University of Oxford

Improving Spatio-Temporal Forecasting by Extracting Neighbourhood Indicators Based on Past Observations

by

Jessica Kristen Rapson

Mansfield College

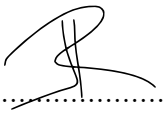
A dissertation submitted in partial fulfilment
of the degree of Master of Science in Statistical Science

Department of Statistics, 24–29 St Giles,
Oxford, OX1 3LB

September 2023

This is my own work (except where otherwise indicated)

Candidate: Jessica Kristen Rapson

Signed:  .....

Date:2023-09-06.....

Abstract

The intersections between space and time inherent to spatio-temporal forecasting problems pose unique challenges that necessitate special consideration. Time-delay embedding, which consists of transforming the problem into a multiple regression task where predictors are previous past values of the series, is the most frequent predictive approach employed by spatio-temporal models. However, many spatio-temporal phenomena depend not only on recent past conditions at the same location, but also on recent past conditions of nearby locations. Spatio-temporal indicators that capture the dynamics of a spatio-temporal neighbourhood can improve prediction accuracy, though the value of these indicators can be sensitive the definition of the spatio-temporal neighbourhood. This dissertation identifies methods for optimising spatio-temporal neighbourhoods to improve indicator extraction for use in spatio-temporal forecasting. The approach is experimentally evaluated on three data sets featuring different types of spatial data. It is shown that indicators which are extracted from optimised spatio-temporal neighbourhoods have an improved forecast skill. Advantages of this approach, which can be viewed as a general approach to spatio-temporal prediction, are discussed.

Acknowledgements

I would like to thank my supervisors Sir Bernard Silverman and Dr Louise Slater for their expertise and advice throughout my work towards this dissertation. I would also like to thank my partner, Alec Xu, for his constant support in all that I do.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Tobler's first law of geography holds that "everything is related to everything else, but near things are more related than distant things" [1]. This quirk of spatial data naturally poses challenges for traditional predictive modelling assumptions that observations and model errors are independent from one another. Spatial forecasting is further complicated when data has a temporal dimension and thus exhibits dependencies across both space and time.

Despite these challenges, spatio-temporal forecasting – making predictions about spatial data that changes over time – is a rapidly evolving field with increasing potential for applications in fields such as hydrology, meteorology, public safety, and transportation, among many others [2, 3, 4]. Common approaches employ methods from time series forecasting and spatial interpolation. However, the intersections between space and time inherent to these problems pose unique challenges that necessitate special consideration.

1.1 Dissertation Structure

The dissertation is structured as follows:

Chapter 1 gives an overview of the current state of spatio-temporal forecasting, including descriptions of commonly used forecasting methods and recent advancements in spatio-temporal models. It also presents relevant literature on spatio-temporal neighbourhoods, spatio-temporal indicators, feature selection methods, and spatio-temporal model performance evaluation that are applied in the experiments.

Chapter 2 provides a detailed overview of the methodology used to optimise spatio-temporal neighbourhoods including a description of the three sample data sources used to evaluate the methodology, a synopsis of the baseline spatio-temporal forecasting models upon which the improvements are tested, an explanation of the process for extracting spatio-temporal indicators, a justification for the methods of spatio-temporal neighbourhood optimisation that are ultimately applied, an outline of the forecasting methodology applied, and a brief account of the techniques used to evaluate the results of the experiment.

Chapter 3 provides the results of the experiments using the optimised spatio-temporal neighbourhood indicators and illustrates the differences in neighbourhood parameters for each sample data source.

Lastly, Chapter 4 discusses the main contributions of the dissertation, any resulting limitations, and provides several suggestions for additional research.

An appendix providing a glossary of relevant terminology and code samples for replicating results on the CAMELS-GB data is also provided.

1.2 Problem Formulation

The spatio-temporal forecasting problem can be posed as follows: take a set of locations $L = \{l_1, ..., l_n\}$, a set of time-stamps $T = \{t_1, ..., t_m\}$, and a set of observations

$$\mathcal{D} = \{\{y_{1,1}, (x^1_{1,1}, ..., x^k_{1,1})\}, ..., \{y_{i,j}, (x^1_{i,j}, ..., x^k_{i,j})\}\}^{j\in\{1,2,...,m\}}_{i\in\{1,2,...,n\}}, \tag{1.1}$$

where $y_{i,j}$ and $x^k_{i,j}$ correspond, respectively, to the values of the output vector Y and the predictor matrix X_k at time t_i in geographical location l_j [5]. The goal is to predict the value of Y at a location of interest, $l_s, s \in \{1, 2, .., n\}$, at a time in the future, t_f, given the observed values $y_{i,j}$ and predictor vector $\mathbf{x_{i,j}}$ such that $t_m < t_f$.

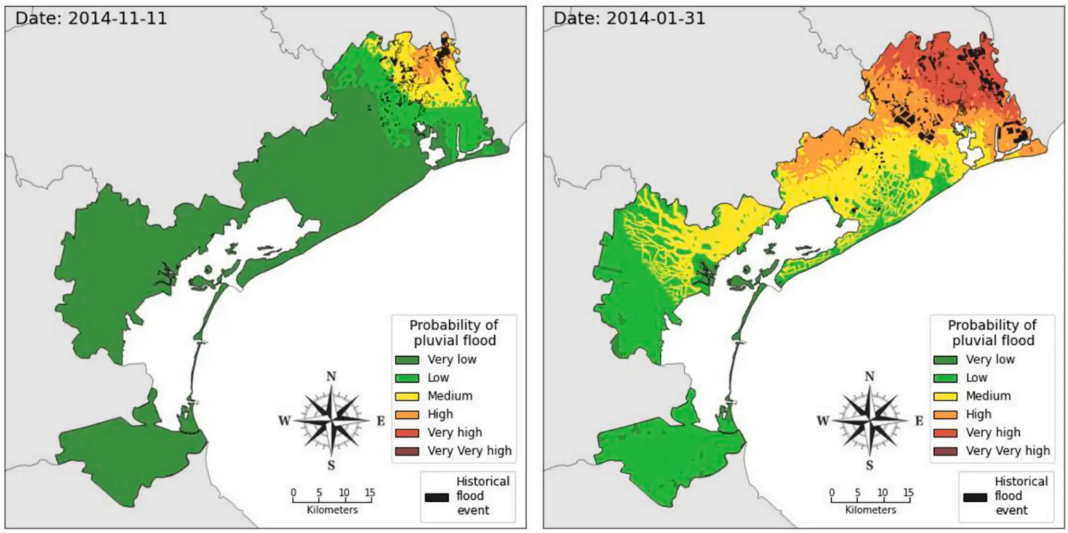


Figure 1.1: A spatio-temporal forecast for pluvial flood areas in Venice [6].

The most frequent predictive approach employed by spatio-temporal machine learning models consists of transforming the original problem into a multiple regression task, where the target variable is the future value of the series and the predictors are previous past values of the series up to a certain time window [7]. This transformation technique is known as time-delay embedding [8]. The goal is to provide the modelling algorithm with information on the recent dynamics of the time series by leveraging past predictors.

These approaches assume that the future conditions depend on recently observed conditions in the same location. However, certain spatio-temporal variables of interest (e.g. flooding, human trafficking, wind speeds) depend not only on the recent past conditions at the same location, but also on recent past conditions of nearby locations. In other words, the "spatial radius of influence" may decrease for older observations such that observations that are more proximate in both time and location have a larger influence on the outcome variable than temporally and spatially distant observations. Thus, models that are fed only with values from the same location for which a future prediction is required may be limiting.

Ohashi and Torgo [7] address this problem by proposing spatio-temporal indicators, a method based on data pre-processing that uses statistics summarising past data within a spatio-temporal distance of the target observation as predictors. These spatio-temporal indicators assume that data from distant neighbours becomes less relevant as the model looks further back in time. The main idea behind their proposal was to develop predictors capable of capturing the spatio-temporal dynamics of a given time series. These extra predictors then provide the model with important information on the recent spatio-temporal patterns of the time series, which in turn improve the model prediction accuracy.

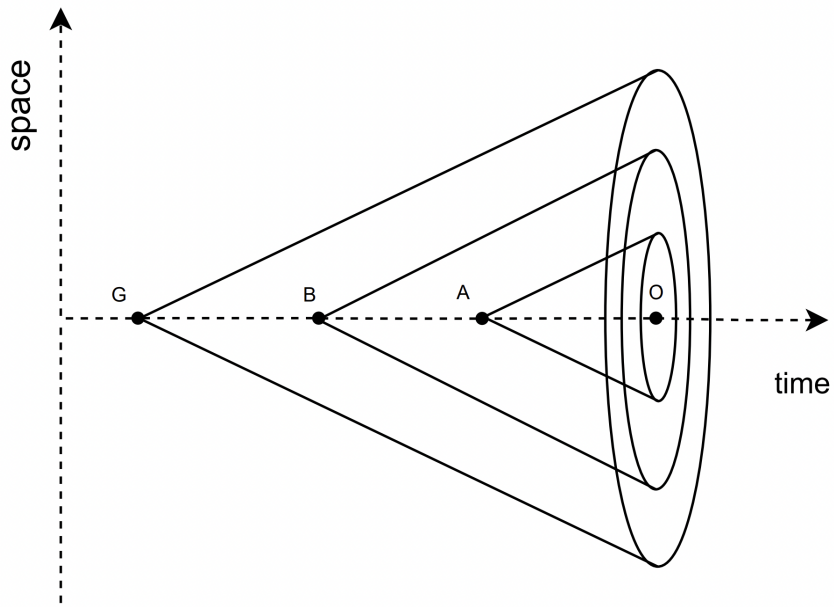


Figure 1.2: Spatio-temporal neighbourhoods of different sizes [7].

The model is specified as follows: the spatio-temporal indicators summarise values within spatio-temporal neighbourhoods of an observation, $x_{i,j}$, whose value is the target of the prediction [5]. This variable $x_{i,j}$ is measured at time t_i and location l_j. Its spatio-temporal neighbourhood contains all past observations within specified boundary, β, of spatio-temporal distance

$$D_A = \alpha \cdot D_A^S + (1 - \alpha) \cdot D_A^T \leq \beta, \tag{1.2}$$

where D_A^S is the spatial distance between the locations where O and a reference point A occur, D_A^T is the temporal distance between observations O and A, α is a weighting factor that determines the relative importance of spatially vs. temporally proximate observations, and β is the chosen maximum spatio-temporal distance.

Indicators are only based on past observations, so temporal distance is defined as

$$D_A^T = \begin{cases} \frac{t_i - t_A}{t_{max} - t_{min}}, & \text{if } t_A < t_i \\ \infty, & \text{otherwise.} \end{cases} \tag{1.3}$$

Note that the spatial and temporal distances D_A^S and D_A^T are normalized to be in the $[0, 1]$ interval such that the spatio-temporal boundaries are defined relatively, accounting for spatio-temporal distances differing within and between data sets.

Having defined the spatio-temporal distance between two observations, the spatio-temporal neighbourhood of an observation $x_{i,j}$ can then be defined as the set of points within a certain spatio-temporal distance of that observation

$$\mathcal{N}_x^\beta = \{A \in \mathcal{D} : D_A \leq \beta\}, \tag{1.4}$$

where $\mathcal{D}$ is the available spatio-temporal data set.

It is also true that spatio-temporal variables of interest may take some time to "travel" from one point to another [5]. In this "reversed cone" paradigm, spatially distance observations have a larger effect on the predicted variable when they are further away temporally. For example, in order to predict flooding in a given region, it may be more useful to utilise observations from further upstream locations than immediately proximate ones. Similarly, human trafficking activity in a major city may be predictive of future human trafficking in smaller cities located further away after sufficient time has passed.

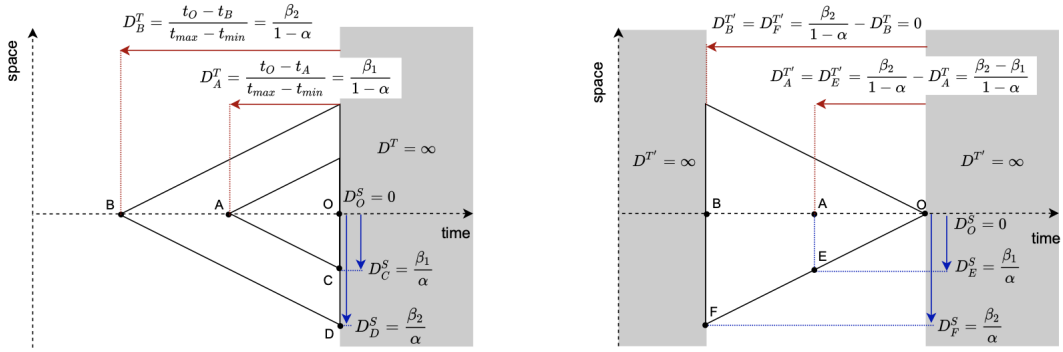
In this spatio-temporal relationship, the new temporal distance is defined as

$$D_A^{T'} = \begin{cases} \frac{\beta}{1-\alpha} - D_A^T, & \text{if } \frac{\beta}{1-\alpha} < D_A^T, t_A < t_O \\ \infty, & \text{otherwise.} \end{cases} \tag{1.5}$$

Thus the combined spatio-temporal distance can be recalculated as

$$\begin{aligned} D_A^{'} &= \alpha \cdot D_A^S + (1 - \alpha) \cdot D_A^{T'} \leq \beta \\ &= \alpha \cdot D_A^S + \beta - (1 - \alpha) D_A^T \leq \beta \\ &= \alpha \cdot D_A^S - (1 - \alpha) \cdot D_A^T \leq 0. \end{aligned} \tag{1.6}$$

These spatio-temporal indicators summarise the dynamics of the series within the neighbourhood. Given the above definitions, the spatio-temporal neighbourhood of a point can be thought of as a cone within space-time (Figure 2) [7]. Such cones represent which past values may influence the future value of the time series at that location and can be regarded as the spatio-temporal equivalents of time-delay embedding. Notably, different settings for α and β lead to cones of difference sizes and different definitions of temporal distance can change the orientation of the cone, altering the spatio-temporal embedded relationship.



(a) Original cone (b) Reversed cone

Figure 1.3: Reversed spatio-temporal neighbourhood paradigm [7].

Ohashi and Torgo [7] as well as Oliveira [5] combine spatio-temporal indicators with a wide range of modeling approaches (linear regression, multivariate adaptive regression splines, support vector machines, regression trees, and random forests) to greatly improve spatio-temporal predictions for specific data sets. However, the spatio-temporal relationships specified by the indicator equation are highly bespoke to the domain in which the forecast is produced. Thus changing parameters of the indicator greatly affect its utility as a predictor.

Presently, there have been no efforts to determine the optimal spatio-temporal cone shape or orientation for a given data set. In order to generalize spatio-temporal indicators for broader applications, methods for selecting the optimal spatio-temporal indicator specification need to be developed. This is achieved in two steps:

1. Identify methods for optimising parameters of spatio-temporal cone shape, α and β, as well as orientation, determined by D_A^T, for a given data set

2. Apply block cross-validation to evaluate parameter tuning methods on a variety of spatio-temporal data sets using different modeling approaches

Methodology for selecting an optimal set of spatio-temporal indicators has the potential to substantially improve the quality of spatio-temporal forecasts. The applications of such indicators are uniquely relevant to dynamical models – where input values can be altered to study how outputs change virtually in real time – as spatio-temporal indicators can leverage real-time data collected from neighbouring locations to make predictions about the future [9, 7]. The parameters that specify which spatio-temporal cone that is ultimately used in modelling also have

an interpretable meaning for the subject area, enabling researchers to learn what spatio-temporal relations exist within their data.

1.3 Literature Review

1.3.1 Spatio-Temporal Forecasting Models

In a world of increasing data availability and computational power, spatio-temporal forecasting is a rapidly growing field. Such forecasts play a role in various fields such as hydrology, environmental science, epidemiology, urban planning, and public safety. A multitude of methods have been developed to capture complex spatio-temporal relationships in data, aiming to provide accurate predictions and insights for decision-making. This section discusses a selection of relevant papers that highlight the diverse approaches in spatio-temporal forecasting – which may be improved through the introduction of optimised spatio-temporal neighbourhood indicators.

The simplest spatio-temporal models are autoregressive models. These include the space time autoregressive (STAR) model, introduced in the 1970s as a multivariate expansion for the univariate time series model [10]. Similar to autoregressive integrated moving average (ARIMA) models, which incorporate temporal dependencies into a linear regression framework, the STAR model incorporates linear dependencies in both temporal and spatial dimensions. This basic model was expanded upon by Ruchjana, Borovkova, and Lopuhaa [11] with the generalised space time autoregressive (GSTAR) model, which allowed the autoregressive parameters to vary between locations. This improvement enabled the GSTAR model to be applicable for heterogeneous characteristics of sample locations.

Further modifications to the generic spatio-temporal autoregressive model have been proposed to solve specialised problems, such as the autoregressive conditional Poisson (ACP) model proposed by Congdon in 2022 to model spatio-temporal patterns in COVID-19 infection rates [12]. Similar to STAR and GSTAR models, ACP considers different specifications of autoregressive dependence and introduces parameters that allow for both spatial and time variation in autoregressive effects. ACP generalizes these representations to area-time count data using Poisson regression to predict infection counts in spatially close areas. In meteorology, Sigrist, Künsch, and Stahel [13] propose accounting for spatial and temporal dependencies in precipitation with a latent Gaussian variable that follows a Markovian temporal evolution. This model is proposed as precipitation can be modelled using a power-transformed Normal distribution. Such a specification enables precipitation to be linked to non-separable covariances in space and time.

Another common method of spatio-temporal forecasting is kriging, sometimes referred to as Gaussian process regression [14, 15]. While kriging is most commonly used for spatial interpolation, purely spatial kriging approaches can be extended to include temporal covariates, enabling interpolation across time. This spatio-temporal extension is developed by Gräler, Pebesma and Heuvelink [16], who use it to forecast particulate matter concentrations measured at rural air quality monitor-

ing stations across Germany. They introduce separable, product-sum, metric, and sum-metric spatio-temporal covariance functions that parameterise the relationship between spatial and temporal covariance, both of which are individually specified using variograms – measures of spatial auto-correlation in a spatial dataset – equal to those from purely spatial variogram models. Similar models are applied by Lindström et al. [17] to forecast air pollution, Donnelly et al. [18] to forecast inundation depth for flood modelling, and Zhang et al. [19] to develop efficient implementations for forecasting precipitation and air temperature.

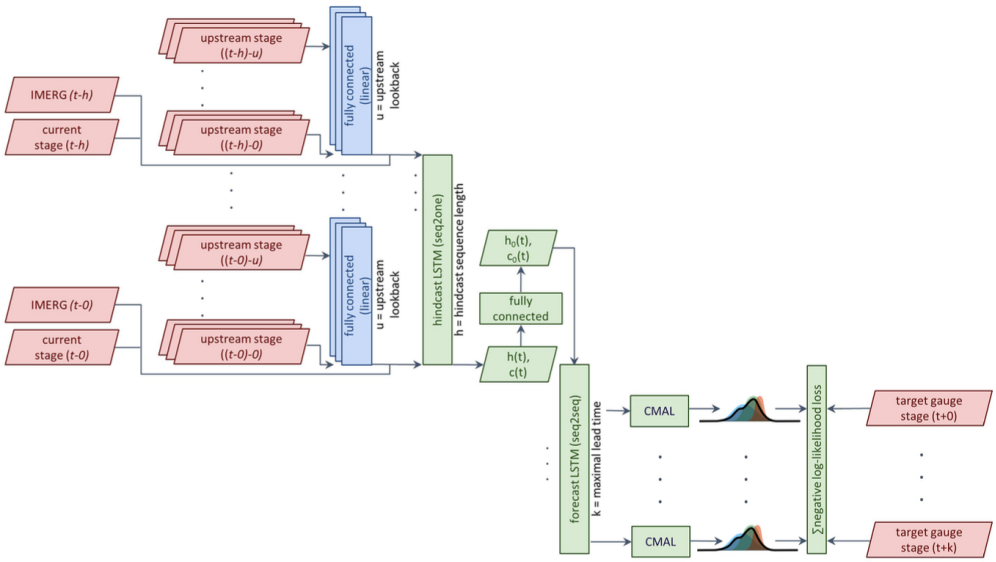


Figure 1.4: LSTM model employed by Google's flood forecasting system [20].

While autoregressive and kriging methods rely on strong assumptions about the data generation process, other methods exist that leverage computationally-intensive algorithms with more flexible hypothesis classes. Though these methods are often not able to provide probabilistic forecasts that enable calculating prediction intervals, they can be used to model more complex relationships between spatial and temporal components of the data. One of the most common of these methods is random forests. Hengle et al. [21] present a random forest for spatial predictions framework (RFsp) that uses buffer distances from observation points as explanatory variables, thus incorporating geographical proximity effects into the prediction process. They show that show that RFsp can obtain equally accurate and unbiased predictions to kriging, at the cost of increased computational complexity. Tyralis, Papacharalampous, and Langousis [22] review the use of these models in water resources applications, finding that so-called "data-driven" models such as random forests can out-perform physical models when sufficient data is available.

Finally, deep learning spatio-temporal forecasting models rely heavily on computational resources to build highly flexible, but potentially less interpretable, models of spatio-temporal processes. Examples include the use of artificial neural networks (ANNs) by Li, Dunham, and Xiao [23] to learn spatial correlations between locations

and fed with temporal forecasts of neighbouring locations to produce a spatially-influenced forecast for the target location. This spatially-influenced forecast is then combined with a separate temporal using regression to obtain a final forecast for the target location. Cheng and Wang [24] build on this model by substituting the static feed-forward ANN with with a dynamic recurrent network using feedback connections to predict wildfire area.

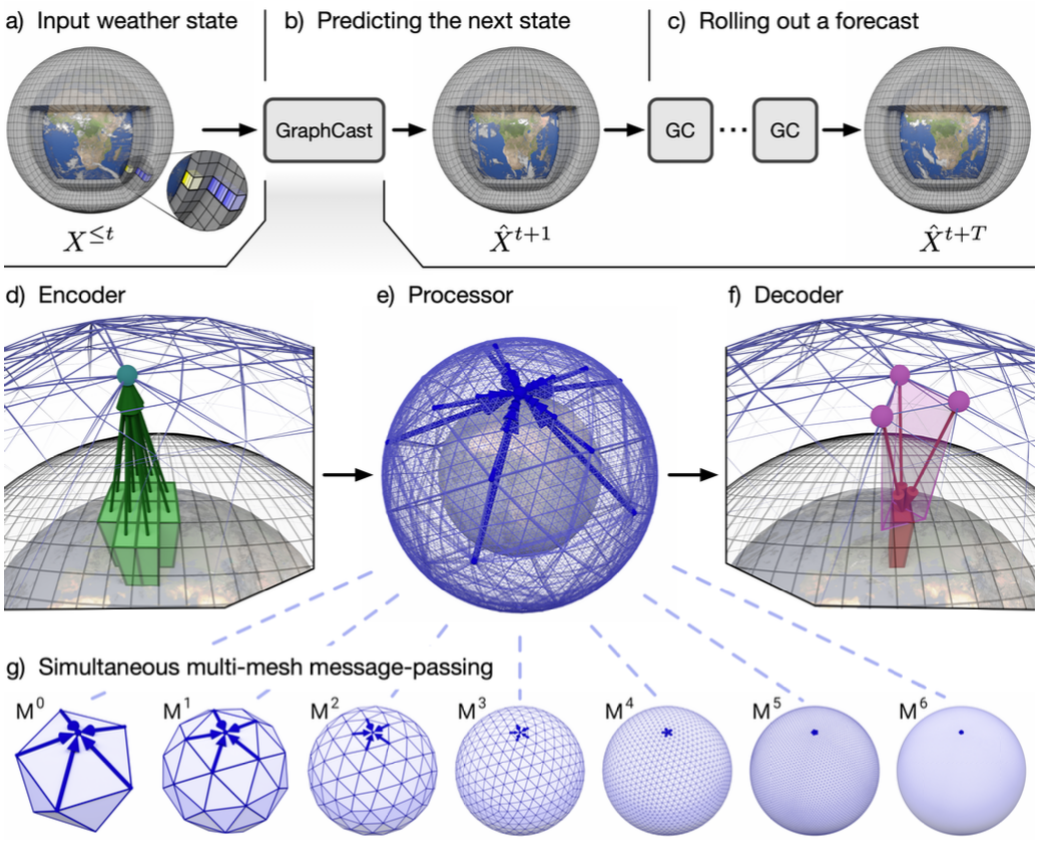


Figure 1.5: Spatio-temporal prediction process used by GraphCast [25].

Newer deep learning advancements have taken advantage of recurrent neural networks (RNNs), or bi-directional ANNs, to better incorporate the sequential nature of spatio-temporal data. This is currently the most active area for developments in spatio-temporal forecasting. Ghaderi, Sanandaji, and Ghaderi [26] present a wind speed forecasting algorithm using this method, which enables them to obtain forecasts of all nodes of the graph at the same time based on one framework. One of the leading models in the field, Google's flood forecasting model, is also produced using RNNs [20]. Specifically, the model uses a long short-term memory (LSTM) model, a type of RNN that overcomes the vanishing and exploding gradient issues inherent to recurrent models. The model uses hourly stage, or water level, data collected from local authorities and satellite-derived precipitation data as inputs to one LSTM model, which processes data from the past few days sequentially and feeds its output into a second LSTM model that produces the final forecast. Each pixel in the inundation map is treated as a separate classification task, predicting

whether the pixel will be inundated or not.

Klemmer et al. [27] propose a generalised adversarial network (GAN)-based approach for generating spatio-temporal weather patterns conditioned on detected extreme events. This method, a form of generative learning wherein two neural networks compete in a zero-sum game, enables predictions to be made for extreme weather events while capturing spatio-temporal processes simultaneously.

The most recent advancements in spatio-temporal forecasting incorporate spatially-driven deep learning methods, such as convolutional neural networks (CNNs) or graph neural networks (GNNs). Jin et al. [28] propose a spatio-temporal graph neural network (STGNN) that enables the extraction of complex spatio-temporal dependencies by integrating GNNs and various temporal learning methods. Lam et al. [25] similarly take advantage of GNNs' ability to model arbitrary sparse interactions in the internal multi-mesh representation of their global weather forecasting system, GraphCast. Their system has homogeneous spatial resolution over the globe and allows for long-range interactions within few information aggregation steps. The system works by mapping input data from the original spatial grid onto learned features on the multi-mesh representation. Subsequently, a 16-layer GNN performs information aggregation. A decoder then maps the final representation back to the original spatial grid. The model took a total of three weeks to train using 32 Cloud TPU v4 devices with batch parallelism.

1.3.2 Spatio-Temporal Indicators

In most domains, applying domain-specific knowledge to design input features can improve model performance. In computer vision, for instance, calculating features like texture descriptors or edges provides additional information to the neural network. This ability to improve models through incorporating external knowledge about the problem has spurred research into identifying spatio-temporal indicators that can achieve this effect for spatio-temporal models.

The use of indicators to boost model performance is a commonly applied practice in time series forecasting tasks. While the naive approach to time series forecasting – known as time-delay embedding – involves simply transforming the original problem into a multiple regression task where the target variable is the future value of the series and the predictors are previous past values of the series, indicators can also be leveraged to add further information to the model [7]. For example, in financial forecasting, technical indicators that summarise key properties of the time series such as tendency, acceleration, and momentum can be incorporated into the multiple regression problem as additional predictors.

Extending the idea of temporal indicators to the spatio-temporal context necessitates the integration of both spatial and temporal information. The most basic implementation of this concept is space-time autoregression (STAR), discussed in the previous section [10]. Further work by authors such as Pace et al. [29] have validated the utility of weighted autoregressive spatio-temporal information for improving prediction accuracy on real-world data.

Other authors have looked beyond the STAR model to incorporate domain-specific relationships as spatio-temporal indicators. Tang, Yang, and Yang [30] propose an LSTM network that uses a combination of spatial and temporal features stored as a matrix to produce short-term forecasts for rail transit use. Specifically, they build predictor matrices out of domain-relevant information such as the time it takes passengers to travel between stations. Likewise, Draya, Legenrea, and Peres-Neto develop a method they call principal coordinate analysis of neighbour matrices (PCNM), which consists of diagonalising a spatial weighting matrix then extracting the eigenvectors that maximize Moran's index of autocorrelation [31].

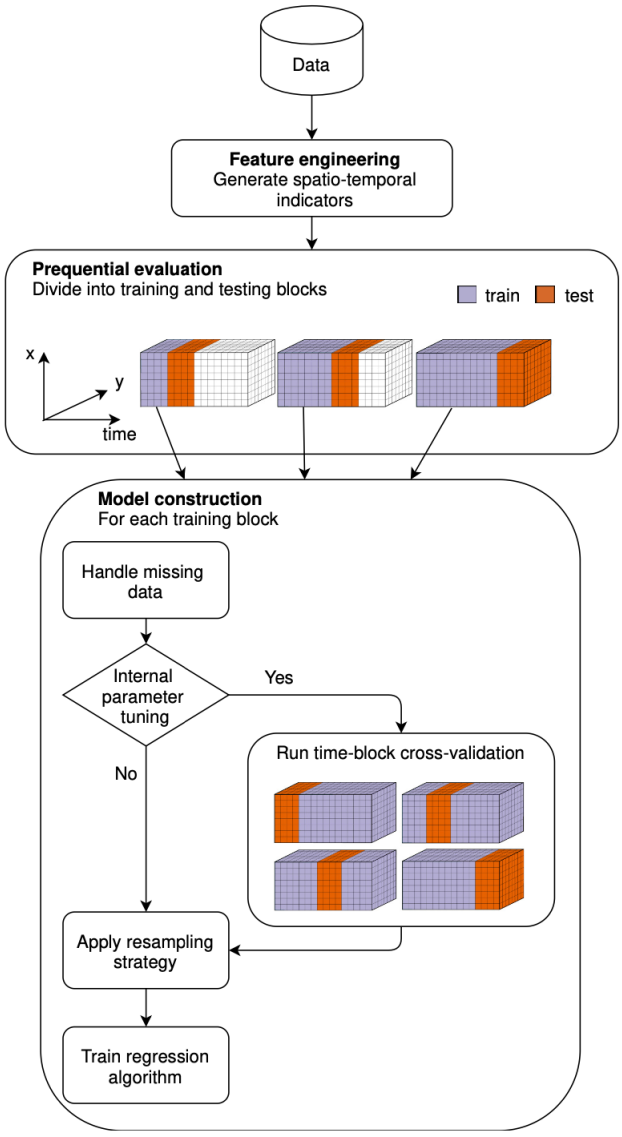


Figure 1.6: Workflow for forecasting with spatio-temporal indicators [5].

Ohashi and Torgo [7] produce a definition for spatio-temporal neighbourhoods from which relevant indicators can be extracted, making it possible to extend the time-embedding methods prevalent in time series forecasting to spatio-temporal problems.

These spatio-temporal neighbourhoods enable the specification of a variety of spatio-temporal relationships that can be specified by adjusting a relative weight given to spatial versus temporally close observations – or by adjusting the size of the neighbourhood itself.

These spatio-temporal indicators are successfully applied by Ohashi and Torgo [7] to improve wind speed predictions and by Lovisotto et al. [32] to predict spatial trends in internet usage. Both authors show that predictions which incorporate indicators from spatio-temporal neighbourhoods outperform baseline models that only contain same-location temporally lagged terms.

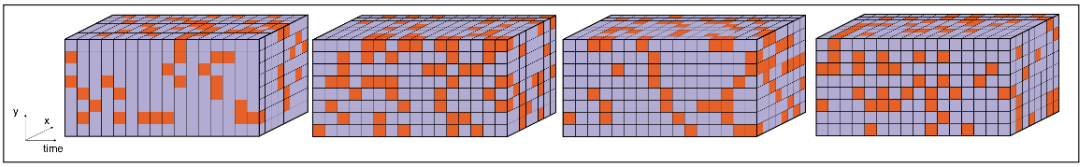
Oliveria, Torgo, and Costa [33] further build on their neighbourhood indicator method by taking the relative direction of each neighbour into consideration to capture effects of dominant winds when predicting the wildfire spread. Oliveria [5] also develops a new formulation for spatio-temporal neighbourhoods, specified in equations 1.5 and 1.6, that allows features to be extracted from more spatially and temporally distant variables of interest, modelling spatio-temporal problems that involve delays between observations.

The literature on spatio-temporal indicators illustrates the clear advantages of applying these indicators to spatio-temporal forecasts. One of the most salient advantages to this approach is that is it a pre-processing step and hence can be applied to any model that treats the problem as a multiple regression task. As the field continues to evolve, the integration of spatial and temporal information into indicators remains a compelling strategy for enhancing predictive accuracy across diverse domains.

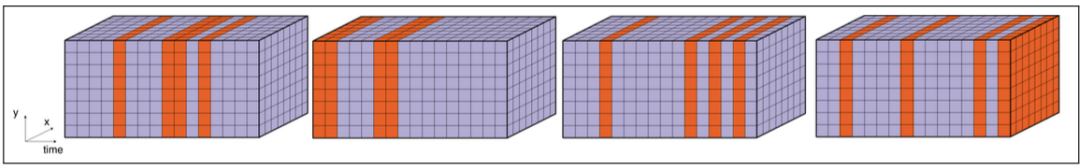
1.3.3 Feature Selection Methods

Guyon and Elisseeff [34] provide a comprehensive overview of feature selection methods, presenting definitions for various aspects including feature construction, ranking, multivariate selection, search methods, and validation techniques. They divide feature selection methods into three main categories: filter, wrapper, and embedded methods. Filter methods are often the most computationally efficient method, and are based on statistical measures of information which decouple attribute selection from the learning algorithm. Wrapper methods, though computationally expensive, leverage predictive accuracy to determine feature quality, often leading to superior combinations. Embedded methods simultaneously perform model fitting and feature selection, balancing efficiency and effectiveness.

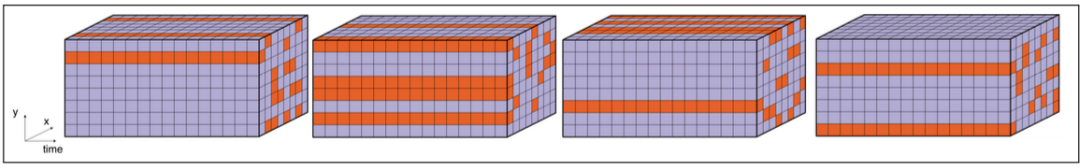
One method of feature selection that selects variables of interest based on relative importance scores from a random forest model has gained attention in recent years [35]. As a filter method that separates the selection process from the learning algorithm, this process if favourable for use in deep learning models that are computationally expensive to train. Genuer, Poggi, and Tuleau-Malo offer insights into the behavior of variable importance indices based on random forests by proposing a stepwise variable introduction strategy that combines ranking of explanatory variables using random forest scores of importance. This approach enhances the interpretability of feature importance and contributes to effective variable selection.



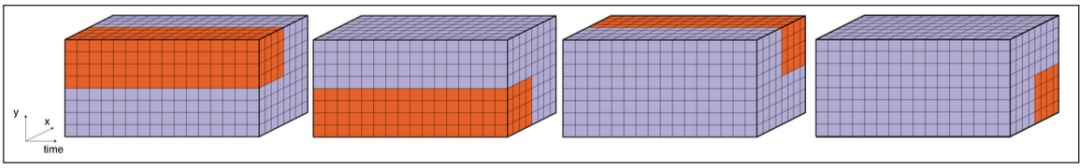
(a) Standard CV (CV)



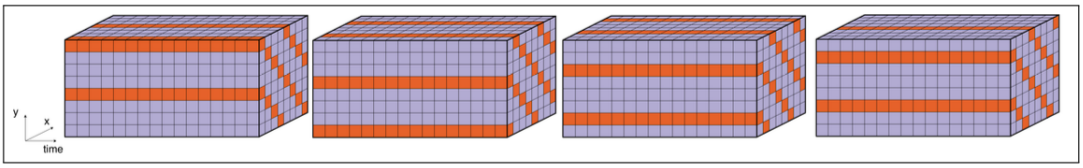
(b) Time-sliced CV (CV-Tsl)



(c) Spatial-block CV (CV-Sb)



(d) Contiguous spatial-block CV (CV-Sb-cont)



(e) Systematic spatial-block CV (CV-Sb-sys)

Figure 1.7: Block cross-validation methods; hold-out data is orange [5].

1.3.4 Spatio-Temporal Model Performance Evaluation

In the context of spatio-temporal forecasting, addressing the inherent autocorrelation challenges is imperative during model evaluation. Oliveira, Torgo, and Costa [33] underscore the complexities introduced by the interdependence among observations in spatio-temporal datasets. To overcome these challenges, they explore diverse a variety of cross-validation (CV) and out-of-sample (OOS) procedures. Through empirical analysis involving artificial and real-world spatio-temporal data sets, they demonstrate that CV fundamentally underestimates error for spatio-temporal models. To remedy this, the authors suggest applying CV methods that using data that has been blocked in both space and time for evaluation. This insight underscores the critical consideration of temporal and spatial dependencies in designing robust evaluation methodologies for spatio-temporal forecasting models.

1.4 Project Objectives

1.4.1 Research Goal

This dissertation further develops the work of Ohashi and Torgo [7] by proposing methodology to optimise the dimensions of the spatio-temporal neighbourhood from which past observations are extracted. This enables spatio-temporal indicators to be applied to a wide variety of data sets across different domains. Further, as the indicators presented by Ohashi and Torgo are model-agnostic and can be applied in the pre-processing stage of any spatio-temporal forecasting model, the proposed methodology has the potential benefit of improving spatio-temporal forecasting more broadly by enabling the optimal extraction of useful spatio-temporal features to refine the predictions of existing models.

Finally, the identification of an optimal spatio-temporal neighbourhood has the added benefit of providing useful information to domain specialists by revealing the spatio-temporal structure of the data. For example, human trafficking researchers may be interested in whether or not global trafficking flows can be modelled with a reverse cone-shaped spatio-temporal neighbourhood – and if so, what spatio-temporal distance exists between reported trafficking incidences. Identifying optimal neighbourhoods in this manner has the added benefit of potentially limiting the amount of data that needs to be collected to produce a useful forecast or make it possible to increase the forecast lead time using a smaller amount of data.

1.4.2 Hypothesis

It is hypothesized that forecasts performed with indicators extracted from optimally defined spatio-temporal neighbourhoods will perform significantly better than forecasts with neighbourhoods that are chosen uniformly as a function of the data set time intervals and number of locations.

Chapter 2

Methodology

2.1 Sample Data

Three data sets are used to illustrate the impact of extracting indicators from optimised spatio-temporal neighbourhoods. The data sets cover a range of domains, spatial data types, and temporal frequencies. The goal of this diversity is to illustrate that extracting indicators from an optimised spatio-temporal neighbourhood cone can improve forecast skill for a wide variety of domains.

Table 2.1: Overview of sample data sets

Source	**Type**	**Variables**	**Freq.**	**# Time**	**# Loc**	**Obs.**	**Avail.**
CAMELS-GB	Point	Catchment discharge Temperature Precipitation Pot. evotranspiration Shortwave radation Wind speed Humidity	Daily	16,436	671	11,028,556	10%
HadUK Grid	Raster	Air temperature Precipitation Sunshine duration Wind speed Air pressure Relative humidity Days of ground frost Days of snow cover	Monthly	1,656	1,305,000	15,660,000	100%
CTDC	Network	Trafficking victims Population HDI HDI female GNI per capita Inequality coefficient Gender inequality Rule of law Ongoing conflict	Annually	21	142	2,840	100%

2.1.1 CAMELS-GB

Catchment Attributes and MEteorology for Large-sample Studies, Great Britain (CAMELS-GB) is a large-sample catchment hydrology data set [36]. The data collates river flows, catchment attributes and catchment boundaries from the UK National River Flow Archive together with a variety of meteorological time series and catchment attributes for 671 catchments. Daily time series data is available for between October 1, 1970 to September 30, 2015. Data is collected from gauges placed at catchment outlets – the area of from which water drains before flowing into streams, rivers, and lakes – across the country which measure the quantity of runoff discharged from the land surface and subsurface as the result of a rainfall event within the catchment. The daily catchment discharge – or volumetric flow rate – is measured in by these gauges in cubic metres per second.

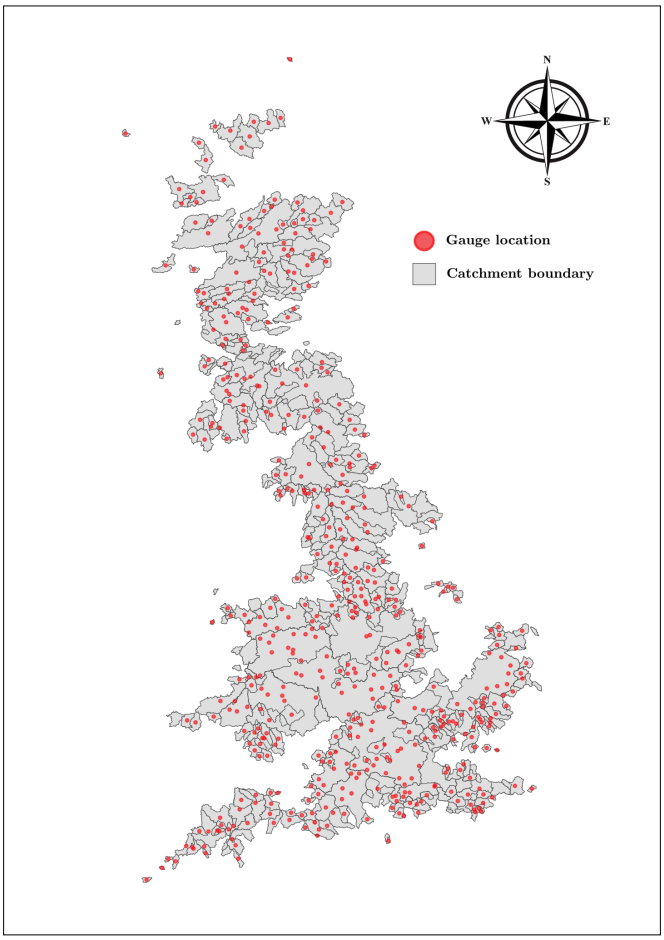


Figure 2.1: Locations of CAMELS-GB gauges and catchment areas.

For the purpose of the experiment, discharge volume is treated as the target variable, with daily temperature, precipitation, streamflow precipitation elasticity, potential evotranspiration, shortwave radiation, wind speed, and humidity as predictors. To reduce complexities resulting for seasonal variations, the output variable was transformed from the absolute discharge volume to the percent difference in discharge volume from the historic daily average for that catchment. The same is done for all

time series predictors.

To address the substantive missing data (only 10% of possible output values is available), the training data is restricted to dates after the year 2000, which has substantially more data availability. Remaining missing data is imputed using the values from the previous day and all imputations are flagged. Any gauge with more than two consecutive days of missing data is excluded from the analysis, restricting the sample to 467 catchments.

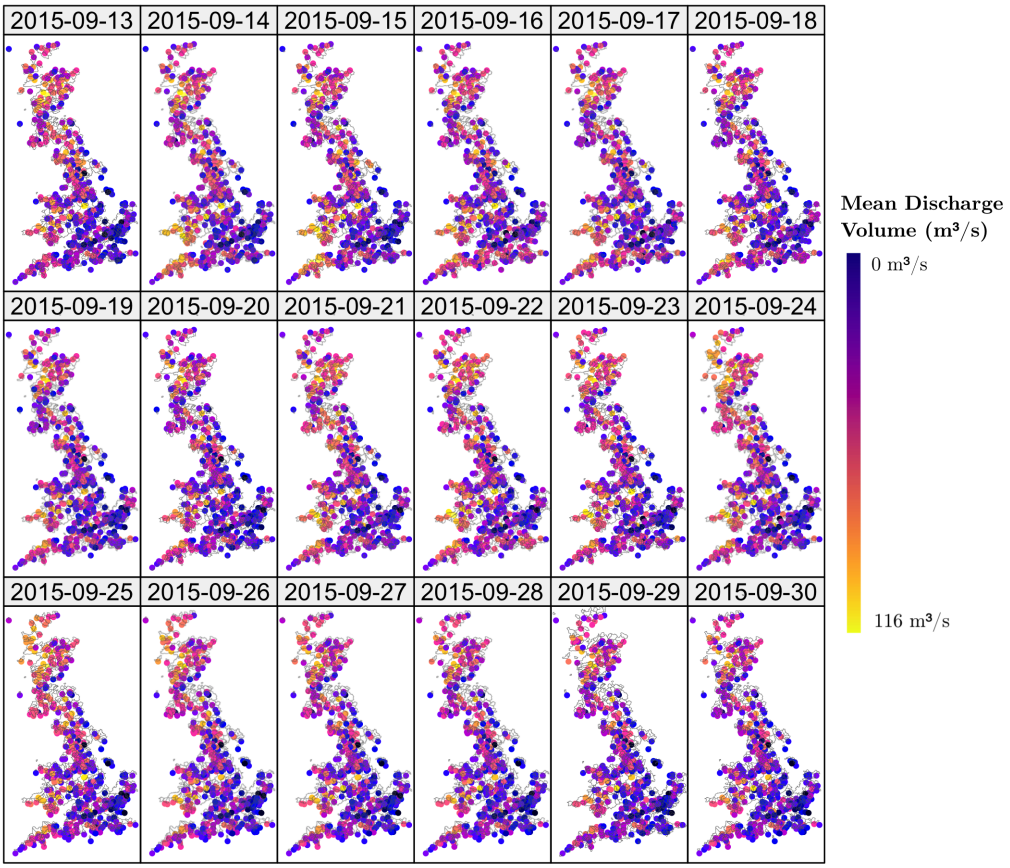


Figure 2.2: Spatial time series for a selection of CAMELS-GB data.

2.1.2 HadUK-Grid

HadUK-Grid, produced by the Met Office Hadley Centre for Climate Science and Services and published by the Centre for Environmental Data Analysis, is a collection of gridded climate variables interpolated from meteorological station data to form a uniform grid with 1km resolution [37]. The data covers the period between 1884 and 2021 and is produced for daily, monthly, seasonal and annual timescales, though for the purposes of the experiment, only the monthly time scale was used.

Monthly average temperature was modelled as the output variable, with average precipitation, average sunshine duration, average wind speed, average air pressure, average relative humidity, total days of ground frost, and total days of snow cover chosen as predictors. These variables were chosen for either their direct impact on air

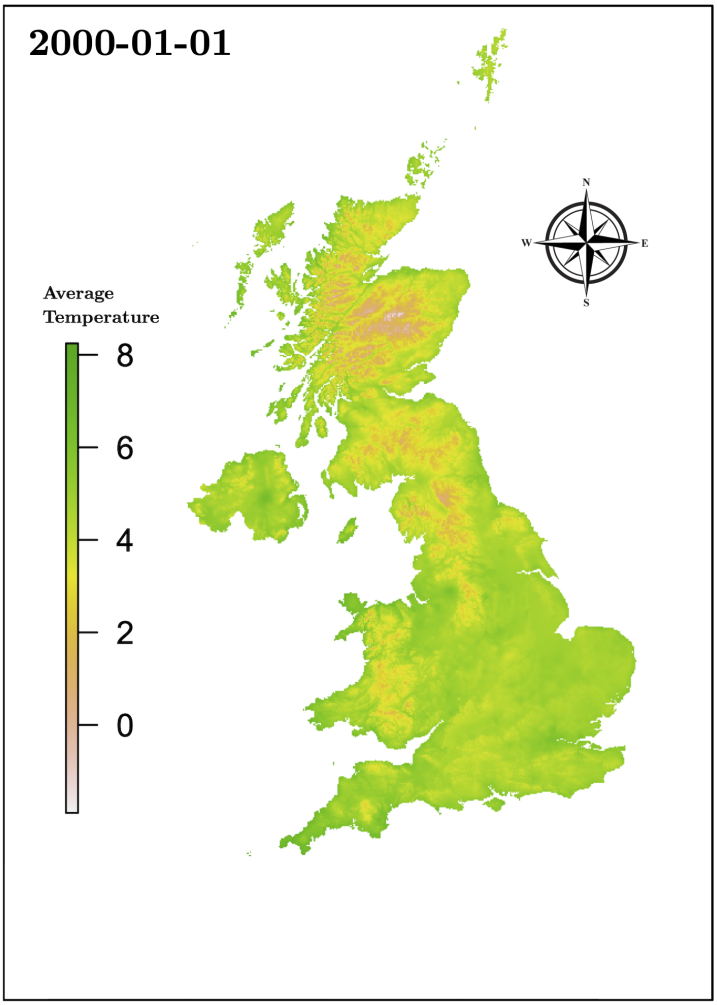


Figure 2.3: Monthly average temperature for each 1km cell in the HadUK Grid.

temperature or their potential indirect impact on air temperature through feedback effects. Similar to the CAMELS-GB data, all variables are transformed from the absolute values to the percent difference in value from the historic monthly average for that 1km cell. In addition to removing variability from physical features not accounted for by the predictors and controlling for seasonality, this transformation has the added benefit of normalising the predictor values.

To simplify the model, the training data is restricted to dates after the year 2000. As the HadUK-Grid data is already interpolated in order to generate grid-level meteorological data, there is no missing data or need for further imputations.

2.1.3 CTDC Global Synthetic Dataset

The Counter Trafficking Data Collaborative (CTDC) Global Synthetic Dataset combines data from human trafficking organisations to provide individual-level microdata on international human trafficking cases [38]. The data contain details on socio-demographic profile of victims, trafficking processes, and exploitation type.

To avoid identification, a synthetic data generation pipeline developed by Microsoft Research in collaboration with the International Organization for Migration is applied. Essentially, the process identifies unique entries that have fewer than k values and splits the original entries into multiple entries where different features are masked for each new entry. This process preserves the original identifiable traits, but prevents rare attributes from being linked together [39]. It is an alternative to k-anonymisation, which simply removes unique entries with fewer than k values. For the CTDC Global Synthetic Dataset, $k = 10$.

For the purposes of the experiment, each country is treated as a vertex in a network and the flow of refugees between countries is treated as a directed edge. Self-loops are allowed to account for any domestic trafficking cases. Spatial distance is defined as the normalised weighted connection between countries using previous-year trafficking flows, with nth degree neighbours calculated using the nth exponentiated past year normalised weighted adjacency matrix, which represents the cumulative effect of n-step connections, considering both the weights of the edges and the normalisation. The total number of reported trafficking victims is treated as the output, with country population, Human Development Index (HDI), female-only HDI, Gross National Income (GNI) per capita, coefficient of inequality, gender equality index, rule of law index, and an ongoing conflict indicator are used as predictors.

For anonymisation reasons, the CTDC data has a large number of entries that have the year or country of origin variables masked. These entries are unusable for the purposes of the analysis and are thus removed. However, the synthetic generation process ensures that the total number of entries for a given feature remain, so there is no need for further imputation on the output data. Missing predictor values are imputed as the most recently available past value for that country.

As the data is reported on an annual basis, no seasonality adjustment is required. For normalisation purposes, all non-index variables are treated as per capita values and all index variables are treated as a ratio of the maximum possible index score.

2.2 Spatio-Temporal Indicator Extraction

Spatio-temporal indicators are extracted for the outcome variable and all predictors using the definitions provided by Ohashi and Torgo [7]. These indicators, calculating using features associated with values in the chosen spatio-temporal neighbourhood, are the spatio-temporal typical value, the spatio-temporal spread, and the spatio-temporal tendency. Defined respectively as:

$$\bar{x}(\mathcal{N}_x^\beta) = \frac{1}{|\mathcal{N}_x^\beta|} \sum_{x \in \mathcal{N}_x^\beta} x, \tag{2.1}$$

$$\sigma_x(\mathcal{N}_x^\beta) = \sqrt{\frac{1}{|\mathcal{N}_x^\beta|} \sum_{x \in \mathcal{N}_x^\beta} (x - \bar{x}(\mathcal{N}_x^\beta))^2}, \tag{2.2}$$

$$\bar{X}_{\beta_1,\beta_2} = \frac{\bar{x}(\mathcal{N}_x^{\beta_1})}{\bar{x}(\mathcal{N}_x^{\beta_2})}, \tag{2.3}$$

where x is a variable of interest, $\mathcal{N}_x^{\beta}$ is the spatio-temporal neighbourhood defined in equation 1.4, and β is the spatio-temporal neighbourhood size specified in equation 1.2, with $\beta_1 < \beta_2$. The typical value and tendency indicators are also calculated using spatio-temporal weights, where the variable of interest is multiplied by the inverse of the spatio-temporal distance calculated in equation 1.2 as follows:

$$\tilde{x}(\mathcal{N}_x^{\beta}) = \frac{1}{|\mathcal{N}_x^{\beta}|} \sum_{x \in \mathcal{N}_x^{\beta}} x \cdot w_{x,A}, \tag{2.4}$$

$$\tilde{X}_{\beta_1,\beta_2} = \frac{\tilde{x}(\mathcal{N}_x^{\beta_1})}{\tilde{x}(\mathcal{N}_x^{\beta_2})}, \tag{2.5}$$

where $w_{x,A} = \frac{1}{D_A}$, the inverse spatio-temporal distance with D_A defined as in equation 1.2. The weights are normalised to satisfy $\sum_{x \in \mathcal{N}_x^{\beta}} w_{x,A} = 1$.

All indicators all also calculated for the aforementioned "reverse cone" paradigm, to account for relationships that involve a delay in spatio-temporal autocorrelation, using the temporal distance defined in equation 1.5 such that the reversed spatio temporal neighbourhood is calculated as:

$$\mathcal{N}_x^{\beta\prime} = \{A \in \mathcal{D} : D_A^{'} \leq \beta\}. \tag{2.6}$$

Spatio-temporal tendency is not calculated for reversed neighbourhoods as the spatio-temporal distance is not affected by β, as shown in equation 1.6.

Note that the temporal distances are calculated using equation 1.3 for the "forward cone" paradigm and equation 1.5 for the "reverse cone" paradigm. For the first two data sets, spatial distance is calculated using great circle distance, or the arc length between two points on a circumference passing through the center of an ellipsoid. For the CTDC Global Synthetic Dataset, spatial distance is calculated as the normalised weighted connectione between countries using previous-year trafficking flows, where nth degree neighbours are calculated with the nth exponentiated past year normalised weighted adjacenty matrix. In all cases, spatial distance is normalised to be a value between zero and one using min-max normalisation.

Once these indicators are calculated, they are simply incorporated into the multiple regression problem to forecast the value of the target time series k time steps ahead at location l_j, formulated as:

$$
\begin{aligned}
y_{i,j+k} = f(&x_{i,j}, x_{i,j-1}, \ldots, x_{i,j-m}, \\
&\bar{x}(\mathcal{N}_x^{k_1}), \bar{x}(\mathcal{N}_O^{k_2}), ..., \bar{x}(\mathcal{N}_x^{k_n}), \bar{X}_{k_1,k_2}, \bar{X}_{k_2,k_3}, ..., \bar{X}_{k_{n-1},k_n}, \\
&\tilde{x}(\mathcal{N}_x^{k_1}), \tilde{x}(\mathcal{N}_O^{k_2}), ..., \tilde{x}(\mathcal{N}_x^{k_n}), \tilde{X}_{k_1,k_2}, \tilde{X}_O^{k_2,k_3}, ..., \tilde{X}_{k_{n-1},k_n}, \\
&\sigma_x(\mathcal{N}_x^{k_1}), \sigma_x(\mathcal{N}_x^{k_2}), ..., \sigma_x(\mathcal{N}_x^{k_n}), \\
&\bar{x}(\mathcal{N}_x^{k_1\prime}), \bar{x}(\mathcal{N}_x^{k_2\prime}), ..., \bar{x}(\mathcal{N}_x^{k_n\prime}), \\
&\tilde{x}(\mathcal{N}_x^{k_1\prime}), \tilde{x}(\mathcal{N}_O^{k_2\prime}), ..., \tilde{x}(\mathcal{N}_x^{k_n\prime}), \\
&\sigma_x(\mathcal{N}_x^{k_1\prime}), \sigma_x(\mathcal{N}_x^{k_2\prime}), ..., \sigma_x(\mathcal{N}_x^{k_n\prime})),
\end{aligned}
\tag{2.7}
$$

where $f()$ is the unknown regression function used to model the training data $\mathcal{D}$, m is the number of same-location temporally lagged terms used, k_i are different neighbourhood sizes, and n is the total number of neighbourhoods selected.

2.3 Spatio-Temporal Neighbourhood Optimisation

Optimal spatio-temporal neighbourhoods are identified by performing a constrained search over a sample of plausible values for α and β, as well as a search over the same set of variables under the "reverse cone" paradigm. The resulting search space produces 99 unique neighbourhoods orientations for each time stamp, as reversed neighbourhoods are not affected by changes to the parameter β:

Table 2.3: Spatio-temporal neighbourhood optimisation search space

Parameter	**Search Space**
Neighbourhood type	{cone, reversed cone}
Dimension weight α	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}
Neighbourhood radius β	{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.01}

Note that increasing α increases the weight on spatially close autoregressive data while decreasing α increases the weight on temporally close data. Increasing β expands the overall number of spatio-temporal neighbourhood values considered when calculating the spatio-temporal indicators. Reversing the neighbourhood cone includes more data from points that are further away spatially and temporally, accounting for effects that might take time to travel from one location to another.

A sample of time periods from the training data is chosen to perform the search (though all training data could be used if computational resources allow). Typical value indicators are extracted using this sample for each of the search space neighbourhoods for all variables in the dataset – i.e. both the outcome and predictor variables. All spatio-temporal features ignore missing values, which occur at the earliest date in the available data due to insufficient past values, in their calculations. A random forest regression model, implemented in the R package `randomForest` by

Liaw and Wiener [40], is then trained the resulting data, where the target variable of interest is treated as the outcome variable and the extracted spatio-temporal typical values are treated as the predictors.

Feature importance for this model, defined by the increase in node purity, is then extracted. The implementation in R is based on mean squared error reduction:

$$\text{MSE reduction} = \frac{n_{\text{left}}}{n_{\text{total}}} \cdot \text{MSE}_{\text{left}} + \frac{n_{\text{right}}}{n_{\text{total}}} \cdot \text{MSE}_{\text{right}} - \text{MSE}_{\text{parent}}, \tag{2.8}$$

where n is the number of samples in the specified node after the decision tree split and MSE is the mean squared error of the specified node. The mean squared error reduction indicates how much the variance in the response variable is reduced after a split occurs in a particular node of the decision tree.

The top feature identified by this process for each variable in the model is then selected to be used in the final models. In the case where data was drawn from a sample of time periods in the training data, the feature importance for different samples is calculated to ensure that this method was robust for different samples. Preliminary evaluation confirms that the most important spatio-temporal neighbourhood is consistent for different samples of the training data.

Once the optimised neighbourhood is identified for each variable using the sample data, the optimised neighbourhood indicators are extracted for the remainder of the training data. In addition to the top neighbourhood identified by the random forest feature importance, the surrounding $\beta^* \pm 0.01$ values are calculated for the identified optimal α^* value. For example, if the optimal neighbourhood was identified to have $\alpha^* = 0.5$ and $\beta^* = 0.03$, $\beta^*_- = 0.02$ and $\beta^*_+ = 0.04$ are also calculated for the final model. When β^* is a boundary value, the next two highest or lowest values of β are used. These additional neighbourhood radii make it possible to calculate the spatio-temporal tendency, as specified in equation 2.3.

Identifying the optimal spatio-temporal neighbourhood parameters for each variable enables the extraction of the spatial and temporal distances by which the outcome variable of interest at the origin point is most influenced, D_A^{S*} and D_A^{T*}, by setting each variable equal to zero in equation 1.2. By re-normalising the spatial and temporal distance equations, it is possible to transform the identified α^* and β^* values into the original units in which distance and time were measured in the data set.

Setting $D_A^T = 0$ and substituting the min-max normalisation equation used to calculate the adjusted spatial distance calculates the optimal spatial lag:

$$\begin{aligned} \alpha^* \cdot D_A^{S*} + (1-\alpha^*)_A^T &\leq \beta^* \\ \alpha^* \cdot D_A^{S*} &\leq \beta^* \\ \frac{d^* - d_{\min}}{d_{\max} - d_{\min}} &\leq \frac{\beta^*}{\alpha^*} \\ d^* &\leq d_{\max} \cdot \frac{\beta^*}{\alpha^*}, \end{aligned} \tag{2.9}$$

where d^* is the optimal spatial distance in the original spatial units from the data set and $d_{\min}$ is equal to zero as past measurements from the same location are included in the spatio-temporal neighbourhood. Likewise, setting $D_A^S = 0$ and substituting the min-max normalisation equation used to calculate the adjusted temporal distance calculates the optimal temporal lag:

$$\begin{aligned} \alpha^* \cdot D_A^S + (1-\alpha^*) \cdot D_A^{T*} &\leq \beta^* \\ (1-\alpha^*) \cdot D_A^{T*} &\leq \beta^* \\ \frac{t^*}{t_{\max} - t_{\min}} &\leq \frac{\beta^*}{1-\alpha^*} \\ t^* &\leq (t_{\max} - t_{\min}) \cdot \frac{\beta^*}{1-\alpha^*}, \end{aligned} \tag{2.10}$$

where a^* is the optimal distance in the original temporal units from the data set.

These calculations provide the dimensions of the optimal spatio-temporal neighbourhood cone, with d^* representing the spatial radius of influence and t^* representing the temporal length of influence. It can be seen that the same re-normalisation equations hold for the "reverse cone" paradigm.

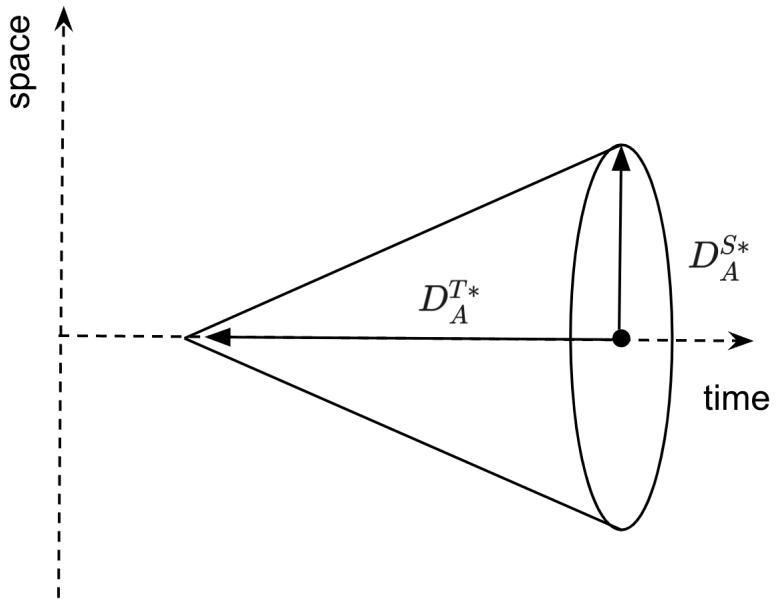


Figure 2.4: Dimensions of the optimal spatio-temporal neighbourhood.

Random forest feature importance is chosen as the optimisation method due to its established ability to handle high-dimensional data with small sample sizes while providing more robust feature subsets than single feature selection techniques [41]. This method also has the advantages of being able to handle multicollinearity well and providing highly interpretable feature selection results [42].

Ohashi and Torgo [7] use grid search methods to obtain better estimates over a small selection of hand-picked neighbourhoods. There method requires training the entire model on new sets of indicators, which would be computationally expensive for most state-of-the-art spatio-temporal forecasting models that rely on deep learning such as LSTM networks, making the technique inappropriate for a search over an extensive parameter space. As a filter method, applying random forest feature selection prior to model training ensures that the complexity of feature selection is independent of model choice.

2.4 Forecasting Models

To validate the impact of using indicators extracted from optimised spatio-temporal neighbourhood, several models were built using the indicators. The first two models, the spatio-temporal autoregression and kriging models, respectively leverage the autoregressive properties and covariance of the data to produce traditional estimates of spatio-temporal values. The remaining five models apply the spatio-temporal indicator method by transforming the forecast problem into a multi-regression task where the spatio-temporal indicators are used as predictors, as in equation 1.12. This configuration is then compared to a model where spatio-temporal neighbourhood indicators are extracted without optimising the neighbourhood parameters (using a naive choice of $\alpha = 0.5$ and $\beta = 0.01$) and a model where only same-location past values are considered, as such:

$$y_{i,j+k} = f(x_{i,j}, x_{i,j-1}, \ldots, x_{i,j-m}) \tag{2.11}$$

These three model configurations, applied to the sample data sets, are intended to validate the utility of spatio-temporal neighbourhood optimisation for indicator extraction and to reaffirm the utility of spatio-temporal indicators for spatio-temporal forecasting in general. They are compared using the following models:

Space-Time Autoregression A method based on the R package `gstar` by Ruchjana, Lopuhaä, and Borovkova [11] that predicts spatio-temporal values with autoregressive values and lag differencing across space and time, using a normalised spatial distance matrix. For the purposes of the experiment, this model is used as a baseline to compare a spatio-temporal autoregressive model with models that reframe the forecasting problem as a multiple-regression task.

Kriging A method based on the R package `gstat` by Gräler, Pebesma and Heuvelink [16] that extends spatial kriging to a spatio-temporal context by developing covariance models that link spatial and temporal covariance. Specifically, through separable, product-sum, metric and sum-metric spatio-temporal covariance functions. These functions are used to find a spatio-temporal variogram that is then used to interpolate values across space and time. For the purposes of the experiment, this model is used as a baseline to compare a spatio-temporal covariance model with models that reframe the forecasting problem as a multiple-regression task.

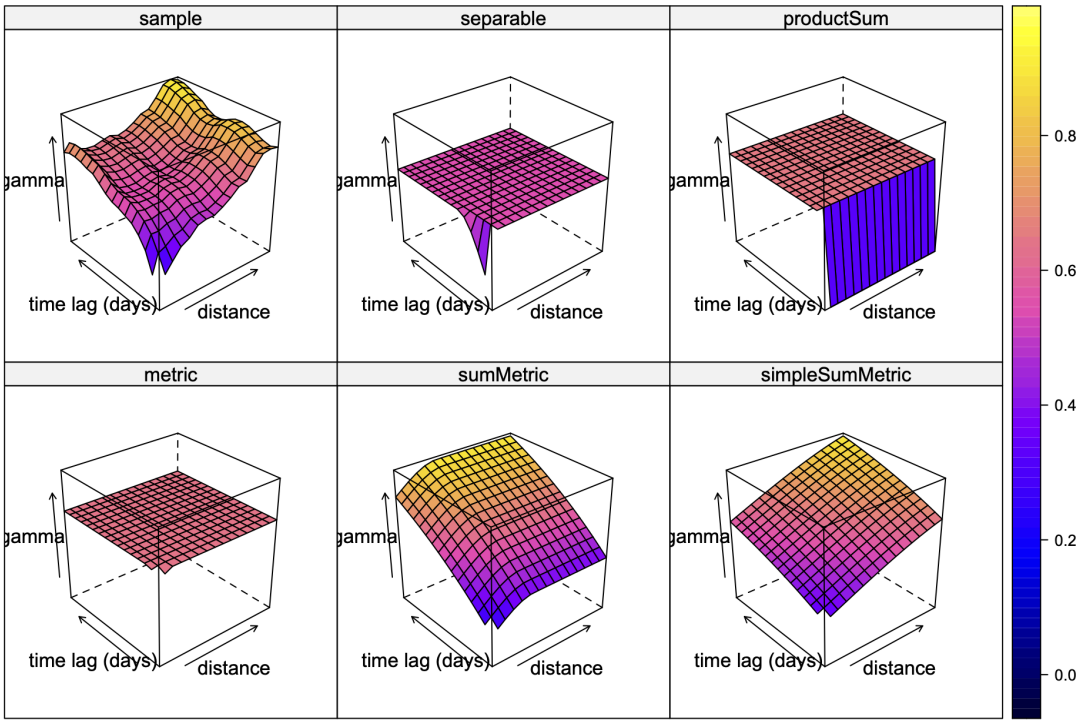


Figure 2.5: A comparison of spatio-temporal variograms for CAMELS-GB data.

Linear Regression A basic method used to model a linear relationship between a response value and a set of predictors, implemented the R package `stats` [43]. For the purposes of the experiment, this model is used to test the effect of optimising the spatio-temporal neighbourhood indicators on forecast skill by reframing the forecasting problem as a mutiple-regression task.

Support Vector Machines A method that finds a hyperplane to best fit data points in a continuous space [44]. Implemented with the R package `e1071` using a polynomial kernel [45]. For the purposes of the experiment, this model is used to test the effect of optimising the spatio-temporal neighbourhood indicators on forecast skill by reframing the forecasting problem as a mutiple-regression task.

Regression Trees A method that partitions the feature space into segments and predicts the target value by averaging the values in the corresponding segment [46]. Implemented in the R package `rpart` by Therneau and Atkinson [47]. For the purposes of the experiment, this model is used to test the effect of optimising the spatio-temporal neighbourhood indicators on forecast skill by reframing the forecasting problem as a mutiple-regression task.

Random Forest A method that builds multiple decision trees and combines their predictions to improve model accuracy and reduce overfitting [46]. Implemented in the R package `randomForest` by Liaw and Wiener [40]. For the purposes of the experiment, this model is used to test the effect of optimising the spatio-temporal neighbourhood indicators on forecast skill by reframing

the forecasting problem as a mutiple-regression task.

LSTM Network A type of recurrent neural network designed to capture and model sequential patterns by effectively learning and managing long-range dependencies within sequential data [48]. Implemented in Python using Tensorflow [49]. For the purposes of the experiment, this model is used to test the effect of optimising the spatio-temporal neighbourhood indicators on forecast skill by reframing the forecasting problem as a mutiple-regression task.

2.5 Forecasting Process

Data is divided into k temporally continuous blocks for a given region. These blocks form the basis of the k-fold block cross validation used to evaluate the models. Models are trained on all data excluding the temporal block of interest, which is then used to evaluate the predictions. Forecasts are made for each k blocks.

A sliding window approach is employed to predict new values by using fixed-size window of past data to predict a certain number of future time steps. The window moves forward as the resulting prediction is used to compute new spatio-temporal indicators for each variable. The size of the window is equal to the chosen spatio-temporal neighbourhood for each variable.

Figure 2.6: Sliding window approach with k-fold temporal blocks.

For the purposes of this experiment, potential issues with detecting extreme or rare events as discussed by Oliveria et al. [50] are ignored. Though it should be noted that biased re-sampling strategies can be applied to improve extreme predictions. This potential future work is discussed in section 4.3.

2.6 Evaluation

Overall, three configurations are evaluated for each forecasting model:

1. A same-location-only autoregressive configuration
2. A naive spatio-temporal neighbourhood configuration
3. An optimised spatio-temporal neighbourhood configuration

Naive neighbourhoods are chosen using a normal cone orientation with $\alpha = 0.5$ and $\beta = 0.01$. The mean absolute error (MAE) of the prediction estimate is calculated for each of the k folds. The average MAE for each fold is calculated and compared between models. Significant differences between configurations are tested using the standard deviation of the MAE between each fold.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.12}$$

The three multiple regression models specified above are also compared to predictions made using space-time autoregression and spatio-temporal kriging to provide a baseline to compare the impact of re-framing the spatio-temporal forecasting problem as a multiple regression problem.

Chapter 3

Results

3.1 Model Comparison

The random forest feature selection process used to identify optimal spatio-temporal neighbourhoods makes it possible to calculate how well the out-of-bag predictions explain the target variance of the training set. In general, the optimised spatio-temporal neighbourhood configuration performed quite well compared to the naive spatio-temporal neighbourhood configuration in the feature selection models.

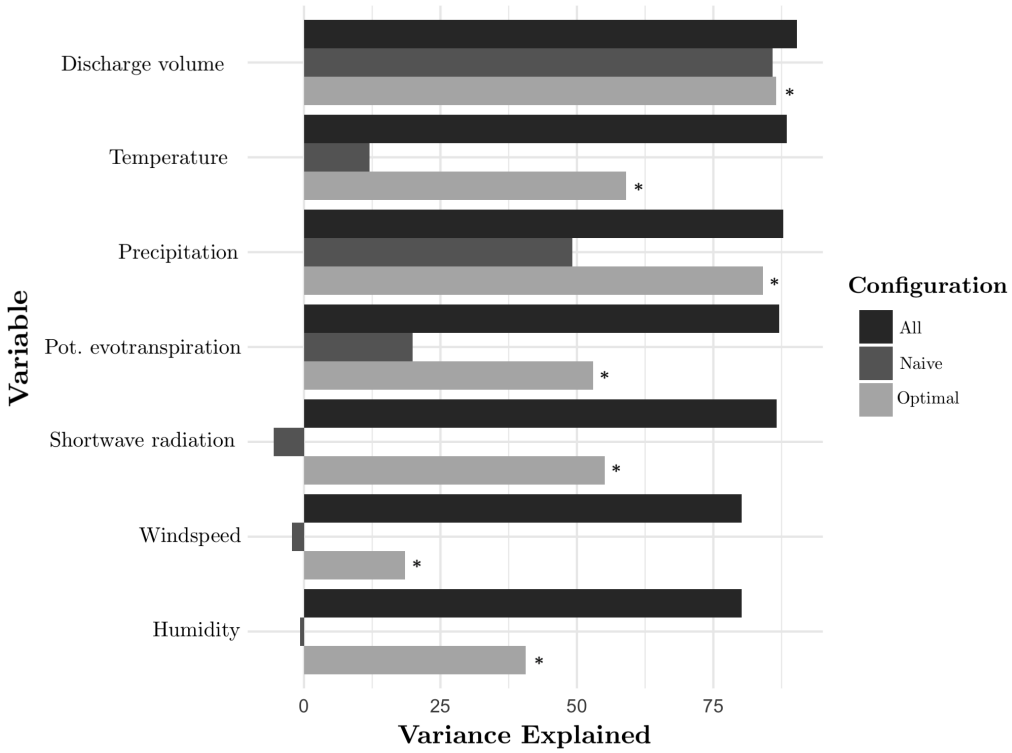


Figure 3.1: Variance explained by different spatio-temporal neighbourhood configurations in CAMELS-GB data. Instances where the optimal neighbourhood configuration outperforms the naive neighbourhood configuration are marked.

As can be seen in Figure 3.1, the random forest feature selection model built using only the spatio-temporal indicators from the optimal spatio-temporal neighbourhood consistently had a higher percentage of its out-of-bag variance explained that the random forest model built using the naive configuration of $\alpha = 0.5$ and $\beta = 0.01$. Indeed, the some some cases, the naive configuration had negative variance explained, indicating that choosing the wrong neighbourhoods can be worse than not using them any neighbourhood predictors at all. The performance of both configurations is compared to the feature selection model containing all of the spatio-temporal neighbourhood indicators in the search space, which should be expected to have the highest percentage of out-of-bag variance explained.

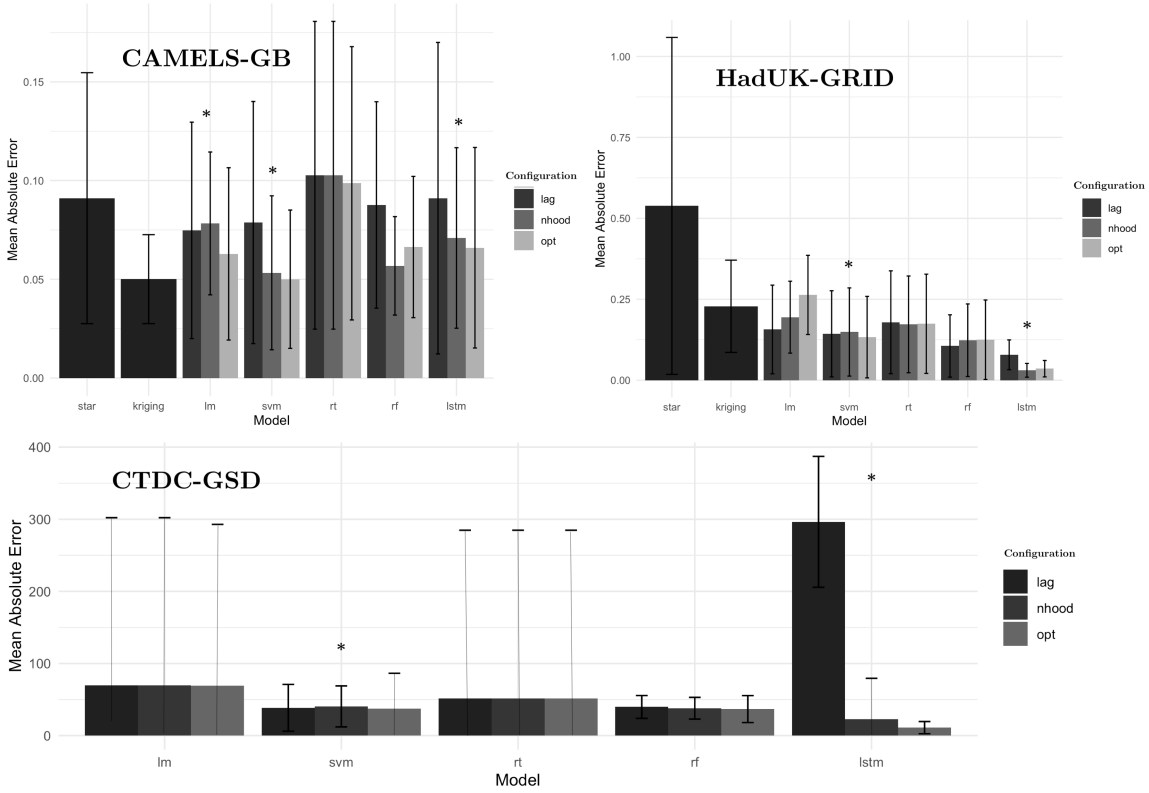


Figure 3.2: A comparison of prediction mean absolute error calculated using temporal block cross validation for the three configurations. Statistically significant differences between the optimal neighbourhood configuration model and the naive neighbourhood configuration model are marked.

Using the top neighbourhood parameters indicated by the selection process, the three configurations were compared for five different models. Using temporal block cross validation, the average mean absolute error was calculated from the k-fold blocks assessed using a sliding window predictive approach. The standard deviation of these error values was also calculated. Of the 15 total comparisons made between the specified configurations, the configuration using optimised spatio-temporal neighbourhoods outperformed the configuration using naive spatio-temporal neighbourhoods 11 times, meaning that the average mean absolute error for the k predictions made using the optimised spatio-temporal neighbourhood indicators was less than

the average mean absolute error for the k predictions made using the naive spatio-temporal neighbourhood indicators.

Only 7 of these 11 differences, however, were statistically significant, due to high variance in error rates for some compared models. This high error rate could likely be reduced under further fine-tuning of the chosen models. This includes model specific improvements, like ensuring that the correct basic function for the SVM model is chosen or increasing the number of trees for the random forest model. It might also be beneficial to perform further variable selection for each model individually to remove redundant lags and indicators that may increase prediction errors. Despite the high variance in prediction errors, the optimised configuration still performed better than the naive configuration on average. To illustrate this, the naive neighbourhood indicator configuration was only significantly better than the optimised configuration for 1 of the 15 total comparisons (specifically, the HadUK-Grid linear model). The optimised neighbourhood configuration also outperformed the lag-only model in 13 of the 15 total comparisons.

Other model properties beyond the prediction error provide further support for the use of optimised spatio-temporal neighbourhood indicator extraction. In most of the linear models tested, the adjusted R^2 values were substantially higher for the optimised configuration than the naive configuration, indicating that a greater percentage of variance was explained. Further, the percentage of out-of-bag variance explained by the random forest prediction models was also generally higher for the optimised configuration than for the naive configuration, similar to the feature selection models. These findings are not discussed in detail as they are not the focus of the dissertation, but may warrant consideration for future work.

3.2 Optimal Spatio-Temporal Neighbourhoods

The spatio-temporal neighbourhood optimisation process makes it possible to detect the spatial and temporal distances by which the outcome variable of interest is most affected by its spatio-temporal neighbours. Recall that increasing α increases the weight on spatially close autoregressive data while decreasing α increases the weight on temporally close data. Increasing β expands the overall number of spatio-temporal neighbourhood values considered when calculating the spatio-temporal indicators. Reversing the neighbourhood cone includes more data from points that are further away spatially and temporally, accounting for effects that might take time to travel from one location to another. Thus calculating the optimal spatio-temporal neighbourhood parameters reveals potentially domain-relevant information about the relationship between predictors and the outcome variable.

Figure 3.3 illustrates the optimal spatio-temporal neighbourhoods extracted for three variables in the CAMELS-GB data set, as indicated by increase in random forest node purity. As can be seen, the top neighbourhoods indicated by the selection model have similar values for α^* and β^*, and are consistently either all in the normal or reversed cone paradigm. Figure 3.3 illustrates that discharge volume is more related to past discharge volume values that are measured closer in time to

the prediction. Similarly, temperature values measured at a greater spatio-temporal distance from the catchment of interest are consistently more predictive of discharge volume – an example of the "reverse cone" paradigm that indicates a spatio-temporal delay in the affect of temperature on discharge volume. Finally, precipitation values are much more predictive of discharge volume when they are measured at closer geographic distances to the catchment of interest.

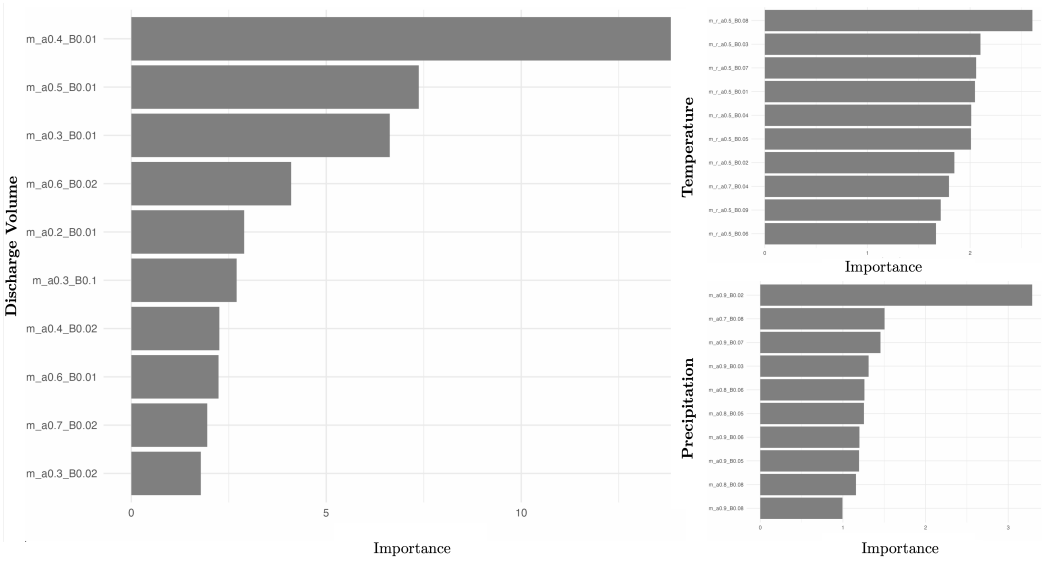


Figure 3.3: Highest increase in node purity for CAMELS-GB discharge volume, temperature, and precipitation random forest feature selection models

To ensure that the recommended optimised spatio-temporal neighbourhoods were stable, feature importance was also calculated for different random seeds and samples of dates from the training data. This process revealed that the illustrated rankings were highly consistent for all samples, with little to no variation among the rankings for different seedings and samples.

In addition to describing the general spatio-temporal relationships between predicors and the outcome variable of interest, the optimal spatio-temporal neighbourhood extraction process also makes it possible to quantify these relationships by extracting the optimal spatial and temporal relationship distance using equations 2.9 and 2.10. These values are only granular down to the values explored in the search space, so α^* values are in one tenth increments and values for β^* are in one one thousandth increments. In theory these values could be optimised further to provide a more granular understanding of spatio-temporal relationships in the data.

Table 3.1 provides an example of the optimal neighbourhood parameters extracted for the CAMELS-GB variables. Note that the large values for time in the reversed cone paradigm accord with the reversed cone definition, as the outcome is more closely predicted by values of this variable that are further away in both space and time. If these parameters were used in practice to improve models, a full temporal lag of the value presented would not strictly be needed, but instead features from

Table 3.1: Optimised spatio-temporal neighbourhoods for CAMELS-GB variables

Variable	α^*	β^*	**Reversed**	d^*	t^*
Catchment discharge	0.4	0.01	No	20.4km	96 days
Temperature	0.5	0.08	Yes	130.6km	920 days
Precipitation	0.9	0.02	No	18.1km	1,150 days
Pot. evotranspiration	0.3	0.09	Yes	244.9km	739 days
Shortwave radiation	0.9	0.10	No	90.7km	5,751 days
Wind speed	0.9	0.04	No	9.1km	575 days
Humidity	0.7	0.09	Yes	104.9km	1,725 days

as many past values as possible within the lag time would be used. These large temporal distances may indicate the presence of annual climatologist trends which require further exploration. Expanding upon the neighbourhood parameter search space can likely improve the granularity of these estimates.

In summary, the results indicate that indicators extracted from optimised spatio-temporal neighbourhoods can reduce prediction error rates and thus improve forecast skill compared to indicators extracted from naively selected spatio-temporal neighbourhoods. Results could be further improved by expanding the neighbourhood search space or further optimising the prediction models.

Chapter 4

Discussion

4.1 Main Contributions

Ohashi and Torgo [7] illustrate the utility of extracting spatio-temporal indicators for improving forecast skill. This dissertations expands upon their work by developing methodology for optimising the spatio-temporal neighbourhoods from which these indicators are extracted. It is shown that this optimisation process further improves forecast skill, providing a new tool for spatio-temporal forecasters.

The process for extracting spatio-temporal indicators, as well as the process for optimising these indicators, is a pre-processing step that is agnostic to model choice and domain. This means that the proposed methodology can be applied to improve forecast accuracy for existing models without altering the model development pipeline. In addition, the optimised dimensions for spatio-temporal neighbourhoods themselves provide information about spatio-temporal relationships that may have domain value. For example, dimensions that are optimised at smaller spacial and temporal distances can reveal the optimal amount of past data to include, potentially reducing computational time and data collection expenses. Conversely, dimensions that are optimised at larger spacial and temporal distances may reveal that expanding data collection improves forecast accuracy.

The optimisation method proposed in this dissertation has several advantages over alternative optimisation methods, such as grid search methods. A filter method for selecting the spatio-temporal neighbourhoods is proposed, maintaining consistent computational complexity regardless of model choice. The complexity of building a single decision tree is generally $O(N \cdot M \log M)$, where N is the number of samples and M is the number of features [46]. This is because at each node, the algorithm considers a subset of features to split the data and sorts the samples based on those features. As random forest algorithm trains multiple decision trees independently, the overall training complexity of a random forest with n trees is approximately $O(n \cdot N \cdot M \log M)$. Though random forests can become computationally intensive as these parameters increase, its use here as a pre-processing step means that the neighbourhood selection process is not affected by model choice. Further, the proposed methodology maintains a constant number of predictors and allows for

a sample of the training data to be used for feature selection. On the data sets examined, feature selection using the random forest models – after spatio-temporal indicators were extracted – was able to be conducted in seconds. Grid search methods, on the other hand, require retraining the final forecasting model under different parameter choices. For certain forecasting methods with potentially long training times, such as deep learning methods, it may not be feasible to have a large search space for optimising spatio-temporal neighbourhoods.

In addition to computational advantages, random forest feature selection provides interpretable results than enable researchers to verify consistency in spatio-temporal neighbourhood selection. As can be seen by the optimal neighbourhoods chosen for the sample data, the most important neighbourhood parameters – as determined by increase in node purity – are consistent for each variable. This indicates that the feature selection models are detecting true underlying spatio-temporal relationships between variables and not just random noise.

4.2 Limitations

Spatio-temporal forecasting using past variables as indicators is inherently limited by its autoregressive nature. Using past values of the target variable itself as a predictor, including as a data point for calculating spatio-temporal indicators, permits errors to arise from inaccuracies in the estimation of initial states. For long term forecasts, these errors may accumulate and seriously harm the accuracy of predictions. Models that are based on reliable, independent forecasts of predictor variables which are not themselves outputs of the forecasting process at hand do not have this issue. However, though autoregressive issues can pose challenges for distant forecasts, they can still be highly accurate for short-term predicitons.

Furthermore, even though the chosen mechanism of feature selection itself is not particularly computationally intensive, the process for extracting spatio-temporal indicators requires a significant number of repetitive calculations that may take a long time to perform. This is because different indicators must be calculated for each time period. Such calculations may not be feasible for data sets with high temporal resolution or a large number of predictor variables. Ideally, researchers should be able to identify a optimal spatio-temporal neighbourhoods based on a small sample of locations and time periods and apply the identified parameterisation to extract indicators for predictions. This reduces the calculation burden substantially compared to using the entirety of the training data for feature selection.

4.3 Next Steps

Additional work to develop spatio-temporal indicator extraction could focus on evaluating differences in accuracy for forecasts made using a single location or the entirety of available locations as training data. Though the later option increases computational complexity, it might also provide more data that can be used to improve forecast skill. It also would enable static data at the location-level to be

incorporated region-specific differences. In a similar vein, efforts can be taken to consider additional physical complexities within spatio-temporal data, such as directional relationships (ansiotropy) and differential impacts of location on the target variable – for example, how differences in height affects discharge volume.

There are also opportunities to consider alternative definitions for spatio-temporal neighbourhoods defined in equations 1.2 and 1.4, such as non-linear parameterisations or definitions that provide more weight to variables from certain spatial locations based on domain knowledge.

Further, it may be beneficial to explore alternative optimisation techniques such as multi-objective evolutionary algorithms or surrogate-assisted evolutionary algorithms, which may provide more efficient ways of calculating model-specific spatio-temporal neighbourhood optimisations [51].

In the models used for the experiments, no efforts were taken to address potential imbalances in the available data that might limit the models' ability to forecast extreme events, which is often a variable of interest in spatio-temporal forecasting. It might thus be worthwhile to apply Oliveira's [50] biased re-sampling methods to improve rare-event forecasting and determine whether this affects the forecast skill for the three explored conceptualisations.

Finally, efforts could be taken to improve computational efficiency of the implemented spatio-temporal neighbourhood optimisation method. Though all calculations in the experiments take, at maximum, only a few hours to perform optimisation and feature extraction using a single CPU, efforts could be made to speed up the process even further using GPUs and parallelisation.

4.4 Conclusion

This dissertation provides a novel, computationally efficient, and easily implementable method for improving spatio-temporal forecasts using indicators extracted from optimal spatio-temporal neighbourhoods. Further work should be undertaken to validate forecast skill improvements by applying the methodology to existing models.

Bibliography

[1] W. R. Tobler. "A computer movie simulating urban growth in the Detroit region." In: *Economic Geography* 46 (1970), pp. 234–240. DOI: https://doi.org/10.2307/143141.

[2] Lei Xu et al. "Spatiotemporal forecasting in earth system science: Methods, uncertainties, predictability and future directions." In: *Earth-Science Reviews* 222 (2021). DOI: https://doi.org/10.1016/j.earscirev.2021.103828.

[3] Yao Yao et al. "Spatiotemporal distribution of human trafficking in China and predicting the locations of missing persons." In: *Computers, Environment and Urban Systems* 85 (2021). DOI: https://doi.org/10.1016/j.compenvurbsys.2020.101567.

[4] Dhivya Bharathia, Lelitha Vanajakshia, and Shankar C. Subramanianb. "Spatio-temporal modelling and prediction of bus travel time using a higher-order traffic flow model." In: *Physica A: Statistical Mechanics and its Applications* 596 (2022). DOI: https://doi.org/10.1016/j.physa.2022.127086.

[5] Mariana Rafaela Figueiredo Ferreira de Oliveira. "Predictive Analytics for Spatio-Temporal Data." PhD thesis. Braga, Portugal: Universidades do Minho, Aveiro, e Porto, 2021.

[6] Zanetti Marcoab et al. "Spatio-temporal cross-validation to predict pluvial flood events in the Metropolitan City of Venice." In: *Journal of Hydrology* 612 (2022). DOI: https://doi.org/10.1016/j.jhydrol.2022.128150.

[7] Orlando Ohashi and Luís Torgo. "Wind speed forecasting using spatio-temporal indicators." In: *Dynamical Systems and Turbulence, Warwick 1980* 898 (2012), pp. 366–381. DOI: doi:10.3233/978-1-61499-098-7-975.

[8] Floris Takens. "Detecting strange attractors in turbulence." In: *Dynamical Systems and Turbulence, Warwick 1980* 898 (1980), pp. 366–381. DOI: https://doi.org/10.1007/BFb0091924.

[9] Louise Slater et al. "Hybrid forecasting: using statistics and machine learning to integrate predictions from dynamical models." 2022. DOI: https://doi.org/10.5194/hess-2022-334.

[10] Edoardo Otranto and Massimo Mucciardi. "A Flexible Specification of Space–Time AutoRegressive Models." In: *Centre for North South Economic Research* (2016).

[11] Budi Nurani Ruchjana, Svetlana A. Borovkova†, and H. P. Lopuhaa. "Least squares estimation of Generalized Space Time AutoRegressive (GSTAR) model and its properties." In: *AIP Conference Proceedings* 1450 (61 2012). DOI: doi:10.1063/1.4724118.

[12] Peter Congdon. "A spatio-temporal autoregressive model for monitoring and predicting COVID infection rates." In: *Journal of Geographical Systems* 24 (2022), pp. 583–610. DOI: https://doi.org/10.1007/s10109-021-00366-2.

[13] Fabio Sigrist, Hans R. Kunsch, and Werner A. Stahel. "An autoregressive spatio-temporal precipitation model." In: *Procedia Environmental Sciences* 3 (2011), pp. 2–7. DOI: doi:10.1016/j.proenv.2011.02.002.

[14] Danie G. Krige. "A Statistical Approaches to Some Basic Mine Valuation Problems on the Witwatersrand". In: *Journal of the Chemical, Metallurgical and Mining Society of South Africa* 52 (1951), pp. 119–139. DOI: doi:10.10520/AJA0038223X_4792.

[15] Georges Matheron. *Traité de géostatistique appliquée.* Éditions Technip, 1962.

[16] Benedikt Gräler, Edzer J. Pebesma, and Gerard B. M. Heuvelink. "Spatio-Temporal Interpolation using gstat". In: *R J.* 8 (2016), p. 204. URL: https://api.semanticscholar.org/CorpusID:195948536.

[17] Johan Lindström et al. "A Flexible Spatio-Temporal Model for Air Pollution with Spatial and Spatio-Temporal Covariates." In: *AIP Conference Proceedings* 21 (3 2014), pp. 411–433. DOI: doi:10.1007/s10651-013-0261-4.

[18] James Donnelly et al. "Gaussian process emulation of spatio-temporal outputs of a 2D inland flood model." In: *Water Research* 225 (2022). DOI: https://doi.org/10.1016/j.watres.2022.119100.

[19] Junpeng Zhang et al. "An efficient implementation for spatial–temporal Gaussian process regression and its applications." In: *Automatica* 147 (2023). DOI: https://doi.org/10.1016/j.automatica.2022.110679.

[20] Sella Nevo1a et al. "Flood forecasting with machine learning models in an operational framework." In: *Hydrology and Earth System Science* 26 (2022). DOI: https://doi.org/10.5194/hess-26-4013-2022.

[21] Tomislav Hengl et al. "Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables." In: *PeerJ* 6 (2018). DOI: doi:10.7717/peerj.5518.

[22] Hristos Tyralis, Georgia Papacharalampous, and Andreas Langousis. "A Brief Review of Random Forests for Water Scientists and Practitioners and Their Recent History in Water Resources." In: *Water* 11 (910 2019). DOI: https://doi.org/10.3390/w11050910.

[23] Zhigang Li, Margaret H. Dunham, and Yongqiao Xiao. "STIFF: A Forecasting Framework for SpatioTemporal Data." In: *Lecture Notes in Computer Science* 2797 (2002), pp. 183–198. DOI: https://doi.org/10.1007/978-3-540-39666-6_12.

[24] Tao Cheng and Jiaqiu Wang. "Integrated Spatio-temporal Data Mining for Forest Fire Prediction." In: *Transactions in GIS* 12 (5 2008), pp. 591–611. DOI: https://doi.org/10.1111/j.1467-9671.2008.01117.x.

[25] Remi Lam et al. "GraphCast: Learning skillful medium-range global weather forecasting." In: *arXiv* (). DOI: arXiv:2212.12794v1.

[26] Amir Ghaderi, Borhan M. Sanandaji, and Faezeh Ghaderi. "Deep Forecast: Deep Learning-based Spatio-Temporal Forecasting." In: *arXiv* (2017). DOI: arXiv:1707.08110.

[27] Konstantin Klemmer et al. "Generative modeling of spatio-temporal weather patterns with extreme event conditioning." In: *arXiv* (2021). DOI: arXiv:2104.12469.

[28] Guangyin Jin et al. "Spatio-Temporal Graph Neural Networks for Predictive Learning in Urban Computing: A Survey." In: *arXiv* (2023). DOI: arXiv:2303.14483.

[29] R. Kelley Pace et al. "Spatiotemporal Autoregressive Models of Neighborhood Effects." In: *Journal of Real Esatet Finance and Economics* 17 (1 1998), pp. 15–33. DOI: https://doi.org/10.1023/A:1007799028599.

[30] Qicheng Tang, Mengning Yang, and Ying Yang. "ST-LSTM: A Deep Learning Approach Combined Spatio-Temporal Features for Short-Term Forecast in Rail Transit." In: *Journal of Advanced Transportation* 2019 (2019). DOI: https://doi.org/10.1155/2019/8392592.

[31] Stephane Dray, Pierre Legendre, and Pedro R. Peres-Neto. "Spatial modelling: a comprehensive framework for principal coordinate analysis of neighbour matrices (PCNM)." In: *Ecological Modelling* 196 (2006), pp. 483–493. DOI: https://doi.org/10.1016/j.ecolmodel.2006.02.015.

[32] Enrico Lovisotto et al. "Cell Traffic Prediction Using Joint Spatio-Temporal Information." In: *International Conference on Modern Circuits and Systems Technologies* 6 (2006). DOI: https://doi:10.1109/MOCAST.2017.7937674.

[33] Mariana Oliveira, Luís Torgo, and Vítor Santos Costa. "Predicting Wildfires: Propositional and Relational Spatio-Temporal Pre-processing Approaches." In: *Lecture Notes in Artificial Intelligence* 9956 (2016), pp. 183–197.

[34] Isabelle Guyon and André Elisseeff. "An Introduction to Variable and Feature Selection." In: *Journal of Machine Learning Research* 3 (2003), pp. 157–1182. DOI: https://doi:10.5555/944919.944968.

[35] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot. "Variable selection using random forests." In: *Pattern Recognition Letters* 31 (2010), pp. 2225–2236. DOI: https://doi:10.1016/j.patrec.2010.03.014.

[36] Gemma Coxon et al. "CAMELS-GB: hydrometeorological time series and landscape attributes for 671 catchments in Great Britain." In: *Earth System Science Data* 12 (4 2020). DOI: https://doi.org/10.5194/essd-12-2459-2020.

[37] Centre for Environmental Data Analysis. *HadUK-Grid gridded and regional average climate observations for the UK*. 2023. URL: https://catalogue.ceda.ac.uk/uuid/4dc8450d889a491ebb20e724debe2dfb (visited on 08/07/2023).

[38] The Counter Trafficking Data Collaborative. *The Global Synthetic Dataset*. 2023. URL: https://www.ctdatacollaborative.org/page/global-synthetic-dataset (visited on 08/07/2023).

[39] Darren Edge et al. "Design of a Privacy-Preserving Data Platform for Collaboration Against Human Trafficking." In: *arXiv* (2020). DOI: https://arxiv.org/abs/2005.05688.

[40] Andy Liaw and Matthew Wiener. *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*. R package version 4.6-14. 2021. URL: https://CRAN.R-project.org/package=randomForest.

[41] Yvan Saeys, Thomas Abeel, and Yves Van de Peer. "Robust Feature Selection Using Ensemble Feature Selection Techniques". In: *Machine Learning and*

Knowledge Discovery in Databases 5212 (2008), pp. 313–325. DOI: https://doi.org/10.1007/978-3-540-87481-2_21.

[42] Franc Drobnič, Andrej Kos, and Matevž Pustišek. "On the Interpretability of Machine Learning Models and Experimental Feature Selection in Case of Multicollinear Data". In: *Electronics* 9 (5 2020), p. 761. DOI: https://doi.org/10.3390/electronics9050761.

[43] R Core Team. *R: A Language and Environment for Statistical Computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[44] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2009.

[45] Evgenia Dimitriadou et al. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*. R package version 1.7-7. 2021. URL: https://CRAN.R-project.org/package=e1071.

[46] Leo Breiman et al. *Classification and Regression Trees*. CRC press, 1984.

[47] Terry M. Therneau and Elizabeth J. Atkinson. *rpart: Recursive Partitioning*. R package version 4.1-15. 2021. URL: https://CRAN.R-project.org/package=rpart.

[48] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[49] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. arXiv: arXiv:1603.04467.

[50] Mariana Oliveira et al. "Biased resampling strategies for imbalanced spatio-temporal forecasting". In: *International Journal of Data Science and Analytics* 12 (2021), pp. 205–228. DOI: https://doi.org/10.1007/s41060-021-00256-2.

[51] Raquel Espinosa, Fernando Jiménez, and Jose Palma. "Multi-objective evolutionary spatio-temporal forecasting of air pollution". In: *Future Generation Computer Systems* 136 (May 2022). DOI: doi:10.1016/j.future.2022.05.020.

Appendix

A. Definitions

Terminology	**Definition**
Block cross-validation	A method similar to K-fold cross-validation where each fold is a sequential, non-interrupted time series instead of each fold containing a random subset of observations
Catchment	An area with a natural boundary (for example ridges, hills or mountains) where all surface water drains to a common channel to form rivers or creek
Circular variogram model	A model used to describe the spatial correlation structure for a variogram that assumes spatial correlation is constant within a circular range then drops off sharply outside that range; the simplest variogram model
Co-kriging	An extension to kriging that allows for the use of predictor variables beyond the target variable
Diebold-Mariano test	A statistical test used to compare the forecast accuracy of competing models by calculating a test statistic based on differences in the models' mean squared error
Dynamic spatio-temporal model	A spatio-temporal model that incorporates dynamic components that account for changes in spatial dependencies and temporal trends over time; includes state-space models and Bayesian hierarchical models
Embedded methods	A feature selection method that achieves model fitting and feature selection simultaneously
Filter methods	A feature selection method that separates feature selection from the learning algorithm such that the influence of the learning algorithm does not interact with the selection algorithm

Great-circle distance	The shortest distance between two points on the surface of a sphere, measured along the surface of the sphere; can be used to describe the spatial distance component of the spatio-temporal distance
Imbalanced task	When certain ranges of values in the target variable are more important to the end-user, but severely underrepresented in the training data such as predicting extreme values
Inundation	The amount of water that occurs above normally dry ground as a result of flooding
Kriging	A spatial interpolation technique used to estimate values at unobserved locations based on known values at nearby locations using a fitted variogram model; finds the optimal linear combination of nearby known values; assumes spatial stationarity
Leaves	The areas or regions in a image that are missing and need to be filled in
Moran's I	A measure of spatial autocorrelation where negative one indicates perfectly negative correlation, zero indicates no correlation, and positive one indicates perfect correlation
Muli-objective evolutionary algorithm	An optimisation technique that applies the concept of natural selection to competing objective functions, establishing a set of non-dominated solutions; computationally expensive
Nowcast	A forecast that yields local details across the mesoscale and small scale, over a period from the present up to six hours ahead and which provides a detailed description of the present weather
Nugget	A characteristic of spatial dependence that represents the semivariance value at a lag distance of zero or very close to zero; represents the spatial variance at very short distances; associated with measurement errors that are not captured in model
Pentaspherical variogram model	A model used to describe the spatial correlation structure for a variogram that extends the tetraspherical variogram model by allowing for additional parameters to capture more gradual transitions in semivariance

Range	A characteristic of spatial dependence that represents the distance beyond which the spatial correlation is considered to be negligible; the maximum spatial extent over which nearby locations exhibit significant similarity
Semivariance	A measure of the average difference between pairs of spatial locations within each lag distance class; quantifies differences as a function of the lag distance; used in calculating variograms
Sill	A characteristic of spatial dependence that represents the maximum semivariance value observed in a variogram plot; represents the variance of data when the lag distance approaches infinity; provides an upper bound for the semivariance
Sliding window approach	An approach to time series forecasting that uses predictions of previous time periods as predictors for future time periods; can exponentially increase errors
Spatial stationarity	When the spatial correlation structure of spatial data remains constant over time
Spatio-temporal autoregressive model	A spatio-temporal model that explicitly models spatio-temporal dependence by considering the influence of neighbourhing locations at previous time points; essentially extends spatio autoregressive models to also include temporal lags
Spatio-temporal block cross-validation	A form of block cross-validation where time is grouped in sequential blocks and locations are simultaneously grouped together randomly, in contiguous geographic regions, or in a systematic checkered pattern
Spatio-temporal distance	The distance between any two points in the space-time dimension
Spatio-temporal neighbourhood	The set of points within a certain spatio-temporal distance of a given point
Spatio-temporal spread	A spatio-temporal indicator equal to the standard deviation of the target variable within the spatio-temporal neighbourhood of the target location; weighted version uses the weighted spatio-temporal typical value

Spatio-temporal tendency	A spatio-temporal indicator equal to the ratio between two spatio-temporal weights calculated with different spatio-temporal neighbourhood sizes; weighted version uses the weighted spatio-temporal typical value
Spatio-temporal typical value	A spatio-temporal indicator equal to the average value of target variable within the spatio-temporal neighbourhood of the target location; weighted version multiplies the spatio-temporal weight to the data points
Spatio-temporal weight	The inverse of the spatio-temporal distance between two points
Spherical variogram model	A model used to describe the spatial correlation structure for a variogram that assumes spatial correlation reaches a maximum value within a circular range then levels off beyond that range; increases in semivariance until the sill and then is constant
Stage	The water level in a river or stream with respect to a chosen reference height
State-space model	A statistical model that describes the evolution of a system of time; the current state typically is represented as a linear combination of the previous state and random error; also called a dynamic linear model or a hidden Markov model
Step-ahead predicitons	A method of time series forecasting that predicts the next value of the series based on recent values of the series; the most common form of time series forecasting
Surrogate-assisted evolutionary algorithm	A class of optimization techniques that combine evolutionary algorithms with surrogate modeling, which constructs a simplified mathematical model of a complex, computationally expensive objective function
Technical indicators	The statistical information that is used to expand the set of lagged variables used in time-delay embedding; statistical summaries of certain properties of the time series; both distance and time components are normalised
Tetraspherical variogram model	A model used to describe the spatial correlation structure for a variogram that extends the spherical variogram model by allowing for a slower decrease in semivariance beyond the range

Time-delay embedding	A process of forecasting future values by including past values and predictors as lagged variables in a prediction model
Time-varying spatial weights	A set of spatial weights stored in matrix format that vary over time; reflect changes in spatial dependence over time
Variogram	A measure of the spatial dependence or auto-correlation of a spatial data set; describes how the similarity of values between pairs of spatial locations changes with distance; used in kriging; also called a semivariogram or an experimental variogram
Variogram Plot	A plot of the semivariance values against the lag distances on a scattler plot; visualises the relationship between spatial separation and semivariance; a variogram describes this function
Wrapper methods	A feature selection method that uses the predictive accuracy of a predetermined learning algorithm to determine the quality of the selected features; includes forwards selection, backwards selection, and bi-directional selection; computationally expensive

B. Code

i) CAMELS-GB Data Cleaning

```
#### PREPARE WORKSPACE ####

# Set working directory
setwd("~/Desktop/Classes/Dissertation")

# Import data manipulation libraries
library(plyr)
library(dplyr)
library(data.table)

# Import spatio-temporal data libraries
library(sf)
library(zoo)
library(lubridate)

# Import file compression libraries
library(fst)
```

```
#### LOAD DATA ####

# Create list of feature data files
feature_files <- list.files(
  path = "./2-Data Collection/camels-gb/data/",
  pattern = "*.csv")

# Read in the dataframe for each feature
feature_list <- lapply(feature_files, function(file) read.csv(paste0(
  "./2-Data Collection/camels-gb/data/", file)))

# Create list of time series data files
timeseries_files <- list.files(
  path = "./2-Data Collection/camels-gb/data/timeseries/",
  pattern = "*.csv")

# Read in the time series dataframe for each catchment
timeseries_list <- lapply(timeseries_files, function(file) read.csv(paste0(
  "./2-Data Collection/camels-gb/data/timeseries/", file)))

# Read in catchment boundaries
catchment_boundaries <- st_read(paste0(
  "./2-Data Collection/camels-gb/data/boundaries/",
  "CAMELS_GB_catchment_boundaries.shp"))


#### COMBINE DATA ####

# Add gauge id to each time series file
for(i in (1:length(timeseries_files))){
  timeseries_list[[i]]$gauge_id <- as.integer(strsplit(
    timeseries_files[i], "_")[[1]][5])
}

# Combine time series data vertically
timeseries_df <- do.call(rbind, timeseries_list) %>%
  # Set gauge id to integer
  mutate(gauge_id = as.integer(gauge_id)) %>%
  # Optimise memory usage by storing data as a data table
  data.table()

# Calculate centre points of catchment boundaries
center_points <- st_centroid(catchment_boundaries)
```

```
#### CLEAN DATA ####

# Edit dataframe
df_cleaned <- df %>%
  # Filter out data before the year 2000
  filter(substring(date, 1, 2) == "20") %>%
  # Optimise memory usage by storing data as a data table
  data.table() %>%
  # Link data to catchment boundaries
  merge(center_points["ID"], by.x = "gauge_id", by.y = "ID") %>%
  # Edit columns
  mutate(
    # Flag imputations
    impute_flag = is.na(discharge_vol),
    # Impute missing values with previous day values for up to two days
    discharge_vol = na.locf(discharge_vol, maxgap = 2),
    # Convert spatial column to non-spatial data
    x = st_coordinates(geometry)[, "X"],
    y = st_coordinates(geometry)[, "Y"],
    # Convert dates to date data type
    date = as.Date(date),
    # Create column for month and day
    month_day = format(date, "%m-%d")) %>%
  # Group by gauge id and month day pairing
  group_by(gauge_id, month_day) %>%
  # Edit columns
  mutate(
    # Create columns for historic daily averages
    discharge_vol_avg = mean(discharge_vol, na.rm = T),
    temperature_avg = mean(temperature, na.rm = T),
    precipitation_avg = mean(precipitation, na.rm = T),
    pet_avg = mean(pet, na.rm = T),
    shortwave_rad_avg = mean(shortwave_rad, na.rm = T),
    windspeed_avg = mean(windspeed, na.rm = T),
    humidity_avg = mean(humidity, na.rm = T),
    # Create columns for percent change from historic daily averages
    discharge_vol_change =
      (discharge_vol - discharge_vol_avg) / discharge_vol_avg,
    temperature_change =
      (temperature - temperature_avg) / temperature_avg,
    precipitation_change =
      (precipitation - precipitation_avg) / precipitation_avg,
    pet_change =
      (pet - pet_avg) / pet_avg,
    shortwave_rad_change =
      (shortwave_rad - shortwave_rad_avg) / shortwave_rad_avg,
```

```
    windspeed_change =
      (windspeed - windspeed_avg) / windspeed_avg,
    humidity_change =
      (humidity - humidity_avg) / humidity_avg) %>%
  # Ungroup data
  ungroup() %>%
  # Remove geometry column
  dplyr::select(-geometry) %>%
  # Select columns of interest
  dplyr::select(
    gauge_id, date, discharge_vol, discharge_vol_change, impute_flag,
    temperature_change, precipitation_change, pet_change,
    shortwave_rad_change, windspeed_change, humidity_change, x, y)

# Calculate largest sections of consecutive missing data for each gauge
v <- c()
for(i in unique(df$gauge_id)){
 runs <- rle(is.na(df_cleaned[df_cleaned$gauge_id == i,]$discharge_vol_change))
  v = append(v, max(runs$lengths[runs$values]))
}

# Remove gauges with over two consecutive days of missing data
missing_gauges <- unique(df_cleaned$gauge_id)[which(v > 2)]
df_cleaned <- df_cleaned %>% filter(!gauge_id %in% missing_gauges)

# Remove gauges with missing drainage slopes
missing_slopes <- unique(df[is.na(df$dpsbar),]$gauge_id)
df_cleaned <- df_cleaned %>% filter(!gauge_id %in% missing_slopes)

# Check for missing values
any(is.na(df_cleaned))


#### CALCULATE DISTANCE MATRIX ####

# Identify spatial components
points <- SpatialPoints(
  df_cleaned[df_cleaned$date == min(df_cleaned$date), c("x", "y")])

# Set coordinate reference system for spatial component
proj4string(points) <- CRS("+init=epsg:27700")

# Create the spatial distance matrix with weights
dist <- matrix(spDists(points), ncol = length(unique(df_cleaned$gauge_id)),
               nrow = length(unique(df_cleaned$gauge_id)))
dist_matrix <- as.data.frame(dist)
```

```
colnames(dist_matrix) <- unique(df_cleaned$gauge_id)
rownames(dist_matrix) <- unique(df_cleaned$gauge_id)


#### EXPORT DATA ####

# Export dataframe
write.fst(df_cleaned, "./3-Model Building/camels-gb/data/cleaned-data.fst")

# Export spatial distance matrix
write.csv(dist_matrix, "./3-Model Building/camels-gb/data/dist-matrix.csv")
```

ii) CAMELS-GB Indicator Extraction Function

```
#### PREPARE WORKSPACE ####

# Import data manipulation libraries
library(plyr)
library(dplyr)
library(tibble)


#### WRITE FUNCTION TO CALCULATE NEIGHBOURHOOD INDICATORS ####

get_indicators <- function(
    df, dist_matrix, target_gauge, value, sample_dates, a, B,
    metrics = c("m", "md", "s", "M", "Md"), forward = T, reverse = T) {

  # Select target distance
  target_dist <- dist_matrix[rownames(dist_matrix) == target_gauge, ] %>%
    # Transpose rows and columns
    t() %>%
    # Convert to dataframe
    as.data.frame() %>%
    # Set index as a column
    rownames_to_column(var = "gauge_id") %>%
    # Rename distance column
    rename(D_A = as.character(target_gauge)) %>%
    # Normalise distance values
    mutate(D_A = D_A / (max(D_A) - min(D_A)))

  # Filter to target gauge
  df_indicators <- df %>%
    filter(gauge_id == target_gauge & date %in% sample_dates)
```

```
# Create reference dataframe
df_ref <- df %>%
  # Merge distance from target gauge
  merge(target_dist, by = "gauge_id", all.x = T) %>%
  # Order by date and gauge id
  arrange(date, gauge_id)

# Loop through time stamps
for(t in 1:nrow(df_indicators)){
  # Initialise indicator list
  indicators <- c()
  # Loop through neighbourhood scaling factor
  for(i in 1:length(a)){
    # Loop through neighbourhood size
    for(k in 1:length(B)){

      # Create name for selected neighbourhood
      id <- paste0("_a", a[i], "_B", B[k])
      nhood_name <- paste0("N", id)
      nhood_name_r <- paste0("N_r", id)

      # Print iteration step
      print(paste0(
        df_indicators[t,]$date,
        ": Calculating indicators for neighbourhood ", nhood_name))

      # Identify neighbourhood for selected time step
      df_ref_time <- df_ref %>%
        # Edit columns
        mutate(
          # Calculate temporal distance from target point
          D_T = ifelse(
            as.numeric(df_indicators[t,]$date - date) > 0,
            as.numeric(df_indicators[t,]$date - date)
            / as.numeric(max(date) - min(date)), Inf),
          # Calculate spatio-temporal distance
          D = a[i] * D_A + (1 - a[i]) * D_T,
          D_r = a[i] * D_A - (1 - a[i]) * D_T,
          # Create indicator for spatio-temporal neighbourhood
          !!nhood_name := ifelse(
            date < df_indicators[t,]$date & D <= B[k], 1, 0),
          !!nhood_name_r := ifelse(
            date < df_indicators[t,]$date & D_r <= 0, 1, 0))

      # Filter reference data to spatio-temporal neighbourhood
      if(forward){
```

```
    nhood <- df_ref_time[df_ref_time[[nhood_name]] == 1, ] %>%
      # Renormalise spatio-temporal distance
      mutate(D_norm = 1/D / sum(1/D))}
  if(reverse){
   nhood_r <- df_ref_time[df_ref_time[[nhood_name_r]] == 1, ] %>%
      # Renormalise spatio-temporal distance
      mutate(D_r_norm = 1/D_r / sum(1/D_r))}

  # Calculate indicators
  indicator_m <- c()
  name_m <- c()
  if("m" %in% metrics & forward){
    indicator_m <- mean(nhood[[value]], na.rm = T)
    name_m <- paste0("m", id)}

  indicator_m_r <- c()
  name_m_r <- c()
  if("m" %in% metrics & reverse){
    indicator_m_r <- mean(nhood_r[[value]], na.rm = T)
    name_m_r <- paste0("m_r", id)}

  indicator_md <- c()
  name_md <- c()
  if("md" %in% metrics & forward){
   indicator_md <- mean(nhood[[value]] * nhood$D_norm, na.rm = T)
    name_md <- paste0("md", id)}

  indicator_md_r <- c()
  name_md_r <- c()
  if("md" %in% metrics & reverse){
indicator_md_r <- mean(nhood_r[[value]] * nhood_r$D_r_norm, na.rm = T)
    name_md_r <- paste0("md_r", id)}

  indicator_s <- c()
  name_s <- c()
  if("s" %in% metrics & forward){
    indicator_s <- sd(nhood[[value]], na.rm = T)
    name_s <- paste0("s", id)}

  indicator_s_r <- c()
  name_s_r <- c()
  if("s" %in% metrics & reverse){
    indicator_s_r <- sd(nhood_r[[value]], na.rm = T)
    name_s_r <- paste0("s_r", id)}

  # Rename indicators
```

```
        indicators_id <- setNames(
          c(indicator_m, indicator_m_r, indicator_md,
            indicator_md_r, indicator_s, indicator_s_r),
          c(name_m, name_m_r, name_md, name_md_r, name_s, name_s_r))
        # Combine indicators
        indicators <- c(indicators, indicators_id)
      }
    }
    # Add indicators to time stamp
    df_indicators[t, names(indicators)] <- indicators
  }
  # Calculate spatio temporal tendencies
  if("M" %in% metrics){
    for(i in 1:length(a)){
      for(k in 1:(length(B) - 1)){
        id <- paste0("_a", a[i], "_B", B[k], "_B", B[k + 1])
        # Create name for selected neighbourhood comparison
        id_1 <- paste0("_a", a[i], "_B", B[k])
        id_2 <- paste0("_a", a[i], "_B", B[k + 1])
        # Calculate trend
        df_indicators[[paste0("M", id)]] <-
          df_indicators[[paste0("m", id_1)]] /
          df_indicators[[paste0("m", id_2)]]
      }
    }
  }
  if("Md" %in% metrics){
    for(i in 1:length(a)){
      for(k in 1:(length(B) - 1)){
        id <- paste0("_a", a[i], "_B", B[k], "_B", B[k + 1])
        # Create name for selected neighbourhood comparison
        id_1 <- paste0("_a", a[i], "_B", B[k])
        id_2 <- paste0("_a", a[i], "_B", B[k + 1])
        # Calculate weighted trend
        df_indicators[[paste0("Md", id)]] <-
          df_indicators[[paste0("md", id_1)]] /
          df_indicators[[paste0("md", id_2)]]
      }
    }
  }
  # Return neighbourhood indicators
  return(df_indicators)
}
```

iii) CAMELS-GB Neighbourhood Optimisation

```
#### PREPARE WORKSPACE ####

# Set working directory
setwd("~/Desktop/Classes/Dissertation")

# Import data manipulation libraries
library(plyr)
library(dplyr)
library(tibble)

# Import statistical libraries
library(randomForest)

# Import file compression libraries
library(fst)

# Import data visualisation libraries
library(ggplot2)

# Read in indicator function
source("./3-Model Building/camels-gb/data/get_indicators.R")

# Set randomisation seed
set.seed(123)


#### LOAD DATA ####

# Load cleaned data
df <- read.fst("./3-Model Building/camels-gb/data/cleaned-data.fst") %>%
  # Order by date and gauge id
  arrange(date, gauge_id)

# Load distance matrix
dist_matrix <- read.csv("./3-Model Building/camels-gb/data/dist-matrix.csv",
                        row.names = 1)
colnames(dist_matrix) <- rownames(dist_matrix)


#### SELECT OPTIMAL NEIGHBOURHOOD INDICATORS ####

# Select target gauge
target_gauge <- 39023

# Select target variable
value <- "discharge_vol_change"
```

```
# Get random sample of dates
sample_dates <- sample(unique(df$date), 2)

# Set neighbourhood parameters
a <- c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)
B <- c(0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1)

# Calculate dataframe of indicators for the gauge of interest
df_indicators <- get_indicators(
  df, dist_matrix, target_gauge, value, sample_dates, a, B, metrics = "m")

# Select indicator
indicator <- "m"

# Build random forest model
cols <- grep(paste0("^", indicator, "_"), colnames(df_indicators), value = T)
data <- df_indicators[c("discharge_vol_change", value, cols)]
formula <- paste(cols, collapse = " + ")
formula <- as.formula(paste("discharge_vol_change", "~", formula))
rf_model <- randomForest(formula, data = na.omit(data))

# Plot variable importance
var_imp <- data.frame(importance(rf_model)) %>%
  rownames_to_column(var = "variable") %>%
  mutate(
    a = substr(variable, 4, 6),
    B = substr(variable, 9, 12)) %>%
  arrange(desc(IncNodePurity)) %>%
  head(10)
plot <- ggplot(var_imp, aes(x = reorder(variable, IncNodePurity),
                            y = IncNodePurity)) +
  geom_bar(stat = "identity", fill = "dodgerblue") +
  coord_flip() +
  labs(x = "Variable", y = "Importance") +
  theme_minimal() +
  theme(plot.background = element_rect(fill = "white"))


#### EXPORT DATA ####

# Export dataframe
write.fst(df_indicators, paste0(
  "./3-Model Building/camels-gb/data/indicators/test_indicators_",
  value, ".fst"))
```

```
# Export variable importance
write.csv(var_imp, paste0(
  "./3-Model Building/camels-gb/data/importance/imp_", value, ".csv"),
  row.names = F)

# Save plot
ggsave(paste0(
  "./3-Model Building/camels-gb/figures/importance_", value, ".png"),
  plot, width = 8, height = 6, dpi = 300)
```

iv) CAMELS-GB Indicator Extraction

```
#### PREPARE WORKSPACE ####

# Set working directory
setwd("~/Desktop/Classes/Dissertation")

# Import data manipulation libraries
library(plyr)
library(dplyr)
library(tibble)
library(stringr)
library(purrr)

# Import file compression libraries
library(fst)

# Read in indicator function
source("./3-Model Building/camels-gb/data/get_indicators.R")


#### LOAD DATA ####

# Load cleaned data
df <- read.fst("./3-Model Building/camels-gb/data/cleaned-data.fst") %>%
  # Order by date and gauge id
  arrange(date, gauge_id)

# Load distance matrix
dist_matrix <- read.csv("./3-Model Building/camels-gb/data/dist-matrix.csv",
                        row.names = 1)
colnames(dist_matrix) <- rownames(dist_matrix)


#### PREPARE DATA ####
```

```
# Create list of variables
vars <- c(
  "discharge_vol_change", "temperature_change", "precipitation_change",
 "pet_change", "shortwave_rad_change", "windspeed_change", "humidity_change")

# Create list of optimised neighbourhoods
nhoods <- list(
  c("_a0.4_B0.01", "_a0.4_B0.02", "_a0.4_B0.03"),
  c("_r_a0.5_B0.07", "_r_a0.5_B0.08", "_r_a0.5_B0.09"),
  c("_a0.9_B0.01", "_a0.9_B0.02", "_a0.9_B0.03"),
  c("_r_a0.3_B0.08", "_r_a0.3_B0.09", "_r_a0.3_B0.1"),
  c("_a0.9_B0.08", "_a0.9_B0.09", "_a0.9_B0.1"),
  c("_a0.9_B0.01", "_a0.9_B0.02", "_a0.9_B0.03"),
  c("_r_a0.7_B0.08", "_r_a0.7_B0.09", "_r_a0.7_B0.1"))

# Set naive neighbourhood
naive <- c("_a0.5_B0.01", "_a0.5_B0.02", "_a0.5_B0.03")

# Set parameters
i <- 2
target_gauge <- 39023
value <- vars[i]
dates <- unique(df$date)
a <- unique(c(as.numeric(gsub(".*a(\\d+\\.\\d+).*", "\\1", nhoods[[i]][1])),
              as.numeric(gsub(".*a(\\d+\\.\\d+).*", "\\1", naive[1]))))
B <- unique(c(as.numeric(gsub(".*B(\\d+\\.\\d+).*", "\\1", nhoods[[1]])),
           as.numeric(gsub(".*B(\\d+\\.\\d+).*", "\\1", nhoods[[i]]))))
reverse <- grepl("r", nhoods[[i]][1])

# Calculate dataframe of indicators for the selected variable
df_indicators <- get_indicators(
  df, dist_matrix, target_gauge, value, dates, a, B, reverse = reverse)


#### EXPORT VARIABLE DATA ####

# Export dataframe
write.fst(df_indicators, paste0(
 "./3-Model Building/camels-gb/data/indicators/indicators_", value, ".fst"))


#### COMBINE DATA ####

# Create function to change column names
add_prefix <- function(colname, prefix){
```

```
  paste0(prefix, "_", colname)
  }

# Load extracted indicators for each variable
df_names <- c()
for(i in 1:length(vars)){
  name <- paste0("df_indicators_", vars[i])
  assign(name, read.fst(
    paste0("./3-Model Building/camels-gb/data/indicators/",
           "indicators_", vars[i], ".fst"))[, -(1:13)] %>%
      rename_with(~ add_prefix(.x, vars[i]), everything()))
  df_names <- c(df_names, name)
  }

# Create function to calculate lags
calculate_lags <- function(df, var, lags){
  map_lag <- lags %>% map(~partial(lag, n = .x))
  return(df %>% mutate(across(
    .cols = {{var}}, .fns = map_lag, .names = "{.col}_lag{lags}")))
}

# Calculate final data frame with indicators and lagged values
indicators <- mget(df_names)
df_full <- df %>%
  # Filter to target gauge
  filter(gauge_id == target_gauge) %>%
  # Calculate same location lagged values
  calculate_lags(discharge_vol_change, 1:3) %>%
  calculate_lags(temperature_change, 1:3) %>%
  calculate_lags(precipitation_change, 1:3) %>%
  calculate_lags(windspeed_change, 1:3) %>%
  calculate_lags(humidity_change, 1:3) %>%
  # Combine extracted indicators for each variable
  cbind(indicators)

# Fix column names
colnames(df_full) <- sub("^[^.]*\\.", "", colnames(df_full))

#### EXPORT COMBINED DATA ####

# Export combined data where rows with missing data are removed
write.fst(na.omit(df_full), paste0(
  "./3-Model Building/camels-gb/data/indicators/all_indicators.fst"))
```

v) CAMELS-GB Linear Model Predictions

```
#### PREPARE WORKSPACE ####

# Set working directory
setwd("~/Desktop/Classes/Dissertation")

# Import data manipulation libraries
library(plyr)
library(dplyr)
library(data.table)
library(purrr)

# Import file compression libraries
library(fst)

# Set randomisation seed
set.seed(123)

# Read in indicator function
source("./3-Model Building/camels-gb/data/get_indicators.R")


#### LOAD DATA ####

# Load model data
df_full <- read.fst(
  "./3-Model Building/camels-gb/data/indicators/all_indicators.fst")

# Load cleaned data
df <- read.fst("./3-Model Building/camels-gb/data/cleaned-data.fst") %>%
  # Order by date and gauge id
  arrange(date, gauge_id)

# Load distance matrix
dist_matrix <- read.csv("./3-Model Building/camels-gb/data/dist-matrix.csv",
                        row.names = 1)
colnames(dist_matrix) <- rownames(dist_matrix)


#### FORMAT DATA ####

# Create function to calculate lags
calculate_lags <- function(df, var, lags){
  map_lag <- lags %>% map(~partial(lag, n = .x))
  return(df %>% mutate(across(
    .cols = {{var}}, .fns = map_lag, .names = "{.col}_lag{lags}")))
}
```

```
# Split data into cross validation blocks
block_size <- 7
df_full$block <- c(
  rep(1:(nrow(df_full) %/% 7), each = 7), 1:(nrow(df_full) %% 7))

# Check for missing data
any(is.na(df_full))

# Select cross validation blocks
n <- length(unique(df_full$block))
cv <- sample(unique(df_full$block), n)

# Create empty data frame for storing results
nrow <- length(cv) * 3
results <- data.frame(
  config = rep(c("lag", "nhood", "opt"), length(cv)),
  k = rep(cv, each = 3), mae = numeric(nrow))


#### BUILD LAG CONFIGURATION MODEL ####

for(k in cv){
  # Create training and test set
  df_train <- df_full[!df_full$block == k,]
  df_test <- df_full[df_full$block == k,]

  # Train model for lag configuration
  predictors <- colnames(df_full)[grep("_lag", colnames(df_full))]
  data <- df_train[, c("discharge_vol_change", predictors)]
  model_lag <- lm(discharge_vol_change ~ ., data)

  # Make predictions on hold-out data using a sliding window approach
  remove <- colnames(df_full)[
    grep("discharge_vol_change_lag", colnames(df_full))]
  df_pred <- df_full
  # Remove prediction values
  df_pred[which(df_pred$block == k), remove] <- NA
  # Loop through dates
  for(i in 1:nrow(df_test)){
    # Predict new value
    id <- which(df_pred$date == df_test[i,]$date)
    pred <- predict(model_lag, newdata = df_test[i,])
    df_pred[id, "discharge_vol_change"] <- pred
    # Update prediction values
    update <- df_pred %>%
```

```
      calculate_lags(discharge_vol_change, 1:3)
    df_test[, -4] <- update[which(df_pred$block == k), -4]
  }

  # Evaluate model performance
  preds <- df_pred[which(df_pred$block == k),]$discharge_vol_change

  # Store evaluation
  mae <- mean(abs(df_test$discharge_vol_change - preds))
  results[results$config == "lag" & results$k == k,]$mae <- mae
}


#### BUILD NEIGHBOURHOOD CONFIGURATION MODEL ####

for(k in cv){
  # Create training and test set
  df_train <- df_full[!df_full$block == k,]
  df_test <- df_full[df_full$block == k,]

  # Train model for neighbourhood configuration
  nhood_1 <- "discharge_vol_change.*(a0.5_B0.01|_lag)"
  nhood_2 <- "temperature_change.*(m_a0.5_B0.01|_lag)"
  nhood_3 <- "precipitation_change.*(m_a0.5_B0.01|_lag)"
  nhood_4 <- "pet_change.*(m_a0.5_B0.01|_lag)"
  nhood_5 <- "shortwave_rad_change.*(m_a0.5_B0.01|_lag)"
  nhood_6 <- "windspeed_change.*(m_a0.5_B0.01|_lag)"
  nhood_7 <- "humidity_change.*(m_a0.5_B0.01|_lag)"
  predictors_1 <- colnames(df_full)[grepl(nhood_1, colnames(df_full))]
  predictors_2 <- colnames(df_full)[grepl(nhood_2, colnames(df_full))]
  predictors_3 <- colnames(df_full)[grepl(nhood_3, colnames(df_full))]
  predictors_4 <- colnames(df_full)[grepl(nhood_4, colnames(df_full))]
  predictors_5 <- colnames(df_full)[grepl(nhood_5, colnames(df_full))]
  predictors_6 <- colnames(df_full)[grepl(nhood_6, colnames(df_full))]
  predictors_7 <- colnames(df_full)[grepl(nhood_7, colnames(df_full))]
  data <- df_train[, c(
    "discharge_vol_change", predictors_1, predictors_2, predictors_3,
    predictors_4, predictors_5, predictors_6, predictors_7)]
  model_nhood <- lm(discharge_vol_change ~ ., data)

  # Make predictions on hold-out data using a sliding window approach
  remove <- colnames(df_full)[
    grep("discharge_vol_change_lag", colnames(df_full))]
  df_pred <- df_full
  # Remove prediction values
  df_pred[which(df_pred$block == k), remove] <- NA
```

```
  # Loop through dates
  for(i in 1:nrow(df_test)){
    # Predict new value
    id <- which(df_pred$date == df_test[i,]$date)
    pred <- predict(model_nhood, newdata = df_test[i,])
    df_pred[id, "discharge_vol_change"] <- pred
    # Update prediction values
    update <- df_pred %>%
      calculate_lags(discharge_vol_change, 1:3)
    df_test[, -4] <- update[which(df_pred$block == k), -4]
  }

  # Evaluate model performance
  preds <- df_pred[which(df_pred$block == k),]$discharge_vol_change

  # Store evaluation
  mae <- mean(abs(df_test$discharge_vol_change - preds))
  results[results$config == "nhood" & results$k == k,]$mae <- mae
}


#### BUILD OPTIMISED CONFIGURATION MODEL ####

for(k in cv){
  # Create training and test set
  df_train <- df_full[!df_full$block == k,]
  df_test <- df_full[df_full$block == k,]

  # Train model for optimised configuration
  nhood_1 <- "discharge_vol_change.*(a0.4_B0.01|_lag)"
  nhood_2 <- "temperature_change.*(m_r_a0.5_B0.08|_lag)"
  nhood_3 <- "precipitation_change.*(m_a0.9_B0.02|_lag)"
  nhood_4 <- "pet_change.*(m_r_a0.3_B0.09|_lag)"
  nhood_5 <- "shortwave_rad_change.*(m_a0.9_B0.1|_lag)"
  nhood_6 <- "windspeed_change.*(m_a0.9_B0.01|_lag)"
  nhood_7 <- "humidity_change.*(m_r_a0.7_B0.09|_lag)"
  predictors_1 <- colnames(df_full)[grepl(nhood_1, colnames(df_full))]
  predictors_2 <- colnames(df_full)[grepl(nhood_2, colnames(df_full))]
  predictors_3 <- colnames(df_full)[grepl(nhood_3, colnames(df_full))]
  predictors_4 <- colnames(df_full)[grepl(nhood_4, colnames(df_full))]
  predictors_5 <- colnames(df_full)[grepl(nhood_5, colnames(df_full))]
  predictors_6 <- colnames(df_full)[grepl(nhood_6, colnames(df_full))]
  predictors_7 <- colnames(df_full)[grepl(nhood_7, colnames(df_full))]
  data <- df_train[, c(
   "discharge_vol_change", predictors_1, predictors_2, predictors_3,
   predictors_4, predictors_5, predictors_6, predictors_7)]
```

```
  model_opt <- lm(discharge_vol_change ~ ., data)

  # Make predictions on hold-out data using a sliding window approach
  remove <- colnames(df_full)[
    grep("discharge_vol_change_lag", colnames(df_full))]
  df_pred <- df_full
  # Remove prediction values
  df_pred[which(df_pred$block == k), remove] <- NA
  # Loop through dates
  for(i in 1:nrow(df_test)){
    # Predict new value
    id <- which(df_pred$date == df_test[i,]$date)
    pred <- predict(model_opt, newdata = df_test[i,])
    df_pred[id, "discharge_vol_change"] <- pred
    # Update prediction values
    update <- df_pred %>%
      calculate_lags(discharge_vol_change, 1:3)
    df_test[, -4] <- update[which(df_pred$block == k), -4]
  }

  # Evaluate model performance
  preds <- df_pred[which(df_pred$block == k),]$discharge_vol_change

  # Store evaluation
  mae <- mean(abs(df_test$discharge_vol_change - preds))
  results[results$config == "opt" & results$k == k,]$mae <- mae
}


#### EXPORT RESULTS ####

# Export results
write.csv(results, "./3-Model Building/camels-gb/models/results/lm.csv")
table <- results %>% group_by(config) %>%
  summarise(mean = mean(mae, na.rm = T), sd = sd(mae, na.rm = T))
write.csv(table, "./3-Model Building/camels-gb/models/results/table_lm.csv")

# Print t-test results
m1 = table[table$config == "nhood",]$mean
m2 = table[table$config == "opt",]$mean
sd = table[table$config == "opt",]$sd
t = (m1 - m2) / sqrt(2 * sd / n)
print(paste0("t-value: ", t))
print(paste0("p-value: ", 2 * (1 - pt(abs(t), df = n - 1))))
```