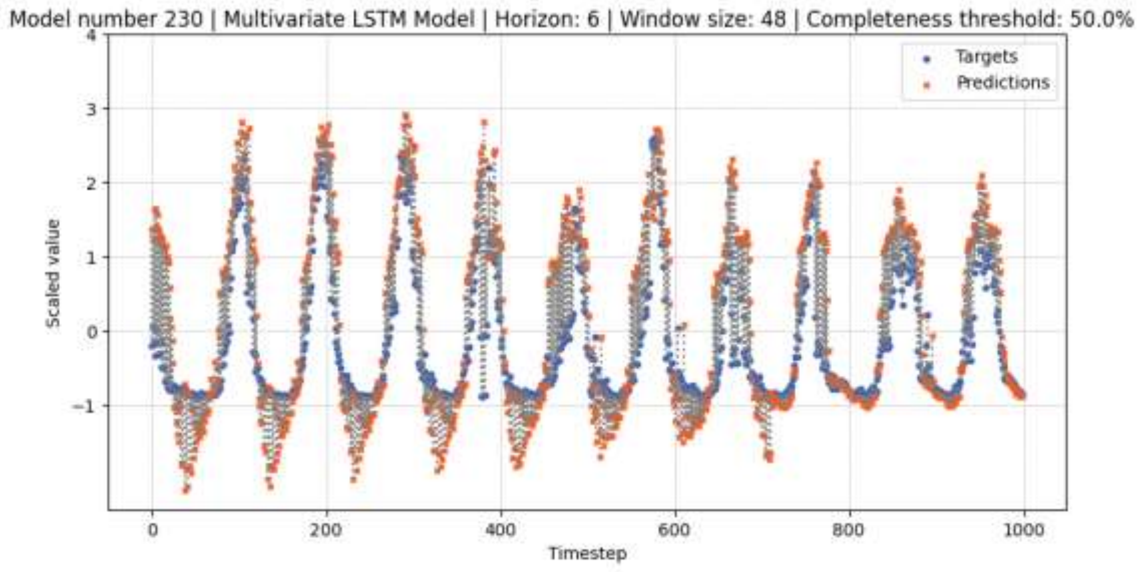


Figure 25 Targets v Predictions for a Segment of the Validation Data- Multivariate

Figure 26 and Figure 27 show the predicted anomalies for the multivariate linear and LSTM models. Whilst the anomalies are correctly predicted here, it appears the model is overtraining on the periodicity features. Because the periodicity features give a relatively accurate prediction, and the learning rate for models was set quite high at 0.1 (using step function to decrease it as it trains), it is possible that the learning rate does is not low enough to capture the nuances in the target data during training. Running the model for more epochs would help to understand this further. It is also possible that because the model still has the same number of neurons in the hidden layer as the univariate model, but has an order of magnitude more input features, that more parameters are needed to capture the more complex patterns.

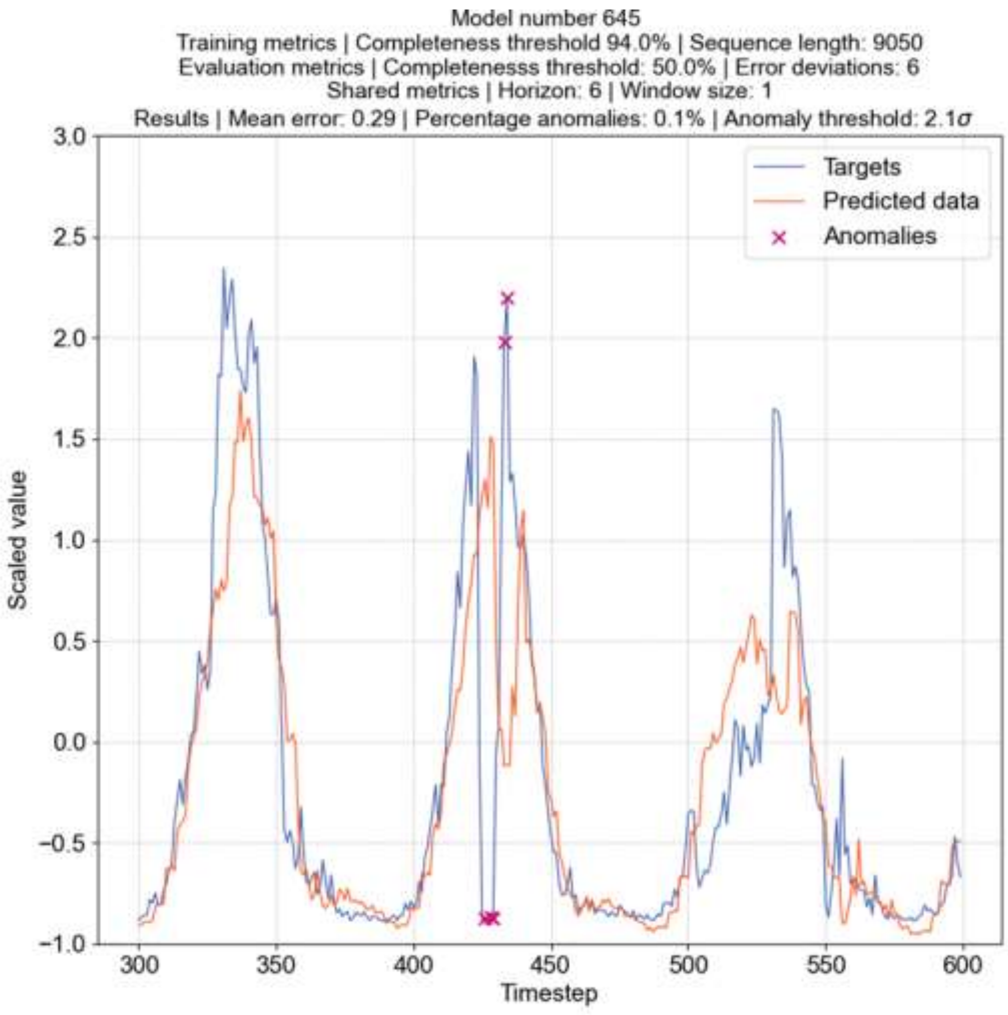


Figure 26 Sample Anomaly Predictions - Multivariate Linear

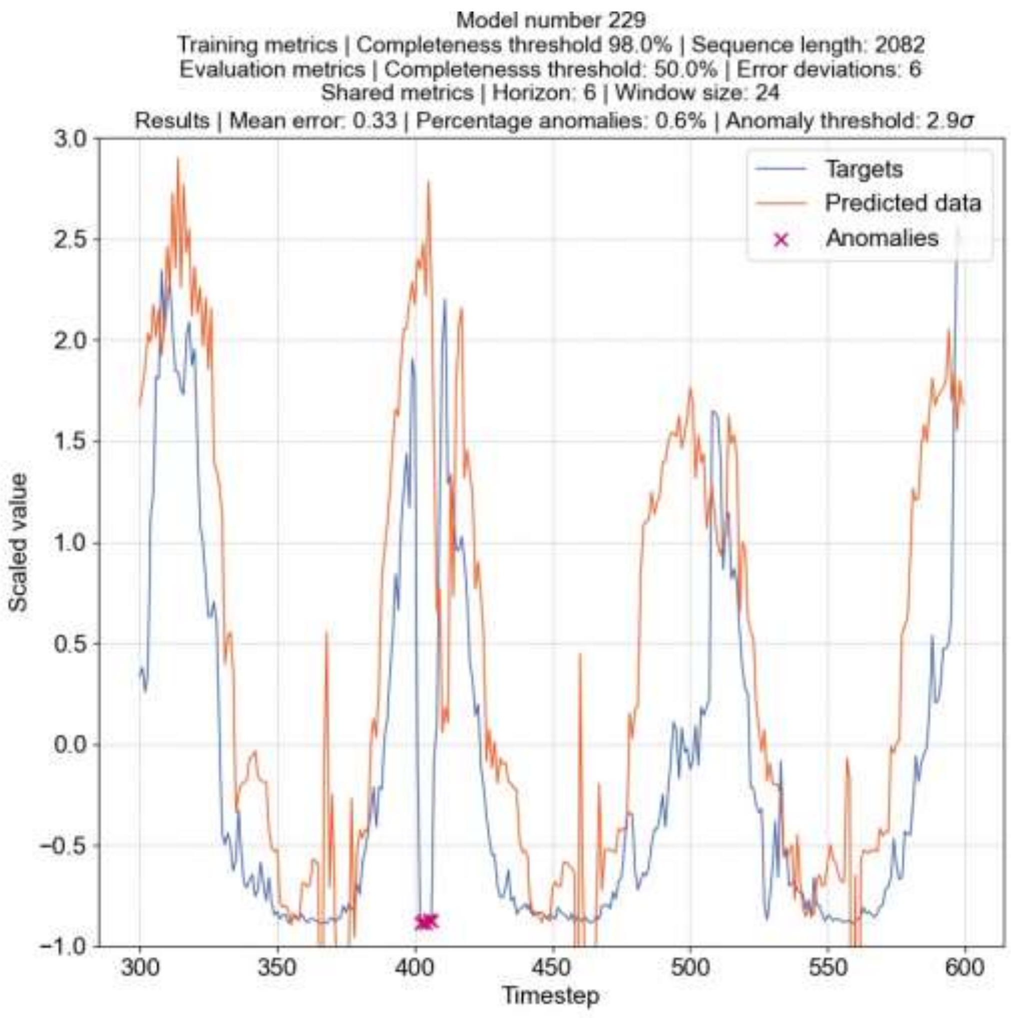


Figure 27 Sample Anomaly Predictions - Multivariate LSTM

Figure 28 show what percentage of the validation data are labelled as anomalies for each value of ERROR_DEV. The pattern seems to be very similar to the univariate results.

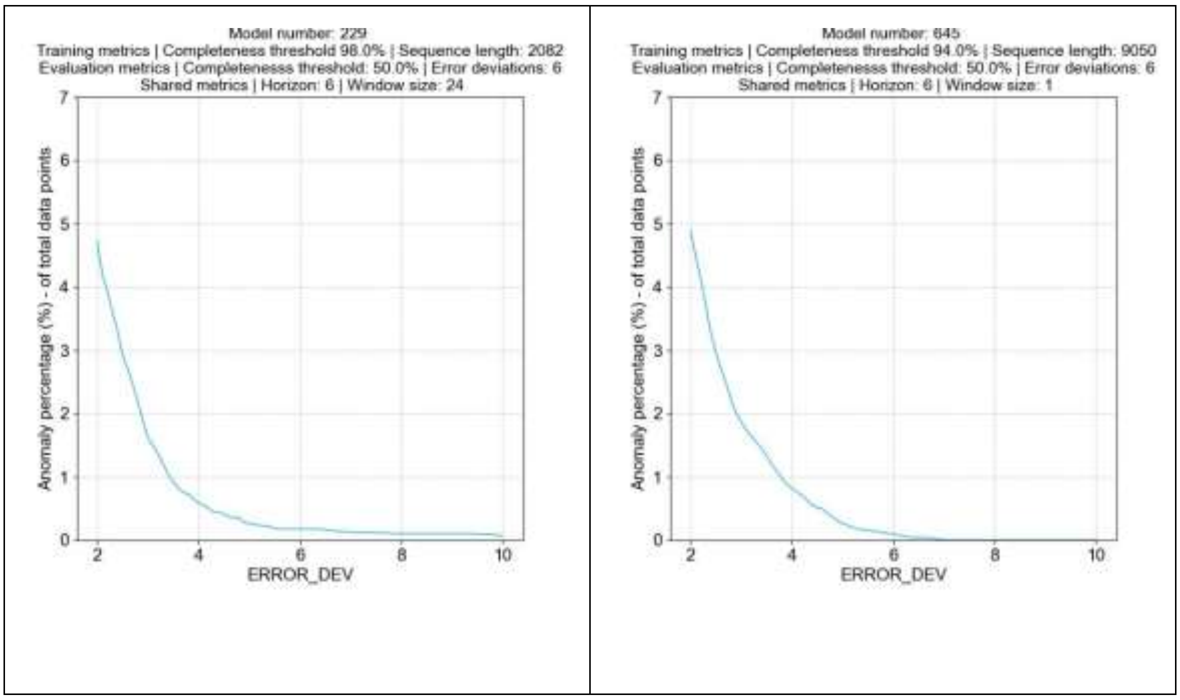


Figure 28 Percentage Anomalies Detected for Values of ERROR_DEV – Multivariate

Table 10 below shows the results for both univariate and multivariate LSTM models for window size 48 and horizon 6. Each have been labelled according to whether they correctly predicted the sequence of anomalies shown above.

What is interesting to note in Table 10 is that the correct anomaly locations in the sample period are only identified when the model is trained on 0.98 completeness. Why is this? There are two possible reasons. The first is that on the lower completeness thresholds the models require a lower error deviation in order to pick up the anomalies. This would likely arise from the models performing better on the longer training sequence lengths found in the lower completeness thresholds due to the increased amount of data. The second is that the data with lower completeness thresholds used to train the models, contain many anomalies that are artefacts of the data pre-processing. For example, the sudden jumps on the second and third cycles that might have arisen as a result of jumping two or more time periods in one timestep. To test this would require checking the timesteps against the expected time periods to highlight the data gaps. The model will likely generalise more on the data with longer sequence lengths, but because there are more anomalies the model's predictions are likely to be less accurate as they increase the uncertainty of the model. The question here is whether the model is overgeneralising as result of the presence of anomalies. There is a pattern in the number of standard deviations σ in the anomaly threshold is lower in the 98% completeness model numbers which is effectively the cut-off point for what is considered an anomaly from the MAE. Higher MAE and a higher anomaly threshold would almost certainly lead to poorer model performance. This means that the model is making more conservative predictions so the error is higher meaning the error deviations would likely cover a larger range relative to models with lower MAE. By running the evaluation again in with an error deviation of 5 as the cut-off for anomalies it might be possible to see if the models perform any better.

The sample identification column shows quite clearly the effect of data quality on the ability to predict anomalies. However, it is also evident that the models trained on the high completeness thresholds have the ability to recognise anomalies in the incomplete data. It is expected that as the completeness threshold goes down (remember the completeness threshold c refers to each day not to each sequence and the sequence of length n days each containing at least a factor of c of the expected data). Because of this the completeness of a sequence of length n can always be expected to be greater than or equal to the completeness threshold.

Table 10 Results from Anomaly Evaluation for LSTM

Model	Model Number	Completeness	Sequence Length	Horizon	Window Size	MAE	Percentage Anomalies (%)	Anomaly Threshold (σ)	Correctly Identified?
Univariate LSTM	**125**	**0.98**	**694**	**6**	**48**	**0.29**	**0.15**	**2.30**	**Y**
Univariate LSTM	**153**	**0.98**	**1388**	**6**	**48**	**0.28**	**0.10**	**2.40**	**Y**
Univariate LSTM	**181**	**0.98**	**2082**	**6**	**48**	**0.32**	**0.16**	**2.60**	**Y**
Univariate LSTM	**209**	**0.98**	**2776**	**6**	**48**	**0.27**	**0.12**	**2.40**	**Y**
Univariate LSTM	**237**	**0.98**	**3470**	**6**	**48**	**0.30**	**0.12**	**2.30**	**Y**
Univariate LSTM	265	0.96	1288	6	48	0.29	0.04	2.60	N
Univariate LSTM	293	0.96	2576	6	48	0.45	0.00	3.40	N
Univariate LSTM	321	0.96	3864	6	48	0.33	0.00	3.00	N
Univariate LSTM	349	0.96	5152	6	48	0.36	0.63	3.60	N
Univariate LSTM	377	0.96	6440	6	48	0.32	0.09	2.70	N
Univariate LSTM	405	0.94	1810	6	48	0.28	0.00	2.50	N
Univariate LSTM	433	0.94	3620	6	48	0.29	0.00	2.40	N
Univariate LSTM	461	0.94	5430	6	48	0.32	0.35	2.80	N
Univariate LSTM	489	0.94	7240	6	48	0.30	0.04	2.60	N
Univariate LSTM	517	0.94	9050	6	48	0.38	0.03	2.80	N
Multivariate LSTM	**160**	**0.98**	**694**	**6**	**48**	**0.31**	**0.147**	**2.5**	**Y**
Multivariate LSTM	**195**	**0.98**	**1388**	**6**	**48**	**0.35**	**0.147**	**2.6**	**Y**
Multivariate LSTM	**230**	**0.98**	**2082**	**6**	**48**	**0.35**	**0.074**	**2.8**	**Y**

Multivariate LSTM	265	0.98	2776	6	48	0.38	0.000	3.1	N
Multivariate LSTM	300	0.98	3470	6	48	0.39	0.000	3.1	N
Multivariate LSTM	335	0.96	1288	6	48	0.28	0.015	2.3	N
Multivariate LSTM	370	0.96	2576	6	48	0.28	0.088	2.5	N
Multivariate LSTM	405	0.96	3864	6	48	0.33	0.074	3.1	N
Multivariate LSTM	440	0.96	5152	6	48	0.26	0.088	2.2	N
Multivariate LSTM	475	0.96	6440	6	48	0.32	0.044	2.6	N
Multivariate LSTM	510	0.94	1810	6	48	0.39	0.088	3.4	N
Multivariate LSTM	545	0.94	3620	6	48	0.30	0.147	2.6	N
Multivariate LSTM	580	0.94	5430	6	48	0.28	0.147	2.3	N
Multivariate LSTM	615	0.94	7240	6	48	0.33	0.029	2.4	N
Multivariate LSTM	650	0.94	9050	6	48	0.38	0.279	3.8	N

4.3 Horizon Prediction Test

For this test a different set of completeness values is being used. This is so that it is possible to see the effect on less complete data than for the previous section.

4.3.1 Test Objective 3

Objective 3: Use both univariate and multivariate workflows to measure the decrease in prediction accuracy as the prediction horizon increases.

Figure 29 and Figure 30 show the variation in MAE for different horizon lengths. The linear model has very little interquartile range variance, which suggests what the model is learning does not change for the different horizon lengths. The weights for the different univariate linear models should give us some indication of what is going on.

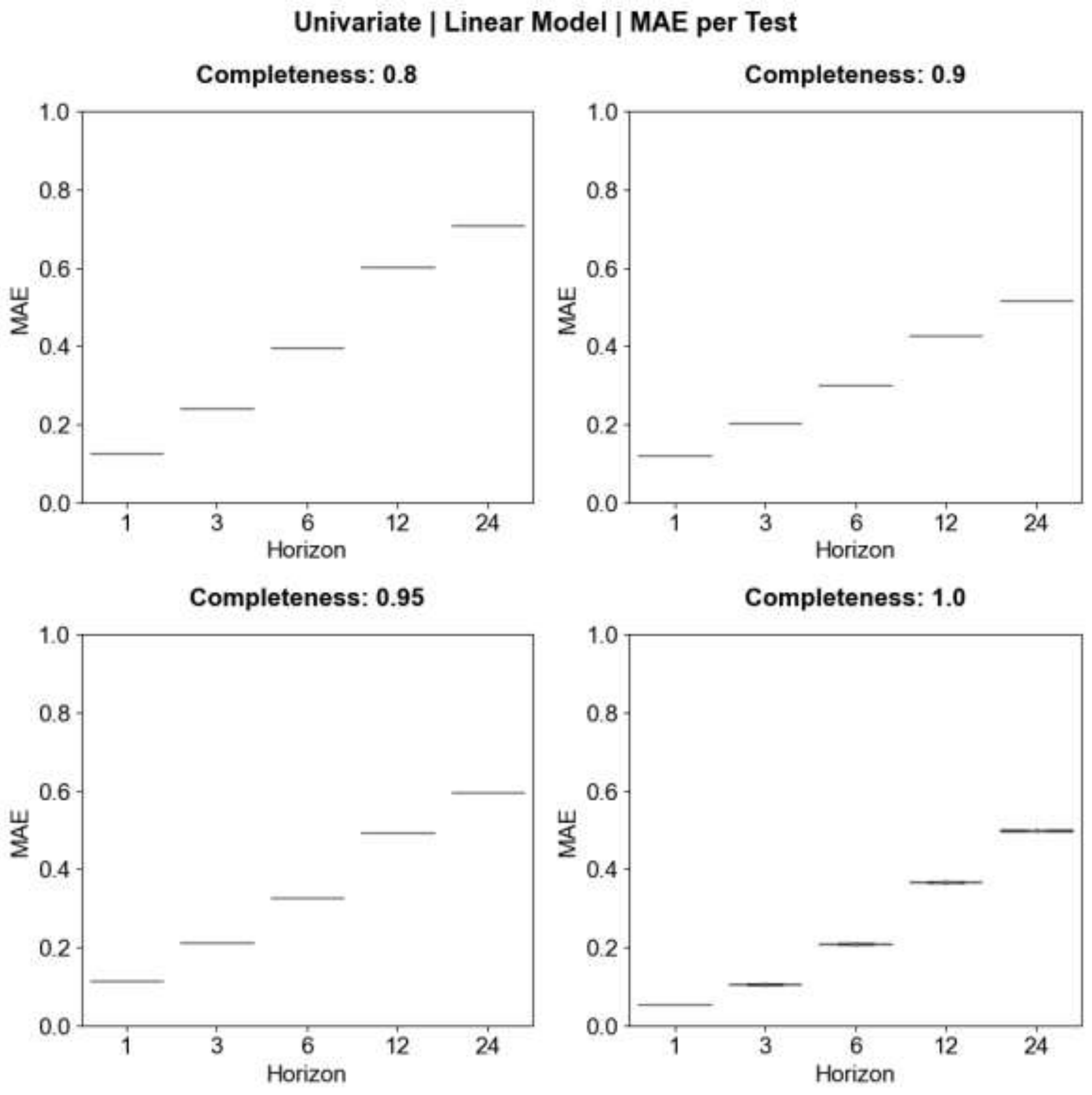


Figure 29 Horizon Length and MAE for Univariate Linear

The univariate LSTM performs slightly better than the linear model, but not significantly better. MAE becomes quite high as the completeness values are reduced.

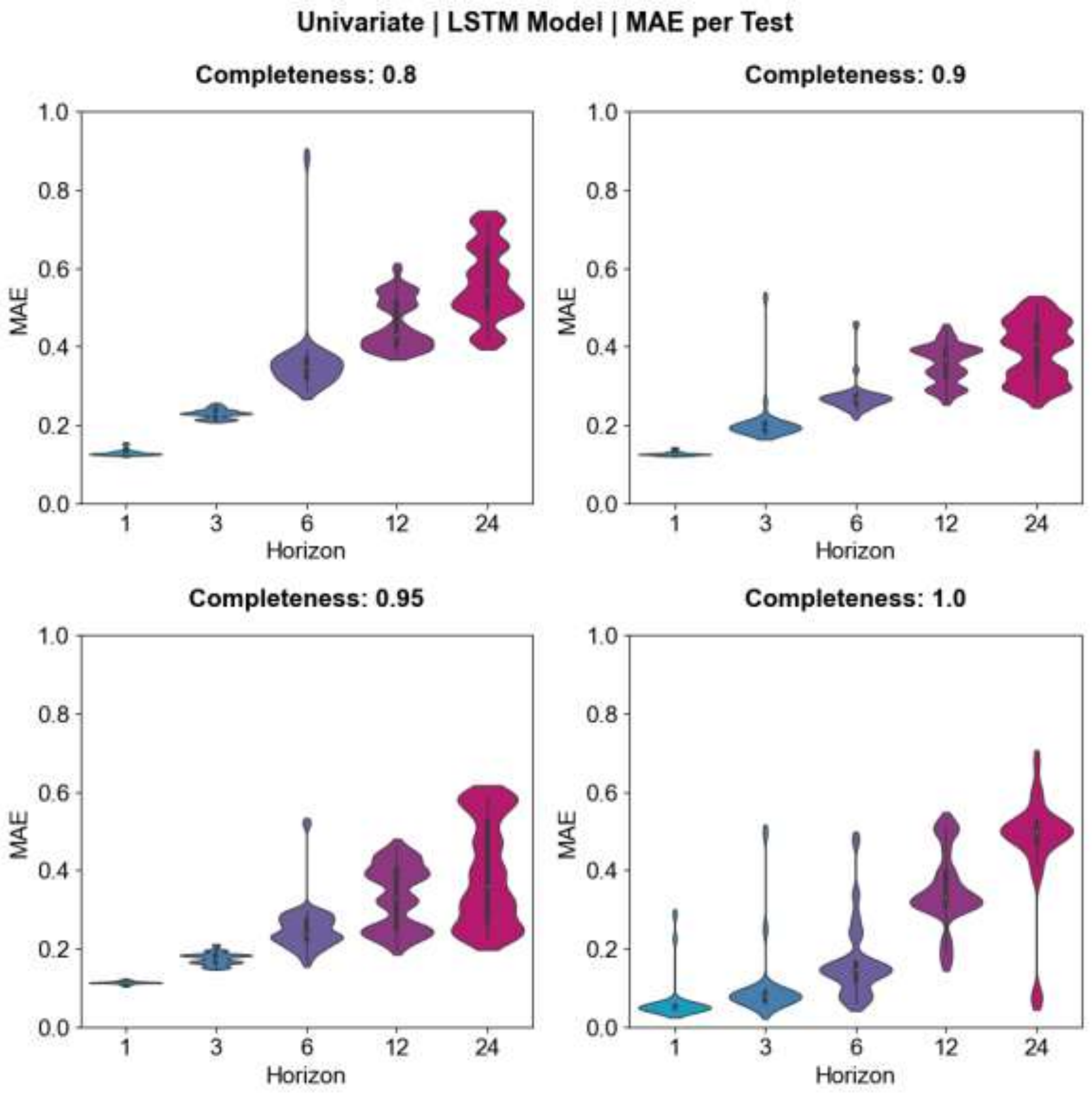


Figure 30 Horizon Length and MAE for Univariate LSTM

Figure 31 and Figure 32 show the effect of horizon length on the mean absolute error for the multivariate models. For the LSTM results, it can be seen that horizon appears to be highly correlated with MAE, as the interquartile range for each horizon value shown in the violin plot is low but varies significantly between horizon values. The same goes for completeness, as the interquartile are very different between each of the subplots. It is evident that the LSTM performs much better than the linear model.

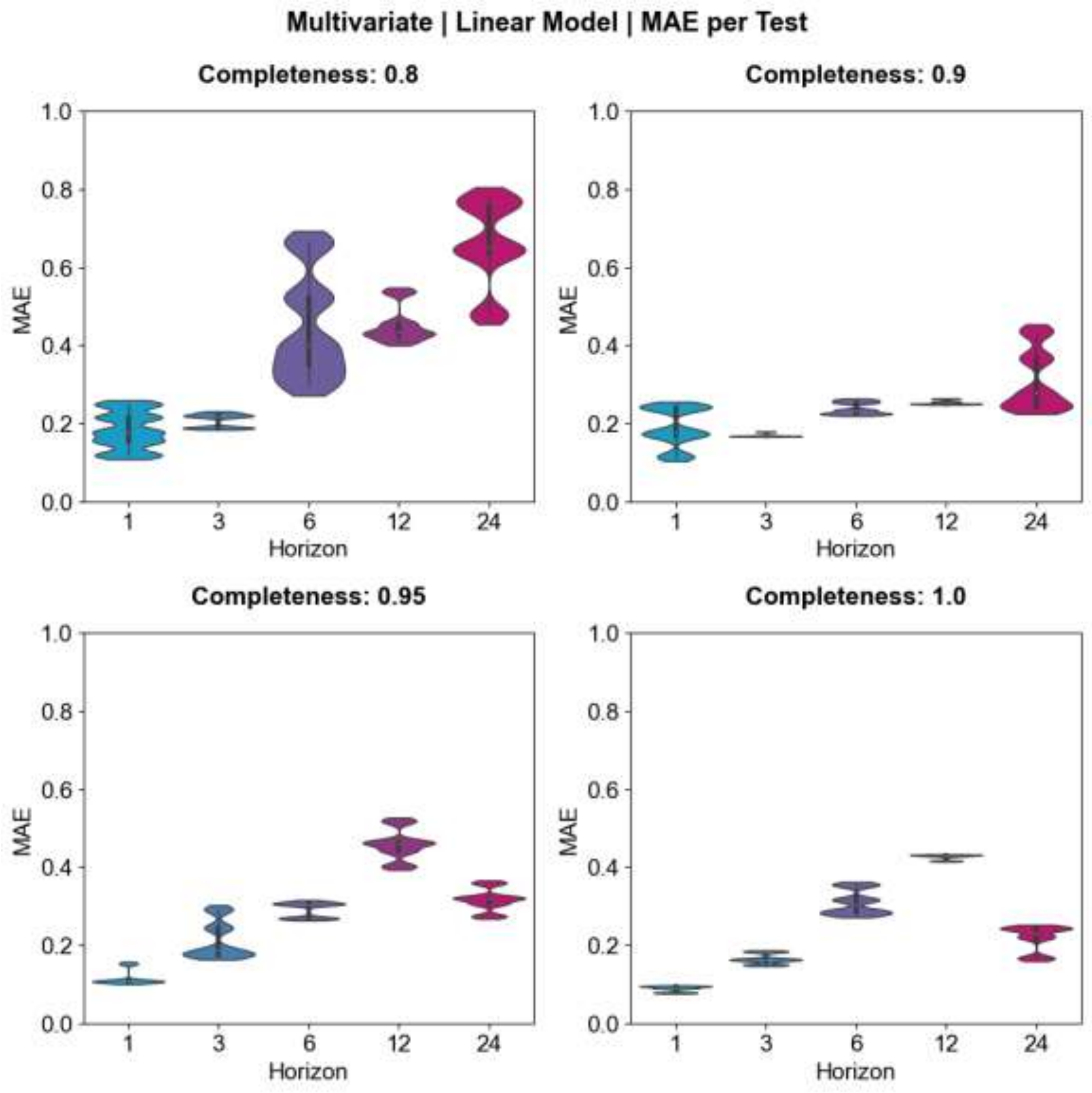


Figure 31 Horizon Length and MAE for Multivariate Linear

Figure 32 shows the multivariate LSTM. Compared to the univariate LSTM the MAE values are much lower and the interquartile range is much shorter.

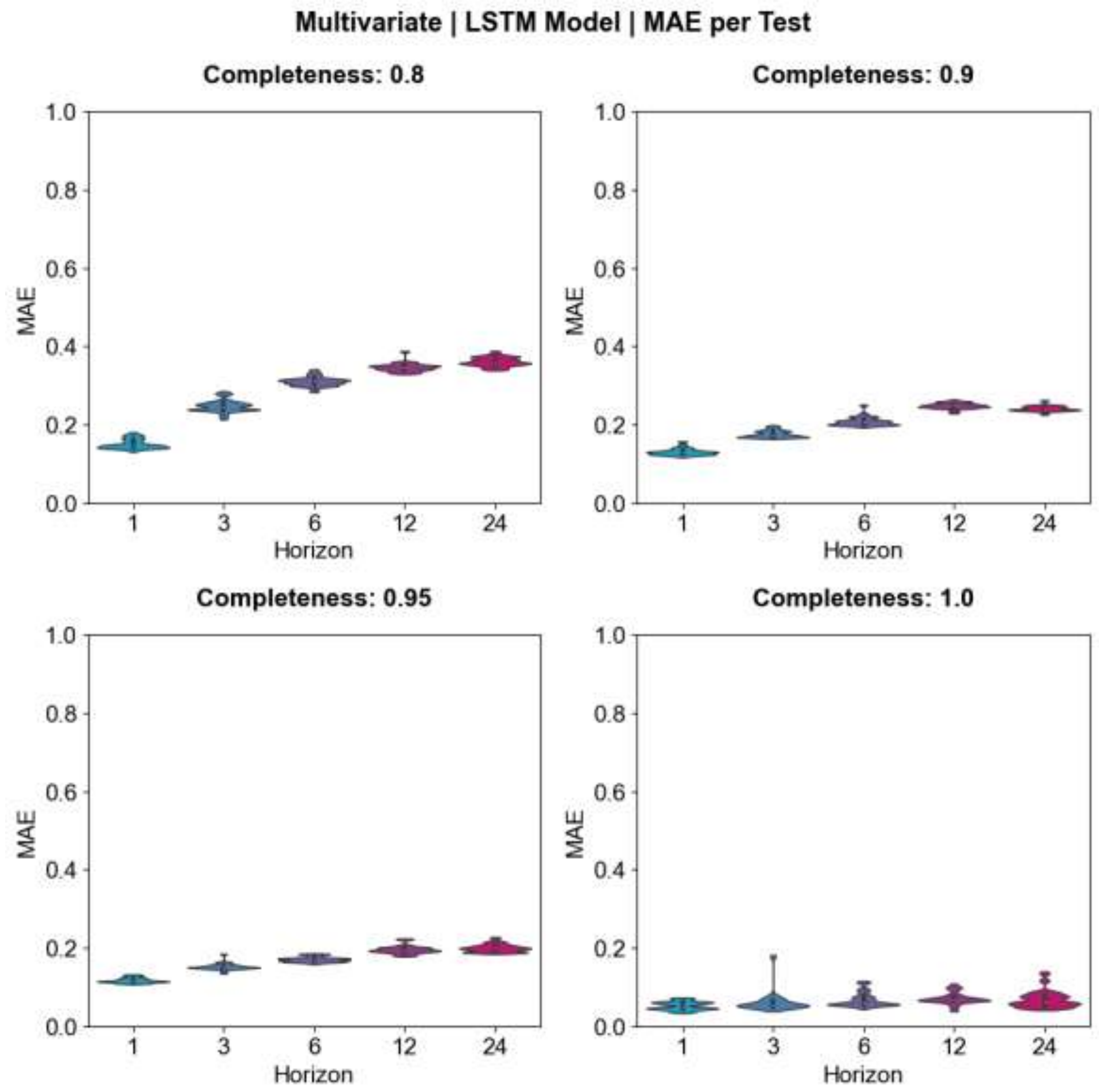


Figure 32 Horizon Length and MAE for Multivariate LSTM

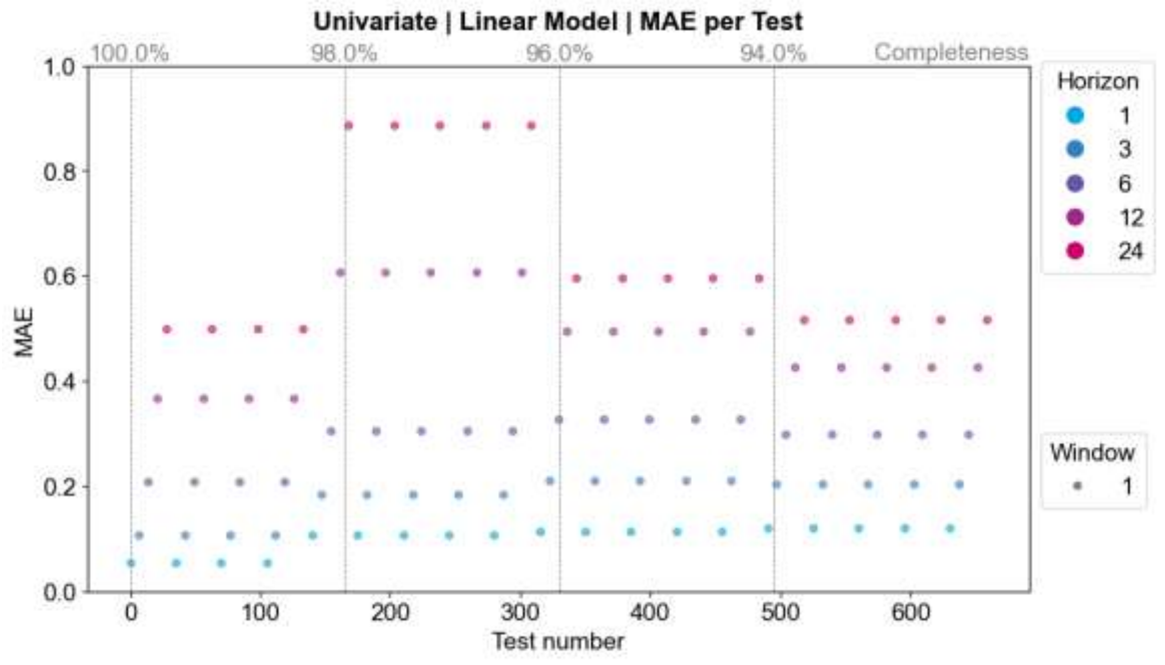
4.4 Data Robustness Test

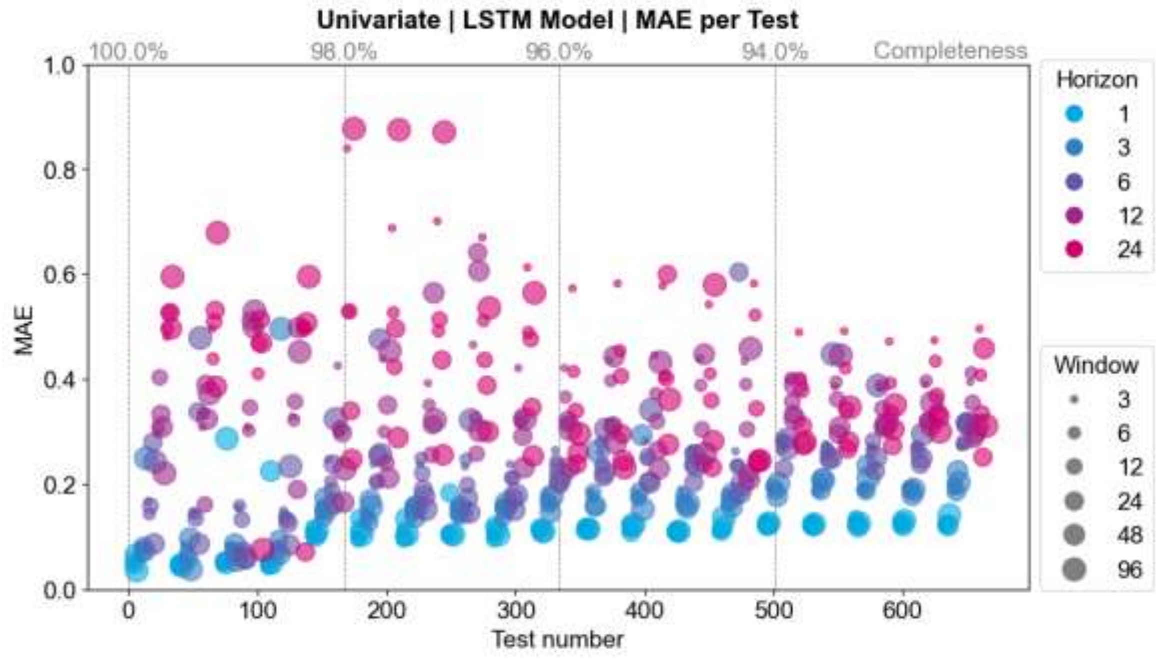
4.4.1 Test Objective 4

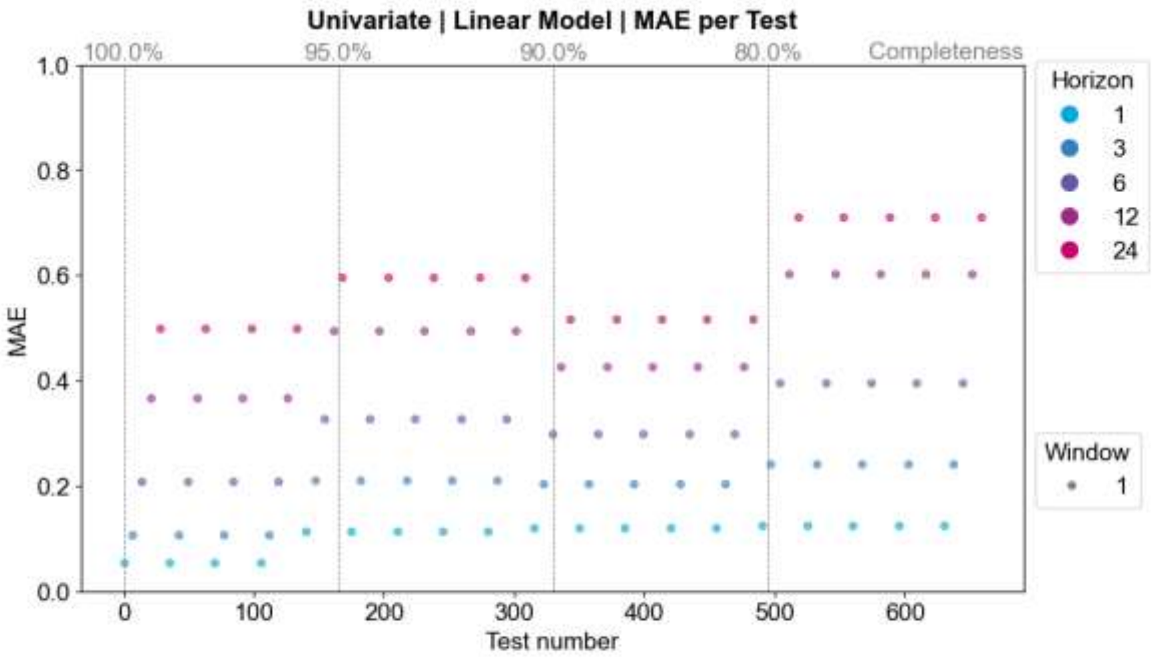
Objective 4: Compare the prediction accuracy for different completeness thresholds for the univariate and multivariate workflow.

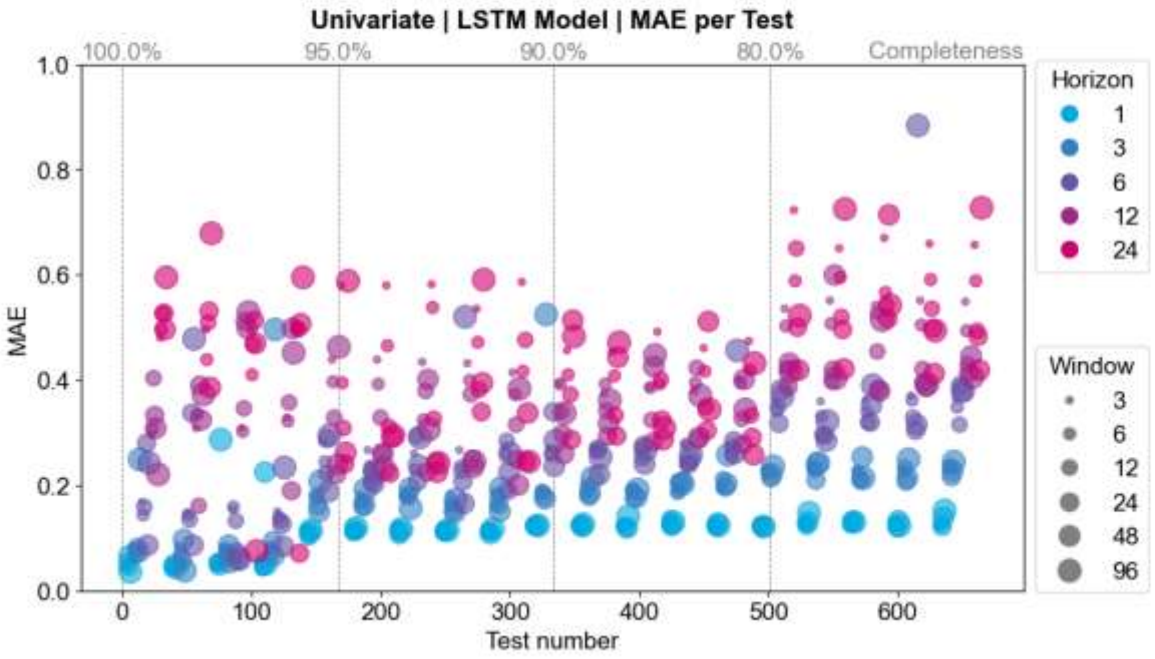
Figure 33 below shows the performance for each different test. The results show that for the multivariate models the performance of the model is highest at the 100%

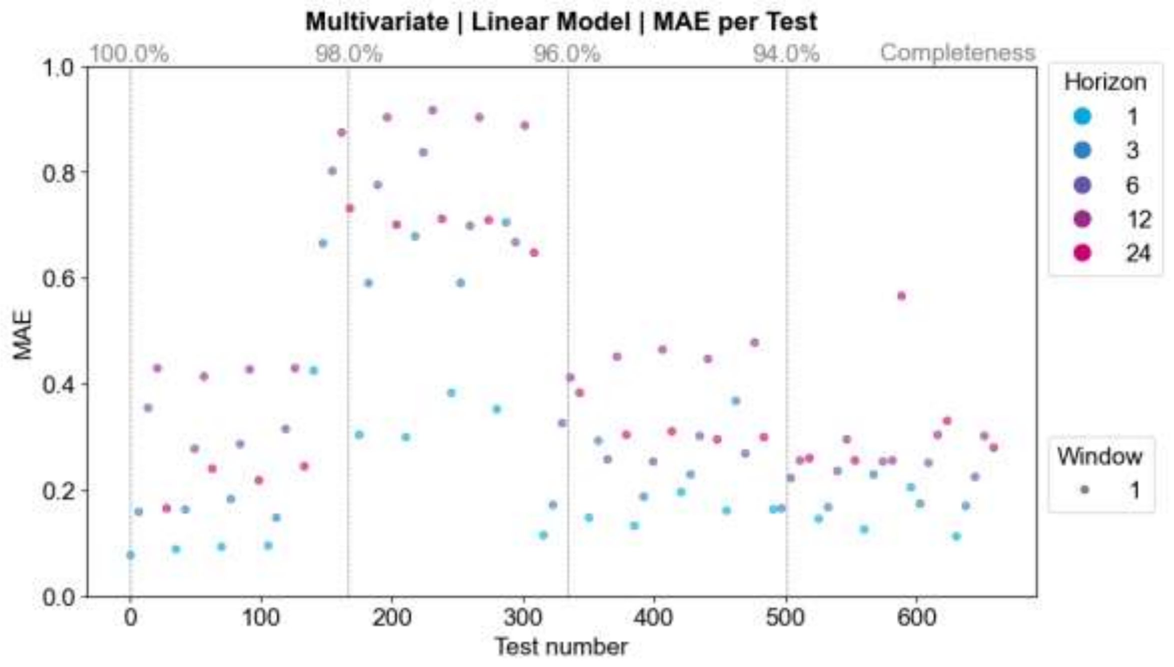
completeness threshold. Whether this is a result of the model overtraining will have to be seen as these results have not been cross validated. What is interesting is that at the lower completeness threshold, the predictions return to the same level at around 96%. This is the point at which the sequence lengths reach about 2000 consecutive time-step. What this suggests is that by adding the periodicity features, the model can take in more low-quality data, without compromising its prediction accuracy.


Univariate | Linear Model | MAE per Test
100.0%
98.0%
96.0%
94.0%
Completeness
MAE
Test number
Horizon
1
3
6
12
24
Window
1
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500
600


Univariate | LSTM Model | MAE per Test
100.0%
98.0%
96.0%
94.0%
Completeness
MAE
Test number
Horizon
1
3
6
12
24
Window
3
6
12
24
48
96
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500
600


Univariate | Linear Model | MAE per Test
100.0%
95.0%
90.0%
80.0%
Completeness
MAE
Test number
Horizon
1
3
6
12
24
Window
1


Univariate | LSTM Model | MAE per Test
100.0%
95.0%
90.0%
80.0%
Completeness
MAE
Test number
Horizon
1
3
6
12
24
Window
3
6
12
24
48
96



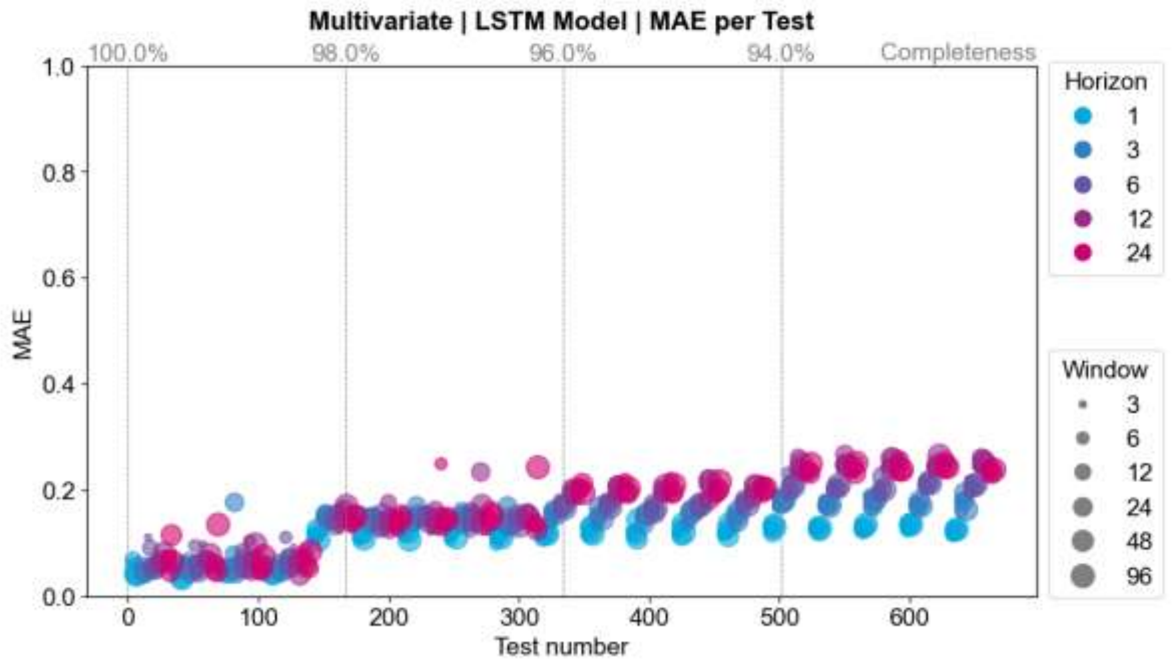
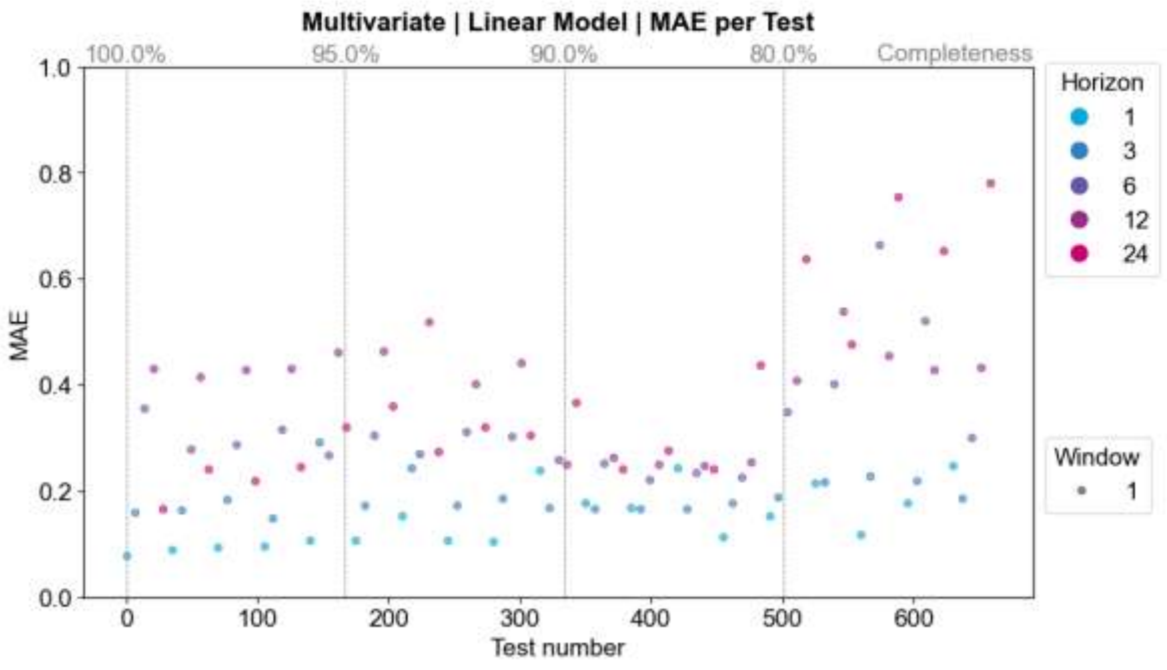
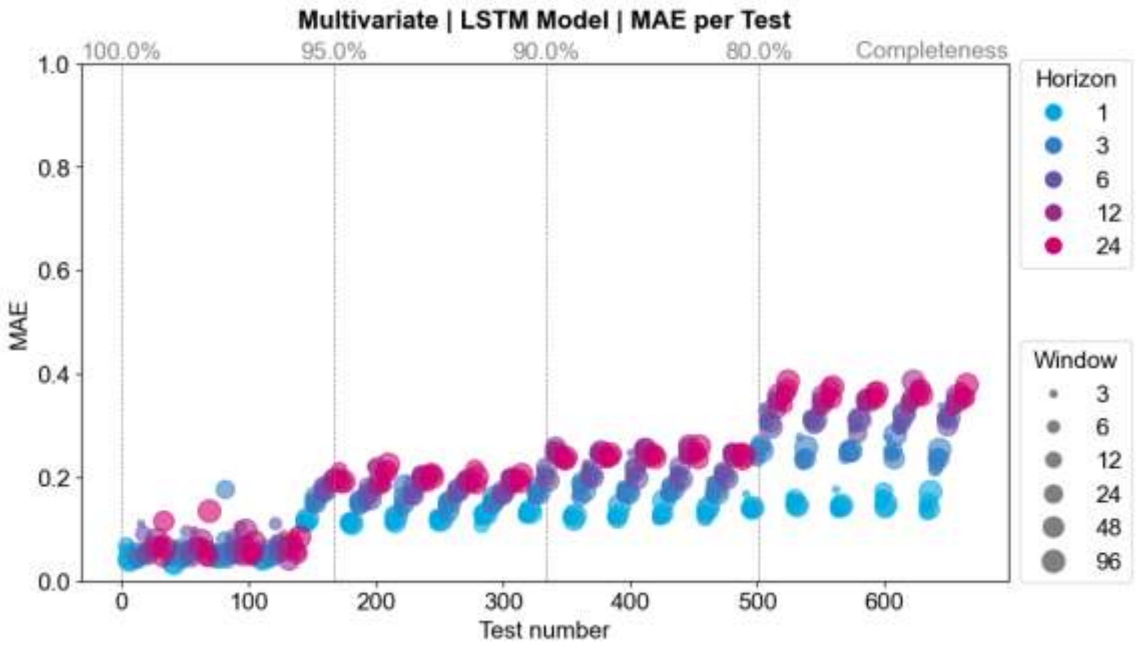


Figure 33 Mean Absolute Error – Multivariate Models

If we then compare this to another test run with lower completeness thresholds, the MAE is still relatively low compared to the univariate workflow.





4.5 Summary of Results

From the analysis of the results, it is evident that data quality has a significant impact on the quality of the results.

- By including periodicity features in the input data much lower MAE can be for tests involving incomplete data.
- The test models cannot accurately predictions beyond a horizon size of 6 (an hour and a half).
- Whilst the multivariate models had much lower MAE, this does not help to predict anomalies.

5 DISCUSSION

This thesis has presented some ideas around anomalies prediction in pedestrian data and how the quality of data affects the ability of the models to make accurate predictions. 1310 model states have been created in the training phase. Each of these model states was then subjected to several analysis runs, and the main narratives and findings for each test objective from have been extracted. In this section, the implications of the findings will be discussed, relating them back to the thesis background and literature review. The first section of the discussion will be focussed on the implications of this research for large scale urban data repositories like the urban observatory. The limitations and future steps will subsequently be discussed.

5.1 Data for AI

Generating accurate predictive models for time-series data is often a time-consuming process. The machine learning approach requires a hyperparameter optimisation which often requires testing a lot of different models. This work has gone some way in developing a method of finding hyperparameters that result in accurate model performance.

This section will discuss the quality of data needed to get high quality results from ‘out of the box’ predictive machine learning models. It is evident from the literature that high quality data is necessary to effectively train machine learning models. It is therefore an important step in any machine learning pipeline to clean the data beforehand. There is an incentive here for open data providers to utilise the anomaly detection algorithms highlighted above to clean their data at the source to offer a more easily deployable product.

The idea is that the data can be taken straight from the source such as from the urban observatory databases in the cities around the UK so that simple machine learning models can be trained and developed quickly for a range of different purposes. The models should be applicable to a variety of different time-series data sources for anomaly detection, such as from air pollution or hydrological sensors.

The results have shown us that for a simple predictive model to identify anomalies requires high data quality for training (see Objective 2). However, this might be a result of the way the data was pre-processed. Moving forward it is necessary to investigate whether interpolation methods or similar would solve the problems in anomaly detection experienced by the models trained on lower quality data. Another option that has resulted from this research is to use the models that are trained on the 98% complete data to label anomalies in the raw data. Objective 2 shows that models trained on this data are able to identify at least some of the anomalies. It is likely a combination of both methods would be necessary, as they target different anomaly types. Interpolation methods can fill in the missing values with relative ease, but where the data is likely wrong as was shown in section 4.2.1 the model can step in.

5.2 Future Work

- Testing the effect of randomly removing data from the training set. Whilst this work has indicated that a completeness of around 98% is the minimum required for anomaly detection, it would be worthwhile testing the effect of randomly removing data from the training set using the hyperparameter combinations that led to successful detection.

- Spatial performance – how do the models perform when evaluated on data from the western sensor? What about another sensor further down the street? When applied to anomaly detection this would allow another type of outlier to be identified. Collective outliers are those that are anomalous to the wider set of data. In this case the wider set of data could be the data from other sensors nearby. Collective outliers occur may occur within the normal range and context for a particular time series, but when combined with other time series, anomalous behaviour is evident.

- Multi-dimensional clustering of performance metrics – in order to identify the best hyperparameters for a model we can run an hyperparameter optimisation similar to the functionality created in this project, but that clusters the hyperparameter values according to post evaluation performance. The objective here will be for the program to arrive at an effective set of hyperparameter values, given only a time-series as an input. A neural layer would then learn from the matrix of hyperparameter values adapt it feeding back into the hyperparameter matrix for training future models. This could built on the work of (Lee et al., 2021) who focus on building a self-adaptive lightweight anomaly detection model for real-time recurrent time series using a simple LSTM.

- Different model architecture – it would be useful to know where the limits on model size come in. One option is to use an LSTM with more layers, or one with an encoder, decoder architecture. Transformer have also shown great promise in an extraordinary number of fields, including time-series analysis. It would be worth comparing the performance of a lightweight transformer with that of the LSTM used in this project.

6 CONCLUSION

In conclusion, this work has provided an exploration of anomaly prediction within pedestrian data and its interplay with data quality's influence on model precision. An investigation covering 1310 model states was undertaken. Key findings and narratives from distinct test objectives have been derived, providing essential insights that reverberate against the backdrop of the thesis' foundational premise and existing literature. The implications of this research extend to large-scale urban data repositories, such as urban observatories, marking a significant stride toward harnessing the potential of data-driven situational awareness.

The discussion underscores the pivotal role of data quality in attaining robust predictive machine learning outcomes, elucidating the need for data cleansing prior to model deployment. The proposition to integrate anomaly detection algorithms for data refinement at the source not only expedites model development but also serves as a pragmatic step for data providers to offer enhanced deployability. Furthermore, this study sheds light on the requirement of high-quality data for training simplistic yet effective predictive models. While the empirical evidence highlights the significance of data quality for training predictive models (Objective 2), the inquiry arises into exploring interpolation techniques' applicability to rectify anomaly detection challenges within lower-quality data models.

There are several future research avenues that have emerged, including the testing of data removal impact on training sets, spatial performance evaluations, and multidimensional clustering of performance metrics. This envisioned trajectory stands to enrich the field of anomaly detection, capitalising on self-adaptive methodologies and diverse model architectures, such as transformers, to foster further advancements.

7 REFERENCES

AGHABOZORGI, S., SHIRKHORSHIDI, A. S. & WAH, T. Y. 2015. Time-series clustering–a decade review. *Information systems,* 53**,** 16-38.

AKINWANDE, O. J., BI, H. & GELENBE, E. 2015. Managing crowds in hazards with dynamic grouping. *IEEE Access,* 3**,** 1060-1070.

ALBANI, V. V. L., ALBANI, R. A. S., MASSAD, E. & ZUBELLI, J. P. 2022. Nowcasting and forecasting COVID-19 waves: the recursive and stochastic nature of transmission. *R Soc Open Sci,* 9**,** 220489.

ALVAREZ CASTRO, D. & FORD, A. 2021. 3D agent-based model of pedestrian movements for simulating COVID-19 transmission in university students. *ISPRS International Journal of Geo-Information,* 10**,** 509.

ASHER, M., OSWALD, Y. & MALLESON, N. 2023. Predicting Pedestrian Counts using Machine Learning. *AGILE: GIScience Series,* 4**,** 18.

BAGNALL, A., LINES, J., BOSTROM, A., LARGE, J. & KEOGH, E. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery,* 31**,** 606-660.

BAI, L., YAO, L., WANG, X., LI, C. & ZHANG, X. 2021. Deep spatial–temporal sequence modeling for multi-step passenger demand prediction. *Future Generation Computer Systems,* 121**,** 25-34.

BAMAQA, A., SEDKY, M., BOSAKOWSKI, T., BASTAKI, B. B. & ALSHAMMARI, N. O. 2022. SIMCD: SIMulated crowd data for anomaly detection and prediction. *Expert Systems with Applications,* 203**,** 117475.

BARLOW, M. 2013. *Real-time big data analytics: Emerging architecture*, " O'Reilly Media, Inc.".

BARR, S., JOHNSON, S., MING, X., PEPPA, M., DONG, N., WEN, Z., ROBSON, C., SMITH, L., JAMES, P. & WILKINSON, D. 2020. Flood-prepared: A nowcasting system for real-time impact adaption to surface water flooding in cities. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* 6**,** 9-15.

BATTY, M. 2008. Fifty years of urban modeling: Macro-statics to micro-dynamics. *The dynamics of complex urban systems: An interdisciplinary approach***,** 1-20.

BATTY, M. 2013. *The new science of cities*, MIT press.

BATTY, M., AXHAUSEN, K. W., GIANNOTTI, F., POZDNOUKHOV, A., BAZZANI, A., WACHOWICZ, M., OUZOUNIS, G. & PORTUGALI, Y. 2012. Smart cities of the future. *The European Physical Journal Special Topics,* 214**,** 481-518.

BATTY, M. & MILTON, R. 2021. A new framework for very large-scale urban modelling. *Urban Studies,* 58**,** 3071-3094.

BELHADI, A., DJENOURI, Y., NØRVÅG, K., RAMAMPIARO, H., MASSEGLIA, F. & LIN, J. C.-W. 2020. Space–time series clustering: Algorithms, taxonomy, and case study on urban smart cities. *Engineering Applications of Artificial Intelligence,* 95**,** 103857.

BENGIO, Y., SIMARD, P. & FRASCONI, P. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks,* 5**,** 157-166.

BLÁZQUEZ-GARCÍA, A., CONDE, A., MORI, U. & LOZANO, J. A. 2021. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys (CSUR),* 54**,** 1-33.

BLOOMFIELD, P. 2004. *Fourier analysis of time series: an introduction*, John Wiley & Sons.

BOYD, J. R. 1996. The essence of winning and losing. *Unpublished lecture notes,* 12**,** 123-125.

BREIMAN, L. 2001. Random forests. *Machine learning,* 45**,** 5-32.

CAO, X., ZHONG, Y., ZHOU, Y., WANG, J., ZHU, C. & ZHANG, W. 2017. Interactive temporal recurrent convolution network for traffic prediction in data centers. *IEEE Access,* 6**,** 5276-5289.

CECAJ, A., LIPPI, M., MAMEI, M. & ZAMBONELLI, F. 2020. Comparing deep learning and statistical methods in forecasting crowd distribution from aggregated mobile phone data. *Applied Sciences,* 10**,** 6580.

CHATFIELD, C. 2003. The Analysis of Time Series.

CHEN, L., GRIMSTEAD, I., BELL, D., KARANKA, J., DIMOND, L., JAMES, P., SMITH, L. & EDWARDES, A. 2021. Estimating Vehicle and Pedestrian Activity from Town and City Traffic Cameras. *Sensors (Basel),* 21**,** 4564.

CHEN, M.-S., HAN, J. & YU, P. S. 1996. Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and data Engineering,* 8**,** 866-883.

CHO, K., VAN MERRIËNBOER, B., BAHDANAU, D. & BENGIO, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

CHOI, K., YI, J., PARK, C. & YOON, S. 2021. Deep learning for anomaly detection in time-series data: review, analysis, and guidelines. *IEEE Access,* 9**,** 120043-120065.

COM. 2023. *Pedestrian Counting System (counts per hour)* [Online]. City of Melbourne. Available: https://data.melbourne.vic.gov.au/explore/dataset/pedestrian-counting-system-monthly-counts-per-hour/information/ [Accessed].

COOK, A. A., MıSıRLı, G. & FAN, Z. 2019. Anomaly detection for IoT time-series data: A survey. *IEEE Internet of Things Journal,* 7**,** 6481-6494.

CUEVAS, E. 2020. An agent-based model to evaluate the COVID-19 transmission risks in facilities. *Computers in biology and medicine,* 121**,** 103827.

DAI, F., HUANG, P., XU, X., QI, L. & KHOSRAVI, M. R. 2020. Spatio-temporal deep learning framework for traffic speed forecasting in IoT. *IEEE Internet of Things Magazine,* 3**,** 66-69.

DANI, M.-C., JOLLOIS, F.-X., NADIF, M. & FREIXO, C. Adaptive threshold for anomaly detection using time series segmentation. Neural Information Processing: 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, Proceedings Part III 22, 2015. Springer, 82-89.

DASGUPTA, D. & FORREST, S. Novelty detection in time series data using ideas from immunology. Proceedings of the international conference on intelligent systems, 1996. Citeseer, 82-87.

DING, H., TRAJCEVSKI, G., SCHEUERMANN, P., WANG, X. & KEOGH, E. 2008. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment,* 1**,** 1542-1552.

DOUCET, A., DE FREITAS, N. & GORDON, N. J. 2001. *Sequential Monte Carlo methods in practice*, Springer.

DRAGHICI, A. & STEEN, M. V. 2019. A Survey of Techniques for Automatically Sensing the Behavior of a Crowd. *ACM Computing Surveys,* 51**,** 1-40.

DU, B., PENG, H., WANG, S., BHUIYAN, M. Z. A., WANG, L., GONG, Q., LIU, L. & LI, J. 2020. Deep Irregular Convolutional Residual LSTM for Urban Traffic Passenger Flows Prediction. *IEEE Transactions on Intelligent Transportation Systems,* 21**,** 972-985.

ELMAN, J. L. 1990. Finding structure in time. *Cognitive science,* 14**,** 179-211.

ENDSLEY, M. R. 1995. Toward a theory of situation awareness in dynamic systems. *Human factors,* 37**,** 32-64.

FALOUTSOS, C., RANGANATHAN, M. & MANOLOPOULOS, Y. 1994. Fast subsequence matching in time-series databases. *ACM Sigmod Record,* 23**,** 419-429.

FELEMBAN, E. A., REHMAN, F. U., BIABANI, S. A. A., AHMAD, A., NASEER, A., MAJID, A. R. M. A., HUSSAIN, O. K., QAMAR, A. M., FALEMBAN, R. & ZANJIR, F. 2020. Digital Revolution for Hajj Crowd Management: A Technology Survey. *IEEE Access,* 8**,** 208583-208609.

FU, R., ZHANG, Z. & LI, L. Using LSTM and GRU neural network methods for traffic flow prediction. 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), 2016. IEEE, 324-328.

FU, X., YU, G. & LIU, Z. 2021. Spatial–temporal convolutional model for urban crowd density prediction based on mobile-phone signaling data. *IEEE Transactions on Intelligent Transportation Systems,* 23**,** 14661-14673.

GEURTS, P. Pattern extraction for time series classification. European conference on principles of data mining and knowledge discovery, 2001. Springer, 115-127.

GIANNONE, D., REICHLIN, L. & SMALL, D. 2008. Nowcasting: The real-time informational content of macroeconomic data. *Journal of Monetary Economics,* 55**,** 665-676.

GOODFELLOW, I., BENGIO, Y. & COURVILLE, A. 2016. *Deep learning*, MIT press.

GUYON, I. & ELISSEEFF, A. 2003. An introduction to variable and feature selection. *Journal of machine learning research,* 3**,** 1157-1182.

HAGHANI, M. & LOVREGLIO, R. 2022. Data-based tools can prevent crowd crushes. *Science,* 378**,** 1060-1061.

HELBING, D. & MOLNAR, P. 1995. Social force model for pedestrian dynamics. *Physical review E,* 51**,** 4282.

HEPPENSTALL, A., CROOKS, A., MALLESON, N., MANLEY, E., GE, J. & BATTY, M. 2021. Future Developments in Geographical Agent-Based Models: Challenges and Opportunities. *Geogr Anal,* 53**,** 76-91.

HEPPENSTALL, A., MALLESON, N. & CROOKS, A. 2016. "Space, the final frontier": How good are agent-based models at simulating individuals and space in cities? *Systems,* 4**,** 9.

HEPPENSTALL, A. J., CROOKS, A. T., SEE, L. M. & BATTY, M. 2011. *Agent-based models of geographical systems*, Springer Science & Business Media.

HINCH, R., PROBERT, W. J., NURTAY, A., KENDALL, M., WYMANT, C., HALL, M., LYTHGOE, K., BULAS CRUZ, A., ZHAO, L. & STEWART, A. 2021. OpenABM-Covid19—An agent-based model for non-pharmaceutical interventions against COVID-19 including contact tracing. *PLoS computational biology,* 17**,** e1009146.

HOANG, M. X., ZHENG, Y. & SINGH, A. K. FCCF: forecasting citywide crowd flows based on big data. 2016. ACM.

HOCHREITER, S. & SCHMIDHUBER, J. 1997. Long short-term memory. *Neural computation,* 9**,** 1735-1780.

HOERTEL, N., BLACHIER, M., BLANCO, C., OLFSON, M., MASSETTI, M., RICO, M. S., LIMOSIN, F. & LELEU, H. 2020. A stochastic agent-based model of the SARS-CoV-2 epidemic in France. *Nature medicine,* 26**,** 1417-1421.

HUNT, B. R., KOSTELICH, E. J. & SZUNYOGH, I. 2007. Efficient data assimilation for spatiotemporal chaos: A local ensemble transform Kalman filter. *Physica D: Nonlinear Phenomena,* 230**,** 112-126.

HYNDMAN, R. J. & ATHANASOPOULOS, G. 2018. *Forecasting: principles and practice*, OTexts.

IDE, T., KATSUKI, T., MORIMURA, T. & MORRIS, R. 2017. City-Wide Traffic Flow Estimation From a Limited Number of Low-Quality Cameras. *IEEE Transactions on Intelligent Transportation Systems,* 18**,** 950-959.

JAIN, A. K. 2010. Data clustering: 50 years beyond K-means. *Pattern recognition letters,* 31**,** 651-666.

JAVED, A., LEE, B. S. & RIZZO, D. M. 2020. A benchmark study on time series clustering. *Machine Learning with Applications,* 1**,** 100001.

JIANG, F., MA, J. & LI, Z. 2022. Pedestrian volume prediction with high spatiotemporal granularity in urban areas by the enhanced learning model. *Sustainable Cities and Society,* 79**,** 103653.

JING, Y., HU, H., GUO, S., WANG, X. & CHEN, F. 2021. Short-Term Prediction of Urban Rail Transit Passenger Flow in External Passenger Transport Hub Based on LSTM-LGB-DRS. *IEEE Transactions on Intelligent Transportation Systems,* 22**,** 4611-4621.

KALPAKIS, K., GADA, D. & PUTTAGUNTA, V. Distance measures for effective clustering of ARIMA time-series. Proceedings 2001 IEEE international conference on data mining, 2001. IEEE, 273-280.

KANTZ, H. & SCHREIBER, T. 2004. *Nonlinear time series analysis*, Cambridge university press.

KERR, C. C., STUART, R. M., MISTRY, D., ABEYSURIYA, R. G., ROSENFELD, K., HART, G. R., NÚÑEZ, R. C., COHEN, J. A., SELVARAJ, P. & HAGEDORN, B. 2021. Covasim: an agent-based model of COVID-19 dynamics and interventions. *PLOS Computational Biology,* 17**,** e1009149.

KIEU, M., NGUYEN, H., WARD, J. A. & MALLESON, N. 2022. Towards real-time predictions using emulators of agent-based models. *Journal of Simulation***,** 1-18.

KRETZ, T. 2011. A level of service scheme for microscopic simulation of pedestrians that integrates queuing, uni-and multi-directional flow situations. *European Transport Research Review,* 3**,** 211-220.

LÄNGKVIST, M., KARLSSON, L. & LOUTFI, A. 2014. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern recognition letters,* 42**,** 11-24.

LECUN, Y., BOTTOU, L., BENGIO, Y. & HAFFNER, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE,* 86**,** 2278-2324.

LEE, M.-C., LIN, J.-C. & ERNST 2021. SALAD: Self-Adaptive Lightweight Anomaly Detection for Real-time Recurrent Time Series. *arXiv pre-print server*.

LI, M., GAO, S., QIU, P., TU, W., LU, F., ZHAO, T. & LI, Q. 2022. Fine-grained crowd distribution forecasting with multi-order spatial interactions using mobile phone data. *Transportation Research Part C: Emerging Technologies,* 144**,** 103908.

LI, X. & WU, Y.-J. 2021. Real-time estimation of pedestrian volume at button-activated midblock crosswalks using traffic controller event-based data. *Transportation research part C: emerging technologies,* 122**,** 102876.

LI, X., YU, Q., ALZAHRANI, B., BARNAWI, A., ALHINDI, A., ALGHAZZAWI, D. & MIAO, Y. 2021. Data fusion for intelligent crowd monitoring and management systems: A survey. *IEEE Access,* 9**,** 47069-47083.

LIN, J., KEOGH, E., LONARDI, S. & CHIU, B. A symbolic representation of time series, with implications for streaming algorithms. Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, 2003. 2-11.

LIU, Y., ZHOU, Y., YANG, K. & WANG, X. 2023. Unsupervised Deep Learning for IoT Time Series. *IEEE Internet of Things Journal*.

LONARDI, J. & PATEL, P. Finding motifs in time series. Proc. of the 2nd Workshop on Temporal Data Mining, 2002. 53-68.

MAKRIDAKIS, S., SPILIOTIS, E. & ASSIMAKOPOULOS, V. 2018. Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PloS one,* 13**,** e0194889.

MALLESON, N., MINORS, K., KIEU, L.-M., WARD, J. A., WEST, A. A. & HEPPENSTALL, A. 2019. Simulating crowds in real time with agent-based modelling and a particle filter. *arXiv preprint arXiv:1909.09397*.

MEHRANG, S., HELANDER, E., PAVEL, M., CHIEH, A. & KORHONEN, I. Outlier detection in weight time series of connected scales. 2015 IEEE international conference on bioinformatics and biomedicine (BIBM), 2015. IEEE, 1489-1496.

MEI, J., LIU, M., WANG, Y. F. & GAO, H. 2016. Learning a Mahalanobis Distance-Based Dynamic Time Warping Measure for Multivariate Time Series Classification. *IEEE Trans Cybern,* 46**,** 1363-74.

MOD 2022. Defence Artificial Intelligence Strategy. Ministry of Defence.

PAPERSWITHCODE. 2023. *Trends* [Online]. Available: https://paperswithcode.com/trends [Accessed 14/07 2023].

PASCANU, R., MIKOLOV, T. & BENGIO, Y. On the difficulty of training recurrent neural networks. International conference on machine learning, 2013. Pmlr, 1310-1318.

PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N. & ANTIGA, L. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems,* 32.

PEDDINTI, V., POVEY, D. & KHUDANPUR, S. A time delay neural network architecture for efficient modeling of long temporal contexts. Sixteenth annual conference of the international speech communication association, 2015.

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R. & DUBOURG, V. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research,* 12**,** 2825-2830.

PEPPA, M. V., BELL, D., KOMAR, T. & XIAO, W. 2018. Urban Traffic Flow Analysis Based on Deep Learning Car Detection from Cctv Image Series. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* XLII-4**,** 499-506.

PEPPA, M. V., KOMAR, T., XIAO, W., JAMES, P., ROBSON, C., XING, J. & BARR, S. 2021. Towards an End-to-End Framework of CCTV-Based Urban Traffic Volume Detection and Prediction. *Sensors (Basel),* 21**,** 629.

SCARSELLI, F., GORI, M., TSOI, A. C., HAGENBUCHNER, M. & MONFARDINI, G. 2008. The graph neural network model. *IEEE transactions on neural networks,* 20**,** 61-80.

SHARMA, A., MCCLOSKEY, B., HUI, D. S., RAMBIA, A., ZUMLA, A., TRAORE, T., SHAFI, S., EL-KAFRAWY, S. A., AZHAR, E. I. & ZUMLA, A. 2023. Global mass gathering events and deaths due to crowd surge, stampedes, crush and physical injuries-lessons from the Seoul Halloween and other disasters. *Travel medicine and infectious disease,* 52.

SHI, W., GOODCHILD, M. F., BATTY, M., KWAN, M.-P. & ZHANG, A. 2021. *Urban Infomatics*, Springer.

SHI, X., GAO, Z., LAUSEN, L., WANG, H. & YEUNG, D.-Y. Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model. 31st Conference on Neural Information Processing Systems, 2017 Long Beach, CA, USA.

SILVA, P. C., BATISTA, P. V., LIMA, H. S., ALVES, M. A., GUIMARÃES, F. G. & SILVA, R. C. 2020. COVID-ABS: An agent-based model of COVID-19 epidemic to simulate health and economic effects of social distancing interventions. *Chaos, Solitons & Fractals,* 139**,** 110088.

SITARAM, D., DALWANI, A., NARANG, A., DAS, M. & AURADKAR, P. A measure of similarity of time series containing missing data using the mahalanobis distance. 2015 second international conference on advances in computing and communication engineering, 2015. IEEE, 622-627.

STILL, K. 2019. *Crowd Safety and Crowd Risk Analysis* [Online]. [Accessed 04/08 2023].

SUN, S., ZHANG, C. & YU, G. 2006. A Bayesian network approach to traffic flow forecasting. *IEEE Transactions on intelligent transportation systems,* 7**,** 124-132.

SUTTON, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning,* 3**,** 9-44.

TANG, D. & MALLESON, N. 2022. Data assimilation with agent-based models using Markov chain sampling. *arXiv pre-print server*.

TAYLOR, L. 2016. No place to hide? The ethics and analytics of tracking mobility using mobile phone data. *Environment and Planning D: Society and Space,* 34**,** 319-336.

TERNES, P., WARD, J. A., HEPPENSTALL, A., KUMAR, V., KIEU, L.-M. & MALLESON, N. 2022. Data assimilation and agent-based modelling: towards the incorporation of categorical agent parameters. *Open Research Europe,* 1.

THURNER, S., HANEL, R. & KLIMEK, P. 2018. *Introduction to the theory of complex systems*, Oxford University Press.

UO. 2023. *Urban Observatory* [Online]. Newcastle University. Available: https://newcastle.urbanobservatory.ac.uk/ [Accessed].

VANDERPLAS, J. T. 2018. Understanding the lomb–scargle periodogram. *The Astrophysical Journal Supplement Series,* 236**,** 16.

VIRTANEN, P., GOMMERS, R., OLIPHANT, T. E., HABERLAND, M., REDDY, T., COURNAPEAU, D., BUROVSKI, E., PETERSON, P., WECKESSER, W. & BRIGHT, J. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods,* 17**,** 261-272.

VIVACITY. 2023. *Smart Traffic Monitoring* [Online]. VivaCity Labs. Available: https://vivacitylabs.com/products/smart-traffic-monitoring-solution/ [Accessed].

WANG, M., WANG, Q. & ZAKI, T. A. 2019. Discrete adjoint of fractional-step incompressible Navier-Stokes solver in curvilinear coordinates and application to data assimilation. *Journal of Computational Physics,* 396**,** 427-450.

WANG, Z., YAN, W. & OATES, T. Time series classification from scratch with deep neural networks: A strong baseline. 2017 International joint conference on neural networks (IJCNN), 2017. IEEE, 1578-1585.

WIRZ, M., FRANKE, T., ROGGEN, D., MITLETON-KELLY, E., LUKOWICZ, P. & TRÖSTER, G. 2013. Probing crowd density through smartphones in city-scale mass gatherings. *EPJ Data Science,* 2**,** 1-24.

WOLFRAM, S. 2002. *A new kind of science*, Wolfram media Champaign.

WU, C.-H., HO, J.-M. & LEE, D.-T. 2004. Travel-time prediction with support vector regression. *IEEE transactions on intelligent transportation systems,* 5**,** 276-281.

WU, Q. & LIN, H. 2019. Daily urban air quality index forecasting based on variational mode decomposition, sample entropy and LSTM neural network. *Sustainable Cities and Society,* 50.

XIE, P., LI, T., LIU, J., DU, S., YANG, X. & ZHANG, J. 2020. Urban flow prediction from spatiotemporal data using machine learning: A survey. *Information Fusion,* 59**,** 1-12.

YADAV, S., GULIA, P., GILL, N. S. & CHATTERJEE, J. M. 2022. A real-time crowd monitoring and management system for social distance classification and healthcare using deep learning. *Journal of Healthcare Engineering,* 2022.

8 GLOSSARY

8.1 Time-Series Analysis

8.1.1 Indexing

The process of identifying and assigning identifiers to time series data in a database, aiding in rapid data retrieval. It involves using a measure of similarity or dissimilarity $D(Q, C)$ between a query time series Q and the time series in the database DB (Faloutsos et al., 1994).

8.1.2 Clustering

The process of grouping time series data based on similarity. This often involves using a measure of similarity or dissimilarity $D(Q, C)$ to identify and group related time series data in a database DB (Kalpakis et al., 2001).

8.1.3 Classification

The process of assigning an unlabelled time series Q to one of two or more predefined classes based on certain characteristics or patterns in the data (Geurts, 2001).

8.1.4 Anomaly Detection

The process of identifying anomalies or deviations from a defined “normal” behavior within a given time series Q. This often involves the use of models that define normal behavior and algorithms that detect deviations from these models (Dasgupta and Forrest, 1996).

8.2 Machine Learning

8.2.1 RF (Random Forest)

RF is a machine learning technique that builds multiple decision trees and merges them together to get more accurate and stable predictions. The principle behind the RF algorithm is that a group of "weak learners" can come together to form a "strong learner". Each tree in the random forest is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. Also, when splitting a node during the construction of the tree, the split is chosen from a random subset of the features, which makes the forest robust against overfitting. RF is widely used for both classification and regression tasks (Breiman, 2001).

8.2.2 Activation Functions

A critical component of artificial neural networks that introduce non-linear properties into the network's mapping from input to output. By doing so, they enable the network to learn and represent more complex patterns in the data. Activation functions range from simple binary step functions to more complex functions like sigmoid, hyperbolic tangent (tanh), Rectified Linear Unit (ReLU), and others. The choice of activation function can significantly affect the network's training process and final performance. One of the critical roles of an activation function is to limit the output of the neurons, adding an element of normalization within the network (Goodfellow et al., 2016).

8.2.3 ANN (Artificial Neural Network)

ANNs are computational models inspired by biological neural networks and form the backbone of modern machine learning. These networks consist of interconnected layers of nodes or "neurons" that transmit and transform information from the input to the output layer. Each node takes a weighted sum of its inputs, applies an activation function, and sends the result to nodes in the next layer. The weights are learned through a process called backpropagation, where the network adjusts its weights to minimize the difference between the actual and predicted outputs. ANNs can approximate complex functions and are widely used for tasks like classification, regression, and clustering (Goodfellow et al., 2016).

8.2.4 CNN (convolution neural network)

CNNs are a category of artificial neural networks specially designed to process grid-like data such as images. They consist of one or more convolutional layers, often followed by pooling layers, and then fully connected layers as in a traditional neural network. The key feature of CNNs is their ability to preserve the spatial structure of the data. The convolution operation allows CNNs to automatically and adaptively learn spatial hierarchies of features, which makes them highly successful in tasks where there is a need to identify spatial patterns in high-dimensional data (LeCun et al., 1998).

8.2.5 RNN (Recurrent Neural Network)

RNNs are a type of neural networks specifically designed to handle sequential data. They operate by passing information from one step in the sequence to the next, creating a form of internal memory. Each timestep's hidden state is computed using the current input and the previous timestep's hidden state, allowing the network to capture temporal dependencies in the data. However, RNNs struggle with learning long-term dependencies due to the vanishing and exploding gradient problem (Elman, 1990).

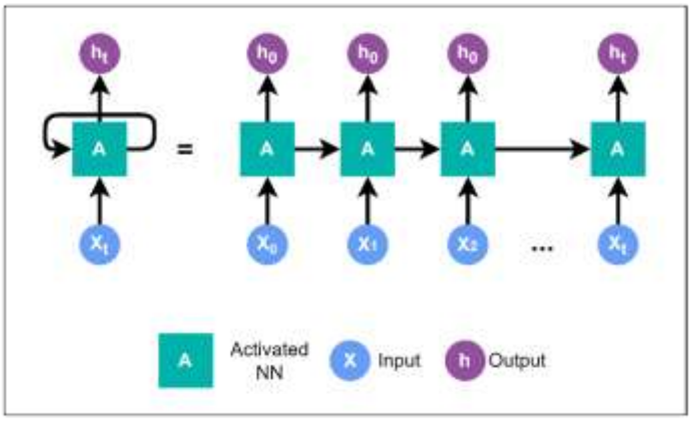


Figure 34 RNN Architecture

8.2.6 LSTM (long short-term memory)

A variant of RNNs, LSTMs process sequential data by leveraging a current data point and the previous timestep's hidden state. The LSTM model's novelty lies in its cell state, a structure allowing information to flow with minimal transformation (c_{t-1} and c_t in Figure 35). This cell state is modulated via gates, mechanisms that control information flow through sigmoid functions and pointwise multiplication, dictating what proportion of the information should be retained or discarded (Hochreiter and Schmidhuber, 1997).

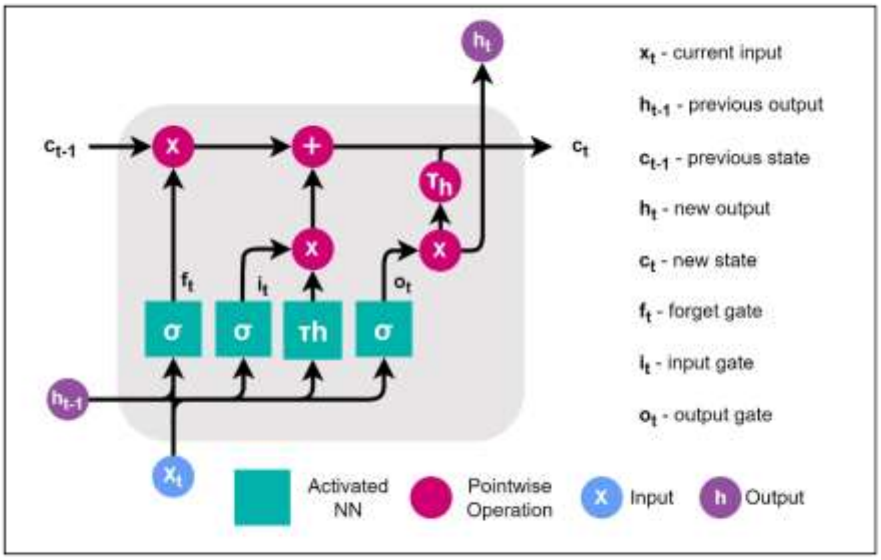


Figure 35 LSTM unit

8.2.7 GRU (Gated Recurrent Unit)

GRUs are a variation of RNNs designed to capture long-term dependencies more effectively. Similar to LSTMs, GRUs utilise gating mechanisms to control information flow across timesteps. However, GRUs simplify the LSTM model by using only two gates: an update gate, which determines how much information from the previous hidden state should be kept, and a reset gate, which controls how much of the past information should be forgotten. This design helps combat the vanishing gradient problem and enables the network to learn longer sequences, while often being more computationally efficient than LSTMs (Cho et al., 2014).

8.2.8 GNN (Graph Neural Network)

GNNs are a class of neural networks designed to operate over data structured as graphs. In GNNs, nodes aggregate information from their neighbours via message-passing mechanisms, using learned functions to transform and combine this information. This iterative local aggregation allows GNNs to capture the complex relational structures present in graph data. However, while enabling learning on non-Euclidean data, the network's performance is contingent on the graph's structure, with poorly constructed graphs often leading to sub-optimal performance. GNNs are extensively used in such network analysis where data is naturally represented as graphs (Scarselli et al., 2008).

9 APPENDIX

9.1 Methodology

9.1.1 preprocess_data()

```
def preprocess_data(
    df: pd.DataFrame,
    completeness_threshold: float,
    frequency: str = None,
    additional_features: list = None,
    show_prints=True,
    remove_dir=True,
    daily_completeness=True,
    consecutive_days=True,
    term_dates=True,
    periodicity=True,
    scale=True,
    resample=False,
) -> dict:
    """
    Preprocess a given time series dataframe based on various criteria and operations.

    Parameters:
    - df (pd.DataFrame): The input dataframe, expected to have a datetime index.
    - completeness_threshold (float): Threshold for the data completeness. If data is below this
    threshold, it may be dropped or processed accordingly.
    - frequency (str, optional): Desired frequency for resampling the data. Defaults to None, implying
    no resampling.
    - additional_features (list, optional): List of additional features to be considered. Defaults to
    None.
    - show_prints (bool, optional): If True, shows print statements. Defaults to True.
    - remove_dir (bool, optional): If True, removes the directionality from the data. Defaults to True.
    - daily_completeness (bool, optional): If True, selects based on daily completeness threshold.
    Defaults to True.
    - consecutive_days (bool, optional): If True, selects data with a certain number of consecutive
    days. Defaults to True.
    - term_dates (bool, optional): If True, adds term dates information. Defaults to True.
    - periodicity (bool, optional): If True, adds periodicity features to the data. Defaults to True.
    - scale (bool, optional): If True, scales the data using a standard scaler. Defaults to True.
    - resample (bool, optional): If True, resamples the data based on the provided frequency. Defaults
    to False.

    Returns:
    - dict: A dictionary containing:
      * "data": The preprocessed dataframe.
      * "index": The index of the dataframe.
      * "columns": The columns of the dataframe.

    Example:
    --------
    >>> data = pd.DataFrame(...)
    >>> preprocessed = preprocess_data(data, completeness_threshold=0.8)

    Notes:
    ------
    - The dataframe is expected to have a datetime index.
    - The function will modify the input dataframe based on the provided flags and thresholds.
    """
    if remove_dir:
        df = remove_directionality(df, additional_features)

    if daily_completeness:
        df = select_daily_completeness_threshold(
            df, completeness_threshold, show_prints=show_prints
        )

    if consecutive_days:
        df = find_longest_sequence(df, completeness_threshold, show_prints=show_prints)

    if term_dates:
        df = add_term_dates(df)

    if periodicity:
        df = add_periodicity_features(df)

    if scale:
        scaled_data = scale_data(df)

    if resample:
        df = resample_frequency(df, frequency)
        return {"data": df, "index": df.index, "columns": df.columns}
    else:
        df.drop(columns=["date"], inplace=True)
        return {"data": scaled_data, "index": df.index, "columns": df.columns}
```

9.1.2 check_completeness_daily()

```
def check_completeness_daily(
    df: pd.DataFrame,
    df_dayofyear: int,
    df_year: int,
    day_number: int,
    days: int,
    year: int,
    completeness: float,
) -> pd.DataFrame:
    """
    Checks whether data completeness for each day in a given sequence meets a specified threshold.

    Parameters:
    ----------
    df (pd.DataFrame):
        The dataframe containing the time series data.
    df_dayofyear (pd.Series):
        A series indicating the day of the year for each timestamp in the dataframe.
    df_year (pd.Series):
        A series indicating the year for each timestamp in the dataframe.
    day_number (int):
        The starting day of the sequence to be checked.
    days (int):
        The number of days in the sequence to be checked.
    year (int):
        The year of the sequence to be checked.
    completeness (float):
        The data completeness threshold (as a fraction of a full day's data).

    Returns:
    ----------
    bool:
        True if the data completeness for each day in the sequence meets or exceeds the threshold; False
otherwise.

    Notes:
    ------
    - The function assumes there are 24*4 = 96 data points for a full day (i.e., data at 15-minute
intervals).
    """
    for i in range(days):
        ts_data = df[(df_dayofyear == day_number + i) & (df_year == year)]
        if len(ts_data) < completeness * 24 * 4:
            return False
    return True
```

9.1.3 find_longest_sequence()

```
def find_longest_sequence(
    df: pd.DataFrame, completeness: float, show_prints: bool = True
) -> pd.DataFrame:
    """
    Identifies the longest continuous sequence of days where data completeness meets the specified threshold.

    Parameters:
    ----------
    df (pd.DataFrame):
        The dataframe containing the time series data.
    completeness (float):
        The data completeness threshold.
    show_prints (bool, optional):
        If True, prints statements detailing the identified sequence. Default is True.

    Returns:
    ----------
    pd.DataFrame:
        A dataframe containing the longest continuous sequence of days where data completeness meets the
threshold.
        If no such sequence is found, returns None.
    Example:
    --------
    >>> data = pd.DataFrame({...}, index=pd.date_range(...))
    >>> sequence = find_longest_sequence(data, 0.9)

    Notes:
    ------
    - The function uses the `check_completeness_daily` function to assess data completeness for each
potential sequence.
    - The dataframe should have a DatetimeIndex.
    """
    df_dayofyear = df.index.to_series().dt.dayofyear
    df_year = df.index.to_series().dt.year
    unique_years = df_year.unique()

    days = 0  # Initialize days counter
    max_day_sequence_start = None  # Initialize starting day of max sequence
    max_sequence_year = None  # Initialize year of max sequence

    while True:
        for year in unique_years:
            min_day_number = df_dayofyear[df_year == year].min()
            max_day_number = df_dayofyear[df_year == year].max()
            # Check each possible sequence starting from each day of the year
            for day_number in range(min_day_number, max_day_number - days + 1):
                if check_completeness_daily(
                    df, df_dayofyear, df_year, day_number, days, year, completeness
                ):
                    days += 1  # Increase the days counter
                    max_day_sequence_start = (
                        day_number  # Update starting day of max sequence
                    )
                    max_sequence_year = year  # Update year of max sequence
                    break
            else:
                continue
            break
        else:
            if days > 0:
                sequence_df = df[
                    (df_dayofyear >= max_day_sequence_start)
                    & (df_dayofyear < max_day_sequence_start + days)
                    & (df_year == max_sequence_year)
                ]
                if show_prints:
                    print(f"Maximum consecutive days: {days - 1}")
                    print(
                        f"Starting from day number {max_day_sequence_start} in {max_sequence_year}"
                    )
                return sequence_df
            else:
                if show_prints:
                    print("No consecutive days found.")
                return None  # If no sequence was found, return None
```

9.1.4 add_periodicity_features()

```
def add_periodicity_features(df: pd.DataFrame) -> pd.DataFrame:
    """
    Add periodicity-based features to a time series dataframe to capture potential cyclical patterns.

    The function generates sine and cosine features based on daily, half-day, quarter-yearly,
    and yearly periods. These are useful for capturing day-night cycles, seasonal changes,
    and other cyclical behaviors observed in time series data.

    Parameters:
    ----------
    df (pd.DataFrame):
        The input dataframe with a DatetimeIndex containing the timestamps for which periodicity
        features are to be generated.

    Returns:
    ----------
    pd.DataFrame:
        A dataframe containing the original data and the following additional columns:
        - 'sin_day': Sine value representing the time of day with a 24-hour period.
        - 'cos_day': Cosine value representing the time of day with a 24-hour period.
        - 'sin_half_day': Sine value representing the time of day with a 12-hour period.
        - 'cos_half_day': Cosine value representing the time of day with a 12-hour period.
        - 'sin_quarter': Sine value representing the day of the year with a ~91.25-day period
        (quarter-yearly).
        - 'cos_quarter': Cosine value representing the day of the year with a ~91.25-day period
        (quarter-yearly).
        - 'sin_year': Sine value representing the day of the year with a 365-day period.
        - 'cos_year': Cosine value representing the day of the year with a 365-day period.

    Example:
    --------
    >>> data = pd.DataFrame({'value': [...]}, index=pd.date_range(start="2022-01-01", periods=365))
    >>> enhanced_data = add_periodicity_features(data)

    Notes:
    ------
    - The input dataframe is expected to have a datetime index.
    - This function does not mutate the original dataframe. It returns a new dataframe with added features.
    """
    # Make a copy of the input DataFrame to avoid modifying it
    df = df.copy()
    dt_index = df.index
    df["sin_day"] = np.sin(2 * np.pi * dt_index.hour / 24)
    df["cos_day"] = np.cos(2 * np.pi * dt_index.hour / 24)

    df["sin_half_day"] = np.sin(2 * np.pi * dt_index.hour / 12)
    df["cos_half_day"] = np.cos(2 * np.pi * dt_index.hour / 12)

    df["sin_quarter"] = np.sin(2 * np.pi * dt_index.dayofyear / 91.25)
    df["cos_quarter"] = np.cos(2 * np.pi * dt_index.dayofyear / 91.25)

    df["sin_year"] = np.sin(2 * np.pi * dt_index.dayofyear / 365)
    df["cos_year"] = np.cos(2 * np.pi * dt_index.dayofyear / 365)
    return df
```

9.1.5 LinearModel()

```
class LinearModel(nn.Module):
    """
    A simple linear regression model suitable for time series forecasting.

    Parameters:
    - input_size (int): Number of input features.

    Attributes:
    - linear (nn.Linear): A linear layer that transforms input features into a single output.

    Methods:
    - forward(x: torch.Tensor) -> torch.Tensor: Implements the forward propagation of the model.

    Example:
    --------
    >>> model = LinearModel(input_size=10)
    >>> input_data = torch.randn(32, 10)  # Batch of 32, each with 10 features
    >>> output = model(input_data)

    Notes:
    ------
    - The forward method can process both 2D (batch_size, num_features) and
      3D (batch_size, sequence_len, num_features) input tensors. If the input is 3D,
      it gets reshaped to 2D.
    """

    def __init__(self, input_size):
        super(LinearModel, self).__init__()
        self.linear = nn.Linear(input_size, 1)

    def forward(self, x):
        # If x is 3D (batch_size, sequence_len, num_features), we might need to reshape it
        x = x.reshape(x.size(0), -1)
        return self.linear(x)
```

9.1.6 LSTMModel()

```
class LSTMModel(nn.Module):
    """
    LSTM-based model designed for time series forecasting. Suitable for both univariate and
    multivariate time series.

    Parameters:
    - feature_dim (int): Number of expected features in the input `x`.
    - hidden_size (int, optional): Number of features in the hidden state. Default: 50.
    - output_dim (int, optional): Number of features in the output. Default: 1.
    - num_layers (int, optional): Number of recurrent layers. Default: 1.

    Attributes:
    - lstm (nn.LSTM): LSTM layer.
    - linear (nn.Linear): Linear layer to produce the final output.

    Methods:
    - forward(x: torch.Tensor) -> torch.Tensor: Implements the forward propagation of the model.

    Example:
    --------
    >>> model = LSTMModel(feature_dim=10)
    >>> input_data = torch.randn(32, 7, 10)  # Batch of 32, sequence length of 7, each with 10 features
    >>> output = model(input_data)
    """

    def __init__(self, feature_dim, hidden_size=50, output_dim=1, num_layers=1):
        super().__init__()
        self.lstm = nn.LSTM(feature_dim, hidden_size, num_layers, batch_first=True)
        self.linear = nn.Linear(hidden_size, output_dim)

    def forward(self, x):
        """
        Forward propagation method for the LSTM model.

        Args:
        - x (torch.Tensor): Input tensor with sequences. Expected shape: [batch_size, sequence_length,
        feature_dim].

        Returns:
        - torch.Tensor: Output tensor with predictions. Shape: [batch_size, output_dim].
        """
        x, _ = self.lstm(x)
        x = self.linear(x)
        return x[:, -1, :]  # Selecting the last output of the sequence
```

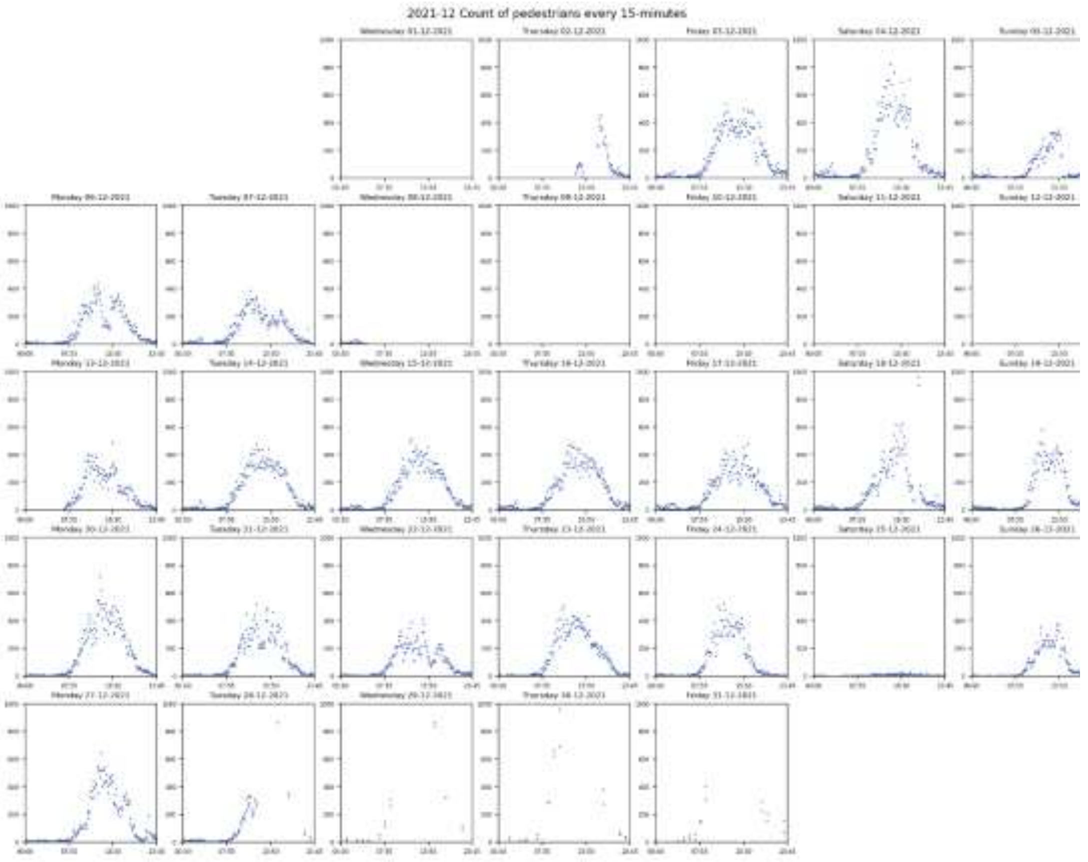
9.1.7 sliding_windows()

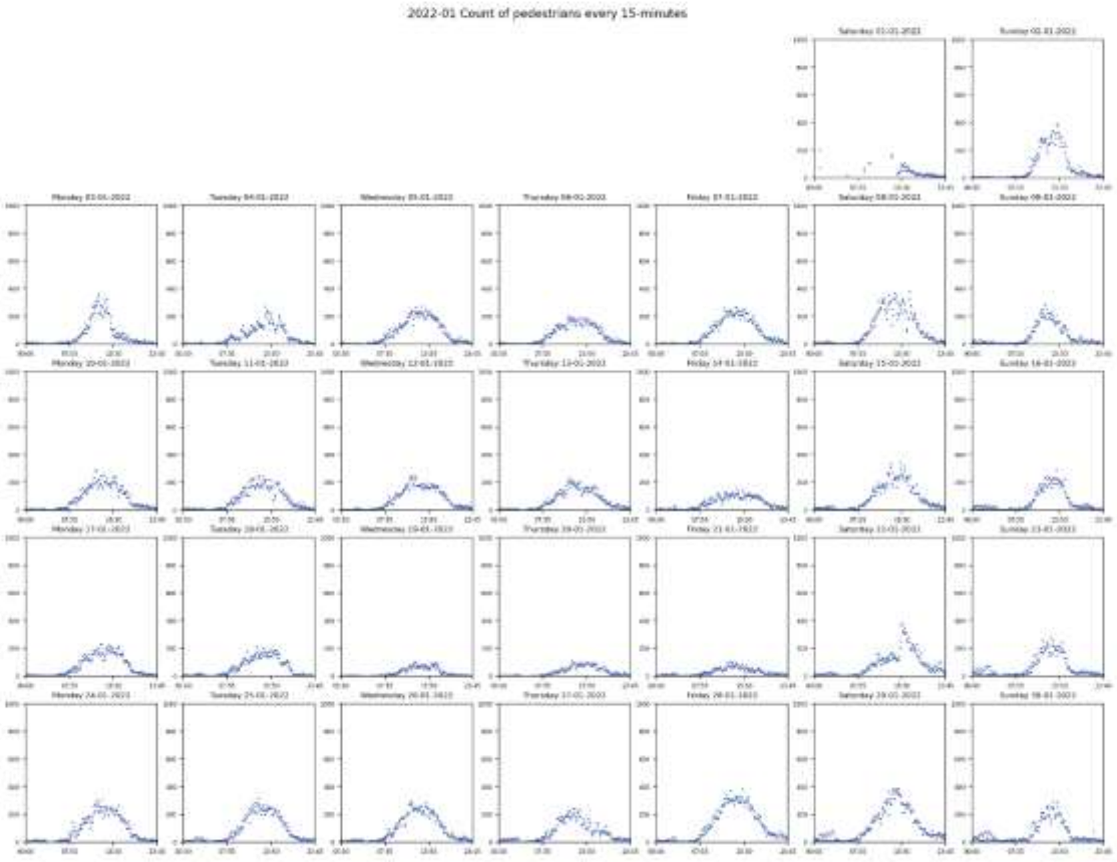
```
def sliding_windows(
    data: np.ndarray,
    window_size: int,
    input_feature_indices: list,
    target_feature_index: int,
    horizon: int,
    stride=1,
    shapes=False,
)-> tuple:
    """
    Generate sliding windows from the provided time-series data for sequence learning.

    Parameters:
    - data (np.ndarray): The time-series data from which windows will be generated.
    - window_size (int): Specifies the size of each sliding window.
    - input_feature_indices (list of ints): The indices of features to be considered as input.
    - target_feature_index (int): Index of the feature that needs to be predicted.
    - horizon (int): How many steps ahead the prediction should be.
    - stride (int, optional): Steps between the start of each window. Defaults to 1.
    - shapes (bool, optional): If set to True, it prints shapes of input and target for the first
    window. Defaults to False.

    Returns:
    - tuple: Contains inputs and targets as torch tensors.
    """
    inputs = []
    targets = []
    for i in range(0, len(data) - window_size - horizon + 1, stride):
        input_data = data[
            i : i + window_size, input_feature_indices
        ]  # selects only the features indicated by input_feature_indices
        target_data = data[
            i + window_size + horizon - 1, target_feature_index
        ]  # selects the feature indicated by target_feature_index, horizon steps ahead
        if i == 0 and shapes:
            print(
                f"Input shape: {input_data.shape} | Target shape: {target_data.shape}"
            )
        inputs.append(input_data)
        targets.append(target_data)

    # Convert lists of numpy arrays to numpy arrays
    inputs = np.array(inputs)
    targets = np.array(targets)

    return torch.tensor(inputs), torch.tensor(targets)
```

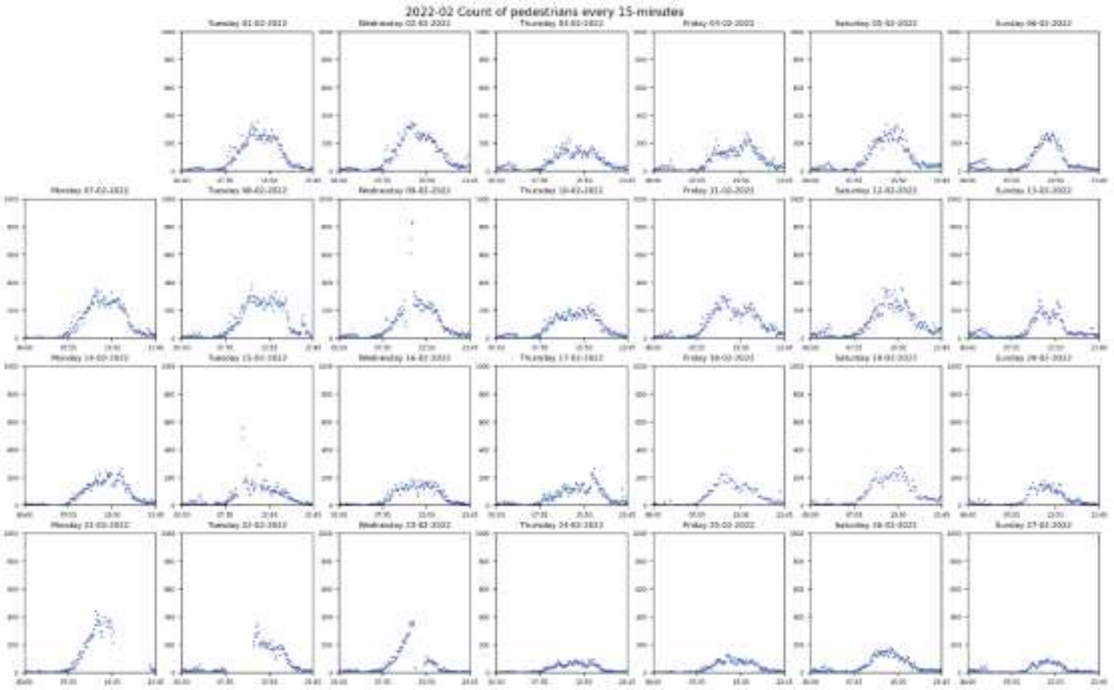
9.1.8 prepare_dataloaders()

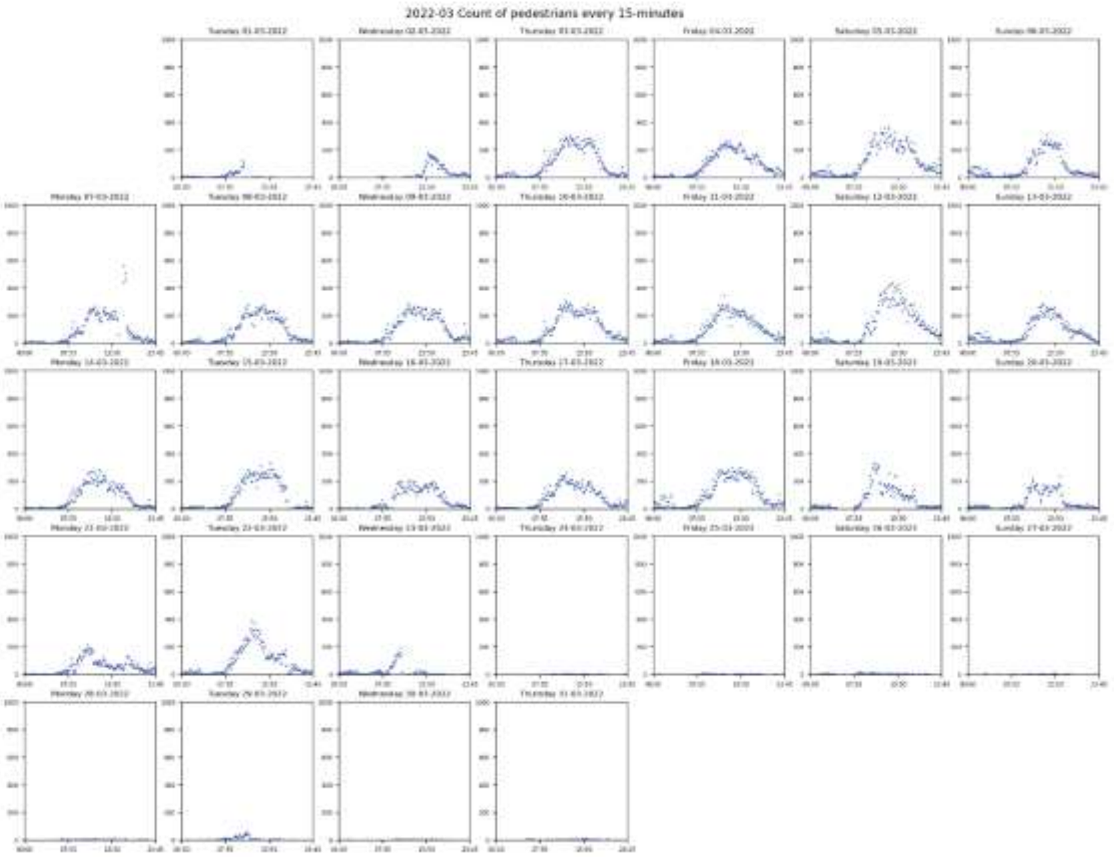
```
def prepare_dataloaders(
    data: np.ndarray,
    window_size: int,
    input_feature_indices: list,
    target_feature_index: int,
    horizon: int,
    stride: int,
    batch_size: int,
    shuffle=False,
    num_workers=0,
) -> tuple(Dataset, Dataset, np.ndarray, np.ndarray, np.ndarray, np.ndarray):
    """
    Prepares training and test dataloaders using sliding windows on the given time-series data.

    Parameters:
    - data (np.ndarray): Time-series data.
    - window_size (int): Size of each sliding window.
    - input_feature_indices (list of ints): Indices of features to be considered as input.
    - target_feature_index (int): The index of the feature that needs to be predicted.
    - horizon (int): Steps ahead for the prediction.
    - stride (int): Steps between the start of each window.
    - batch_size (int): Number of samples per batch to load.
    - shuffle (bool, optional): Whether to shuffle the data samples. Defaults to False.
    - num_workers (int, optional): Number of subprocesses to use for data loading. Defaults to 0.

    Returns:
    - tuple: Contains train DataLoader, test DataLoader, test inputs, test targets, train inputs, and
    train targets.
    """
    inputs, targets = sliding_windows(
        data=data,
        window_size=window_size,
        input_feature_indices=input_feature_indices,
        target_feature_index=target_feature_index,
        horizon=horizon,
        stride=stride,
    )
```

9.1.9 Full Overview of Raw Data

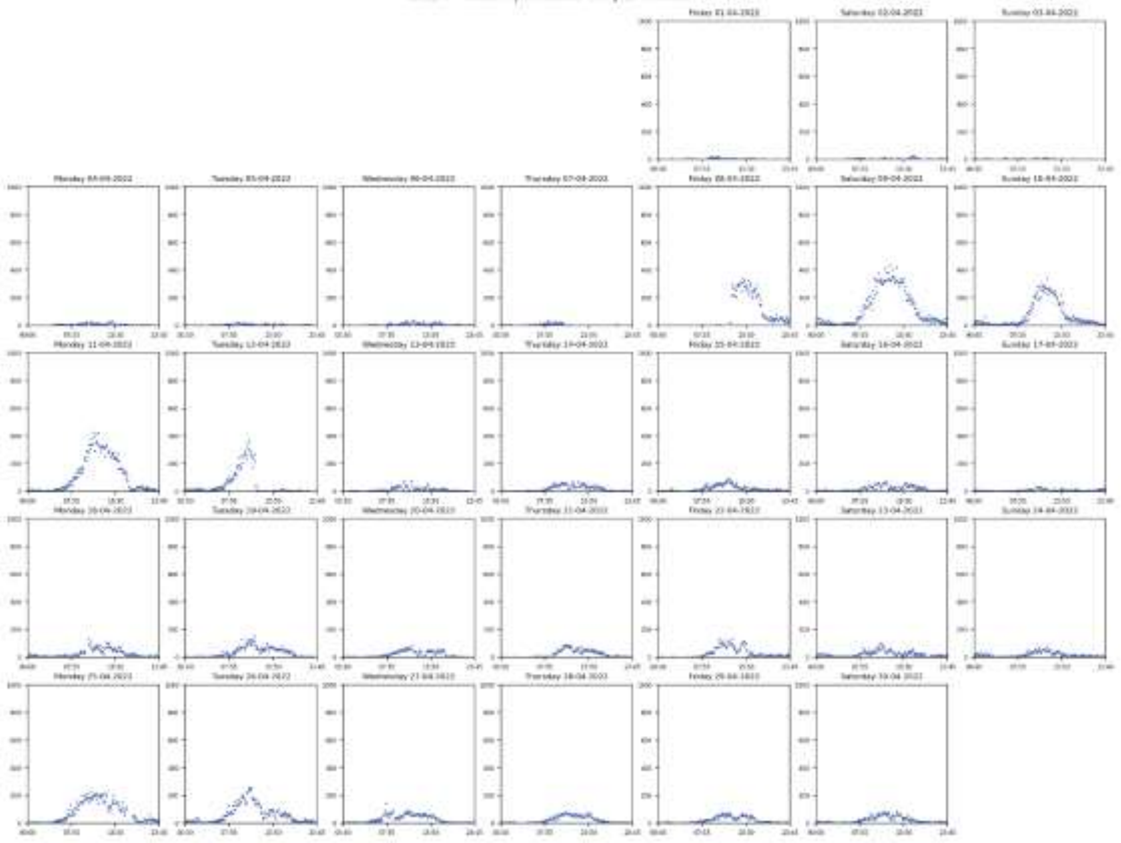

2021-12 Count of pedestrians every 15-minutes


2022-01 Count of pedestrians every 15-minutes

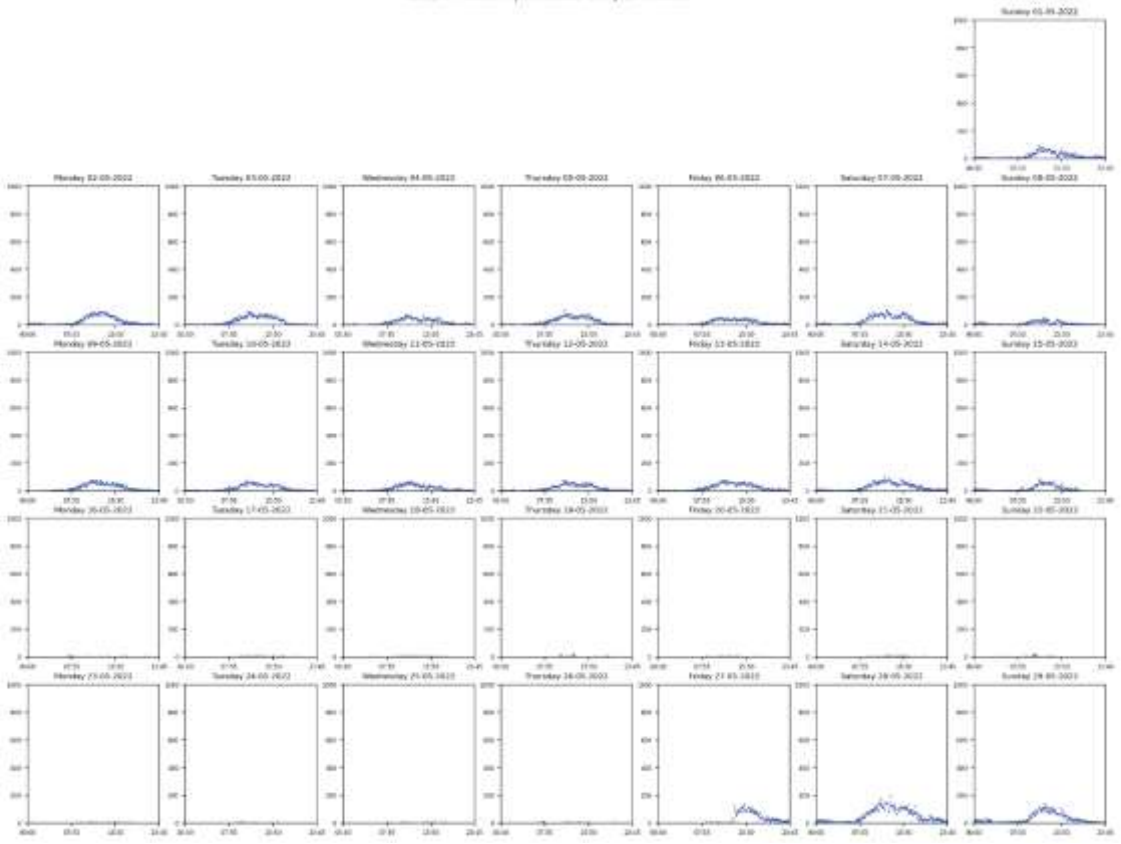

2022-02 Count of pedestrians every 15-minutes

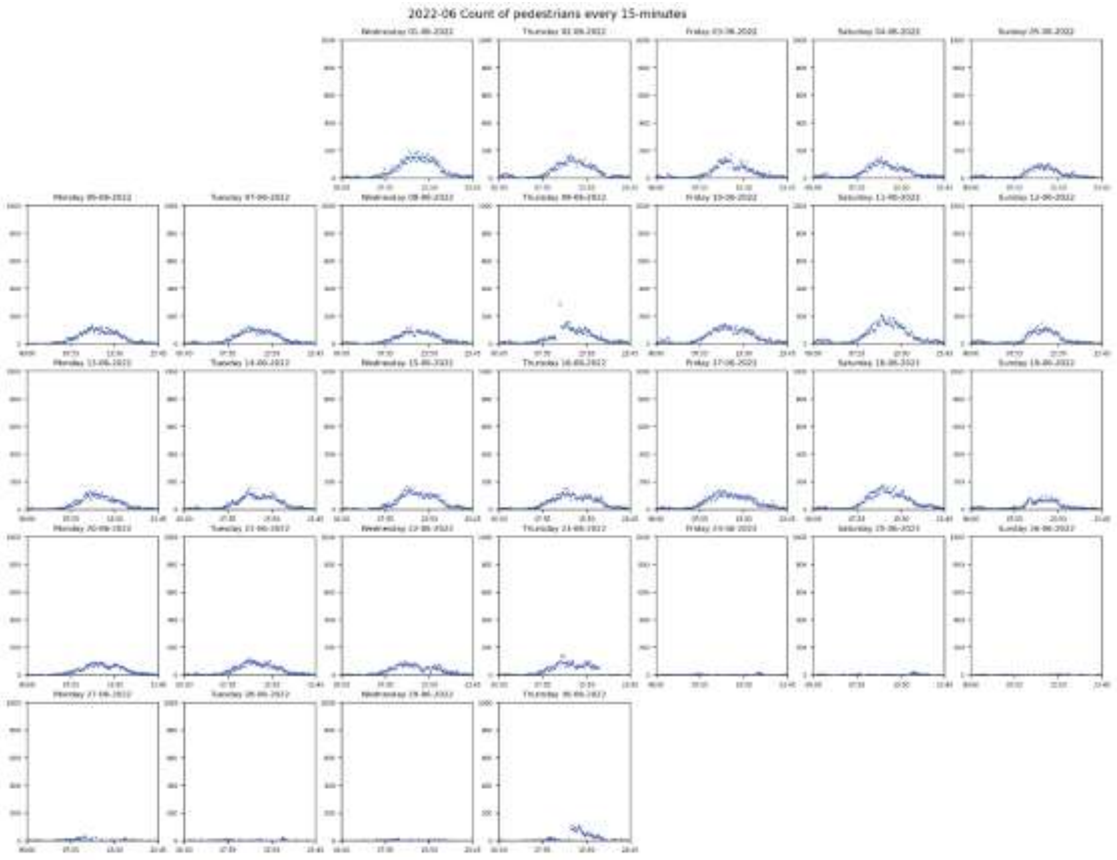

2022-03 Count of pedestrians every 15-minutes

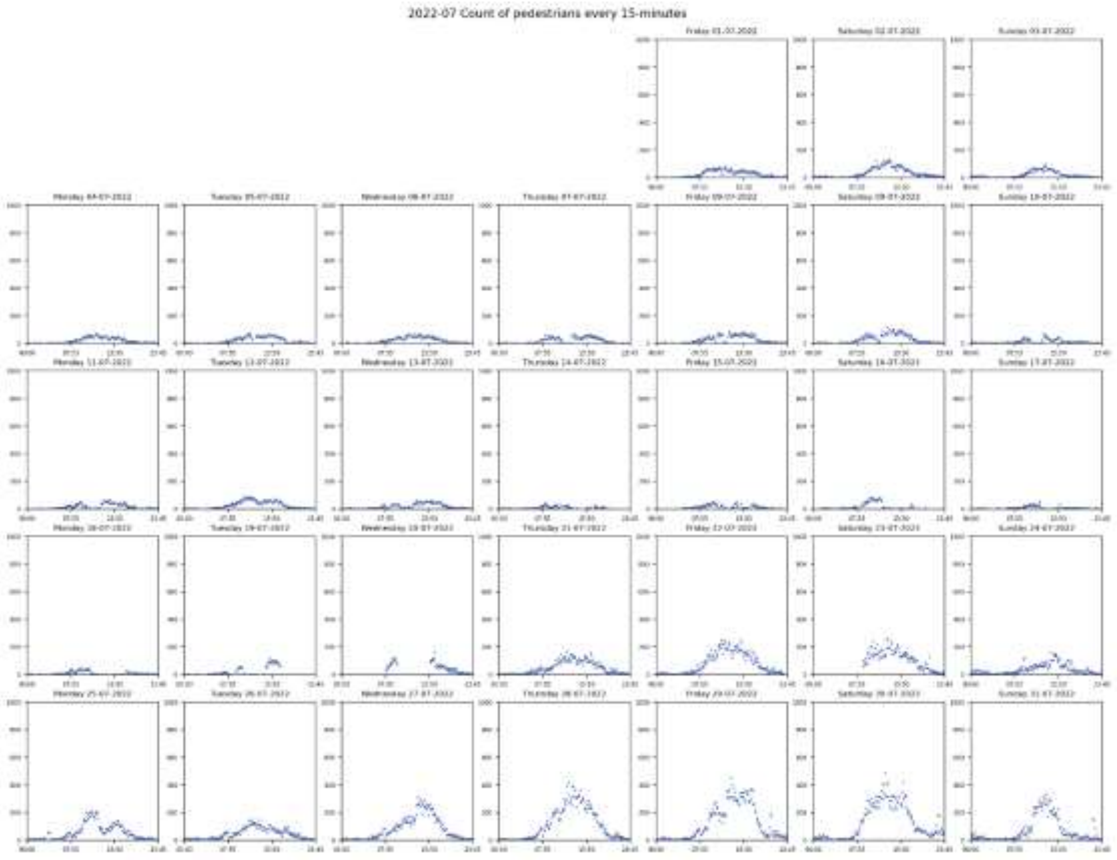
2022-04 Count of pedestrians every 15-minutes

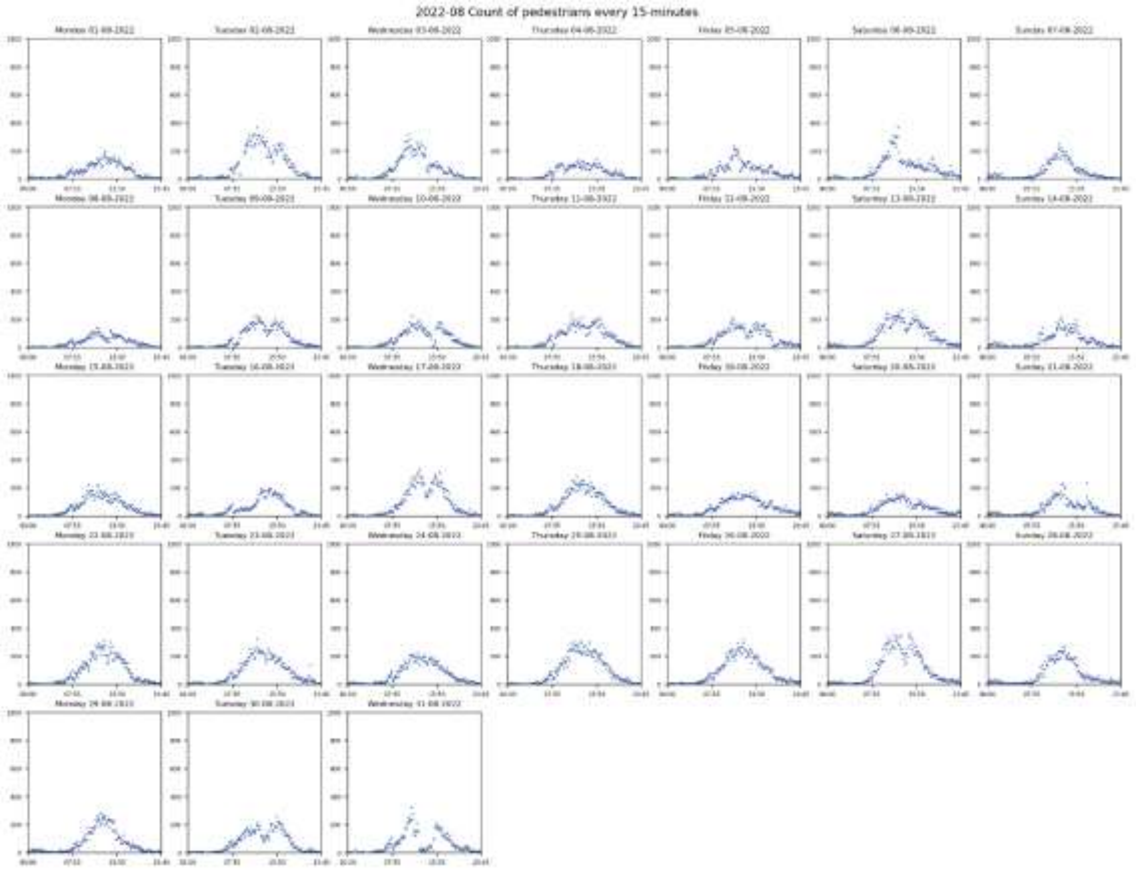


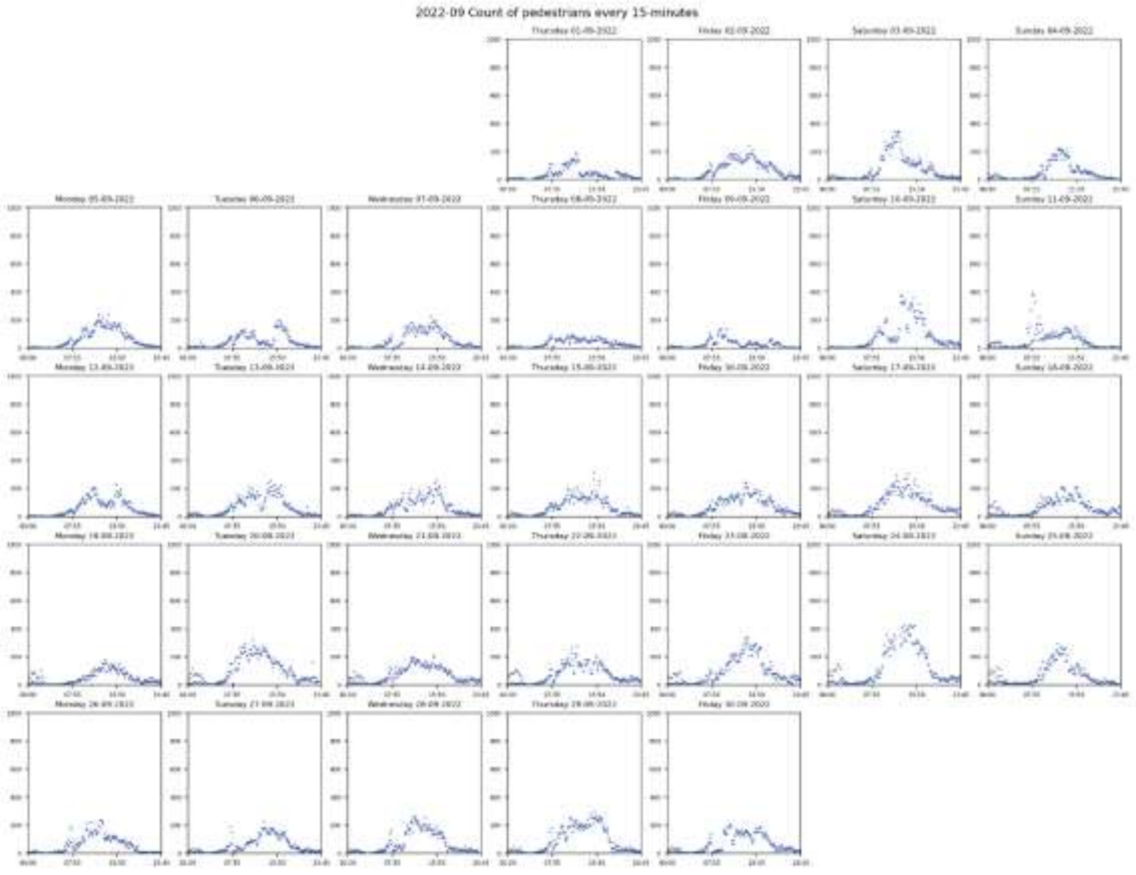
2022-05 Count of pedestrians every 15-minutes

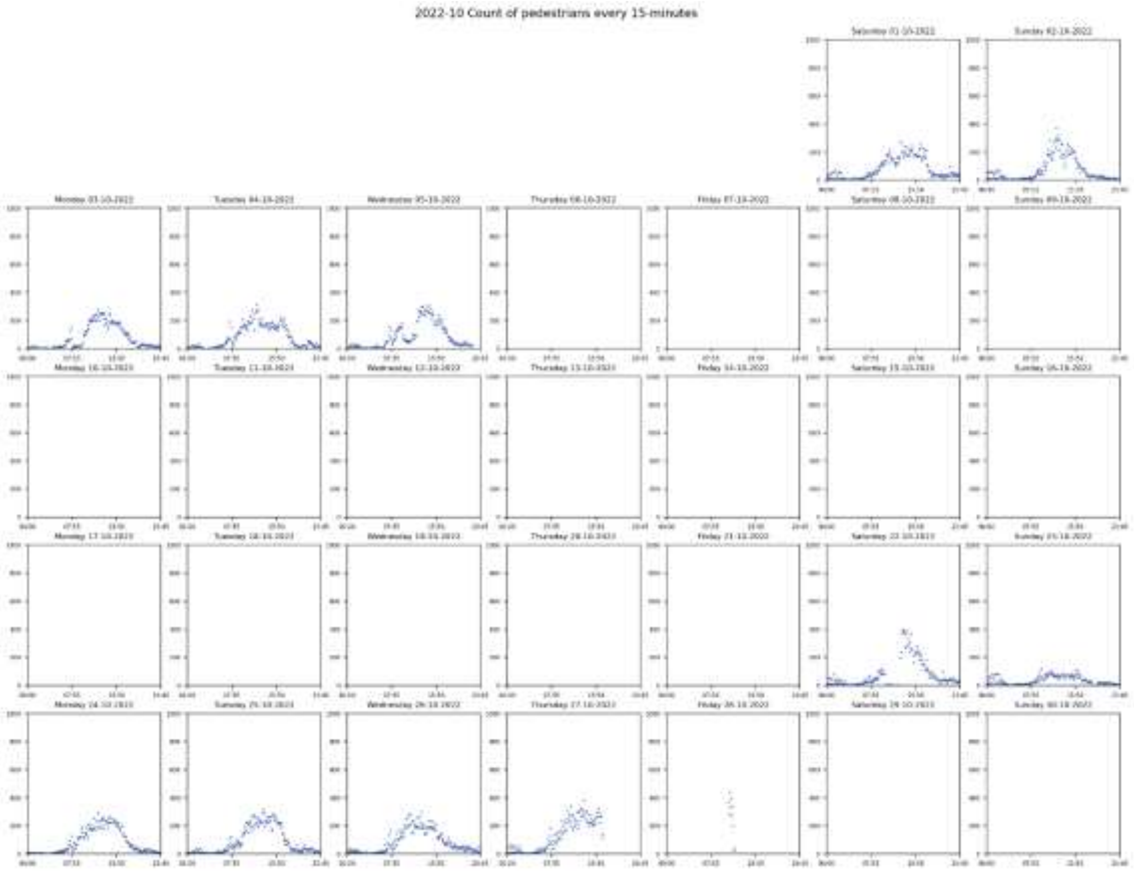


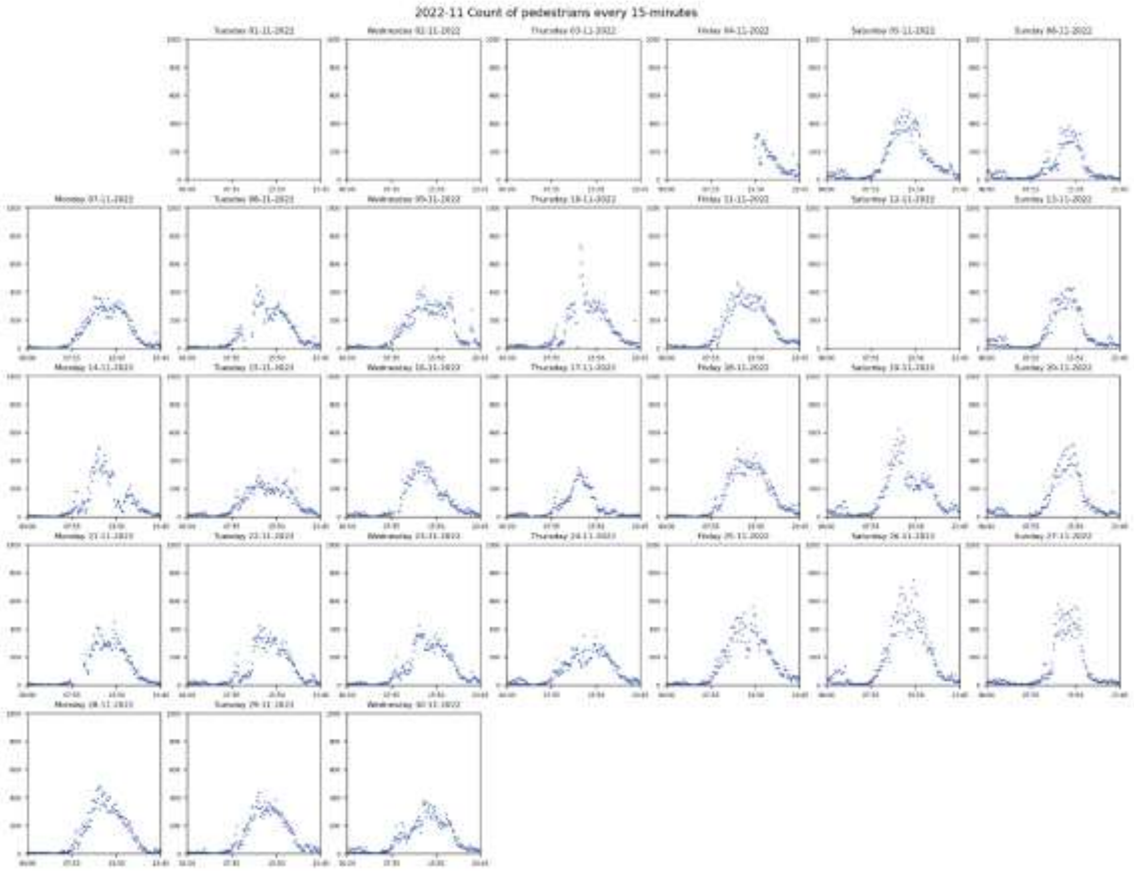

2022-06 Count of pedestrians every 15-minutes

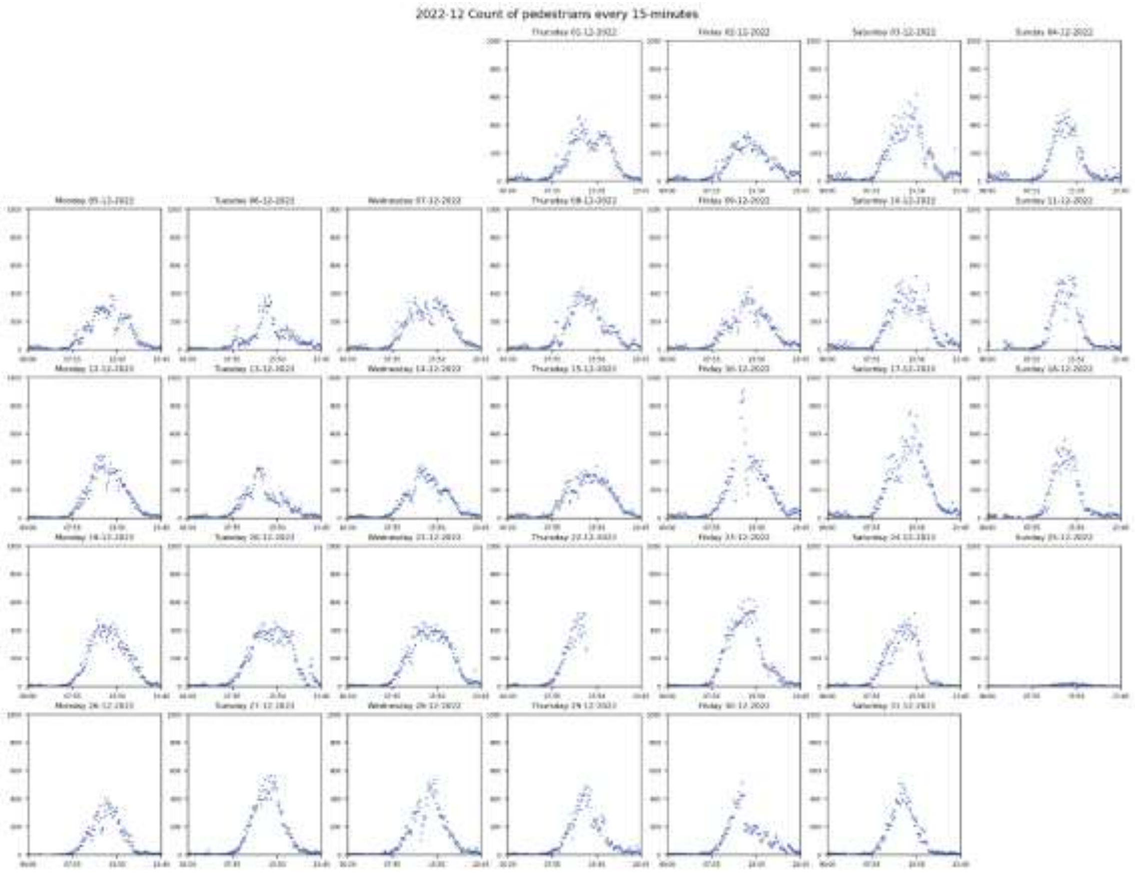

2022-07 Count of pedestrians every 15-minutes

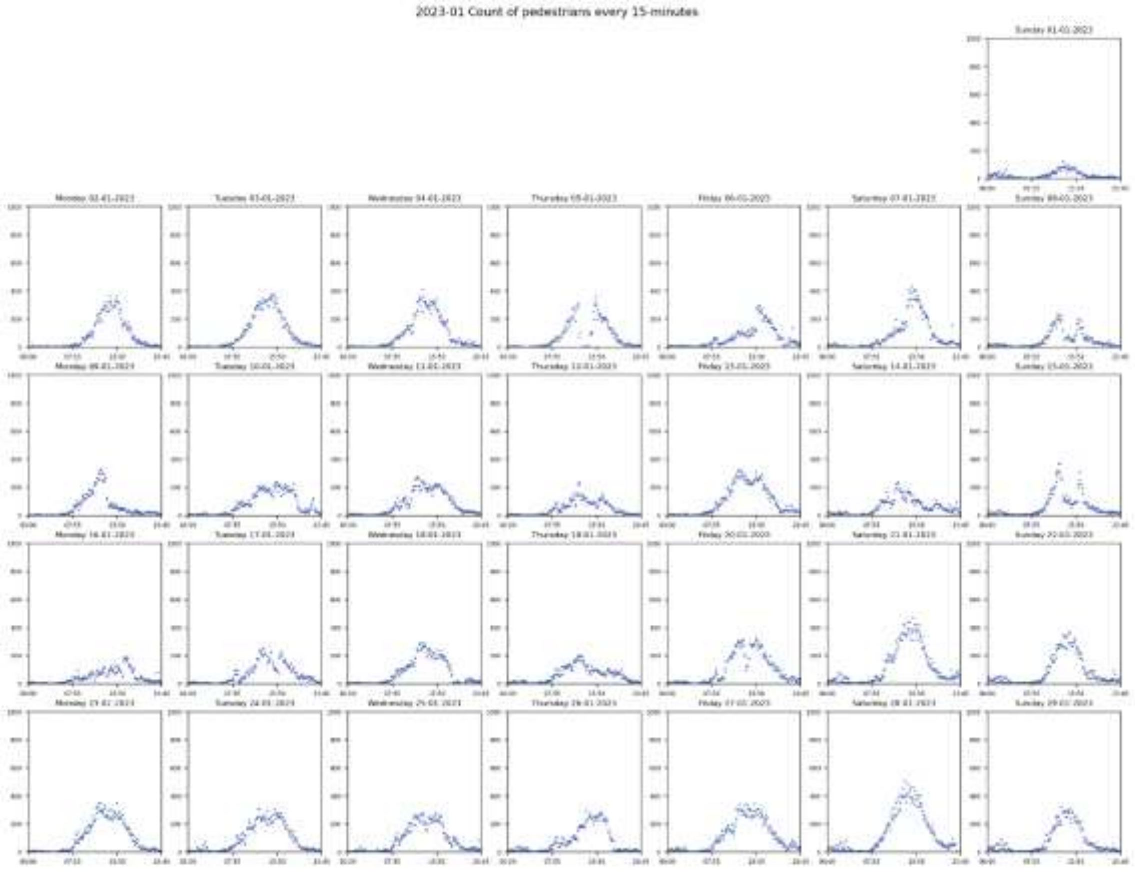

2022-08 Count of pedestrians every 15-minutes

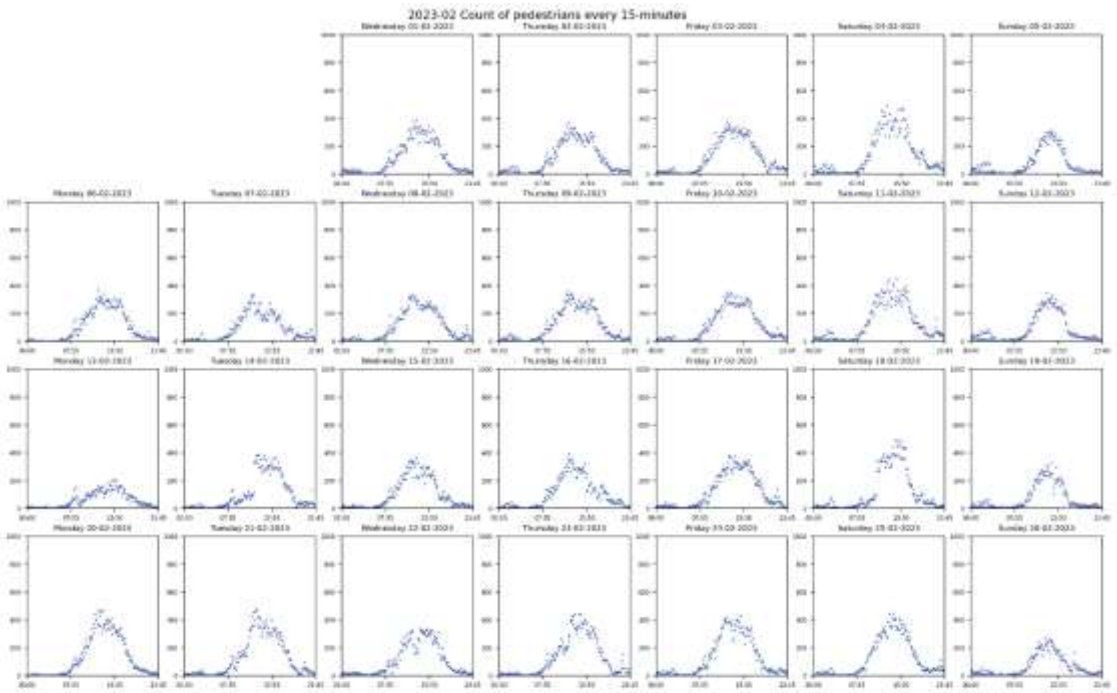

2022-09 Count of pedestrians every 15-minutes

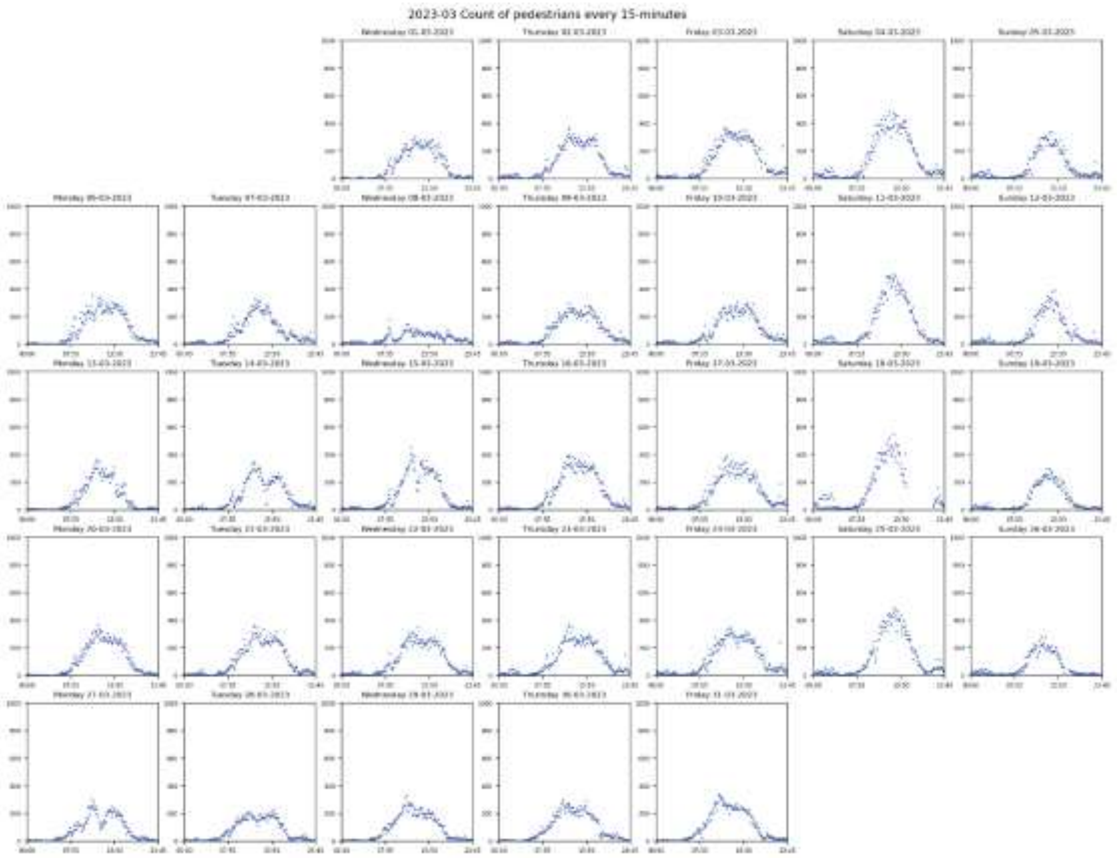

2022-10 Count of pedestrians every 15-minutes


2022-11 Count of pedestrians every 15-minutes


2022-12 Count of pedestrians every 15-minutes


2023-01 Count of pedestrians every 15-minutes


2023-02 Count of pedestrians every 15-minutes


2023-03 Count of pedestrians every 15-minutes

9.2 Results

9.2.1 Univariate Anomaly Detection

Model	Model Number	Completeness	Sequence	Horizon	Window Size	MAE	Anomaly Percentage	Anomaly Threshold
Linear	1	1	216	3	1	0.26	0.234	1.9
LSTM	2	1	216	3	3	0.27	0.161	2.1
LSTM	3	1	216	3	6	0.33	0.044	2.9
LSTM	4	1	216	3	12	0.36	0.644	3.8
LSTM	5	1	216	3	24	0.39	0.029	3.2
LSTM	6	1	216	3	48	0.28	0.103	2.3
LSTM	7	1	216	3	96	0.30	0.044	2.6
Linear	8	1	216	6	1	0.43	0.000	2.9
LSTM	9	1	216	6	3	0.46	0.000	3.6
LSTM	10	1	216	6	6	0.48	0.000	3.6
LSTM	11	1	216	6	12	0.53	0.015	4.2
LSTM	12	1	216	6	24	0.77	0.000	4.8
LSTM	13	1	216	6	48	0.55	0.000	4
LSTM	14	1	216	6	96	0.58	0.000	4.9
Linear	15	1	216	12	1	0.65	0.000	4.1
LSTM	16	1	216	12	3	0.66	0.000	4.4
LSTM	17	1	216	12	6	0.69	0.000	4.6
LSTM	18	1	216	12	12	0.69	0.000	4.4
LSTM	19	1	216	12	24	0.67	0.000	4.5
LSTM	20	1	216	12	48	0.65	0.000	4.2
LSTM	21	1	216	12	96	0.71	0.000	4.2
Linear	22	1	216	24	1	0.77	0.000	4.8
LSTM	23	1	216	24	3	0.78	0.000	4.8
LSTM	24	1	216	24	6	1.22	0.000	12.8
LSTM	25	1	216	24	12	0.66	0.000	4.5
LSTM	26	1	216	24	24	0.79	0.015	4.9
LSTM	27	1	216	24	48	0.58	0.000	4
LSTM	28	1	216	24	96	0.82	0.000	4.9
Linear	29	1	432	3	1	0.26	0.234	1.9
LSTM	30	1	432	3	3	0.27	0.073	2.2
LSTM	31	1	432	3	6	0.31	0.015	2.5
LSTM	32	1	432	3	12	0.35	0.117	3.2
LSTM	33	1	432	3	24	0.35	0.000	2.8
LSTM	34	1	432	3	48	0.27	0.044	2.2
LSTM	35	1	432	3	96	0.68	0.000	5
Linear	36	1	432	6	1	0.43	0.000	2.9
LSTM	37	1	432	6	3	0.49	0.000	3.9
LSTM	38	1	432	6	6	0.49	0.000	3.5
LSTM	39	1	432	6	12	0.50	0.059	3.9

LSTM	40	1	432	6	24	0.53	0.044	4.3
LSTM	41	1	432	6	48	0.72	0.015	5.1
LSTM	42	1	432	6	96	0.52	0.680	5.5
Linear	43	1	432	12	1	0.65	0.000	4.1
LSTM	44	1	432	12	3	0.68	0.000	4.7
LSTM	45	1	432	12	6	0.67	0.000	4.6
LSTM	46	1	432	12	12	0.67	0.000	4.7
LSTM	47	1	432	12	24	0.60	0.000	4.2
LSTM	48	1	432	12	48	0.65	0.000	4.7
LSTM	49	1	432	12	96	0.73	0.178	4.5
Linear	50	1	432	24	1	0.77	0.000	4.8
LSTM	51	1	432	24	3	0.78	0.000	4.8
LSTM	52	1	432	24	6	0.73	0.000	4.8
LSTM	53	1	432	24	12	1.03	0.059	6.8
LSTM	54	1	432	24	24	0.83	0.000	5
LSTM	55	1	432	24	48	0.76	0.000	5.1
LSTM	56	1	432	24	96	0.80	0.000	4.6
Linear	57	1	648	3	1	0.26	0.234	1.9
LSTM	58	1	648	3	3	0.30	0.015	2.4
LSTM	59	1	648	3	6	0.25	0.234	2.1
LSTM	60	1	648	3	12	0.38	0.000	3.4
LSTM	61	1	648	3	24	0.33	0.000	2.8
LSTM	62	1	648	3	48	0.33	0.059	2.8
LSTM	63	1	648	3	96	1.58	0.000	9.6
Linear	64	1	648	6	1	0.43	0.000	2.9
LSTM	65	1	648	6	3	0.50	0.000	4
LSTM	66	1	648	6	6	0.51	0.000	4.1
LSTM	67	1	648	6	12	0.52	0.000	3.8
LSTM	68	1	648	6	24	0.48	0.000	3.5
LSTM	69	1	648	6	48	0.46	0.000	3.7
LSTM	70	1	648	6	96	0.48	0.000	4
Linear	71	1	648	12	1	0.65	0.000	4.1
LSTM	72	1	648	12	3	0.69	0.000	4.7
LSTM	73	1	648	12	6	0.70	0.161	4.9
LSTM	74	1	648	12	12	0.70	0.000	4.5
LSTM	75	1	648	12	24	0.68	0.000	4.2
LSTM	76	1	648	12	48	0.64	0.000	4.5
LSTM	77	1	648	12	96	0.76	0.000	5
Linear	78	1	648	24	1	0.77	0.000	4.8
LSTM	79	1	648	24	3	0.78	0.000	4.8
LSTM	80	1	648	24	6	0.77	0.000	4.8
LSTM	81	1	648	24	12	0.78	0.000	4.8
LSTM	82	1	648	24	24	0.60	0.000	4.6
LSTM	83	1	648	24	48	0.62	0.000	4.3
LSTM	84	1	648	24	96	0.75	0.000	4.7

Linear	85	1	864	3	1	0.26	0.234	1.9
LSTM	86	1	864	3	3	0.27	0.073	2.2
LSTM	87	1	864	3	6	0.26	0.234	2.1
LSTM	88	1	864	3	12	0.44	0.000	4.2
LSTM	89	1	864	3	24	0.76	0.000	6.9
LSTM	90	1	864	3	48	0.37	0.250	3.6
LSTM	91	1	864	3	96	0.30	0.015	2.5
Linear	92	1	864	6	1	0.43	0.000	2.9
LSTM	93	1	864	6	3	0.50	0.000	3.9
LSTM	94	1	864	6	6	0.51	0.000	3.9
LSTM	95	1	864	6	12	0.46	0.000	3.3
LSTM	96	1	864	6	24	0.49	0.000	4.1
LSTM	97	1	864	6	48	0.58	0.000	3.6
LSTM	98	1	864	6	96	0.77	0.015	4.7
Linear	99	1	864	12	1	0.65	0.000	4.1
LSTM	100	1	864	12	3	0.69	0.000	4.6
LSTM	101	1	864	12	6	0.73	0.176	4.9
LSTM	102	1	864	12	12	0.65	0.000	4.5
LSTM	103	1	864	12	24	0.71	0.000	4.5
LSTM	104	1	864	12	48	0.66	0.000	4.6
LSTM	105	1	864	12	96	0.69	0.000	4.4
Linear	106	1	864	24	1	0.77	0.000	4.8
LSTM	107	1	864	24	3	0.78	0.000	4.8
LSTM	108	1	864	24	6	0.74	0.000	4.8
LSTM	109	1	864	24	12	0.84	0.000	5.1
LSTM	110	1	864	24	24	0.86	0.000	6
LSTM	111	1	864	24	48	0.82	0.000	5.1
LSTM	112	1	864	24	96	1.18	0.000	8
Linear	113	0.98	694	3	1	0.24	0.205	1.9
LSTM	114	0.98	694	3	3	0.22	0.205	1.9
LSTM	115	0.98	694	3	6	0.23	0.219	1.9
LSTM	116	0.98	694	3	12	0.21	0.293	1.8
LSTM	117	0.98	694	3	24	0.22	0.308	1.7
LSTM	118	0.98	694	3	48	0.22	0.294	1.7
LSTM	119	0.98	694	3	96	0.83	0.000	4
Linear	120	0.98	694	6	1	0.39	0.029	2.7
LSTM	121	0.98	694	6	3	0.35	0.088	2.6
LSTM	122	0.98	694	6	6	0.33	0.088	2.7
LSTM	123	0.98	694	6	12	0.28	0.249	2.3
LSTM	124	0.98	694	6	24	0.36	0.103	2.3
LSTM	125	0.98	694	6	48	0.29	0.147	2.3
LSTM	126	0.98	694	6	96	0.30	0.104	2.5
Linear	127	0.98	694	12	1	0.67	0.000	3.6
LSTM	128	0.98	694	12	3	0.56	0.000	3.6
LSTM	129	0.98	694	12	6	0.47	0.000	3.5

LSTM	130	0.98	694	12	12	0.39	0.000	2.9
LSTM	131	0.98	694	12	24	0.35	0.000	2.7
LSTM	132	0.98	694	12	48	0.38	0.000	3
LSTM	133	0.98	694	12	96	0.44	0.000	3
Linear	134	0.98	694	24	1	0.86	0.000	3.9
LSTM	135	0.98	694	24	3	0.73	0.000	4.2
LSTM	136	0.98	694	24	6	0.79	0.000	4
LSTM	137	0.98	694	24	12	0.54	0.000	3.5
LSTM	138	0.98	694	24	24	0.49	0.000	3.5
LSTM	139	0.98	694	24	48	0.60	0.000	4.1
LSTM	140	0.98	694	24	96	0.83	0.000	3.9
Linear	141	0.98	1388	3	1	0.24	0.205	1.9
LSTM	142	0.98	1388	3	3	0.22	0.190	1.9
LSTM	143	0.98	1388	3	6	0.22	0.219	1.9
LSTM	144	0.98	1388	3	12	0.20	0.351	1.7
LSTM	145	0.98	1388	3	24	0.24	0.161	1.9
LSTM	146	0.98	1388	3	48	0.27	0.191	2
LSTM	147	0.98	1388	3	96	0.23	0.177	1.9
Linear	148	0.98	1388	6	1	0.39	0.029	2.7
LSTM	149	0.98	1388	6	3	0.35	0.059	2.6
LSTM	150	0.98	1388	6	6	0.33	0.132	2.6
LSTM	151	0.98	1388	6	12	0.29	0.146	2.4
LSTM	152	0.98	1388	6	24	0.38	0.073	2.5
LSTM	153	0.98	1388	6	48	0.28	0.103	2.4
LSTM	154	0.98	1388	6	96	0.28	0.133	2.6
Linear	155	0.98	1388	12	1	0.67	0.000	3.6
LSTM	156	0.98	1388	12	3	0.59	0.000	3.5
LSTM	157	0.98	1388	12	6	0.44	0.000	3.4
LSTM	158	0.98	1388	12	12	0.38	0.015	2.9
LSTM	159	0.98	1388	12	24	0.36	0.015	2.8
LSTM	160	0.98	1388	12	48	0.63	0.000	3.6
LSTM	161	0.98	1388	12	96	0.51	0.000	3.2
Linear	162	0.98	1388	24	1	0.86	0.000	3.9
LSTM	163	0.98	1388	24	3	0.72	0.015	4.3
LSTM	164	0.98	1388	24	6	0.59	0.015	3.9
LSTM	165	0.98	1388	24	12	0.53	0.000	3.5
LSTM	166	0.98	1388	24	24	0.49	0.000	3.4
LSTM	167	0.98	1388	24	48	0.55	0.000	4.1
LSTM	168	0.98	1388	24	96	0.56	0.000	4.5
Linear	169	0.98	2082	3	1	0.24	0.205	1.9
LSTM	170	0.98	2082	3	3	0.23	0.190	1.9
LSTM	171	0.98	2082	3	6	0.23	0.263	1.9
LSTM	172	0.98	2082	3	12	0.21	0.263	1.8
LSTM	173	0.98	2082	3	24	0.20	0.264	1.7
LSTM	174	0.98	2082	3	48	0.22	0.147	1.8

LSTM	175	0.98	2082	3	96	0.20	0.207	1.8
Linear	176	0.98	2082	6	1	0.39	0.029	2.7
LSTM	177	0.98	2082	6	3	0.36	0.073	2.7
LSTM	178	0.98	2082	6	6	0.31	0.088	2.6
LSTM	179	0.98	2082	6	12	0.27	0.264	2.3
LSTM	180	0.98	2082	6	24	0.26	0.191	2.1
LSTM	181	0.98	2082	6	48	0.32	0.162	2.6
LSTM	182	0.98	2082	6	96	0.37	0.104	2.8
Linear	183	0.98	2082	12	1	0.67	0.000	3.6
LSTM	184	0.98	2082	12	3	0.55	0.000	3.5
LSTM	185	0.98	2082	12	6	0.47	0.000	3.4
LSTM	186	0.98	2082	12	12	0.40	0.059	3.1
LSTM	187	0.98	2082	12	24	0.42	0.000	2.7
LSTM	188	0.98	2082	12	48	0.39	0.000	3.3
LSTM	189	0.98	2082	12	96	0.40	0.000	2.7
Linear	190	0.98	2082	24	1	0.86	0.000	3.9
LSTM	191	0.98	2082	24	3	0.71	0.000	4.4
LSTM	192	0.98	2082	24	6	0.75	0.000	4
LSTM	193	0.98	2082	24	12	0.51	0.000	3.6
LSTM	194	0.98	2082	24	24	0.48	0.000	3.5
LSTM	195	0.98	2082	24	48	0.52	0.000	4.1
LSTM	196	0.98	2082	24	96	0.48	0.000	3.5
Linear	197	0.98	2776	3	1	0.24	0.205	1.9
LSTM	198	0.98	2776	3	3	0.23	0.175	1.9
LSTM	199	0.98	2776	3	6	0.22	0.263	1.9
LSTM	200	0.98	2776	3	12	0.21	0.293	1.8
LSTM	201	0.98	2776	3	24	0.23	0.293	1.8
LSTM	202	0.98	2776	3	48	0.24	0.176	1.8
LSTM	203	0.98	2776	3	96	0.22	0.103	1.9
Linear	204	0.98	2776	6	1	0.39	0.029	2.7
LSTM	205	0.98	2776	6	3	0.43	0.015	2.8
LSTM	206	0.98	2776	6	6	0.33	0.117	2.6
LSTM	207	0.98	2776	6	12	0.32	0.132	2.4
LSTM	208	0.98	2776	6	24	0.33	0.088	2.3
LSTM	209	0.98	2776	6	48	0.27	0.118	2.4
LSTM	210	0.98	2776	6	96	0.38	0.044	2.5
Linear	211	0.98	2776	12	1	0.67	0.000	3.6
LSTM	212	0.98	2776	12	3	0.56	0.029	3.6
LSTM	213	0.98	2776	12	6	0.46	0.000	3.5
LSTM	214	0.98	2776	12	12	0.40	0.000	3.1
LSTM	215	0.98	2776	12	24	0.37	0.029	2.8
LSTM	216	0.98	2776	12	48	0.42	0.000	3.3
LSTM	217	0.98	2776	12	96	0.43	0.000	3.5
Linear	218	0.98	2776	24	1	0.86	0.000	3.9
LSTM	219	0.98	2776	24	3	0.77	0.000	4.3

LSTM	220	0.98	2776	24	6	0.75	0.000	4
LSTM	221	0.98	2776	24	12	0.52	0.000	3.8
LSTM	222	0.98	2776	24	24	0.46	0.000	3.5
LSTM	223	0.98	2776	24	48	0.52	0.000	4.2
LSTM	224	0.98	2776	24	96	0.54	0.000	4.3
Linear	225	0.98	3470	3	1	0.24	0.205	1.9
LSTM	226	0.98	3470	3	3	0.22	0.205	1.9
LSTM	227	0.98	3470	3	6	0.22	0.322	1.9
LSTM	228	0.98	3470	3	12	0.21	0.249	1.9
LSTM	229	0.98	3470	3	24	0.20	0.249	1.7
LSTM	230	0.98	3470	3	48	0.22	0.294	1.7
LSTM	231	0.98	3470	3	96	0.19	0.310	1.7
Linear	232	0.98	3470	6	1	0.39	0.029	2.7
LSTM	233	0.98	3470	6	3	0.35	0.044	2.6
LSTM	234	0.98	3470	6	6	0.32	0.190	2.5
LSTM	235	0.98	3470	6	12	0.28	0.278	2.4
LSTM	236	0.98	3470	6	24	0.39	0.015	2.8
LSTM	237	0.98	3470	6	48	0.30	0.118	2.3
LSTM	238	0.98	3470	6	96	0.85	0.000	4
Linear	239	0.98	3470	12	1	0.67	0.000	3.6
LSTM	240	0.98	3470	12	3	0.55	0.000	3.6
LSTM	241	0.98	3470	12	6	0.45	0.000	3.5
LSTM	242	0.98	3470	12	12	0.39	0.044	3.1
LSTM	243	0.98	3470	12	24	0.38	0.000	2.5
LSTM	244	0.98	3470	12	48	0.35	0.000	2.8
LSTM	245	0.98	3470	12	96	0.38	0.000	3.1
Linear	246	0.98	3470	24	1	0.86	0.000	3.9
LSTM	247	0.98	3470	24	3	0.75	0.000	4.3
LSTM	248	0.98	3470	24	6	0.61	0.015	3.9
LSTM	249	0.98	3470	24	12	0.67	0.763	5.1
LSTM	250	0.98	3470	24	24	0.42	0.000	3.4
LSTM	251	0.98	3470	24	48	0.43	0.000	3.5
LSTM	252	0.98	3470	24	96	0.58	0.000	4.4
Linear	253	0.96	1288	3	1	0.25	0.234	1.9
LSTM	254	0.96	1288	3	3	0.25	0.146	2
LSTM	255	0.96	1288	3	6	0.24	0.161	2
LSTM	256	0.96	1288	3	12	0.24	0.234	2
LSTM	257	0.96	1288	3	24	0.21	0.220	1.8
LSTM	258	0.96	1288	3	48	0.21	0.162	1.9
LSTM	259	0.96	1288	3	96	0.26	0.163	2
Linear	260	0.96	1288	6	1	0.40	0.015	2.8
LSTM	261	0.96	1288	6	3	0.39	0.029	2.8
LSTM	262	0.96	1288	6	6	0.37	0.015	2.9
LSTM	263	0.96	1288	6	12	0.34	0.029	2.7
LSTM	264	0.96	1288	6	24	0.32	0.029	2.6

LSTM	265	0.96	1288	6	48	0.29	0.044	2.6
LSTM	266	0.96	1288	6	96	0.68	0.000	4.8
Linear	267	0.96	1288	12	1	0.63	0.000	3.9
LSTM	268	0.96	1288	12	3	0.61	0.000	4
LSTM	269	0.96	1288	12	6	0.57	0.000	3.8
LSTM	270	0.96	1288	12	12	0.51	0.000	3.5
LSTM	271	0.96	1288	12	24	0.43	0.000	3.4
LSTM	272	0.96	1288	12	48	0.44	0.000	3.5
LSTM	273	0.96	1288	12	96	0.43	0.000	3.4
Linear	274	0.96	1288	24	1	0.77	0.000	4.8
LSTM	275	0.96	1288	24	3	0.72	0.000	4.8
LSTM	276	0.96	1288	24	6	0.64	0.000	4.4
LSTM	277	0.96	1288	24	12	0.55	0.000	4
LSTM	278	0.96	1288	24	24	0.57	0.000	4.4
LSTM	279	0.96	1288	24	48	0.58	0.000	4.4
LSTM	280	0.96	1288	24	96	0.76	0.000	5.3
Linear	281	0.96	2576	3	1	0.25	0.234	1.9
LSTM	282	0.96	2576	3	3	0.25	0.146	2
LSTM	283	0.96	2576	3	6	0.23	0.219	1.9
LSTM	284	0.96	2576	3	12	0.24	0.146	2
LSTM	285	0.96	2576	3	24	0.23	0.220	1.9
LSTM	286	0.96	2576	3	48	0.23	0.338	2.2
LSTM	287	0.96	2576	3	96	0.33	0.843	4.4
Linear	288	0.96	2576	6	1	0.40	0.015	2.8
LSTM	289	0.96	2576	6	3	0.41	0.015	2.9
LSTM	290	0.96	2576	6	6	0.38	0.073	4.5
LSTM	291	0.96	2576	6	12	0.37	0.029	2.8
LSTM	292	0.96	2576	6	24	0.31	0.059	2.6
LSTM	293	0.96	2576	6	48	0.45	0.000	3.4
LSTM	294	0.96	2576	6	96	0.40	0.015	2.9
Linear	295	0.96	2576	12	1	0.63	0.000	3.9
LSTM	296	0.96	2576	12	3	0.62	0.000	4
LSTM	297	0.96	2576	12	6	0.54	0.000	3.8
LSTM	298	0.96	2576	12	12	0.46	0.000	3.4
LSTM	299	0.96	2576	12	24	0.43	0.000	3.2
LSTM	300	0.96	2576	12	48	0.51	0.000	3.9
LSTM	301	0.96	2576	12	96	0.43	0.000	3.5
Linear	302	0.96	2576	24	1	0.77	0.000	4.8
LSTM	303	0.96	2576	24	3	0.76	0.000	4.8
LSTM	304	0.96	2576	24	6	0.70	0.000	4.6
LSTM	305	0.96	2576	24	12	0.52	0.000	4.1
LSTM	306	0.96	2576	24	24	0.48	0.000	4.1
LSTM	307	0.96	2576	24	48	0.50	0.000	4
LSTM	308	0.96	2576	24	96	0.74	0.000	4.9
Linear	309	0.96	3864	3	1	0.25	0.234	1.9

LSTM	310	0.96	3864	3	3	0.26	0.263	2.1
LSTM	311	0.96	3864	3	6	0.25	0.132	2
LSTM	312	0.96	3864	3	12	0.24	0.132	2
LSTM	313	0.96	3864	3	24	0.23	0.147	2.1
LSTM	314	0.96	3864	3	48	0.23	0.191	2.1
LSTM	315	0.96	3864	3	96	0.24	0.177	2
Linear	316	0.96	3864	6	1	0.40	0.015	2.8
LSTM	317	0.96	3864	6	3	0.38	0.029	2.8
LSTM	318	0.96	3864	6	6	0.36	0.044	2.9
LSTM	319	0.96	3864	6	12	0.37	0.059	2.9
LSTM	320	0.96	3864	6	24	0.38	0.249	3
LSTM	321	0.96	3864	6	48	0.33	0.000	3.0
LSTM	322	0.96	3864	6	96	0.35	0.015	2.9
Linear	323	0.96	3864	12	1	0.63	0.000	3.9
LSTM	324	0.96	3864	12	3	0.60	0.000	4
LSTM	325	0.96	3864	12	6	0.55	0.000	3.7
LSTM	326	0.96	3864	12	12	0.45	0.000	3.4
LSTM	327	0.96	3864	12	24	0.63	0.000	4.1
LSTM	328	0.96	3864	12	48	0.41	0.000	3.4
LSTM	329	0.96	3864	12	96	0.50	0.000	3.9
Linear	330	0.96	3864	24	1	0.77	0.000	4.8
LSTM	331	0.96	3864	24	3	0.70	0.000	4.7
LSTM	332	0.96	3864	24	6	0.61	0.000	4.4
LSTM	333	0.96	3864	24	12	0.58	0.000	4.1
LSTM	334	0.96	3864	24	24	0.76	0.000	4.9
LSTM	335	0.96	3864	24	48	0.52	0.000	4
LSTM	336	0.96	3864	24	96	0.60	0.000	4.9
Linear	337	0.96	5152	3	1	0.25	0.234	1.9
LSTM	338	0.96	5152	3	3	0.25	0.234	2.1
LSTM	339	0.96	5152	3	6	0.24	0.190	2
LSTM	340	0.96	5152	3	12	0.24	0.249	2.1
LSTM	341	0.96	5152	3	24	0.27	0.073	2.5
LSTM	342	0.96	5152	3	48	0.26	0.147	2
LSTM	343	0.96	5152	3	96	0.33	0.118	2.2
Linear	344	0.96	5152	6	1	0.40	0.015	2.8
LSTM	345	0.96	5152	6	3	0.39	0.015	2.9
LSTM	346	0.96	5152	6	6	0.36	0.044	2.7
LSTM	347	0.96	5152	6	12	0.36	0.044	2.8
LSTM	348	0.96	5152	6	24	0.31	0.029	2.6
LSTM	349	0.96	5152	6	48	0.36	0.632	3.6
LSTM	350	0.96	5152	6	96	0.48	0.015	3.9
Linear	351	0.96	5152	12	1	0.63	0.000	3.9
LSTM	352	0.96	5152	12	3	0.60	0.000	3.9
LSTM	353	0.96	5152	12	6	0.56	0.000	3.8
LSTM	354	0.96	5152	12	12	0.48	0.000	3.4

LSTM	355	0.96	5152	12	24	0.42	0.000	3.3
LSTM	356	0.96	5152	12	48	0.47	0.015	3.6
LSTM	357	0.96	5152	12	96	0.51	0.000	4.1
Linear	358	0.96	5152	24	1	0.77	0.000	4.8
LSTM	359	0.96	5152	24	3	0.76	0.000	4.8
LSTM	360	0.96	5152	24	6	0.62	0.000	4.3
LSTM	361	0.96	5152	24	12	0.59	0.000	4.1
LSTM	362	0.96	5152	24	24	0.47	0.000	4.1
LSTM	363	0.96	5152	24	48	0.65	0.000	4.5
LSTM	364	0.96	5152	24	96	0.48	0.000	3.9
Linear	365	0.96	6440	3	1	0.25	0.234	1.9
LSTM	366	0.96	6440	3	3	0.25	0.175	2
LSTM	367	0.96	6440	3	6	0.25	0.176	2
LSTM	368	0.96	6440	3	12	0.24	0.146	2
LSTM	369	0.96	6440	3	24	0.23	0.147	2
LSTM	370	0.96	6440	3	48	0.26	0.059	2.1
LSTM	371	0.96	6440	3	96	0.23	0.192	2
Linear	372	0.96	6440	6	1	0.40	0.015	2.8
LSTM	373	0.96	6440	6	3	0.40	0.015	2.9
LSTM	374	0.96	6440	6	6	0.38	0.029	2.8
LSTM	375	0.96	6440	6	12	0.35	0.059	2.9
LSTM	376	0.96	6440	6	24	0.41	0.000	3
LSTM	377	0.96	6440	6	48	0.32	0.088	2.7
LSTM	378	0.96	6440	6	96	0.45	0.000	4.3
Linear	379	0.96	6440	12	1	0.63	0.000	3.9
LSTM	380	0.96	6440	12	3	0.59	0.000	3.9
LSTM	381	0.96	6440	12	6	0.56	0.000	4
LSTM	382	0.96	6440	12	12	0.47	0.000	3.4
LSTM	383	0.96	6440	12	24	0.44	0.000	3.2
LSTM	384	0.96	6440	12	48	0.44	0.000	3.6
LSTM	385	0.96	6440	12	96	0.68	1.465	8.4
Linear	386	0.96	6440	24	1	0.77	0.000	4.8
LSTM	387	0.96	6440	24	3	0.68	0.000	4.7
LSTM	388	0.96	6440	24	6	0.60	0.015	4.4
LSTM	389	0.96	6440	24	12	0.54	0.000	4.2
LSTM	390	0.96	6440	24	24	0.48	0.000	4.2
LSTM	391	0.96	6440	24	48	0.48	0.000	4
LSTM	392	0.96	6440	24	96	0.50	0.000	4
Linear	393	0.94	1810	3	1	0.24	0.219	1.9
LSTM	394	0.94	1810	3	3	0.24	0.161	1.9
LSTM	395	0.94	1810	3	6	0.23	0.161	1.9
LSTM	396	0.94	1810	3	12	0.23	0.146	1.9
LSTM	397	0.94	1810	3	24	0.21	0.191	1.8
LSTM	398	0.94	1810	3	48	0.21	0.220	1.8
LSTM	399	0.94	1810	3	96	0.23	0.133	2

Linear	400	0.94	1810	6	1	0.40	0.015	2.8
LSTM	401	0.94	1810	6	3	0.36	0.044	2.7
LSTM	402	0.94	1810	6	6	0.35	0.015	2.6
LSTM	403	0.94	1810	6	12	0.31	0.015	2.6
LSTM	404	0.94	1810	6	24	0.30	0.000	2.5
LSTM	405	0.94	1810	6	48	0.28	0.000	2.5
LSTM	406	0.94	1810	6	96	0.32	0.030	2.6
Linear	407	0.94	1810	12	1	0.63	0.000	3.9
LSTM	408	0.94	1810	12	3	0.56	0.000	3.8
LSTM	409	0.94	1810	12	6	0.49	0.015	3.5
LSTM	410	0.94	1810	12	12	0.42	0.059	3.5
LSTM	411	0.94	1810	12	24	0.40	0.000	3.3
LSTM	412	0.94	1810	12	48	0.61	0.000	4
LSTM	413	0.94	1810	12	96	0.63	0.015	4
Linear	414	0.94	1810	24	1	0.77	0.000	4.8
LSTM	415	0.94	1810	24	3	0.72	0.000	4.7
LSTM	416	0.94	1810	24	6	0.59	0.000	4.2
LSTM	417	0.94	1810	24	12	0.62	0.000	4.3
LSTM	418	0.94	1810	24	24	0.57	0.073	4.5
LSTM	419	0.94	1810	24	48	0.46	0.000	3.8
LSTM	420	0.94	1810	24	96	0.47	0.000	3.9
Linear	421	0.94	3620	3	1	0.24	0.219	1.9
LSTM	422	0.94	3620	3	3	0.24	0.161	2
LSTM	423	0.94	3620	3	6	0.23	0.146	1.9
LSTM	424	0.94	3620	3	12	0.24	0.263	2.1
LSTM	425	0.94	3620	3	24	0.21	0.191	1.9
LSTM	426	0.94	3620	3	48	0.21	0.235	1.8
LSTM	427	0.94	3620	3	96	0.26	0.192	2.5
Linear	428	0.94	3620	6	1	0.40	0.015	2.8
LSTM	429	0.94	3620	6	3	0.37	0.088	2.9
LSTM	430	0.94	3620	6	6	0.34	0.044	2.6
LSTM	431	0.94	3620	6	12	0.31	0.000	2.5
LSTM	432	0.94	3620	6	24	0.30	0.044	2.3
LSTM	433	0.94	3620	6	48	0.29	0.000	2.4
LSTM	434	0.94	3620	6	96	0.40	0.104	2.7
Linear	435	0.94	3620	12	1	0.63	0.000	3.9
LSTM	436	0.94	3620	12	3	0.57	0.000	3.9
LSTM	437	0.94	3620	12	6	0.52	0.015	3.6
LSTM	438	0.94	3620	12	12	0.50	0.000	3.4
LSTM	439	0.94	3620	12	24	0.42	0.000	3.1
LSTM	440	0.94	3620	12	48	0.41	0.000	3.1
LSTM	441	0.94	3620	12	96	0.59	0.000	3.9
Linear	442	0.94	3620	24	1	0.77	0.000	4.8
LSTM	443	0.94	3620	24	3	0.74	0.000	4.7
LSTM	444	0.94	3620	24	6	0.58	0.000	4.2

LSTM	445	0.94	3620	24	12	0.49	0.000	3.9
LSTM	446	0.94	3620	24	24	0.50	0.000	4.3
LSTM	447	0.94	3620	24	48	0.43	0.000	3.6
LSTM	448	0.94	3620	24	96	0.54	0.000	4.3
Linear	449	0.94	5430	3	1	0.24	0.219	1.9
LSTM	450	0.94	5430	3	3	0.24	0.161	2
LSTM	451	0.94	5430	3	6	0.22	0.176	1.9
LSTM	452	0.94	5430	3	12	0.22	0.322	2
LSTM	453	0.94	5430	3	24	0.21	0.161	1.8
LSTM	454	0.94	5430	3	48	0.22	0.162	2
LSTM	455	0.94	5430	3	96	0.23	0.296	2
Linear	456	0.94	5430	6	1	0.40	0.015	2.8
LSTM	457	0.94	5430	6	3	0.37	0.029	2.8
LSTM	458	0.94	5430	6	6	0.34	0.015	2.6
LSTM	459	0.94	5430	6	12	0.30	0.102	2.5
LSTM	460	0.94	5430	6	24	0.28	0.088	2.3
LSTM	461	0.94	5430	6	48	0.32	0.353	2.8
LSTM	462	0.94	5430	6	96	0.35	0.030	2.7
Linear	463	0.94	5430	12	1	0.63	0.000	3.9
LSTM	464	0.94	5430	12	3	0.56	0.015	3.7
LSTM	465	0.94	5430	12	6	0.50	0.000	3.5
LSTM	466	0.94	5430	12	12	0.41	0.000	3.1
LSTM	467	0.94	5430	12	24	0.37	0.000	3.1
LSTM	468	0.94	5430	12	48	0.39	0.000	3.2
LSTM	469	0.94	5430	12	96	0.62	0.000	3.9
Linear	470	0.94	5430	24	1	0.77	0.000	4.8
LSTM	471	0.94	5430	24	3	0.73	0.000	4.7
LSTM	472	0.94	5430	24	6	0.59	0.015	4.2
LSTM	473	0.94	5430	24	12	0.51	0.000	4
LSTM	474	0.94	5430	24	24	0.47	0.000	3.9
LSTM	475	0.94	5430	24	48	0.58	0.000	4.3
LSTM	476	0.94	5430	24	96	0.46	0.000	3.9
Linear	477	0.94	7240	3	1	0.24	0.219	1.9
LSTM	478	0.94	7240	3	3	0.23	0.146	1.9
LSTM	479	0.94	7240	3	6	0.24	0.205	2.1
LSTM	480	0.94	7240	3	12	0.21	0.234	1.9
LSTM	481	0.94	7240	3	24	0.21	0.147	1.8
LSTM	482	0.94	7240	3	48	0.21	0.250	1.8
LSTM	483	0.94	7240	3	96	0.23	0.148	1.9
Linear	484	0.94	7240	6	1	0.40	0.015	2.8
LSTM	485	0.94	7240	6	3	0.37	0.044	2.7
LSTM	486	0.94	7240	6	6	0.33	0.000	2.6
LSTM	487	0.94	7240	6	12	0.34	0.029	2.6
LSTM	488	0.94	7240	6	24	0.30	0.073	2.5
LSTM	489	0.94	7240	6	48	0.30	0.044	2.6

LSTM	490	0.94	7240	6	96	0.30	0.030	2.5
Linear	491	0.94	7240	12	1	0.63	0.000	3.9
LSTM	492	0.94	7240	12	3	0.56	0.000	3.7
LSTM	493	0.94	7240	12	6	0.50	0.015	3.7
LSTM	494	0.94	7240	12	12	0.38	0.029	3.2
LSTM	495	0.94	7240	12	24	0.43	0.000	3.3
LSTM	496	0.94	7240	12	48	0.36	0.000	3.1
LSTM	497	0.94	7240	12	96	0.56	0.000	3.9
Linear	498	0.94	7240	24	1	0.77	0.000	4.8
LSTM	499	0.94	7240	24	3	0.67	0.000	4.6
LSTM	500	0.94	7240	24	6	0.55	0.000	4.1
LSTM	501	0.94	7240	24	12	0.50	0.000	4
LSTM	502	0.94	7240	24	24	0.47	0.000	4.1
LSTM	503	0.94	7240	24	48	0.46	0.000	3.8
LSTM	504	0.94	7240	24	96	0.70	0.000	4.9
Linear	505	0.94	9050	3	1	0.24	0.219	1.9
LSTM	506	0.94	9050	3	3	0.23	0.175	1.9
LSTM	507	0.94	9050	3	6	0.23	0.176	1.9
LSTM	508	0.94	9050	3	12	0.22	0.190	1.9
LSTM	509	0.94	9050	3	24	0.23	0.132	2.1
LSTM	510	0.94	9050	3	48	0.24	0.176	2
LSTM	511	0.94	9050	3	96	0.23	0.488	2.2
Linear	512	0.94	9050	6	1	0.40	0.015	2.8
LSTM	513	0.94	9050	6	3	0.36	0.044	2.7
LSTM	514	0.94	9050	6	6	0.32	0.000	2.6
LSTM	515	0.94	9050	6	12	0.32	0.088	2.6
LSTM	516	0.94	9050	6	24	0.32	0.059	2.5
LSTM	517	0.94	9050	6	48	0.38	0.029	2.8
LSTM	518	0.94	9050	6	96	0.31	0.030	2.5
Linear	519	0.94	9050	12	1	0.63	0.000	3.9
LSTM	520	0.94	9050	12	3	0.57	0.000	3.7
LSTM	521	0.94	9050	12	6	0.50	0.029	3.5
LSTM	522	0.94	9050	12	12	0.40	0.103	3.1
LSTM	523	0.94	9050	12	24	0.41	0.000	3.3
LSTM	524	0.94	9050	12	48	0.63	0.000	4.1
LSTM	525	0.94	9050	12	96	0.41	0.000	3.2
Linear	526	0.94	9050	24	1	0.77	0.000	4.8
LSTM	527	0.94	9050	24	3	0.66	0.000	4.6
LSTM	528	0.94	9050	24	6	0.59	0.000	4.1
LSTM	529	0.94	9050	24	12	0.48	0.000	3.9
LSTM	530	0.94	9050	24	24	0.78	0.000	4.7
LSTM	531	0.94	9050	24	48	0.46	0.000	4
LSTM	532	0.94	9050	24	96	0.71	0.000	4.8

9.2.2 Multivariate Anomaly Detection

Model	Model Number	Completeness	Sequence	Horizon	Window Size	MAE	Anomaly Percentage	Anomaly Threshold
Linear	1	1	216	1	1	0.53	0.000	2.6
LSTM	2	1	216	1	3	0.76	0.000	5.8
LSTM	3	1	216	1	6	1.04	0.000	8.2
LSTM	4	1	216	1	12	0.78	0.000	3.8
LSTM	5	1	216	1	24	0.59	0.000	3.5
LSTM	6	1	216	1	48	0.55	0.103	5.4
LSTM	7	1	216	1	96	0.62	0.000	4.8
Linear	8	1	216	3	1	1.20	0.000	5.4
LSTM	9	1	216	3	3	1.08	0.000	7.6
LSTM	10	1	216	3	6	0.55	0.000	5.1
LSTM	11	1	216	3	12	1.38	0.000	10.2
LSTM	12	1	216	3	24	0.53	0.337	4
LSTM	13	1	216	3	48	0.50	0.250	4
LSTM	14	1	216	3	96	0.51	0.000	4.6
Linear	15	1	216	6	1	2.48	0.000	10.9
LSTM	16	1	216	6	3	1.27	0.000	9.8
LSTM	17	1	216	6	6	0.91	0.000	6.7
LSTM	18	1	216	6	12	1.45	0.000	10.2
LSTM	19	1	216	6	24	0.86	0.103	5.8
LSTM	20	1	216	6	48	0.83	0.000	7.1
LSTM	21	1	216	6	96	1.36	0.000	10.9
Linear	22	1	216	12	1	3.01	0.000	13
LSTM	23	1	216	12	3	0.63	0.000	3.6
LSTM	24	1	216	12	6	1.99	0.000	11.7
LSTM	25	1	216	12	12	0.98	0.000	6
LSTM	26	1	216	12	24	1.16	0.000	7.2
LSTM	27	1	216	12	48	0.62	0.015	3.9
LSTM	28	1	216	12	96	0.80	0.000	6
Linear	29	1	216	24	1	1.34	0.000	6.4
LSTM	30	1	216	24	3	1.26	0.000	8.2
LSTM	31	1	216	24	6	1.06	0.000	6.1
LSTM	32	1	216	24	12	0.99	0.000	5.7
LSTM	33	1	216	24	24	0.74	0.000	4.8
LSTM	34	1	216	24	48	0.48	0.000	3.5
LSTM	35	1	216	24	96	0.47	0.000	4
Linear	36	1	432	1	1	0.43	0.000	2
LSTM	37	1	432	1	3	1.40	0.000	12.2
LSTM	38	1	432	1	6	1.84	0.000	14
LSTM	39	1	432	1	12	0.78	0.000	5.9
LSTM	40	1	432	1	24	0.79	0.000	6.9
LSTM	41	1	432	1	48	0.52	0.015	3.7

LSTM	42	1	432	1	96	0.30	0.074	2.7
Linear	43	1	432	3	1	0.91	0.000	4.2
LSTM	44	1	432	3	3	0.77	0.000	4.3
LSTM	45	1	432	3	6	0.89	0.000	8.4
LSTM	46	1	432	3	12	0.60	0.000	3.9
LSTM	47	1	432	3	24	0.63	0.000	3.8
LSTM	48	1	432	3	48	1.08	0.000	10
LSTM	49	1	432	3	96	0.47	0.015	3.4
Linear	50	1	432	6	1	2.15	0.000	9.5
LSTM	51	1	432	6	3	0.87	0.000	5.8
LSTM	52	1	432	6	6	1.54	0.000	10.2
LSTM	53	1	432	6	12	1.10	0.000	8
LSTM	54	1	432	6	24	0.82	0.015	6
LSTM	55	1	432	6	48	0.69	0.000	3.8
LSTM	56	1	432	6	96	0.69	0.000	6.4
Linear	57	1	432	12	1	2.89	0.000	12.8
LSTM	58	1	432	12	3	0.82	0.000	5.2
LSTM	59	1	432	12	6	0.74	0.000	4.7
LSTM	60	1	432	12	12	1.20	0.000	9.5
LSTM	61	1	432	12	24	1.12	0.000	7.3
LSTM	62	1	432	12	48	1.03	0.029	5.1
LSTM	63	1	432	12	96	0.53	0.000	4.1
Linear	64	1	432	24	1	1.94	0.000	8.7
LSTM	65	1	432	24	3	1.26	0.000	8.8
LSTM	66	1	432	24	6	0.93	0.073	6.4
LSTM	67	1	432	24	12	0.88	0.015	5.5
LSTM	68	1	432	24	24	0.99	0.000	5.3
LSTM	69	1	432	24	48	0.54	0.000	3.6
LSTM	70	1	432	24	96	0.65	0.000	4
Linear	71	1	648	1	1	0.57	0.000	2.6
LSTM	72	1	648	1	3	0.55	0.000	3.8
LSTM	73	1	648	1	6	2.11	0.000	14.6
LSTM	74	1	648	1	12	0.70	0.000	4.7
LSTM	75	1	648	1	24	1.33	0.000	11.6
LSTM	76	1	648	1	48	0.58	0.000	4.2
LSTM	77	1	648	1	96	0.25	0.059	1.9
Linear	78	1	648	3	1	1.16	0.000	5.4
LSTM	79	1	648	3	3	0.90	0.000	7.4
LSTM	80	1	648	3	6	1.84	0.000	12.6
LSTM	81	1	648	3	12	0.86	0.000	6.3
LSTM	82	1	648	3	24	0.87	0.000	4.7
LSTM	83	1	648	3	48	0.48	0.000	2.8
LSTM	84	1	648	3	96	0.37	0.458	3.5
Linear	85	1	648	6	1	1.89	0.000	8.4
LSTM	86	1	648	6	3	0.80	0.000	4.7

LSTM	87	1	648	6	6	1.78	0.000	10.5
LSTM	88	1	648	6	12	1.02	0.015	5.6
LSTM	89	1	648	6	24	1.04	0.000	5.9
LSTM	90	1	648	6	48	0.77	0.000	5.8
LSTM	91	1	648	6	96	0.71	0.030	5.6
Linear	92	1	648	12	1	3.18	0.000	13.7
LSTM	93	1	648	12	3	1.29	0.000	7.8
LSTM	94	1	648	12	6	0.82	0.029	4.7
LSTM	95	1	648	12	12	1.10	0.000	6.8
LSTM	96	1	648	12	24	1.36	0.117	10.3
LSTM	97	1	648	12	48	0.76	0.044	4.8
LSTM	98	1	648	12	96	0.62	0.044	5
Linear	99	1	648	24	1	1.96	0.000	8.7
LSTM	100	1	648	24	3	2.05	0.000	15.4
LSTM	101	1	648	24	6	1.26	0.000	12.4
LSTM	102	1	648	24	12	1.66	0.000	14.6
LSTM	103	1	648	24	24	1.01	0.029	5.7
LSTM	104	1	648	24	48	0.77	0.516	6.9
LSTM	105	1	648	24	96	0.96	0.015	6.2
Linear	106	1	864	1	1	0.50	0.000	2.4
LSTM	107	1	864	1	3	0.63	0.000	4.7
LSTM	108	1	864	1	6	1.12	0.000	8.7
LSTM	109	1	864	1	12	0.90	0.000	6
LSTM	110	1	864	1	24	1.02	0.000	6.7
LSTM	111	1	864	1	48	0.92	0.000	6.8
LSTM	112	1	864	1	96	0.41	0.000	3.2
Linear	113	1	864	3	1	0.97	0.000	4.2
LSTM	114	1	864	3	3	1.55	0.000	11.3
LSTM	115	1	864	3	6	1.08	0.000	6.4
LSTM	116	1	864	3	12	0.77	0.000	5.9
LSTM	117	1	864	3	24	0.52	0.000	4.1
LSTM	118	1	864	3	48	0.72	0.000	6.3
LSTM	119	1	864	3	96	0.50	0.030	3.9
Linear	120	1	864	6	1	1.83	0.000	8.1
LSTM	121	1	864	6	3	0.57	0.000	4.4
LSTM	122	1	864	6	6	1.23	0.000	7
LSTM	123	1	864	6	12	0.82	0.000	6
LSTM	124	1	864	6	24	0.95	0.000	6.9
LSTM	125	1	864	6	48	0.68	0.000	5
LSTM	126	1	864	6	96	0.49	0.000	4
Linear	127	1	864	12	1	3.11	0.000	13.5
LSTM	128	1	864	12	3	0.69	0.000	4.7
LSTM	129	1	864	12	6	1.10	0.000	9.3
LSTM	130	1	864	12	12	0.80	0.000	5.3
LSTM	131	1	864	12	24	0.90	0.147	6.3

LSTM	132	1	864	12	48	0.67	0.221	4.6
LSTM	133	1	864	12	96	0.42	0.015	3.7
Linear	134	1	864	24	1	2.03	0.000	9
LSTM	135	1	864	24	3	0.85	0.000	6
LSTM	136	1	864	24	6	0.84	0.000	5.5
LSTM	137	1	864	24	12	0.98	0.000	5.8
LSTM	138	1	864	24	24	1.30	0.000	12.5
LSTM	139	1	864	24	48	0.59	0.000	3.9
LSTM	140	1	864	24	96	0.42	0.000	3.4
Linear	141	0.98	694	1	1	1.34	0.000	4.8
LSTM	142	0.98	694	1	3	0.26	0.117	2
LSTM	143	0.98	694	1	6	0.27	0.102	1.9
LSTM	144	0.98	694	1	12	0.18	0.249	1.4
LSTM	145	0.98	694	1	24	0.22	0.147	1.6
LSTM	146	0.98	694	1	48	0.34	0.573	3
LSTM	147	0.98	694	1	96	0.24	0.089	1.9
Linear	148	0.98	694	3	1	2.20	0.000	7.8
LSTM	149	0.98	694	3	3	0.39	0.058	2.9
LSTM	150	0.98	694	3	6	0.27	0.132	2.1
LSTM	151	0.98	694	3	12	0.28	0.117	2.2
LSTM	152	0.98	694	3	24	0.30	0.088	2.2
LSTM	153	0.98	694	3	48	0.28	0.118	2.2
LSTM	154	0.98	694	3	96	0.29	0.118	2.5
Linear	155	0.98	694	6	1	2.67	0.000	9.7
LSTM	156	0.98	694	6	3	0.37	0.088	3.1
LSTM	157	0.98	694	6	6	0.34	0.117	2.7
LSTM	158	0.98	694	6	12	0.33	0.117	2.8
LSTM	159	0.98	694	6	24	0.32	0.205	2.7
LSTM	160	0.98	694	6	48	0.31	0.147	2.5
LSTM	161	0.98	694	6	96	0.30	0.163	2.6
Linear	162	0.98	694	12	1	2.97	0.000	10.7
LSTM	163	0.98	694	12	3	0.39	0.220	3
LSTM	164	0.98	694	12	6	0.39	0.015	3
LSTM	165	0.98	694	12	12	0.33	0.029	2.8
LSTM	166	0.98	694	12	24	0.38	0.044	2.9
LSTM	167	0.98	694	12	48	0.39	0.103	3
LSTM	168	0.98	694	12	96	0.36	0.237	3.1
Linear	169	0.98	694	24	1	2.69	0.000	9.8
LSTM	170	0.98	694	24	3	0.38	0.029	3.1
LSTM	171	0.98	694	24	6	0.45	0.469	3.8
LSTM	172	0.98	694	24	12	0.43	0.117	3.6
LSTM	173	0.98	694	24	24	0.38	0.000	3
LSTM	174	0.98	694	24	48	0.48	0.088	3.8
LSTM	175	0.98	694	24	96	0.43	0.726	4.7
Linear	176	0.98	1388	1	1	0.94	0.000	3.8

LSTM	177	0.98	1388	1	3	0.26	0.278	2
LSTM	178	0.98	1388	1	6	0.27	0.102	2.2
LSTM	179	0.98	1388	1	12	0.23	0.059	1.7
LSTM	180	0.98	1388	1	24	0.27	0.088	2.2
LSTM	181	0.98	1388	1	48	0.21	0.191	1.5
LSTM	182	0.98	1388	1	96	0.26	0.103	1.9
Linear	183	0.98	1388	3	1	1.90	0.000	6.7
LSTM	184	0.98	1388	3	3	0.33	0.073	2.6
LSTM	185	0.98	1388	3	6	0.34	0.117	2.8
LSTM	186	0.98	1388	3	12	0.27	0.176	2.1
LSTM	187	0.98	1388	3	24	0.31	0.205	2.3
LSTM	188	0.98	1388	3	48	0.32	0.044	2.4
LSTM	189	0.98	1388	3	96	0.27	0.207	2.4
Linear	190	0.98	1388	6	1	2.58	0.000	9.1
LSTM	191	0.98	1388	6	3	0.33	0.146	2.9
LSTM	192	0.98	1388	6	6	0.34	0.102	2.6
LSTM	193	0.98	1388	6	12	0.31	0.117	2.6
LSTM	194	0.98	1388	6	24	0.33	0.088	2.4
LSTM	195	0.98	1388	6	48	0.35	0.147	2.6
LSTM	196	0.98	1388	6	96	0.29	0.192	2.4
Linear	197	0.98	1388	12	1	3.08	0.000	11.1
LSTM	198	0.98	1388	12	3	0.42	0.146	3.2
LSTM	199	0.98	1388	12	6	0.43	0.190	3.3
LSTM	200	0.98	1388	12	12	0.34	0.059	2.7
LSTM	201	0.98	1388	12	24	0.39	0.088	3
LSTM	202	0.98	1388	12	48	0.35	0.309	3.3
LSTM	203	0.98	1388	12	96	0.36	0.118	3
Linear	204	0.98	1388	24	1	2.56	0.000	9.3
LSTM	205	0.98	1388	24	3	0.39	0.015	3
LSTM	206	0.98	1388	24	6	0.37	0.000	3
LSTM	207	0.98	1388	24	12	0.36	0.000	2.9
LSTM	208	0.98	1388	24	24	0.44	0.000	3.2
LSTM	209	0.98	1388	24	48	0.42	0.044	3.5
LSTM	210	0.98	1388	24	96	0.36	0.000	2.8
Linear	211	0.98	2082	1	1	0.92	0.000	3.5
LSTM	212	0.98	2082	1	3	0.30	0.073	2.4
LSTM	213	0.98	2082	1	6	0.25	0.117	2
LSTM	214	0.98	2082	1	12	0.28	0.644	2.5
LSTM	215	0.98	2082	1	24	0.27	0.000	2.1
LSTM	216	0.98	2082	1	48	0.27	0.397	2.3
LSTM	217	0.98	2082	1	96	0.27	0.089	2
Linear	218	0.98	2082	3	1	2.22	0.000	7.8
LSTM	219	0.98	2082	3	3	0.32	0.205	2.7
LSTM	220	0.98	2082	3	6	0.30	0.293	2.7
LSTM	221	0.98	2082	3	12	0.34	0.117	3

LSTM	222	0.98	2082	3	24	0.26	0.191	2
LSTM	223	0.98	2082	3	48	0.28	0.132	2.2
LSTM	224	0.98	2082	3	96	0.28	0.148	2.3
Linear	225	0.98	2082	6	1	2.77	0.000	10.1
LSTM	226	0.98	2082	6	3	0.31	0.146	2.5
LSTM	227	0.98	2082	6	6	0.34	0.102	2.8
LSTM	228	0.98	2082	6	12	0.33	0.161	2.6
LSTM	229	0.98	2082	6	24	0.33	0.176	2.9
LSTM	230	0.98	2082	6	48	0.35	0.074	2.8
LSTM	231	0.98	2082	6	96	0.29	0.089	2.4
Linear	232	0.98	2082	12	1	3.10	0.000	11.2
LSTM	233	0.98	2082	12	3	0.37	0.146	3.1
LSTM	234	0.98	2082	12	6	0.39	0.073	3
LSTM	235	0.98	2082	12	12	0.40	0.073	3.3
LSTM	236	0.98	2082	12	24	0.37	0.044	2.9
LSTM	237	0.98	2082	12	48	0.38	0.147	3
LSTM	238	0.98	2082	12	96	0.34	0.089	2.7
Linear	239	0.98	2082	24	1	2.57	0.000	9.3
LSTM	240	0.98	2082	24	3	0.42	0.000	3.2
LSTM	241	0.98	2082	24	6	0.39	0.000	2.8
LSTM	242	0.98	2082	24	12	0.48	0.000	4.7
LSTM	243	0.98	2082	24	24	0.32	0.000	2.7
LSTM	244	0.98	2082	24	48	0.37	0.000	3
LSTM	245	0.98	2082	24	96	0.40	0.000	3.1
Linear	246	0.98	2776	1	1	1.29	0.000	4.6
LSTM	247	0.98	2776	1	3	0.31	0.044	2.3
LSTM	248	0.98	2776	1	6	0.20	0.161	1.6
LSTM	249	0.98	2776	1	12	0.30	0.059	2.2
LSTM	250	0.98	2776	1	24	0.26	0.117	2
LSTM	251	0.98	2776	1	48	0.28	0.088	2
LSTM	252	0.98	2776	1	96	0.23	0.163	1.9
Linear	253	0.98	2776	3	1	1.87	0.000	6.7
LSTM	254	0.98	2776	3	3	0.28	0.146	2.4
LSTM	255	0.98	2776	3	6	0.32	0.146	2.6
LSTM	256	0.98	2776	3	12	0.32	0.117	2.5
LSTM	257	0.98	2776	3	24	0.31	0.073	2.3
LSTM	258	0.98	2776	3	48	0.30	0.044	2.3
LSTM	259	0.98	2776	3	96	0.28	0.207	2.2
Linear	260	0.98	2776	6	1	2.33	0.000	8.4
LSTM	261	0.98	2776	6	3	0.32	0.161	2.7
LSTM	262	0.98	2776	6	6	0.36	0.102	2.9
LSTM	263	0.98	2776	6	12	0.31	0.176	2.6
LSTM	264	0.98	2776	6	24	0.32	0.073	2.5
LSTM	265	0.98	2776	6	48	0.38	0.000	3.1
LSTM	266	0.98	2776	6	96	0.32	0.192	2.5

Linear	267	0.98	2776	12	1	3.03	0.000	10.7
LSTM	268	0.98	2776	12	3	0.44	0.029	3.5
LSTM	269	0.98	2776	12	6	0.45	0.059	3.2
LSTM	270	0.98	2776	12	12	0.38	0.117	3.3
LSTM	271	0.98	2776	12	24	0.38	0.000	2.8
LSTM	272	0.98	2776	12	48	0.48	0.912	5.3
LSTM	273	0.98	2776	12	96	0.34	0.207	3
Linear	274	0.98	2776	24	1	2.54	0.000	9.2
LSTM	275	0.98	2776	24	3	0.44	0.000	3.2
LSTM	276	0.98	2776	24	6	0.42	0.000	3.3
LSTM	277	0.98	2776	24	12	0.43	0.029	3.3
LSTM	278	0.98	2776	24	24	0.36	0.000	2.8
LSTM	279	0.98	2776	24	48	0.34	0.074	2.9
LSTM	280	0.98	2776	24	96	0.34	0.000	2.9
Linear	281	0.98	3470	1	1	1.19	0.000	4.5
LSTM	282	0.98	3470	1	3	0.19	0.336	1.5
LSTM	283	0.98	3470	1	6	0.31	0.058	2.3
LSTM	284	0.98	3470	1	12	0.27	0.249	1.9
LSTM	285	0.98	3470	1	24	0.27	0.073	1.8
LSTM	286	0.98	3470	1	48	0.23	0.147	1.7
LSTM	287	0.98	3470	1	96	0.24	0.103	1.9
Linear	288	0.98	3470	3	1	2.31	0.000	8.5
LSTM	289	0.98	3470	3	3	0.28	0.161	2.2
LSTM	290	0.98	3470	3	6	0.30	0.161	2.3
LSTM	291	0.98	3470	3	12	0.34	0.263	2.7
LSTM	292	0.98	3470	3	24	0.26	0.191	2
LSTM	293	0.98	3470	3	48	0.25	0.103	2
LSTM	294	0.98	3470	3	96	0.25	0.148	2.2
Linear	295	0.98	3470	6	1	2.24	0.000	8
LSTM	296	0.98	3470	6	3	0.30	0.117	2.5
LSTM	297	0.98	3470	6	6	0.35	0.102	3.1
LSTM	298	0.98	3470	6	12	0.31	0.161	2.3
LSTM	299	0.98	3470	6	24	0.34	0.088	2.8
LSTM	300	0.98	3470	6	48	0.39	0.000	3.1
LSTM	301	0.98	3470	6	96	0.30	0.177	2.4
Linear	302	0.98	3470	12	1	2.97	0.000	10.5
LSTM	303	0.98	3470	12	3	0.33	0.073	2.7
LSTM	304	0.98	3470	12	6	0.43	0.102	3.2
LSTM	305	0.98	3470	12	12	0.46	0.044	3.4
LSTM	306	0.98	3470	12	24	0.36	0.000	2.9
LSTM	307	0.98	3470	12	48	0.34	0.059	2.6
LSTM	308	0.98	3470	12	96	0.31	0.089	2.7
Linear	309	0.98	3470	24	1	2.39	0.000	8.8
LSTM	310	0.98	3470	24	3	0.44	0.000	3.4
LSTM	311	0.98	3470	24	6	0.40	0.000	3.3

LSTM	312	0.98	3470	24	12	0.40	0.000	3.2
LSTM	313	0.98	3470	24	24	0.38	0.000	3.1
LSTM	314	0.98	3470	24	48	0.36	0.029	3
LSTM	315	0.98	3470	24	96	0.38	0.000	2.8
Linear	316	0.96	1288	1	1	0.25	0.073	1.5
LSTM	317	0.96	1288	1	3	0.21	0.132	1.5
LSTM	318	0.96	1288	1	6	0.19	0.175	1.7
LSTM	319	0.96	1288	1	12	0.24	0.015	2.4
LSTM	320	0.96	1288	1	24	0.31	0.029	2.2
LSTM	321	0.96	1288	1	48	0.18	0.132	1.5
LSTM	322	0.96	1288	1	96	0.17	0.177	1.4
Linear	323	0.96	1288	3	1	0.37	0.044	2.3
LSTM	324	0.96	1288	3	3	0.24	0.117	2.1
LSTM	325	0.96	1288	3	6	0.23	0.205	2
LSTM	326	0.96	1288	3	12	0.23	0.146	2
LSTM	327	0.96	1288	3	24	0.24	0.088	1.9
LSTM	328	0.96	1288	3	48	0.35	0.015	2.7
LSTM	329	0.96	1288	3	96	0.29	0.000	2.7
Linear	330	0.96	1288	6	1	0.81	0.000	4
LSTM	331	0.96	1288	6	3	0.29	0.044	2.5
LSTM	332	0.96	1288	6	6	0.32	0.088	2.8
LSTM	333	0.96	1288	6	12	0.34	0.015	3.1
LSTM	334	0.96	1288	6	24	0.31	0.029	2.5
LSTM	335	0.96	1288	6	48	0.28	0.015	2.3
LSTM	336	0.96	1288	6	96	0.42	0.000	4.2
Linear	337	0.96	1288	12	1	0.97	0.000	5
LSTM	338	0.96	1288	12	3	0.36	0.000	3
LSTM	339	0.96	1288	12	6	0.39	0.000	3.3
LSTM	340	0.96	1288	12	12	0.36	0.000	3
LSTM	341	0.96	1288	12	24	0.41	0.000	3.5
LSTM	342	0.96	1288	12	48	0.33	0.000	2.7
LSTM	343	0.96	1288	12	96	0.35	0.104	3.1
Linear	344	0.96	1288	24	1	1.48	0.000	7.2
LSTM	345	0.96	1288	24	3	0.38	0.000	3.1
LSTM	346	0.96	1288	24	6	0.41	0.000	3.4
LSTM	347	0.96	1288	24	12	0.53	0.000	4.4
LSTM	348	0.96	1288	24	24	0.44	0.000	3.7
LSTM	349	0.96	1288	24	48	0.41	0.000	3.2
LSTM	350	0.96	1288	24	96	0.38	0.000	3.1
Linear	351	0.96	2576	1	1	0.38	0.029	2
LSTM	352	0.96	2576	1	3	0.18	0.219	1.5
LSTM	353	0.96	2576	1	6	0.20	0.102	1.7
LSTM	354	0.96	2576	1	12	0.25	0.498	2.5
LSTM	355	0.96	2576	1	24	0.22	0.059	1.6
LSTM	356	0.96	2576	1	48	0.19	0.088	1.6

LSTM	357	0.96	2576	1	96	0.23	0.015	2.1
Linear	358	0.96	2576	3	1	0.52	0.015	2.7
LSTM	359	0.96	2576	3	3	0.41	0.000	4
LSTM	360	0.96	2576	3	6	0.25	0.073	2.1
LSTM	361	0.96	2576	3	12	0.24	0.088	2
LSTM	362	0.96	2576	3	24	0.30	0.088	2.7
LSTM	363	0.96	2576	3	48	0.35	0.000	3.2
LSTM	364	0.96	2576	3	96	0.23	0.118	2
Linear	365	0.96	2576	6	1	0.36	0.000	2.4
LSTM	366	0.96	2576	6	3	0.31	0.029	2.5
LSTM	367	0.96	2576	6	6	0.32	0.102	2.9
LSTM	368	0.96	2576	6	12	0.32	0.029	2.6
LSTM	369	0.96	2576	6	24	0.32	0.059	2.5
LSTM	370	0.96	2576	6	48	0.28	0.088	2.5
LSTM	371	0.96	2576	6	96	0.33	0.030	2.6
Linear	372	0.96	2576	12	1	1.31	0.000	6.2
LSTM	373	0.96	2576	12	3	0.42	0.015	3.9
LSTM	374	0.96	2576	12	6	0.40	0.000	3.2
LSTM	375	0.96	2576	12	12	0.38	0.000	3.2
LSTM	376	0.96	2576	12	24	0.43	0.000	3.6
LSTM	377	0.96	2576	12	48	0.39	0.015	3.1
LSTM	378	0.96	2576	12	96	0.47	0.000	3.8
Linear	379	0.96	2576	24	1	1.14	0.000	5.8
LSTM	380	0.96	2576	24	3	0.43	0.000	3.6
LSTM	381	0.96	2576	24	6	0.43	0.000	3.2
LSTM	382	0.96	2576	24	12	0.42	0.000	3.5
LSTM	383	0.96	2576	24	24	0.42	0.000	3.5
LSTM	384	0.96	2576	24	48	0.40	0.000	3.2
LSTM	385	0.96	2576	24	96	0.41	0.000	3.2
Linear	386	0.96	3864	1	1	0.18	0.161	1.2
LSTM	387	0.96	3864	1	3	0.16	0.161	1.3
LSTM	388	0.96	3864	1	6	0.18	0.175	1.4
LSTM	389	0.96	3864	1	12	0.23	0.059	1.7
LSTM	390	0.96	3864	1	24	0.19	0.161	1.4
LSTM	391	0.96	3864	1	48	0.18	0.353	1.6
LSTM	392	0.96	3864	1	96	0.18	0.118	1.5
Linear	393	0.96	3864	3	1	0.23	0.146	1.8
LSTM	394	0.96	3864	3	3	0.28	0.088	2.1
LSTM	395	0.96	3864	3	6	0.23	0.161	2
LSTM	396	0.96	3864	3	12	0.22	0.161	1.9
LSTM	397	0.96	3864	3	24	0.28	0.191	2.2
LSTM	398	0.96	3864	3	48	0.24	0.088	2
LSTM	399	0.96	3864	3	96	0.23	0.192	1.8
Linear	400	0.96	3864	6	1	0.37	0.000	2.4
LSTM	401	0.96	3864	6	3	0.56	0.000	4.6

LSTM	402	0.96	3864	6	6	0.34	0.029	3
LSTM	403	0.96	3864	6	12	0.39	0.000	3.7
LSTM	404	0.96	3864	6	24	0.27	0.176	2.3
LSTM	405	0.96	3864	6	48	0.33	0.074	3.1
LSTM	406	0.96	3864	6	96	0.28	0.030	2.3
Linear	407	0.96	3864	12	1	0.97	0.000	5.1
LSTM	408	0.96	3864	12	3	0.44	0.059	4.1
LSTM	409	0.96	3864	12	6	0.42	0.000	3.1
LSTM	410	0.96	3864	12	12	0.40	0.000	3.6
LSTM	411	0.96	3864	12	24	0.41	0.000	2.9
LSTM	412	0.96	3864	12	48	0.49	0.000	4.1
LSTM	413	0.96	3864	12	96	0.45	0.592	5
Linear	414	0.96	3864	24	1	0.86	0.000	4.7
LSTM	415	0.96	3864	24	3	0.47	0.000	3.5
LSTM	416	0.96	3864	24	6	0.42	0.000	3.7
LSTM	417	0.96	3864	24	12	0.46	0.000	4
LSTM	418	0.96	3864	24	24	0.40	0.000	3.3
LSTM	419	0.96	3864	24	48	0.45	0.000	3.8
LSTM	420	0.96	3864	24	96	0.44	0.000	3.8
Linear	421	0.96	5152	1	1	0.60	0.015	2.9
LSTM	422	0.96	5152	1	3	0.24	0.015	2.2
LSTM	423	0.96	5152	1	6	0.17	0.132	1.3
LSTM	424	0.96	5152	1	12	0.20	0.205	1.7
LSTM	425	0.96	5152	1	24	0.15	0.161	1.2
LSTM	426	0.96	5152	1	48	0.19	0.176	1.5
LSTM	427	0.96	5152	1	96	0.23	0.000	2.1
Linear	428	0.96	5152	3	1	0.65	0.000	3.3
LSTM	429	0.96	5152	3	3	0.22	0.102	1.8
LSTM	430	0.96	5152	3	6	0.24	0.088	1.9
LSTM	431	0.96	5152	3	12	0.26	0.044	2.3
LSTM	432	0.96	5152	3	24	0.27	0.103	2
LSTM	433	0.96	5152	3	48	0.32	0.073	2.6
LSTM	434	0.96	5152	3	96	0.24	0.089	1.9
Linear	435	0.96	5152	6	1	0.42	0.000	2.8
LSTM	436	0.96	5152	6	3	0.38	0.000	3.6
LSTM	437	0.96	5152	6	6	0.35	0.132	2.9
LSTM	438	0.96	5152	6	12	0.35	0.073	3
LSTM	439	0.96	5152	6	24	0.34	0.015	2.8
LSTM	440	0.96	5152	6	48	0.26	0.088	2.2
LSTM	441	0.96	5152	6	96	0.30	0.104	2.5
Linear	442	0.96	5152	12	1	1.15	0.000	5.6
LSTM	443	0.96	5152	12	3	0.39	0.000	3.3
LSTM	444	0.96	5152	12	6	0.40	0.000	3.7
LSTM	445	0.96	5152	12	12	0.41	0.000	3.5
LSTM	446	0.96	5152	12	24	0.43	0.000	3.6

LSTM	447	0.96	5152	12	48	0.50	0.000	3.2
LSTM	448	0.96	5152	12	96	0.32	0.000	2.7
Linear	449	0.96	5152	24	1	0.81	0.000	4.6
LSTM	450	0.96	5152	24	3	0.43	0.000	3.5
LSTM	451	0.96	5152	24	6	0.41	0.000	3.4
LSTM	452	0.96	5152	24	12	0.44	0.015	3.1
LSTM	453	0.96	5152	24	24	0.42	0.000	3.6
LSTM	454	0.96	5152	24	48	0.43	0.000	3.5
LSTM	455	0.96	5152	24	96	0.46	0.000	3.8
Linear	456	0.96	6440	1	1	0.39	0.058	2
LSTM	457	0.96	6440	1	3	0.17	0.117	1.4
LSTM	458	0.96	6440	1	6	0.18	0.278	1.4
LSTM	459	0.96	6440	1	12	0.24	0.029	1.9
LSTM	460	0.96	6440	1	24	0.17	0.176	1.5
LSTM	461	0.96	6440	1	48	0.18	0.206	1.6
LSTM	462	0.96	6440	1	96	0.19	0.118	1.6
Linear	463	0.96	6440	3	1	0.89	0.000	4.3
LSTM	464	0.96	6440	3	3	0.23	0.132	1.8
LSTM	465	0.96	6440	3	6	0.23	0.132	2
LSTM	466	0.96	6440	3	12	0.29	0.146	2.5
LSTM	467	0.96	6440	3	24	0.25	0.117	1.9
LSTM	468	0.96	6440	3	48	0.24	0.059	1.9
LSTM	469	0.96	6440	3	96	0.27	0.074	2.2
Linear	470	0.96	6440	6	1	0.51	0.000	3
LSTM	471	0.96	6440	6	3	0.34	0.073	3.2
LSTM	472	0.96	6440	6	6	0.33	0.117	3.1
LSTM	473	0.96	6440	6	12	0.35	0.000	2.8
LSTM	474	0.96	6440	6	24	0.28	0.073	2.4
LSTM	475	0.96	6440	6	48	0.32	0.044	2.6
LSTM	476	0.96	6440	6	96	0.30	0.059	2.7
Linear	477	0.96	6440	12	1	1.35	0.000	6.4
LSTM	478	0.96	6440	12	3	0.39	0.000	3.3
LSTM	479	0.96	6440	12	6	0.34	0.000	2.9
LSTM	480	0.96	6440	12	12	0.38	0.015	3.3
LSTM	481	0.96	6440	12	24	0.45	0.000	3.6
LSTM	482	0.96	6440	12	48	0.40	0.000	3.4
LSTM	483	0.96	6440	12	96	0.36	0.015	3.2
Linear	484	0.96	6440	24	1	0.94	0.000	5
LSTM	485	0.96	6440	24	3	0.61	0.000	6.1
LSTM	486	0.96	6440	24	6	0.47	0.000	4.1
LSTM	487	0.96	6440	24	12	0.45	0.000	3.8
LSTM	488	0.96	6440	24	24	0.44	0.000	3.9
LSTM	489	0.96	6440	24	48	0.43	0.000	3.4
LSTM	490	0.96	6440	24	96	0.40	0.000	3.4
Linear	491	0.94	1810	1	1	0.82	0.000	4.3

LSTM	492	0.94	1810	1	3	0.22	0.073	1.6
LSTM	493	0.94	1810	1	6	0.19	0.073	1.7
LSTM	494	0.94	1810	1	12	0.23	0.102	1.5
LSTM	495	0.94	1810	1	24	0.17	0.147	1.4
LSTM	496	0.94	1810	1	48	0.25	0.015	2.1
LSTM	497	0.94	1810	1	96	0.19	0.148	1.5
Linear	498	0.94	1810	3	1	0.22	0.249	1.6
LSTM	499	0.94	1810	3	3	0.34	0.015	2.6
LSTM	500	0.94	1810	3	6	0.23	0.088	1.9
LSTM	501	0.94	1810	3	12	0.25	0.073	2.1
LSTM	502	0.94	1810	3	24	0.32	0.015	2.7
LSTM	503	0.94	1810	3	48	0.30	0.059	2.6
LSTM	504	0.94	1810	3	96	0.29	0.059	2.6
Linear	505	0.94	1810	6	1	0.31	0.073	2.1
LSTM	506	0.94	1810	6	3	0.27	0.044	2.2
LSTM	507	0.94	1810	6	6	0.29	0.000	2.5
LSTM	508	0.94	1810	6	12	0.31	0.029	2.3
LSTM	509	0.94	1810	6	24	0.41	0.147	4.3
LSTM	510	0.94	1810	6	48	0.39	0.088	3.4
LSTM	511	0.94	1810	6	96	0.29	0.089	2.4
Linear	512	0.94	1810	12	1	0.36	0.000	2.4
LSTM	513	0.94	1810	12	3	0.54	0.029	3.9
LSTM	514	0.94	1810	12	6	0.36	0.073	3.3
LSTM	515	0.94	1810	12	12	0.36	0.000	2.9
LSTM	516	0.94	1810	12	24	0.40	0.015	3.3
LSTM	517	0.94	1810	12	48	0.33	0.000	2.8
LSTM	518	0.94	1810	12	96	0.34	0.030	3
Linear	519	0.94	1810	24	1	0.74	0.000	4
LSTM	520	0.94	1810	24	3	0.42	0.000	3.5
LSTM	521	0.94	1810	24	6	0.40	0.000	3
LSTM	522	0.94	1810	24	12	0.41	0.000	3.6
LSTM	523	0.94	1810	24	24	0.40	0.000	3.3
LSTM	524	0.94	1810	24	48	0.38	0.000	3.2
LSTM	525	0.94	1810	24	96	0.44	0.000	3.5
Linear	526	0.94	3620	1	1	0.47	0.000	2.4
LSTM	527	0.94	3620	1	3	0.23	0.102	1.6
LSTM	528	0.94	3620	1	6	0.19	0.132	1.4
LSTM	529	0.94	3620	1	12	0.30	0.000	2
LSTM	530	0.94	3620	1	24	0.17	0.103	1.4
LSTM	531	0.94	3620	1	48	0.19	0.132	1.5
LSTM	532	0.94	3620	1	96	0.19	0.488	1.9