Application of Machine Learning in Operational Flood Forecasting and Mapping

Syed Rezwan Kabir

Submitted for the degree of Doctor of Philosophy

Heriot-Watt University

School of Energy, Geoscience, Infrastructure and Society

September 2020

ABSTRACT

Considering the computational effort and expertise required to simulate 2D hydrodynamic models, it is widely understood that it is practically impossible to run these types of models during a real-time flood event. To allow for real-time flood forecasting and mapping, an automated, computationally efficient and robust data driven modelling engine - as an alternative to the traditional 2D hydraulic models - has been proposed. The concept of computationally efficient model relies heavily on replacing time consuming 2D hydrodynamic software packages with a simplified model structure that is fast, reliable and can robustly retains sufficient accuracy for applications in real-time flood forecasting, mapping and sequential updating.

This thesis presents a novel data-driven modelling framework that uses rainfall data from meteorological stations to forecast flood inundation maps. The proposed framework takes advantage of the highly efficient machine learning (ML) algorithms and also utilities the state-of-the-art hydraulic models as a system component. The aim of this research has been to develop an integrated system, where a data-driven rainfall-streamflow forecasting model sets up the upstream boundary conditions for the machine learning based classifiers, which then maps out multi-step ahead flood extents during an extreme flood event.

To achieve the aim and objectives of this research, firstly, a comprehensive investigation was undertaken to search for a robust ML-based multi-step ahead rainfall-streamflow forecasting model. Three potential models were tested (Support Vector Regression (SVR), Deep Belief Network (DBN) and Wavelet decomposed Artificial Neural Network (WANN)). The analysis revealed that SVR-based models perform most efficiently in forecasting streamflow for shorter lead time. This study also tested the portability of model parameters and performance deterioration rates.

Secondly, multiple ML-based models (SVR, Random Forest (RF) and Multi-layer Perceptron (MLP)) were deployed to simulate flood inundation extents. These models were trained and tested for two geomorphologically distinct case study areas. In the first case of study, of the models trained using the outputs from LISFLOOD-FP hydraulic model and upstream flow data for a large rural catchment (Niger Inland Delta, Mali). For the second case of study similar approach was adopted, though 2D Flood Modeller software package was used to generate target data for the machine learning algorithms and to model inundation extent for a semi-urban floodplain (Upton-Upon-Severn, UK).

In both cases, machine learning algorithms performed comparatively in simulating seasonal and event based fluvial flooding.

Finally, a framework was developed to generate flood extent maps from rainfall data using the knowledge learned from the case studies. The research activity focused on the town of Upton-Upon-Severn and the analysis time frame covers the flooding event of October-November 2000. RF-based models were trained to forecast the upstream boundary conditions, which were systematically fed into MLP-based classifiers. The classifiers detected states (wet/dry) of the randomly selected locations within a floodplain at every time step (e.g. one hour in this study). The forecasted states of the sampled locations were then spatially interpolated using regression kriging method to produce high resolution probabilistic inundation (9m) maps. Results show that the proposed data centric modelling engine can efficiently emulate the outcomes of the hydraulic model with considerably high accuracy, measured in terms of flood arrival time error, and classification accuracy during flood growing, peak, and receding periods.

The key feature of the proposed modelling framework is that, it can substantially reduce computational time, i.e. ~14 seconds for generating flood maps for a flood plain of ~4 km^2 at 9m spatial resolution (which is significantly low compared to a fully 2D hydrodynamic model run time).

DEDICATION

I dedicate this thesis to my beloved parents Dr. Syed Md. Abu Taleb and Ms. Moududa Akhter.

ACKNOWLEDGEMENTS

First and foremost, I wish to acknowledge with gratitude the support and concern of my primary supervisor Dr Sandhya Patidar. She has always been supportive to my work from the day I arrived in this University and helping me out with all the problems that I have faced during this period. At the end of my 1st year final examination I went through an emotional turmoil, and she has been one of those persons who stood behind me. After this tough period, I had to regroup and begin the PhD work all over again. Without her support and backup, I would not be able to do it and for this I will forever be grateful to her. One more thing worth acknowledging is the way she maintained a right balance between allowing me the space to carry out research at my own pace and meeting every week to follow up the work associated with the PhD study.

I would like to express my gratitude to my second supervisor Professor Gareth Pender for his invaluable inputs regarding hydraulic modelling. Although he is extremely busy with his administrative works, but he always had time for me to discuss about my PhD progression, future plans and any problem that I might be facing that needs his help. Before starting this PhD, I had absolutely zero experience of running complex 2D hydraulic models. To gain hand in experience of running 2D models, he personally contacted Dr Jeffrey Neal from University of Bristol to give me some training on LISFLOOD-FP hydraulic model.

My sincere thanks and gratitude to Dr Jeffrey Neal for walking me through the LISFLOOD-FP model. He spent a lot of time to teach me the features of the model, and also sourced some data for the PhD. One of the biggest challenges that I faced in this PhD was the lack of data. The data provided by Jeff essentially became a major part of this thesis. The hydro-meteorological and geomorphological data that he has provided directly contributed to developing different data-driven and 2D models in Chapter four and Chapter five.

I would also like to express my gratitude to Dr Matthew Tyburski from Global Surface Intelligence, where I have worked for more than two years as a part time employee. Matt has a very influential personality, ever so helpful and motivating. He showed incredible trust in me while I was working there. He has not only poured me with knowledge and experience about industry and commercial aspects of a research, he also shared his PhD journey and the hurdles that he had to overcome. I feel extremely lucky and honoured to

have known and worked with him. ‘Syed, you have a fan in me’- words that I will never forget.

I am grateful to Dr Eva Reindl and I thank her for everything she has done for me. Especially for being the emotional support during intensely distressing times. I understand how painful it must have been to proofread my thesis which is totally an alien field of study for her, and she did this twice.

Finally, special thanks to my parents. I cannot describe in words what do they mean to me, and what they have done for me. I only wish to see them happy, healthy and proud. Without the continuous help and support of these incredible people, this PhD would have remained only as a dream.

Research Thesis Submission

Please note this form should be bound into the submitted thesis.

Name*:*	Syed Rezwan Kabir		
School:	School of Energy, Geoscience, Infrastructure and Society		
Version:*(i.e. First, Resubmission, Final)*	Final	Degree Sought:	PhD (Civil)

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1. The thesis embodies the results of my own work and has been composed by myself
2. Where appropriate, I have made acknowledgement of the work of others
3. The thesis is the correct version for submission and is the same version as any electronic versions submitted*.
4. My thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
5. I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.
6. I confirm that the thesis has been verified against plagiarism via an approved plagiarism detection application e.g. Turnitin.

ONLY for submissions including published works

Please note you are only required to complete the Inclusion of Published Works Form (page 2) if your thesis contains published works)

7. Where the thesis contains published outputs under Regulation 6 (9.1.2) or Regulation 43 (9) these are accompanied by a critical review which accurately describes my contribution to the research and, for multi-author outputs, a signed declaration indicating the contribution of each author (complete)
8. Inclusion of published outputs under Regulation 6 (9.1.2) or Regulation 43 (9) shall not constitute plagiarism.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate*:*		Date:	06/10/2020

Submission

Submitted By *(name in capitals):*	SYED REZWAN KABIR
Signature of Individual Submitting:	
Date Submitted:	06/10/2020

For Completion in the Student Service Centre (SSC)

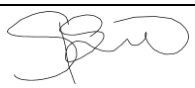
Limited Access	Requested	Yes		No		Approved	Yes		No	
1.1 E-thesis Submitted (**mandatory for final theses)**										
Received in the SSC by *(name in capitals):*						Date:				

Inclusion of Published Works

Please note you are only required to complete the Inclusion of Published Works Form if your thesis contains published works under Regulation 6 (9.1.2)

Declaration

This thesis contains one or more multi-author published works. In accordance with Regulation 6 (9.1.2) I hereby declare that the contributions of each author to these publications is as follows:

Citation details	Kabir, S., Patidar, S. and Pender, G. 2020. Investigating capabilities of machine learning techniques in forecasting streamflow. *Proceedings of the Institution of Civil Engineers – Water Management*,173-2, pp. 69-86.
Author 1	Designed methodology, developed models, performed experiments and wrote the paper
Author 2	Supervised the experiments and reviewed the manuscript
Author 3	Involved in designing the methodology
Signature:	
Date:	06/10/2020

Citation details	Kabir, S., Patidar, S. and Pender, G. 2020. A machine learning approach for forecasting and visualizing flood inundation information. *Proceedings of the Institution of Civil Engineers – Water Management*, https://doi.org/10.1680/jwama.20.00002
Author 1	Designed methodology, developed models, performed experiments and wrote the paper
Author 2	Supervised the experiments and reviewed the manuscript
Author 3	Involved in designing the methodology
Signature:	
Date:	06/10/2020

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure 7.9: (A) Probabilistic flood inundation map forecasted during growing phase using the proposed data-driven modelling framework; (B) deterministic flood map generated by FM software during growing phase; (C) probabilistic flood inundation map forecasted during receding phase using the proposed data-driven modelling framework; (D) deterministic flood map generated by FM software during receding phase.

LIST OF ABBREVIATIONS

ADI	Alternating Direction Implicit
ANFIS	Adaptive Neuro-Fuzzy Inference Systems
ANN	Artificial Neural Network
AO	Arctic Oscillation
ASCE	American Society of Civil Engineers
BNN	Bayesian Neural Network
BPNN	Back Propagation Neural Network
CEH	Centre for Ecology and hydrology
CGM	Coarse Grid Model
CNN	Convolutional Neural Network
CTUS	Central in Time and Upwind in Space
DBM	Data-Based Mechanistic
DBN	Deep Belief Network
DEM	Digital Elevation Model
DL	Deep Learning
DSM	Digital Surface Model
DWT	Discrete Wavelet Transform
EA	Environment Agency
ENN	Ensemble Neural Network
EPR	Evolutionary Polynomial Regression
FFBP	Feed Forward Back Propagation
FFC	Flood Forecasting Centre

FGM	Fine Grid Model
G2G	Grid-to-Grid
GA	Genetic Algorithm
GAM	Generalized Additive Models
GIS	Geographical Information System
GFS	Global Forecasting System
GP	Genetic Programming
GRDC	Global Runoff Data Centre
HEC-HMS	Hydrologic Engineering Centre–Hydrologic Modelling System
HMM	Hidden Markov Model
HRU	Hydrological Response Units
IDE	Interactive Development Environment
IHIP	Intelligent Hydroinformatics Integration Platform
IPE	Ideal Point Error
KNN	K-Nearest Neighbours
LETM	Landsat Enhanced Thematic Mapper
LRN	Layer Recurrent Network
LSSVR	Least Square Support Vector Regression
LSTM	Long-Short Term Memory
MARE	Mean Absolute Relative Error
MARS	Multivariate Adaptive Regression Splines
MIDAS	Met Office Integrated Data Archive System

MIMO	Multiple-Input-Multiple-Output
MISO	Multiple-Input-Single-Output
ML	Machine Learning
MLP	Multi-layer Perceptron
MLR	Multiple Linear Regression
MNLR	Multiple Non-Linear Regression
MT	Model Trees
NOAA	National Oceanic and Atmospheric Administration
NSE	Nash-Sutcliffe efficiency
PCA	Principal Component Analysis
PDM	Probability Distributed Model
PNA	Pacific-North American
PSO	Particle Swarm Optimisation
RBM	Restricted Boltzmann Machines
RF	Random Forests
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SAC-SMA	Sacramento Soil Moisture Accounting
SAR	Synthetic Aperture Radar
SCE-UA	Shuffled Complex Evolution
SCS	Soil Conservation Service
SDSM	Statistical Down Scaling Model
SNN	Single Neural Network

SOM	Self-Organizing Map
SRM	Structural Risk Minimization
SRTM	Shuttle Radar Topography Mission
SSA	Singular Spectrum Analysis
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SVR	Support Vector Regression
SWAT	Soil Water Assessment Tool
SWE	Shallow Water Equation
SWMM	Storm Water Management Model
TVD	Total Variation Diminishing
WANN	Wavelet- Artificial Neural Network
WT	Wavelet Transform

LIST OF PUBLICATIONS

This thesis contains contents from the following published journal articles-

[1] Kabir, S., Patidar and Pender, G. (2019) Investigating capabilities of multiple machine learning techniques in forecasting streamflow for UK catchments. Proceedings of the Institution of Civil Engineers- Water Management. https://doi.org/10.1680/jwama.19.00001.

[2] Kabir, S., Patidar, S. and Pender, G. (2020a) A machine learning approach for forecasting and visualizing flood inundation information. Proceedings of the Institution of Civil Engineers- Water Management. https://doi.org/10.1680/jwama.20.00002.

Chapter 1 **Introduction**

> *Given the meagreness of our intelligence in comparison with the complexity and subtlety of nature, if we want to say things which are true, as well as things which are useful and things which are testable, then we had better relate our bids for truth, application and testability in some fairly sophisticated ways. This is what modelling does.*
>
> *Suarez and Morton (2001)*

1.1 Research background

In various fields of hydrology, two-dimensional (2D) hydrodynamic models are studied and used across a wide range of applications. These models are designed to capture - as precisely as possible - complex physical/natural processes, while incorporating both the mass and the momentum conservation, for providing a considerably good description of the various hydraulic processes occurring in the area under investigation (Liu and Pender, 2015). Recent advancements in technology and availability of key remotely sensed data, such as surface elevation and river morphological parameters in data-sparse areas, have created new potentials that facilitate the implementation of hydrodynamic models from regional to global scales in real world applications (de Paiva et al., 2013, Yamazaki et al., 2011). In spite of these developments, there are some key issues that need urgent attention, for example, improving computational efficiency: conventional hydrodynamic simulation on a standard PC needs several hours to complete one simulation run. In addition, expert knowledge, skills, and detailed data/information are required to calibrate the model (Altenau et al., 2017). One possible solution to tackle some of these limitations which are associated with the traditional approach is to exploit the potentials of the highly efficient parallel computing facilities, but conclusively it has been found that such approaches request additional utility costs and expertise (Liu and Pender, 2015). Also, high performance computing facilities are mostly available in developed countries. Therefore, in practice, real-time flood forecasting is mostly limited to the generation of discharge hydrographs at specific gauges without the simulation of 2D hydrodynamic models (Bhola et al., 2018).

To address some of these challenges, an efficient flood forecasting system is required that could provide information on certain key variables, such as water depths, velocities, flood

arrival times, and inundation maps within a suitable lead time to allow decision makers to plan and take action (Leedal et al., 2010). Within the context of deterministic forecasting models, providing all this information accurately and consistently for all the variables and for long lead times is challenging. This is mainly due to the high sensitivity of the catchment system to the initial conditions and the limited ability of the physically based deterministic models that involve complex meteorological simulations to accurately forecast all these key variables (Leedal et al., 2010).

Conventionally used 2D flood inundation models transform the bulk river discharge into potential flood hazards, specifying the flood extent, depth and velocity. Thus, they provide most of the information required by the decision makers working in the areas of flood risk management (Bates et al., 2014). However, for the purpose of operational flood forecasting, hydrological modelling-based approaches are widely applied. These approaches can provide discharge hydrographs and mostly do not involve the simulation of 2D hydrodynamic models (Bhola et al., 2018). This is mainly because the 2D hydrodynamic models, in general, have a complex model structure with high parameter dimensionality, and thus requires a considerably high computational time for producing dynamic real-time flood maps.

In recent years, researchers have developed several computationally efficient numerical models as an alternative for adaptive water level forecasting at gauging stations (Young, 2006) For example, the forecasting performance of the Data-Based Mechanistic (DBM) approach has been investigated at several gauging stations along the river Severn, UK, reach and showed promising results for different lead times (Romanowicz et al., 2008, Romanowicz et al., 2006). The DBM model was further exploited by Leedal et al. (2010) for visualization of inundation information in real-time. In their research, a DBM model was coupled with a simplified 2D hydrodynamic model (LISFLOOD-FP) to generate 6h ahead flood inundation forecasting. To further reduce the computational time for the simulation of 2D hydrodynamic models during a real-time operation, some offline methods have also been proposed (Bhola et al., 2018, Henonin et al., 2013). In an offline method pre-recorded inundation maps are stored in a database for a multitude of scenarios and an appropriate map is queried using a discharge forecast at upstream gauge during a flood event (Bhola et al., 2018, Leedal et al., 2010). Using such a system does not require live 2D hydrodynamic simulations since the maps are prepared beforehand. However, such a method has some limitations, for example: i) it requires multiple platforms working

in parallel; ii) generating scenarios to build the large databases of probabilistic maps could be labour intensive and demand large storage capacities; iii) the infrastructure requires regular maintenance (hardware), which could also be a major issue in operational applications; and iv) the database needs to be updated for any topographical changes, especially for urban areas.

Another widely applied approach for generating flood extent maps is through the satellite image processing. Optical satellite images (e.g. Modis, Landsat, Sentinel-2) are often used for producing validation results for hydraulic models or near real-time flood inundation mapping. However, these images are mainly constrained by the satellite revisit time and are often impacted by the cloud covers over the region, which makes them a less feasible method for real-time applications. On the other hand, radar images do not suffer from clouds and other weather conditions, and are available for both day and night, making them a more suitable candidate. The recent constellation of Sentinel-1 satellites and Synthetic Aperture Radar (SAR) from other platforms have made imageries readily available for extraction of flood extent in near real-time (Clement et al., 2018, Mason et al., 2014). However, processing of SAR images for delineating water bodies in the urban areas can be extremely difficult due to layover and shadow caused by infrastructures and trees (Mason et al., 2014). In recent years, the German Remote Sensing Data Centre (DFD) has developed the so-called DFD Water Suit which is a package of four distinct algorithms for extracting water surface from SAR images. These water surface extraction schemes are: Water Mask Processor (WaMaPro), Rapid Mapping of Flooding (RaMaFlood) tool, TerraSAR-X Flood Service (TFS) and TanDEM-X Water Indication Mask processor (TDX WAM) (Martinis et al., 2015, Huth et al., 2015, Martinis et al., 2009). Specifically, SAR-based methods can be used to map flood extent and can be further utilised to derive water levels in near real-time using Digital Elevation Model (DEM) (Brown et al., 2016). Although extracting inundation extent through SAR images is computationally less demanding, these approaches are restrained by the satellite overpass (revisit time). That is to say, SAR images are not available at all times during a flood event. It may take several days (three or more days depending on satellite platforms) to acquire a new image from the last available product. This particular feature makes the SAR image-based flood mapping an inefficient technique, specifically in the areas where water accumulates and drains quickly.

These various issues involved with the different methods used for real-time flood inundation mapping (as discussed above) could potentially be minimized through the application of machine learning (ML) methods. These approaches harness the potential of growing volumes of available data to learn input-output patterns of a system and facilitate data-driven decision making (Witten et al., 2011). ML techniques are known to offer an effective environment for analysing and modelling dynamic, non-linear, and noisy data, especially when the underlying physical relationships cannot be fully understood due to the complexity of the system/processes (Nourani et al., 2011). However, most of the previous data-driven modelling approaches in hydrology have been designed to define rainfall-runoff relationships and/or other hydrological processes such as measuring soil water content (Wu et al., 2008b), suspended sediment load prediction (Melesse et al., 2011) or groundwater modelling (Sahoo et al., 2017). The potential of ML as a tool for real-time flood inundation mapping is yet to be explored systematically. Possibly, Liu et al. (2009) made the first attempt to introduce ML techniques with an intention to improve performance of fast flood inundation models. The concept of fast inundation models relies on replacing time-consuming hydrodynamic models with a simplified model structure to reach an acceptable solution without significantly losing the accuracy of model outputs (Liu and Pender, 2015). The methodology proposed by Liu et al. (2009) investigates the possibility of exploiting the potential of Support Vector Regression (SVR) in association with a Coarse Grid Model (CGM) for the purpose of predicting the outputs of an inundation model. A computational framework similar to Liu et al. (2009) was applied by Liu and Pender (2015) to a site (Aldeburgh Marshes, UK) which is susceptible to tidal flooding to predict water depth and velocity. Chang et al. (2018a), Chang et al. (2018c), Chang et al. (2018b), Chang et al. (2014b), Shen and Chang (2013), Chang et al. (2010) have also shown that one could possibly apply hybrid ML techniques not only to forecast/estimate water level/discharge at gauging points along the channel but also for hydraulic modelling. This interesting feature has remained unexplored for years, but could potentially set a new paradigm in the field of real-time flood inundation of ML.

However, while ML may have the potential to be exploited for inundation modelling, this method has to be rigorously tested to be considered as a fast, reliable and robust method suitable for practical applications. This step is essential, specifically, to address the concerns raised by the hard-core hydrological modelling research communities who relies

on the conventional physically based hydrological modelling, and can be benefitted from the advantages of the data-driven models: -

> *"Unfortunately, hydrological modellers emphasise the merits of their own approaches while discarding those of others. In particular, physical process-oriented modellers have no confidence in the capabilities of data-driven models' outputs with their heavy dependence on training sets, while the more system engineering-oriented modellers claim that data-driven models produce better forecasts than complex physically based models. Implicit in this controversy is the basic question: should 50 years of research by scientists seeking better representations of hydrological processes be jettisoned?"* (Todini, 2007).

Thus, there is a timely need for reconciliation of these two streams of modelling communities (physically based and data-driven). This can be achieved by fusing advantages of both forms of modelling techniques, defining their roles and determining their fields of application.

In the context of the UK, simple regression-based models to physically-based hydrodynamic models are applied to design systems for flood risk management. Currently, both the Flood Forecasting Centre (FFC), England and Wales (Price et al., 2012), and the Scottish Flood Forecasting Services use Grid-to-Grid (G2G) as a countrywide flood forecasting system to provide six hour to five days ahead forecasts (Pilling et al., 2014, Price et al., 2012). The G2G is a physical–conceptual distributed hydrological model developed by the Centre for Ecology and Hydrology (CEH), Wallingford. Model relies on solving simple depth-integrated formulations for producing runoff and flow routing, thus making G2G model computationally efficient compared to the complex hydrodynamic models and allowing for the G2G model to run within a 1km grid for nationwide real-time flood forecasting. However, the performance of the G2G model varies spatially and on average, as one would expect, the local models such as the Probability Distributed Model (PDM) (Moore, 2007) outperform G2G. A comprehensive review on the wide range of physical models used in the UK's Environment Agency (EA) for flood forecasting can be referred elsewhere Bell et al. (2001).

1.2 Rationale

1.2.1 Research question

Previous section of this chapter presented: i) an overview of the different ML algorithms for streamflow and inundation extent forecasting, ii) key research gaps in the application areas of ML techniques for generating real-time inundation information, and iii) limitations/drawbacks associated with current approaches. The limitations associated with the existing approaches used in real-time flood forecasting provides a strong motivation for the PhD research presented here, which also acted as a testbed for the evaluation of new novel hybrid data-analytic based modelling techniques for the real-time operational flood forecasting. Considering that the potential for ML-based approaches in real-time inundation modelling is immense, this thesis attempts to answer the big research question: whether it is possible to develop a novel ML driven end-to-end (rainfall-inundation) modelling system for visualising inundation information, which would utilize the advantages of state-of-the-art 2D hydraulic models while accounting the limitations/drawbacks associated with the previous ML-based approaches.

1.2.2 Aim and objectives

The overarching aim of this research project is to develop a computationally efficient and robust data-driven ML based end-to-end (rainfall to inundation) modelling engine on a single platform as an alternative to the computationally intensive physically based hydraulic/hydrodynamic models for operational flood inundation mapping. To achieve this goal, key objectives of this project has been defined, and are listed below:

1. To conduct a thorough literature review of widely applied ML algorithms, in parallel to the hydrological/hydraulic models, and their potential application in the areas of operational flood-forecasting and inundation mapping (Chapter 2 and 3).
2. Investigate capabilities of multiple ML algorithms for identifying and developing a suitable ML-based model for forecasting multi-hour ahead stream flow using observed rainfall data (Chapter 4).
3. To develop a novel ML based modelling approach for seasonal flooding using observed river flow and outputs from the LISFLOOD-FP hydraulic model for a large natural catchment (Chapter 5).

4. To develop a novel ML based modelling approach for classifying inundated cells within a semi-urban floodplain during an extreme flood event using outputs from Flood Modeller software, observed and synthetic flow hydrographs (Chapter 6).
5. To develop an integrated modelling framework for producing high resolution dynamic probabilistic flood maps from rainfall data using ML and spatial interpolation technique (Chapter 7).

1.2.3 Dependencies

This subsection lists the dependencies (e.g. software, data) required for conducting the individual tasks to meet the research objectives.

- Hydro-meteorological data – It is not possible to develop data-driven models without a large collection of hydro-meteorological data sets. Rainfall and flow time-series were collected from multiple sources; the specific data sources are described in the relevant chapters.

- Study area selection – Unfortunately, there were not many options available to select study areas due to the lack of data availability. Hourly rainfall and flow data available for three UK catchments were used to develop the ML-based models in Chapter 4. Multichannel flow hydrographs with additional topographical data, are available for one watershed. Hence, this dataset was used to develop the inundation classifiers in Chapter 5. High-resolution spatial dataset and inflow hydrograph are available for a small town (Upton-Upon-Severn) in the UK, these datasets were used to build the models in Chapter 6 and 7.

- The selection of ML-based models was based on the following criteria: i) to what extent these models had been successfully applied in previous studies, ii) model simplicity, and iii) representative of a standard, a hybrid and a DL method.

- Hydrodynamic models: a raster-based simplified 2D hydraulic model (LISFLOOD-FP) and a full commercial 2D model (Flood Modeller) were used to generate water depth values for inundation modelling.

- *R* and *Python* programming language were used predominantly to implement the models, and MATLAB was utilized for wavelet decomposition.

- Geostatistical modelling and spatial interpolation techniques were used to estimate the spatial distribution of the inundated cells.

1.3 Thesis outline

This section describes the structure of the PhD thesis with a short synopsis of the individual chapters.

Chapter 1 is intended to provide a context for the PhD project to define aim and objectives. The chapter presents a background to the topic along with an overview of the key challenges associated with the existing real-time flood forecasting approaches. It defines the scope for the proposed ML based modelling system in fast-tracking real-time flood inundation and visualisation.

Chapter 2 gives an introduction to various widely applied ML methods along with a brief description of key techniques applied in the present research. The PhD project is interdisciplinary, and 2D hydraulic modelling is a vital component of this work. Therefore, this chapter also provides a brief introduction to hydrological cycle and hydrologic/hydraulic modelling approaches.

Chapter 3 reviews some of the major, relevant and recent studies conducted in this area. This chapter is intended to present a comprehensive review of the key research that have used different ML techniques to forecast streamflow sequences and real-time flood inundation modelling.

Chapter 4 describes the development procedure of three widely used ML techniques and conducts an intensive investigation to assess the capabilities of these approaches in forecasting 1 to 6-hour ahead streamflow. The model has been applied across three catchments within the UK. The chapter aims to identify the best performing technique (among the three investigated) with ability to forecast streamflow using antecedent rainfall and flow data with greater accuracy, and specifically without requiring changing the model parameters. Further, model uncertainty (by means of residual analysis), performance deterioration rates, portability and scalability for each modelling technique moving forward in time has been thoroughly investigated.

Chapter 5 describes the development of a novel ML based modelling approach for inundation extent mapping. The ML-based models have been developed and tested to generate dynamic flood maps at daily time steps for a natural catchment. The chapter presents details of the training and validation procedures along with the key outcomes.

Chapter 6 presents the development procedure of an ML based modelling approach to produce dynamic inundation maps for a semi-urban floodplain. Key development stages have been detailed, that involving: i) generation of multiple synthetic flood hydrographs for training and validation purposes; ii) ML-based model configuration for producing binary flood extent maps with higher temporal (hourly) and spatial (30m) resolution.

Chapter 7 describes the step-by-step development procedure of modelling framework that utilises hourly rainfall data for forecasting real-time flood inundation extents. The modelling framework integrates a non-linear regression based hydrological model, a computationally efficient wet/dry cell classification module, and a data visualisation schematic. This chapter demonstrates the application of the novel computationally inexpensive ML based approach in simulating flood-extents directly from the rainfall data input.

Chapter 8 summarises the results of the different analytical experiments conducted in Chapters 4 to 7 and makes concluding remarks on major outcomes from the thesis. It also highlights the key limitations of the research conducted and the recommendations for the future development opportunities.

Finally, this research has contributed to two peer reviewed journal publications. Table 1.1 provides a brief about how the aim and objectives are addressed in these publications.

Article	Objectives	Chapter reference	Comments
Kabir et al. (2019)	2	3	This paper presents a comparative study of three ML-based models in streamflow forecasting and investigates the transportability of model parameters across time and space.
Kabir et al. (2020a)	5	7	Proposed the ML-based rainfall-inundation modelling framework. The framework is built upon the knowledge gained from objective 2, 3 and 4.

Table 1.1: Research objectives and knowledge contribution.

Chapter 2 **Machine Learning and Hydrological Modelling**

> *Every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves.*
>
> *Macarthy, J., in the opening statement of Dartmund conference, 1956*

2.1 Introduction

With the changing climate, flooding is becoming a major environmental challenge for the UK and the rest of the world. To explore the dynamics of complex hydrological process that contributes to flooding, different hydrologic and hydraulic impact models are available (Liu and Pender, 2015). A comprehensive investigation of flood-risk assessment essentially includes flood inundation models which simulate the outputs generated from the hydrological models. However, as discussed in Chapter 1, most of the hydraulic modelling procedures are computationally intensive, thus making their application practically challenging for performing real-time simulations. Due to different constraints involved in calibration of traditional hydrological/hydraulic models (e.g. scarcity of data on catchment characteristics, physical knowledge of catchments, lack of expertise, etc.) there is now growing interest in developing effective ML based models. Recent studies (Kourgialas et al., 2015, Chang et al., 2014a, Chen et al., 2013) have shown advantages of using ML based models over conventional hydrologic models. A thorough review of some of these approaches to forecast different hydrological variables, e.g. runoff, streamflow, flood water depth, and flood extent has been presented in Chapter 3.

However, before reviewing the applications of ML in the context of hydrological forecasting, it is imperative to have a theoretical understanding of ML and catchment hydrological system modelling. ML and hydrodynamic modelling are the two most critical components of this research. Therefore, the objective of this chapter is to provide a comprehensive overview of the key concepts and background information necessary to appreciate the interdisciplinary aspects of the present thesis that involves a ML based flood inundation modelling framework.

This chapter outlines the concept of ML and the theoretical basics of key ML techniques that have been investigated in the thesis. This includes: Artificial Neural Networks (ANN), SVR, Deep Belief Networks (DBN) and Random Forests (RF), detailed in the **Sect.** 2.2-2.3, respectively. Section 2.4 presents an overview of the key catchment hydrological processes, different types of widely applied hydrological and hydrodynamic modelling scheme for flood water propagation.

2.2 What is machine learning?

ML is a branch of computer science, dedicated to the development and applications of self-learning algorithms that find insights into large volume of data (Raschka and Mirjalili, 2017). The term ML is closely related to the term AI. While AI is the broad science of mimicking human behaviour, ML is a subset of AI that involves learning from data, identifying patterns, and making predictions.

Fundamentally, ML refers to a set of linear and non-linear approaches for estimating the function *f(.):*

$$Y = f(X) + \in \tag{2.1}$$

That best describes the true underlying relationship between the input variables *X* (also labelled as independent variables, exploratory variables or predictors) and the output variable Y (also termed as response or target variable) in a given dataset, with $\in$ representing a random error term. The estimated function parameters can then be used for predictive analytics or inferential analysis, i.e. predicting the output variable for new input variables.

2.2.1 Types of machine learning

This section describes three major ML techniques - supervised learning, unsupervised learning and reinforcement learning - and their fundamental differences (**Fig.** 2.1).

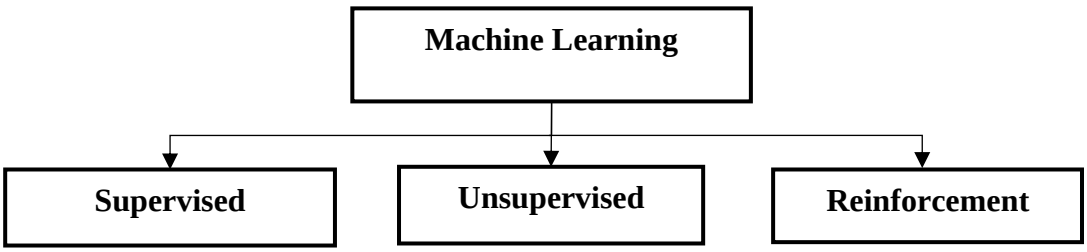


Figure 2.1: Machine learning methods.

2.2.1.1 Supervised learning

Supervised learning is the most popular method applied in data science. In a supervised learning method, the ML-based model learns from labelled training data. The term 'supervised' means that the desired output signals (labels) are predefined in the training samples. (**Fig.** 2.2).

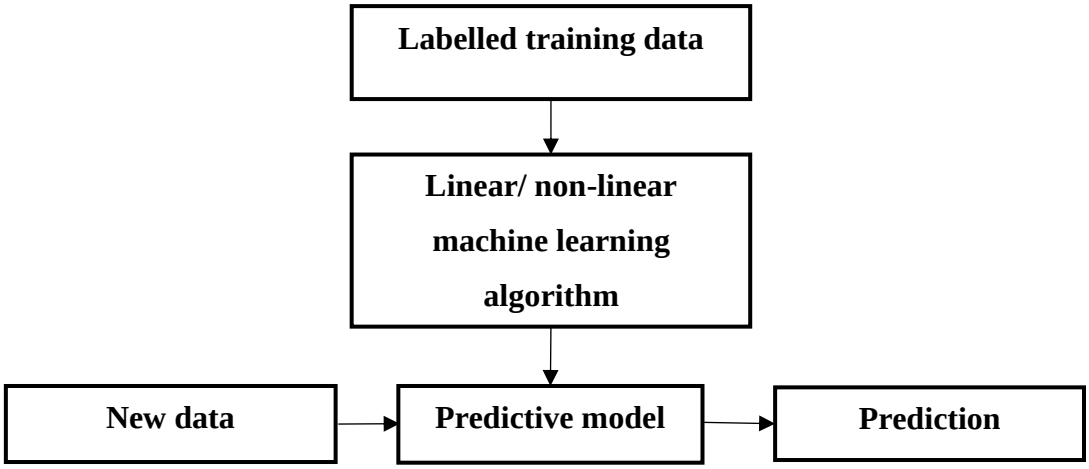


Figure 2.2: Supervised learning mechanism.

Such predictive models can be applied to solve both classification and regression-based problems. In classification problems the goal is to predict categorical classes/labels of unseen data (for which the model was not trained) based on past observations. Spam email detection and face recognition are some of the examples of classification problems solved by the supervised learning method.

Regression problems involve prediction of continuous outcomes. In regression analysis, a fitting relationship between predictor and response variables is obtained from sample data (training data) that allows one to predict an outcome in the future. Some relevant examples include, predicting river flow sequences (dependent variable) using antecedent rainfall, flow and temperature profiles as independent input variables, predicting height (dependent variable) using age, gender and diet as independent variables.

2.2.1.2 Unsupervised learning

In unsupervised learning the labels or structure of the data is unknown. Unsupervised learning is applied to infer a function that explores the hidden structure/patterns of the unlabelled data. This particular method of learning describes the somewhat more

challenging state of affairs in which for every observation there is a vector of measurements X_i but no associated response Y_i. Hence it is not possible to fit a regression model, since there is no response variable to train the model on. It leads to the fact that with unsupervised learning - since the model is unaware of the labels of the training data - there is no way for it to measure its accuracy/error. Therefore, unlike supervised learning, in unsupervised learning- accuracy (classification problems) or error statistics (regression problems) of the model cannot be used as metrics to assess model performance.

Essentially, with unsupervised learning the model is being given the unlabelled dataset and the model will attempt to learn some type of structure from the data and extract useful information or features from it. The model will aim to create a mapping from the given set of inputs to generate outputs based on what it is learning about the structure of this data without any labels. One of the most popular applications of unsupervised learning is the clustering algorithms. Figure 2.3 illustrates how clustering can be applied to group males and females based on the similarity of their height and weight features.

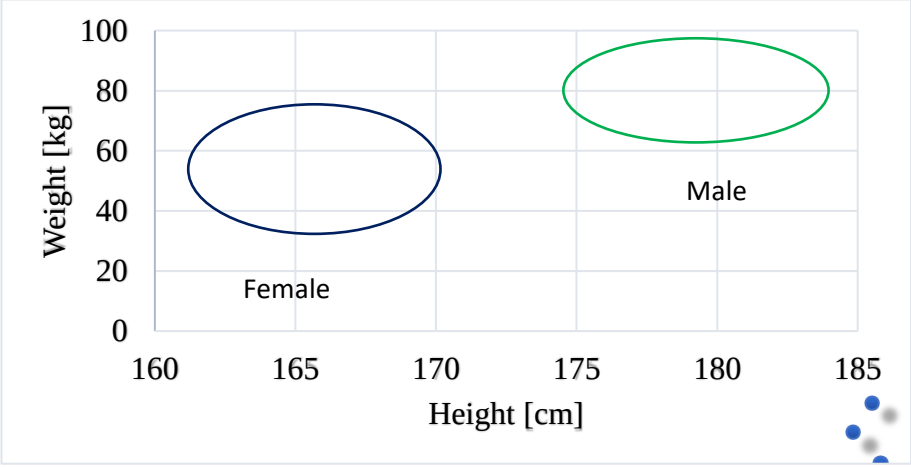


Figure 2.3: Clustering can be used to organize unlabelled data into groups.

2.2.1.3 Reinforcement learning

Another subfield of ML is reinforcement learning. It is mainly concerned with the foundational issue of how a system (agent) can learn from interactions with the environment to make a good sequence of decisions. It involves an autonomous agent learning to navigate an uncertain environment with the goal of maximizing a numeric reward. The agent needs to follow a set of rules and strategies in order to maximize the reward. The agent's actions change the state of the environment. Therefore, a model

would need to be able to take a state and action as inputs and generate the maximum expected reward as output. Since this only gets the model to the next state, one needs to take into account the total expected reward for every action from the current till the end state.

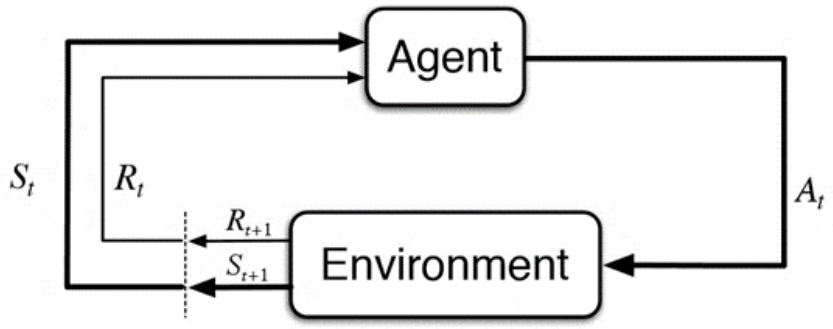


Figure 2.4: A general reinforcement learning scheme.

A general scheme of reinforcement learning is shown is Figure 2.4. Each state S_t is associated with a reward R_t, which can be positive or negative, and the agent then uses reinforcement learning to learn actions A_t which maximize the reward through interaction with the environment.

Reinforcement learning can be somewhat related to supervised learning as information about the current state of the environment also includes a reward signal. However, supervised learning is concerned with making sense of the environment based on historical examples whereas reinforcement learning is a measure of reward. An agent can gain rewards for every correct action it makes or can also lose rewards for any wrong action. The objective here is to maximize the rewards given the current state of an environment.

This section is intended to provide a very basic overview of different ML techniques. Detailed overview of various ML and data mining techniques with examples focusing on regression, classification and clustering can be referred elsewhere Raschka and Mirjalili (2017). Further, specific application of reinforcement-based learning can be referred elsewhere Mnih et al. (2013).

2.2.2 Machine learning life cycle

In the previous sections, the basic concepts of ML and the three types of the learning methods were briefly discussed. This section attempts to present different stages involves in a complete ML system development (please refer to **Fig.** 2.5).

The first stage in ML system designing is to define the problem (classification, regression etc.). Often, we assume that a problem comes predefined, but it is important to acknowledge the limitations and assumptions that one could make with any problem definition. The next stage is to identify and acquire essential dataset to develop a robust and reliable model. This step involves identifying key independent variables, suitable temporal resolution of dataset, and sufficient length/duration of dataset for training, testing and validation. In addition to data acquisition, it is important to ensure overall quality of the data used for model development. Data quality could have significant impact on the overall efficiency/reliability of the outputs generated from the ML-based models. Other important aspects, such as feature selection, data labelling, data transformation, scaling- often termed as data pre-processing, should be carefully considered to ensure a robust ML-based model development.

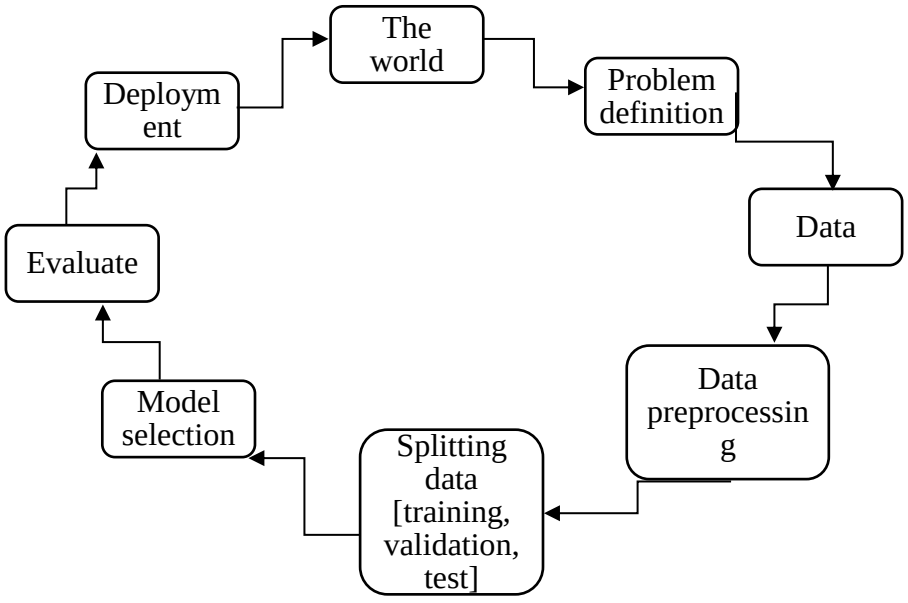


Figure 2.5: A machine learning life cycle.

Next stage to the data pre-processing is data segregation, i.e. to split the data appropriately into training, validation and testing subsets. Usually, the majority of the data is used for training purpose. Validation data are separate than the training data and are used during

the development procedure to check the model performance. Testing data are used for final evaluation of the model performance. The data splitting stage ensure that the ML-based model performs reasonably well for the data other than the training dataset. The next stage is model selection that involves training and testing of different ML-based model forms to identify and select suitable model structure with abilities to compound effects of data intricacies. Different models form requires different optimization procedures, for example, neural networks use back propagation method to estimate model parameters. Model selection procedure is primarily based on evaluating overall model performances with the validation data. Once the suitable model structure is selected, the overall model performance is evaluated by assessing model's ability in generating outputs from the test dataset. To assess overall model performance, it is important to select an appropriate performance evaluation metrics. For classification-based problems, a confusion matrix is derived that defines the classification accuracy of the ML-based models by estimating false positive and false negative rates. For regression-based problems, a wide-range of error metrics are used (e.g. root mean squared error, mean squared error, coefficient of determination etc.). A vast amount of literature is available that discusses the different applications, advantages and effect of error metric selection on model efficiency.

2.3 Machine learning algorithms

This section presents a brief overview of key ML algorithms that have been used as part of the PhD project to develop different modelling schemes presented in the following chapters.

2.3.1 Artificial Neural Networks

ANNs, conceptually inspired by biological neurons, are one of the widely applied ML technique. ANNs are adaptive and are capable of dealing with data heteroscedasticity, i.e. they can process highly nonlinear and noisy data effectively.

The Multi-layer Perceptrons (MLP), most common form of ANN, are consist of several weighted connections between nodes and are arranged in three different layers: *input layer*, *hidden layer* and *output layer* (**Fig.** 2.6). The processing units (neurones) in the hidden and output layers multiply the input signals propagating through the layers with a set of weights from the weight matrix and transform it into the overall system output/response using an *activation function.*

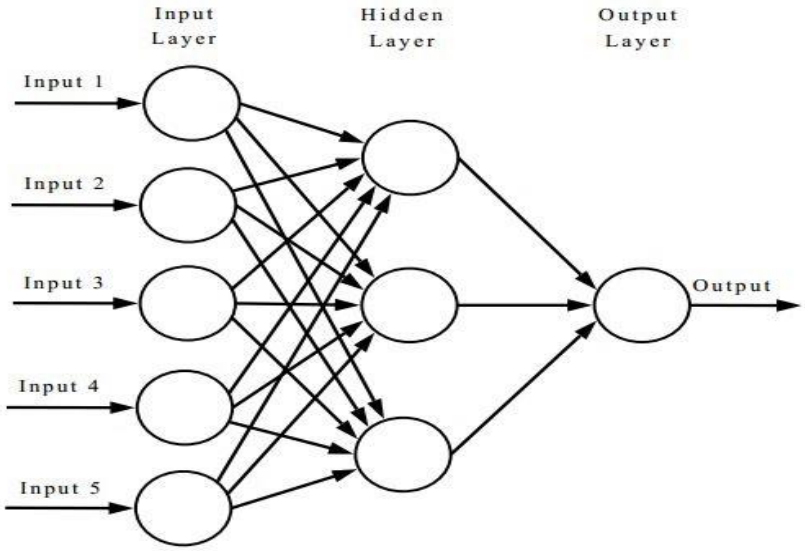


Figure 2.6: General architecture of a multi-layer perceptron.

The activation functions are differentiable functions and depending on the problem under investigation these functions could be both linear and non-linear in nature (Hagan et al., 2002). Activation function maps the response of a neuron for a given set of input variables. A typical neuron in a hidden layer, presented in Figure 2.7, receives signals from all preceding nodes/neurons. For example, the output *y* of the *j*-th neuron of a hidden layer can be expressed as:

$$y_j = g(b_{0j} + \sum_{i=1}^{n} x_i w_{ij}), j=1, 2, 3, \ldots, N_{hidden} \quad (2.2)$$

A range of activation functions, *g (·)*, are available such as sigmoid, linear, hyperbolic tangent, rectified linear units, etc. Although, there are no particular rules or selection criteria, a suitable form of activation function is selected that mainly depends on the problem under investigation to meet the requirements of the system. The weight *w* and bias *b* are adjustable scalar parameters. Following an iterative approach, known as *learning rule*, optimum values for weight and *bias* are estimated to minimise overall error measurement.

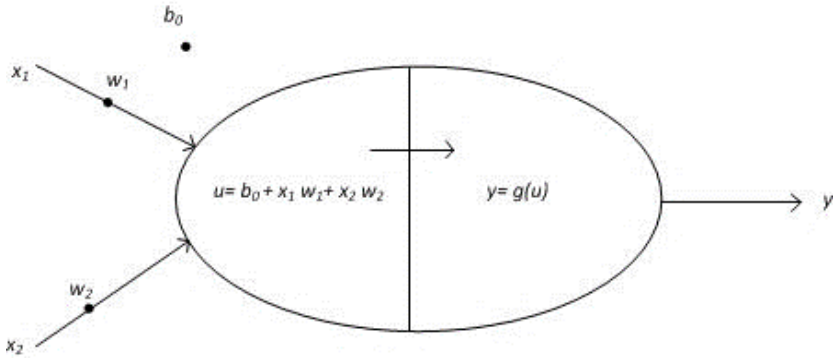


Figure 2.7: A processing unit in a hidden layer with two inputs.

An iterative procedure that computes the network error (**Eq.** 2.3) repeatedly with respect to the target values with a given set of input vectors and modifies the weight and bias values according to the *learning rule* is referred as *training* of neural network (**Fig.** 2.8). The training process is continued until a stopping criterion is achieved (error goal, and/or a maximum number of epochs), and the *least squared* method is generally used to express the system error *E*.

$$E = \frac{1}{2}\sum_k (f_k - z_k)^2 \qquad (2.3)$$

Where, f_k is the observed output vector and Z_k is the model estimate from a same set of input vector.

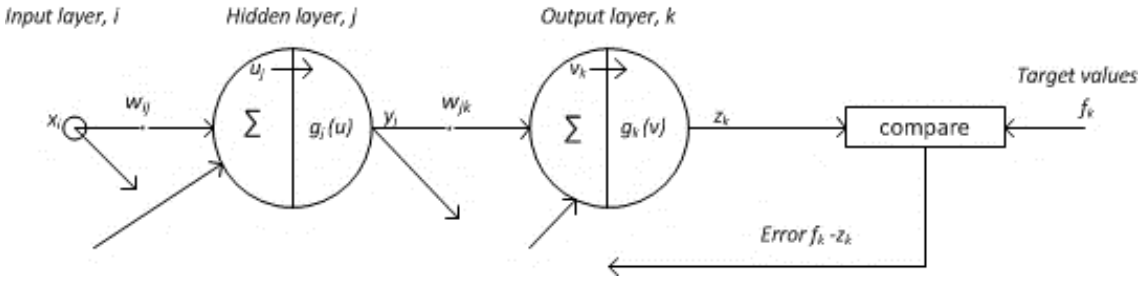


Figure 2.8: Training the network through minimizing estimated error.

A more detailed explanation of the learning rules and other forms of artificial neural networks can be referred in Hagan et al. (2002), Fausett (1994).

2.3.2 Deep Belief Networks

The network structure of DBN resembles to a feed-forward neural network (the connections between the layers do not form a cycle, for example- MLPs) with many hidden layers. Typically, hidden layers of a DBN are organised as a stack of Restricted Boltzmann Machines (RBM), which are used as building blocks for layer-wise unsupervised learning (Hinton et al., 2006).

RBM, first introduced by Hinton (2002), is an undirected graphical model that can automatically find patterns in the data by reconstructing the input. It is a shallow two-layer network, where the first layer *v* is called visible layer and the second layer *h* is known as the hidden layer. All units in the visible layer are connected to the stochastic hidden units of the hidden layer using undirected weighted connections (Teh and Hinton, 2000). However, there are no connections within the hidden units or the visible units (**Fig.** 2.9). The weights and biases are estimated within the RBM structure by establishing the interrelationship between the input features and identifying underlying patterns. Mathematically, the model defines a probability distribution over *v* and *h* units by means of an energy function *E*:

$$E(v,h|\theta) = -\sum_{m=1}^{V}\sum_{o=1}^{H} w_{mo} v_m h_o - \sum_{m=1}^{V} b_m v_m - \sum_{o=1}^{H} a_o h_o \qquad (2.4)$$

Where, $\theta = (w, b, a)$ is the parameter set, w is the bi-direction weight between the units of hidden layers $l-1$; b *and* a are the corresponding biases; V and H represent the number of the units in *v* and *h* respectively.

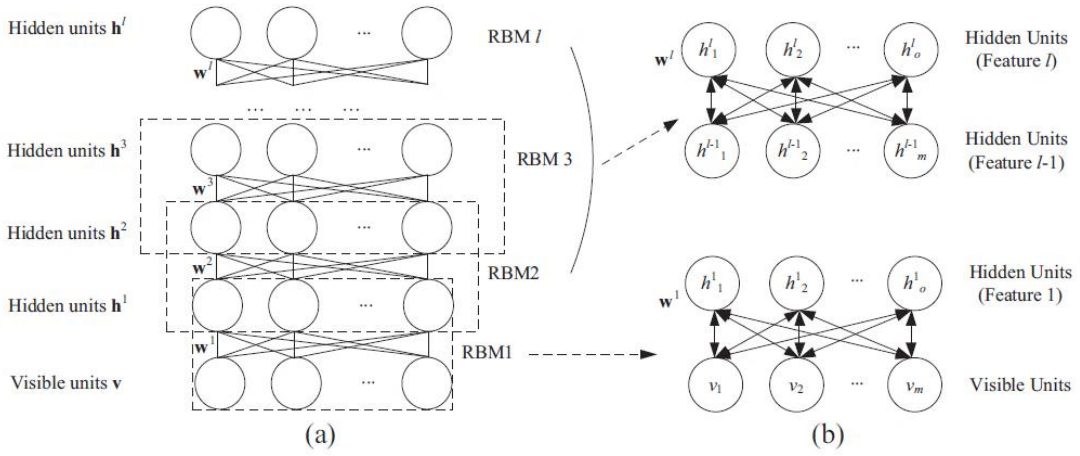


Figure 2.9: (a) A DBN structure with *l* hidden layers. (b) RBM structures. The higher state RBM is trained by the output of lower-level RBM (Bai *et al.*, 2016).

For binary state nodes, i.e. $v_m, h_o \in [0,1]$, the conditional probability distributions *P*, given when v_m or $h_o = 1$, are defined as follows:

$$P(h_o = 1|v) = Sigm(\sum_{m=1}^{H} w_{mo} v_m + a_o) \quad (2.5)$$

$$P(v_m = 1|h) = Sigm(\sum_{o=1}^{H} w_{mo} h_o + b_m) \quad (2.6)$$

Where, $Sigm(\cdot)$ is a sigmoid function. The overall aim of the training procedure of a DBN-based model is to train the probabilities of visible units defined in **Eq.** 2.7, conditioned to maximise *P(v)*. To solve this problem, Hinton (2002) proposed a contrastive divergence algorithm, which is briefly described below:

a) Initialize the value of v using the input data and then compute the value of h using the conditional probability distributions given in **Eq.** 2.5;

b) Using the estimated value of *h* (in step a), estimate the reconstruction state v' using **Eq.** 2.6; and

c) Repeat **Eq.** 2.5 to update the value of hidden nodes using the estimated values of v'(in step b) to estimate the value of h'.

Updated values of weight can be calculated using the following equation:

$$\Delta w_{mo} = \mu(\langle v_m h_o \rangle - \langle v'_m h'_o \rangle) \quad (2.7)$$

Where, μ is the learning rate, and $\langle \cdot \rangle$ represents the expectation of the training data.

Several RBMs can be stacked together to construct a DBN. The training procedure of a DBN involves the following steps:

Step 1) the first RBM in the stack is trained (following the procedure detailed above) to reconstruct input values.

Step 2) the second RBM is trained using the results obtained in the first (the hidden layer of the first RBM is treated as the visible layer for the second RBM).

Step 3) the training procedure (step 1 and 2) is continued until all the RBM layers of the DBN are trained.

Once all the layers are trained, the network of stacked RBMs can detect the inherent patterns of the input data. In the final stage of training, the target values/labels are

introduced to the detected patterns and a supervised learning procedure is followed to fine-tune the model. Please refer to Hinton (2012) for a comprehensive guide on the training procedure of RBMs/DBN.

2.3.3 Support Vector Machines

SVM, introduced by Vladimir Vapnik and Alexey Ya in 1963, is a novel artificial intelligence-based method developed from the statistical learning technique. The SVM approach has been applied extensively for the problems involving classification and/or approximation of a function (Raghavendra and Deka, 2014, Abe, 2010).

The basic feature of an SVM approach is to map the input space into a high dimensional feature space where nonlinearity of input vectors becomes linearly separable and to establish a decision boundary to solve classification problems (Raghavendra and Deka, 2014, Müller et al., 1997). To solve regression problems, known as the Support Vector Regression (SVR) method, the SVM uses a kernel function which fits a tube *(ε)* (**Fig.** 2.10) in a higher-dimensional space (Liu and Pender, 2015).

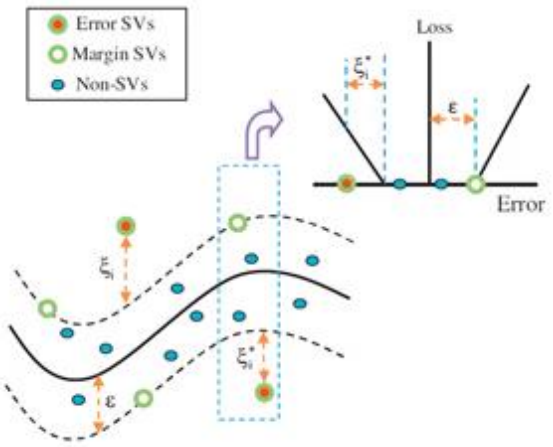


Figure 2.10: SVR with ε-tube adapted from Yu et al. (2006).

This approach transforms the regression problem into the format of a convex optimization problem:

$$minimize \quad Q(w,b,\xi,\xi^*) = {}^1\!/_2 \|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \qquad (2.8)$$

$$subject\ to \quad y_i - w^T\varphi(\boldsymbol{x}_i) - b \le \varepsilon + \xi_i$$

$$w^T\varphi(\boldsymbol{x}_i) + b - y_i = \varepsilon + \xi_i^*$$

$$\xi_i \geq 0, \xi_i^* \geq 0$$

Where, w is the weight vector, $\varphi(X)$ is the mapping function, b is the bias term, C is a positive constant that determines the trade-off between the margins and the quantity up to which the estimation error of the training data can be tolerated. To integrate data lying outside the ε-tube nonnegative slack variables ξ_i and ξ_i^* are introduced.

The kernel parameters, cost parameter C, and the radius of the tube ε are mutually dependent, and hence, changing the value of one parameter could affect the value of other mutually linked parameters. The effectiveness of an SVR-based approach mainly depends upon these model parameters, e.g. a larger value of C could over-fit the training data. The ε parameter controls the width of the tube influencing the number of support vectors, thus influencing the overall generalization ability of an SVR-based model. Furthermore, it is often challenging to estimate appropriate values of these parameters which mainly follows a heuristic trial and error process or exhaustive grid search approach. For a thorough discussion of the various technical aspects of the SVM and SVR method, please refer to Abe (2010).

2.3.4 Random Forests

The RF is one of the most popular and powerful supervised ML algorithms that, from a computational standpoint, is capable of both regression and classification tasks, is relatively fast to train, and depends on only a small number of tuning parameters (Cutler, 2014, Cutler et al., 2011). The RF, first introduced by Leo Breimen (Breiman, 2001), is a decision-tree-based ensemble method. This section will present a brief introduction to the RF method in the context of both classification and regression-based problems, with a brief discussion on the schematics of decision-tree-based approaches.

2.3.4.1 Decision-tree

Decision-tree, a flowchart-like structure, can be created through a binary recursive partitioning procedure, which generates binary splits of distinct individual variables in the predictor space (input space). The entire predictor space often referred to as the *root node*, is held within the topmost node of the decision-tree. Other internal nodes of the decision-tree, in general, splits into two descendant nodes. The decision-tree (originating from the root node) selects the next descendant node based on an attribute testing

algorithm that intent to optimise decision making. The end nodes of the decision-tree are called leaf or terminal nodes.

If the predictor variable is continuous then a split-point condition can be defined to partition a non-terminal node. The split-point condition routes values less than the split-point condition to one of the descendant nodes and the greater values to the other descendent node. For the categorical cases, the predictor variable is required to opt values from a finite set of categories, which can be further partitioned into distinct subsets of categories based upon some splitting criterion. For example, for a regression-based problem, if the observed response values at some node are- $y_1, y_2, y_3, \dots, y_n$, then a splitting criterion based on mean squared residual at that node can be defined by the following equation:

$$S = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 \qquad (2.9)$$

Where, $\bar{y}$ is an average of the response values at the node. In case of a classification tree, the Gini index, a typical splitting criterion can be expressed as:

$$S = \sum_{n \neq n'}^{N} \widehat{p_n}\widehat{p_{n'}} \qquad (2.10)$$

Where, *N* is the number of classes, denoted as 1, 2, 3, … , *N;* and $\widehat{p_n}$ is the proportion of class *n* observations in the node:

$$\widehat{p_n} = \frac{1}{m}\sum_{i=1}^{m} I(y_i = n) \qquad (2.11)$$

The splitting process continues recursively until a user-defined stopping criterion is reached. To demonstrate splitting procedure described above (e.g. in continuous cases), a simple example utilising arbitrarily generated data, a regional level response variable and two unitless predictor variables: X1 and X2, has been constructed. A regression-tree has been fitted to the predictor variables and has been illustrated in Figure 2.11.

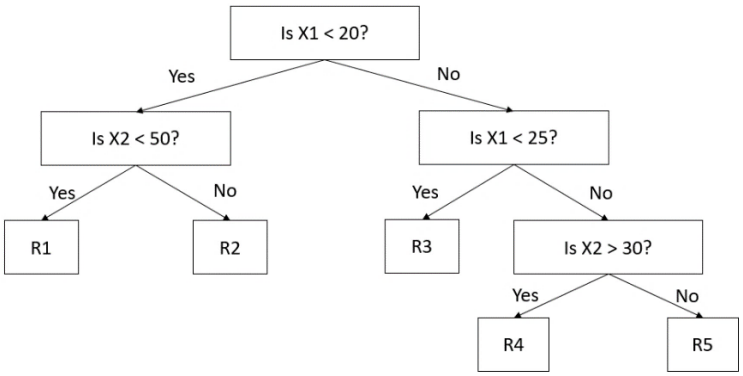


Figure 2.11: The fitted decision tree diagram and splitting of each terminal node result in a single predicted response.

Decision trees are applied to address a wide range of problems, partly because they are theoretically easy to build, can effectively handle missing values in the predictor variables, are suitable for a large dataset and their tree-like structure is easy to interpret. However, the decision-trees have limited abilities in providing highly accurate outcomes, as quoted in Section 10.7 of 'The Elements of Statistical Learning (2009)'- popularly known as the bible of statistical learning*: "Trees have one aspect that prevents them from being the ideal tool for predictive learning, namely inaccuracy."* (Hastie et al., 2009). They appear to provide reasonable prediction with the data used to construct them, but they have limited abilities to process and generate reliable predictions with new datasets. To address some these limitations, the RF-based model is proposed. RF-based model is designed to improve overall prediction accuracies and has a simplistic structural design, inspired by the decision-trees (as discussed below).

2.3.4.2 RF algorithm

RF-based model uses the so-called bootstrapping method to create random datasets from the training data[1]. Within the RF algorithm, the size of the bootstrapped dataset is same as of the original dataset. The decision-trees (base learners of RF) are fitted to the datasets (generated from the bootstrapping method) for a random subset of predictor variables at each step.

[1] Bootstrapping is a method of creating multiple datasets where the samples are randomly selected from the original training set.

Using a bootstrapped sample and randomly selecting a subset of the predictor variables at each step results in a wide variety of different forms of decision-trees. The variety is what makes RF more effective than individual decision trees. When a new sample is passed to the trees created from bootstrapped data, the responses from these individual trees are combined by unweighted voting (classification) or unweighted averaging (regression) to make the final optimised decision.

The process of bootstrapping training data for generating multiple decision trees and aggregating the responses to make the final decision is called bagging. Bootstrapped dataset allows duplicate entries, and the samples that are not selected in the bootstrapped dataset are called out-of-bag dataset. Since the out-of-bag datasets are not used to construct the decision-trees, they are used to measure the accuracy of the RF-based model. Initially, it was suggested by Breiman (2001) that trees should be large, while more recently Segal and Xiao (2011) suggested that the maximum number of terminal nodes should be controlled. Please refer to Cutler (2014) for a detailed description of RF algorithms.

2.4 Catchment hydrology

The catchment is an area in the landscape which drains water to an outlet through the lateral flow over the surface and underground (Mulligan and Wainwright, 2013). In other words, a catchment or a watershed is a single fluvial system that is linked internally by a drainage network (network of channels). The catchment boundaries define the separation of surface flow from one hydrologic system to another (Fryirs and Brierley, 2012).

The process of natural water movement, in a hydrologic system is *distributed* in terms of time and space derivatives (Mujumdar and Kumar, 2012). The science of understanding the movement of water in a catchment involves delineating the water balance of individual Hydrological Response Units (HRUs) or hydrologically similar surfaces (Mulligan and Wainwright, 2013, Kirkby et al., 2002). The water balance accounts for the difference between the total precipitation falling on the surface of a catchment and the total losses occurring due to the interaction of soil, topography and atmosphere.

The following sections present a brief overview of key hydrologic/hydraulic processes/procedures, e.g. precipitation partitioning (essential for water balance/budget estimation), estimating water discharge (hydrologic modelling) and visualising flood inundation (hydraulic modelling).

2.4.1 Water cycle and water balance

The water cycle, in general, is a series of flows of water between various water stores. Precipitation is a process of water falling onto the surface of the earth in the forms of rain, snow or hail. Precipitation that directly falls into the oceans becomes part of the ocean surface and can be churned by wave and wind action into ocean currents. Rain and snow that falls directly on rivers and streams become the part of streamflow. However, the precipitation that falls onto the land takes a different path to the river as does the snow and the ice that collects on the mountain tops. Precipitation falling on land surface (some of it is intercepted by vegetation) can either runoff, if the ground is hardscaped (asphalt or concrete) or if the soil is too wet (saturated to absorb more water), otherwise infiltrates into the soil surface and percolates into the ground. The groundwater laterally discharges towards a lake, river or sea. Water moves into the atmosphere from the stores through the evaporation and evapotranspiration. Evaporated water molecules mix with other atmospheric particles, cools and condenses into visible masses of water vapour (clouds) and forms the water droplets that fall to the earth surface as precipitation.

The water balance equation that combines the factors influencing run-off variation is used to estimate the amount of water in each component of the water cycle. The equation read as:

$$P - Q - ET = \Delta S \tag{2.12}$$

Where P stands for precipitation (input), Q is streamflow (output), ET is evapotranspiration and ΔS is the change in storage (in soil or groundwater). The hydrologic models simulating the process of water movement utilises the water balance equation to calculate the flux of water moving across the storages and atmosphere. Figure 2.12 illustrates various hydrological processes that occur during a storm or heavy precipitation event. The key focus is on demonstrating the partitioning of precipitation that flows through various routes and resulting in the streamflow.

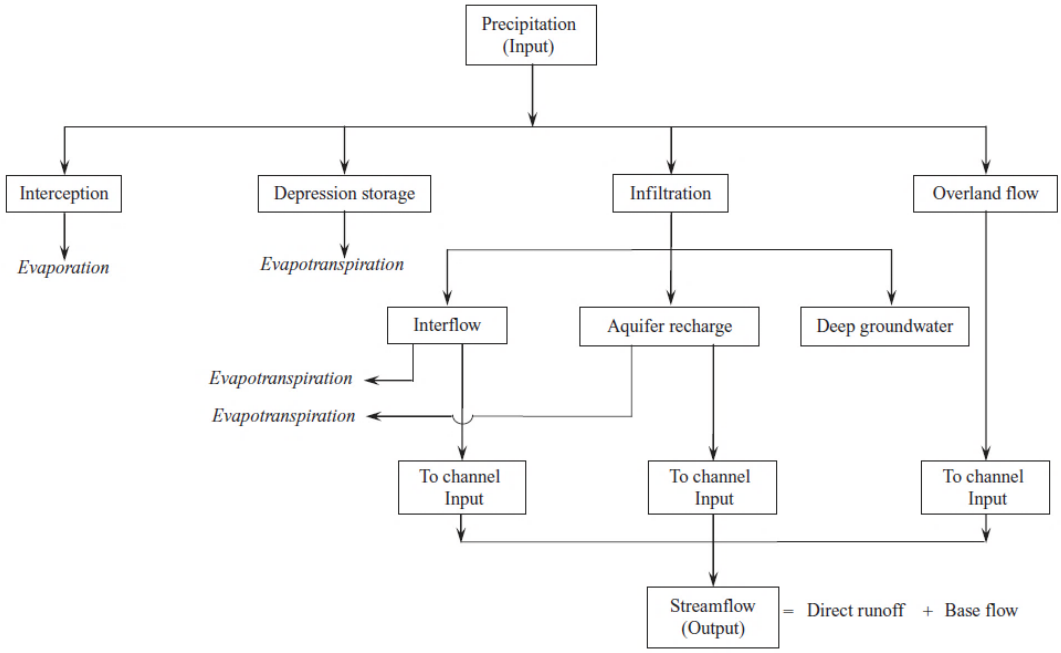


Figure 2.12: Hydrological processes for routing precipitation in the streamflow (Mujumdar and Kumar, 2012).

Please referred elsewhere, e.g. Mujumdar and Kumar (2012), for a detailed explanation on various components of rainfall partitioning.

2.4.2 Hydrologic modelling of flood

The hydrologic models used for simulating the flooding events transform rainfall to streamflow and depends on various hydrologic processes, as described above. Simulation of streamflow during a flood event requires a thorough assessment of rainfall and runoff at the selected locations of the watershed either through a distributed way or a lumped way at the outlet of the watershed. A flood hydrograph is used to present the modelled streamflow at the outlet of a watershed.

A large variety of hydrologic models are available to understand and predict the hydrologic behaviour of the catchments, specifically for catchment-scale rainfall-runoff modelling. These hydrologic models are used for a range of applications, for example, to estimate flood hydrographs at specified locations, floodwater routing and inundation assessment in conjunction with GIS (Mujumdar and Kumar, 2012). The catchment-scale rainfall-runoff models range from a simple input (rainfall) output (runoff) type model (black-box) to complex physically based fully distributed models.

Models also vary depending on output types, for example, if an experiment/simulation is repeated multiple times under a given set of initial conditions (input parameters/variables, boundary conditions remain unchanged at each iteration), a deterministic model will produce identical outputs, whereas a stochastic model will produces different outputs (Mulligan and Wainwright, 2013). Thus, stochastic models can be used to model inherent uncertainties of the system (that could be due to natural processes, model selection, parameter choices and input dataset). Please refer to Beven (2012), Singh and Woolhiser (2002) for an exhaustive review of hydrologic models. Some of the widely applied deterministic hydrologic model types are discussed in the following.

In the catchment scale hydrologic modelling, the first stage involves the development of a conceptual modelling framework of the watershed. This is done through the delineation of the watershed, selection/calibration of model parameters and model formulation (either as a lumped, or a black-box, or a distributed model). Once the conceptual modelling framework is formulated, the calibrated model can be applied to simulate the runoff for any given rainfall conditions in the selected area, i.e. to perform either event-based simulation or continuous simulation.

Deterministic hydrologic models, depending upon the parameter selection, can be classified into three main categories: lumped, semi-distributed and distributed.

Lumped models are applied when the parameters of the watershed do not vary spatially within the catchment, and the response is required to be estimated only at the outlet. The lumped model estimate runoff at the outlet without explicitly accounting for the response of individual sub-catchments. Key attributes of lumped models are:

- Parameters do not represent physical features of the hydrologic process
- Discharge predictions are provided at the outlet only
- Simple and minimal data requirements
- Example: Soil Conservation Service (SCS) based models (e.g. TR-20)

Semi-distributed models are applied when the parameters of the watershed are partially allowed to vary in space (by dividing the catchment into several small sub-catchments). There are mainly two types of semi-distributed models: (1) kinematic wave theory-based models, which are a simplified version of surface flow equations of a physically-based model (implementation of the St. Venant equations); and (2) probability distributed models, where the spatial resolution is accounted by using probability distributions of

input parameters across the catchment. The semi-distributed models are more physically based than the lumped models. However, they are computationally less demanding and require considerably less amount of input data than the fully distributed models. Some key examples of semi-distributed models include: the Hydrological Simulation (HBV) model (Bergström, 1992), the Hydrologic Engineering Centre–Hydrologic Modelling System (HEC-HMS) (Feldman, 1981), and the Probability Distributed Model (PDM) (Moore, 2007) etc.

Distributed models are applied when the parameters of the watershed are fully allowed to vary in space at a user-specified resolution. They incorporate the spatial distribution of the model parameters together with the computational algorithms to facilitate simulation of catchment hydrology at any user-specified spatial point and timestamp. Physically based deterministic models can provide a better understanding of various hydrologic phenomena occurring in a watershed and can also simulate the impacts of any potential morphological changes that may affect these phenomena. However, the availability of a large volume of datasets required for model calibration/validation could be practically challenging, and often yields poor results. Whilst spatial information regarding catchment topography is readily available from remote sensing, other catchment properties such as soil and surface properties cannot be collected remotely. Intensive fieldwork is required to collate detailed measurement of various catchment properties, and even then, it is often practically challenging to access and provide complete spatial coverage for a comprehensive modelling. As a result, the uncertainty involved in model parameterization is considerably high, which impact the overall reliability of model outcomes. Please refer to Beven (1996), Grayson et al. (1992a), Grayson et al. (1992b) for detailed discussion on the equifinality problem of distributed models. Some widely applied example of distribution models includes, the Soil Water Assessment Tool (SWAT) (Arnold et al., 2012), the generalized river modelling package (MIKE-SHE) (Refsgaard and Storm, 1995) etc.

Different hydrologic modelling schemes for runoff estimation and their deriving equations can be found in standard textbooks, e.g. Chow et al. (1988).

2.4.3 Mechanics behind inundation modelling

This section will discuss the natural process of a flood wave propagation along a channel. Some brief overview of the theoretical framework used for estimating the flood wave

propagation is also presented. Due to extensive application across a range of flood management projects, topics discussed in this section are important and highly relevant to the thesis.

A flood wave travelling along a river continuously changes its velocity and depth with time and distance. A flood hydrograph is used to represent the changes of wave front properties (velocity and depth) in time (**Fig.** 2.13). The process of estimating these changes in time and distance is known as 'flood routing'. For practical applications, given a flood wave (hydrograph) is generated at an upstream, the general interest is to determine the wave properties at some location along the channel for a specified time.

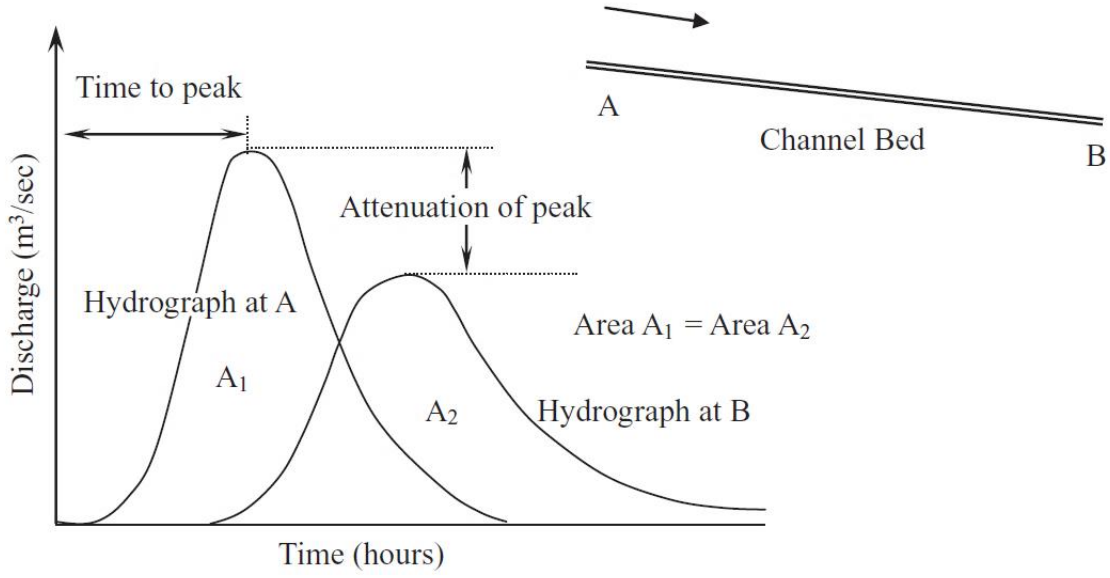


Figure 2.13: Flow hydrograph at two geographical locations (upstream and downstream) (Mujumdar and Kumar, 2012).

The principles of continuity (mass conservation) and momentum (Newton's second law of motion) form the basis of the flood wave propagation equations (the hydraulic method of flood routing). These equations are known as shallow water equations or Saint-Venant wave equations. The governing equations for a one-dimensional (1D) open channel flow can be written as:

$$\frac{\partial Q}{\partial x} + \frac{\partial A}{\partial t} = 0 \qquad (2.13)$$

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}\left(\frac{Q^2}{A}\right) + gA\frac{\partial h}{\partial x} - gAS_0 + gAS_f = 0 \qquad (2.14)$$

(a) (b) (c) (d) (e)

Where, **Eq.** 2.13 represents the mass conservation equation and **Eq.** 2.14 represents the momentum conservation equation. The terms (a-e) in the momentum conservation equation are explained below:

(a) Local acceleration: defines the rate of change in momentum in relation to time in the control volume.

(b) Convective acceleration: describes the rate of change in momentum due to the change in velocity from upstream to downstream.

(c) Pressure term: the pressure force acting in the upstream is not the same as it is in the downstream. The pressure term defines the force proportional to the change in water depth from upstream to the downstream along the channel.

(d) Bed slope: is the gravity driving the flow downhill.

(e) Friction slope: is the friction force of the slope acting towards slowing down the flow.

However, in 2-D, flow is computed for both *x* and *y* directions. Thus, the mass conservation equation (**Eq.** 2.15) also includes the rate of change of water surface elevation in the *y*-direction and **Eq.** 2.16 along with **Eq.** 2.17 represent two momentum conservation equations in *x* and *y* directions (direction of motion):

$$\frac{\partial h}{\partial t} + \frac{\partial hu}{\partial x} + \frac{\partial hv}{\partial y} = 0 \qquad (2.15)$$

$$\frac{\partial hu}{\partial t} + \frac{\partial (hu^2)}{\partial x} + \frac{\partial (huv)}{\partial y} + gh\frac{\partial h}{\partial x} - ghS_{0x} + ghS_{fx} = 0 \qquad (2.16)$$

$$\frac{\partial hv}{\partial t} + \underbrace{\frac{\partial (huv)}{\partial x} + \frac{\partial (hv^2)}{\partial y}}_{\text{(b)}} + gh\frac{\partial h}{\partial y} - ghS_{0y} + ghS_{fy} = 0 \qquad (2.17)$$

(a) (b) (c) (d) (e)

In the above system of equations, Q is the discharge, A is the area, x and y denote the distances along x and y direction, u and v are the two components of the horizontal velocity, h is the water depth, S_{0x} and S_{0y} are the bed slopes, S_{fy} and S_{fx} represent the friction slopes. These shallow water equations define the motion of water, in terms of a depth-averaged 2D velocity and water depth, in response to the forces of gravity and friction (JACOBS, 2017). The terms (a-e) used in **Eq.** 2.17 are same as described earlier for the case of the 1-D equation.

For most practical applications, numerical methods (such as finite difference) are applied to solve the above 1D/2D equations. The procedure of solving the Saint-Venant equations numerically using a finite-difference based method (in the form of a simple wave equation) is presented below:

Consider a simple wave equation (**Eq.** 2.18) that describes the movement of a constant uniform wave.

$$\frac{\partial u}{\partial t} + U\frac{\partial u}{\partial x} = 0 \tag{2.18}$$

Where, *u* is the velocity, *U* is the wave celerity, and here *U* is assumed constant. In the computational plane (time versus distance) of the finite differencing method, space and time are split into the elementary steps, Δx and Δt (**Fig.** 2.14):

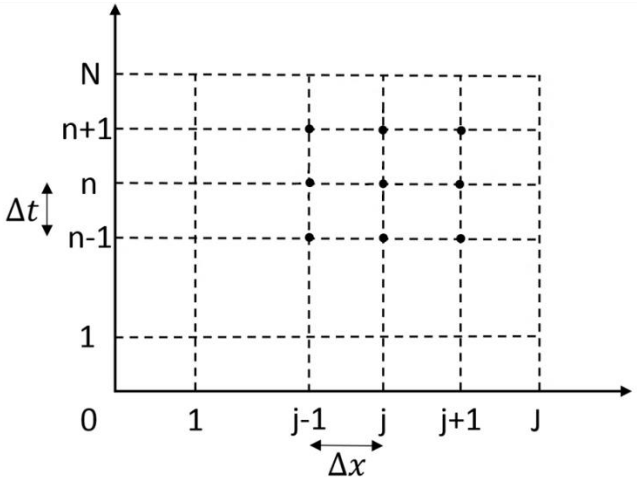


Figure 2.14: Space-time computational plane.

Every "node" on the plane is given a spatial index *j* and a time index *n*, such that:

$$x_j = j\Delta x \tag{2.19}$$

$$t^n = n\Delta t \tag{2.20}$$

Hence the wave at point (*j*,*n*) can be expressed as:

$$u_j^n = u\left(x_j, t^n\right) \tag{2.21}$$

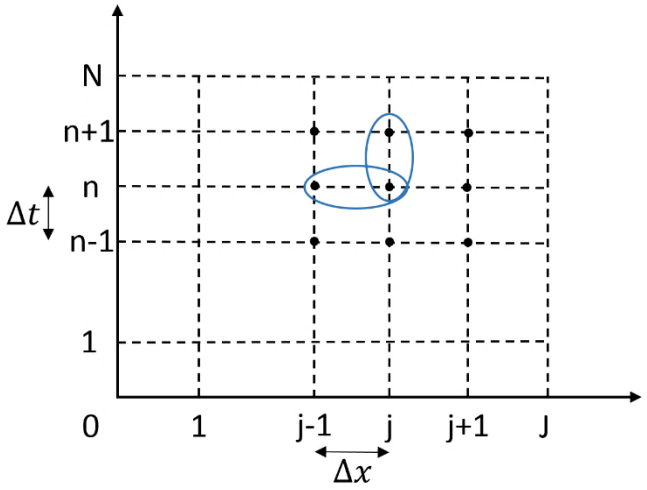


Figure 2.15: Explicit finite difference formulation at point (*j*,*n*).

In the explicit finite difference scheme formulation (**Fig.** 2.15), the values of *du/dx* remains constant during a change in the time step. Thus differentiating **Eq.** 2.21 at point (*j*,*n*) results in:

$$\left.\frac{\partial u}{\partial x}\right|_j^n \approx \frac{u(x_j,t^n)-u(x_{j-1},t^n)}{\Delta x} = \frac{u_j^n-u_{j-1}^n}{\Delta x} \qquad (2.22)$$

$$\left.\frac{\partial u}{\partial t}\right|_j^n \approx \frac{u(x_j,t^{n+1})-u(x_j,t^n)}{\Delta t} = \frac{u_j^{n+1}-u_j^n}{\Delta t} \qquad (2.23)$$

Therefore, at point (*j*,*n*) the wave equation can be approximated as:

$$\frac{u_j^{n+1}-u_j^n}{\Delta t} + U\frac{u_j^n-u_{j-1}^n}{\Delta x} = 0 \qquad (2.24)$$

Eq. 2.24, also called the forward in time and backward in space (upwind) scheme, can be rearranged as follows:

$$u_j^{n+1} = u_j^n - C\left(u_j^n - u_{j-1}^n\right) \qquad (2.25)$$

Where, $C = U\,\Delta t/\Delta x$ is the "Courant" number. In explicit schemes, the finite difference approximations used for space derivatives involve node values at time level only. The node values at time level *n+1* can be expressed as a function of node values at time level *n* and can be estimated recursively in successive levels. On the other hand, in implicit formulation, space derivative (*du/dx*) is calculated by averaging space derivatives at time levels *n* and *n+1* (**Fig.** 2.16). The time and space derivatives at point (*j*,*n*) in an implicit finite difference scheme are approximated as:

$$\left.\frac{\partial u}{\partial t}\right|_j^n \approx \frac{u_j^{n+1}-u_j^n}{\Delta t} \tag{2.26}$$

$$\left.\frac{\partial u}{\partial x}\right|_j \approx \frac{1}{2}\left(\frac{u_j^n-u_{j-1}^n}{\Delta x}+\frac{u_j^{n+1}-u_{j-1}^{n+1}}{\Delta x}\right) \tag{2.27}$$

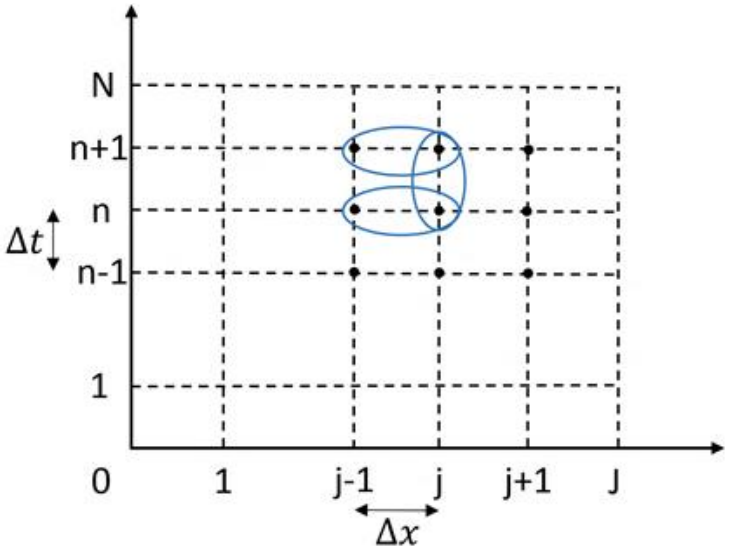


Figure 2.16: Implicit finite difference formulation at point (*j*,*n*).

Substituting **Eq.** 2.26 and 2.27 in the wave equation yield:

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+U\frac{1}{2}\left(\frac{u_j^n-u_{j-1}^n}{\Delta x}+\frac{u_j^{n+1}-u_{j-1}^{n+1}}{\Delta x}\right)=0 \tag{2.28}$$

With $C = U\,\Delta t/\Delta x$, **Eq.** 2.28 can be rearranged as:

$$\left(1+\frac{C}{2}\right)u_j^{n+1}-\frac{C}{2}u_{j-1}^{n+1}=\left(1-\frac{C}{2}\right)u_j^n+\frac{C}{2}u_j^{n-1} \tag{2.29}$$

Eq. 2.29 is called Central in Time and Upwind in Space (CTUS). In implicit schemes, the finite difference approximations used for space derivatives involve node values at both time levels *n* and *n+1*. Therefore, node values at time level *n+1* cannot be computed directly from node values at time *n*. Distinct equations of the form of **Eq.** 2.29 can be formulated for each node in the model, and the system of these equations can be solved using matrix calculus. The implicit schemes, as discussed above, are considered as mathematically more accurate and stable than the explicit schemes. However, the calculations of this system of equations at each time step can be slow and computationally demanding.

The explicit and implicit formulation of the finite difference approach have been discussed in this section using a simple wave equation. The process of transforming the shallow water equations into a finite difference approximation problem is conceptually similar to the representation provided in **Eq.** 2.18, although a little complex in terms of implications. In this case, the approximation procedure is applied simultaneously at each time step to the continuity equation and the momentum conservation equation(s), and an appropriate equation for the wave celerity (in both *x* and *y* directions) is also used to replace the constant *U*.

To deal with the complexity involved in the numerical solution of the 2D system, a frequently applied Alternating Direction Implicit (ADI) approach is discussed. The ADI approach involves the decoupling of the solutions obtained in the *x* and *y* directions at each time step. That is, at each time step, the equations are solved for $\Delta t/2$ in the *x*-direction and then for $\Delta t/2$ in the *y*-direction. This approach considerably reduces the size of the systems of equations that require to be solved simultaneously. Detailed information on 1D/2D shallow water equations and various numerical methods to solve these equations is available elsewhere, e.g. Chow et al. (1988).

Diffusive wave models are proposed to simplify dynamic wave (momentum) equations. In the diffusive wave models, acceleration terms are discarded, and only pressure, slope and friction terms are considered. This approximation is valid in most real flood inundation scenarios due to natural causes (e.g. fluvial flooding). However, the approximation approach is not suitable if modelling is done at high resolution or in the case of the dam break and flood defence failure scenarios. LISFLOOD-FP (Bates and De Roo, 2000) is an example of a diffusive wave model. In some cases, the models are further simplified by removing the pressure term. Such models are called kinematic wave models.

2.5 Conclusion

This chapter has presented a brief theoretical overview of the key ML and conventional hydrological modelling schemes. The conceptual frameworks and structural design of four widely applied ML algorithms have been described. These algorithms are selected following an intensive literature review of previous studies conducted in the areas of the application of ML approaches for streamflow forecasting (as detailed in Chapter 3).

To integrate the inter-disciplinary aspect of the thesis, this chapter also presented a brief overview of key hydrological processes and hydrological models used to understand different types/forms of physical/natural processes. The mathematical equations used in inundation modelling and finite difference-based methods for numerically solving these complex systems of equations have been illustrated using a simple wave equation model. The theoretical knowledge acquired in these sections is of vital importance for the sake of understanding the physicality involved in the hydrodynamic models that the current research is looking to simplify through ML.

Chapter 3 **Application of Machine Learning in Hydrological Modelling-A Review**

Modelling in science remains, partially at least, an art. Some principles do exist, however, to guide the modeller. A first, though at first sight, not a very helpful principle, is that all models are wrong; some, though, are more useful than others and we should seek those. At the same time we must recognize that eternal truth is not within our grasp.

McCullagh and Nelder (1989)

3.1 Introduction

Possibly due to considerably high spatial and temporal variability in rainfall distribution processes, partly attributed to highly nonlinear and inordinately complex rainfall-runoff process (Guo et al., 2011), flood forecasting remains one of the most challenging albeit important tasks of operational hydrology (Chang et al., 2007). For decades, a considerable amount of research that incorporating physical, conceptual, and stochastic models has been carried out to understand the complexity of rainfall-runoff processes. Although physical and conceptual models are important in understanding hydrological processes, such models are difficult to implement and calibrate as they require detailed information on several parameters (often not readily available), expertise in setting up complicated software tools, time and computational resources (ASCE, 2000a, Chang et al., 2007, Raghavendra and Deka, 2014). Due to these various reasons, these modelling procedures often failed to provide any useful information to decision makers during a real-time event.

A significant amount of studies has been done to investigate and compare the suitability of various data-driven modelling techniques for rainfall-runoff, flood forecasting and stream flow modelling studies (Solomatine and Ostfeld, 2008). The American Society of Civil Engineers (ASCE) task force committee has produced a two-part series review on ANN in hydrology. The first part of the series is an introductory paper and discusses the basics of ANN. It also provides the strengths and weaknesses of these models compared

to other modelling approaches, such as the physically-based models (ASCE, 2000a). The second part of the series provides a comprehensive review of past studies, which includes different model architectures, activation functions that had been used and concluding remarks on future developments (ASCE, 2000b). In most recent years, Mosavi et al. (2018) have reviewed 180 articles that have used multiple ML techniques for both short and long term flood prediction. They have particularly focused on the robustness, accuracy, and effectiveness of the state-of-the-art ML-based models. Besides, the need for model hybridization, data decomposition, model optimization, and algorithm ensemble, to improve the effectiveness of ML-based models has also been reported. Yaseen et al. (2015) reviewed AI-based models for streamflow forecasting from 2000 to 2015, with a focus on current challenges and opportunities for prospective research and future applications. Chandwani et al. (2015) have reviewed different soft computing techniques, namely ANN, Genetic Algorithm (GA)- which can be considered as grey-box method and fuzzy logic for rainfall-runoff modelling. Furthermore, Raghavendra and Deka (2014) have comprehensively reviewed the SVM applications in the field of hydrology. They have also emphasised the potential of hybrid SVM based models, for instance, SVMs integrated with Singular Spectrum Analysis (SSA), Wavelet Transform (WT), Discrete Wavelet Transform (DWT), and Particle Swarm Optimization (PSO) techniques. The application of the ensemble-based method, such as the RF (can be defined as grey-box method), for water scientists and practitioners, has been extensively reviewed by Tyralis et al. (2019). However, most of the ML research in hydrology involves river flow forecasting. There are only a handful of studies that applied ML to model flood inundation.

For flood inundation modelling, researchers have studied Cellular Automata (CA), a discrete and abstract computational system, to reduce computational costs. For example, Nkwunonwo et al. (2019), Liu et al. (2015), Ghimire et al. (2011) have developed CA-based fast urban flood simulators. The CA is composed of regular finite grids (defined by the DEM) and at each time-step, the grids instantiate one of a finite set of states (flooded or dry). The states of the grids and its local neighbourhood are updated following fixed rules.

As mentioned earlier, the application of ML for streamflow forecasting has been intensively reviewed in the past, and their advantages and limitations are well

documented, the potential of the ML approach for inundation modelling, however, is still understudied.

Application of ML techniques for rainfall-runoff forecasting has been investigated for more than two decades. It is beyond the scope of this chapter to summarize all of them, but within the context of the present research, findings from some key and representative studies are presented in **Sect.** 3.2. In contrast, research on the applications of ML for flood inundation modelling/simulation is very limited, and these studies, to the author's knowledge, have never been reviewed. In **Sect.** 3.3, some of the key works done (as known to the author) regarding the ML approaches in flood inundation modelling/ simulation are reviewed.

3.2 Machine learning for hydrological modelling

In the past, researchers reviewed the applicability, challenges, and opportunities of multiple ML techniques for streamflow forecasting from different perspectives. For example, the efficiency of models used in short-term and long-term flow forecasting, hybridization, ensemble forecasting has been studied. In this section, some of the previous works which have been done in the field of rainfall-runoff forecasting will be reviewed from a stationarity aspect of the models. That is, the papers were classified into two groups, i.e. into a static and a dynamic modelling approach. In the following sections, the applications of both modelling approaches will be described. While reviewing these articles, special attention has been paid to the types of the model used, the model selection criteria, selection of input variables, aims and objectives of the research, parameter optimization techniques, and key findings.

3.2.1 Static modelling approach

This is the most common form of rainfall-runoff modelling approaches. Over the past two decades, static data-driven models have gained popularity in the field of hydrology due to their ability to mimic hydrological responses of a catchment. Among these non-linear regression-based modelling techniques, specifically, ANNs have been the most popular and successfully applied as a means of predicting the runoff response (Dibike and Solomatine, 2001, Minns and Hall, 1996). Dawson et al. (2006), Dawson and Wilby (2001), Dawson and Wilby (1998) applied a Back Propagation Neural Network (BPNN) for rainfall-runoff modelling at different flood-prone catchments in the UK. In all these case studies, the BPNNs were found to have produced comparative results when tested

against standard models (e.g. UK FFS model results). Tokar and Markus (2000) have applied BPNN to model the runoff processes in three basins with different climatic and geomorphologic characteristics. Birikundavyi et al. (2002) applied ANN to forecast streamflow up to 5 days ahead.

Khan and Coulibaly (2006) applied a Bayesian learning approach to train their neural network for daily river flow and reservoir inflow in a cold river basin in Canada. Their results showed that the Bayesian Neural Network (BNN) model outperformed both the conceptual (HBV rainfall-runoff model) and standard ANN model. In addition, the BNN has a significant advantage over the standard ANN, namely the uncertainty quantification in the form of confidence intervals which are important in operational water resources applications. Makkeasorn et al. (2008), used Genetic Programming (GP) and ANN models for forecasting river discharges in a semi-arid watershed in south Texas. The findings indicated that GP-derived streamflow forecasting models were generally favoured for forecasting over ANNs and the most forward-looking GP models could forecast streamflow 30 days ahead of time with the r-square value of 0.84. Chang et al. (2007) investigated three types of ANNs, namely, Multiple-Input Multiple-Output (MIMO), Multiple-Input Single-Output (MISO) and serial-propagated structure, for multi-step ahead flood forecasting at two watersheds in Taiwan. The results showed that both MISO and serial-propagated neural networks can provide accurate short-term forecasting for up to one- or two-steps-ahead. However, for long term (more than two steps) forecasts, only the serial-propagated neural network was shown to provide satisfactory results. Kourgialas et al. (2015) applied ANN for river flow forecasting for the next two decades under climate change scenarios of a small agricultural watershed, whereas Hassan et al. (2012) applied Statistical Down Scaling Model (SDSM)2 data as input to an ANN model for hydrological trend analysis in a sub-catchment.

Jain and Srinivasulu (2004) investigated different ANN models to predict discharge at Kentucky River basin. Their main aim was to develop a robust training method for ANNs and include some physicality in the model to make it a so-called grey-box method. The data-driven models are considered as black-box where a user transforms a set of input data to desired output using an objective function without having to pass any physical

[2] SDSM is an open source software package for producing high-resolution site-specific climate information from the coarse global climate model simulations using statistical downscaling methods.

information of the system to the model. Usually, total rainfall is used as one of the input variables for data-driven modelling approaches. However, in this grey-box method, effective rainfall was used instead of total rainfall to capture the underlying physical dynamics of the rainfall-runoff process. Effective rainfall is calculated using conceptual equations, i.e. modelling the infiltration process through conceptual infiltration equations that account for soil moisture conditions. Four NN models were developed: ANN trained with BP algorithm, ANN trained with real coded GA, grey-box ANN trained with BP algorithm and grey-box ANN trained with GA. The results showed that the ANN-GA can significantly improve the predictive performance for low flow, whereas, for high magnitude flows, no significant advantages of using ANN-GA or grey-box method have been observed. Also, training with GA can be computationally intensive than BP.

Riad et al. (2004) applied ANN at Aghbalou station, Morocco, to predict daily runoff and compared the results against Multiple Linear Regression (MLR) models. The results found were better than MLR and were in close agreement with observed data. In the model, the authors used rainfall and runoff data for the preceding seven days and the rainfall expected for the day as input variables. Jeong and Kim (2005) developed a Single Neural Network (SNN) and an Ensemble Neural Network (ENN) for predicting monthly inflow at the Daecheong multipurpose dam in Korea. Performance of these models was compared against a conceptual hydrological model (TANK model). The ENN outperformed both SNN and TANK. The ENN also considerably improved the probabilistic forecasting accuracy of the present ensemble streamflow prediction (ESP) system in the studied area that used the TANK model. To model flood events, an ANN coupled with a simple linear model was applied by Rajurkar et al. (2004) to seven distinct catchments around the world, and the outcomes were compared to the observed flows. For all seven catchments, the estimated Nash-Sutcliffe efficiency (NSE) (Nash and Sutcliffe, 1970) values were over 0.75. In an earlier study, Rajurkar et al. (2002) applied ANN coupled with an MISO model for modelling daily flows during monsoon flood events of the Narmada River in Madhya Pradesh (India). To increase the performance of the ANN model, Mallikarjuna et al. (2009) included residuals from simple time-series (STS) and linear autoregressive (ARX) models as an additional input variable for forecasting monthly runoff at Keesara, Damarcherla and Madhira gauging sites of Krishna basin. Campolo et al. (2003), proposed an ANN based flood forecasting model with a 1-6h lead time for river Arno basin. The model was built using real-time hydro-meteorological data and dam operation information. Performance of the ANN based

models was evaluated by de Vos and Rientjes (2005) for daily and hourly flow forecasting at Geer catchment, Belgium. They concluded that the ANN could be a potentially more suitable method for daily runoff estimation than an hourly one. Machado et al. (2011) developed 24 ANN models using several sets of monthly data from 1976-1986 for monthly rainfall-runoff modelling of the Jangada River basin, Paraná, Brazil. The three best performing ANN models were selected and compared against a conceptual model (IPHMEN model). The study showed that the ANN models produced better and more consistent results than the conceptual model.

Rasouli et al. (2012) developed three ML methods, i.e. BNN, SVR and Gaussian process, to forecast flows at lead times of 1–7 days for a catchment located in British Columbia, Canada. They used input data from a weather forecasting model, i.e. the National Oceanic and Atmospheric Administration (NOAA) Global Forecasting System (GFS), along with several climate indices and observed local hydro-meteorological data. The climate indices used in this study were: the sea surface temperature in the Niño 3.4 region, the Pacific-North American (PNA) teleconnection, the Arctic Oscillation (AO) and the North Atlantic Oscillation (NAO). The model outcomes were compared against the MLR outputs (used as the benchmark). The observation showed that the BNN was superior to the other models. Moreover, it was also found that for shorter lead times, among various combinations of predictors, local observations along with the GFS can produce optimum results, whereas, for longer lead times (5-7 days) optimum results can be derived using the local observations and climate indices.

Solomatine and Dulal (2003) compared the performance of Model Trees (MT) and ANN for 1, 3 and 6h ahead flow forecasting. They concluded that both models are compatible and well suited for generating 1h ahead forecasting. Further, both the models performed with an acceptable range of accuracy in generating 3h ahead forecasting, though the results produced by ANN appears to be slightly better for generating forecasts at higher lead times (3 and 6h). Another popular soft computing method, i.e. Adaptive Neuro-Fuzzy Inference Systems (ANFIS), has been used by Anusree and Varghese (2016), for predicting daily streamflow at the outlet of Karuvannur river basin. The ANFIS model produced better results when compared against the ANN and Multiple Non-Linear Regression (MNLR) models. Alvisi et al. (2006) have also applied fuzzy logic and ANN models to forecast water level for the Reno River at Casalecchio di Reno, Italy.

Possibly due to a strong theoretical and statistical framework based on Structural Risk Minimization (SRM) principle (Vapnik, 1997), the SVR method offers greater advantages over ANN. Liong and Sivapragasam (2002) have compiled an extensive list of advantages of using SVM over ANN and also argued that SVM based models are not so-called 'black-box' models. The popularity of SVM has grown substantially since its induction as a non-linear classifier by Boser et al. (1992), and SVM has been applied extensively in a range of hydrological problems.

Asefa et al. (2006) showed the potential of SVMs in predicting site-specific, real-time flows. Wu et al. (2008a) applied a Distributed SVR (D-SVR) with a two-step GA parameter optimization method to predict river stages. The D-SVR method disaggregates the original training set into few subsets and then generates a local SVR for each subset independently. The performance of a D-SVR when compared against the predictions generated from Linear Regression (LR), nearest neighbour method, and GA-based ANN (ANN-GA) methods. The D-SVR-based model was shown to predict the water level comparatively better than the other models. Granata et al. (2016) developed an SVM based rainfall-runoff model for urban drainage and compared it with a Storm Water Management Model (SWMM); both approaches showed comparable results, with SVM slightly underestimating flow, while SWMM was overestimating it. Han et al. (2007) applied SVR for flood forecasting over the Bird Creek catchment, UK; the model performed better when compared with some benchmark models, e.g. linear transfer function model, a trend and a naïve model. Misra et al. (2009) used SVR and ANN to simulate daily, weekly and monthly runoff and sediment yield from an Indian watershed, and they concluded that SVM provided significantly improved results as compared to ANN based model estimation. Yu et al. (2006) employed SVR for real-time multi-step ahead flood stage forecasting and demonstrated for a river in Taiwan that the SVR-based models could effectively predict the flood stage forecasts one-to-six-hours ahead. Liong and Sivapragasam (2002) compared SVR and ANN for flood stage forecasting and found that forecasting accuracy of SVR was better than in the ANN model.

One of the most comprehensive investigations of the predictive capabilities of different data-driven modelling techniques was done by (Elshorbagy et al., 2010b). They compared six modelling techniques, namely, ANN, GP, Evolutionary Polynomial Regression (EPR), SVR, M5 model trees, and K-Nearest Neighbours (KNN) on five different case studies of rainfall-runoff, evapotranspiration, and soil moisture content. Their findings

suggested that for highly nonlinear cases, ANNs, GP and K-NN could be the better option, whereas M5 performs reasonably well for linear and semi-linear data. Further, they suggested that the EPR based models could potentially perform close to the GP, given the data sets are linear. To simulate monthly streamflow in five Ethiopian rivers, Shortridge et al. (2016) developed several nonlinear data-driven models: Generalized Additive Models (GAM), Multivariate Adaptive Regression Splines (MARS), ANN, RF and M5. The results were compared to assess predictive accuracy, model interpretability and uncertainty under extreme climate conditions. The authors found that while RF and GAM had better predictive performance and interpretability, both models suffered from higher uncertainty under the influence of high temperatures.

One can observe that in the comparative modelling studies, one of the fundamental means to assess model performance is through evaluating it against other techniques using, in general, the correlation coefficient, Root Mean Squared Error (RMSE), and the NSE. However, these comparative studies are often considered impaired because of the less-than-comprehensive approach being adopted (Elshorbagy et al., 2010a). To highlight the shortcomings of the ML based comparative studies, Abrahart et al. (2008) used the example of neural network applications and suggested solutions or modelling guidelines and frameworks. In general, the usefulness of data-driven modelling techniques mainly depends on overall data representation achieved in the modelling framework. Therefore, exploratory data analysis is identified as an essential step to extract key features of historical data to ensure greater predictive accuracy (Bai et al., 2016).

For this reason, recently researchers have put much of their efforts on the feature extraction, data pre-processing and data transformation processes (Bai et al., 2016). The wavelet-ANN (WANN) is one of the reliable hybrid models used in time-series forecasting problems. Recently, wavelet decomposition has become a popular signal processing tool because of its ability to elucidate both spectral and temporal information within the signal (Okkan, 2012, Nourani et al., 2011). In this method, a time-series of input variables is divided into subseries using wavelet decomposition, which is then fed to the neural network. Nayak et al. (2013) applied WANN for river flow modelling and compared results with a standard ANN and conceptual North American Mesoscale (NAM) model. They concluded that WANN performed better than ANN and NAM in estimating hydrograph characteristics. Krishna et al. (2011) proposed a WANN to model the flow sequences of the Malaprabha river basin in India and compared

results with standard ANN and stochastic autoregressive (AR) model. They, too, concluded that the WANN produced better results than other models. Performances of standard ANN and ANFIS models were compared with their hybridized form (WANN and wavelet-ANFIS) by Badrzadeh et al. (2015). These models were used to forecast runoff at 1, 6, 12, 24, 36 and 48h lead-time. The hybrid models significantly outperformed their standard versions. Shiri and Kisi (2010) investigated a wavelet-ANFIS model for predicting daily, monthly and yearly streamflow. Shoaib et al. (2014) compared different WANNs to find the best wavelet function for rainfall-runoff modelling. They tested 92 WANN-based models with 23 different wavelet functions.

Although wavelet-based hybrid models seemed to dominate the hybrid rainfall-runoff modelling paradigm, many other variants of hybrid models have also been proposed. For example, Moradkhani et al. (2004) and Jhong et al. (2018) proposed hybrid models that integrated a Self-Organizing Map (SOM) and a neural network for rainfall-runoff process modelling. Wu and Chau (2011), conducted Singular Spectrum Analysis (SSA) to the input data and then the transformed data were used to train the ANN for daily streamflow forecasting. Hosseini and Mahjouri (2016) proposed an integrated SVR - geomorphologic ANN (GANN) model for daily rainfall-runoff modelling and compared the results with several other hybrid models. Their SVR-GANN model outperformed other comparative models in all three case studies.

In addition to standard and hybrid techniques, Deep Learning (DL) methods have also been used in hydrological forecasting to some extent. The significance of feature learning in predictive problems is well understood, and therefore, recently DL methods have gained interest. DL techniques, capable of extracting key features from the historic data, can improve the forecasting or predictive performance of the model. Hinton et al. (2006) proposed DBN as one of the DL techniques based on probabilistic generative models. Several hidden layers are combined to construct the deep structure of the network for realizing the latent changes through layer-wise learning. This technique has already been applied successfully in many different areas, e.g. image processing (Liu et al., 2016), character recognition (Sazal et al., 2014), draught forecasting (Chen et al., 2012), and reservoir inflow forecasting (Bai et al., 2016). However, very few studies have been reported utilising DBNs or similar DL methods for rainfall-runoff modelling. Bai et al. (2016) proposed a DBN for daily reservoir inflow forecasting. The results obtained from DBN-based model are superior to those obtained from the standard ANN, Least Square-

SVR (LSSVR), and WANN. Recently Li et al. (2018) applied Convolutional Neural Networks (CNN) for rainfall-runoff modelling.

3.2.2 Dynamic modelling approach

The articles that have been reviewed and discussed in the previous subsection have predominantly used a fixed set of input variables for forecasting streamflow/water level at $t+n$ time-step ahead. One of the major drawbacks of the static approaches is that the information on the sequential order of the inputs is often lost (Kratzert et al., 2018). To address this issue, lagged values of the input variables are also considered as inputs. However, the problem of selecting the right number of time lags is a critical parameter that needs to be determined carefully (Hsu et al., 1997). The Recurrent Neural Networks (RNN), a special kind of ANN, has been introduced as a new paradigm to facilitate the processing of the input in its sequential order (Elman, 1990, Jordan, 1997), and has made the search for the lags obsolete. The benefit of using RNNs is that they can be trained to learn the dynamics or time-varying patterns and thus, they are considered a very powerful processing tool for time-series analysis (Chang et al., 2014a). The RNNs have been applied extensively in financial time-series analysis (Wang et al., 2016) and automatic text generation (Sutskever et al., 2011). Although the RNNs as a dynamic modelling approach for hydrological time-series forecasting have shown promising results (Coulibaly and Baldwin, 2005), interestingly the number of research articles available in this area is substantially low compared to the work done in static modelling approaches. This could be probably because the RNN's are computationally intensive, and/or the superiority of the dynamic approach over the static approach is still debatable (Chen et al., 2013).

First records of conducting studies using RNNs for rainfall-runoff modelling can be found in Carriere et al. (1996) and Hsu et al. (1997). Carriere et al. (1996) tested the application of RNNs in a virtual laboratory environment, whereas Hsu et al. (1997) compared the performance of an RNN against the traditional ANN. The results showed that the RNN and ANN performed equally well. In another case study, Nagesh Kumar et al. (2004) applied RNN for monthly streamflow prediction and showed that the RNN outperformed the traditional ANN.

Recently, Chen et al. (2013) proposed a multi-step ahead reinforced real-time recurrent learning algorithm for RNNs (R-RTRL NN). The purpose of this method is to improve

upon the original RTRL algorithm, which is an online learning algorithm designed to make one step ahead forecasts (Williams and Zipser, 1989). The online learning system uses the latest observed information to adjust the model parameters that facilitates improved mapping between instances and true values. However, one of the key limitations of this method is that it requires continual true values. This means that a lack of true values makes the use of RTRL for multi-step ahead forecasts difficult. The R-RTRL algorithm, incorporating the latest antecedent forecasted and closest observed values, adopt a recursive approach for updating weights. The model was tested for 2, 4 and 6 steps ahead forecast, and the results were compared against the original RTRL, a dynamic Layer Recurrent Network (LRN), and a static ANN model. Authors found that R-RTRL not only outperformed comparative models but also significantly improved the precision of multi-step ahead forecasts.

Chang et al. (2014a) constructed one static ANN and two different types of RNNs, namely, the Elman neural network (Elman, 1990) and a nonlinear autoregressive network with exogenous inputs (NARX network) for a multi-step ahead Floodwater Storage Pond (FSP) water level forecast. The models were tested under two scenarios: (I) rainfall and FSP water level data as model inputs, and (II) only rainfall data as model input. The results showed that the NARX model produced the best results in forecasting FSP stages for 10–60 min (6-time steps) ahead (coefficients of efficiency within 0.7–0.9 (scenario I) and 0.5–0.7 (scenario II)). These findings suggest that the recurrent connections from the output layer (used in NARX network) are more effective than those of the hidden layer (used in Elman NN). The authors also emphasized the effectiveness of applying the Gamma test (Durrant, 2001) for selecting appropriate inputs (rainfall stations in this case).

The most advanced type of dynamic rainfall-runoff modelling system is probably the Long-Short Term Memory (LSTM) network-based rainfall-runoff model. The LSTM based dynamic model offers a powerful feature over widely applied RNNs, it is the ability to learn long-term dependencies between the input-output instances of the network (Le et al., 2019, Kratzert et al., 2018). This feature is particularly important for modelling storage effects in catchments with long memory or snow influence. Traditional RNNs have known difficulties in remembering sequences over 10-time steps (Bengio et al., 1994). This short sequence of learning period is often not suitable for large catchments where lag times between precipitation and discharge could have a significant impact and could vary more than 10 time-steps. However, LSTMs are designed to overcome the

limitation of traditional RNNs and have abilities to learn from long-term dependencies. Kratzert et al. (2018) tested an LSTM based rainfall-runoff model on 241 catchments in the US and compared it to the results of the known Sacramento Soil Moisture Accounting model (SAC-SMA) coupled with the snow-17 snow routine. In their models, they used daily precipitation, min/max temperature, solar radiation and vapour pressure time-series as inputs. The results showed that the LSTMs performed better in the snow influenced catchments and catchments with higher mean annual precipitation. However, the performance of the models was not comparatively good in the more arid regions (0 discharged values observed for longer periods). Although the performance of LSTM deteriorated in the arid catchments, the forecasted results still outperformed the benchmark model. Analysing the results obtained in snow affected catchments, the LSTM models appear to learn the long-term dependencies, i.e. the time lag between precipitation falling as snow during the winter and runoff generation in spring. They also showed the possibility of transferring pre-trained models into different catchments, which could be very useful for making predictions in ungauged or data-scarce basins.

3.2.3 Summary of ML in hydrological modelling

To summarize, in a rainfall-flow forecasting model, antecedent rainfall data is used as the key input driving the model. However, studies have shown that using antecedent rainfall data alone does not provide the best forecasts; this is due to mediums of water losses (e.g. catchment size and memory, infiltration flow, evaporation etc.). Therefore, except for the comparison purposes (e.g. Elshorbagy et al. (2010b), de Vos and Rientjes (2005)), all studies have incorporated other hydro-meteorological data to increase model accuracy. The list of other input variables includes - but is not limited to - antecedent flow/water level, solar radiation, temperature, potential evapotranspiration, soil moisture, and wind. Amongst these, antecedent flow/water level and antecedent rainfall have been the most popular input combination. Many studies have also shown that in addition to antecedent rainfall and flow, the inclusion of other variables did not improve model predictability (Abrahart et al., 2008, Toth and Brath, 2007, Anctil et al., 2004). The rainfall/precipitation data from ground-based rain gauges are commonly used for these forecasting models. With technological advances, remotely sensed rainfall data, available from satellites, has also been used as model input. For example, Chang and Tsai (2016) have used radar rainfall data in their SOM-ANFIS model, whereas, Nanda et al. (2016), Uysal et al. (2016) and Akhtar et al. (2009) have used satellite rainfall products in their ML-based models.

One of the most labour intensive but extremely important task for setting up an ML-based model is tuning the model parameters. The predictive ability of these models is heavily dependent on how the model parameters are optimized. However, to date, there is no rule of thumb for parameter selection method. Most of the past studies do not provide much detail on calibrating model parameters. The most common and traditional way of optimizing model parameters is the so-called trial and error (empirical) approach. In this method, the modeller iteratively searches for the best set of parameters (e.g. number of hidden units, activation functions for ANN, and gamma and cost parameters for SVR, number of estimators for RF). Although the trial and error approach is time-consuming and labour intensive, the parameter selection process gets easier with practice in a particular field of study (Dibike et al., 2001). To automate the parameter optimization process, several other methods have been suggested, for example, GA (Dhamge et al., 2012, Ghose et al., 2013), heuristic grid search method (Rubio et al., 2011), a two-step grid search and an evolutionary algorithm called Shuffled Complex Evolution (SCE-UA), as applied by (Lin et al., 2006).

In terms of model complexity, hybrid and dynamic models inevitably have a more complex structure than standard static models. However, one of the interesting questions for real-time applications is whether the added complexity of the hybrid models and the dynamic nature of the recurrent models add extra benefit for real-time multistep ahead flow forecasting. In other words, how much can the predictive capacity be improved by hybridizing a selected model and do dynamic models offer more than static models with regards to forecasting ability? It is clear from the reviewed articles that model hybridization can potentially improve the performance, however, it is not clear whether hybridizing a particular model has the capacity to exceed the standard form of other ML methods. For example, numerous works have shown that WANN outperforms standard ANN, but there are limited studies that show a WANN would outperform a standard SVR or RF in any given conditions. On the other hand, it is arguable whether a dynamic model can supersede static models. However, the dynamic models such as the LSTM models have shown to provide better flow forecasting performance in the snow dominated regions, which is probably because of their exceptional capacity of storing long-term sequential information. A summary table of key articles relevant to this research is presented in Table 3.1.

Article	Objective	Method used	Parameter estimation method	Inputs	Results
Yu et al. (2006)	Real time flood stage forecasting	SVR	Step grid search	Lagged river stage data from 2 previous locations and rainfall	Model can forecast flood stage one to six hour ahead.
Granata et al. (2016)	Rainfall-Runoff modelling in urban drainage	SVR	Trial error	Rainfall, flow observations	Compared with storm water management model (SWMM), both showed comparable results, with SVM slightly underestimating flow while SWMM overestimating it.
Sivapragasam et al. (2001)	Rainfall and runoff forecast	Hybrid SVR	Not mentioned	Decomposed rainfall and runoff time-series using singular spectrum analysis (SSA)	1-day lead rainfall and runoff forecasted results show SVM performed better compared to nonlinear prediction.
Han et al. (2007)	Flood forecasting	SVR	Trial and error	Rainfall, lagged flow	Performed better than transfer function model, trend and naïve models.
Misra et al. (2009)	Simulating daily, weekly, monthly runoff and sediment	SVR	Not mentioned	Rainfall, runoff, and sediment yield	Performed better than ANN based model estimation.
Lin et al. (2006)	Long term discharge prediction	SVR	Shuffled complex evaluation algorithm performed through exponential transformation	Lagged flow	Compared with ARMA and ANN model and SVM model predicted long term discharge better than ARMA and ANN

Li et al. (2010)	Stream flow prediction	SVR	Genetic algorithm	Past stream flow, surface temperature (SST), sea level pressure (SLP), and outgoing longwave radiation (OLR)	Outperformed multiple linear regression model and simple SVR-based model.
Guo et al. (2011)	Monthly streamflow forecasting	Hybrid SVR	Particle swarm optimization (PSO) algorithm	Wavelet de-noised past stream flow	Performed better than ANN
Dibike et al. (2001)	Stream flow prediction	SVR	Trial and error	Rainfall, evaporation, stream flow	SVR improved prediction accuracy compared to ANN and conceptual model SMAR.
Sedighi et al. (2016)	Rainfall-runoff modelling of snow affected watershed	SVR	v-fold cross validation	Precipitation, temperature, snow water equivalent	SVM with radial basis function performed better compared to other SVM and ANN models.
Elshorbagy et al. (2010b)	Predictive capabilities of different data driven models in hydrology	SVR, ANN, GP, K-nearest neighbor, M5 model tree, Evolutionary polynomial regression	Different methods for different algorithms, e.g., for SVR, parameters were selected from an exponential range of values	Precipitation, lagged discharge	All model performances were on par. RBF kernel was found to be more effective kernel function for SVR-based models.
Nayak et al. (2013)	River flow modelling	WANN	Trial and error	Lagged rainfall, discharge, and evapotranspiration	WANN performed better than standard ANN and conceptual North American Mesoscale (NAM) model.
Okkan (2012)	Monthly reservoir inflow prediction	WANN	Trial and error	Precipitation and temperature	WANN outperformed standard ANN and multiple linear regression.

Bai et al. (2016)	Reservoir inflow	Hybrid DBN	Empirical/ Trial and error	Antecedent inflow	DBN performed better than four other models
Kourgialas et al. (2015)	Simulate extreme flow events under current and future climatic conditions	ANN	Empirical/ Trial and error	Precipitation, river flow discharge and evapotranspiration.	Results were compared against the observed flow and simulations were made for 2012-2030 using two climate scenarios.
Badrzadeh et al. (2015)	Multi-step ahead river flow forecast	WANN, Wavelet-ANFIS	Trial and error	Rainfall, discharge	Wavelet based models outperformed standard ANN and ANFIS.
Krishna et al. (2011)	Daily river flow	WANN	Trial and error	Antecedent flow	Provided better results compared to standard ANN.
Kratzert et al. (2018)	Regional and catchment scale runoff	LSTM	Empirical method	Precipitation, aridity, snow fraction, seasonality	LSTM was able to achieve better model performance as the SAC-SMA + Snow-17.

Table 3.1: Application of ML in hydrological modelling.

In water resource management, it is indeed very important to learn whether a model is portable and scalable enough to be applied in a different location for which the model was not trained. This is key because reusable models can be extremely beneficial in terms of saving costs. Interestingly, a wide range of studies investigated multi-step ahead forecasting capacity (scalability) of the models, their portability aspect remains understudied. In this thesis, a test of model parameter portability is conducted for multi-step ahead streamflow forecasting using multiple ML techniques (Chapter 4).

3.3 Machine learning for inundation modelling

In the research background section (**Sect.** 1.1), some of the common drawbacks of the 2-D models were presented. The ideal solution to these various technical challenges would be to develop simple models that could significantly reduce the computational burden

while still being able to generate practically useful and statistically significant information.

To reduce the computational expense of high resolution 2D models, Néelz et al. (2007) developed the idea of improving the accuracy of coarse resolution model predictions by effectively extrapolating useful information by facilitating a specified number of high resolution model, i.e. fine grid model (FGM), runs. The idea underpinning this approach is the interpolation of the residuals between the coarse and fine model runs and then using the outcomes of the analysis to improve the accuracy of consequent predictions. This method has been shown to improve the predictive accuracy of the coarse TUFLOW model (50m grids) for Thamesmead site. However, this method is not suitable for practical applications, as different areas in the floodplain have different thresholds of flooding, and therefore it is not possible to characterize them by the same FGM run. Further, this approach has limited success when flood depth is low. Several attempts have been made in the past to reduce the overall computational time, such as the Rapid Flood Spreading Models (RFSMs) (Liu and Pender, 2013, Lhomme et al., 2008, Krupka et al., 2007). These types of modelling approaches were seen capable of reducing total computational costs, and the results obtained were shown to demonstrate a good degree of agreement to those obtained using TUFLOW modelling suites. Although these techniques have achieved considerable success in predicting maximum water depth (no water depth time-series is produced in this method), it has been widely recognised that they have limited success in accurately estimating velocity of flow.

To improve upon this, Liu et al. (2009) shifted the development of the fast flood modelling paradigm towards ML for the first time. In their work, they have shown that outputs from a coarse grid model (CGM), which are computationally less expensive than the FGMs, can be used with a non-linear regression model (SVR) to predict the outputs from an FGM. And by doing so, it is possible to emulate the outputs of an FGM without having to run FGM simulations. Their idea was to generate water depth and velocity values at some selected locations, using both the CGM and FGM runs for multiple inflow hydrographs. The authors used a TUFLOW hydrodynamic model to generate these outputs. Then the outputs from CGM and flow values were used as input variables, and outputs from FGM were used as the target variable to train the SVR-based model. Besides, the authors used the Particle Swarm Optimisation (PSO) (Kennedy and Eberhart, 1995) technique to search for the SVR parameters. The predicted water depth and velocity

values for a new inflow hydrograph were mostly comparable to the FGM outputs. However, the agreement was not as good as expected at some points. The authors acknowledged that a small number of training samples could be the main reason for this disagreement.

To further reduce the dependency upon CGM runs, Liu and Pender (2015) proposed another SVR-based modelling approach, which ignores the outputs from CGMs in the training process. They only used flow observation time and flow rates as input variables and FGM outputs (water depth and velocity) as the target variable. In this method, water depth/velocity time-series of selected locations are normalised to its arrival time to overcome the complexity associated with discontinuity of the flood wave arrival time. At a given location, for a hydrograph, the flood wave arrival time can be calculated using a linear regression model as suggested by Néelz et al. (2007). Calculated arrival times for different locations would be added to the predicted water depth/velocity values to de-normalise the final product. The results showed that the proposed model could sufficiently predict the water depth and velocity values when compared against the FGM generated outcomes. The ISIS-2D inundation model was used to generate FGM training and validation samples.

With a different concept, Chang et al. (2010) proposed a hybrid ML based inundation model that uses a clustering (K-means) step to categorize the different flooding characteristics (e.g. water depth, surface elevation) in the study area. Then the centre of the clusters is identified as control points. Once the control points are detected, a separate BPNN is trained for each control point using rainfall data from the nearby stations and water depth values at these points (extracted from 2D hydrodynamic model simulations). LR models are fitted to the grids that are highly correlated to the control points in each cluster, and a multi-grid BPNN is used for the grids that do not correlate. The multi-grid BPNN models use rainfall, water depth values of the corresponding control point at time t and $t+1$, manning's roughness value, the elevation of the grid, distance from control point to the grid, and water depth at t and $t-1$. As it is not possible to have observed depth values during an event, the models use time-lag water depths values as an input. Therefore, the authors have used the output from the BPNN models as an input for the next time step. In their study, they constructed 5 single-grid BPNNs for 5 control points, 5 multi-grid BPNNs for 1474 loosely correlated grids (nonlinear), and 1022 linear regression models for highly correlated grids, to forecast 1h ahead inundation depths.

This hybrid approach was tested in the Dacun Township in central Taiwan and results showed that the model has ability to forecast 1h ahead flood inundation maps at 80m x 80m spatial resolution, continuously and adequately. Pan et al. (2011) used only rainfall data to forecast flood depth at nineteen sites in Yunlin County, Taiwan. They used rainfall data from nearby stations with different order as inputs and Principal Component Analysis (PCA) for dimensionality reduction; then BPNN models were trained (the target was the water depth values simulated from a 2D model) to forecast $t+1$h ahead water depth.

In many previous studies, it has been discussed and suggested that it is not an efficient approach to recursively adopt forecasted values for many time-steps ahead in the future, which could result in accumulation of model error and performance degradation at each time step (Chen et al., 2013, Parlos et al., 2000). To reduce the error accumulation, Shen and Chang (2013) investigated the possibility of applying multiple RNN structures on 13 sites in Taiwan. They concluded that the recurrent configuration of the NARX (R-NARX) network can significantly inhibit error accumulation when implemented for generating multi-step ahead flood inundation forecasts for a longer period in the future.

Chang et al. (2014b) presented a new ANN based hybrid model for real-time multistep-ahead flood inundation mapping. In their approach, they first divided the study area into several sub-areas according to features such as villages, towns and cities. In next stage, the spatial distribution of flood depths information (collated from 989 datasets) was clustered into 25 inundation topological maps using a SOM, a special type of ANN designed for the clustering operation. These topological maps represented the spatial distribution of flooding for 25 types of hydrological conditions. Then the R-NARX network was constructed (one for each sub-area) using rainfall and total inundated volume. The total inundated volume is the value that quantifies the inundation condition of the sub-area and can be calculated by aggregating the depths of all the grids in the sub-area. The output of the R-NARX model is the total inundated volume that quantifies a specific hydrological condition. The forecasted total inundation depth was then compared to the total inundation volume of the 25 topological maps. The closely matched map was selected, and the total inundation volume of the map was adjusted using the forecasted total inundation volume. Further, an updated 40m x 40m map of a sub-area was produced from the adjusted total inundation volume through a proportional interpolation method. The results were compared against the hybrid model proposed by Chang et al. (2010) for

a design storm event in the Yilan County, Taiwan, and provided better forecasting performance in terms of smaller mean absolute errors and maximum absolute errors.

Using the same idea, Chang et al. (2018a) developed a SOM-RNARX based regional flood inundation model for up to 12h ahead forecasting in the Kemaman River Basin, Malaysia. This ML based modelling approach was further improvised in Chang et al. (2018b), where they have proposed an Intelligent Hydroinformatics Integration Platform (IHIP) to provide a user-friendly web interface for improved online forecasting capability and flood risk management.

In recent years, Bermúdez et al. (2019) presented a least squared-support vector machine (LS-SVM) based method for spatial prediction of flood hazards. In their study, the authors applied the ML-based model to compute the spatial distribution of the maximum water depth and velocity in a coastal urban area using three sets of data for upstream flow and tidal level. Berkhahn et al. (2019) proposed an ensemble neural network model for real-time prediction of urban flooding. The model successfully predicted the maximum water levels during a flash flood event induced by spatially uniform rainfall. Wu et al. (2020) developed gradient boosting decision tree (GBDT)-based model to predict flood depths in urban areas. In their work, a data warehouse was first constructed using available structured and unstructured flood data, which was then used to fit the regression model.

3.3.1 Summary of ML in inundation modelling

To summarize, the application of ML in inundation modelling is understudied area to-date, and a limited number of studies investigated the potential of ML in outputting deterministic water depth maps. The research works done in this area can be grouped into two classes: i) fitting nonlinear models using inflow hydrographs for predicting water depths at selected locations, and ii) fitting cluster method and regression models to estimate water depths from rainfall measurements.

Approach 1, e.g., Bermúdez et al. (2019); Berkhahn et al. (2019); Liu and Pender (2015); Pan et al. (2011), aims to estimate water depth values at different locations within a floodplain using inflow hydrographs or rainfall as inputs to the model or rainfall from nearby gauging stations. Approach 2, e.g., Chang et al. (2010, 2014b); Chang et al. (2018a, 2018b, 2018c), applies clustering method to group flood zones under different hydrological conditions and linear/nonlinear models are used to generate water depth values for a particular hydrological condition (rainfall condition). Both the approaches

have produced promising results by utilizing the advantages of 2D hydrodynamic models and simple ML algorithms.

However, the proposed modelling approaches have limitations and need to be researched further. For example, none of this research attempted to quantify the model uncertainties. Specifically, in approach 2, when the forecasted water depth values have been used as an input recurrently to forecast water depths for the next time-step resulted in the accumulation of forecasting error and thus gradually increasing model uncertainty. Application of such models for generating deterministic flood maps in real-time for a longer temporal horizon can be misleading (significantly over or underestimating water depths). Furthermore, approach 2 requires a large number of inundation maps generated from 2D hydraulic models under a multitude of hydrological conditions for the clustering purpose. Producing a large number of inundation scenarios at very high spatial resolution can be extremely laborious and computationally expensive. Moreover, to reduce computational costs ground resolution of about 40 meters was used, which is considerably large, specifically for the urban areas.

While spatial resolution and recurrent use of water depth values were two key limitations of approach 2, approach 1 followed a different strategy to extract water depths. The approach 1 methods do not require a large number of inundation maps and are simpler than approach 2, but these approaches have certain limitations as well: i) they require an interpolation-based method to spatially distribute the water depth values, ii) their ability to generate multi-step ahead forecasting in real-time has not been fully tested yet, and iii) application of these approaches requires the upstream boundary conditions should be forecasted/measured before this method can be applied. Another point to take into account is that the performance of this modelling approach has not been investigated in large multichannel floodplains, and most of the applications are primarily focused on producing deterministic maps without an uncertainty quantification. An alternative to some of these limitations could be an approach of passing real-time inundation information as probabilities in the modelling procedure, which is novel and has been investigated in the present PhD work.

Chapter 4 **Streamflow Forecasting Using Machine Learning**

4.1 Introduction

To generate upstream boundary conditions for an integrated real-time rainfall-inundation modelling engine, the first step is to develop a robust rainfall-streamflow model. The idea is to generate multistep ahead streamflow forecasts of the channel at the outlet, which could be used for classifying inundated cells within a floodplain in the downstream. This is achieved by developing and then thoroughly accessing the predictive capabilities of selected ML-based models to investigate the best performing techniques for multistep ahead flow forecast.

Section 3.2 presented an overview of some key studies on modelling different hydrological variables. Most of these studies adopted a couple of ML techniques and compared the results against a conceptual model or a benchmarking model. These studies involve rainfall-runoff/streamflow modelling for generating multi-step ahead prediction using different data-driven techniques. Multi-step ahead forecasting with portability of the model parameters is significantly understudied. For the case of rainfall-runoff modelling, the widely applied procedure involves the use of a single type of ML technique and at each time step a separate model is trained and optimised for generating multi-step ahead forecasts. The predictive capabilities of modelling techniques are assessed by comparing the results against a reference model output (Chang et al., 2007) However, optimizing a system involving several ML-based models could be a tedious process. Further, optimizing a separate model for every time-step into the future for each catchment outlet may limit the transferability and scalability of the models.

To address some of these fundamental issues, it is important to answer two key questions:

1. Is it possible to calibrate an identical set of model parameters for a selected ML technique without compromising much of its model efficiency and forecasting abilities?
2. If it is possible to reuse model parameters for an ML method across multiple catchments, then for how many time steps into the future the set up can be used to generate reliable forecasts?

The objective of this chapter is to scrutinize these questions and identify a modelling technique which could be used for multi-step ahead flow forecasting.

In previous chapter (Chapter 3), an extensive discussion on the performances of different ML techniques for streamflow forecasting has been presented. It is beyond the scope of this thesis to compare the performances of all the ML techniques used for multi-step ahead streamflow forecasting. There are many techniques which can be chosen as the candidate models based on their types, for example, black-box models (e.g., ANN, SVR) and grey-box type models (e.g., GA/GP, RF). However, for the current study, three black-box models are carefully chosen based on their popularity and complexity, and thoroughly compared. Rationales for selecting model options are briefly discussed here. Previous studies (see **Sect.** 3.2) showed that the hybrid ANN models perform better than the standard ones. There are many data pre-processing techniques, e.g. Singular Value Decomposition (SVD) or Principle Component Analysis (PCA), which have been applied to hybridize ANN models. Amongst all, wavelet decomposition is one of the prevailing methods in model hybridisation and widely used for time-series forecasting. Therefore, a wavelet-based standard Feed Forward Back Propagation (FFBP) network is chosen as a candidate hybrid model. Considering the growing applications in various fields, including image processing and object detection, a deep neural network with multiple hidden layers is selected as the second candidate model (DBN). Finally, the SVR technique is selected as the third candidate model considering its robustness and predictive capabilities in nonlinear problem-solving (see **Sect.** 3.2).

Thus, considering the lack of comprehensive evaluation of multiple modelling techniques for multi-step ahead runoff forecasting, this chapter attempts to test the forecasting abilities of three techniques, namely, a hybrid model (WANN), a standard model (SVR), and a deep learning model (DBN) on three UK catchments for up to 1 to 6 hours ahead. The chapter is structured in the following manner: **Sect.** 4.2 describes the methodology adopted to reach the objectives of this study; **Sect.** 4.3 presents the experimental set up from data scaling to model implementation; **Sect.** 4.4 demonstrates the forecasting accuracies and uncertainty assessment of the candidate modelling schemes by comparing multiple error statistics, scatter and density plots of model residuals, and the model performance deterioration rate. It also presents results of a non-parametric gamma test, applied on the input dataset to gain some insight into the predictability of the output variable (river discharge)). **Sect.** 4.5and 4.6 present a discussion and the conclusion that

highlights the key findings and challenges of this study, as well as the scope for future developments.

4.2 Methodology

To conduct a comparative investigation of multistep ahead forecasting capabilities of multiple data-driven models, firstly, three data-driven techniques, i.e. WANN, SVR, and DBN, and three case study sites within the UK, i.e. the Lune at Killington, the Leven at Newby Bridge, and the Eden at Kirkby Stephen, were selected.

Secondly, the identical set of input data, i.e. rainfall and historical streamflow, were used in each of the three specified modelling schemes. As this study mainly focused on comparing predictive capabilities of widely used data-driven modelling techniques for multi hours ahead forecasting, less emphasis has been given on optimising the selection of input variables and datasets. Thus, for as long as the set of input variables remains the same, an unbiased analysis can be carried out to successfully achieve the overall objectives of this research. Preparation of input/output datasets will be explained in more details in a later section of this chapter.

Thirdly, Elshorbagy et al. (2010a) suggested that the forecasting accuracy of the various modelling schemes can be evaluated using four key error related statistics, i.e. the Root Mean Squared Error (RMSE), the Mean Absolute Relative Error (MARE), the Mean Bias (MB), and the correlation coefficient (R). In addition to the set of these four model performance evaluation criteria, the Nash-Sutcliffe model efficiency coefficient (NSE) was also used for model evaluation. The formulae of these five error measurements are as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(O_i - P_i)^2}{N}} \tag{4.1}$$

$$MARE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{O_i - P_i}{O_i}\right| \tag{4.2}$$

$$MB = \frac{1}{N}\sum_{i=1}^{N}(O_i - P_i) \tag{4.3}$$

$$R = \frac{\sum_{i=1}^{N}(O_i - \bar{o_i})(P_i - \bar{p})}{\sqrt{\sum_{i=1}^{N}(O_i - \bar{o})^2 \sum_{i=1}^{N}(P_i - \bar{p})^2}} \tag{4.4}$$

$$NSE = 1 - \frac{\sum_{i=1}^{N}(O_i - P_i)^2}{\sum_{i=1}^{N}(O_i - \bar{o})^2} \tag{4.5}$$

Where, N is the sample size, O_i and P_i are the observed and the predicted values, $\bar{o}$ and $\bar{p}$ are the mean of observed and predicted values. However, due to several statistics being used for error measurement, quite often conflicting results regarding the performance of different modelling techniques could be observed. To avoid such issues, Elshorbagy *et al.* (2010a) proposed a supplemental error measure that combines the effect of the four error measurements stated above as a single consistent performance indicator. The new error measurement statistic, referred to as Ideal Point Error (IPE), is based on identifying the ideal point in the four-dimensional space that each model aims to reach. The IPE, (**Eq.** 4.6), defines how far the model is from the ideal point. The value of IPE ranges from 0.0 for the best model to 1.0 for the worst model. Although the IPE gives a single indicator evaluation criterion making it simple for the assessor to evaluate a model, it should be noted that the IPE could potentially suffer from additional bias. This is because R and RMSE take model bias and variance into account whereas MARE and MB only consider bias. Therefore, the application of single indicator over multiple performance metrices is tested in this experiment.

$$IPE_{ij} = \left\{0.25\left[\left(\frac{RMSE_{ij}-0.0}{maxRMSE_{ij}}\right)^2 + \left(\frac{MARE_{ij}-0.0}{maxMARE_{ij}}\right)^2 + \left|\frac{MB_{ij}-0.0}{max|MB_{ij}|}\right|^2 + \left(\frac{R_{ij}-1.0}{1/maxR_{ij}}\right)^2\right]\right\}^{1/2} \quad (4.6)$$

Where, i denotes model and j denotes technique. Here in this study, $i = 1,2,3,4,5,6$ and $j = 1,2,3$.

Fourthly, the uncertainty quantification of these candidate forecasting models was conducted through a residual analysis approach. Estimated residuals for each forecasted time-series (generated by three candidate models) are presented using the regression plot, along with a probability density curve, which is fitted to the model residuals.

In addition, to facilitate a robust model validation, a Gamma test, namely *winGamma* (Jones et al., 2001, Durrant, 2001), was performed on the input variables. For a given set of input dataset, the Gamma test estimates the variance of noise associated with each output, which will be an estimate of the best mean squared error that a smooth model can achieve for the corresponding output (Jones et al., 2001, Durrant, 2001). The main aim of conducting the Gamma test is to gain some insight into the potential of selected input variables in robustly predicting the desired output variable. The test assumes that the non-determinism in a smooth model is due to the noise present in the outputs (**Eq.** 4.7):

$$Y = f(X_1 \dots \dots X_n) + \varepsilon \quad (4.7)$$

Where, $f(\cdot)$ is a smooth function and ε is the noise, and the variance of the noise is bounded. The Gamma test is based on *k* nearest neighbours: $X_{L[i,k]}(1 \leq k \leq p)$ for each input vector $X_i(1 \leq i \leq N)$ (Stefansson *et al.*, 1997). δ and γ functions can be defined as:

$$\delta_N(k) = \frac{1}{N}\sum_{i=1}^{N}\left|X_{L(i,k)} - X_i\right|^2 \quad (1 \leq k \leq p) \qquad (4.8)$$

$$\Gamma_N(k) = \frac{1}{2N}\sum_{i=1}^{N}\left|Y_{L(i,k)} - Y_i\right|^2 \quad (1 \leq k \leq p) \qquad (4.9)$$

Where, $Y_{L(i,k)}$ is the output corresponding to the output value for the *k* nearest neighbours; $X_{L[i,k]}$, and *p* are pre-selected values (Stefánsson et al., 1997). A least squares regression line can be fitted for the *p* points, using **Eq.** 4.8 and **Eq.** 4.9, from which the Gamma statistic can be estimated:

$$\gamma = \mathrm{A}\delta + \Gamma \qquad (4.10)$$

The Γ value is the intercept on the vertical axis (Jones et al., 2001). When $\delta_N(k)$ approaches 0, $\gamma_N(k)$ approaches the statistical noise variance. In addition to the Gamma value, three other important statistics can be estimated: (1) *gradient*: which is the slope of the regression line. The slope of the regression line indicates the complexity of the model, i.e. a steeper gradient indicates higher complexity. (2) *V-ratio*: is calculated by dividing Γ by the variance of the output noise. A V-ratio closer to zero indicates greater predictability of the output variable. And, (3) *M-test*: this gives an idea of the amount of the data required to generate a stable asymptote of Γ (Jones et al., 2001).

4.3 Experimental setup

4.3.1 Study area and data

To investigate the multistep ahead flow forecasting capabilities of the specified ML techniques, the catchment average rainfall and flow data from 1st September 2015 to 31st December 2015 were collected for three catchments located around Lake District National Park, located in the North-Western part of England (data available from the University of Bristol (**Fig.** 4.1)).

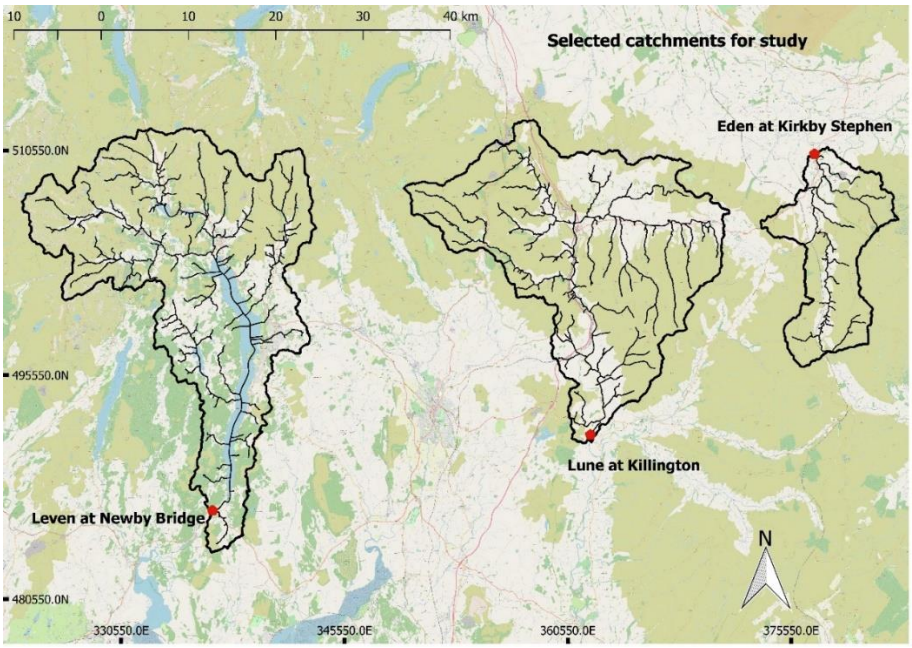


Figure 4.1: Selected catchments for the study.

Catchment 1: The Lune at Killington catchment has an area of *219 km*^{2}. Lune is a wet, natural, and rural catchment. It drains the Eastern Lake District Fells and North-western parts of the Yorkshire Dales National Park. Land cover and land use are dominated by moorland, grass, and arable farming. There are no abstractions or artificial discharges affecting the runoff; therefore, the gauged flow is considered to be within 10% of the natural flow at, or in excess of, the *Q95* flow (Young et al., 2003).

Catchment 2: The Leven at Newby Bridge catchment has an area of *247 km*^{2} and drains southern parts of the Lake District massif. The landscape is mainly dominated by grassland and is largely wooded in lower reaches. The catchment contains Windermere, Esthwaite Water, Rydal Water, Grasmere and numerous other small natural water bodies. Catchment runoff is greatly affected by reservoirs, i.e. it is reduced by water supply abstraction and increased by effluent returns.

Catchment 3: The Eden at Kirkby Stephen is a high relief catchment draining Carboniferous Limestone which forms most of the watershed. This is a small catchment covering an area of about *69.4 km*^{2} and is completely natural. No known abstractions affecting the runoff for the catchment. Descriptive statistics of the runoff for all three catchments are presented in Table 4.1, demonstrating that the selected catchments have considerably distinct statistical and hydrological characteristics.

	Lune at Killington (m^3/s)	Leven at Newby Bridge (m^3/s)	Eden at Kirkby Stephen (m^3/s)
Minimum	1.21	1.35	0.19
Maximum	621	220.75	139
Q5	1.32	2.21	0.21
Q95	104.77	90.88	36.51
Mean	28.69	30.51	8.06
Std. deviation	58.34	38.45	17.53
Coefficient of variation	211.06	119.47	217.51

Table 4.1: Descriptive statistics for all three catchments.

4.3.2 Data sorting and scaling

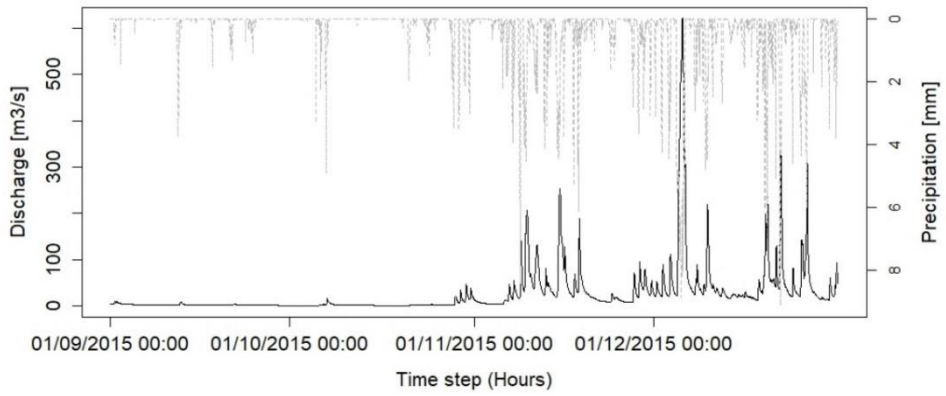
The original rainfall and flow time-series data are available at a temporal resolution of 15-minute intervals. Hourly rainfall-flow time-series were created from these 15-minute datasets by averaging the values associated within the hour. The rainfall-streamflow response curves show that variations in river discharge in September were not considerably significant (please refer to **Fig.** 4.2). Therefore, rainfall and flow data of September was discarded from further analysis (training/validation).

The hydro-meteorological datasets could have different units, and quite often their values do not represent the same quantities, hence, the dataset used for constructing models were normalised (to values between 0 and 1) using the following formula:

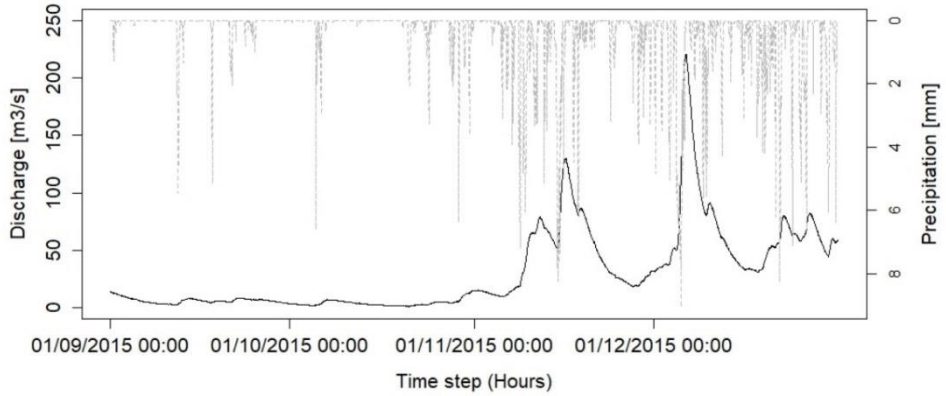
$$Z_{i,norm} = \frac{X_i - X_{min}}{X_{max} - X_{min}} \quad (4.11)$$

Where, X_i is the input variable; X_{max} and X_{min} are the maximum and minimum values of the variables, respectively. Normalized datasets were then divided into training (70% of the data) and testing subsets (30% of the data).

A



B



C

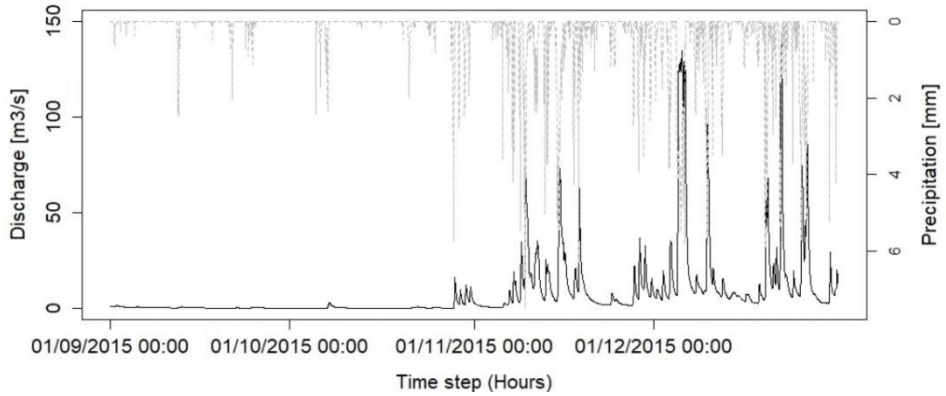


Figure 4.2: Rainfall-runoff response curves. Observation starts at 00:00 hours on September 1st, 2015 and ends at 23:59 hours on December 31st, 2017. Lune at Killington (A), Leven at Newby Bridge (B), and Eden at Kirkby Stephen (C).

4.3.3 Model implementation

First step towards model construction was to define the number of input variables (e.g. rainfall, discharge), lags associated with the hydro-meteorological variables, and to determine the model parameters. As already mentioned, to facilitate an unbiased analysis/comparison of multiple data-driven modelling techniques, same input variables were used for each modelling technique. Rainfall and flow values with their lags (as input variables) were estimated and assigned systematically to select the optimum set of input variables. Gamma test was applied to different input combinations, and the combination that produced the lowest Gamma score was selected for model development (see **Sect.** 4.7 for summary statistics of the Gamma tests). To further validate the application of the Gamma test in selecting the optimum combination of input variables, WANN-based model was trained and tested using few combinations of input variables to forecast 1h ahead flow for Lune at Killington station. The results obtained for these different combinations were compared using NSE and R^2 value. The output from the Gamma test analysis suggested an optimum combination of input variables, which consisted 5 lags of rainfall (R) variable and 3 lags of discharge (Q) variable.

Thus, the input feature matrix for the predictive models contained eight variables. The model structure for performing 1 to 6 hours ahead forecasting is defined as:

$$Q(t+n) = f_{(t)}[Q(t+1-Q_l), R(t+1-R_l)] \quad (4.12)$$

Where, $Q_l = 1,2,3$; $R_l = 1,2,3,4,5$; $f_{(t)}$ indicates the model function, i.e. WANN, SVR and DBN; t is the time index and $n = 1,2,3,4,5,6$.

The training and validation sets were carefully selected to ensure that for all the three catchments, the training set included the global maxima (the peak). The Lune at Killington training set had seven peak over thresholds (POTs), i.e. 127.364 m^3/s, events and the validation set included four POT events. The Leven at Newby Bridge had three POT (36.999 m^3/s) events in both the training and the validation set. The training set for the Eden at Kirkby Stephen catchment had five POT (48.092 m^3/s) events, and the validation set had four POT events. Figure 4.3 shows the flow series used for the training process for all the three catchments. The flow starts from 1st October 2015 at 00:00 hours counted as time step 0, and approximately 70% of the total data was used to train the models.

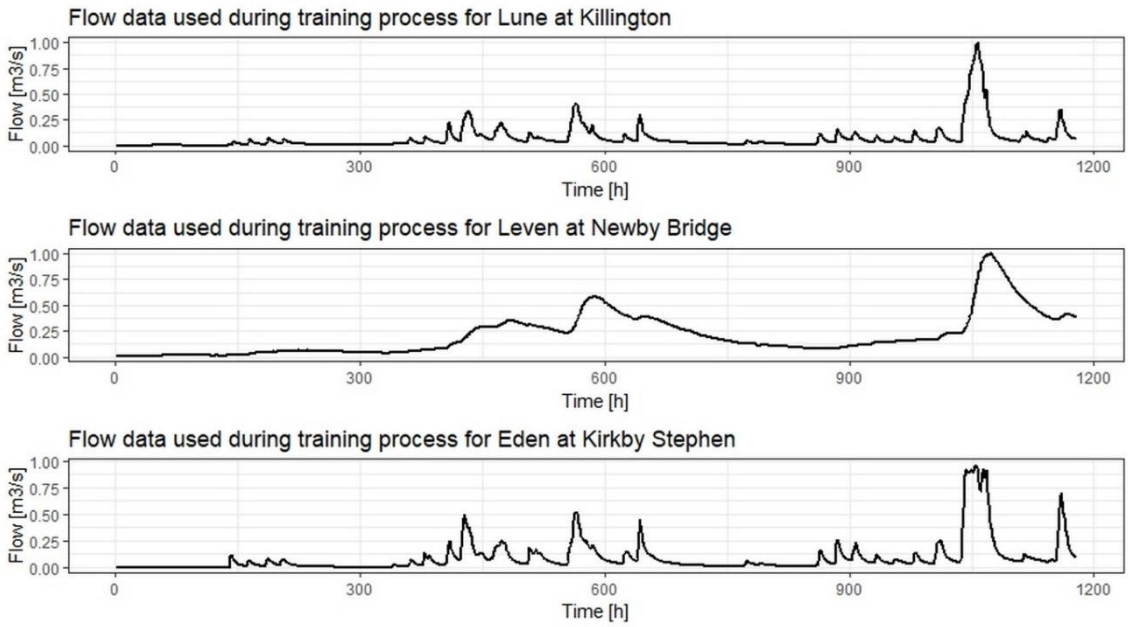


Figure 4.3: Observed flow data used to train the models. The Y axes are normalised flow data.

4.3.4 Wavelet - Artificial Neural Network (WANN) modelling approach

4.3.4.1 Wavelet transformation

In time-series data, the information gain from time-amplitude representation is limited and often not the best representation for signal processing applications. Therefore, transformations are done to extract information in data which is not readily visible in its original form (Shoaib et al., 2014). The Fourier Transform (FT), is possibly one of the most widely used approaches for extracting information in the frequency domain from a stationary signal (Kaur and Singh, 2014). FT of a signal in the time domain gives a frequency-amplitude representation. However, FT is not always suitable for a non-stationary signal where the frequency components do not exist at all times. This is where wavelet transformation (WT) is probably the most effective approach because it provides a time-frequency representation of a non-stationary signal.

In this study, the WT was carried out on the input feature matrix which contains time-series of the eight input variables (discharge at *t, t-1* and *t-2*, and rainfall at *t, t-1, t-2, t-3* and *t-4*) (described in **Sect. 4.3.3**) using the *Daubechies* wavelet (db5 with level 3) in MATLAB (see Shoaib et al. (2014) for more details on wavelet decomposition, wavelet families, and levels). A snippet of the code is given below, where at first, a signal (time-series data) is decomposed to several subseries called the approximation and detail

coefficients. For example, at level 3, the time-series s1 (rainfall) was decomposed into four subseries (i.e. one approximation and three detail coefficients). The WT was applied to all eight input variables, resulting in a total number of 32 decomposed input time-series.

```
% Perform a multilevel wavelet decomposition to hourly time-series (s1)

% Daubechies wavelets (db5)

% At level 3

[C1,L1] = wavedec(s1,3,'db5');

% Reconstruct the Level 3 approximation and the Level 1, 2, and 3 details

A31 = wrcoef('a',C1,L1,'db5',3);

D11 = wrcoef('d',C1,L1,'db5',1);

D21 = wrcoef('d',C1,L1,'db5',2);

D31 = wrcoef('d',C1,L1,'db5',3);
```

Figure 4.4 displays the approximation and detail coefficients of the rainfall time-series. WT decomposed the signal into a weighted linear combination of scaling function (father wavelet) and wavelet function (mother wavelet). Coefficients (weights) associated with the scaling function, called the approximation coefficient, captured low frequency information associated with the scaling function, while the detail coefficients captured high-frequency information associated with the wavelet function.

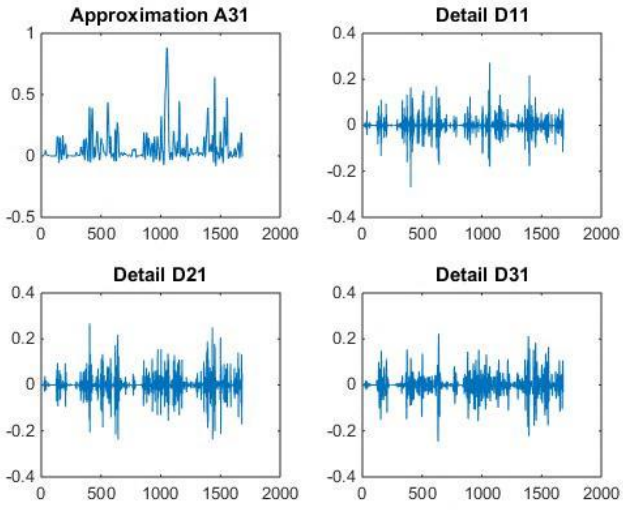


Figure 4.4: The resulted approximation and detail coefficients using Daubechies 5 wavelet.

4.3.4.2 ANN configuration

The approximation and detail coefficients from all eight input variables (total 32) are fed into a fully connected FFBP neural network with one hidden layer. While more than one hidden layer can be used, a large number of previous studies suggest that for rainfall-runoff modelling one hidden layer with sufficient number of hidden units is enough for achieving a desired output (Vijayashanthar et al. (2018), Kourgialas et al. (2015), Jain and Kumar (2007), Khan and Coulibaly (2006)). Thus, the structure of the ANN model used herein consisted one input layer, one hidden layer and one output layer. The input layer included 32 nodes for processing 32 input variables, flow values at $t+1h, t+2h, \ldots.,$ and $t+6h$ were used as the target data for the output layer. The structure of WANN is shown in Figure 4.5, where, W_{ij} and W_{jk} represent weights between the layers. Please note, a bias term is added to the hidden and output layers. Details of the symbols are provided in **Sect.** 2.3.1.

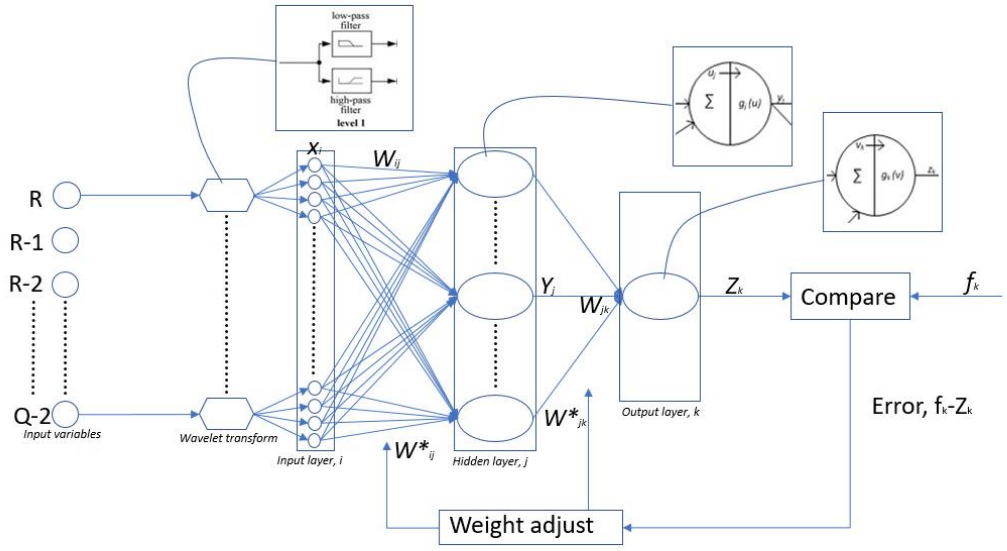


Figure 4.5: Architecture of the WANN-based model.

Tuning hyperparameters of a general ANN structure, i.e. selecting hidden units, learning rate, number of training iteration, activation function, optimization algorithm, in order to optimize model performance is a labour intensive process because these are the parameters that are not learned by the network during the training process (Vijayashanthar

et al., 2018). A range of hidden units was tested during the training process (10 to 60 at an interval of 10). Since there were 32 input variables at each input-output instance, this was a reasonable range of hidden units to build the model without overfitting. Multiple activation functions were also considered: The candidate activation functions were *sigmoid*, *tanh*, and *linear*. The training of WANN-based model was carried out following a heuristic trial and error approach, and the network performance was monitored at each iteration. The number of nodes in the hidden layer was systematically increased from 10 to 60, and the optimum number of nodes were estimated for the combination of lowest SSE and highest R values. The final WANN-based model architecture consisted 50 neurons in the hidden layer. In this case, the best results were found using the *sigmoid* activation function for both the hidden and output layer. The widely used *Levenberg-Marquardt* algorithm was used as the optimizer, and the optimal network configuration was derived using T+1h flow as the target output. As one of the research objectives was to observe the rate of performance degradation with forecasting going ahead in the time, the network architecture was kept identical for training all 18 (1 to 6 hours for all three catchments) WANN-based models.

4.3.5 Support Vector Regression (SVR) modelling approach

A general consensus can be drawn from the previous studies that the SVR-based rainfall-runoff model is a strong technique for predicting future river discharge (**Sect.** 3.2.1). However, issues around finding the optimum parameter settings for calibrating the best model remains unclear. The two-step exponential search method automates the searching task but requires large computational memory and time. Also, it does not guarantee a global maximum. On the other hand, applying trial and error process to find parameters can be extremely labour intensive. A possible way to reduce the workload could be using nu-SVR, a variant of traditional epsilon SVR, where the percentage of support vectors is chosen instead of parameter ε (Abe, 2010). This approach overcomes the problem of estimating the optimal value of ε, which, by definition, can be any positive real number.

Thus, in this experiment, the nu-SVR (ν-SVR), a variant of SVR, was selected as the preferred technique. The SVR is sensitive to the kernel choice; therefore, the kernel function is required to be selected carefully. Numerous studies have suggested that for modelling nonlinear regressions, the 'radial basis function (RBF)' is more effective than other kernel functions (Elshorbagy et al., 2010b, Han et al., 2007, Lin et al., 2006). The *e1071* package (Dimitriadou et al., 2011) in the *R* software was used to implement the

SVR method to all normalized datasets. Note, there was no data pre-processing method such as WT was applied for the SVR-based modelling. Therefore, the time-series of eight input variables (discharge at *t, t-1* and *t-2*, and rainfall at *t, t-1, t-2, t-3* and *t-4*) was directly used as the model inputs and flow values at $t + 1h, t + 2h, \ldots,$ and $t + 6h$ were used as the target. The RBF kernel was used for training the model, and an exhaustive grid search method was applied to find the optimal values (a *Python* script was written using *Scikit-learn* library for parameter optimization through grid search). Interestingly, the optimal parameter values obtained from the grid search method produced lower accuracy than the values obtained from the trial and error approach. The 'cost', 'gamma' and 'nu' values were searched from 0.001 to 1000, 0.001 to 1, and 0.001 to 1, respectively for the SVR-based model. The parameter values obtained through the trial and error process providing better results than the grid search method was selected for the further predictive analysis. The selected parameter values were: *C =10, nu= 0.5, gamma= 0.5.* 18 distinct SVR-based models were developed and tested using the corresponding testing dataset. The *R* code snippet of an SVR-based model, as an example, fitted to Lune at Killington data for a 1h ahead flow forecast is given below:

```
#Import libraries

library(e1071)

#Load normalized training data

df_tr <- read.csv("Lune_Killington_train_1hr.csv")

#Load normalized test data

df_test <- read.csv("Lune_Killington_test_1hr.csv")

df_test <- df_test[,-9]

#'nu-regression' type model fit

nu.svm.fit<-svm(df_tr$Q~.,      df_tr[,-9],      type="nu-regression",
kernel='radial', cost=10, nu=0.5, gamma = 0.5, scale=F)

summary(nu.svm.fit)

#Make predictions

nu.sim <- predict(nu.svm.fit, df_test)
```

4.3.6 Deep Belief Network (DBN) modelling approach

The DBN network presented here was constructed in *Python* using the *dbn* package (Albertbup, 2017). The model implementation utilizes *Numpy* and *Tensorflow* (Abadi et al., 2015) libraries. Similar to the WANN and SVR techniques, the network parameters, i.e. learning rate, batch size, number of hidden layer nodes, and number of hidden layers, were perturbed manually to optimize the model performance using training datasets. Particular care was taken while selecting the optimum "batch size" - the number of training samples present in a single batch. The DBN technique is susceptible to overfitting if the larger batch size is selected, whereas a smaller batch size could increase the computational cost. The optimum values for the network parameters are shown below in Table 4.2. 18 separate DBN-based models were trained using the same set of model parameters and the training dataset contained eight inputs (discharge at *t*, *t-1* and *t-2*, and rainfall at *t*, *t-1*, *t-2*, *t-3* and *t-4*) and flow values at $t + 1h, t + 2h, \ldots,$ and $t + 6h$ as the targets, and then these models were tested on the corresponding testing datasets.

Parameters	Values
Number of hidden layer nodes	100
Number of hidden layers	5
Learning rate for the RBM	0.01
Learning rate for the DBN	0.01
Number of epochs for RBM	10
Number of back propagation iterations	200
Batch size	10
Activation function	'relu' (Rectified Linear Unit)

Table 4.2: The DBN parameters.

An example of a *Python* code for forecasting 1h ahead flow:

```
# Load libraries

import numpy as np

import pandas as pd

import os

from sklearn.metrics.regression import r2_score, mean_squared_error
```

```
from sklearn.preprocessing import MinMaxScaler

from dbn.tensorflow import SupervisedDBNRegression

# Load raw dataset (unnormalized)

file = pd.read_csv('Lune_Killington_train_1hr.csv')

file_test = pd.read_csv('Lune_Killington_test_1hr.csv')

# Drop the target column and keep input variable columns

train_inp = file.drop(['q1'],axis=1)

test_inp = file_test.drop(['q1', axis=1]

# Keep the target column by dropping the input variable columns

train_out = file.drop(['R','R1','R2','R3','R4','Q','Q1','Q2'],axis=1)

test_out                                                              =
file_test.drop(['R','R1','R2','R3','R4','Q','Q1','Q2'],axis=1)

# train and test data

X_train, Y_train = train_inp, train_out

X_test, Y_test = test_inp, test_out

# Data scaling

min_max_scaler = MinMaxScaler()

X_train = min_max_scaler.fit_transform(X_train)

# Training process

regressor = SupervisedDBNRegression(hidden_layers_structure=[100,5],

                        learning_rate_rbm=0.01,

                        learning_rate=0.01,

                         n_epochs_rbm=10,

                         n_iter_backprop=200,

                         batch_size=10,

                         activation_function='relu')


regressor.fit(X_train, Y_train)

Y_pred_train = regressor.predict(X_train)

Rsqrd = r2_score(Y_train, Y_pred_train)

Mse = mean_squared_error(Y_train, Y_pred_train)
```

```
print(Rsqrd , Mse)

# Testing process

X_test = min_max_scaler.transform(X_test)

Y_pred = regressor.predict(X_test)

Df = pd.DataFrame(Y_pred)

df1 = pd.DataFrame(Y_test)

result = pd.concat([df1,df], axis=1)

os.chdir("C:\\Users\\taaj_\\Downloads\\PhD\\Results\\DBM\\72005\\"

result.to_csv('Lune_Killington_1hr_pred.csv')
```

4.4 Results and analysis

4.4.1 Lune at Killington

The trained predictive models were applied to the test datasets for performance evaluation. It should be noted that before applying the models to the test data, it was made sure that the models were not overfitted. An overfitted model would be very accurate in predicting the outputs for the training data but it would perform poorly when test data is used. This means that the models will not be generalised. To avoid this, training R^2 values were compared with the test R^2 values. Initial tests revealed that the models were not overfitted.

The performance analysis of the WANN, SVR, and DBN techniques for multi-hour ahead streamflow forecasting on the test data was compared for the Lune at Killington case study and is provided in Table 4.3. It can be observed that the different forecasting techniques have different performance responses measured across different error statistics and along different hours in the future. Forecasting techniques for which each error statistics was lowest is highlighted in 'bold'. From the analysis of the values presented in the table, it can be concluded that, for generating forecasts for the first two hours, SVR performed significantly better than the WANN and DBN. However, for forecasting third to sixth hours onwards, the performance of the WANN appears better in comparison to the SVR and DBN techniques. Interestingly, various error statistics (except MB) corresponding to DBN seemed to be considerably large in comparison to the other forecasting techniques, suggesting that the DBN was less biased than the other two

methods. The IPE indicator (presented in Table 4.6) provides a considerably good representation of the overall performance of each technique. The SVR-based models have the lowest IPE values, for forecasting streamflow at one to five hours ahead, though at the sixth hour WANN has lowest IPE value. However, the averaged IPE value corresponding to 6 hours ahead forecasting outcomes indicate that the SVR method has the most efficient forecasting abilities compared to the WANN and DBN methods.

Hour	WANN					SVR					DBN				
	RMSE (m)	*MARE*	*MB (m)*	R^2	*NSE*	*RMSE (m)*	*MARE*	*MB (m)*	R^2	*NSE*	*RMSE (m)*	*MARE*	*MB (m)*	R^2	*NSE*
One	12.04	0.11	1.01	0.95	0.94	**7.84**	**0.04**	0.54	**0.99**	**0.99**	18.12	0.22	**0.41**	0.91	0.90
Two	21.50	0.14	1.30	0.88	0.82	**14.12**	**0.08**	1.02	**0.94**	**0.93**	21.06	0.27	**0.61**	0.85	0.83
Three	**21.14**	0.16	4.19	**0.84**	**0.83**	21.38	**0.12**	1.38	**0.84**	**0.83**	27.10	0.31	**0.38**	0.76	0.72
Four	**21.42**	0.18	4.43	0.85	**0.83**	30.07	**0.16**	1.22	0.71	0.66	32.97	0.34	**0.80**	0.65	0.59
Five	**28.09**	0.24	-7.25	0.83	**0.74**	38.56	**0.21**	2.02	0.56	0.43	39.12	0.37	**-0.63**	0.52	0.42
Six	**28.66**	**0.23**	1.79	0.86	**0.73**	46.77	0.24	4.33	0.42	0.23	46.07	0.41	**0.80**	0.41	0.26

Table 4.3: Comparing five different error statistics (RMSE, MARE, MB, R^2 and NSE) for three forecasting techniques (WANN, SVR and DBN) for 1 to 6 hours ahead forecast for the river Lune at Killington on the test data.

For visual comparison, the predicted values of flow generated through the applications of the three different ML techniques were compared against the test data and illustrated in Figure 4.6. It appears that at the *T+3h* and further ahead the SVR-based models overestimated the peaks with slight delays. Figure 4.7 shows, the scatter plots of observed data compared with the forecasted streamflow data for 1, 3 and 6 hours ahead generated by the WANN, SVR, and DBN methods. From visual inspection of the graph, it can be noted that for forecasting one hour ahead streamflow values the residual scatter plots for SVR generated values fits closest around the 45-degree line, which implies low error values. However, for forecasting 3 hours ahead, the plots for the WANN generated values fit better than the SVR, though, on closely inspecting the points from the best fit line, it appears that the SVR could have performed slightly better than the WANN. Thus, it can be said that the SVR performed comparatively better than the WANN and DBN for forecasting 1-3 ahead streamflow. Moreover, the performance of the WANN is comparatively better than the SVR and DBN methods for forecasting 6 hours ahead streamflow values. Thus, the scatter plots, supplement the conclusion made earlier (in Table 4.3 using the Error statistics and in Table 4.6 using the IPE indicators) on the predictive capabilities of the three techniques.

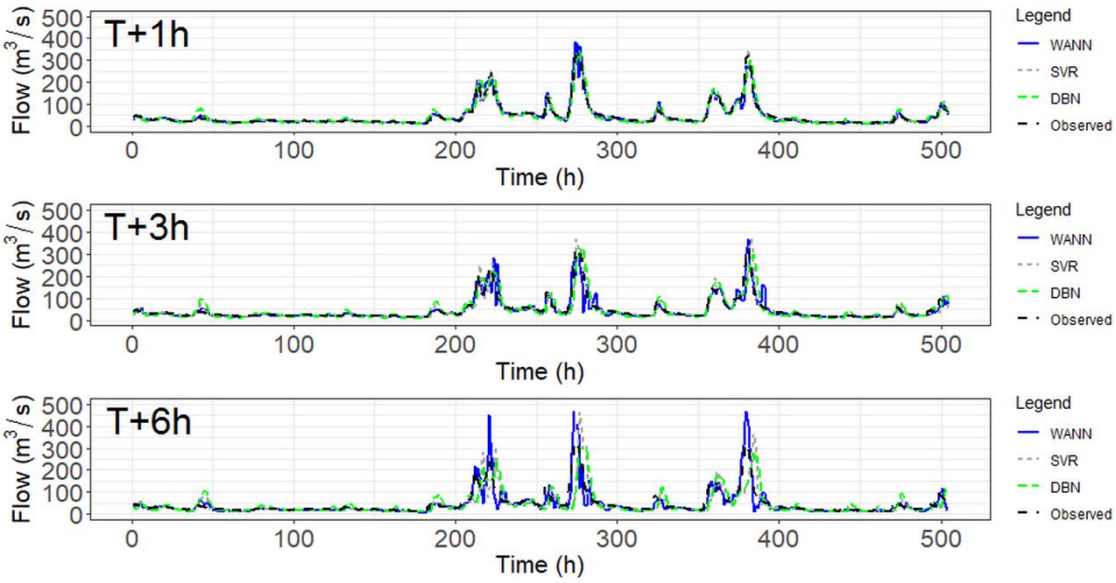


Figure 4.6: Comparing the predicted flow against the test data at different lead times for Lune at Killington catchment.

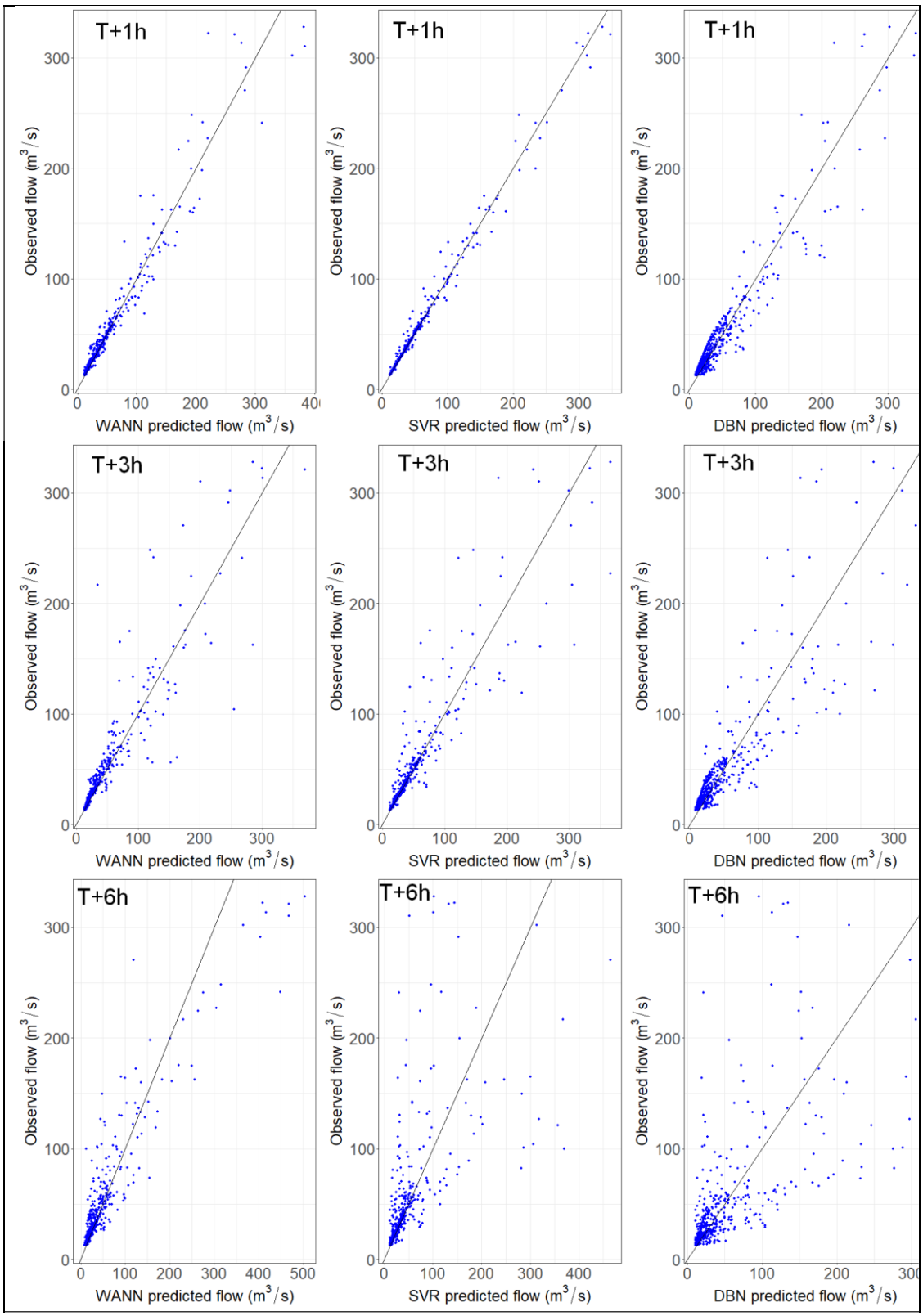


Figure 4.7: Comparing the forecasting techniques (WANN, SVR and DBN) for forecasting one (row 1), three (row 2) and six hours (row 3) ahead streamflow values for the river Lune at Killington on the test data.

4.4.2 Leven at Newby Bridge

The second case study location was river Leven at Newby Bridge with considerably different rainfall-runoff process in comparison to the first case study. The runoff process for the Leven case study is greatly affected by the abstractions. The response of catchment reduces by the water supply abstractions and increases by effluent returns. Table 4.4 presents the analysis of error statistics for 1- to 6-hour-ahead forecasts for the three forecasting techniques. It can be observed that the 18 SVR-based models have consistently performed better than the WANN and DBN-based models for generating 1 to 5 hours ahead forecast values across all the error measuring criteria. For the sixth hour ahead forecast, the WANN-based model appear to perform better than the SVR-based model. The DBN-based models seem to generate satisfactory results, although the error statistics are slightly higher than the other two modelling approaches. The analysis of the IPE indicator, providing a comprehensive representation of the overall performance of each technique, is presented in Table 4.6.

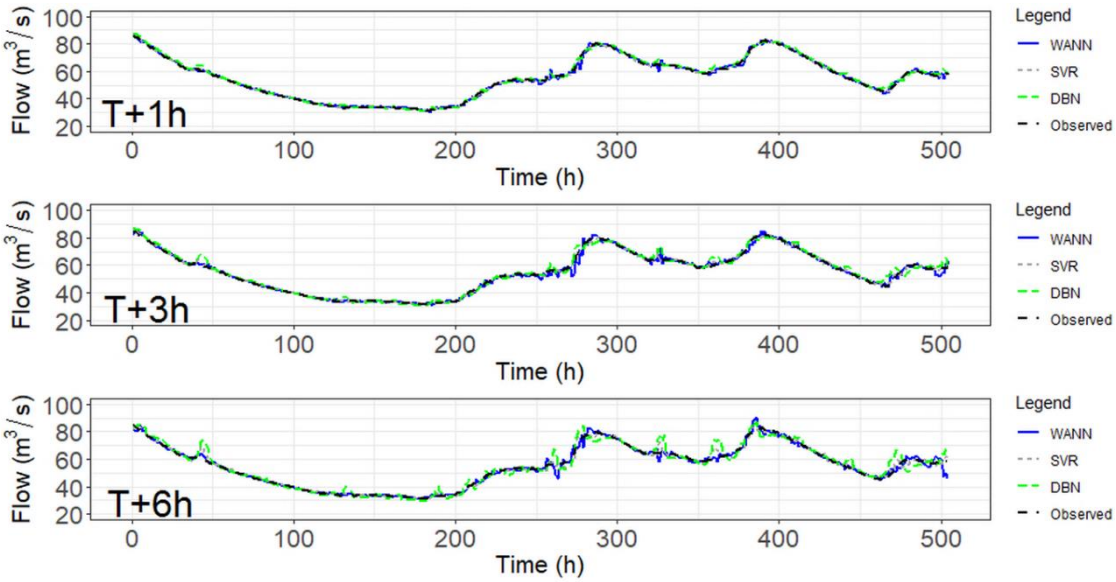


Figure 4.8: Comparing the predicted flow against test data at different lead times for Leven at Newby Bridge catchment.

Hour	WANN					SVR					DBN				
	RMSE (m)	*MARE*	*MB (m)*	*R^2*	*NSE*	*RMSE (m)*	*MARE*	*MB (m)*	*R^2*	*NSE*	*RMSE (m)*	*MARE*	*MB (m)*	*R^2*	*NSE*
One	1.00	0.01	-0.20	1.00	1.00	**0.55**	**0.01**	**0.01**	**1.00**	**1.00**	1.30	0.02	-0.59	0.99	0.99
Two	1.44	0.01	0.18	0.99	0.99	**0.74**	**0.01**	**0.01**	**1.00**	**1.00**	1.81	0.02	-0.55	0.99	0.99
Three	1.54	0.02	0.06	0.99	0.99	**0.99**	**0.01**	**-0.01**	**1.00**	**1.00**	2.18	0.03	-0.56	0.98	0.98
Four	1.62	0.02	0.29	0.99	0.99	**1.27**	**0.02**	**-0.03**	**0.99**	**0.99**	2.75	0.04	-0.28	0.97	0.97
Five	1.86	**0.02**	0.09	0.98	0.98	**1.52**	**0.02**	**-0.06**	**0.99**	**0.99**	4.29	0.04	-0.50	0.95	0.95
Six	**1.86**	**0.02**	**-0.01**	**0.98**	**0.98**	**1.86**	**0.02**	-0.11	**0.98**	**0.98**	4.81	0.05	-0.51	0.94	0.93

Table 4.4: Comparing five different error statistics (RMSE, MARE, MB, R^2 and NSE) for the three forecasting techniques (WANN, SVR and DBN) for 1 to 6 hours ahead forecast for the river Leven at Newby Bridge on the test data.

The visual comparison between the predicted flows against the test data (**Fig.** 4.8) clearly shows that while the SVR-based models were more stable and outperformed the other two models at *T+1h* and *T+3h*, the WANN-based model predicted more accurately the peaks occurring at *T+6h*. Further, Figure 4.9 (following the analogy of Figure 4.7) displays the scatterplots of the observed streamflow values with the forecasted values at 1, 3 and 6 hours ahead in the future. On closely inspecting the scatter plots of the forecasted models, the SVR-based models performed better than the WANN and DBN-based models for 1 to 5 hours ahead forecasting. While the R-squared values for both the WANN and SVR are same for 6 hours ahead forecasting, the scatter plot for the WANN could have close fit around the 45-degree line in comparison to the SVR. The analysis of IPE indicators further confirms these results (presented in Table 4.6).

Moreover, unlike the SVR and DBN-based models, the predictive performance of the WANN-based model does not decrease gradually with every time step ahead in the future. This means that when nonlinearity increases in the model, WANN can provide a considerably more reliable forecast than the SVR and DBN-based models if the model parameters are kept constant.

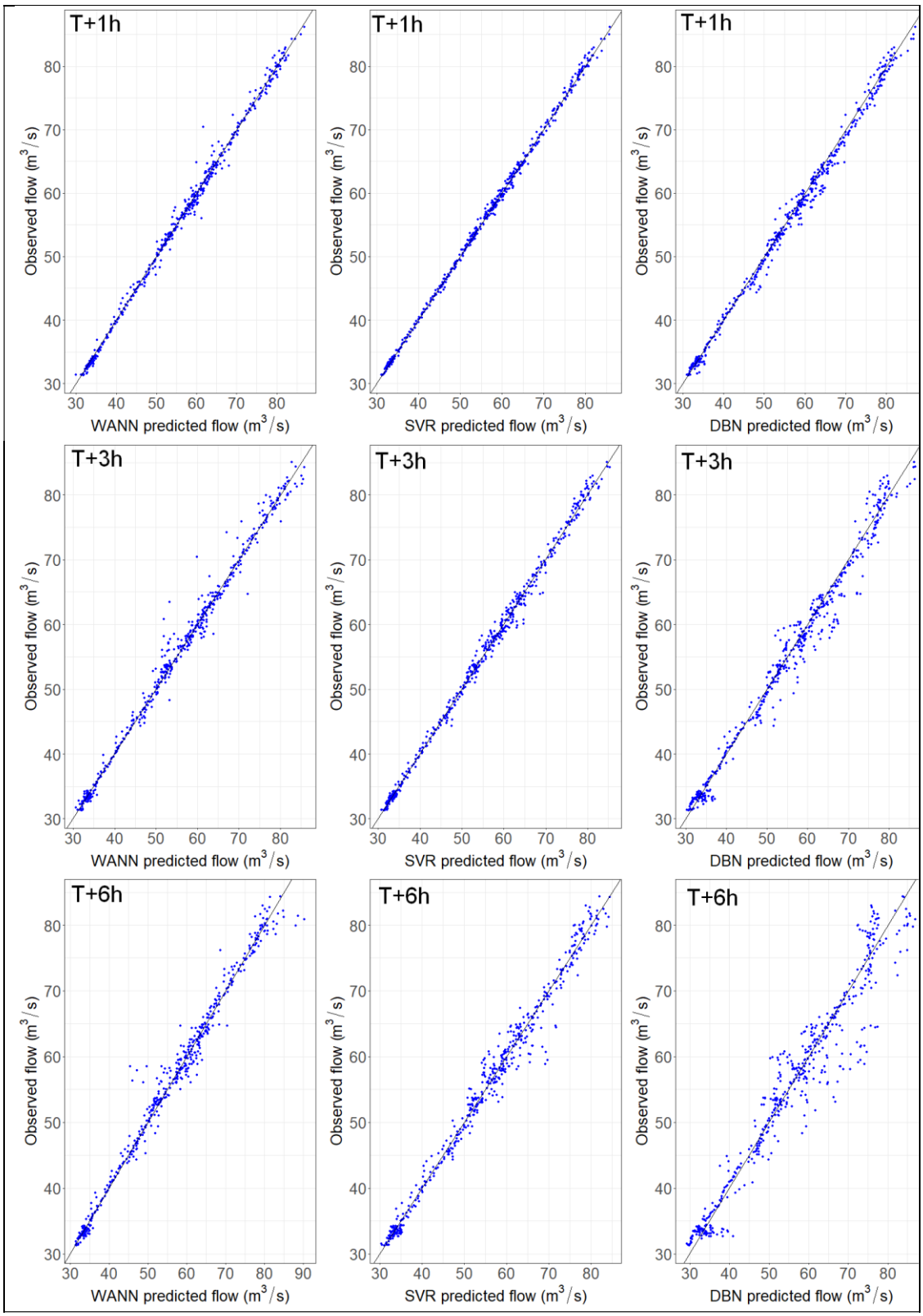


Figure 4.9: Comparing the forecasting techniques (WANN, SVR and DBN) for one- (row 1), three- (row 2) and six-hours (row 3)-ahead forecast for the river Leven at Newby Bridge on the test data.

4.4.3 Eden at Kirkby Stephen

Third case study, Eden at Kirkby Stephen, has similar geomorphological characteristics to that of the Lune at Killington. Comparison of performance analysis, using five different error statistics, for the three ML based modelling techniques is given in Table 4.5. The SVR-based models appear to perform considerably better than the other two modelling approaches for generating short-term forecasts, i.e. 1 and 2 hours ahead in the future, though the WANN-based models could provide better performance with an increase in the complexity of the system, i.e. at generating long-term forecast. Notably, the overall performance of the SVR-based models appear to degrade significantly for forecasting 3-hours ahead time stamp, and the WANN-based models could provide reliable outcomes (with lowest RMSE, highest coefficient of correlation, NSE, and lowest MB observed for four out of six-time steps). The analysis of the IPE indicator further supplements the findings of the error statistics analysis presented in Table 4.5 for 1 and 2 hours ahead forecasting, i.e. the SVR-based models perform better than the other two models. Interestingly, the comparative analysis of IPE measure (Table 4.6) seems to differ than the analysis of general error statistics (presented in Table 4.5) for 3 to 6 hours ahead forecasting. Except for MARE, for all other error statistics, the WANN-based models provide consistent lowest error measures. Thus, it is reasonable to expect that the WANN-based model could have lower IPE values and could perform better than the other two modelling approaches. Although the performance of the DBN-based model is considerably low, the IPE indicator shows that for 5 and 6 hours ahead forecasting, the DBN-based models are performing better than the other two models. This is counterintuitive and requires further investigation.

Figure 4.10 shows the observed and predicted flows from the three different models. The graph shows that the SVR-based models capture the peaks better than the other two models when the lead time is short (1-2 hours), while the WANN-based models are performing better than the SVR and DBN-based models for generating 3 to 6 hours ahead of forecasting. These findings are not in agreement with the IPE indicators.

Hour	WANN					SVR					DBN				
	RMSE (m)	*MARE*	*MB (m)*	*R^2*	*NSE*	*RMSE (m)*	*MARE*	*MB (m)*	*R^2*	*NSE*	*RMSE (m)*	*MARE*	*MB (m)*	*R^2*	*NSE*
One	6.17	0.07	-0.30	0.96	0.96	**1.82**	**0.03**	**0.21**	**0.99**	**0.99**	6.84	0.18	0.51	0.94	0.94
Two	6.46	0.11	**-0.11**	0.95	0.95	**4.82**	**0.07**	0.56	**0.96**	**0.96**	8.35	0.20	0.41	0.90	0.90
Three	**6.72**	0.18	**0.77**	**0.95**	**0.94**	8.25	**0.11**	1.07	0.90	0.90	8.18	0.22	0.32	0.83	0.83
Four	**7.13**	0.16	**-0.20**	**0.88**	**0.87**	9.21	**0.16**	1.64	0.79	0.78	10.25	0.32	1.09	0.74	0.73
Five	**8.92**	0.23	0.90	**0.88**	**0.88**	11.78	**0.20**	1.94	0.65	0.64	12.32	0.28	**0.26**	0.63	0.61
Six	**8.33**	**0.23**	**0.80**	**0.84**	**0.82**	16.31	0.23	2.36	0.50	0.47	16.19	0.30	0.86	0.51	0.48

Table 4.5: Comparing five different error statistics (RMSE, MARE, MB, R^2 and NSE) for the three forecasting techniques (WANN, SVR and DBN) for 1 to 6 hours ahead forecast for the river Eden at Kirkby Stephen on the test data.

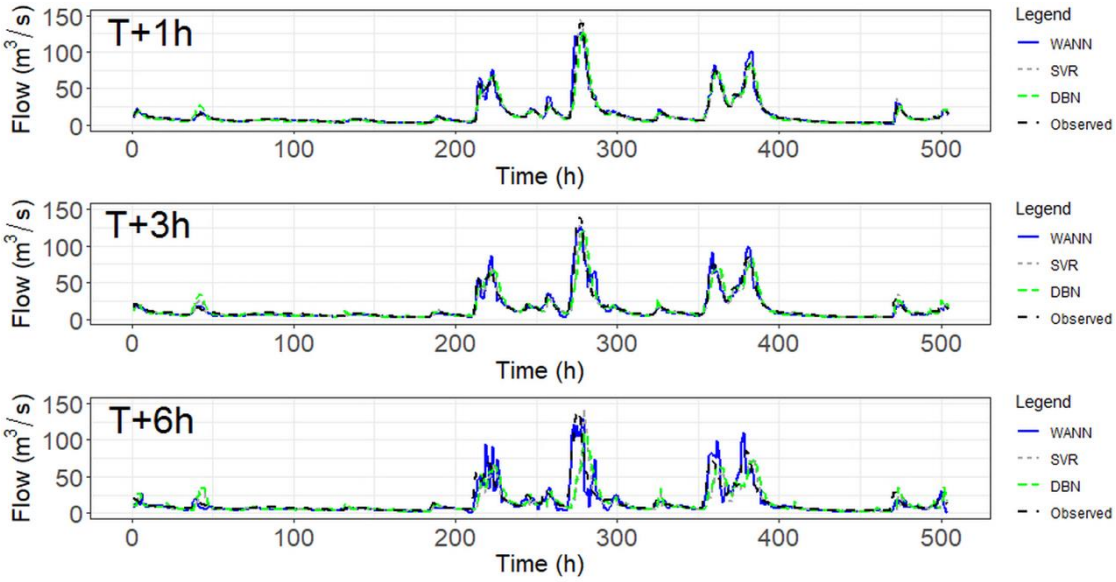


Figure 4.10: Comparing the predicted flow against validation data at different lead times for Eden at Kirkby Stephen catchment.

	Lune at Killington			Leven at Newby Bridge			Eden at Kirkby Stephan		
Hour	IPE WANN	IPE SVR	IPE DBN	IPE WANN	IPE SVR	IPE DBN	IPE WANN	IPE SVR	IPE DBN
One	0.33	**0.13**	0.42	0.54	**0.22**	0.56	0.34	**0.10**	0.41
Two	0.52	**0.27**	0.56	0.61	**0.30**	0.58	0.37	**0.23**	0.43
Three	0.60	**0.40**	0.54	0.59	**0.39**	0.62	0.65	**0.40**	0.48
Four	0.64	**0.51**	0.76	0.81	**0.51**	0.57	**0.57**	0.58	0.80
Five	0.86	**0.69**	0.76	0.71	**0.64**	0.75	0.82	0.72	**0.63**
Six	**0.72**	0.88	0.88	**0.71**	0.87	0.83	0.84	0.88	**0.81**
Avg.	0.61	**0.48**	0.65	0.66	**0.49**	0.65	0.60	**0.49**	0.59

Table 4.6: Comparing the Ideal Point Error (IPE) statistics for the three forecasting techniques (WANN, SVR and DBN) for 1 to 6 hours ahead forecast for all three case studies on the test datasets.

4.4.4 Few words on the model uncertainty

To quantify the model uncertainty, probability density distribution (PDD) graphs of residuals, obtained from the three ML-based modelling techniques, was analysed for all the three case study sites and is presented in Figure 4.11. The plots show the spread of the distribution of residuals for each modelling techniques. For a good fit, the PDD plots of the residuals, in general, follow a normal distribution with a mean close to zero. The

standard deviation (SD) of PDD plots represents quantitative uncertainty in the predictive capabilities of the model. Thus, a PDD plot with large SD represents the large model uncertainty. The SD of PDD plots of the SVR-based models for 1 hour ahead prediction is close to a normally distributed curve for all the three catchments. However, with each step increment in time, the PDD of residuals for SVR-based model deviates from the zero mean, indicating that the model predictions become biased and more uncertain with added nonlinearity.

The PDD plots of residuals for three hours ahead forecast demonstrates that the SVR-based model could perform with less uncertainty than the other two modelling approaches, while WANN could provide predictions with less uncertainty (comparative to the other two approaches) only in the regions of 6 hours ahead of forecasting.

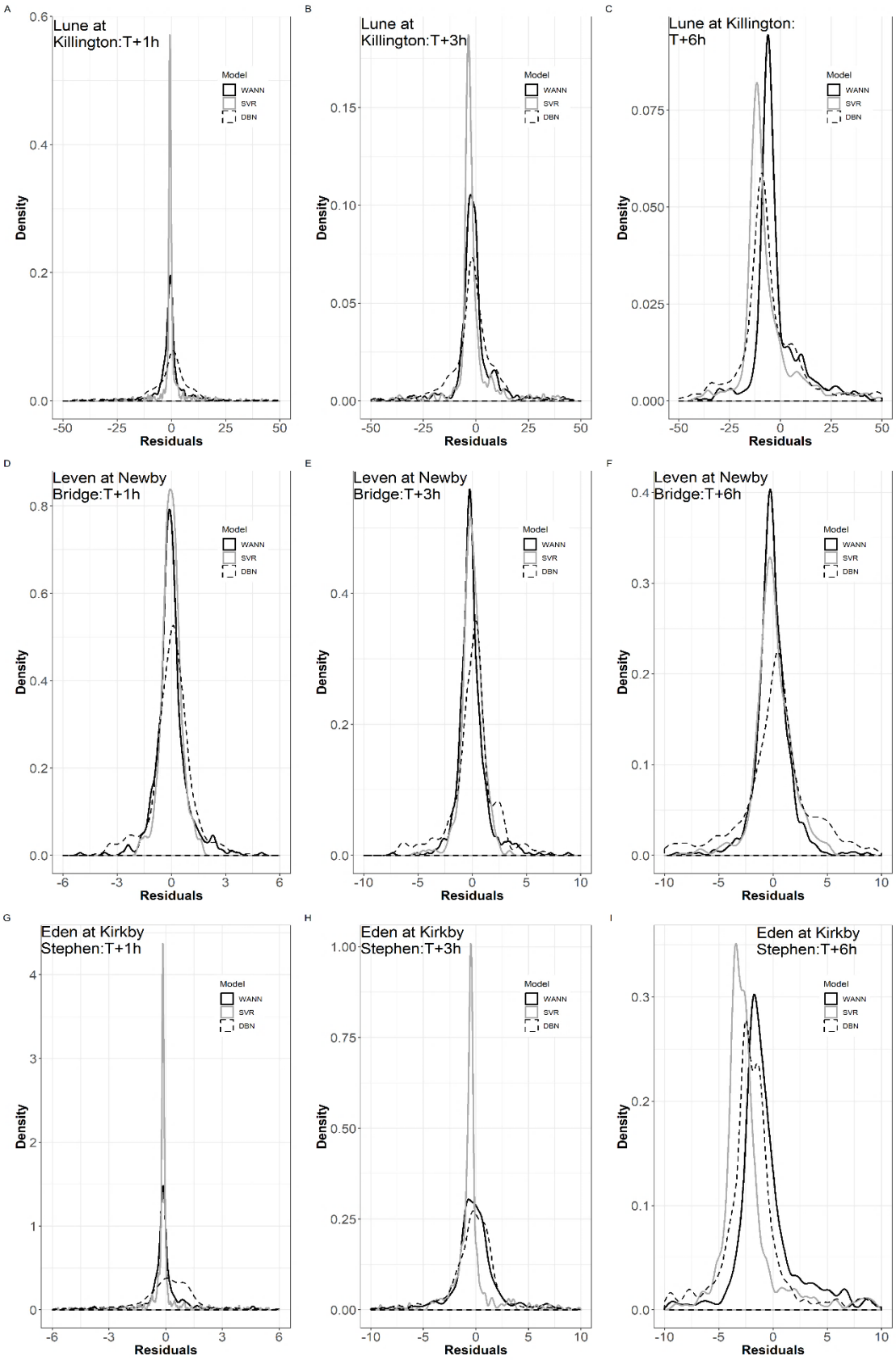


Figure 4.11: PDD plots of model residuals (modelled flow – observed flow in m3/s). Rows 1, 2 and 3 represent Lune at Killington, Leven at Newby Bridge and Eden at Kirkby Stephen, respectively. Columns 1, 2 and 3 compare densities at 1, 3 and 6 hours ahead forecast.

4.4.5 The non-parametric Gamma test

The *winGamma* software (Jones et al., 2001, Durrant, 2001) was used to explore the predictability, and complexities of the three modelling procedure discussed in the previous section (**Sect.** 4.2). Table 4.7 provides the summary statistics of the key components of the test.

	Lune at Killington			**Leven at Newby Bridge**			**Eden at Kirkby Stephen**		
Γ- Stat	1 Hour	3 Hour	6 Hour	1 Hour	3 Hour	6 Hour	1 Hour	3 Hour	6 Hour
Gamma	0.0009	0.0028	0.0219	-0.0015	-0.0019	-0.0023	0.0005	0.0096	0.0499
Gradient	0.0404	0.0857	0.1557	0.0451	0.0520	0.0612	0.0459	0.0682	0.0841
Std. Error	0.0005	0.0009	0.0028	0.0002	0.0003	0.0005	0.0006	0.001	0.0034
V-ratio	0.0035	0.0112	0.0877	-0.0059	-0.0077	-0.0094	0.0018	0.0383	0.1996

Table 4.7: Gamma test results.

The V-ratio, gradient and gamma statistic were calculated using the scaled data. A lower gradient value means that a noncomplex smooth function, can be used for modelling the process, and V-ratio closer to zero indicates the higher predictive abilities of model. The standard error in the gamma statistics shows the reliability of the test. On analysing the V-ratio and gradient values from the table above, it is evident that the model complexity increases as we move from one to six hours ahead for prediction and overall predictability decreases concurrently. The gamma test can also be applied to select the best model among the potential candidate models, and this is particularly useful when multiple techniques perform equally consistent. Further, M-test can be used to define the minimum number of samples required to develop a predictive model. However, in this study, M-test was not used because it was assumed that the training data contained sufficient input-output instances to cover flow variabilities (e.g. high and low flow events).

4.5 Discussion and conclusion

The objective of this study was to investigate whether there are real differences between the predictive capabilities of three major and widely applied machine learning techniques in projecting multi-step ahead streamflow forecasts with a selection of identical model parameters. To develop a full ML based rainfall-inundation model for real-time flood forecasting, it is important to account for the model scalability. Constructing multiple models for multiple lead times for each catchment may not be a viable option. In other

words, it could be more efficient and realistic to develop a single robust model which would forecast multistep ahead flows with minimum added uncertainty. This was done in the current study.

The investigation conducted the performance evaluation of three ML techniques using multiple error measurements and an analysis of the error distributions. The SVR technique-based models are found to be more reliable, than the other two modelling approaches for forecasting 1 to 3 hours ahead streamflow, these findings are drawn from the analysis of multiple error measure statistics, the IPE values, residual regression plots, and PDD. The WANN-based models are found to perform better than the other two approaches when the system nonlinearity/noise is comparatively large, i.e. for forecasting long term flow in the range of 4 to 6 hours ahead. Thus, for short-term forecasting, i.e. *T+1h* to *T+3h*, the SVR-based model should be the preferred modelling option, while the WANN can be adopted for long term forecasting by keeping the model parameters same. Alternatively, a new SVR-based model configuration could be designed to generating reliable forecast for time stamps >*T+3h* ahead.

It was interesting to see that in some cases the IPE values did not reflect the expected outcomes. For instance, from hour 3-6, a lower IPE value was expected for the WANN. This could possibly be the consequence of the error metrices used to calculate the IPE. This is because all four metrices quantify biases, however, variance is quantified by the RMSE and R only. Therefore, the added bias could have resulted in unexpected IPE values. Although, the aggregation of multiple error metrices provide a single measure, it is possibly better to use individual metrices to avoid errors caused by additional biases.

In this study, 70% of the total input-output instances has been used for training the models and 30% for testing. In the realm of the present study, PDD plots showed that most of the models analysed here were not effective in simulating high flow projections for more than 2 hours ahead, and especially if the catchment is sensitive (Lune at Killington and Eden at Kirkby Stephen in this case). For a generalized overview of the deterioration rate, the R and NSE measures were further investigated. The SVR and DBN-based model performance deteriorated drastically for each hour ahead of forecasting. For example, the R and NSE values of the SVR technique deteriorate to 36.4% and 74.5%, the WANN deteriorate to 6.9% and 21.6%, and the DBN deteriorate to 31.5% and 66.3% (Lune at Killington).

The structure of the WANN and DBN-based models was optimised through an iterative process, that is, validate the models using various combination of parameter sets/subsets for forecasting at *T+1h*. In other words, the best fit WANN and DBN-based models were selected from a multitude of candidate models. For SVR, the optimum set of parameters was identified through an exhaustive grid search method and an intensive trial and error process. Interestingly, parameter values found through the trial and error process produced slightly better results than the standard grid search method. Therefore, the parameter values realized through the trial and error process were chosen for SVR-based model configuration. This finding also suggests that in future research one should apply multiple methods to define optimal parameters.

The current study used antecedent rainfall and discharge values to forecast 1-6 h ahead streamflow. It could be possible to include forecasted rainfall as an input variable in the future work. However, inclusion of forecasted rainfall can only be considered if the uncertainty of the forecasts is low.

Chapter 5 **Inundation Modelling Using ML: Case Study 1**

5.1 Introduction

The chapter presents a step-by-step procedure for developing and testing multiple ML algorithms for generating dynamic real-time flood inundation maps. These algorithms were trained and validated using observed upstream flows, and outputs generated from a 2D hydraulic model. To evaluate the performance of the models, a rural catchment which is affected by seasonal flooding was selected (Niger Inland Delta in Mali) as a case study. This is a large rural catchment and the flood dynamics is controlled by two major river channels (i.e. River Niger and Bani). And the yearly hydrometeorological data required to simulate the models are readily available from earlier studies (e.g. Neal et al., (2012)). In this study, a SVR-based and a multilayer perceptron (MLP)-based modelling approach were applied for producing inundation maps at daily time-step.

The focus of this chapter is to investigate the applicability of ML-based algorithms to produce dynamic flood maps (deterministic binary and probabilistic) in real-time. At this stage, less attention was paid to the grid resolution and for absolute calibration of the 2-D hydraulic model (LISFLOOD-FP). It was anticipated that if the proposed ML-based model can satisfactorily reproduce the results of the 2-D model, irrespective of calibration procedure, then the model should be able to emulate outputs from a fully calibrated 2-D model as well. Thus, the performance of the ML-based models in the future can be improved by training outputs from a well calibrated 2-D model.

In the following section, a brief description of the site, along with the details of available hydro-meteorological and topographical data, is presented. Then the strategies for developing SVR and MLP-based models are explained. Results obtained from both the approaches are discussed in **Sect.** 5.7. Concluding remarks are presented in **Sect.** 5.8.

5.2 Site description

The case study site is comprised of an 800 km stretch of the River Niger, covering an area of 210,389 km^2 in Mali. The River Niger has a low gradient, descending ~20 m over the test site, and bifurcates into an Inland Delta below the confluence with the River Bani,

before returning to a single braided channel near Timbuktu. The Niger Inland Delta is in a semiarid region, and the majority of the discharge in the river derives from upstream of the case study site. The surface of the floodplain is covered with a network of small channels (Neal et al., 2012). The ecoregion of the landscape is highly dynamic, complex and interacts with the ebb and flow of seasonal flooding (CILSS, 2016). The dynamics of natural flooding in the region can be seen in Figure 5.1.

The three Landsat images from May, September, and December 2015 capture the dynamics of natural flooding in the region. The first image shows the extreme dryness at the peak of the dry season with the presence of large permanent and semi-permanent water bodies illustrated in dark blue and green colours in the map of the delta. The flooding season starts in July when water levels in both Niger and Bani begin to rise (CILSS, 2016).

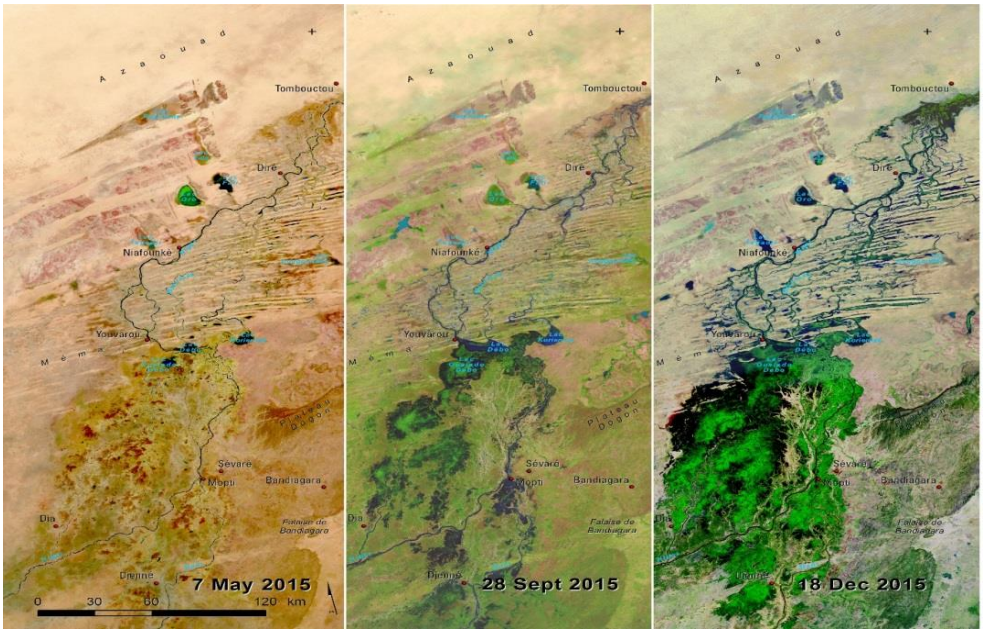


Figure 5.1: Landscape diversity and water dynamics of the Inland Niger Delta (CILSS, 2016).

The late September image (middle panel in **Fig.** 5.1) shows the natural flooding of the Inland Delta. The southern part of the delta gets under the influence of the Niger and the Bani water level rise, while the northern part of the delta experiences a delay in flooding for about two to three months. This is because the flooding permeates a total area of about 40,000 km^2 between the southern and northern floodplain. During the December period (right panel in **Fig.** 5.1) the southern delta is mostly drained while the water level rises in the northern delta (CILSS, 2016).

5.3 Hydro-meteorological and topographical data

The modelling procedure utilises daily river inflow data, which were collected over a continuous period starting from 1st January 2002 to 14th September 2009 and pulled from two gauging stations: i) Kirango Aval [Niger near Markala: Global Runoff Data Centre (GRDC) station 1134250] gauge; and ii) Beneny Kegny (Bani: GRDC station 1134450) located 13km upstream (**Fig.** 5.2). Considering that the total precipitation occurring over the case-study area could have a relatively small impact on the overall discharge, this information has been excluded from the modelling procedure (Neal et al., 2012). However, due to the high evaporation rate observed in the region (Dadson et al., 2010), the evapotranspiration data were included in the LISFLOOD-FP simulation. The summary statistics of the flows for both rivers are given in Table 5.1.

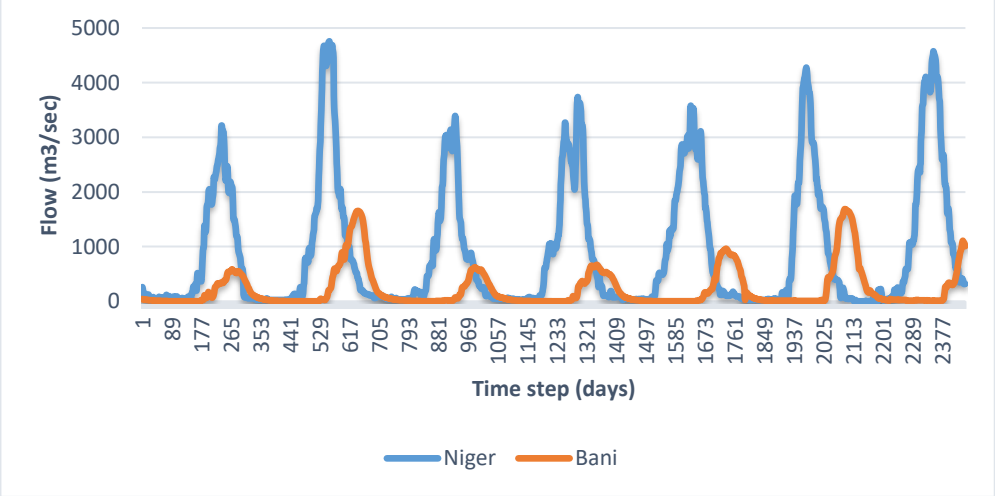


Figure 5.2: Observed flows at the upstream gauging stations. Daily observations start on 1st January 2002 as day 1 and stop on 14th September 2009 as day 2448.

River	Minimum	1st Quartile	Median	Mean	3rd Quartile	Maximum
Niger	0	57.235	301.7	970.7033	1512.75	4758
Bani	0	7.0165	28.67	212.8355	319.15	1653

Table 5.1: Flow statistics of River Niger and Bani.

To run the hydraulic model, topographical information was acquired from the Shuttle Radar Topography Mission (SRTM). Specific to the case-study region, the SRTM DEM has a ground resolution of approximately 90m, and elevation errors noted to have a

standard deviation of 6.68m over Africa (Rodríguez et al., 2006). To reduce the elevation errors and to represent floodplain connectivity between cells over flat terrain for a flow depth of 1m, the DEM was resampled to 905m resolution (Neal et al., 2012). Other river bathymetric data acquired for model development include bank heights (estimated using the elevation of the floodplain adjacent to the channel) and channel widths (estimated from Landsat Enhanced Thematic Mapper (LETM) imagery with a pixel resolution of 30 m) (Neal et al., 2012).

5.4 Model development

This section presents an overview of the novel ML based methodology developed for real-time flood inundation modelling. The underpinning idea is that a database of flood inundation time-series (generated from 2D hydrodynamic/hydraulic models using observed/synthetic inflow hydrographs) for selected locations within a floodplain or an entire floodplain (subject to cell resolution) can be used to develop a purely data-driven model by training a suite of ML algorithms for real-time flood inundation maps and forecasting future flood events. A 2-D model, namely LISFLOOD-FP, was set up and simulated to generate flood inundation (water depth) time-series data.

5.4.1 LISFLOOD-FP simulation

5.4.1.1 LISFLOOD-FP hydraulic model background

The 2-D LISFLOOD-FP (Bates and De Roo, 2000), utilized in this study, is a raster-based flood inundation model for simulating fluvial or coastal flood spreading. The LISFLOOD-FP uses an explicit forward difference scheme on a staggered grid using a local inertial approximation of the 1-D shallow water equations which represent the continuity of mass and continuity of momentum in a plane (Bates et al., 2010).

The LISFLOOD-FP is computationally efficient, easy to calibrate, and suitable for a gradually varied flow. Notably, this model could become unstable at low Manning's n values (less than 0.01) or under supercritical flow conditions (Almeida and Bates, 2013, Bates et al., 2010) but these are trivial in this study. The model is equipped with numerous solvers to simulate flood wave propagation along the river channel and across floodplains. Specific to the present study, the *subgrid* solver (Neal et al., 2012), an extension of the LISFLOOD-FP, was used. The *subgrid* solver allows river channels with any width below the grid resolution to be simulated.

The 2D hydraulic model usually requires input information on floodplain topography, river channel widths, banks heights, friction parameters, hydrological datasets such as inflow hydrographs, and optionally rainfall and evapotranspiration (Neal et al., 2012). To calibrate model, information on floodplain topography was accessed through remotely sensed DEM, and river channel widths and banks heights data were derived from satellite images of the geographic area. Information on hydrological data can be retrieved from land-based river gauges or outputs from hydrologic models, and friction parameters are typically collected from lookup tables or through model calibration.

5.4.1.2 Model setup

External data required to simulate the LISFLOOD-FP model are provided in a particular file type. For example, DEM has to be in '.asc' format. The boundary conditions for the domain, i.e. upstream and downstream locations are provided in a '.bci' file and details of the time-varying boundary conditions, i.e. flow values, are given in a '.bdy' file. For the 'subgrid' channel flow solver, channel width and channel bank elevation files also need to be supplied in '.asc' format. For a detailed description of different file types and modelling options, see the LISFLOOD-FP user manual (Bates et al., 2013). All of these required files along with the model parameters are listed in a so-called parameter file:

Parameter file (.par): the model is controlled by this steering file. It contains all the necessary information, i.e. other key files and controlling parameters, required to run the model. The parameter file instructs LISFLOOD-FP what data to read in and which options to use for the simulation. The parameter file used to simulate the Niger flood inundation modelling case can be seen in Figure 5.3.

```
#Parameter file for the Niger

DEMfile          NigerLLdem.asc
resroot          res
dirroot          resultsnig_test3_Q
sim_time         220838400.0
initial_tstep         500.0
massint          86400.0
saveint          86400.0
bdyfile          NigerLL.bdy
bcifile          NigerLL.bci
evaporation      NigerLL.evap
SGCwidth         NigerLL.width.asc
SGCbank          NigerLL.bank.asc
SGCchangroup     NigerLL.region.asc
SGCchanprams     NigerLL.pram
fpfric                0.05
latlong
SGCbfh_mode
```

Figure 5.3: The parameter file containing the list of controlling files and parameters.

For the present work, as mentioned before, all the required hydro-meteorological and topographical datasets (catered in LISFLOOD-FP file format) were readily available from Neal et al. (2012). Other parameter settings included a uniformly distributed roughness/friction coefficient, which was set to 0.05. Time–step for simulation was set to 500 seconds and the model output, i.e. simulated flood waves involving time-series data of water depths, saving time was set to 86400 seconds (24 hours). Using these parameters, the LISFLOOD-FP model was run from 1st January 2002 to 14th September 2009.

Seven years of flow data from River Niger and Bani allowed one-year initialization period for the model, which is more than adequate to provide initial conditions for the next wet season (Neal et al., 2012). Over the entire simulation period, a total of 2446 raster files (ASCII files) were generated. The model took about 40 minutes on a 2.5 GHz Intel Core i5-2520 processor with 8 GB RAM to finish the simulation.

Considering the focus of present study, which was to demonstrate the applicability of ML algorithms in projecting flood maps using the output from a hydraulic model, detailed information on the procedure of LISFLOOD-FP model calibration and validation are not presented here. Subsequently, detailed validation procedure of flood inundation modelling using different validation measures (satellite images, observed downstream discharge, etc.) is not conducted and presented. Emphasis was given to the facilitate

simulation correlation between the hydraulic model and data driven models for a given set of inflows.

Since the LISFLOOD-FP required one year of initialization period, the output raster files from 1st January to 31st December 2002 were deliberately excluded from the next stage of the work, i.e. the training and testing procedure of the proposed ML algorithms.

5.4.2 Configuration of the regression based SVR algorithm

Similar to the method proposed by Liu and Pender (2015), a simple SVR-based model was developed using the upstream discharge and corresponding discharge observation time as the model inputs and water depths (LISFLOOD-FP output) as the target/output variable.

The SVR-based modelling approach involved six key steps. The proposed methodology is presented below:

Step 1 *Generate input data:* Observed flow data from 1st January 2003 to 14th September 2009 for River Niger and Bani (i.e. upstream time varying boundary condition data) were selected as the primary input for the SVR-based model. In addition to flow data, corresponding time in 'hours' was also used as the third input. The reason for selecting time as a separate input variable is that for any arbitrary pixel in the spatial domain being flooded or not depends on the gauged water level and the time of arrival of the flood wave.

Step 2 *Generate target data:* To create a continuous time-series of target data, water depth values for each pixel obtained from the LISFLOOD-FP simulated raster files (**Sect.** 5.6.1) were extracted from 1st January 2003 to 14th September 2009. The procedure of doing this in *R* language is presented below:

```
##Clear history

rm(list=ls(all=TRUE))

#Import libraries

library(raster)

library(sp)

library(rgdal)

##import rasters
```

```
data<-
list.files("C:\\Users\\taaj_\\Lisflood_Sim_Data\\raw_train_wd",
pattern=".wd", full.names=T)

d<-seq(1, length(data), by=2) #only select water depth (.wd files)

data<-data[d]

my_data<-lapply(data, raster)

r<-stack(my_data)

proj4string(r)<-CRS("+init=epsg:4326")  #set coordinate system

nlayers(r)

r.min = cellStats(r, "min")

r.max = cellStats(r, "max")

##Normalize raster stack

cv<-(r-r.min)/(r.max-r.min)

##random selection of points

##this is to create a csv file with pixel coordinates and

##water depth time-series

rd<-raster(data[1])

proj4string(rd)<-CRS("+init=epsg:4326")

pp<-as.data.frame(rasterToPoints(rd))

write.csv(pp[,3],

"C:\\Users\\taaj_\\Phase2\\Niger\\alldatapoints\\alldatapoints.csv")

##Import              pixel              locations              from
##C:\\Users\\taaj_\\Phase2\\Niger\\alldatapoints

l<-read.csv("C:\\Users\\taaj_\\alldatapoints\\alldatapoints.csv")

coordinates(l)<-~x+y

proj4string(l)<-CRS("+init=epsg:4326")

sp <- SpatialPoints(l)

##extract water depth values

val<-extract(cv,sp)

m<-t(val)

n<-as.data.frame(m)

write.csv(n, "C:\\Users\\taaj_\\Raster_Data_All_Year_Normalized.csv")
```

The ‘csv’ file contains the normalized water depth time-series and coordinates for every cell.

Step 3 *Training the model:* In this experiment, similar to the previous chapter, the nu-SVR (ν-SVR) was selected as the preferred technique. The model was trained using the input and target data produced from steps 1 and 2 (above) for the period 2003 - 2007 and 2009 (training set). The dataset from 2008 was retained and used for validation purpose.

It is important to note that, constructing a separate model, for each pixel within the study domain is practically impossible. Therefore, as an initial step, four random locations within the flood plain were arbitrarily selected to analyse the possibility of using the same model parameters, and to compare the accuracy of the predictions quantitatively (by calculating error statistics) and qualitatively (by visualising whether the model is stable during the high flows). For each of these locations, four separate models were constructed and trained using the data from 2003 to 2008. These models were tested using data from 2009. The optimal model parameters were searched through a labour-intensive trial and error approach. Final parameter values were estimated as: *C =20, nu= 0.5, gamma= 0.5, Kernel= RBF*. The initial investigation showed that during the high flow season, the SVR-based models performed equally for all test locations (for more details see **Sect.** 5.5.1). Thus, a separate ML-based model can be trained for each cell in the domain using an identical parameter configuration. Note that the models were unstable during the dry season, however, considering the objective of the study, i.e. to simulate inundation extent during the flooding season, these observations were deemed irrelevant.

Step 4 *Predicting inundation extent:* To predict the inundation extent, the leanings from step 3 was applied across each pixel in the study area, and a separate model was calibrated for each cell using the identical set of model parameters (calibrated in step 3). The 2008 dataset (validation set) was used to test the model outputs. Pixels that never flooded during the study period were excluded from the training phase, and hence set to a default value of 0 (dry pixels) for the test data. This means that the pixels which were never flooded during the study period were automatically termed as dry pixels. The entire process of training and predicting were conducted following through an iterative process. The *R* code for the combined iterative training and prediction steps is presented below:

```
#Import libraries

library(e1071)

DF<-data.frame(X, Y) #dataframe containing pixel coordinates

##create empty dataframes to store predicted water depths

DF1<-data.frame(matrix(ncol = 1,nrow = 365))

##trDF.set5 is the csv file containing water depth time-series. Each
column represents a ##pixel and each row represents a time step (i.e.
1 day)

for (i in names(trDF.set5)) {

df<- data.frame(wd=trDF.set5[,i])

#'nu-regression' type

dat<-inpDF.set5     #normalized flow and time variable

if(df$wd > 0){

  Qout<-df$wd

  nu.svm.fit<-svm(Qout~.,dat,  type="nu-regression",  kernel='radial',
cost=20, nu=0.5, gamma=0.5, scale=F)

  listofmdls[[i]] <- nu.svm.fit   ##list fitted models

  ##make predictions

  testData<-trDF5  #validation set

  sim<-predict(nu.svm.fit,testData)

  }

else{

  sim<-0

}

DF1[[paste(i)]]<-sim

}

DF1<-DF1[,-1]

gf<-as.data.frame(t(DF1))

gf[gf<0] <- 0

fDF<-cbind(DF,gf)

write.csv(fDF,"SVM_MODEL_PREDICTIONS_FULLRASTER.csv")

save(listofmdls, file="ListOfModels")
```

Step 5 *Binarization:* Outputs were generated on a daily timescale for the entire year of 2008, and for every pixel. Pixel values greater than 0 represents as a wet cell and hence encoded as 1, whereas a 0 value represents a dry pixel. These spatial points were then rasterized to produce dichotomous maps. The output files were then imported into QGIS for visualization and further analysis. A snippet of the *R* commands used are presented below:

```
##save as raster
setwd("C:\\Users\\taaj_\\Niger\\Results\\daywise")
files<-list.files(pattern = ".csv")
for (j in 1:365){
  setwd("C:\\Users\\taaj_\\Niger\\Results\\daywise")
  file<-read.csv(files[j])
  file<-file[,-1]
  # X & Y column of the csv contains the coordinates
  coordinates(file) <- ~ X + Y
  # coerce to SpatialPixelsDataFrame
  gridded(file) <- TRUE
  # coerce to raster
  rasterDF <- raster(file)
  proj4string(rasterDF)<-CRS("+init=epsg:4326")
  #save as raster
  name=paste(files[j])
  setwd("C:\\Users\\taaj_\\Niger\\Results\\TIFF")
  writeRaster(rasterDF,name, format="GTiff")
}
```

Step 6 *Measuring classification accuracy:* for a quantitative measure of model classification accuracy, the following contingency table (Table 5.2), accounting for the number of cells in each category, was used (Aronica et al., 2002):

		2D Model output	
		Dry	Wet
ML-based model output	dry	A= dry-Dry	B= dry-Wet
	wet	C= wet-Dry	D= wet-Wet

Table 5.2: Contingency table for calculating classification accuracy.

Category A: is the total number of the cells correctly classified as dry by both the 2D hydrodynamic model and ML algorithms.

Category B: is the number of cells misclassified as dry by the ML algorithms (false negative).

Category C: is the total number of cells misclassified as wet by the ML algorithms (false positive); and

Category D: is the total number of cells correctly classified as wet by both models. The fit, *F*, between the ML-based models and the hydraulic model simulation can then be calculated using the following formula:

$$F(\%) = \frac{D}{B+C+D} \times 100 \qquad (5.1)$$

5.4.3 Configuration of the classification based MLP algorithm

Training and implementing an MLP can be comparatively more complex than the SVR approach; therefore, it could be computationally challenging to configure a separate MLP-based model for every cell in the case-study domain. The strategy applied to overcome the computational burden was to train a separate MLP classifier at the sampled locations, instead of the entire study domain, to predict the flooding condition of a cell. The procedure of developing a simple MLP based classification approach to predict probabilistic flood extent maps is presented below:

Step 1 *Sample selection:* To simplify the modelling procedure, while accounting for the overall accuracy of the model, a sample of 200 cells from the main floodplain was

randomly selected (**Fig.** 5.4). In classification-based problems, selection of an evenly distributed sample size (consisting of an approximately equal number of wet and dry cells) is widely considered to reduce the prediction bias. Of course, during the wet season, most of the cells will be flooded, while during the dry season, the scenario will be different.

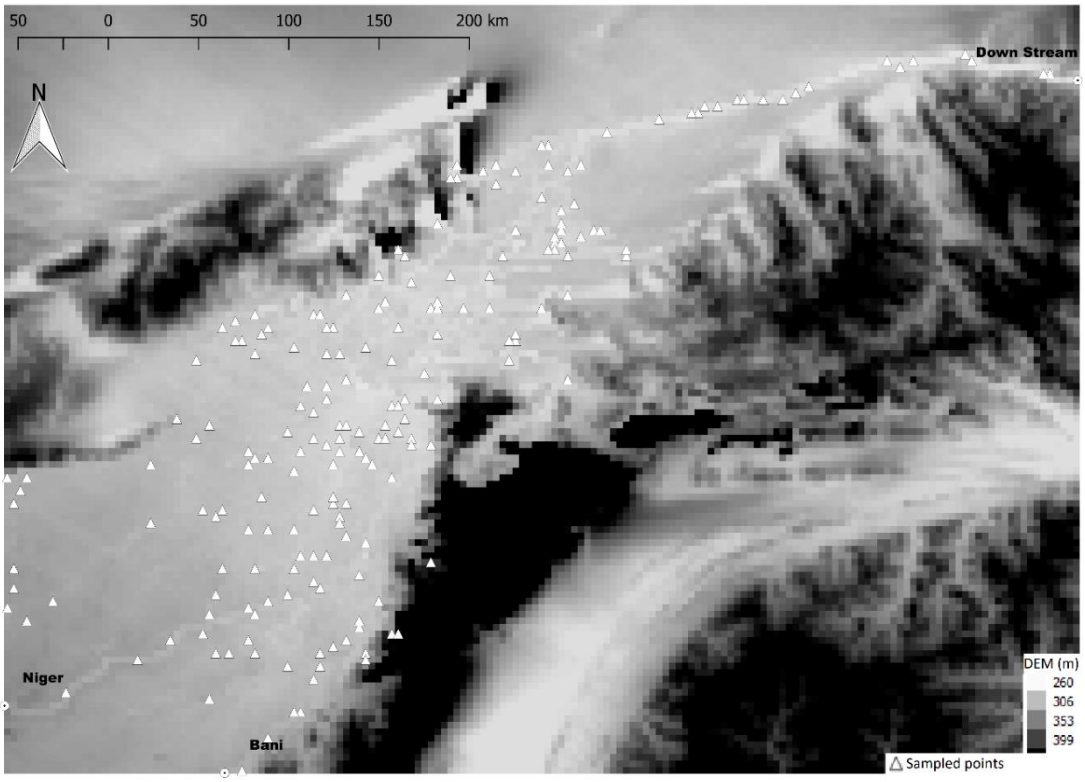


Figure 5.4: DEM of the study site. The 200 sampled locations randomly selected from the river network are in the main floodplain. Water depths at sampled locations were used to train and validate the classification accuracy of the ANN models.

Step 2 *Selecting input data:* For each unit of the MLP-based model, the input matrix consisted of seven elements. This included: six flow variables with lags ($Q_{Niger,Bani\ t-l}$, where, $l = 0,1,2$), and the corresponding time. It should be noted that the number of input variables was different than that of the SVR approach. This is mainly because when using the same number of input variables, i.e. 3, the MLP-based models did not reach to an acceptable classification accuracy (observed during a quick query test), and hence lagged flows were also used as the input variables.

Step 3 *Binarizing target data:* Water depth values at the sampled locations were extracted from the LISFLOOD-FP output raster files. Wet pixels (values > 0) were encoded as 1 and dry pixels as 0 to create a binary map.

Step 4 *Constructing the classifier:* MLPs are a simple stack of fully connected layers, i.e. an input, hidden, and output. The models were developed using the *'Keras'* data framework (Chollet and others, 2015) available in *'Python'* programming language. The model architecture of MLP was comprised of 8, 6, and 1 unit in the input, hidden, and output layer, respectively. In the first and intermediate layer, the 'Rectified Liner Unit' (ReLU) activation function was applied, whereas, in the final layer, a 'sigmoid' activation function was used. The MLP units were compiled with 'ADAM' (an advanced optimization algorithm) and a 'binary_crossentropy' loss function.

Step 5 *Training the classifier:* For training MLP classifier, two subsets of training and testing data were created from the total length of time-series. In the first subset, the classifiers were trained using data from 2003 - 2007 and 2009. 2008 dataset was used to test the model outcomes. In the second case, 2003 to 2008 datasets were used to train the classifier, and 2009 dataset was used for testing purpose. The classification performance of the classifiers was initially tested for a small number of pixels and then applied to the 200 sampled locations.

Step 6 *Generating outputs:* The outputs from the classifiers were produced in a probabilistic form. To binarize these outputs, a standard threshold value of 0.5 was used to classify cells as wet or dry. Any cells with a value less than the threshold value were classified as 0 (dry), and over 1 were classified as the wet cell.

Similar to the SVR training and predicting approach, the MLP-based models were trained and applied to generate outputs iteratively. The *Python* code for steps 2 to 6 is given below:

```
# Import libraries and models

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from sklearn.preprocessing import StandardScaler

from keras.models import Sequential
```

```
from keras.layers import Dense

# import training data

data = pd.read_csv("C:/Users/taaj_/Niger/NNET/inputSet8.csv")

# Specify the training data

X=data.ix[:,0:7]

#import training target data

target= pd.read_csv("C:/Users/taaj_/Niger/NNET/TrainTargetSet8.csv")

dt= target.ix[:,1:]

#Binarize target data

dt[dt !=0] =1

#import validation data

test_inp = pd.read_csv("C:/Users/taaj_/Niger/NNET/test_inp_set8.csv")

test_target                                                    =
pd.read_csv("C:/Users/taaj_/Niger/NNET/test_target_set8.csv")

dtar= test_target.ix[:,1:]

#Binarize validation target data

dtar[dtar !=0] =1

append_y= []

for column in dt:

# Specify the target labels and flatten the array

    y=np.ravel(dt[column])

    X_train = X

    y_train = y

    X_test = test_inp.ix[:,0:7]

    y_test = np.ravel(dtar[column])

# Define the scaler

    scaler = StandardScaler().fit(X_train)

# Scale the train data

    X_train = scaler.transform(X_train)

# Scale the test data

    X_test = scaler.transform(X_test)
```

```
# Initialize the constructor
    model = Sequential()
# Add an input layer
    model.add(Dense(8, activation='relu', input_shape=(7,)))
# Add one hidden layer
    model.add(Dense(6, activation='relu'))
# Add an output layer
    model.add(Dense(1, activation='sigmoid'))
# Model output shape
    model.output_shape
# Model configure
    model.get_config()
# List all weight tensors
    model.get_weights()
    model.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])
    model.fit(X_train, y_train, epochs=20, batch_size=1, verbose=1)
# Prediction
    y_pred = model.predict(X_test)
#Evaluate Model, for a quick performance check
    score = model.evaluate(X_test, y_test,verbose=1)
    print(score)
    pred=pd.DataFrame(y_pred, columns=[column])
#Binarize the probabilities
    pred[pred >=0.5] =1
    pred[pred < 0.5] =0
    append_y.append(pred)
# Save as csv
append_y=pd.concat(append_y, axis=1)
append_y.to_csv('pred_Y.csv', sep=',')
```

Step 7 *Accuracy estimation:* After generating outputs from the classifiers, classification accuracy was quantified for 200 test locations using **Eq.** 5.1, but also adding 'A' to the numerator and denominator. That is, cells classified as 'dry' by both LISFLOOD-FP and MLP-based models were also taken into account for calculating the accuracy. For the test case of SVR-based model accuracy estimation, 'A' was ignored because most of the cells in the domain remained dry for the entire year. These cells would be classified correctly by the model and would add bias towards the total accuracy. In contrast, in the present case only 200 samples were considered, and all of these locations experienced flooding at some point during the wet season. Therefore, the 'A' factor was added to the **Eq.** 5.1 for MLP accuracy assessment.

Step 8 *Spatial interpolation:* Classified values for the selected sampled points (step 6) were then spatially interpolated to produce probabilistic flood inundation maps for the entire domain. The so-called regression kriging (RK), spatial interpolation technique that uses surface elevation as auxiliary information to predict the values for the unsampled locations was applied (Hengl et al., 2007).

5.5 Results and analysis

5.5.1 SVR-based model results

The SVR-based models were first configured at four arbitrarily selected spatially varying locations to predict water depths. Four models with identical parameter settings were developed for these locations, say A, B, C and D. Parameters were first optimised for location A, and then the same parameter settings were applied for locations B, C and D. This initial task was conducted to test the possibility of applying identical model parameters for all the models within the Inland Delta. Figure 5.5 shows the comparison of the SVR-based model predicted and LISFLOOD-FP simulated normalized water depth time-series at these four locations. It is evident from the plots that the SVR-based models, trained using the identical parameters across the four locations, appear to provide comparative results, and thus can be used to classify wet/dry pixels during the high flow periods with acceptable accuracy. Further, it was concluded that the same parameter settings could be applied across all the cells within the Inland Delta. However, it can be noticed that prior to the start of the flooding season the SVR-based models appeared to predict flood water arrival time a bit ahead of the LISFLOOD-FP model (Location B, C and D). This could mean that, although using identical models for the entire delta will

immensely simplify the modelling procedure, this could result in the pre-arrival estimation of the wet season and the flooded area.

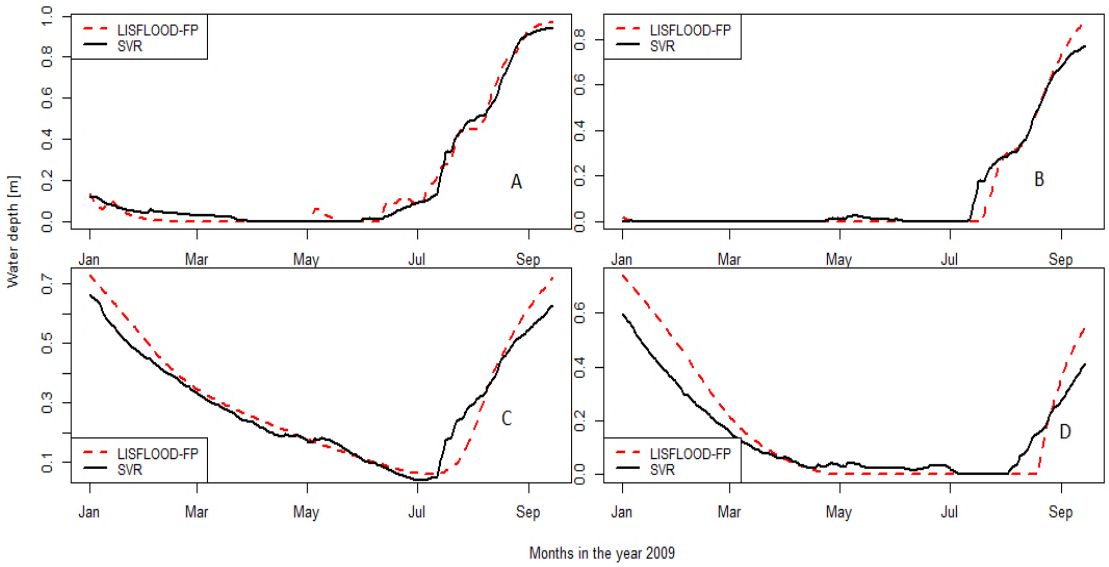


Figure 5.5: Prediction of water depth at four locations on the floodplain.

For the quantitative analysis, the performance of the SVR-based models was evaluated based on the following goodness of fit metrics: RMSE, R-squared, and NSE. Statistics, such as the RMSE and R-squared, usually weigh more towards the high magnitude of flow values (because of the squared difference between observed and predicted flows) (Kourgialas et al., 2015). It should be noted that the RMSE values equal to 0 represent a perfect fit, while for the R-squared and NSE the optimal value for a perfect fit is 1. Table 5.3 presents the resulting RMSE, R-squared, and NSE values at the test locations. Higher NSE, R-squared, and lower RMSE values indicate that the SVR-based model with the same model parameters can be applied to the whole study domain with acceptable accuracy.

Location	RMSE	R-squared	NSE
A	0.031	0.989	0.989
B	0.036	0.979	0.976
C	0.049	0.962	0.941
D	0.079	0.969	0.875

Table 5.3: Error statistics of the SVR prediction with respect to the LISFLOOD-FP simulation at test locations.

Given the nonlinearity of river hydraulics and the results obtained from the initial test of model performance with four randomly selected cells, it can be deduced that there is a potential for testing the applicability of the SVR approach to the entire floodplain. Thus, the configured model was applied to every pixel of the entire study domain and trained using the training dataset. Once trained, the test data generated a vector of time-series at daily time steps for each pixel. Interestingly, the dichotomous maps produced from the test set were highly accurate during the wet season when compared with the LISFLOOD-FP simulated inundation extent maps. Figure 5.6 and 5.7 demonstrate the flood extent maps generated using the LISFLOOD-FP and SVR approaches on 15th October 2008. It can be seen that the predicted flood extent map is identical to the LISFLOOD-FP extent on the selected day (peak flood season). However, as mentioned before, SVR-based models can be unstable during the low flow and transition periods. Figure 5.9 shows that the SVR-based models were overestimating the water depth values, resulting in an extended inundated area when compared with the LISFLOOD-FP simulated flood extent on 13th July 2008 (**Fig.** 5.8).

These dichotomous maps consist of a grid of '0's and '1's, where '0' represents the dry pixels and '1' represents the inundated pixels. The performance of the predictive models in terms of the fit between the LISFLOOD-FP and SVR inundation extent were then further tested for ten consecutive days during the dry and wet season using **Eq.** 5.1.

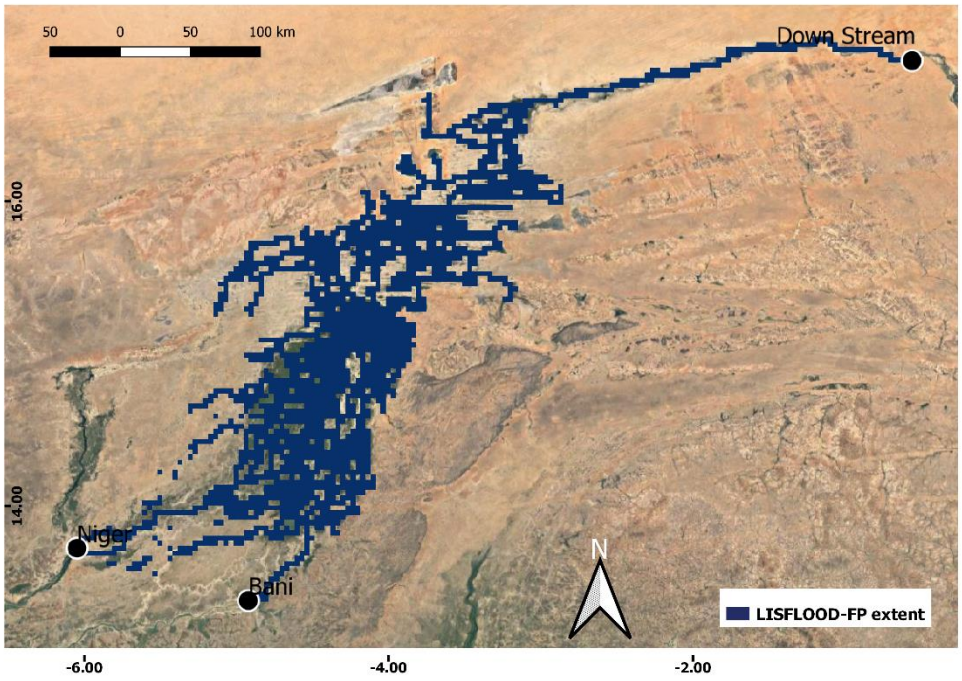


Figure 5.6: LISFLOOD-FP simulated flood extent map on 15th October 2008 [during the flood season]. Map data ©2020 Google.

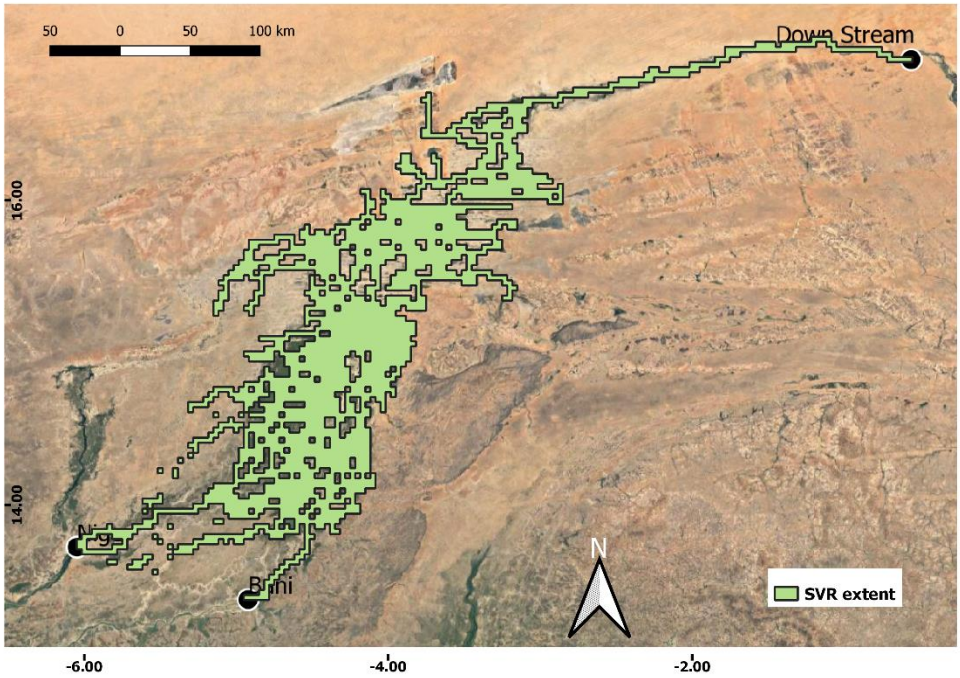


Figure 5.7: SVR simulated flood extent map on 15th October 2008 [during the flood season]. Map data ©2020 Google.

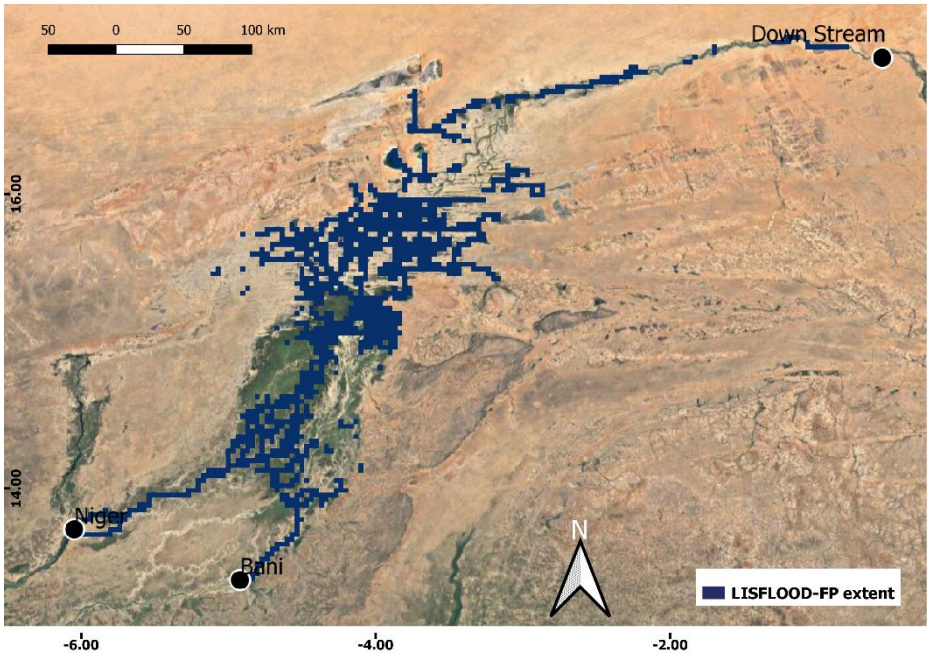


Figure 5.8: LISFLOOD-FP simulated flood extent map on 13th July 2008 [starting of the flood season]. Map data ©2020 Google.

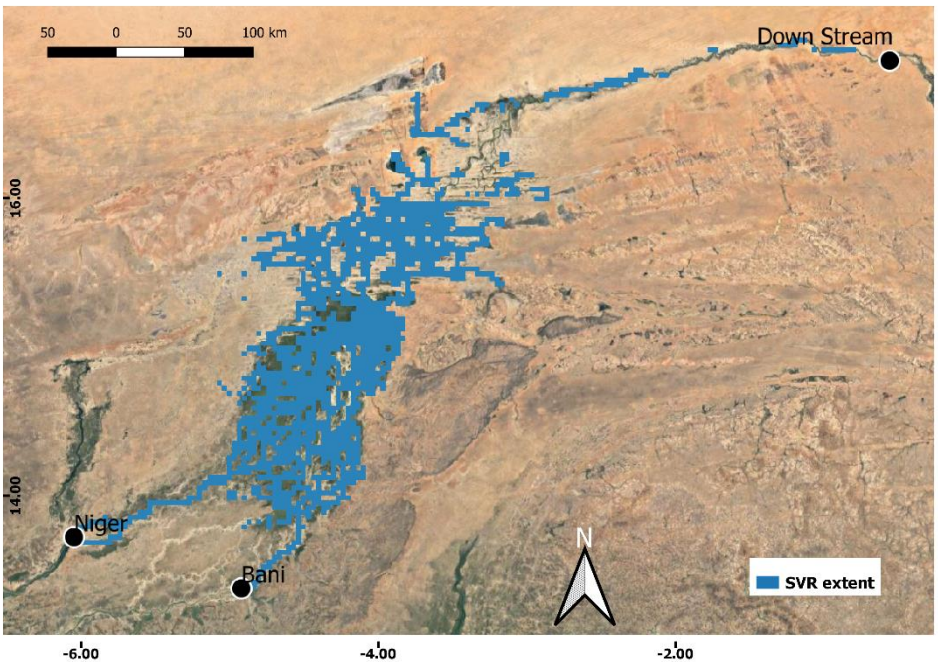


Figure 5.9: SVR simulated flood extent map on 13th July 2008 [starting of the flood season]. Map data ©2020 Google.

Table 5.4 presents the values for fit, *F*, where the value of *F* goes from 0% for a model with no overlap between LISFLOOD-FP and SVR simulated extent maps to 100% for a model with perfect overlap.

Date	Dry season-*F* (%)	Date	Wet season-*F* (%)
13/07/2008	72.11	14/10/2008	100
14/07/2008	71.92	15/10/2008	100
15/07/2008	72.33	16/10/2008	100
16/07/2008	72.20	17/10/2008	100
17/07/2008	71.22	18/10/2008	100
18/07/2008	70.72	19/10/2008	100
19/07/2008	70.43	20/10/2008	100
20/07/2008	70.75	21/10/2008	100
21/07/2008	71.24	22/10/2008	100
22/07/2008	71.42	23/10/2008	100
Average	71.43		100

Table 5.4: Evaluation of fits between SVR and LISFLOOD-FP inundation extent.

Yielded low *F* scores during the dry season (specifically, close to the end of the dry season) suggest that the SVR-based models did not match perfectly to the outputs from LISFLOOD-FP. This was because the SVR-based models were possibly overestimating the inundated area. However, these results were expected, considering the initial investigation of using identical SVR-based models for the entire domain, from which it could be inferred that such an approach could possibly overestimate wet cells before the arrival of the wet season.

5.5.2 MLP-based model results

This section presents the results obtained from the MLP based flood extent mapping method. The MLP predicted values were in the form of probabilities. To validate the model, flooded pixels were set as '1' if they were estimated to have a probability value over 0.5, and pixels with values less than 0.5 were set as '0'. The accuracy of the probabilistic models was evaluated by comparing the model against the LISFLOOD-FP simulated values (for comparison these values were also converted into '0's and '1's), at each of the sampled pixels, i.e. 200, as mentioned in section 5.6.3 Step 7. Table 5.5 shows the accuracy of the MLP based classification method and the SVR-based method at these locations.

Date	MLP- *F* (%)	SVR- *F*(%)	Date	MLP- *F*(%)	SVR- *F*(%)
13/07/2008	84	80	14/10/2008	98.5	100
14/07/2008	83	79	15/10/2008	97.5	100
15/07/2008	84.5	80	16/10/2008	97.5	100
16/07/2008	84	80	17/10/2008	98.5	100
17/07/2008	86.5	80.5	18/10/2008	98.5	100
18/07/2008	84	79.5	19/10/2008	99	100
19/07/2008	84.5	79.5	20/10/2008	99.5	100
20/07/2008	84.5	78.5	21/10/2008	100	100
21/07/2008	84	78	22/10/2008	100	100
22/07/2008	86.5	78.5	23/10/2008	100	100
Average	84.85	79.35		98.7	100

Table 5.5: Evaluation of fits between ANN, SVR, and LISFLOOD-FP at sampled locations.

The values from Table 5.4 and 5.5 should be considered as *approximated* accuracy of each method, as the *F* score will change for different periods during the year. Here an attempt was made to demonstrate the performance of proposed methods over ten consecutive days during the 2008 dry and wet seasons. Note that the MLP-based classifier appeared to perform slightly better than the SVR method during the end of the dry season, i.e. July when the Niger and the Bani rivers begin to rise. However, the SVR-based mapping resulted in better accuracy during September and October, when the natural flooding of the Inland Delta is well underway.

The probabilities estimated using the MLP method for the 200 specified locations were then further interpolated using the RK method to derive probabilistic flood extent for the entire domain. The underpinning idea was that the probability of a pixel being flooded is correlated with the DEM, and this feature can be exploited to generate probabilistic flood maps for the entire floodplain. Possibly due to the fact that the interpolation method did not account for any physical hydraulic processes occurring in the site, the pixels outside the main floodplain, i.e. at the top, top left, and bowed region in the bottom right, were estimated with higher probabilities of being flooded (Figure 5.10 and 5.11). Other factors affecting these regions could be lower surface elevation and the existence of permanent/semi-permanent lakes.

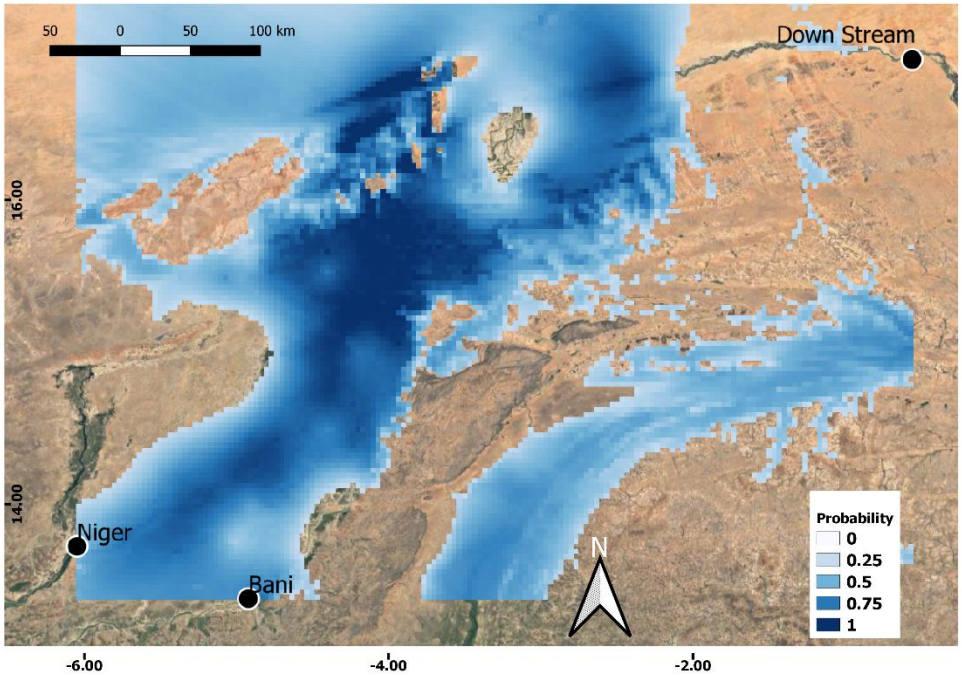


Figure 5.10: Probabilistic flood extent map for 13th July 2008 [starting of the flood season]. Map data ©2020 Google.

Highlighting the fact that the case study domain examined here was considerably large, and the grid resolution used was low. The low gird resolution facilitated the possibility of constructing SVR-based models for each grid in the region. The proposed approach will need further work to investigate the possibility of its application in cases when spatial resolution is required to be high, e.g. in urban areas. In such cases, there is good potential for the application of 'sampling' approach, as demonstrated here.

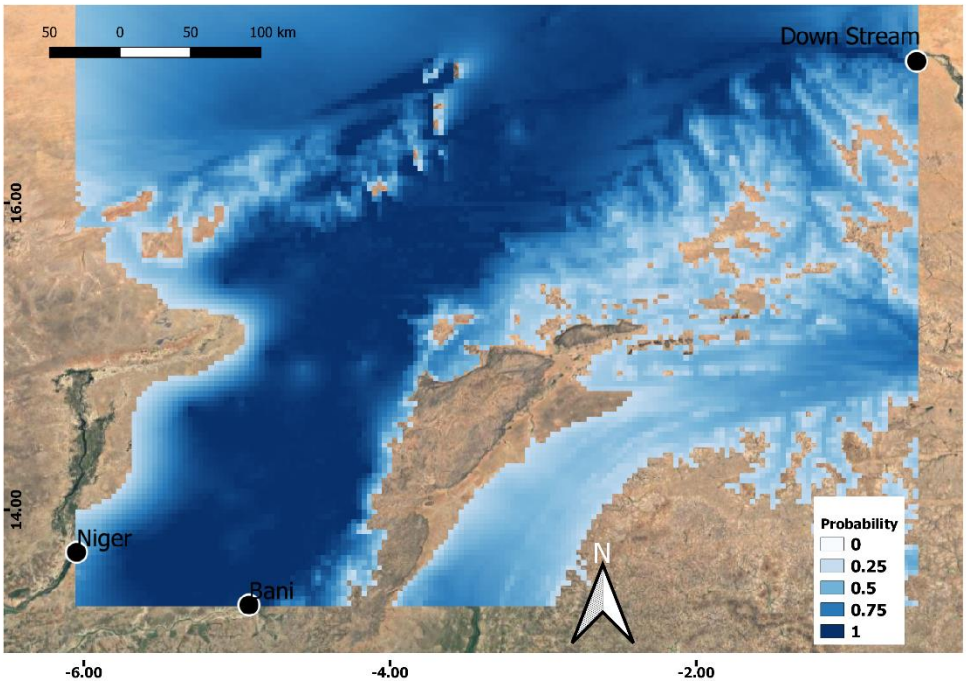


Figure 5.11: Probabilistic flood extent map on 15th October 2008 [during the flood season]. Map data ©2020 Google.

For relative comparison, the flood extent maps derived from different models for flood peak (15th October) are overlaid in Figure 5.12. Due to higher probability values found during the flooding season, across the domain, a higher threshold value (0.95) was applied to convert the MLP probability map into binary map. This value was chosen empirically. The figure shows that the MLP approach overestimated the inundated area. However, the overestimation mostly occurred in the low-lying northern and southern part of the domain along the channel. On the other hand, the SVR derived map fits accurately to the reference LISFLOOD-FP derived extent.

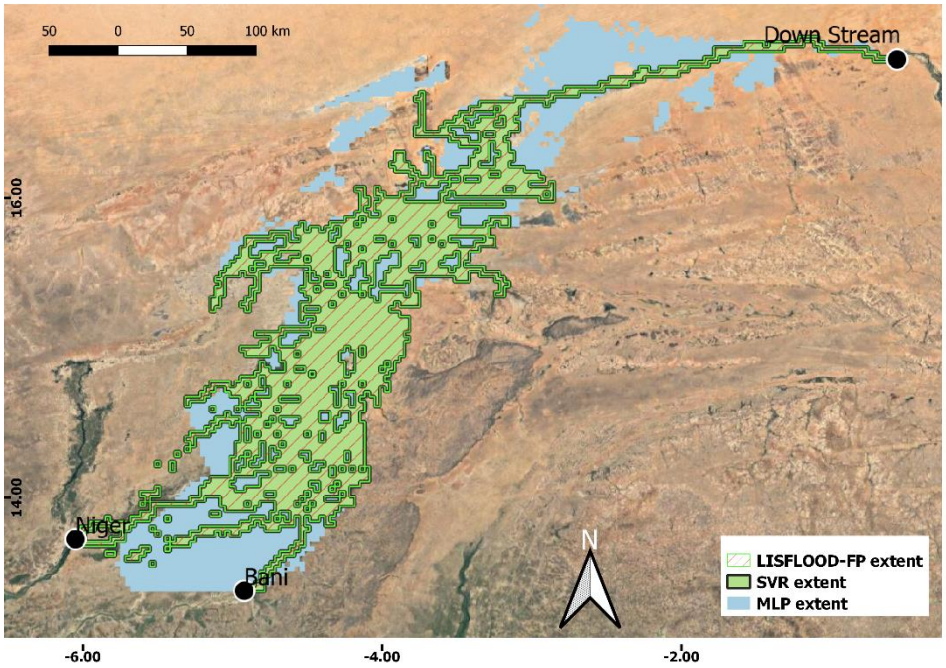


Figure 5.12: Comparison between MLP, SVR and LISFLOOD-FP derived flood extents. Map data ©2020 Google.

Finally, it should be noted that the accuracy of the proposed data-driven models is highly dependent on the data generated from the hydraulic model. Therefore, a fully calibrated 2-D model and a clear pre-definition of uncertainty is essential for training any ML-based model intended for real-time applications.

5.6 Conclusion

In this study, two data-driven modelling techniques were developed and applied to map inundation extent that can be generated in real-time. In real-world applications, hydrodynamic models need to be run thousands of times for conducting a thorough flood risk and model uncertainty analysis. The process is highly time consuming, and therefore, it is practically challenging to simulate these complex models during a real-time flood event. The chapter presented a novel research that investigates the possibility of developing ML based fast flood inundation extent mapping schemes. Two ML based flood extent mapping models, specifically an SVR-based approach and an MLP classifier were developed. These methods were trained using the outputs from LISFLOOD-FP model and were applied on the Niger case study site. Both the modelling schemes were shown to be numerically stable, and scalable with some inaccuracy observed during the dry period. These observed inaccuracies could be inherited from the outcomes of the

'subgrid channel solver' (a component of the LISFLOOD-FP model) which was used for training the proposed models. As identified by Neal et al. (2012): *"A major inaccuracy in the 'subgrid' model's simulation of the level was due to the use of a global channel parameterization, while at low flow the lack of local hydrology and assumption of rectangular channel geometry may become important"*. Nevertheless, the key intention of the present study was to investigate the model performance during the wet season, and therefore, the high error rate occurring during the dry period should not be considered as a major concern.

It is worth noting that a simple binary classification approach (i.e. MLP) is sufficient for classifying wet and dry cells. Thus, dynamically generating inundation extents. Additionally, regression-based method was also implemented in this study. A regression model could be used to estimate both water depths and binary maps. An initial test has shown the potential of using the SVR for depth prediction at four locations (**Sect.** 5.5.1). However, in this PhD work, derivation of the flood extent maps was considered. Therefore, only flood extent maps from both regression and classification-based methods were compared.

As the main focus of the chapter was to investigate the surface water extent, a detailed analysis of other hydrodynamic components, such as surface water elevation or velocity was not conducted. The next chapter will attempt to assess the possibility of further application of these novel data-driven models on geographically and hydrologically distinct catchments, and potentially with higher spatial resolution. This will be achieved by developing an integrated method that could enable end-users to monitor flood extent in real-time with an automated raster updating feature.

Chapter 6 **Inundation Modelling Using ML: Case Study 2**

6.1 Introduction

Due to rapid urbanisation and unprecedented climate change impacts the need for hydrodynamic modelling for urban flood management is greater than ever before (Miller and Hutchins, 2017). Chapter 5 showed that for regions where the main river channels are mostly driven by rainfall and obstruction in water flow through man-made infrastructures is minimal, a simple ML classifier/regressor can effectively map the flood extent. This is achieved by training a ML based model using the output from a 2D hydrodynamic model. Unlike natural catchments, to conduct flood inundation modelling for urban catchments, it is important to incorporate relevant structural (bridges, culverts, flood defences, etc.) data in the modelling procedure, which makes flood inundation modelling in such areas rather complicated.

Therefore, one of the key questions arises, whether we can apply the ML based models in a modified floodplain/urban area to sequentially map the flood extents in real-time. To this question, a thorough evaluation of the applicability of different ML techniques for performing real-time flood extent mapping in a semi-urban area was conducted. To assess the performance of simple ML-based models trained on outputs from a 2D hydrodynamic model, the town of Upton-Upon-Severn in England was considered. The town has a history of flooding caused by overtopping of embankments in the past. High resolution topographical and hydrometeorological data are also available. Therefore, the area was deemed suitable as a test bed for this case study.

The hydrodynamic model used here to simulate 2D flood inundation was Flood Modeller (FM) free version. The free version of the software provides 250 1D nodes and 100,000 2D cells for all three solvers (ADI, TVD and FAST), which was sufficient to demonstrate the potentials of novel models developed in this study.

In this chapter, firstly, a brief description of the case study area and available hydrometeorological datasets is provided in Section 6.2 and 6.3. Step-by-step procedure for methodology development is outlined in Section 6.4. Section 6.5 describes the process of generating synthetic hydrographs and the inundation modelling procedure using FM is

presented in Section 6.6. The input attributes and preparation of training and validation datasets for ML-based modelling is explained in Section 6.7. Then the steps adopted to train the MLP classifiers for classifying the wet cells are extensively described in Section 6.8. The regression-based approach for mapping flood inundation extent is described in Section 6.9. Section 6.10 presents how interpolated flood extent maps were generated from point measurements. The evaluation criteria for assessing modelling outcomes are given in Section 6.11. The results and analysis section (**Sect.** 6.12) document the outputs from all ML-based models developed and applied to compare the results against the FM generated flood maps. Finally, concluding remarks highlighting the key findings and challenges are presented along with discussions on further developments of the proposed rapid flood modelling scheme for real-time flood extent mapping within acceptable accuracy.

6.2 Study area

The town of Upton-upon-Severn is situated on the west bank of the River Severn and is located in the Malvern Hills District of Worcestershire, England. According to the Environment Agency (EA, 2015), since 1970, the area has seen over 70 flood events and has often been dubbed as the most flooded town in the UK.

Some of the most recent flood events in Upton-Upon-Severn are presented in Table 6.1:

Index	**Month-Year**	**Index**	**Month-Year**	**Index**	**Month-Year**	**Index**	**Month-Year**
1	January 1998	**6**	December 1999	**11**	October 2004	**16**	January 2008
2	October 1998	**7**	Oct/Nov 2000	**12**	January 2007	**17**	September 2008
3	January 1999	**8**	December 2000	**13**	March 2007	**18**	November 2008
4	September 1999	**9**	February 2002	**14**	June 2007	**19**	December 2012
5	October 1999	**10**	January 2003	**15**	July 2007	**20**	February 2014

Table 6.1 Recent Upton-Upon-Severn flood events.

Amongst all the flooding events, the July 2007 event is considered as one of the “worst in the living memory”. During the event, the entire town was lashed by severe weather,

heavy rain led to flash floods, and high river levels left parts of the area under ~1.8 meter of water. The town was virtually cut off from rest of the country for several days, and the Army brought in the food and water for those affected. After the devastating event, the EA and the local community worked together to develop a permanent flood defence scheme along the Waterfront area. Since then, this defence scheme is protecting the most at-risk properties located in two separate areas - 'New Street' and 'Waterside' (**Fig.** 6.1).

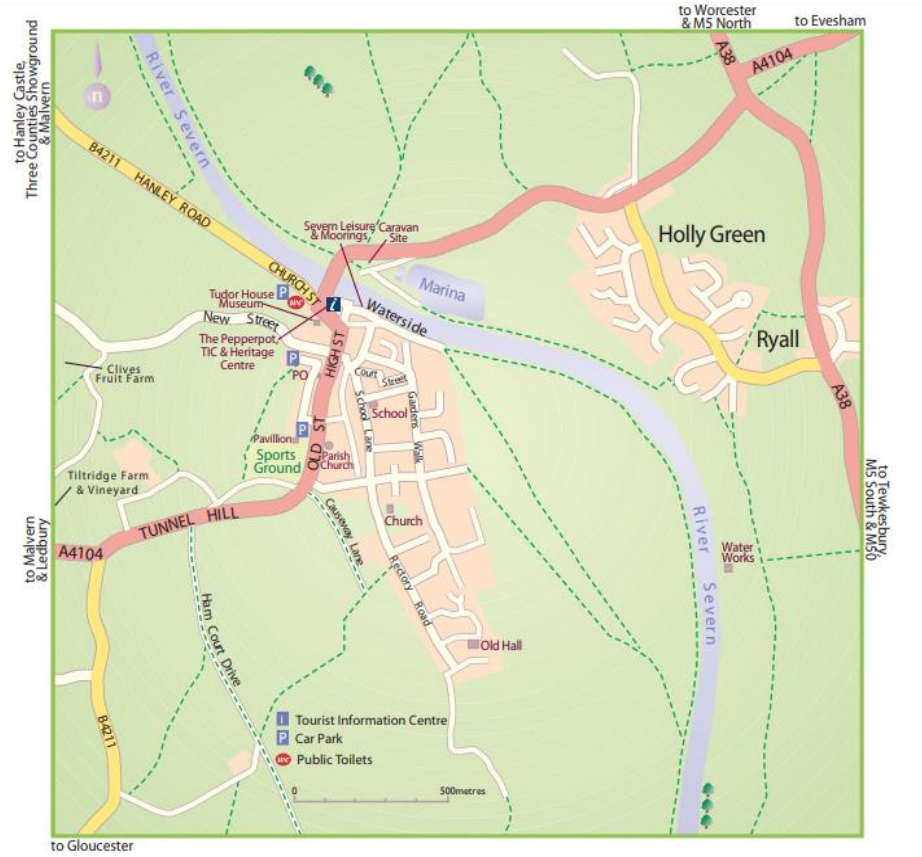


Figure 6.1: Upton-Upon-Severn and two flood risk areas (Waterside and New Street).

The Waterside scheme, a permanent flood wall with glass panels (**Fig.** 6.2), defends properties against floods of 1 in 150 return period in any given year. For New Street, as part of the permanent flood defence scheme, an earth embankment was built, a new flood wall was constructed, and a flood gate across New Street was installed.

Figure 6.2: Upton-Upon-Severn flood defence (Copy rights Environment Agency).

6.3 Hydrometeorological data

The hydrometric data used in this study consisted of hourly flow at upstream, near Severn Stoke (location coordinates: X: 384733, Y: 249922) for October - November 2000 flood event. The time-series of observed flow starts on 28/10/2000 at 13:45 (as time step 0) and stops at 15:45 on 17th November 2000 (time step 485). The hydrograph of the data has a peak (899.19 m^3/s) recorded on 03/11/2000 at time 04:45 and a second peak (832.49 m^3/s) recorded on 08/11/2000 at 20:45 (**Fig.** 6.3).

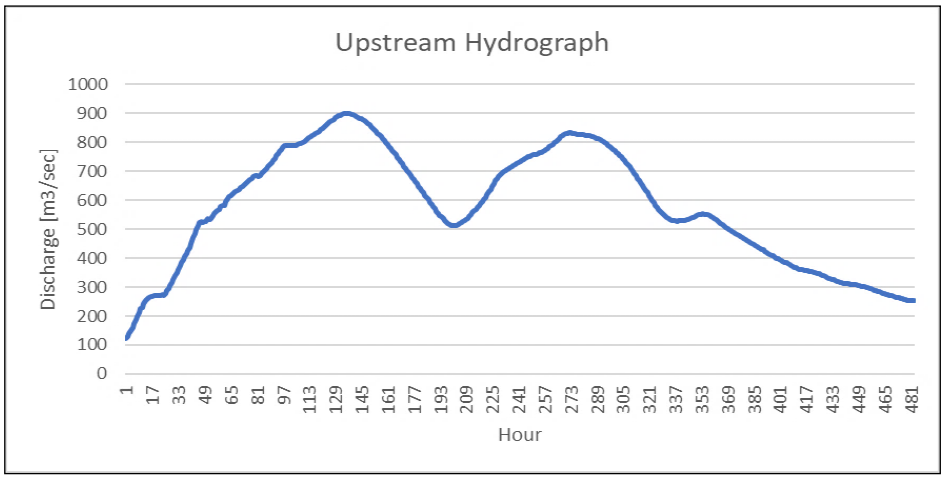


Figure 6.3: Inflow hydrograph at the upstream.

6.4 Methodology

The underpinning idea used here follows a similar analogy to the one presented in Chapter 5. However, the spatial and temporal resolution in the current case study was significantly higher than in the case of Niger. The complete workflow of the proposed methodology was divided into six distinct steps: a) creating synthetic inflow hydrographs of different durations and magnitudes, b) simulating the 2D model using the hydrographs, c) creating training and validation data sets, d) training and validating ML-based models, e) analysing the results, and f) spatial interpolation (**Fig.** 6.4). These steps are explained in detail in the following sections.

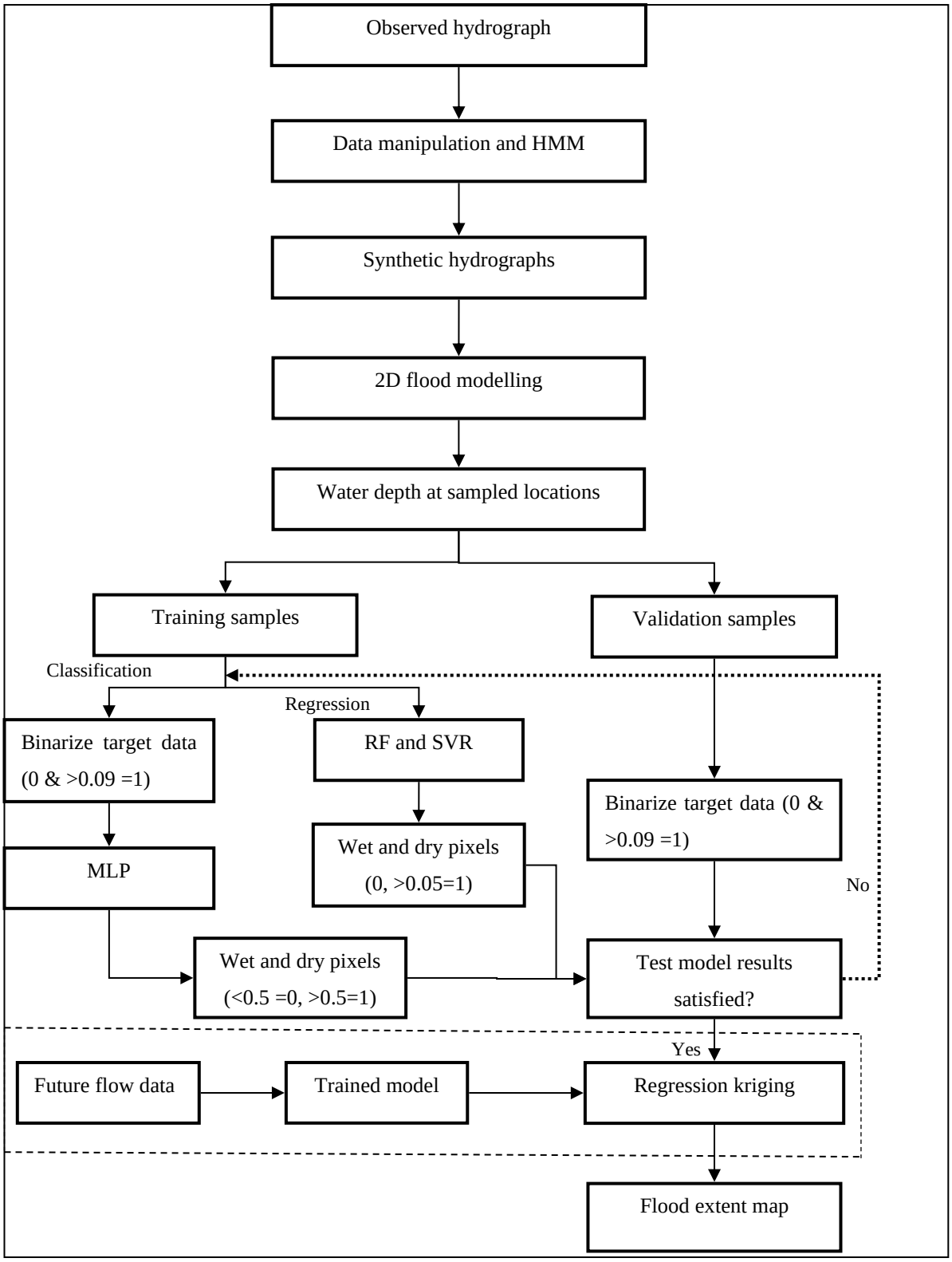


Figure 6.4: Flow diagram of the methodology.

6.5 Synthetic flow generation

To train an ML algorithm sufficient number of input-output instances is required. The number of input-output samples should encompass different hydrological (inflow hydrographs of different magnitudes) conditions that will decrease the likelihood of the model being over- or underfitted. In an ideal scenario one would like to simulate inundation extents using multiple inflow hydrographs which represent different flood return periods. Thus, a hydraulic model output from a single hydrograph is not enough to train a selected form/type of ML-based model. For the Niger study case, seven years of flow data is available, which made it possible to split the hydrograph into seven sets. However, for certain cases when the length of a single observed time-series is limited, it is essential to generate a few additional synthetic time-series to allow sufficient number of input-output instances for a robust training of the ML-based models. In the relevant work done by Liu and Pender (2015) and Liu et al. (2009), multiple inflow hydrographs are generated by changing the magnitude of the peak flow. However, there is a fundamental problem with this type of approach, as it only allows for magnitude of the peak flows to vary in distinct hydrographs whereas the underlying flow pattern remains the same. An alternatively approach to this problem could be to develop or utilise an existing stochastic model for generating synthetic hydrographs. A range of stochastic modelling approaches are available, such as one developed based on Hidden Markov Model (HMM) developed by Patidar et al. (2018) and Pender et al. (2015). See **Sect.** 6.5.2 for a description on HMM model and simulation process. Three synthetic hydrographs were generated using HMM. Further, two hydrographs were generated by subsetting the observed flow. Out of these five synthetic flow hydrographs, four were used to train the models, and one was used for testing.

6.5.1 Subsetting observed data

The data available for this case study is a single hourly time-series. The observed inflow hydrograph was thoroughly analysed and revealed two distinctive peaks: 899.19 m^3/sec (high peak) and 832.48 m^3/sec (low peak) occurring on 03/11/2000 at 04:45 and 08/11/2000 at 20:45, respectively. The data was subsetted to ensure sufficient amount/length of data is available for robust training and validations of the ML-based models. Thus, the observed hydrograph was processed to create two subsets, where each contained one of these flood peaks.

6.5.2 Hidden Markov Model simulated data

Possibly one of the simplest approaches for generating synthetic hydrographs is by altering the observed flow series using the following formula (see **Sect.** 6.7):

$$Q_n = Q_{observed} \times \frac{Peak_{max}}{Q_{max}} \quad (6.1)$$

In the **Eq.** 6.1, $Peak_{max}$ is the user-specified peak flow, $Q_{observed}$ is the observed flow series, and Q_{max} is the observed peak. This method can be applied to generate the desired number of hydrographs with different flood peaks. Though, synthetic hydrographs generated using such a technique do not capture the variability in flow patterns/shapes during the occurrence of flood events. In real-world cases, flow patterns of two statistically significant flood events generally differ in magnitude and duration. The above formula can only capture differences in flow magnitudes, and thus considered insufficient. To limit some of these issues, the desired number of synthetic inflow hydrographs with different shapes, in addition to the two subsetted series, were generated using an HMM based approach proposed by Pender et al. (2015).

The HMM, a generative probabilistic model, was developed in *R* statistical software (R Core Team, 2013) using the 'HMM' package. Three synthetic flow series were generated at an hourly resolution, displaying varied flow scenarios but owns similar statistical properties of the original empirical flow profiles. The steps for generating synthetic flood hydrographs of different shapes are similar to those stated in Patidar et al. (2018), Pender et al. (2015) and Jenkins et al. (2014), however, are also briefly described below:

a. *Define observed states:* the observed states are defined using a percentile distribution of the observed flow series. The subsetted flow series from **Sect.** 6.5.1 were used to define thirteen distinct states (Table 6.2). The time-series values were rounded to nearest integers.

State	Limits	State	Limits
A	Min and 10th percentile	H	90th Percentile and 95th Percentile
B	10th Percentile and 30th Percentile	I	95th and 96th percentile
C	30th and 50th percentile	J	96th Percentile and 97th Percentile
D	50th Percentile and 70th Percentile	K	97th and 98th percentile
E	70th and 80th percentile	L	98th Percentile and 99th Percentile
F	80th Percentile and 85th Percentile	M	99th and 100th percentile
G	85th and 90th percentile		

Table 6.2: Observed states.

b. *Define hidden states:* hidden states account for all the discrete values within the range of each thirteen distinct observed states (all decimal values were rounded). So, for example, state B corresponds to values between 270 m^3/sec to 356 m^3/sec, then the set of hidden states corresponding to state B will be all values between them.

c. *Define state probability transition matrix:* this is an S × S (13 × 13 here) matrix that defines the probability of transition between different states. Values (M) in rows and columns correspond to the probability of transition of flow value from one state to the other. The probability transition matrix, $M_{S\times S}$ can be described as:

$$\begin{bmatrix} M_{AA} & M_{AB} & \dots & \dots & M_{AM} \\ M_{BA} & M_{BB} & \dots & \dots & M_{BM} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ M_{MA} & M_{MB} & \dots & \dots & M_{MM} \end{bmatrix}$$

d. *Define emission probability matrix:* probability matrix of hidden states corresponding to each of the observed states, i.e. an S × N matrix. In other words, matrix containing the emission probabilities of the states. For example, one of the subsetted series from Section 6.5.1 had the minimum value 122 m^3/sec and maximum value 832 m^3/sec. Therefore, the emission matrix will have 13 distinct states (S) and 711 hidden states (N) corresponding to all 13 states, and can be expressed as:

$$\begin{bmatrix} m_{A1} & m_{A2} & \dots & \dots & m_{A711} \\ m_{B1} & m_{B2} & \dots & \dots & m_{B711} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ m_{M1} & m_{M2} & \dots & \dots & m_{M711} \end{bmatrix}$$

e. *Model initialization:* once all the input vectors and matrices are formulated, the HMM model is initialized by calling the *'initHMM()'* function.

f. *Generate synthetic sequences:* Synthetic sequences are then generated for a user specified length using the *'simHMM()'* function.

The sequences generated are the vectors that describe the stochastic process (flow series contain similar statistical characteristics), but they do not resemble flood hydrographs (**Fig.** 6.5).

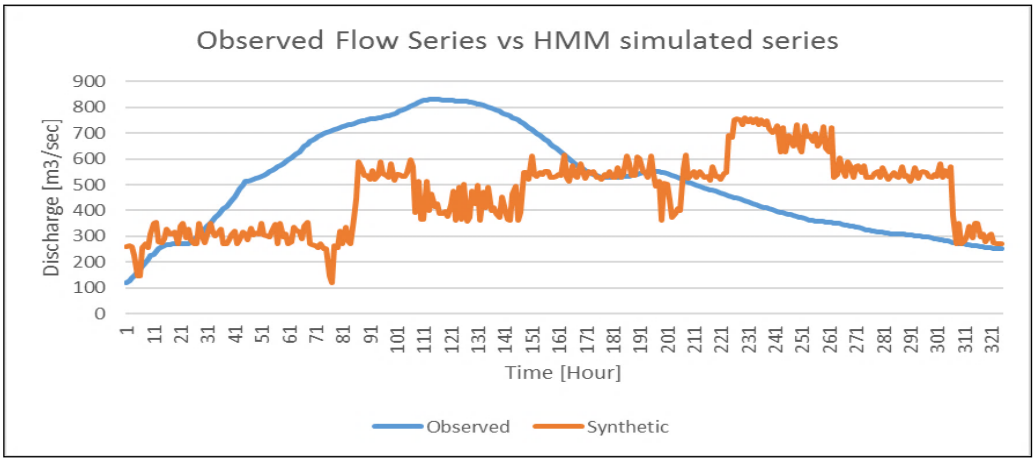


Figure 6.5: Observed flow and HMM generated synthetic series.

In order to create hydrographs with a peak, the values were sorted sequentially based on their states (first ascending order then descending order to make sure there is a peak). Following this procedure, three synthetic hydrographs with a flood peak were created. There were two issues with these synthetic hydrographs- 1) the synthetic hydrographs had peaks below the peak value of the observed hydrograph, and 2) there was not enough difference between the peaks. These issues could be solved by changing the magnitude of the synthetic hydrographs. The magnitude of the synthetic hydrographs was further altered using **Eq.** 6.1 to achieve a sizable difference between the peaks. These hydrographs with varied peaks and the two subsetted hydrographs can be seen in Figure 6.6.

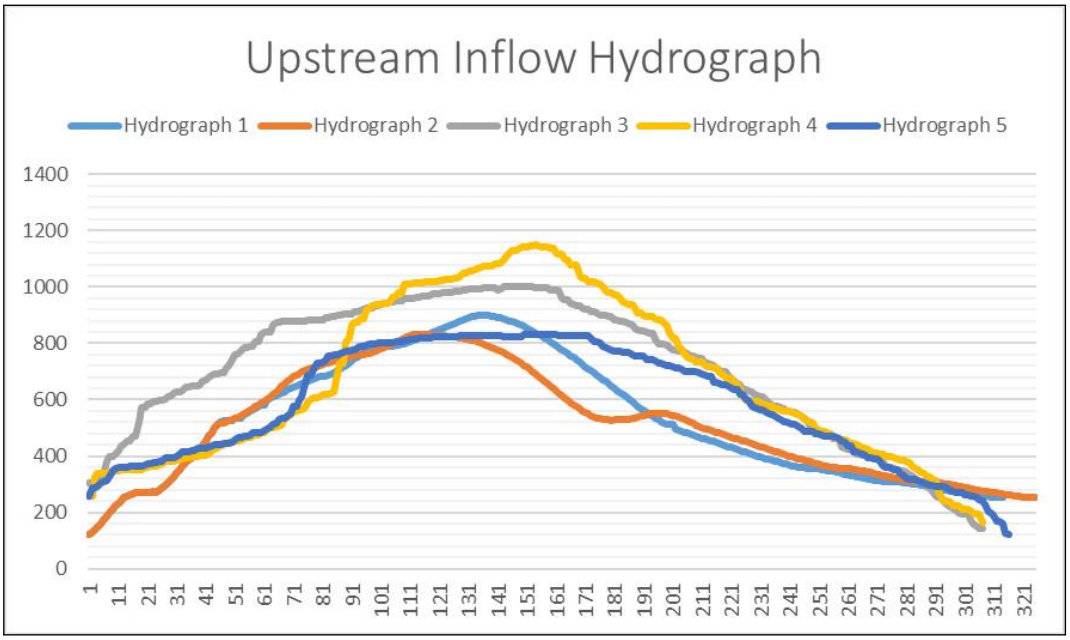


Figure 6.6: Five synthetic inflow hydrographs were created: Two by subsetting the original flow and another three by using HMM approach. Hydrographs 1 and 2 are the two subsetted hydrographs. Hydrographs 3, 4 and 5 are the HMM simulated hydrographs.

6.6 Flood Modeller simulation

The fundamental approach of flood modelling using the Flood Modeller (FM) software (also valid for any fully distributed flood inundation modelling system) comprises of four key steps:

I. Select the best modelling approach
II. Obtain data and build initial network
III. Calibrate and validate the model
IV. Production of results

The first step is perhaps the most important, i.e. selecting an appropriate approach for the particular problem to solve. After selecting the appropriate approach, i.e. 1D, 2D or 1D-2D linked, the next step is data collation and develop an initial model, which then follows through an extensive calibration process using observed data. Finally, model is validated with another dataset/event which is not used in the model calibration. The validated model is used for producing outputs, i.e. flood maps, hazard maps, etc.

In addition, several other factors and potential challenges are associated with the flood modelling process, for example: it is essential to have clearly defined objectives, which helps in selecting best modelling approach. Further, the availability of required datasets,

identification of flood characteristics, skills for using hydraulic modelling software, time and budget available to do the task are some other important constraints to be taken into account.

To facilitate flood inundation modelling using the FM software, following dataset along with several other information, as described in Table 6.3, are required:

Index	Data	Purpose
1	River cross sections (1D), full bathymetry (2D)	Cross-sections from upstream to downstream in a consistent distance are required. These can be surveyed or generated within the FM software from the DEM.
2	Structural dimensions	Bridges, weir, sluice, embankments
3	Topography (raster grids)	The DEM files dictate water flow in the floodplain.
4	Boundary conditions	Usually upstream inflows and downstream water levels.
5	Channel and floodplain roughness values	Roughness values for the channel and the floodplain. Standard sources to find references for roughness value for channels are lookup table or ground survey. Roughness value for floodplain is assigned depending on the land cover types.
6	Initial conditions	It is the starting condition of the hydraulic system. In general, it is assumed that the floodplain is dry and starts with an initial water level, and initial water level in the channel that is consistent with the start of the boundary conditions- e.g. the starting water level of the hydrograph. This water level is assigned to all the cross-section nodes through the model.
7	Calibration data	Observed gauged data in the channel or data from level gauges in the floodplain.
8	Runtime data	The start, end time and time step for the model simulation.
9	Validation data	Usually, a historical flood event data or flood extent maps from satellite images.
10	Miscellaneous	Linear features and factors that could affect flow routes

Table 6.3: Data required for flood modelling.

6.6.1 1D-2D linked model

There are two main types of flood modelling, namely, 1D or 2D modelling. 1D or one-dimensional modelling solves the 1D SWEs of flow in the channel. Solving these equations yield a single water level/flow at a set of points for the river system, e.g. at the cross-section nodes. In addition, 1D models also solve other point features that may occur in the river on the floodplain, and these could include weirs, bridges, sluices etc. 1D

flooding modelling is fast to run. It is good at representing in-channel water level and flows, and also efficient for modelling point features such as the bridges, weirs, sluices etc. However, 1D modelling has some disadvantages too. For example, calibration of 1D model requires detailed information on various parameters such as, full identification of major flow routes, no information provided on velocity distribution on the floodplain, simulation outcomes can be poor in urban areas (modelling flow routs through the urban areas along streets can be challenging). On the other hand, 2D or two-dimensional modelling solves the 2D SWEs and calculates water depth and 'depth averaged' velocity on a grid. 2D modelling does not require predefined flow routes and thus easy to set up and is more accurate for urban flow routing. However, due to increased complexity, 2D models are slow to run, poor in simulating bridges/weirs/sluices, and can need fine grid/mesh for simulating river channel. To address some of the limitations, a linked 1D and 2D model, combining the key features of 1D and 2D modelling, is widely applied (JACOBS, 2017). In a linked model, river channel uses the 1D solvers to model water levels at the cross-sections throughout the river system, and then apply 2D solvers to model the floodplain. FM software facilitates three basic approaches for forming the links between 1D-2D domains:

I. Level link
II. Flow link
III. Weir link

The level link is a process where the 1D model calculates the water levels in the river, which are then sent across to the 2D solver. The 2D solver, which has the information on water levels in the floodplain, then calculates how much flow goes between the two domains, i.e. either from floodplain to channel or from channel to floodplain (**Fig.** 6.7).

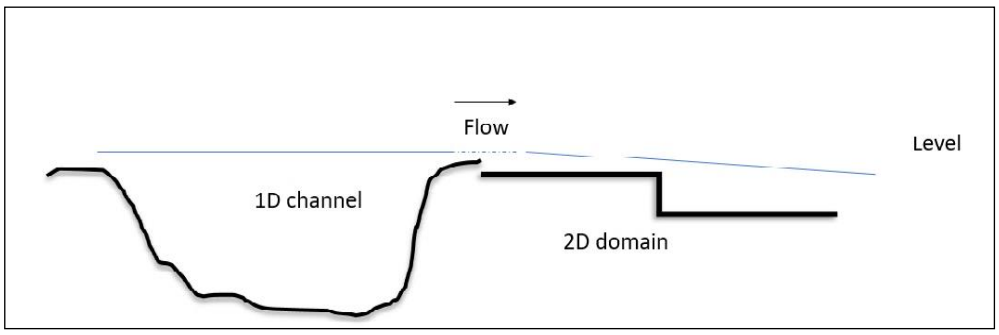


Figure 6.7: 1D-2D Level link.

The flow link approach is where the 1D model calculates how much flow is going to transfer between the 1D and 2D boundary. It calculates this using spilled units which could be representing flow over a flood defence or naturally high ground or embankments at the edge of the river. The calculated excessive flow is then modelled through the domain by means of the 2D solver (**Fig.** 6.8).

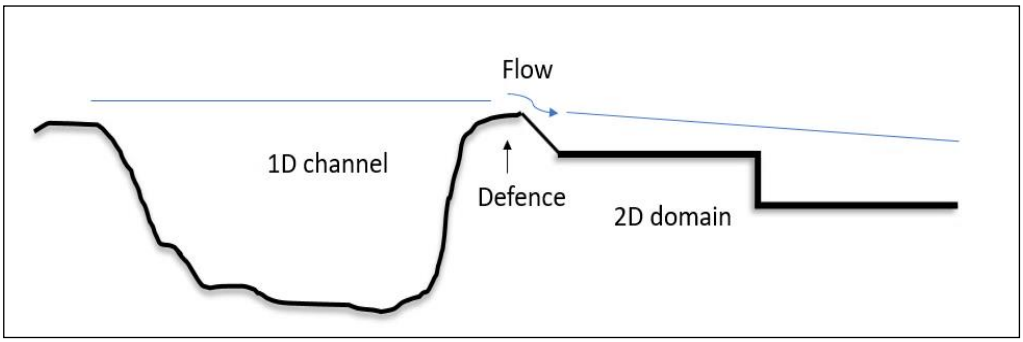


Figure 6.8: 1D-2D Flow link.

The weir link approach is similar to the flow link approach, using weir type equations. However, in this case, it is the 2D model that is applying these weir equations to work out how much flow is going between the domains. The 2D solver has elevations along the top of the crest of the embankments.

6.6.2 Upton-Upon-Severn model

An initial 1D network made available by the Institute for Infrastructure and Environment (IIE), Heriot-Watt University, was used along with a couple of floodplains modelled using “Reservoir” and “Spill” units. The Spill units are a complex and powerful tool that calculates flow over an irregular weir. See the FM tutorial (JACOBS, 2017) for more details on spill units, reservoir, and model building.

The flow link approach uses spills to calculate flow transfer, but the level link does not. Therefore, to keep the model simple, all reservoir and spill units are removed from the network. The network had 53 cross-section nodes, which were numbered between MC010 and MC058. The upstream and downstream boundary conditions were respectively connected to node MC010 and MC058. Also, the two bridges were connected to node MC034U (A4104) and node MC055U (A438). The simplified 1D network used in this study is shown in Figure 6.9.

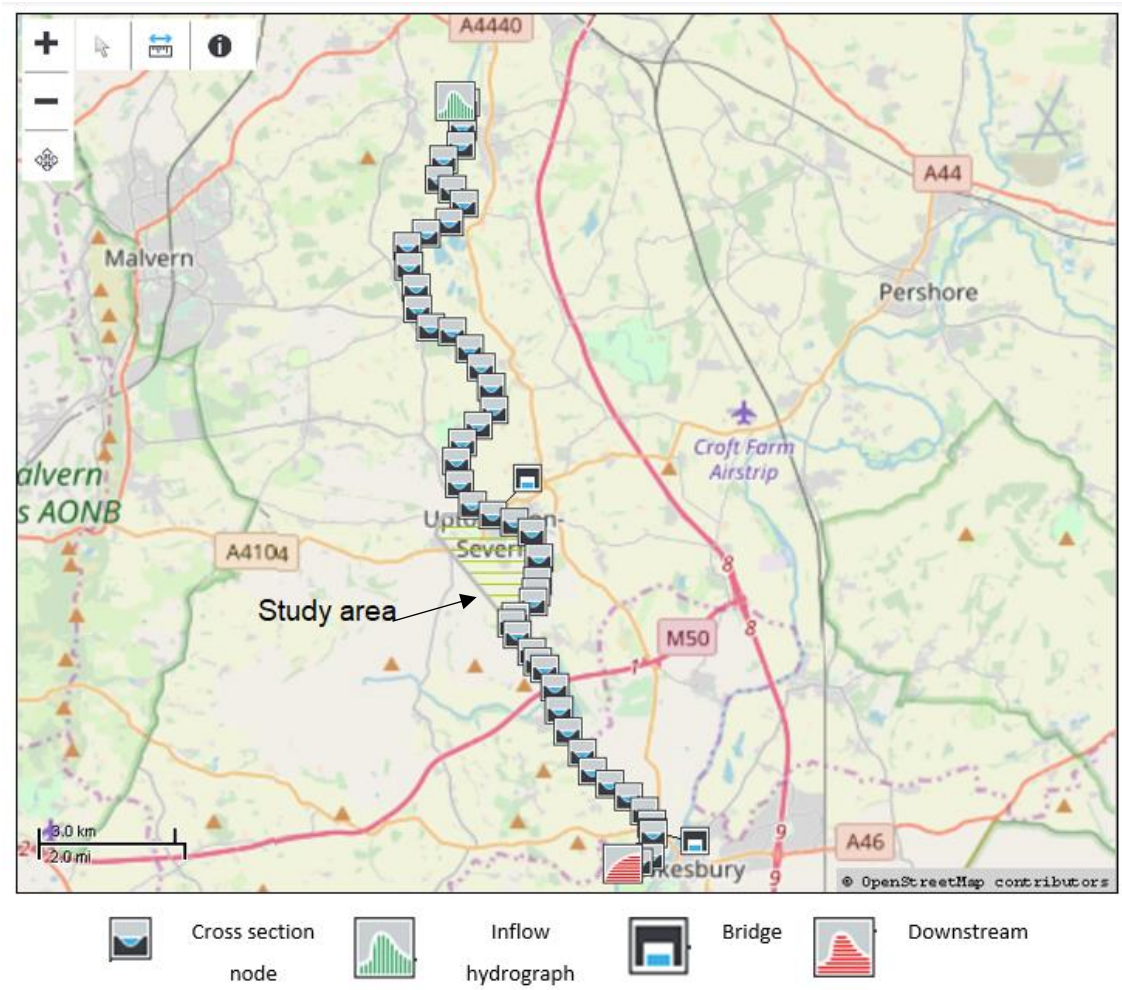


Figure 6.9: Initial 1D network, area of interest, and model output locations.

The simplified network was then used to develop the 1D-2D linked model for the case-study and used to simulate floodplain inundation. The process of simulating flood maps through the 1D-2D linked model for the case-study is described below:

6.6.2.1 Generate initial condition

Since the initial network with cross-sections is already available from the previous studies, therefore, developing a new 1D initial model was not required for this study. However, to generate the initial conditions/ initial water levels for the channel, a 1D steady flow model was created and simulated.

6.6.2.2 1D model run

Once the initial conditions were generated, the initial condition file was imported to the 'Files' tab within the 1D unsteady model simulation window. For the 1D unsteady model setup, the simulation start, end, time interval and time steps are critical. The start time

(0.00 hrs), timestep (10 sec) and interval (3600 sec) were kept constant for each of the hydrographs while the finish time varied depending on the duration of the hydrographs. Values used for the 1D unsteady run are displayed in Figure 6.10. Default values were used in the rest of the tabs. The output (water levels) of the simulations at the cross-sections were saved for each hour.

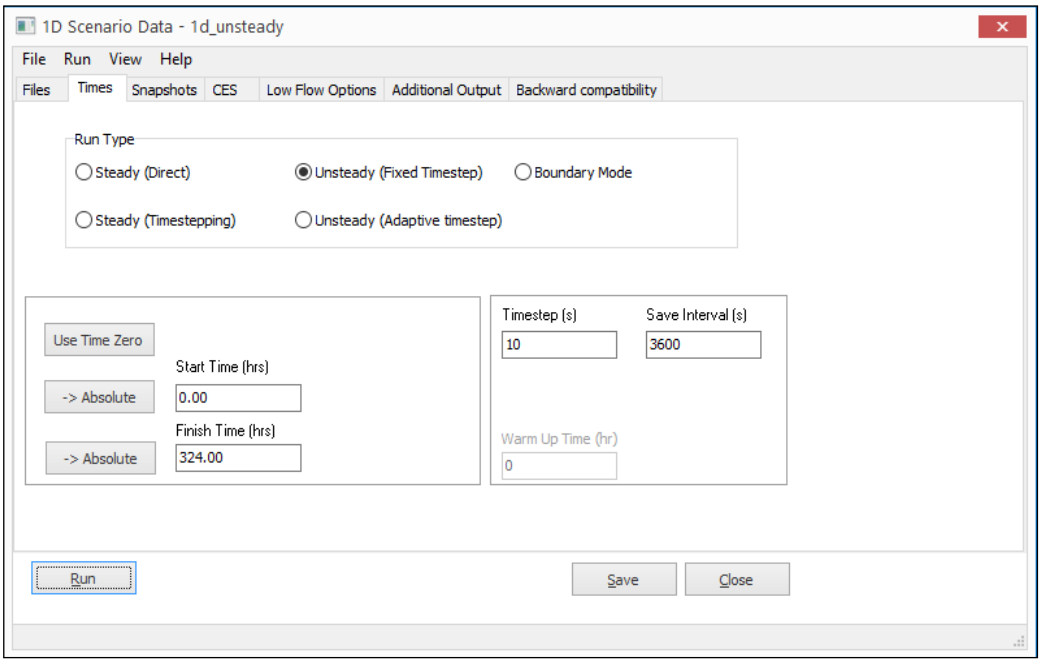


Figure 6.10: 1D unsteady simulation. Finish time used is different for each of the inflow hydrographs.

6.6.2.3 Create active area

To further reduce the complexity of the 2D model, an active area which is a sub-area within the computational grid was defined. An active area is defined in the form of a polygon shapefile, representing the floodplain of interest on the right bank of the river. The software performs calculations only within the active area, thus saving computational time. The active area within the Upton-Upon-Severn floodplain was created within the FM software platform (**Fig.** 6.11).

6.6.2.4 Create 1D-2D link line

Once the active area was defined, the next step was to link it with the 1D channel by creating a link line, i.e. a polyline shape file, between the channel nodes and the 2D domain. The line along the Upton floodplain (which is on the west side, i.e. on the right bank) defines where the data enters the 2D domain; the link line attributes specified

linkages to the 1D channel network. The 'Link Line' Generator option within the FM was used to create this link between the 1D model nodes and the edges of the 2D model active area polygon (**Fig.** 6.11). As described above, link lines can be generated using three different approaches. Since for model simplicity spill units were removed, a 'level link' option was selected to create the link line.

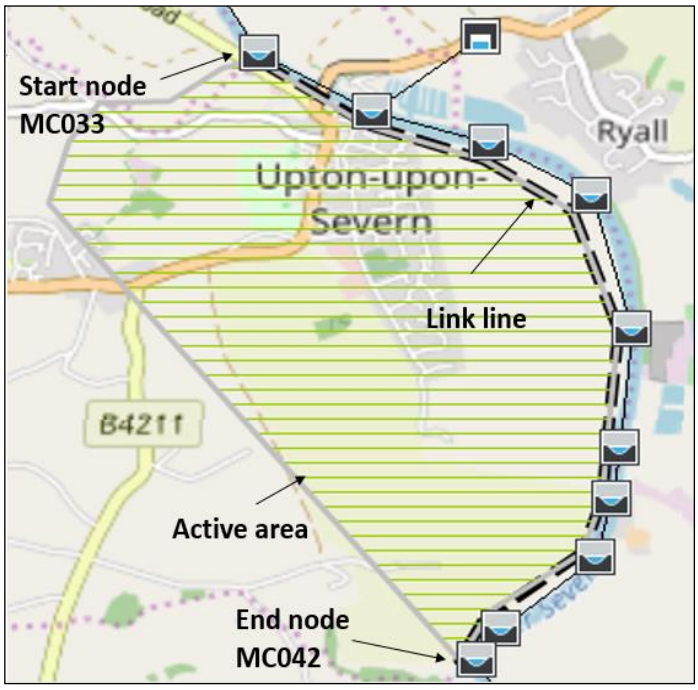


Figure 6.11: Link line and Active area.

6.6.2.5 Simulating linked 1D-2D model

The 1D-2D linked models for flood extent modelling were configured by setting up the initial conditions for each of the inflow hydrographs. Also, a DEM was required to define floodwater propagation on the floodplain. Further, several LiDAR DEM data with multiple resolutions, i.e. 3m, 18m and 90m,) is available for selection. The steps of running the 1D-2D linked model are described below:

a) The first step was to build a 2D simulation. This can be done by creating a new 2D simulation file from the menu: New project > Simulations > New 2D simulation. In the simulation window, the start and end time of the model were entered.
b) Within the simulation window 'Domains' tab, all the floodplain related spatial and temporal information were entered. Figure 6.12 depicts the 2D domain details

used to set up the simulation. The hydrograph was imported using the 'Boundary Conditions' tab, and output parameters were selected in the 'Outputs' tab.

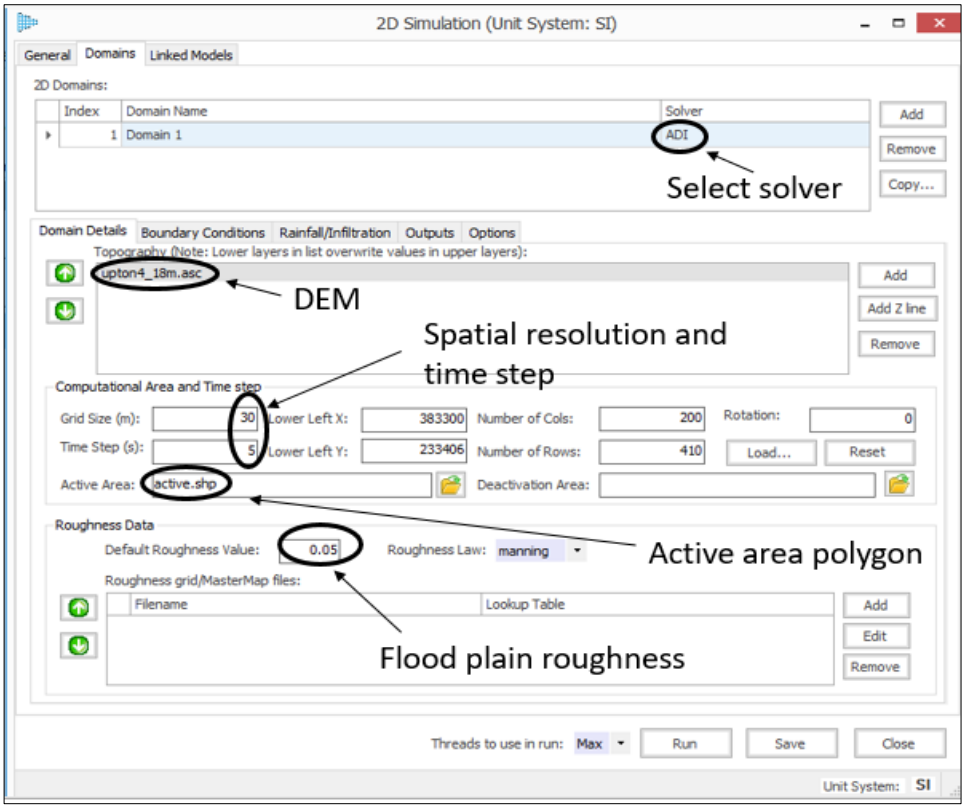


Figure 6.12: Domain details used for the 2D simulation.

c. In the 'Linked Models' tab, the 2D model was linked with the 1D unsteady model through the 1D-2D link line by selecting the previously created 1D simulation file, and the 1D-2D link line shapefile. The rest of the parameters were set to default.

Once all the input files were imported in the model and parameters values were set, the 2D model was simulated to generate the output files containing water depth values, saved as raster files at an hourly frequency. In addition to the raster files, the water depth values at randomly selected 100 locations within the floodplain were also saved as '.csv' files. To generate the water depth outputs for each of the synthetic hydrographs, steps 6.8.2.1, 6.8.2.2, and 6.8.2.5 were repeated.

6.6.2.6 Selection of cell size, solver and roughness values

Often the grid cell resolution is a user-defined parameter and is influenced by the floodplain topography. To effectively represent flow paths between buildings, considerably smaller cell size is required. Notably, selection of the cell size in the model

can be influenced by the total computational time required. The relation between cell dimension and total runtime is approximately given by:

$$Run\ time \sim \frac{1}{\Delta x^3} \tag{6.2}$$

Decreasing the cell size by half could increase the computational time approximately by a factor of 8. This is because halving the cell size would force the FM to calculate for the number of cells which is four times more and may also require to reduce the time-step (JACOBS, 2017). Thus, it is important to select a cell size carefully before running the model. Here, a combination of moderate cell size (30m) and larger time steps (5 seconds) were used to keep the computational time comparatively low.

The robust ADI solver, known to have relatively short runtime and does not require any supercritical flow calculation, is selected. Also, the model roughness parameters (channel and floodplain) need to be chosen carefully. In this study, a uniform value of T0.025 was used as the channel roughness value and 0.05 was used as the floodplain roughness coefficient.

Considering the main objective of the study, which was to investigate if flood extent maps produced by the 2D model could be emulated using different ML algorithms, the FM simulations were simplified to generate water depth values, and full calibration of 2D model was ignored.

6.7 Generating training and validation data

6.7.1 Preparing the output data

The output files (water depths) generated from the FM simulations were saved in two formats, as raster and '.csv' files. The '.csv' files were used as target variables for training and validating ML-based models at the sampled locations, and raster files were used for the visual representation of the flood depths.

6.7.2 Preparing the input data

The work done for the Niger case study suggested that using the upstream flow with its two lagged values and time as a separate input are sufficient to estimate acceptable results with ML-based models. Thus, the two lagged flows with time as an additional input variable at the upstream boundary were considered as input variables initially. Since, the

upstream boundary (MC010) is located about 9 km (vertical distance) away from the top node (MC033) of the 1D-2D link line, to reduce the possible error caused by the distance, and river morphology, water flow at node MC033 - produced by the 1D unsteady simulation - with two lagged flows were also added as input variables. Thus, there were seven input variables (**Eq.** 6.3) for each of the 100 randomly selected locations for five hydrographs.

$$Y_i = f(Q_{MC010,t}, Q_{MC010,t-1}, Q_{MC010,t-2}, Q_{MC033,t}, Q_{MC033,t-1}, Q_{MC033,t-2}, t) \quad (6.3)$$

Where, Y_i is the state of the sampled cell, t is the discharge observation time and Q is the discharged rated with lagged values for MC033 and MC010.

To facilitate sufficient initialisation time for the hydraulic model, rows containing values for the first three hours from the input '.csv' file were not supplied for the training process. Model initialisation is essential to improve the convergence by gradually updating the initial conditions aligning towards the boundary conditions. Thus, estimated water depth values for the first three hours for each of the five hydrographs were removed from the target data.

6.8 Training the MLP

The MLP classifier, as detailed in Chapter 5, was used. These classifiers take seven input variables and presence or absence of water at the sampled locations as training data. Water depth values at the sampled locations were binarized, before inputting in the training process. Since for different pixels flood arrival time is different and a pixel was considered wet only when the water depth value exceeded 0.09m. Water depth values less than or equal to 0.09m were set to 0. In flood risk assessment, different threshold values are selected in previous studies for various purposes, for example, Bermúdez et al. (2019) selected 0.1m, 0.2m and 0.3m of water depths and Aldridge et al. (2016) selected 0.15m and 0.3m. Therefore, since very smaller depth values are insignificant and could potentially add noise to the ML-based models, a threshold value of 0.09m was chosen. This choice of threshold value allows one to select any cut-off value over 0.09m (e.g. 0.1m, 0.15m, etc.) for risk assessments, and eliminates the noise caused by the insignificant depths.

For training the MLP classifier for each of the sampled locations, a total of 1226 input and wet/dry cell instances from four hydrographs were used. The models were trained

following an iterative process, i.e. before repeating the process for the next location, at every iteration, the trained model was saved. Thus, 100 separate models, each with different weight and bias matrices, were trained.

The *'Keras'* neural network package (Chollet and others, 2015) was used to build the model. The step by step modelling approach is described below:

Step 1 *Load 'Jupyter notebook':* Any convenient Interactive Development Environment (IDE) software that supports the *Python* programming language can be used for developing model code. For this study, *'Jupyter notebook'*, a web application, which simultaneously allows creating and sharing the script documents, equations, visualizations as well as texts, was used to build the models. The preinstalled application can be called in a Linux terminal or Anaconda command prompt:

```
$ jupyter notebook
```

Step 2 *Import essential python packages and modules:*

```
##library for data manipulation, data analysis
import pandas as pd
##for plotting
import matplotlib.pyplot as plt
##library containing mathematical functions
import numpy as np
##for saving the models
import h5py
##for data preprocessing
from sklearn.preprocessing import StandardScaler
##neural network model building
from keras.models import Sequential
##core layers for the neural network
from keras.layers import Dense
##for loading the saved trained models
```

```
from keras.models import load_model
```

Step 3 *Load '.csv' data files:* The *pandas* dataframe function was used to import the '.csv' files containing train and test data. *Pandas* is a software library providing efficient, convenient data structures and data analysis tools for *Python*. The test datasets were loaded and used simultaneously for testing purpose in every iteration at a given location.

```
train= pd.read_csv("training_data.csv")

test= pd.read_csv("test_data.csv")
```

Step 4 *Data pre-processing:* before defining the classifier architecture and training, the data were pre-processed for better interpretability and to remove potential outliers. To do so, first, the input variables and target labels were defined. Then the continuous water depth values were binarized.

```
##input variables- each column represents an individual input variable
X = train.iloc[:, 0:7]
X_test = test.iloc[:,0:7]

##target data- each column represents one sampled location
Y = train.iloc[:,7:108]
Y_test = test.iloc[:,7:108]

##replace depth < 0.09 = 0, because 1st arrival of water can happen
between an hour and next that ML can't detect- to reduce
misclassification chances

Y[Y <=0.09] = 0
Y[Y !=0] = 1
Y_test[Y_test <=0.09] = 0
Y_test[Y_test !=0] = 1
```

Step 5 *Data scaling:* The data were normalized by using the *'StandardScaler()'* function from the *'preprocessing'* module in the *sklearn* library (Pedregosa et al., 2011). The function transforms the input data so that the data distribution will have a mean of 0 and

a standard deviation of 1. This is done by subtracting each value in the dataset by the sample mean, and then dividing the remainder by the standard deviation.

```
## Define the scaler
scaler = StandardScaler().fit(X)
## Scale the train set
X_train = scaler.transform(X)
## Scale the test set
X_test = scaler.transform(X_test)
```

Step 6 *Define model architecture:* next task was to construct the ML-based model for the binary classification problem, where 0 means a 'dry' pixel and 1 means a 'wet' pixel. The MLP, a simple stack of fully connected layers, was constructed using the 'Sequential' model - a linear stack of layers from the Keras package. The MLP consisted of one input layer, one hidden layer with multiple neurons and an output layer. The model was defined by adding three 'Dense' layers, i.e. fully connected layers. A 'Dense' layer conducts the following operation:

$$\text{Output} = \text{activation}(\text{dot}(\text{input}, \text{kernel}) + \text{bias}$$

While constructing the model, it is important to define the input shape of the first layer, i.e. the shape of the input variable matrix. The first parameter in the layer is the output size of the layer. The input layer and the hidden layer take the 'relu' activation function, and the output layer takes the 'sigmoid' activation function for probabilistic output.

```
# Initialize the constructor
model = Sequential()
# Add an input layer
model.add(Dense(8, activation='relu', input_shape=(7,)))
# Add one hidden layer
model.add(Dense(6, activation='relu'))
# Add an output layer
model.add(Dense(1, activation='sigmoid'))
```

Constructing the model architecture represents a trade-off which needs to be considered carefully. It is possible to add more hidden layers and units, which enables the model to learn more complex representations, but at the same time, it will be computationally expensive and can be susceptible to overfitting. Here, to keep the model simple, only one hidden layer with six hidden units was used. The input and output layers consisted eight (seven input variables and one bias) and one unit, respectively.

Step 7 *Compile the model:* after defining the model structure, the next step was to compile the model, which was done by passing the appropriate arguments into the *'compile()'* function. Since predicting the presence or absence of water is a binary classification problem, the *'binary_crossentropy'* loss function was used. The computationally efficient and widely used *'adam'* optimizer as an optimization algorithm was passed to the *'optimizer'* argument. Accuracy, i.e. training error - during the training process was monitored by passing the 'metrics' argument.

```
model.compile(loss='binary_crossentropy',

              optimizer='adam',

              metrics=['accuracy'])
```

Step 8 *Fit the model on training data:* The model training process was initiated by calling the *'fit()*' function. The MLP was then trained for 100 epochs using all training data in batches of 20 samples. An epoch is an iteration over the entire training set. The batch size defines the number of samples to be propagated through the network at a time. A small-batch number optimizes network efficiency because the network will not be loaded with too many input patterns into memory at the same time.

```
# select the target labels and flatten the array

Y_train = np.ravel(Y[column])

#Fit the model

model.fit(X_train, Y_train, epochs=100, batch_size=20, verbose=1)
```

Step 9 *Predictions:* Depending on the computer configurations, the training process can vary in duration. For example, using an Intel Core i5-2520 2.5 GHz (RAM 8GB) machine,

the training process can take between 5 and 14 hours. Once the model was trained, the predictions can be made for the test data.

```
# Predict

y_pred = model.predict(X_test)
```

The predicted values were stored as a 'pandas' dataframe. Since the output was a vector of probabilities, conditional arguments were applied, i.e. a value > 0.5 is 1 and < 0.5 is 0, to binarize the values. After binarization, the predicted values were saved for further statistical analysis.

```
pred=pd.DataFrame(y_pred, columns=[column])

pred[pred >= 0.5] =1

pred[pred < 0.5] =0
```

Step 10 *Save the model:* At the end of the full training process, a trained model was saved for each of the sampled locations.

```
fname = column+".h5"

wd = '/c/Users/FloodModeller_Simulation/Models_with_time_MLP/'

model.save(wd+fname)  # creates and saves a HDF5 file 'my_model.h5'
```

6.9 Regression based approach

Following the analogy detailed in Chapter 5, a regression-based approach was applied to test the results by comparing the observed results to those generated by the MLP classifier. Two different types of regressor models, namely, RF and SVR were configured in this study. In the following sections, the modelling approaches for both techniques are described in detail.

6.9.1 Support Vector Regression

Similar to the Niger case-study, a nu-SVR-based model was constructed utilizing the *sklearn* ML library in *Python*. Training and testing datasets used for this exercise were the same as those used for training and testing the MLP based classifier. For this non-

linear regression-based approach, the target values were not binarized before the training process (regression models can only be fitted to continuous data), but the predicted values were converted to 0s and 1s to produce binary maps. Values less than 0.05m were set to 0 and greater than 0.05m were set to 1. This cut-off value was selected to reduce the offset error at 0 water depth values during low flows.

Similar to the MLP-based model building process, essential libraries and modules were imported into the *'Jupyter notebook'* dashboard. Then the input and target datasets were declared, which were the same as above except for the target data format. After loading the scaled input-target data, the regression model was configured. The model parameters were selected using the traditional exhaustive trial and error process. The Mean Squared Error (MSE) and R-squared scores were used to check the model accuracies on the test data.

```
#import libraries and modules
from sklearn.svm import NuSVR
from sklearn.metrics import mean_squared_error, r2_score
#depth value at location id 100
Y_train = Y.Depth_100 #continuous values not binarized
#Configure the model
svr_rbf = NuSVR(kernel='rbf',gamma=0.1, C=1e3, nu=0.5)
#Fit the model
y_rbf = svr_rbf.fit(X_train, Y_train)
#Predict
pred_svr = y_rbf.predict(X_test)
```

The predict values using the test datasets were converted to 0s and 1s for further water presence/absence analysis.

6.9.2 Random Forest

An RF-based model was also configured, trained at selected points, and the resulted water depth values were compared against the SVR-based model. The '*RandomForestQuantileRegressor()*' function within the *skgarden* library was used to configure the model.

Since RF is an ensemble method comprising several decision trees, the main parameter to set in the model is the number of trees in the forest. In this model setup, the *'n_estimators'* argument takes the number of trees to consider. The *'max_depth'* argument is set as *none*, meaning that the nodes are expanded until all leaves contain less than *'min_samples_split'* argument input. The *'min_samples_split'* refers to the minimum number of samples required to split a node. The value passed to the *'max_features'* argument is 7, which is the number of input features. All other parameters were set to the default values.

```
Qrf = RandomForestQuantileRegressor(bootstrap=True, criterion='mse',
max_depth=None, max_features= 7, max_leaf_nodes=None, min_samples_leaf=1,
min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=550,
n_jobs=1, oob_score=False, random_state=0, verbose=0, warm_start=False)
```

A simple step-wise selection method, i.e. 100, 150,…,550, 600, was applied for choosing the number of trees. Generally, the MSE value is used to select the optimal number of trees. Increasing the number of trees often results in a decreased MSE value, but the computational time can increase significantly. Therefore, the number of trees to be used in the forest must be selected carefully. There are two things to consider while selecting the optimal number trees, that is- (a) the trade-off between computational time and the number of decision trees, and (b) look for the number after which there is no considerable improvement in MSE value can be achieved. After careful consideration, the number of trees for the RF-based models was set to 550 for this study.

After configuration, the model was fitted to the training dataset. Notably, a minor scaling of the target data is done by multiplying them by 100 (e.g. 0.25m is rescaled to 25m) to split the water depth values more rigorously. After fitting the model on the training data, one can select trees for examination in greater detail. However, visualizing the trained trees with the through splitting from the root node to leaf nodes could be tedious as it contained 17 layers. Thus, for better understanding, a smaller image was produced by limiting the *'max_depth'* value to 3 (see **Fig.** 6.13) for a randomly chosen tree.

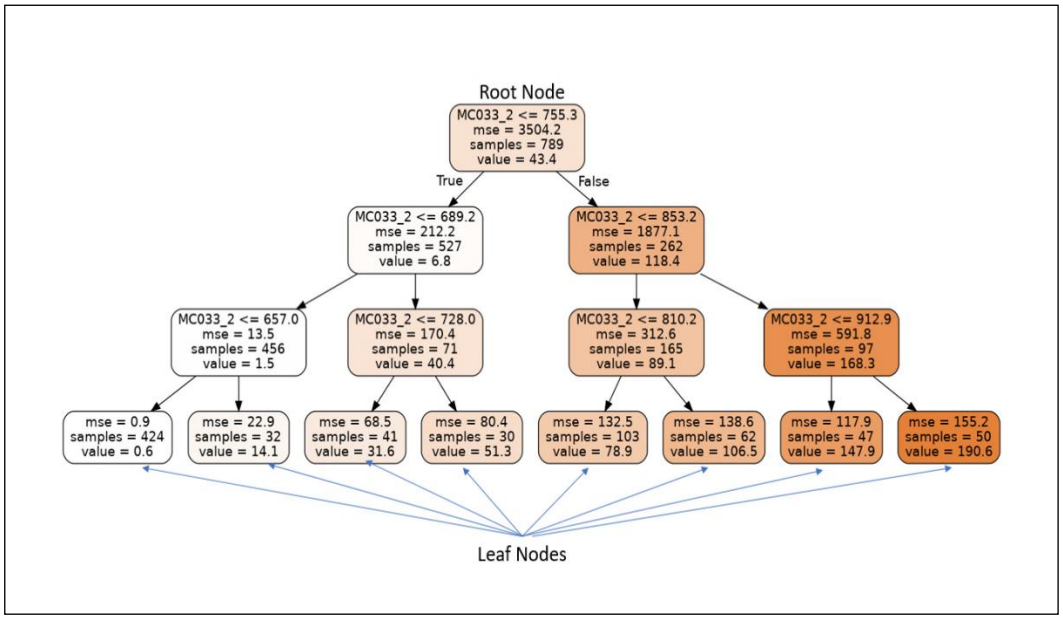


Figure 6.13: A reduced-size tree.

Interestingly from the reduced tree, it is possible to make effective predictions for the new dataset. For example, consider a new observed value of 800 m^3/sec is made available at the cross-section node MC033. Then by starting at the root node and meeting the first splitting criterion, i.e. the observed value is greater than 757.3, we move to the right-wing of the tree. Then, we go further down and meet the next criterion <= 810.2. At this point the leaf node having the lowest MSE will be selected, and hence, we conclude that the water depth at location ID 100 is 78.9cm. An interesting point to note is that in the root node 789 training samples are used, although the total number of samples available is 1228. This is because each tree in the forest was trained on a random subset. Furthermore, from the Figure 6.14, it can be seen that only one variable is used to make the prediction, which means for this particular tree, only MC033_2 feature is important for making water depth predictions.

6.10 Generating flood maps

To produce the flood extent maps, the wet/dry conditions predicted by using both classification and regression approaches, at sampled locations within the study area were spatially interpolated. This was done by estimating the values of target quantity, i.e. water presence at unsampled locations. In geostatistics, since 1970s kriging and several other variants are proposed as spatial interpolation techniques (Hengl et al., 2004). However, from the 1990s, several 'hybrid' interpolation methods that combine both kriging and

'auxiliary' data, i.e. remote sensing images, land-use maps, etc., have been developed (Hengl et al., 2008, Hengl et al., 2007). The so-called regression-kriging (RK), one of such hybrid interpolation technique, integrates regression on auxiliary information with simple kriging to interpolate the residuals from the regression model (Hengl et al., 2004):

$$\widehat{Z}(s_0) = \sum_{k=0}^{p} \widehat{A_k} \cdot q_k(s_0) + \sum_{i=1}^{n} \gamma_i \cdot e(s_i) \qquad (6.4)$$

where, $\widehat{A_k}$ are the regression coefficients estimated from fitting a regression model, q_k are the auxiliary variable values q at target locations k, γ_i are kriging weights estimated from the spatial dependence (spatial autocorrelation) structure of the residual, and $e(s_i)$ is the residual at location s_i. RK has been shown to be an interpolation technique superior to other conventional geostatistical approaches, yielding more accurate and detailed results (Bishop and McBratney, 2001). For more details and empirical examples of RK please refers elsewhere Hengl et al. (2008).

6.10.1 Point data interpolation

The *R* software was used for performing statistical computing and visualization of the RK method, in case of both classification and regression studies. A step by step procedure for producing interpolated flood extent maps is described briefly (with snippets) in the following sections.

6.10.1.1 Import point data

The resulted '.csv' files from classifier and regressor algorithms were loaded into R. Hourly data for each sampled location with their coordinates were then converted to *'SpatialPointsDataFrame'* using the *'sp'* package (Pebesma and Bivand, 2005). To quantify spatial variation, semivariance from point observations was used, which is half the expected squared difference between the values of the variable, i.e. water presence, at two locations.

For a robust model development, at least 50 points should be present at various spacings (Hengl et al., 2007). In the present case, about 99 samples were available (one point was rejected due to an erroneous coordinate), 89 samples were used for calibration, and the remaining 10 were used for validation (**Fig.** 6.14).

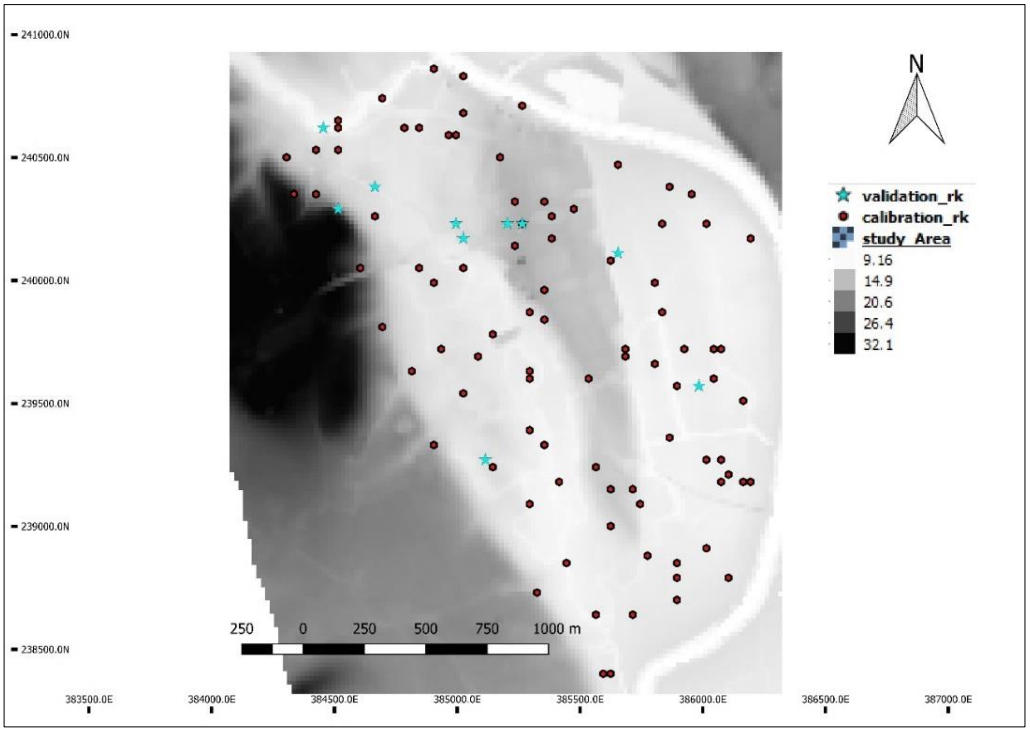


Figure 6.14: Sampled points for RK model calibration and validation plotted over a DEM.

6.10.1.2 Load auxiliary information

The ground elevation values were used as an auxiliary predictor to interpolate the presence and absence of water body within the study domain. A DEM, 18m x 18m raster file, was loaded, and the elevation values from 89 sampled locations were extracted. The elevation values were then added as a separate column to the previously created *SpatialPointsDataFrame*.

```
#import libraries

library(raster)

library(sp)

library(gstat)

library(rgdal)

df<- read.csv("RK_database_withtime.csv") #load point data

##remove columns with ID, X,Y coordinates, 325 is the duration of the flood
simulation in hours

dt<- df[c(1:89), c(4:325)]

##############################
```

```
i = 74 # for fitting RK model

##############################

f<- data.frame(Loc = df$location[1:89], X =df$X[1:89], Y = df$Y[1:89], Hour
= dt[[i]])

coordinates(f)=~X+Y   # this creates a SpatialPointsDataFrame

str(f)

#add coordinate reference system to British National Grid (OSGB:1936)

crs(f) <- CRS("+init=epsg:27700")

#load DEM raster

dem <- raster("study_Area.asc")

#load the same elevation file as asciigrid

slope1 <- read.asciigrid("study_Area.asc")

#assign coordinate system

crs(slope1) <- CRS("+init=epsg:27700")

#extract elevation values from DEM at calibration locations from-
#SpatialPointsDataFrame, f: then add elevations values to a column

#####

elevation <- extract(dem, f)

f$Elevation <- elevation
```

6.10.1.3 Fit regression model

The next step was to select explanatory variables and fit a linear model.

```
lm.depth <- lm(Hour~Elevation, as.data.frame(f))
```

6.10.1.4 Estimate residuals and autocorrelation structure (variogram)

To fit variograms automatically to the raw data and to produce final predictions, the *gstat* package was used. The most commonly used exponential and spherical models were applied separately for fitting the variograms, and the weighted least squares method was used to emphasise shorter distances and lags with the higher number of pairs.

```
#model initial parameters
```

```
null.vgm <- vgm(NA, "Sph", NA, NA)  #spherical

#null.vgm <- vgm(NA, "Exp", NA, NA) #exponential

#fit variogram

vgm_depth_r <- fit.variogram(variogram(Hour~Elevation, f),
model=null.vgm)
```

An example of a fitted variogram, using the *exponential* model is given in **Fig.** 6.15. The plot shows the *semivariance* as a function of the distance between the point locations. Further explanation on the features of the variogram is given in **Sect.** 6.12.3.

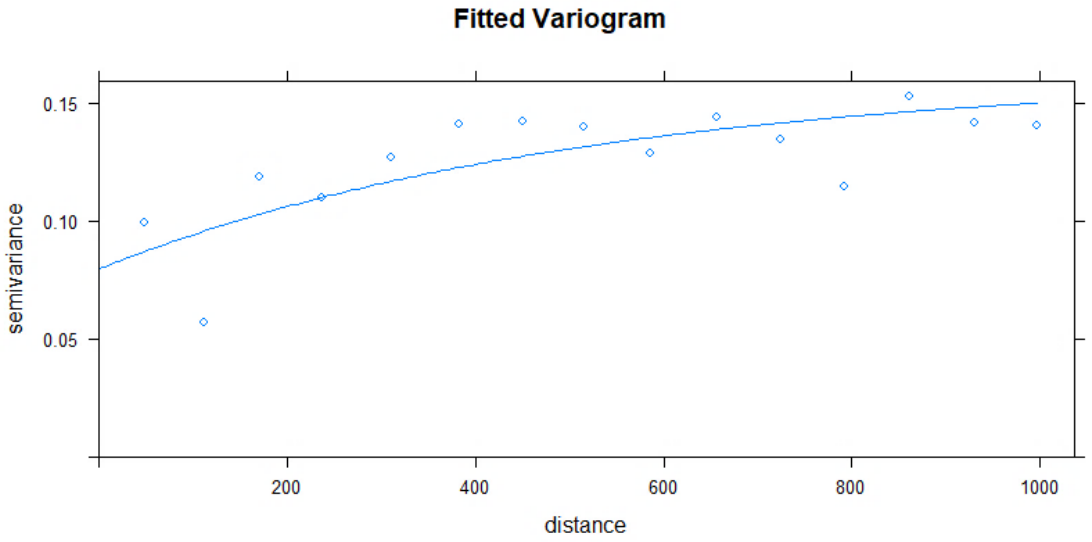


Figure 6.15: The variogram showing spatial autocorrelation structure.

6.10.1.5 Prediction of water presence

The final step was to apply the regression model to all unobserved locations, i.e. a grid, and krige the residuals.

```
wtd <- krige(Hour~Elevation, locations=f, newdata=slope1,
model=vgm_depth_r)
```

The method of regression combined with kriging could yield values outside the physical range and thus, would require manual masking or replacement (Hengl et al., 2004). In

this study, some unusual probabilistic values were found. Thus, values less than 0 and greater than 1 were set to 0 and 1, respectively. In addition to the predicted water presence/absence map, the *gstat* package also enables one to estimate the uncertainty of the prediction model by producing a variance map (Hengl et al., 2007).

The predicted map is a probability map. To create a binary map, values less than 0.5 were set to 0 and values greater than 0.5 were set to 1. After generating a binary map, the predicted outputs were compared at 10 validation sample locations. The validated model was then used for producing hourly flood extent maps for the entire flood duration.

6.11 Evaluation criteria

6.11.1 ML-based model evaluation

Predicted presence/absence of water at the sampled locations generated from different ML-based models were evaluated using multiple error metrics. One could use a confusion matrix to quantify the total accuracy of each of the models. However, by doing so, the results could be misleading as there were about 28 points, which were never flooded and correctly classified by the models as 'dry'. Hence, averaging classification results from the never flooded pixels and the pixels that were both dry/wet at some point during the flooding period would bias the results by overestimating the model accuracy. It is possible to drop these dry pixels from the accuracy analysis and compute the so-called *F* score (**Eq.** 5.1) (Aronica et al., 2002). In addition to using the *F* score, it was decided to keep all the samples and add 'precision' and 'recall' metrics to demonstrate how precisely the model predicts 'true positives' (correctly predicted as flooded by the ML-based models) to the total predicted positives (the sum of correctly and wrongly predicted as flooded) and to the total actual positives (predicted as flooded by the FM):

$$Precision = \frac{True\ positive}{Total\ predicted\ positive} \tag{6.5}$$

$$Recall = \frac{True\ positive}{Total\ actual\ positive} \tag{6.6}$$

$Precision$ can be used to quantify model performance when the cost of 'false positive' is high, meaning that a lower precision value would depict that many 'dry' cells were classified as 'wet'. On the other hand, $Recall$ calculates how well the model captures a 'positive' as 'true positive'. This is the metric to be used for selecting the best model when there is a high cost associated with 'false negative'. $Recall$ quantifies the chance

of a model incorrectly classifying a 'wet' cell as 'dry'. A lower $Recall$ value means an increased probability of a 'dry' cell being labelled as 'wet'.

Instead of running individual metric analyses to find an optimal blend of $Precision$ and $Recall$, it is possible to use the so called *F1-score* which is the harmonic mean of $Recall$ and $Precision$:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6.7}$$

For regression analysis performance of the SVR-based model was first compared at as many as ten locations against the RF-based model using inferential error statistics (RMSE, r-squared and NSE) before performing flood extent modelling. The better performing model was applied further for classifying wetted pixels within the floodplain.

6.11.2 RK model evaluation

10 validation samples left out during the RK model calibration process were used for model evaluation. The model was calibrated using the wet/dry values at the 77th hour when the floodplain started to get heavily flooded. The 77th hour was chosen subjectively during the flood growing period. It is also possible to use wet/dry cell values from a different time instance for model calibration, but care has to be taken to make sure samples are evenly distributed (number of wet and dry cells are close to even). If samples are chosen very early, then most of the cells will be dry, and if samples are taken during the peak flood, then most of the cells will be wet. At 77th hour, number of wet and dry cells were nearly even.

6.11.3 Flood extent evaluation

Since the cell sizes and the extent of the generated RK maps were not identical to the FM simulated maps, before conducting any quantitative analysis, maps generated from both methods were transformed into identical cell sizes and extent. This process is called image co-registration/remapping and can be done using the *gdal* library in *Python*. The cell sizes were set to 30m, which is the resolution of the FM extent maps. The extent of the maps was also set to the extent of the 'Active area'. Accuracy of the final inundated binary maps (*F* score) was then compared against the FM simulated maps using **Eq.** 5.1.

6.12 Results and analysis

6.12.1 MLP-based model results

This section presents the results obtained from the MLP based flood extent mapping method. First, the average predictive accuracy of this method is given in terms of total accuracy and *F1-score*. To do so, a confusion matrix was produced for each sampled location, then sample accuracy was computed by dividing the total 'correctly predicted wet and dry' states by flood duration. Then the total accuracy of the MLP based approach was computed by averaging overall sample accuracy by the number of sampled locations. The '$Precision$' and '$Recall$' values were also computed from the confusion matrix. Once the '$Precision$' and '$Recall$' values were calculated, the *F1-score* can be determined using **Eq.** 6.7.

The initial model, as described in Section 6.8, included 'time' as an input variable. Also, to investigate the 'time' dependency of the models, a separate model was developed that did not incorporate 'time' during the training process. From here, the MLP-based models trained without 'time' variable will be referred as MLP-2. Table 6.4 shows the overall accuracy and the *F1-score* of the MLP based approach at 100 sampled locations. The scores show that predictive performance improves when time is counted as an input during the training process while the exclusion of time decreases overall accuracy only fractionally. Thus, the MLP-based models do not depend on the time of observed flows at upstream. Also, the inferential statistics presented below confirm that the MLP based classifier could effectively detect wet cells during a flood event. The *F1-score*s also indicate that the models were not biased, as indicated by the 'overall accuracy' values. This is because, unlike 'overall accuracy', *F1* values are only calculated for the cells that were flooded at some point during the event.

Model performance with 'time' variable		**Model performance without 'time' variable**	
Overall Accuracy	Overall *F1-score*	Overall Accuracy	Overall *F1-score*
0.992	0.986	0.979	0.979

Table 6.4: The MLP error statistics.

The error measures show a promising model performance in classifying wet and dry cells. However, it is not clear from this analysis how well the MLP-based models perform in

accurately predicting flood arrival time. Flood arrival time is the time at which a particular cell becomes wet from the dry condition. While the calculation of arrival time is not necessary to detect wet/dry cells, it is important to compute the uncertainty range of the model. The uncertainty range here is essentially a 'time window' that allows one to forecast the probable time of a cell 'changing its initial condition' (dry to wet). For example, if we observe the flood arrival time at location ID 17 and 73, we see two opposite responses from the MLP-based models (**Fig.** 6.16). At ID 17, the flood arrives 3 hours ahead of the MLP prediction while at ID 73 there was a 2-hour delay in actual flood arrival. The descriptive statistics of the flood arrival times were calculated at the sampled locations, including a 95% confidence interval (CI) indicating the prediction certainty range.

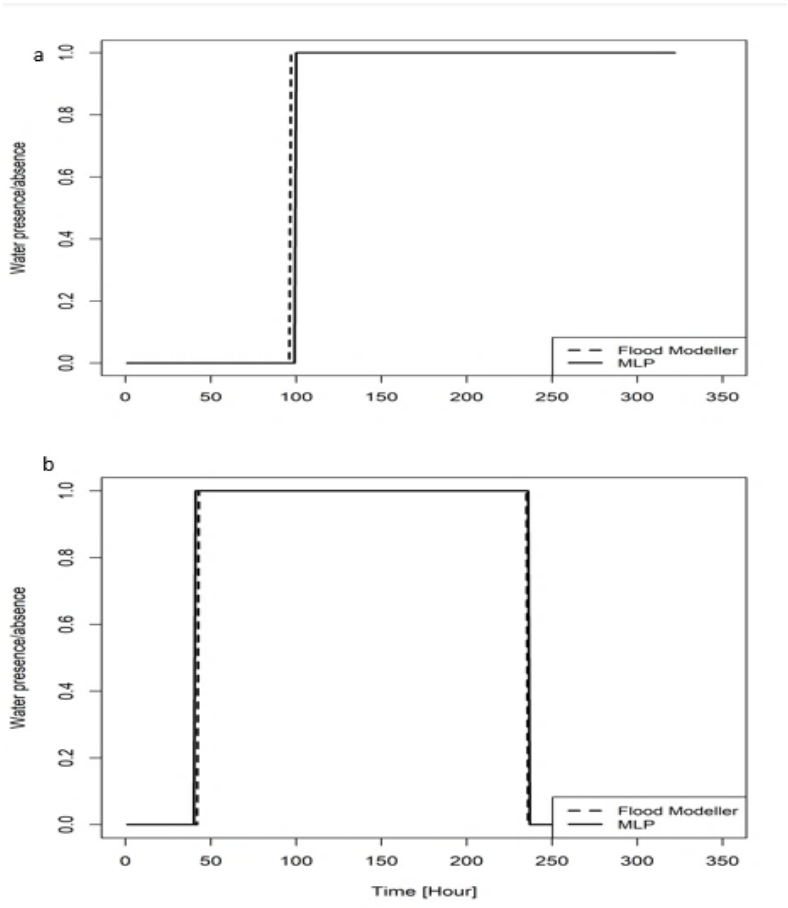


Figure 6.16: (a) Sample location 17 goes from initial 'dry' to 'wet' condition at the 100th hour while MLP changes its state at 97th hour. (b) At ID 73, MLP changes state 2 hours ahead.

Table 6.5 illustrates the outputs generated by the MLP-based model, including 'time' as an additional input variable.

Mean	Median	Min	Max	1st Quartile	3rd Quartile	95% CI
-17.4min	-1h	-2h	4h	-1h	0h	[-2h, +3h]

Table 6.5: Descriptive statistics of predicted flood arrival time at sampled locations (including time as an input).

The average difference in arrival time is approximately -17.4 minutes, meaning that the sampled locations changed their states from 'dry' to 'wet' on average 17.4 minutes earlier than predicted by the MLP classifier.

In contrast, the output from MLP-2 showed unusual characteristics. Although overall accuracy and the *F1-score* were fractionally lower than those of the initial MLP-based model, there were several cases in which the MLP-2 models sent 'False Positives', undermining confidence in the model. 73 locations flooded at some point during the flood event, of which 34% were affected by false alarms. Figure 6.17 shows the flood arrival times of the same location IDs as mentioned above, i.e. 17 and 73. While at ID 73 the model seemed to be performing extremely well, at ID 17 the model did not produce a convincing result. The effect of 'False Positive' and 'False Negative' is apparent: the model predicted a change of state, i.e. from dry to wet, well ahead of the actual transition. In addition, the model triggered a 'False Negative' during the flooding condition and then went back to a 'wet' state, causing an even higher degree of uncertainties. This is probably an artefact caused by the standard cut-off values. The MLP-based models predict outputs in a probabilistic format. The probabilistic values were converted to a binary array using a standard cut-off value, i.e. 0.7. One can evaluate the model performance by tweaking the cut-off value. To assess the impact, a set of cut-off values from 0.4 to 0.7 were set and tested for MLP-2 models. However, the results did not seem to increase much because, for any cut-off value, there seemed to occur a 'False Positive' at some point during the event for most of the sampled locations. Thus, the standard cut-off value was used to derive the final results.

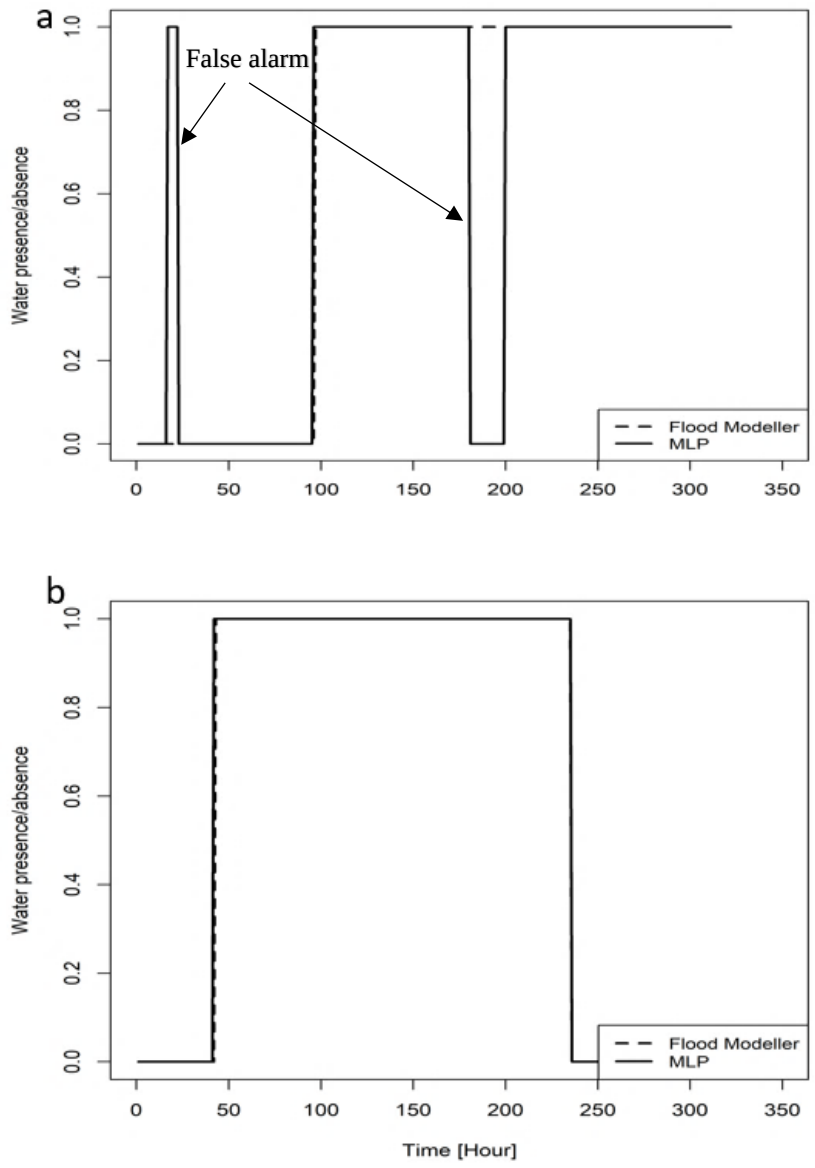


Figure 6.17: (a) multiple false alarms triggered at sample location 17. (b) At ID 73, MLP-2 performed well to predict state transitions.

For MLP-2 models, descriptive statistics with a 90% confidence interval (with 95% CI large predictive uncertainties found) are presented in Table 6.6. The predictive uncertainty of the models was large, which is expected because of the model's tendency to trigger false alarms well ahead of actual flood arrival time.

Mean	Median	Min	Max	1st Quartile	3rd Quartile	90% CI
-11h	-1h	-80h	1h	-25h	-1h	[-33h24m, +1h]

Table 7.6: Descriptive statistics of predicted flood arrival time at sampled locations (without time as an input).

Analysing the accuracy statistics from Table 6.4 and descriptive statistics for the MLP-2 models, it is indeed difficult to draw a conclusive remark on the performance of these models due to their counter-intuitive results. Thus, overall accuracy and *F1-score* show a good match (over 97%) between reference and predicted wet/dry cells, but at some locations, arrival time error is extremely large and unreliable.

To further illustrate the range of false alarms for both modelling approaches, a comparative histogram is presented in Figure 6.18. The histogram shows the spread of arrival time differences. Modelling a highly nonlinear case such as flood extent forecasting, one has to negotiate the false alarms triggered by the models. The aim is to keep the range of false alarms within an acceptable range. Out of all the sampled locations, there was only a small number, i.e. 5, where the MLP-based models triggered a 'False Negative' meaning that the actual flood arrived at least 3 hours ahead of predicted arrival times, and at most locations, the predictions were 0-2 hours ahead of actual flood arrival. On the other hand, a multitude of sampled locations suffered from 'False Positives' when the predictions were made without observation 'time' as an input.

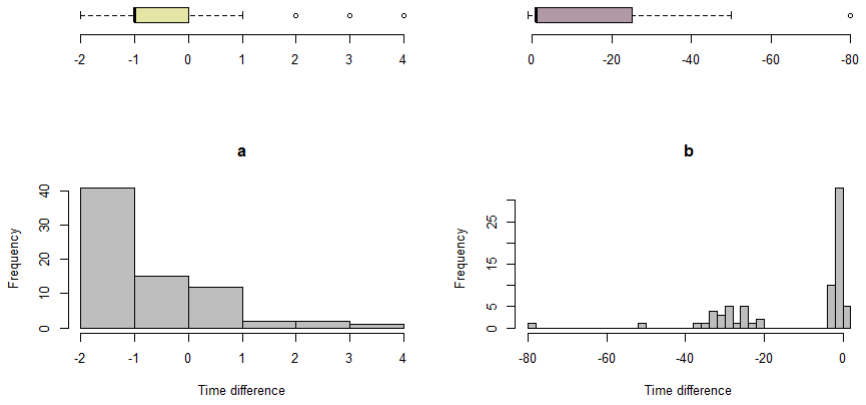


Figure 6.18: Histograms of MLP (a) and MLP-2 (b) showing residual dispersions.

MLP-2 residuals are mostly clustered in two spaces. While most of the residuals are normally distributed at around a mean of 0, the degree of uncertainty of triggering a false alarm over 20 hours ahead is the issue truly hindering the extrapolation of such a modelling approach. The purpose of conducting this analysis was to detect any discernible differences in model outcomes depending on whether input observation times of an inflow hydrograph has been used. The arrival time error analysis shows that the models without time as an input variable are highly uncertain. Therefore, further in this

study, we will only consider the initial MLP-based model outputs (those without the time variable) for generating interpolated flood extent maps.

6.12.2 Regression based model results - SVR vs RF

In Chapter 5, the flood maps produced using the nu-SVR-based approach were in par with the MLP generated maps at 200 sampled locations. It was also observed from the Niger case study that the SVR-based models were less accurate during the flood growing period. In the current case, the spatial resolution of the domain and the temporal resolution of the flow data are both higher. If the models are extremely uncertain in predicting the arrival time, this can significantly reduce the value of ML based flood inundation models for urban areas. Therefore, RF was also considered, and the results were compared against SVR at as many as ten locations before applying any particular regression-based models to all sampled locations. RF-based models are known for their lower susceptibility to overfitting and generalization capabilities.

Initially, the SVR and RF-based models with constant model parameters were trained at 10 randomly selected locations to predict water depths. The comparison of their error statistics is given in Table 6.7. Both models performed extremely well in predicting water depth values. The average RMSE values are almost 0, whereas the R-squared and NSE are both near perfect. While both models perform compatibly equal, the RF produced slightly better results than the SVR on average.

Location ID	RMSE_ RF	RMSE_ SVR	Rsqrd_ RF	Rsqrd_ SVR	NSE_ RF	NSE_ SVR
24	0.024	0.027	0.999	0.999	0.999	0.999
31	0.018	0.019	0.992	0.991	0.992	0.991
48	0.018	0.014	0.998	0.999	0.998	0.999
55	0.027	0.056	0.999	0.994	0.999	0.994
67	0.027	0.030	0.999	0.999	0.999	0.998
71	0.025	0.020	0.996	0.998	0.996	0.997
79	0.027	0.031	0.999	0.998	0.999	0.998
86	0.027	0.041	0.999	0.998	0.999	0.998
99	0.028	0.038	0.995	0.991	0.994	0.990
100	0.021	0.019	0.997	0.997	0.996	0.997
Avg.	**0.024**	**0.030**	**0.997**	**0.996**	**0.997**	**0.996**

Table 6.7: Error statistics of SVR and RF-based models with observation time.

The simulated flows generated using the validation data set are shown in Figure 6.19. It is clear from the analysis that both models are capable of predicting water depth during

the high flows and can be used for flood extent modelling. However, as found in Chapter 5, the SVR-based models were overestimating flooded cells during the transition periods, i.e. a cell going from the dry state to wet in observational data and vice versa, the same effect was also present here. The SVRs were slightly unstable during this period, while the RF-based models performed relatively better. Notably, these instabilities could cause the model to trigger false alarms.

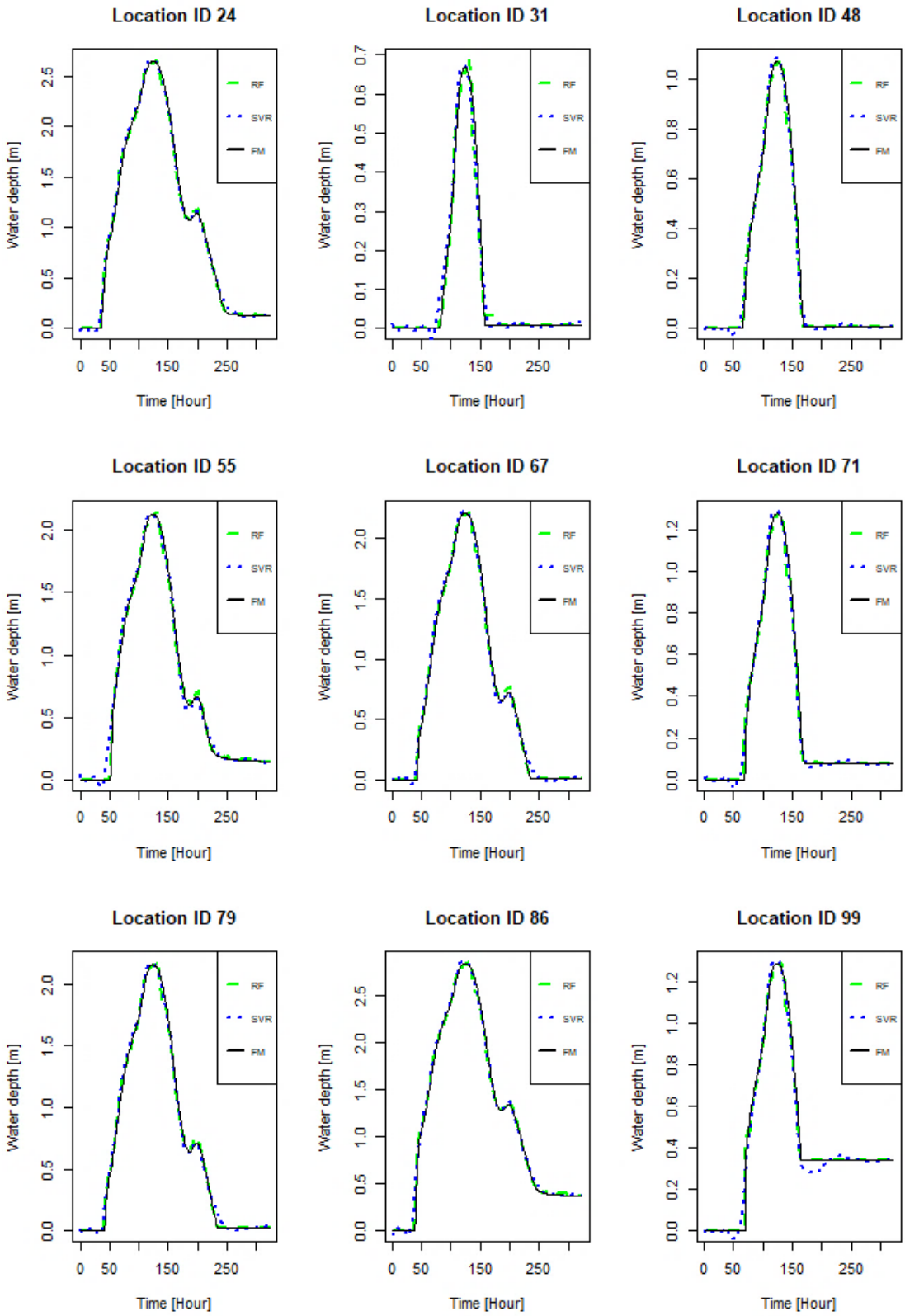


Figure 6.19: Comparison of RF and SVR simulated water depths against FM generated water depths on test data.

Performance of both models were also tested without using observation time as an input variable. The accuracy statistics of these models are shown in Table 6.8. The results deteriorated in both cases compared to the models which use time as an input. However, the RF outperformed the SVR by a small margin. This outcome is expected as the RF-based models are already producing better results during the initial approach, i.e. when we included time during the training process).

Location ID	RMSE_ RF	RMSE_ SVR	Rsqrd_ RF	Rsqrd_ SVR	NSE_ RF	NSE_ SVR
24	0.027	0.036	0.999	0.998	0.999	0.998
31	0.013	0.019	0.997	0.991	0.996	0.991
48	0.016	0.016	0.998	0.998	0.998	0.998
55	0.054	0.072	0.995	0.990	0.994	0.990
67	0.024	0.037	0.999	0.998	0.999	0.998
71	0.024	0.032	0.997	0.995	0.996	0.994
79	0.026	0.037	0.999	0.998	0.999	0.998
86	0.034	0.074	0.999	0.994	0.999	0.994
99	0.043	0.091	0.987	0.947	0.987	0.940
100	0.018	0.017	0.997	0.998	0.997	0.997
Avg.	**0.028**	**0.043**	**0.997**	**0.991**	**0.996**	**0.990**

Table 6.8: Error statistics of SVR and RF-based models without observation time.

The key result to notice is that when time was not taken into account, the SVR simulated flows became more unstable while, the RF stayed somewhat stable for most of the cases, at least visually (**Fig.** 6.20). A rather heavy fluctuation in the outputs from the RF-based models at locations ID 86 and 99 was observed during the low flows. This low flow variabilities can be identified as the dependency of water depth at a specific location on the flow observation time at the sections (MC010 and MC033).

Since in both cases, SVR-based models dominated the RF-based models, the RF technique was selected as the potential candidate for regression-based flood extent modelling.

After the compatibility test of the regression models, a separate RF-based model with the same model parameters was trained at the same sampled locations on which the MLP based models were trained and tested. The continuous values generated by the RF-based models were converted to binary data and then the overall accuracy, and the *F1*-score were computed for both cases, i.e. models that included the time variable and models that

excluded the time variable during the training process. The accuracy values (%) from both modelling approaches appeared to be on the higher side (Table 6.9), indicating that both models performed reasonably well. However, the RF yielded overall accuracy, and *F1*-scores, both fell fractionally behind when compared to the MLP yielded values. The initial MLP classifier scored an overall accuracy of about 0.992, whereas the RF approach scored 0.981. The MLP approach also produced a slightly higher *F1*- score, i.e. 0.986.

Model performance with 'time' variable		**Model performance without 'time' variable**	
Overall Accuracy	Overall *F1-score*	Overall Accuracy	Overall *F1-score*
0.981	0.973	0.975	0.971

Table 6.9: Accuracy test results for the RF-based model.

The overall accuracy and *F1*- score can be used to draw a consensus about the model performances. However, from these scores, it was not possible to infer model robustness in predicting flooded cells with more confidence. To extract further insights, just like the MLP approach, descriptive statistics from the RF predicted results and a 95% CI were computed and presented in Table 6.10. Since the overall accuracy and *F1*-scores for RF-based model were smaller than those from the MLP-based models, the RF based approach was expected to have a larger uncertainty range. At two locations, the actual flood arrived 6 hours earlier than the prediction, yet apart from that, the prediction time difference was between -2 to +4 hours.

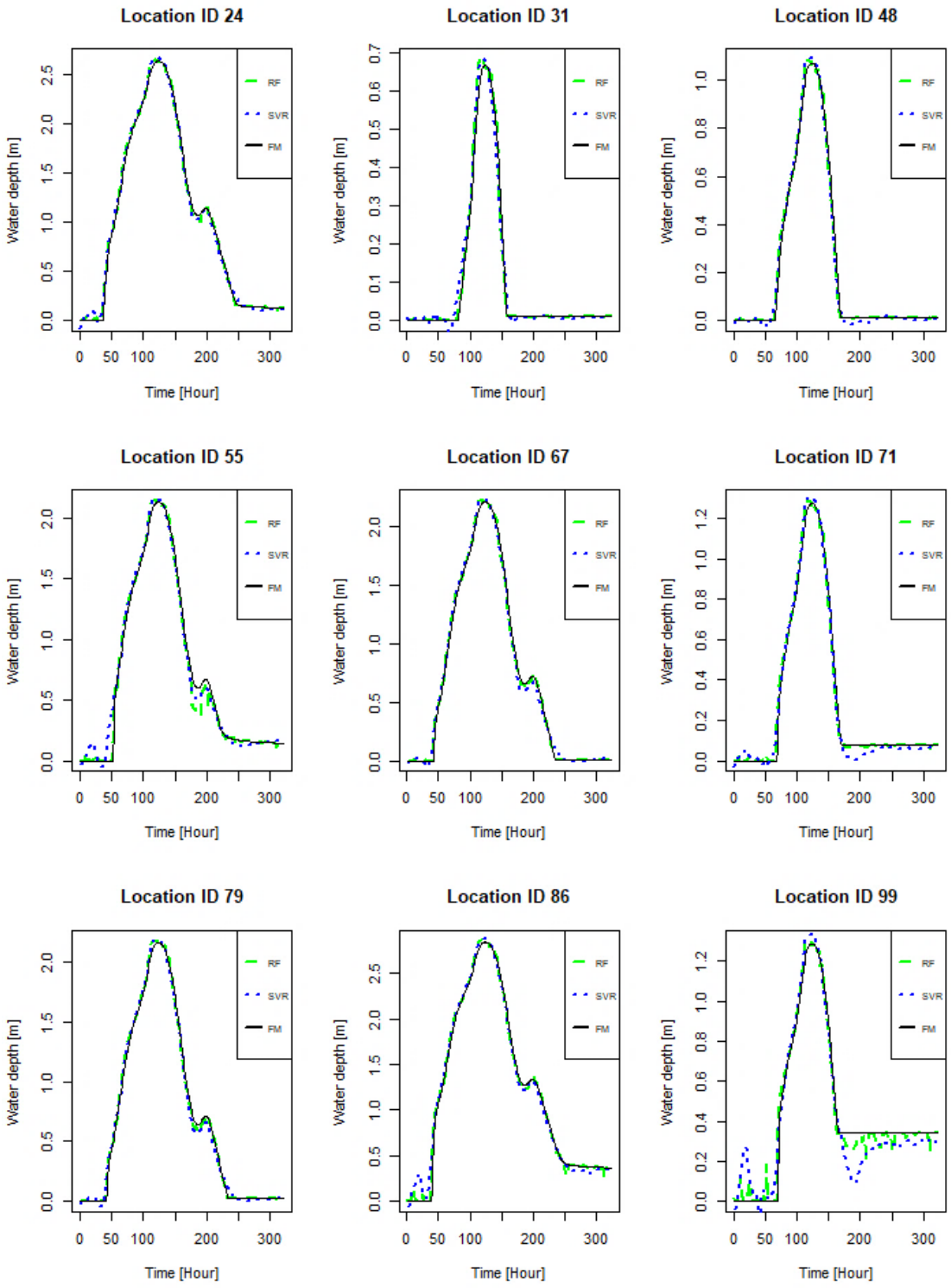


Figure 6.20: Comparison of RF and SVR simulated water depths against FM generated water depths on test data when observation time was excluded during training phase.

However, the mean time difference was about +39.6 minutes, thus on average there could be up to 39.6 minutes delay in flood arrival time prediction. This means that the RF-based models tend to underestimate the number of flooded cells during the transition periods.

An expected flood arrival time between -1 h to +4 h 24 min could be drawn with 95% confidence.

Mean	Median	Min	Max	1st Quartile	3rd Quartile	95% CI
+39.6min	0h	-2h	6h	0h	1h	[-1h, +4h24m]

Table 6.10: Descriptive statistics of RF predicted flood arrival time at sampled locations.

Descriptive statistics were also computed for the RF-based models trained without the time variable (RF-2) (Table 6.11). The results did not vary much from the outcomes of MLP-2 models. The 90% CI shows the high level of uncertainty involved in the modelling approach.

Mean	Median	Min	Max	1st Quartile	3rd Quartile	90% CI
-7h12min	0h	-88h	4h	-1h	1h	[-39h24m, +2h24m]

Table 6.11: Descriptive statistics of predicted flood arrival time at sampled locations from RF-2.

For further visual inspection of the residual distribution, histograms of both the models are presented in **Fig.** 6.21. The residuals from the initial RF method were distributed over a compact space near a mean of 0, whereas a spread between -20 to -50 hours was observed for the second approach.

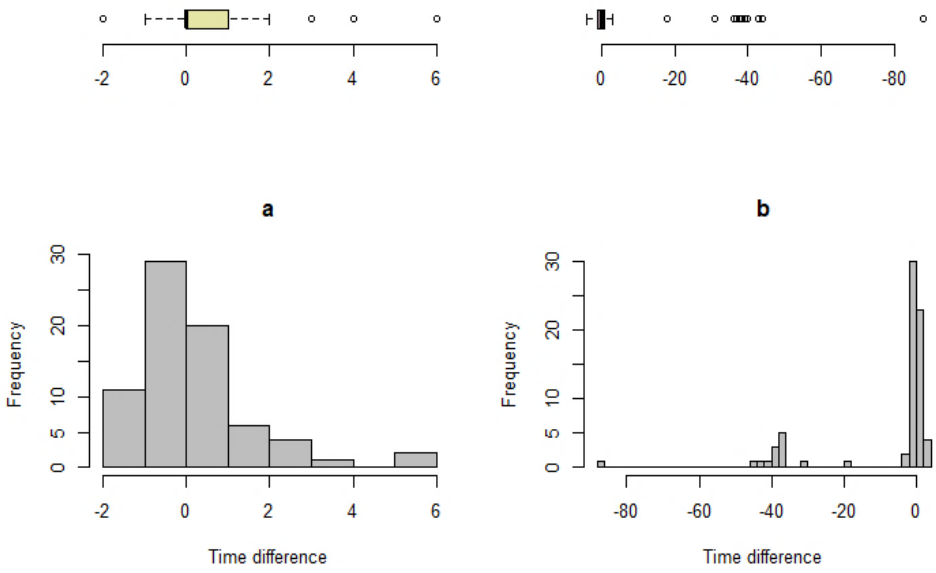


Figure 6.21: Histograms of RF (a) and RF-2 (b) showing residual dispersions.

Given the results yielded from the above quantitative analysis, it can be said that both MLP-2 and RF-2 models are unsuitable for real-time flood extent modelling due to the high level of uncertainties associated with them. However, both the MLP-1 and RF-1 models (models that included time as an input variable) showed promising results in predicting flood arrival times at the sampled locations. Therefore, results from both models were interpolated using the RK method and the interpolated maps were then compared against the FM simulated flood extent maps. The results are presented and discussed in the next sections.

6.12.3 Flood extent generation using RK

It can be computationally challenging to apply predictive models at each cell within the study domain. Therefore, a spatial interpolation technique was used to predict values at the unsampled points. The RK method was adopted to predict the presence/absence of water at the unsampled locations from sampled predictions using surface elevation as an auxiliary piece of information.

There were two sets of water presence/absence data available from two predictive models (MLP and RF) which could be used to train the RK model. The MLP data set at 77^{th} hour (the time when the cells were accumulating flood water) were used for calibrating RK

based interpolation model. This point of time was selected to ensure the samples were not too biased, as selecting an early time period could leave most of the samples dry, and during the flood peak could leave most of the samples flooded.

First, a variogram model was fitted to the sampled water presence/absence dataset. Widely applied exponential and spherical models were selected to fit the variogram (Hengl et al., 2007). The models were calibrated on 89 data points and tested on the remaining 10 locations. The spherical model had an accuracy of 80%, while the exponential model yielded 70%. Thus, the spherical model was chosen for fitting the variogram and then used to predict the values at the unsampled locations (**Fig.** 6.22c). The optimal spherical model parameters found are- nugget = 0.079, sill = 0.138, and range = 627.69m. These terminologies define the variance and the maximum distance between two points after which there is no spatial autocorrelation can be found. The auxiliary predictor, surface elevation, has a coefficient of determination (r-squared) value of 0.38 in the 89-point measurements and is statistically significant at a 0.001 probability level. A conditional density plot is shown in **Fig.** 6.22a, which provides a smoothed overview of how the presence of water (1s) and absence (0s) change across various levels of surface elevation. The correlation plot of elevation versus water presence/absence data is given in **Fig.** 6.22b. Both the conditional density plot and the correlation coefficient demonstrate the water level dependency on surface elevation.

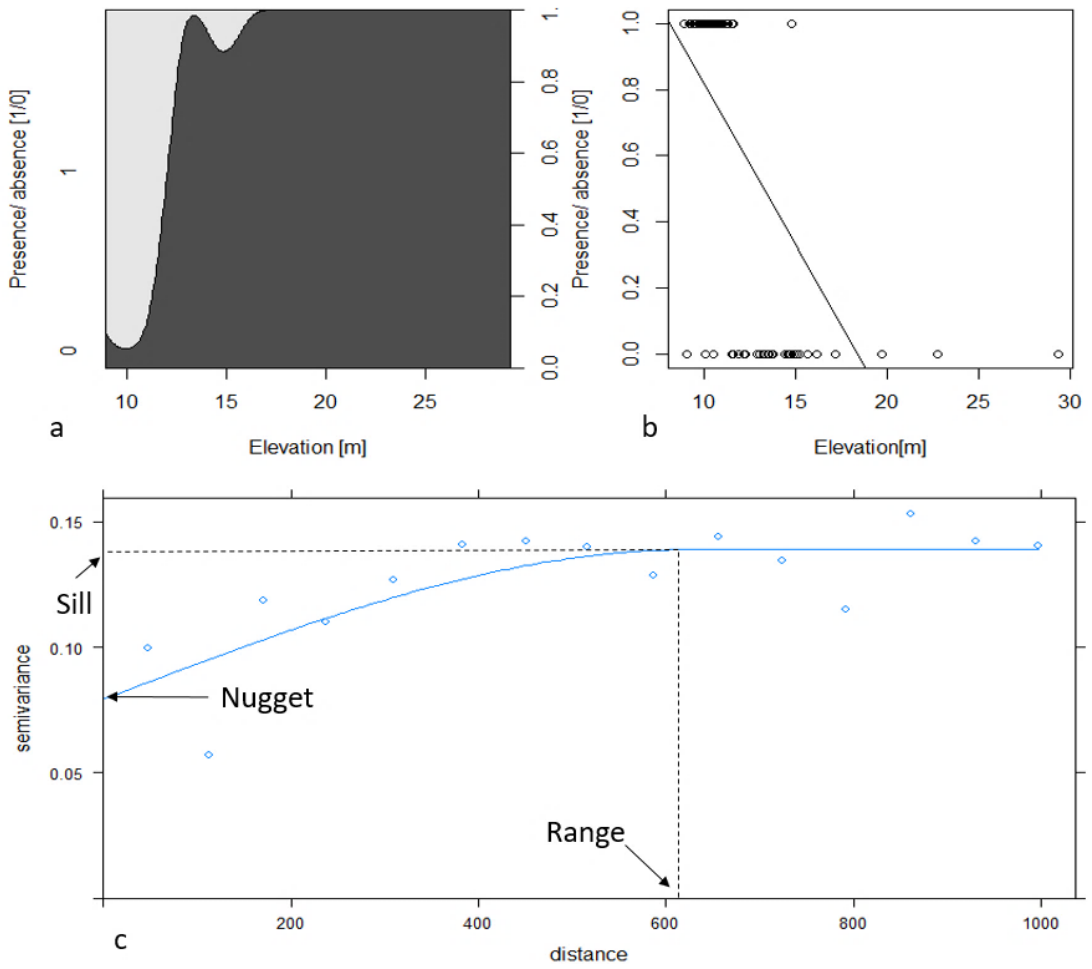


Figure 6.22: Surface elevation used to predict the presence of water: (a) conditional density plot, (b) correlation plot and (c) fitted variogram model.

The results of generating predictions from sampled point measurements with and without the auxiliary DEM can be seen in **Fig.** 6.23. When the DEM was used, the predicted water presence/absence map followed the surface elevation patterns and the geomorphology can be better presented than when pure ordinary kriging is used. The ordinary kriging output is much noisier than the RK output. The uncertainty level can also be observed from the variance plots produced based on the predictions.

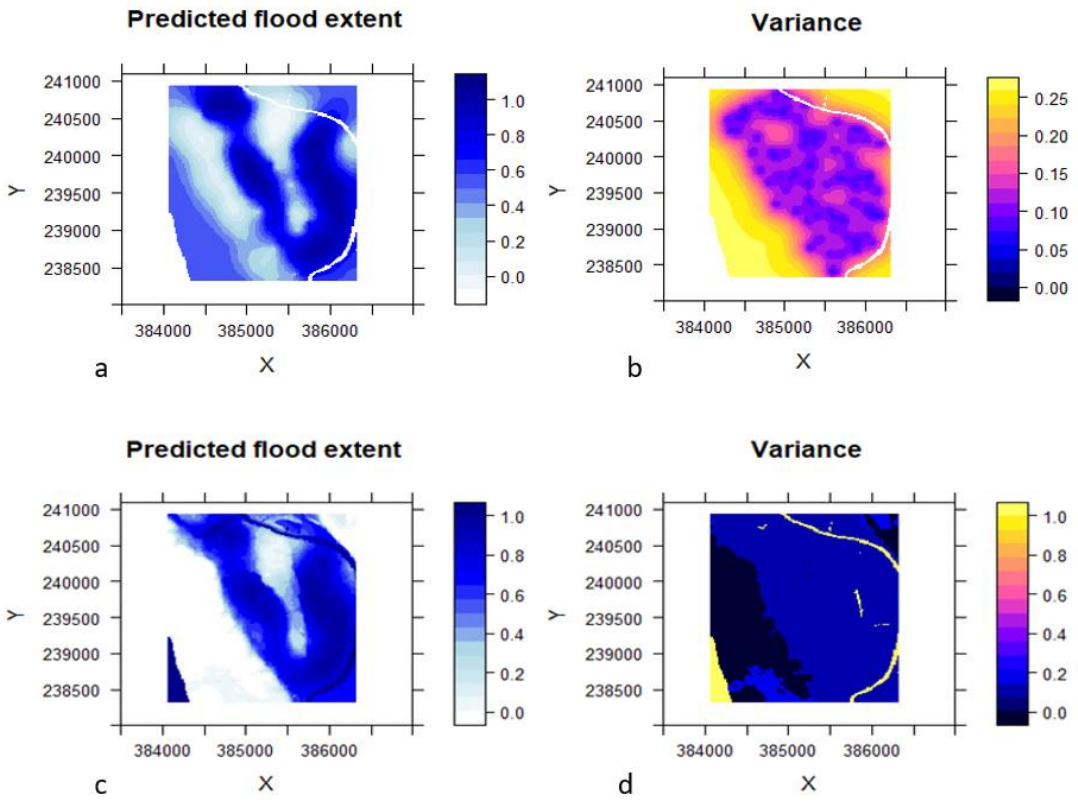


Figure 7.23: Predicted flood extent maps. (a) Ordinary kriging predictions, (b) Associated variance with ordinary kriged map, (c) RK predictions, and (d) Associated variance with RK predicted map.

After exploratory RK analysis, the fitted variogram was used to simulate flood extent maps for the entire duration of the flood event. The probabilistic maps were simulated at every hour from both predicted MLP and RF data sets. These probabilistic maps were then converted to binary values using a 0.5 cut-off value. Note that the simulations cover a rectangular area which is the size of the DEM image used during the kriging process. The extent is larger than the actual extent of the active area which was used during FM simulation. The DEM also has a different cell resolution (18m) than the resolution used during the FM simulation (30m). Therefore, these simulated flood extent maps were remapped to FM map extents. This operation was done using *gdal* library in *Python*:

```
INPFILE=/MLP_RK_Hour_77.tif

OUTFILE= /MLP_RK_Remapped_Hour_77.tif

MINLON=384076

MINLAT=238315

MAXLON=386323
```

```
MAXLAT=240935

# Set resolution to correspond to 30 m

RESOL=30

# Perform remapping

gdalwarp -s_srs EPSG:27700 -t_srs EPSG:27700 -tr $RESOL $RESOL -r ave
rage -overwrite -dstnodata -9999 -te $MINLON $MINLAT $MAXLON $MAXLAT
$INPFILE $OUTFILE
```

These remapped images had the same extent and resolution as the FM generated flood extent maps. Pixel level comparisons were carried out by computing a confusion matrix and F-score using **Eq.** 5.1. In total, 644 extent maps were produced, 322 each for MLP and RF. From these, three maps were selected from each modelling approach that represented three different stages of flooding for mutual comparison. The first comparison was made at hour 46, when the floodplain started to accumulate water. The second comparison was made when the flood was at its peak (163rd hours), and the final comparison was made at a time when the flood water started receding (324th hours). Maps produced using data sets from MLP and RF were compared separately against FM. Table 6.12 presents the confusion matrices and F-scores at hours 46, 163 and 324 for the MLP.

H-046	**MLP**		**F-score (%)**	**H-163**	**MLP**		**F-score (%)**	**H-324**	**MLP**		**F-score (%)**
FM	**0**	**1**		**FM**	**0**	**1**		**FM**	**0**	**1**	
0	7	43	79.5	**0**	8	34	98.1	**0**	154	61	56.9
1	96	538		**1**	13	2438		**1**	420	587	

Table 7.12: MLP vs FM comparison - confusion matrices and F-scores.

When the floodplain started to accumulate water in the west bank, about 96 cells were misclassified as dry by the MLP. The prediction mostly disagreed on the top corner of the east part of the cluster (**Fig.** 6.24a and 6.24b). This is probably caused by the small number of interpolating samples in this particular region (ID 38 and 50) and only ID 50 was predicted as flooded by the MLP. At the same time, 43 dry cells were classified as flooded. However, these observations could be attributed to the fact that FM followed flow routing to estimate the water level, while the interpolated map filled up unsampled cells using spatial autocorrelation approach. The *F* score of 79.5% showed a good agreement between these two maps. The RF based extent map exhibited a similar nature

of flood clustering (**Fig.** 6.24c), but the overall performance was lower than the MLP with an *F* score of only 68.7% (Table 6.13).

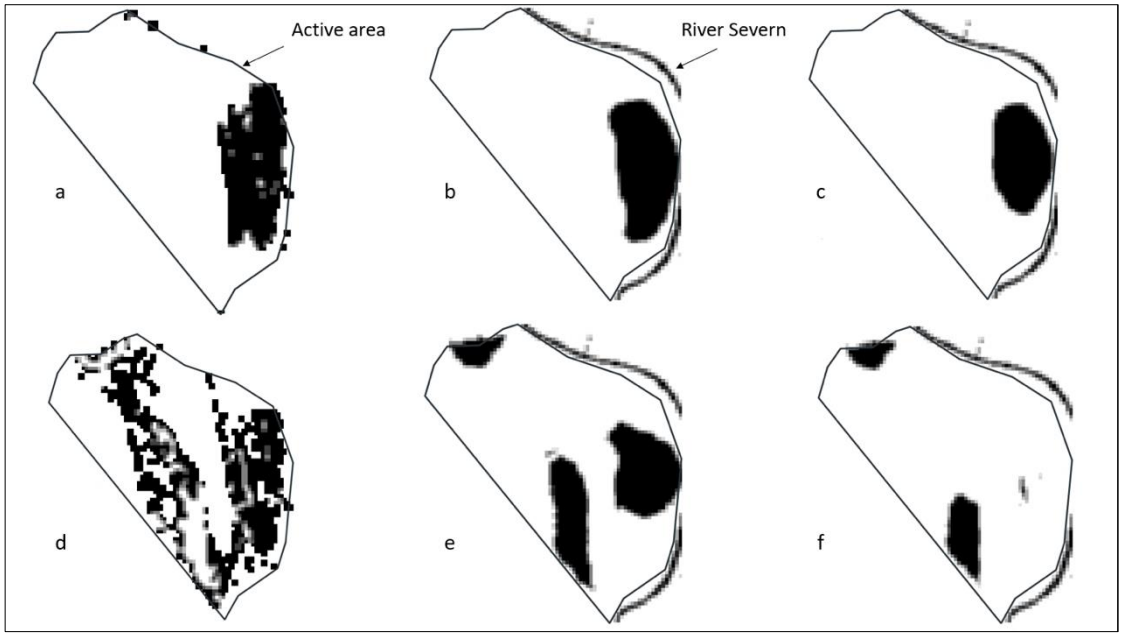


Figure 6.24: Comparison of flood extents during early stage (row 1) and receding stage (row 2)- (a) FM, (b)MLP, (c)RF, (d)FM, (e) MLP and (f) RF.

H-046	**RF**		**F-score (%)**	**H-163**	**RF**		**F-score (%)**	**H-324**	**RF**		**F-score (%)**
FM	**0**	**1**		**FM**	**0**	**1**		**FM**	**0**	**1**	
0	25	25	68.7	**0**	12	30	97.4	**0**	210	5	27.8
1	181	453		**1**	33	2418		**1**	725	282	

Table 6.13: RF vs FM comparison, confusion matrices and F-scores.

Contrarily, when we examine the flood receding phase, there is a severe disagreement between the FM simulated extent and the predicted extents. The *F*-score drastically went down for both MLP and RF. This is partly due to threshold values set for binary classification. A cut-off value of 0.5 was used to convert interpolated probabilistic maps to binary maps for all the three phases. The FM generated deterministic maps were converted to binary maps by setting any values greater than 0.09m to 1. This led to a significant difference during the flood growing and receding period. At this point a vast number of pixels had very small water depth values, while during flood peaks the depth values were larger. Cells with very low depth values were classified as wet cells in the

reference FM map, whereas the same cells were assigned lesser probabilities (< 0.5) by the interpolation method and were labelled as dry cells. Possibly, to this issue, it is better to use a separate cut-off value during different phases of a flood event or simply use probabilistic maps to delineate flood risk zones.

Another potential reason of this severe disagreement between the modelled inundation extents during the flood recession period could be the effect of hysteresis. This is a phenomenon where the response of a system/floodplain to the external influence (e.g. inflow variations, backwater effects etc.) depends not only on the present magnitude but also on the history (Kumar, 2011). The response of the floodplain dynamics to the external influence is a two-valued problem, where one applies during the phase when the influence (e.g. inflow) is increasing and another is when the influence is decreasing. That is, for the same inflow magnitude, the response of the floodplain would be different during flood growing and receding periods. It is beyond the scope of this research to investigate the effect of such phenomenon and should be evaluated in future research.

The effect of ground samples covering the floodplain and the true state of the domain is evident in the maps produced during the peak flood phase. The samples that are situated at the lower elevation were flooded, and the remaining samples were dry. This led the RK to predict the presence of water at the unsampled locations using the surface elevation information more precisely. The FM generated extent map during the peak flow is illustrated in Figure 6.24.

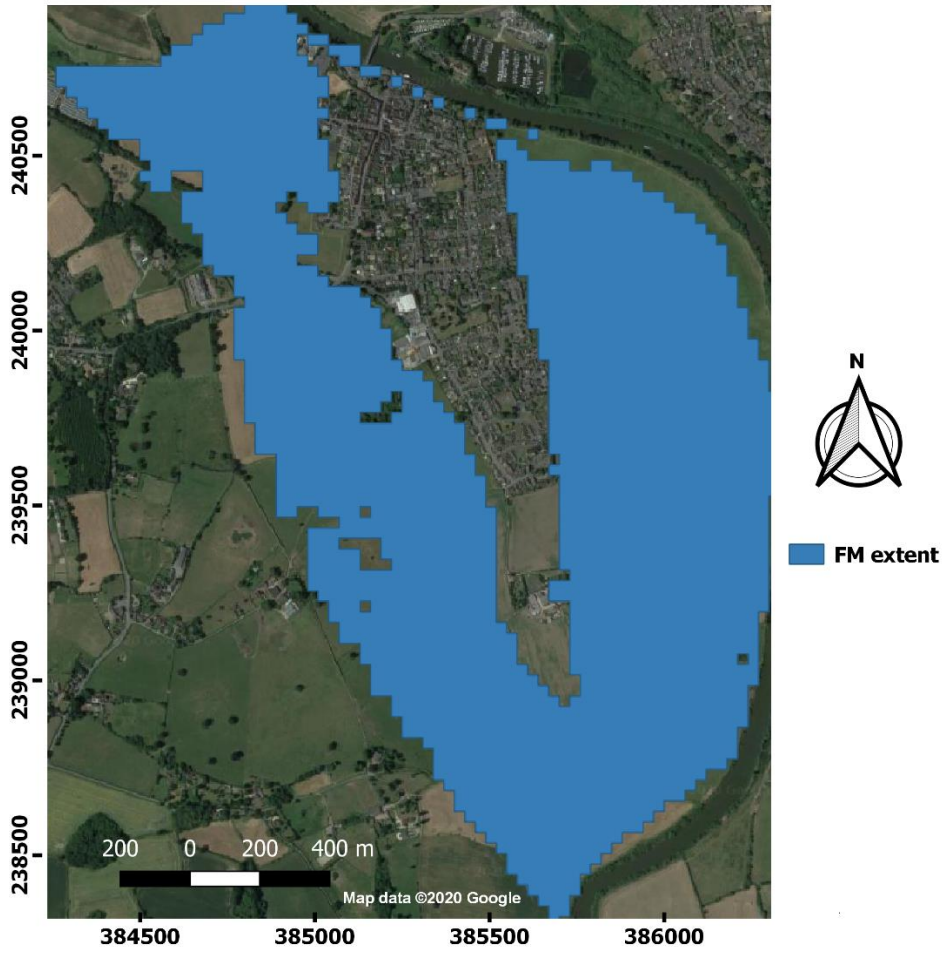


Figure 6.25: Flood extent map from FM during flood peak.

The predicted flood extents from the RK method using MLP and RF data sets are given in Figure 6.26. The predicted maps can capture the true state of the floodplain with a very little discrepancy. Interestingly, both the maps yielded high *F* scores, 98.1% for MLP and 97.4% for RF.

Figure 6.26: Predicted flood extent maps during flood peak - (a) MLP and (b) RF.

The MLP based models seemed to predict wet cells relatively better than the RF-based models, whereas the RF-based models predicted dry cells fractionally better. However, this has been already noted in the Niger case study (see Chapter 5) where the probabilistic classifier appeared to perform better than the regression-based method during the end of the dry season. This means that regression-based models are more prone to underestimation during the low flows. The extents are overlaid in Figure 6.27 for a relative comparison. This shows that QRF and MLP are almost identical and match closely to the reference FM derived extent.

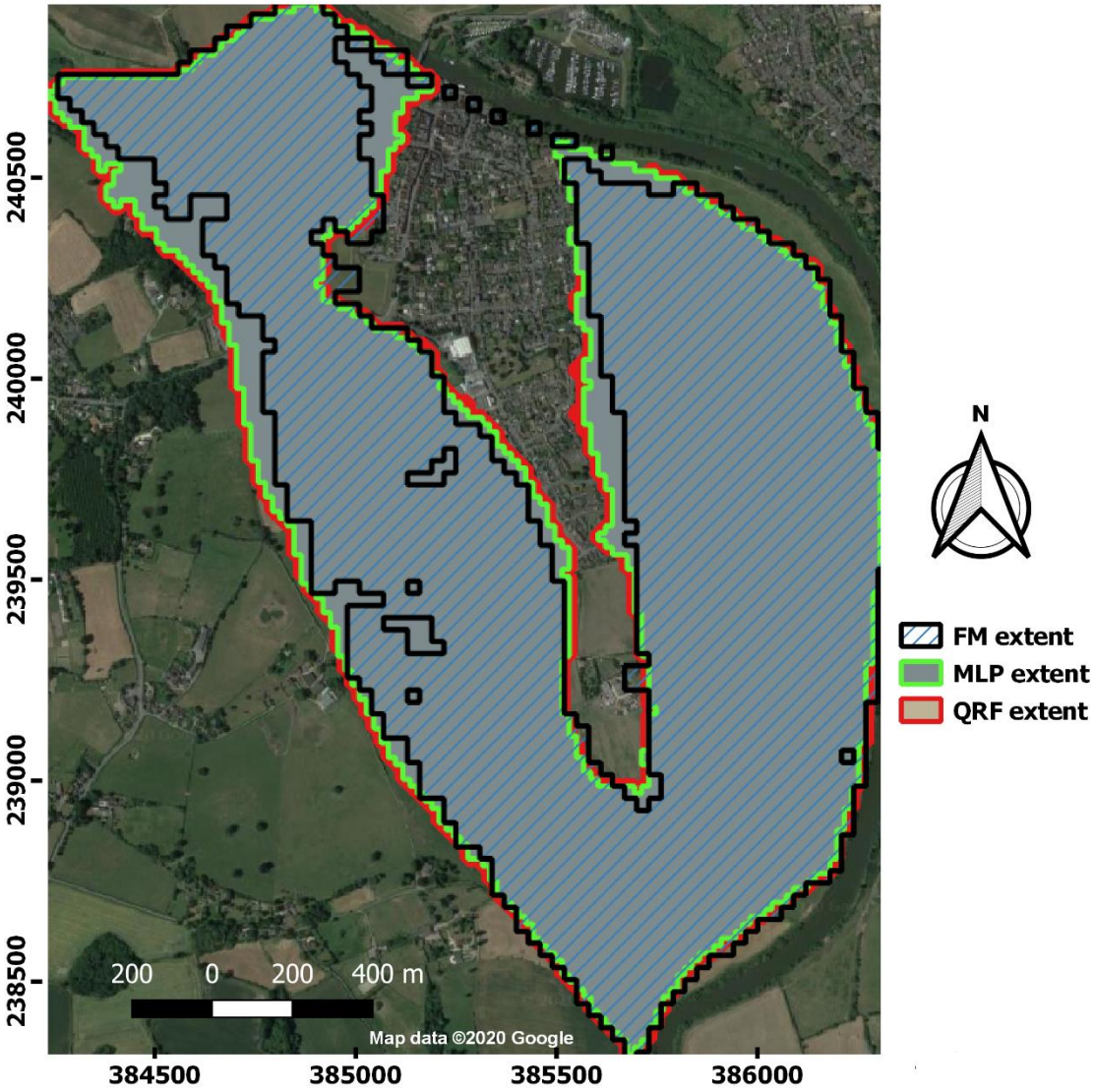


Figure 6.27: Flood extent maps produced from MLP, QRF and FM.

6.13 Conclusion

It can be said from this study, with confidence, that a simple ML based classifier can be used to predict flood arrival time at sparse locations within a floodplain and to subsequently predict the extent of the possible flooded area. Since the probability of a pixel being flooded is correlated with the surface elevation, this feature can be exploited to generate estimations without training computational models for the entire domain.

In this study, the RK model was developed using flood stage data from the 77th hour, which was believed to be covering the wet and dry cells well enough. However, results showed that during low flows, the samples did not represent the true state of the domain.

These biased samples produced unreliable maps during low flow periods. Therefore, it is suggested that in the future studies, samples should be selected more carefully, and rigorous normality test should be carried out before applying the kriging technique.

This study aims to emulate a 2D hydraulic model output using ML techniques where the spatial and temporal resolution are both high. The research showed the potentials of ML techniques in inundation modelling, which could potentially enhance our capabilities for real-time flood extent mapping. In the present study, one of the major challenges dealing with 2D modelling is that the FM was not calibrated to reflect the real case. It was assumed that if the ML based models could replicate the outputs from the FM, then it should be able to reproduce outputs from a calibrated model. Unfortunately, precise calibration work could not be carried out because of several constraining factors such as availability of catchment morphological data, structural information, time limitations, and expert knowledge, but this could be the focus of future studies.

To train an ML-based model, outputs from the hydraulic model (multiple hydrographs) are required. Here, a set of synthetic flow series was generated without considering the return periods. This is simply because of the lack of available historical data. Ideally, hydrographs of 1: N (where, N = 2, 5, 10, 50, 100, 200) year return period should be used for simulating flood extent and subsequent training processes. This will encapsulate a wide range of flood flow patterns enabling the models being less susceptible to overfitting, resulting in better forecasting. In other words, if the models are trained with hydrographs that represent only 2,5,10 year return period events then it is very unlikely that the models will be able to correctly classify for floods that are of over 10 year return period.

Similar to the previous chapter, regression-based methods (SVR and RF) were tested alongside a simple classifier (MLP). Depths at large number of sampled locations were estimated unlike only four in the previous chapter. Both the SVR and RF-based models reconfirmed that regression-based methods can be efficiently used to predict water depths as well as extents as a function of upstream discharge and discharge observation time. However, as mentioned before, since the objective of the thesis is to simulate dynamic flood inundation maps, therefore the estimated depth values were converted to binary maps to compare with the simple MLP-based approach.

Next chapter will present, an integrated modelling framework to predict flood extent from forecasted rainfall data. Two ML-based hydrological models at section MC010 and

MC033 will be used to set up the input variables for the MLP classifier. Also, the total number of samples will be increased to capture the true state of the domain during low flows.

Chapter 7 A Framework for Real-Time Rainfall-Inundation Modelling: A Machine Learning Approach

7.1 Introduction

As described in the introduction (Chapter 1), 2-D hydraulic models are heavily constrained in real-time flood inundation forecasting, raising the need for a new computationally efficient modelling engine which could effectively forecast inundation extent with considerable lead time. To reduce the computational stress and to develop a new innovative method, as an alternative for real-time flood inundation modelling, this chapter extends the work presented in the previous chapters. In the earlier chapters, the potential applicability of ML techniques is demonstrated for forecasting:

i) Multi-hour ahead streamflow from rainfall data, referred to as '*rainfall-streamflow'* modelling (Chapter 4).

ii) Inundation extent from streamflow measurements in the context of fluvial flooding, referred to as '*streamflow-inundation'* modelling (Chapter 5 and 6).

This chapter is focused on developing an integrated framework that combines both the methods, rainfall-streamflow and streamflow-inundation to generate dynamic flood inundation maps directly from the rainfall data. The underpinning idea is that a non-linear regression-based ML-based model can be applied to forecast multi-step ahead streamflow at the upstream and then the forecasted flows can be used to generate flood inundation maps. Conceptually, the framework can be viewed as a 1D-2D linked model, where the initial part (regression) of the system acts as a 1D model and output from the model is turned into a geospatially varied extent map (2D).

The proposed framework integrates a non-linear regression based hydrological model, a computationally efficient wet/dry cell classification module, and a data visualisation schematic. High-resolution data visualisation is achieved by applying a spatial interpolation technique, which uses auxiliary information available from a high-resolution DEM. The research activity is focused on the town of Upton-upon-Severn, which is situated on the west bank of the River Severn in the Malvern Hills District of

Worcestershire, England. The analysis is done for the time frame covering the flooding event of October-November 2000.

7.2 Methods

7.2.1 Framework for data driven flood inundation mapping

This section presents a detailed description of the proposed data-driven modelling framework for generating real-time flood inundation maps from rainfall and upstream flow data. The modelling framework consists of two core components (**Fig.** 7.1):

A. developing the *pre-trained/fitted model databases*: for storing trained ML-based models (hydrological models, hydraulic classifiers); and

B. generating *real-time flood inundation maps*: generated using discharge forecasts.

A. Developing the pre-trained/fitted model databases

In the first step, the regression-based three hours ahead flow forecasting models are trained. These models are trained using antecedent rainfall and flow data. Secondly, the MLP-based models that predict the states (wet/dry) of the sampled cells, i.e. 150 Ground Control Points (GCPs), within the floodplain are trained. These classifiers are trained using the outputs from FM at a coarse grid resolution (30m in this study) as the target and upstream flow as the input variables. One of the major benefits of using MLP is that the model can generate multiple outputs for a specified input dataset. That is, a single MLP classifier (i.e. a global MLP) can classify states (wet/dry) of all the GCPs for a given upstream boundary condition at a time. However, in this study, a separate model is trained for each GCPs (local MLP) instead of a global model. This is because an initial investigation showed that overall results obtained using local MLPs were slightly better than the outputs from a global MLP. However, a global model can be used for large scale modelling as it reduces model training time significantly without a major drop in accuracy. The processes of training the regression and MLP-based models are further described in Section 7.5 and 7.6. The validated flow forecasting and classifier models are stored in the databases for future applications.

B. Generating real-time flood inundation maps

In the third step, for real-time operation, the pre-trained regression-based models are loaded for forecasting three-hour ahead upstream flows using observed rainfall and

discharge. That is, the models forecast the discharge at the upstream with three-hour lead time for every new observation. During a flood event, the forecasted discharge crosses a certain threshold which loads the MLP-based cell classifiers. These models predict the states of the GCPs (i.e. flooded/dry).

Finally, these classified cells (wet/dry) are then used to predict the state of the unsampled cells using Regression Kriging (RK) method (Hengl et al., 2007). The so-called RK is a spatial interpolation technique that uses distance as well as other auxiliary information between sampled points to predict the values for the unsampled points. In this study, surface elevation is used as auxiliary data to predict states of the unsampled cells. The final outcomes are moderately higher resolution (9m) interpolated maps with three-hour lead time which are sequentially generated for every new rainfall and flow observations (hourly). The maps could be made available through 'WebGIS' for real-time decision making.

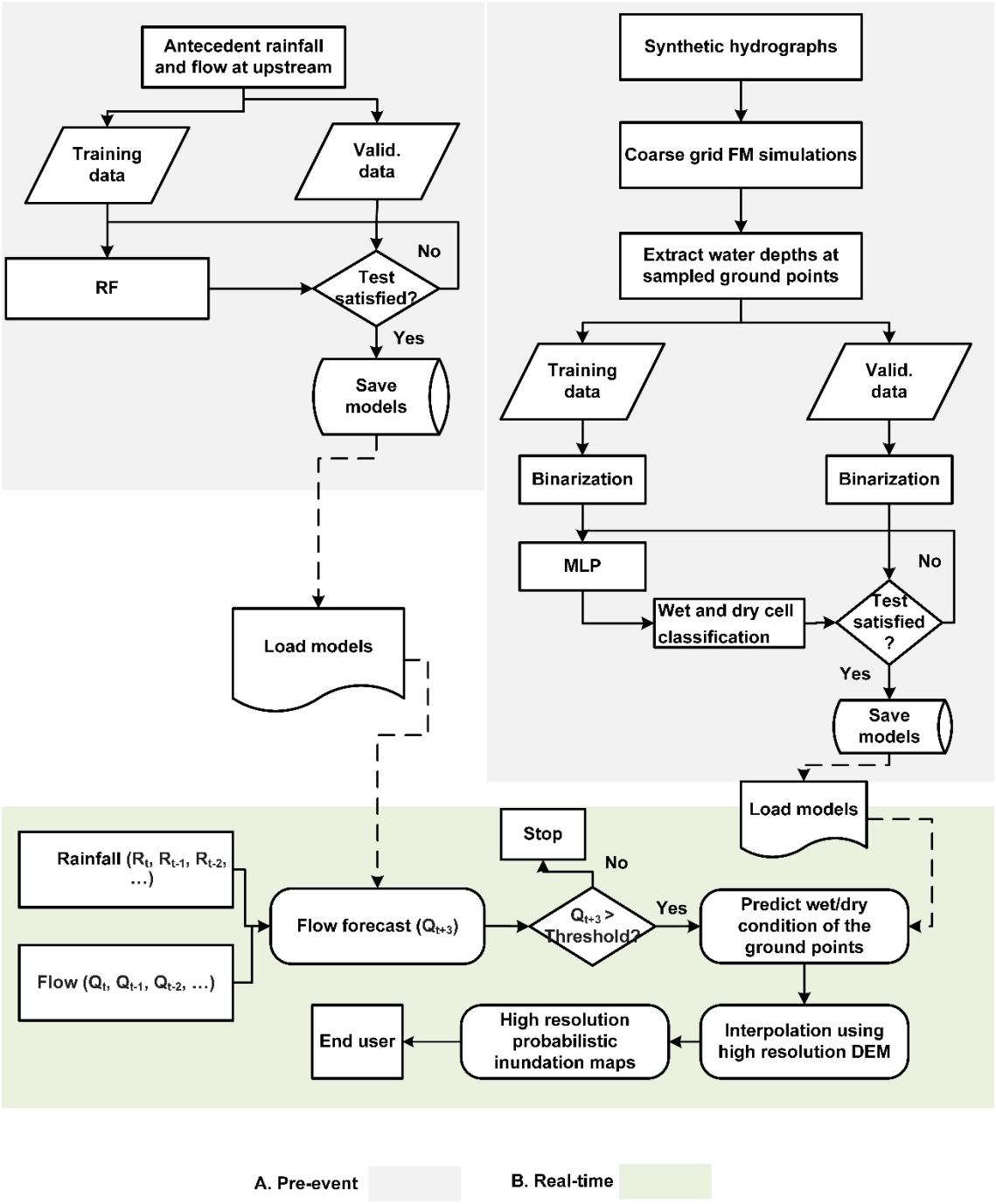


Figure 7.1: Framework for real-time flood inundation forecast.

7.2.2 Evaluation metrices

Recalling from Chapter 4, the SVR-based models were demonstrated to perform better than WANN and DBN-based models for short lead time flow forecasting. However, recalling from Chapter 6, RF-based models appear to perform better than the SVR-based models for water depth prediction. Thus, to develop the framework, the RF-based models

were used along with the SVR-based models for generating 3 hours ahead flow forecasting (please refer to **Sect.** 7.5 for additional justification on selecting the RF technique for flow forecasting). To assess robustness and reliability of the modelling framework, forecasted results generated from the integrated framework are compared using a range of goodness of fit statistics, as listed in Table 7.1.

Metrices	Application	Comments
RMSE NSE R-squared	To detect the best fit model for 3 hours ahead flow forecasting.	Please refer to **Sect.** 3.2 for definitions of the equations.
Accuracy, *F-1* score	Ability of the classifier to detect state changes of a cell.	Please refer to **Sect.** 6.11.1 for more details.
Mean, 95% Confidence interval (CI)	Mean error in predicting flood arrival time and 95% confidence limits.	Descriptive statistics of the residuals show the classifiers error range.
F-score	Classification accuracy at each time step. It shows the percentage of sampled cells, whose true state is correctly predicted by the model for every hour and throughout the flood event.	Please refer to step 6 in **Sect.** 5.4.2 for more details on contingency tables and calculating F-scores.

Table 7.1: Model evaluation metrics used in this study.

To quantify the uncertainty in flood arrival times, the residual variation was analysed. The residual are the predicted time differences between the FM and ML classifiers at sampled locations in the first arrival of the flood wave.

Prediction of flood water presence/absence at sampled pixels was spatially interpolated using a pre-modelled variogram to produce high resolution inundation maps. These maps were visually compared with the reference deterministic maps produced by the FM simulations. For comparison, three maps are presented covering the flood growing, flood saturation and flood receding periods.

Finally, the classification accuracy between FM and the proposed data-driven method is compared against a reference satellite image captured on 8 November 2000.

7.3 Study area and data pre-processing

The study domain covers an area of about 4.27 km^2 within the town of Upton-Upon-Severn. This is the same floodplain used to develop the ML based flood inundation modelling approach in Chapter 6.

To overcome the issue of under-sampling, leading to underestimation of wet cells during the flood receding period as observed in the previous chapter, the sample points were increased from 100 to 150 for this case-study. Figure 7.2 displays the spread of these 150 randomly selected sample points or ground control points (GCPs).

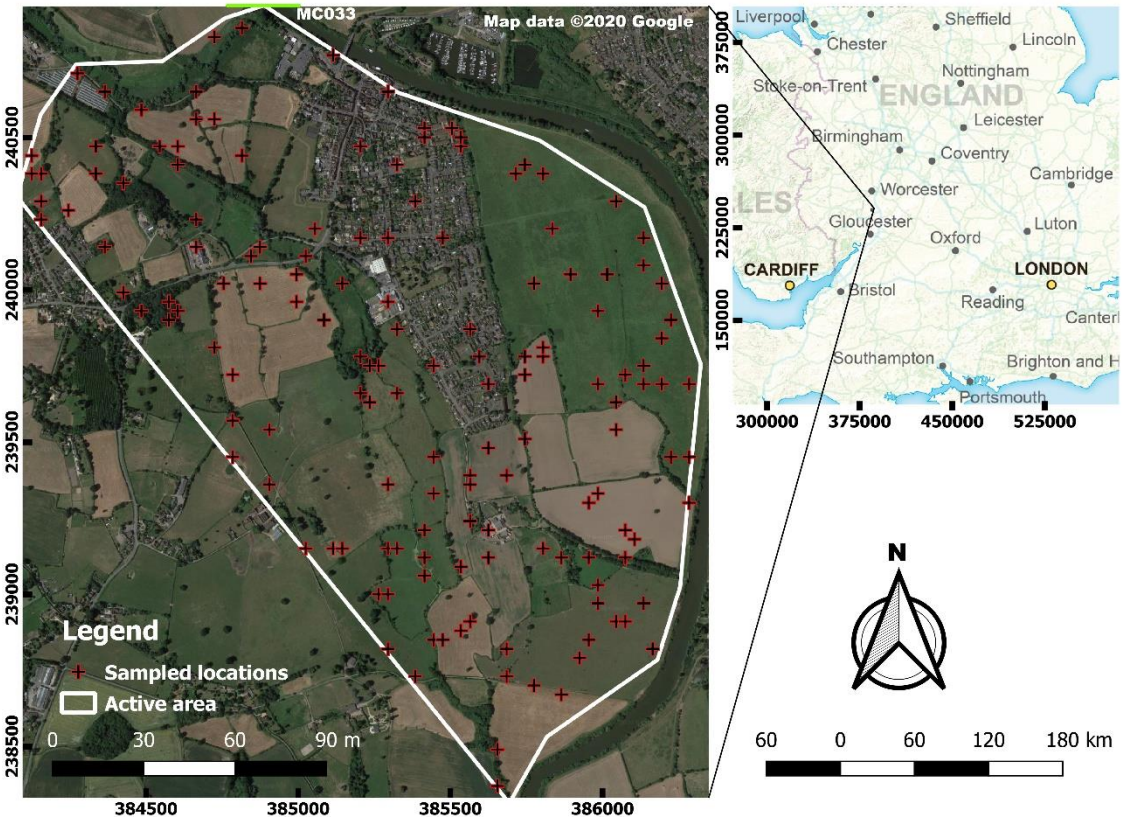


Figure 7.2: Study domain and 150 randomly sampled locations.

Recalling from Chapter 6, one subset (Subset 1) of observed hydrograph along with three synthetic hydrographs derived from observed hydrographs were used to train the model. Also, one subset (subset 2) of observed hydrograph was used to test model performance. However, in this study, only synthetic hydrographs (one more synthetic hydrograph was added to the existing three) were used to train the MLP classifiers and two subsets (subset 1 and subset 2) were used for testing forecasting accuracies of the model (**Fig.** 7.4). The overarching idea was to train the models using synthetic flow data and then to assess the

robustness of the integrated system by testing the model capabilities using two sets of observed data with varied peaks (subsets derived from observed inflow hydrographs). Figure 7.3A depicts the complete flow hydrograph used in the study, including corresponding observed catchment average rainfall. Figure 7.3B and 7.3C shows the subset hydrographs with corresponding rainfall total.

Hourly rainfall data were downloaded from the Met Office Integrated Data Archive System (MIDAS) for Great Malvern (src id 670) and Pershore (src id 657) meteorological stations. Hourly data from both the stations were aggregated and used as a proxy for the catchment average rainfall.

The synthetic hydrographs were generated using an HMM. For more explanation on synthetic flow generation see **Sect.** 6.5.2. Table 7.2 lists the areas of the model development covered by these six hydrographs (four synthetic and two subsetted) along with their peak values.

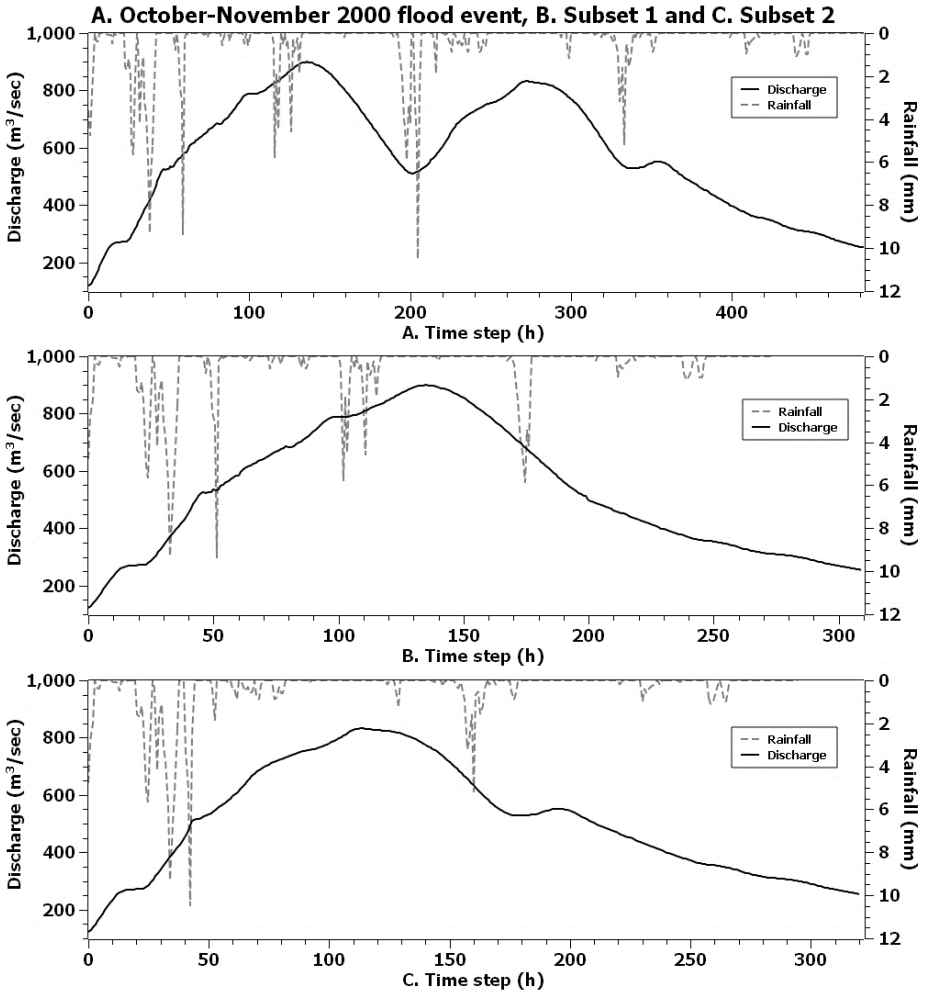


Figure 7.3: (A) Observed inflow hydrograph for total duration including catchment average rainfall (reverse x-axes), (B) subset 1 and (C) subset 2.

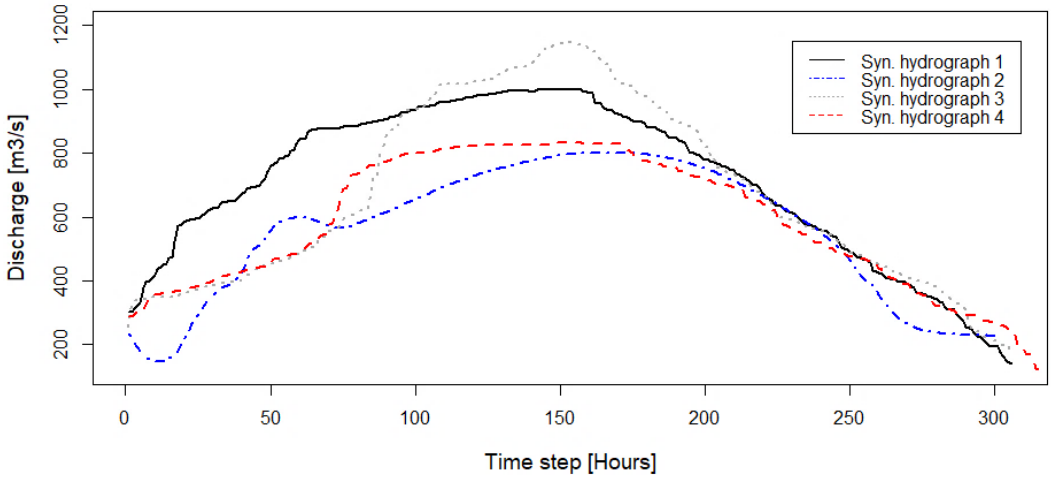


Figure 7.4: Synthetic hydrographs used to train MLP classifiers.

Index	Used for	Peak [m^3/s]
Syn. Hydrograph 1	Training MLP	1000
Syn. Hydrograph 2	Training MLP	802.1
Syn. Hydrograph 3	Training MLP	1150
Syn. Hydrograph 4	Training MLP	831
Subset 1	(1) Train and test SVR and RF for flow forecasting	899.2
	(2) Test MLP classifier	
Subset 2	(1) Train and test SVR and RF for flow forecasting	832.5
	(2) Test MLP classifier	

Table 7.2: Summary of hydro- and meteorological data used in the study.

7.4 Flood Modeller simulation

To generate the target data (water depth) for training and testing the MLP-based models, 1D- 2D linked models were applied. To keep computational time comparatively low, the original DEM of 3m resolution was resampled to 30m and larger time steps of 5 seconds were used.

The Alternating Direction Implicit (ADI) solver was selected due to its shorter runtime and robustness. In addition, it did not require any supercritical flow calculation. The

channel and floodplain roughness values used to run the model were 0.025 and 0.05, respectively. A step by step simulation process has already been described in Chapter 6 (see **Sect.** .6). Here, the simulation process was repeated and six separate 1D-2D linked model runs were performed with six hydrographs to estimate water depth values at 150 sampled locations. Water depth values from these sampled locations were then used to train and validate the MLP classifiers.

To visually compare the final interpolated probabilistic flood extent maps, deterministic flood inundation maps at 9m resolution were also generated for subset 1 and 2 hydrographs. These 9m inundation maps produced by the FM from subset 1 and subset 2 can be treated as the reference maps.

The idea is to train the MLP classifiers using coarse grid (30m) FM outputs and predict real-time flood maps at a higher resolution to compare the resulting maps. That is, to assess if the output of the ML-based models that were trained on coarse resolution (30 m) data can be used to generate predictions on a much higher resolution (9m), and if these predictions are comparable to the 9m inundation maps generated from the 2D hydrodynamic model. If this is possible then the fine grid 2D model runs are not required to generate training samples. In other words, time efficient coarse resolution data would be sufficient to train ML-based models to produce high resolution flood maps; this would further add to the benefits of data driven modelling techniques for real-time flood inundation mapping.

7.5 Inflow hydrograph forecasting

To find the most appropriate flow forecasting model, a rigorous analysis was conducted to compare the different ML based predictive algorithms, namely: WANN, SVR, and DBN for 1 to 6 hours ahead discharge forecasting (see **Sect.** 4.4). The results from the comparative study indicated that the SVR-based models perform significantly well for short lead time forecasting. Hence, the SVR was the initially chosen as the preferred method for 3 hours ahead flow forecasting at the upstream (upstream boundary conditions) for this study. It is, however, indeed possible to select other lead times as well, but 3 hours were assumed to be a good starting lead time considering the size of the study domain.

A two-fold cross validation approach was adopted here for testing the robustness of the forecasting model. This means that the forecast model was trained on subset 1, then tested

on subset 2. After that, the model was trained on subset 2 and tested on subset 1. However, there was a fundamental issue with the training data which consisted of only one peak for both subsets, and subset 1 had a higher peak value than subset 2. Therefore, the model trained on subset 2 could be overfitted, resulting in poor performance on the test set. To negotiate possible overfitting, the RF-based model was introduced as a forecasting model and compared against SVR-based model. RF uses the so-called bagging technique to reduce overfitting by bootstrapping training data and aggregating the results from individual trees to make decisions (Cutler, 2014). In the following sections, the SVR and RF-based modelling approaches will be described in detail.

7.5.1 SVR-based modelling

For SVR-based modelling, the *'Scikit-learn'* (Pedregosa et al., 2011) ML library in *Python* was used. Similar to Chapter 4, the nu-SVR type with RBF kernel was applied. The input-output structure of the model can be defined by the following equation:

$$Q_{t+3} = f[(Q_t, Q_{t-1}, Q_{t-2}), (R_t, R_{t-1}, R_{t-2}, R_{t-3})] \quad (7.1)$$

Where, target (output) variable Q_{t+3}, is the flow at t+3 hours; *f(.)* is the model function; Q_t is the current flow with its lags (in hours) and R_t is the rainfall with its lags (in hours). The total number of input variables was seven. For more details on the selection procedure for input variables please refer to **Sect.** 4.3.3. The rainfall values were up-scaled by multiplying by 100 before any training process (for both SVR and RF). This was done to reduce the effect of fractional values, i.e. to increase the margin between values for RF-based modelling.

Before fitting the nu-SVR-based model to the data, the input variables were transformed by the *StandardScaler()* function. This normalizes each column in the input matrix so that each feature will have a mean and standard deviation of 0 and 1, respectively. The nu-SVR-based model parameter values used to fit the models were*: C= 1000, nu= 0.5, gamma=0.7.*

In addition to the flow forecasting near the Severn Stoke station, i.e. the upstream section MC010, flows needed to be forecasted at section MC033 because flow values at MC010 and MC033 were used as inputs to the MLP classifiers. The upstream boundary of the catchment is located approximately 9 km (vertical distance) away from the study area. Therefore, to reduce the possible error caused by the distance and river morphology, the

flows were forecasted at two locations: at the upstream (Severn Stoke) location and at section MC033 (see **Fig.** 7.2) located just before the study area. As no recorded observed flow profiles was available at section MC033, a proxy flow time-series was used to train and validate the models for flow forecast at MC033. The proxy flow series was generated using a flood modeller (FM) 1D simulation. The model structure can be defined as:

$$Q_{MC033,t+3} = f\left[\left(Q_{MC033,t}, Q_{MC033,t-1}, Q_{MC033,t-2}\right), (R_t, R_{t-1}, R_{t-2}, R_{t-3})\right] \quad (7.2)$$

Where, Q_{MC033} is the flow values as generated by the FM 1D unsteady simulations. See **Sect.** 6.7 for more details on generating input data for the MLP-based model to classify wet/dry cells.

7.5.2 RF-based modelling

RF based models were used to forecast T+3h flows using total catchment rainfall (please refer to Cutler (2014) for detailed information on RF). The RF-based model for forecasting T+3h ahead flow was first developed and cross validated at upstream (Severn Stoke) and then a separate RF-based model was developed and cross validated for generating forecasts at T+3h ahead flow at section MC033. The input variables used in these models are $Q_t, Q_{t-1}, Q_{t-2}, R_t, R_{t-1}, R_{t-2}, R_{t-3}$, where Q_t is the observed flow with its lagged values (in hours), and R_t is the total catchment rainfall with its lagged values (in hours). The *skgarden* library in *Python* programming language has been used to configure the RF-based models.

7.6 Configuring MLP classifier

In the current study, the MLP-based models were trained at 150 sampled locations using four synthetic hydrographs following the steps described in **Sect.** 6.8. A summary of the model architecture and parameters is given below:

I. No. of total training input-output instances/samples: 1219
II. No. of input variables: 7
III. No. of nodes in input layer: 8 (7 for input variables and 1 bias node)
IV. No. of hidden layers: 1
V. No. of nodes in output layer: 1
VI. No. of nodes in hidden layer: 10
VII. Activation function: relu (input and hidden layer), sigmoid (output layer)
VIII. Loss function: binary crossentropy

IX. Optimizer algorithm = adam

X. No. training iterations: 100

XI. Batch size =20

7.7 Developing the pre-trained/fitted model databases

Two databases were developed, as described in Section 7.2, after training the regression-based models and MLP-based models. The database of hydrological models contained the trained best performing regression-based (RF/SVR) three hours ahead flow forecasting models and the database of hydraulic classifiers contained the trained MLP-based models that predict the states (wet/dry) of the GCPs within the floodplain.

7.8 Forecasting wet/dry pixels

To forecast inundated areas within the floodplain, first, T+3h ahead river discharges at upstream and MC033 were forecasted using the pre-trained best performing regression-based models. Then the T+3h flow values were fed into the pre-trained MLP-based classifiers to forecast 'wet/dry' state of the sampled locations. The probabilistic outputs generated by the MLP-based classifiers were converted into 1s and 0s. Cells having a probability of less than 0.5 were encoded as '0' and cells with values greater than 0.5 were encoded as '1'. To assess the robustness of the MLP-based classifiers, the classification task was conducted for both subset 1 and 2 and compared against the corresponding reference data (see results and analysis).

Thus, each of the sampled locations was assigned a binary value during the entire flood event at hourly time steps. Initially, all the sampled pixels were classified as dry and with the increase in flow rate pixels started to change their states. Finally, to predict the state of unsampled locations as the time progressed, the RK method was applied using surface elevation as auxiliary information.

7.9 Inundation map generation

Finally, following analogy to the previous chapters, the RK spatial interpolation technique was used to generate hourly flood inundation maps. Although the model processing steps remained same as in **Sect.** 6.10, some adjustments were made in this study to improve the results as suggested in Chapter 6, for example, selection of time stamp for fitting the variogram and variogram model. In this experiment, sample size was

increased to 150 points from previously used 100 points. Therefore, a new time step was searched at which the domain contained approximately equal number of wet and dry cells. Hence, states at 62nd hour were selected, based on the distribution of the total number of wet and dry samples on the floodplain. At this hour, the total numbers of dry and wet cells were 71 and 79, respectively.

In order to select the best fitting variogram model, a robust analysis was conducted using *Pykrig* module in *Python*. 80% of the total data points, were used to fit the RK model and the rest of the samples were used for validation. Five variogram models, specifically spherical, exponential, linear, power and Gaussian were tested. Best accuracy was found using the exponential model. The *Python* script for variogram selection can be found in the Appendix C. The results are given in the following:

```
========================================

vg model: spherical

Finished learning regression model

Finished kriging residuals

RK score:  0.8666666666666667

========================================

vg model: exponential

Finished learning regression model

Finished kriging residuals

RK score:  0.9

========================================

vg model: linear

Finished learning regression model

Finished kriging residuals

RK score:  0.8666666666666667

========================================
```

```
vg model: power

Finished learning regression model

Finished kriging residuals

RK score:  0.8666666666666667

========================================

vg model: gaussian

Finished learning regression model

Finished kriging residuals

RK score:  0.8
```

After the selection of the best variogram model, i.e. exponential, the remaining RK interpolation task was completed in *R*. The coefficient of determination (R-squared) was found to be 0.314 when the presence and absence of water were regressed against corresponding elevation values. Fitted variogram parameters of the *exponential* model were- nugget = 0.00, sill = 0.19, and range = 220.56m.

The fitted variogram was then applied to a high-resolution DEM (3m) to predict the presence/absence of water at unsampled locations. The results obtained were in the form of a probability value of flood occurrence throughout the study area based on 150 samples at 3m resolution.

7.10 Results and analysis

7.10.1 Inflow hydrograph forecast

The first step towards the framework development involved the fitting of a robust ML-based hydrological model to forecast flow at T+3h for upstream (MC010) and at the MC033 node. For this, both SVR and RF-based rainfall-discharge models were trained and tested. A two-fold cross validation method was adopted: Case 1- the models were trained on subset 1 and validated on subset 2; and Case 2- the models were trained on subset 2 and validated on subset 1. For a systematic quantification of forecasting accuracy, three error statistics were calculated (Table 7.3): i) The R-squared value – it compares the goodness of fit of the forecasting models; ii) The RMSE – it measures the

standard deviation of prediction errors; and iii) the NSE – it quantifies how well the model simulation predicts the output variable. For Case 1, high values for R-squared and NSE, and low values for RMSE indicate that models (both SVR and RF) were highly accurate in forecasting 3h ahead flows (SVR-based models outperformed the RF-based models by a small margin). The forecasted flows at node MC010 and MC033 are shown is Figure 7.5.

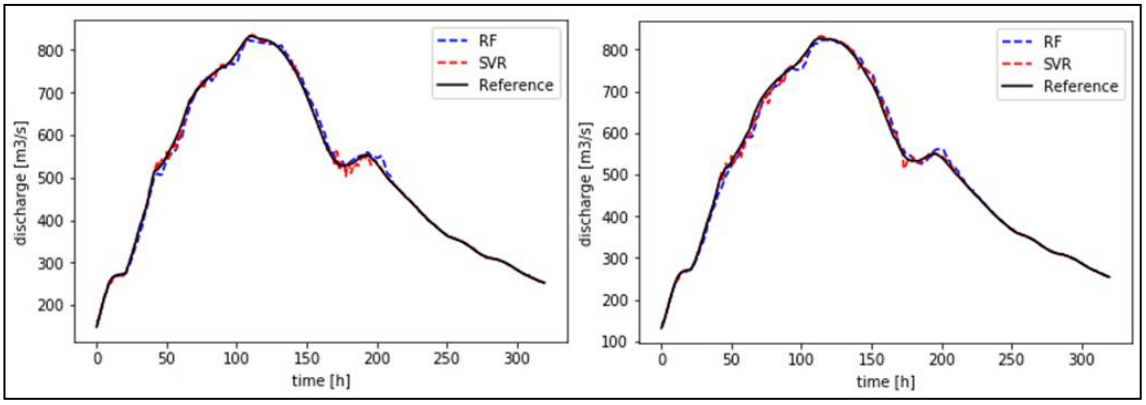


Figure 7.5: Comparing forecasted flows at MC010 (left) and MC033 (right) for Case 1.

However, in Case 2 the SVR-based models showed opposite outcomes than the Case 1 (Table 7.3). The observed performance deterioration could be because the models trained on subset 2 had a lower peak (832.5 m^3/s) than subset 1 (899.2 m^3/s), and are not efficient in predicting values beyond the data ranges used for training. In addition, models suffered from overfitting. Figure 7.6 shows the forecasted flows at MC010 and MC033 nodes for Case 2 and shows that RF-based models performed significantly well and did not suffer from the issues of overfitting.

Section ID	Method	Case 1			Case 2		
		R-squared	RMSE	NSE	R-squared	RMSE	NSE
MC010	SVR	0.999	7.824	0.999	0.852	84.523	0.852
MC010	RF	0.997	9.659	0.997	0.991	20.121	0.991
MC033	SVR	0.998	7.536	0.998	0.861	81.091	0.861
MC033	RF	0.997	11.115	0.996	0.991	20.055	0.991

Table 7.3: Error statistics of RF and SVR.

Based solely on Case 1 statistics, one would select the SVR-based models for forecasting flows. However, considering the possible consequences of model overfitting (observed

in Case 2), the RF technique was chosen for T+3h flow forecasting in this research and SVR-based models were discarded from further analysis.

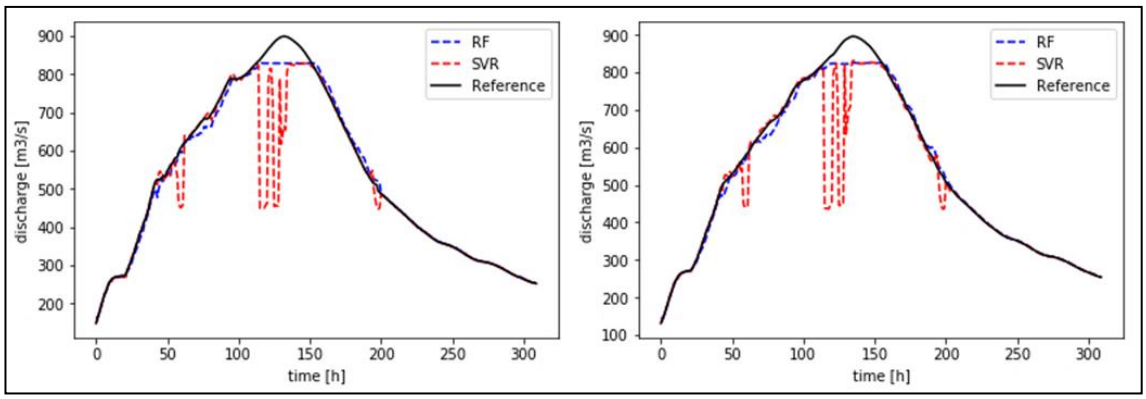


Figure 7.6: Comparing forecasted flows at MC010 (left) and MC033 (right) for Case 2.

7.10.2 Wet/Dry cell forecast

A clear difference between Chapter 6 and the current experiment is that in the current study, the MLP classifiers were trained using water depth values from the synthetic flows. Furthermore, the earlier experiment involved wet/dry cell forecasting using one of the subsetted hydrographs (subset 2). In contrast, in the current study, the classification task was carried out using the RF forecasted T+3h flows at sections MC010 and MC033 which were then fed into the pre-trained MLP-based classifiers to forecast 'wet/dry' states of the 150 sampled locations. Then the forecasting accuracies were compared to the FM simulated results.

The MLP simulated results were first compared against the FM outputs in terms of arrival time (Table 7.4, columns 2-8). The results from all the sampled locations show that for Case 1, the mean error observed in the flood arrival time was about +1h 51min (95% CI [-2h, +6h]). Thus, on average, there could be an up to 1hour 55m delay in predicted arrival times. For case 2 (RF forecasted flows for subset 1), the observed mean error was of order +1h 51min (95% CI [-2h, +6h 27min]).

To estimate the classification accuracy of the models, three distinct measurements were estimated for both subsets: i) overall accuracy; ii) *F1*-score; and iii) *F*-scores. The overall accuracy and *F1-score* (harmonic mean of recall and precision) were calculated from a confusion matrix and values were averaged across all 150 locations (Table 7.4, columns 9-10). The *F-scores* were calculated using the method described in Aronica et al. (2002).

The overall accuracy measure describes how accurately sampled locations were classified during the flood event; the *F1*-score compares how well the models correctly classified flooded cells at sampled locations during the event; and the *F*-score measures how many sampled locations were correctly classified at a particular time step.

	Descriptive statistics of arrival time							Classification accuracy	
Study case	Mean	Median	Min	Max	1st Quartile	3rd Quartile	95% CI	Overall accuracy	Average *F1*- score
Case 1	1h51m	2h	-2h	7h	0h	4h	[-2h, 6h]	0.989	0.98
Case 2	1h55m	2h	-2h	7h	0h	4h	[-2h, 6h27m]	0.985	0.983

Table 7.4: Results of the integrated ML-based model when compared against reference FM outputs for subset 1.

To present the yielded *F*-scores, both Case 1 and 2 flood events were divided into several time segments: the first segment consisted of the dry hours prior to the flood arrival, followed by several segments of 48 hours during the event, with the last segment consisting of the remaining hours of the event. The sampled locations started to get flooded at the 32nd hour. Arrival time errors were detected for the very first flooded sampled location (location ID 81). Location 81 experienced first flood water at the 29th hour into the event while classifiers forecasted the location to be flooded at the 32nd, resulting in 3 hours of delay. Since it was only one location which was misclassified as a dry cell for 3 hours, the F-score was not calculated for this, as it would be a 0 in such cases (**Eq.** 5.1). The F-scores were calculated from the 32nd hour onwards in 48 hours segments. Table 7.5 shows the averaged *F-score* for the various segments.

Case 1		**Case 2**	
Time step	***F-score***	**Time step**	***F-score***
1st 32 hours	No flooded cells	1st 32 hours	No flooded cells
Next 48 hours	90.9%	Next 48 hours	90.0%
Next 48 hours	99.3%	Next 48 hours	99.2%
Next 48 hours	99.6%	Next 48 hours	97.5%
Next 48 hours	98.5%	Next 48 hours	98.9%
Next 48 hours	98.8%	Next 48 hours	97.1%

Last 52 hours	97.8%	Last 41 hours	98.5%
Avg.	**98.82%**		**98.20%**

Table 7.5: Average F-scores defining classification accuracies at the different temporal segments.

The mean overall accuracy and mean F1-score of the two case studies were 0.987 and 0.982, respectively. The estimated average arrival time error and a 95% CI were 1h 53m (delay) and −2h to 6h 13min, respectively. The classifiers, on average, yielded 98.51% accuracy in detecting the true state of the sampled locations for every time step.

7.10.3 Interpolated flood extent maps

High accuracies, as noted for wet/dry cell classification using the ML-based models, indicate that the idea of producing probabilistic maps from data points could be a potential solution to real-time flood inundation visualisation problems. Figures 7.7A and 7.7B respectively show a forecasted probabilistic flood inundation map from the RK method and a deterministic map (water depth, measured in metres) from FM software derived for case 1 at the 122nd hour (flood peak period). Although it is not possible to make a direct comparison between a probabilistic and a deterministic map. However, the significant features can be noticed exclusively, e.g. the water depth values from FM were very high at this time step and the town was severely flooded. The same can be seen in the probabilistic map. Higher probabilistic values indicate that the proposed method was able to successfully detect the flooded cells. In addition, an advanced synthetic aperture radar (ASAR) image captured on 8 November 2000 at 12:18 was acquired. This capture is almost in line with the flood map predicted at the 111th hour of the case 1 runs. Therefore, both the FM simulated water depth map and the interpolated map were binarised and compared with the ASAR image. The resulting F-scores are given in Table 7.6. The results show that the proposed method performed comparatively better than FM. It should also be noted that the proposed method predicted this flooding state 3 h ahead. This is an encouraging outcome and provides a strong case for the suitability of the proposed data-driven modelling framework.

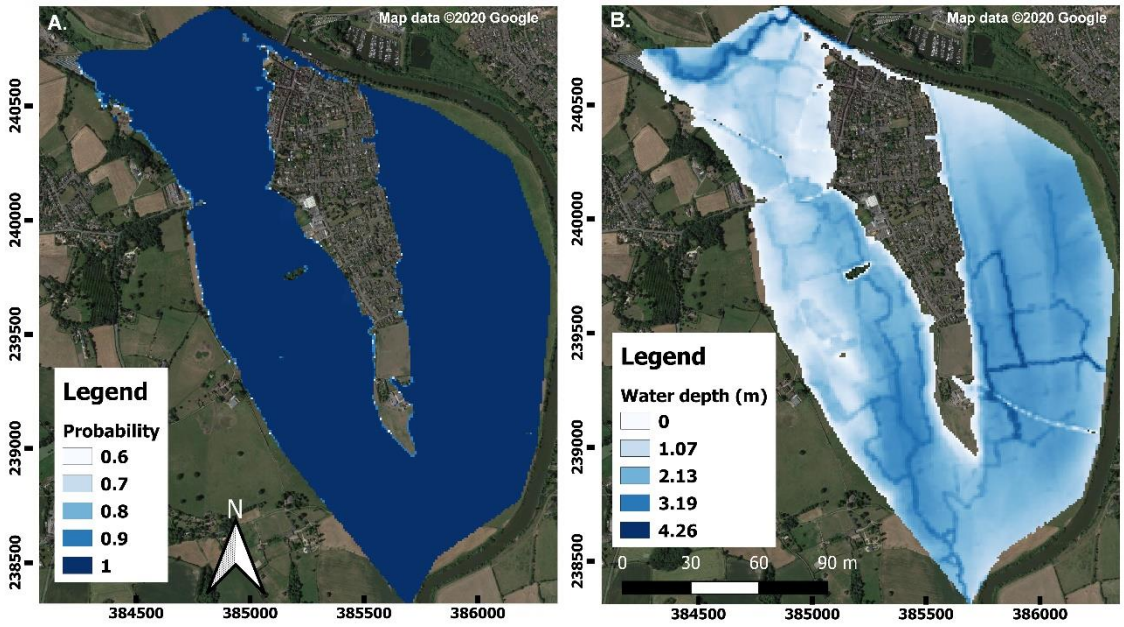


Figure 7.7: (A) Probabilistic flood inundation map forecasted at 122nd hour using the proposed data-driven modelling framework; (B) deterministic flood map generated by FM software.

The flood extent maps are compared in Figure 7.8. The figure shows a significant match among the extents derived from all three sources (i.e. FM, integrated ML method and ASAR). Probabilistic and deterministic maps for case 1 from two other flooding phases (flood growing and receding periods) are illustrated in Figure 7.9.

Model	F-score: %
Proposed ML method	88.22
FM	86.58

Table 7.6: Classification accuracies for comparison of FM and proposed ML method with ASAR image.

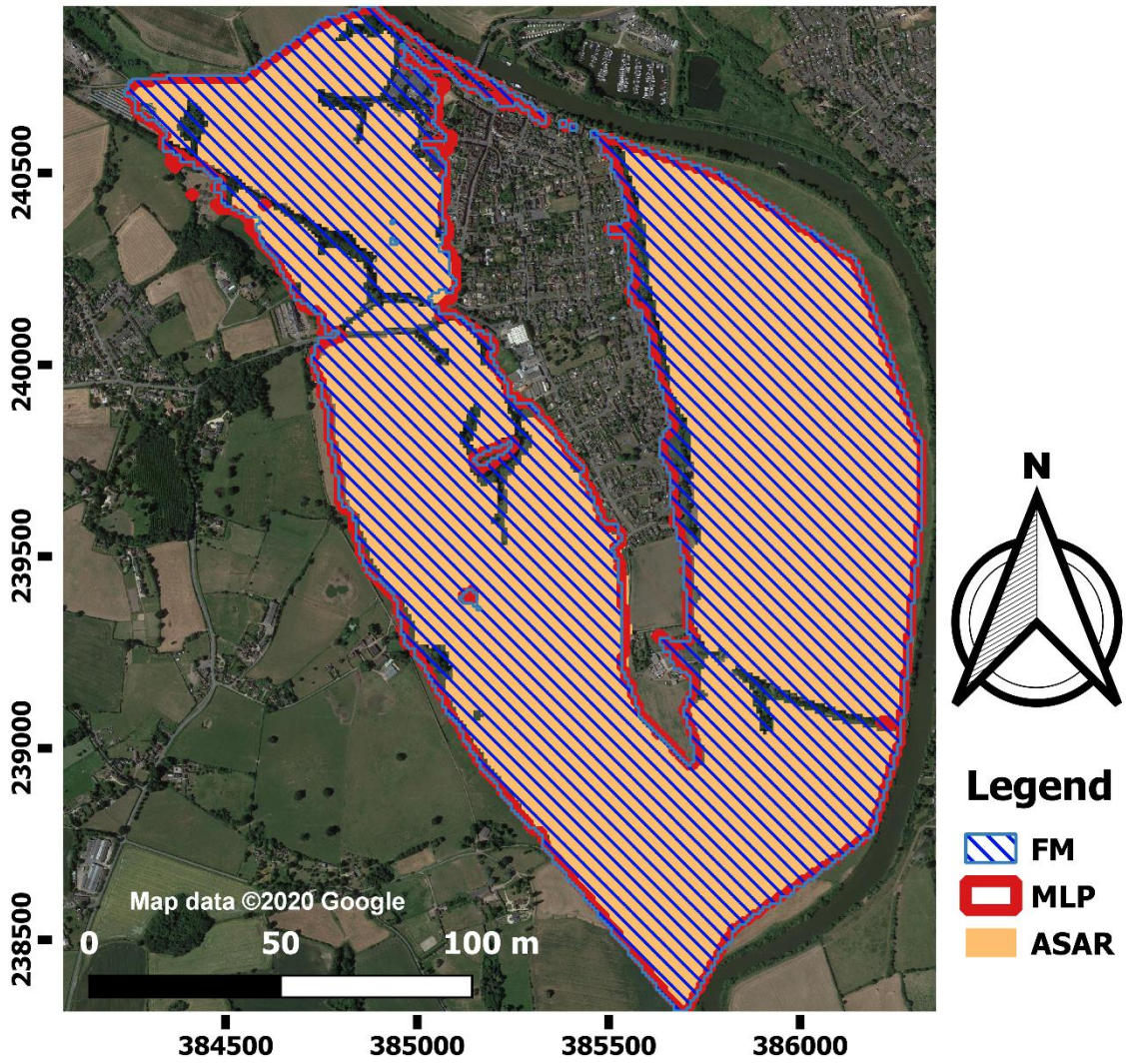


Figure 7.8: Flood extent maps derived from the proposed ML method, the FM and the ASAR image captured on 8 November 2000.

During training of the MLP-based models, water depth values >0.09 m were set to 1. Therefore, the probabilistic map was expected to have high probability values where the water depth values were higher and where flooding was experienced for a longer duration. Figures 7.9A and 7.9B respectively show the probabilistic and deterministic conditions of the event at the 52nd hour (flood growing period). As expected, the probabilistic map shows very low probability values in the regions where the deterministic map shows no or insignificant water depths. On the other hand, probabilities are higher in the regions where the water depth values are large. However, a slight overestimation can be noticed in the north-west part of the domain. During the flood recession period (323rd hour), fragmented water cells are observed in both maps (Figures 7.9C and 7.9D). The dominant regions are clearly visible and a greater threshold value (>0.6 m) can be applied to the probabilistic map to delineate inundated cells.

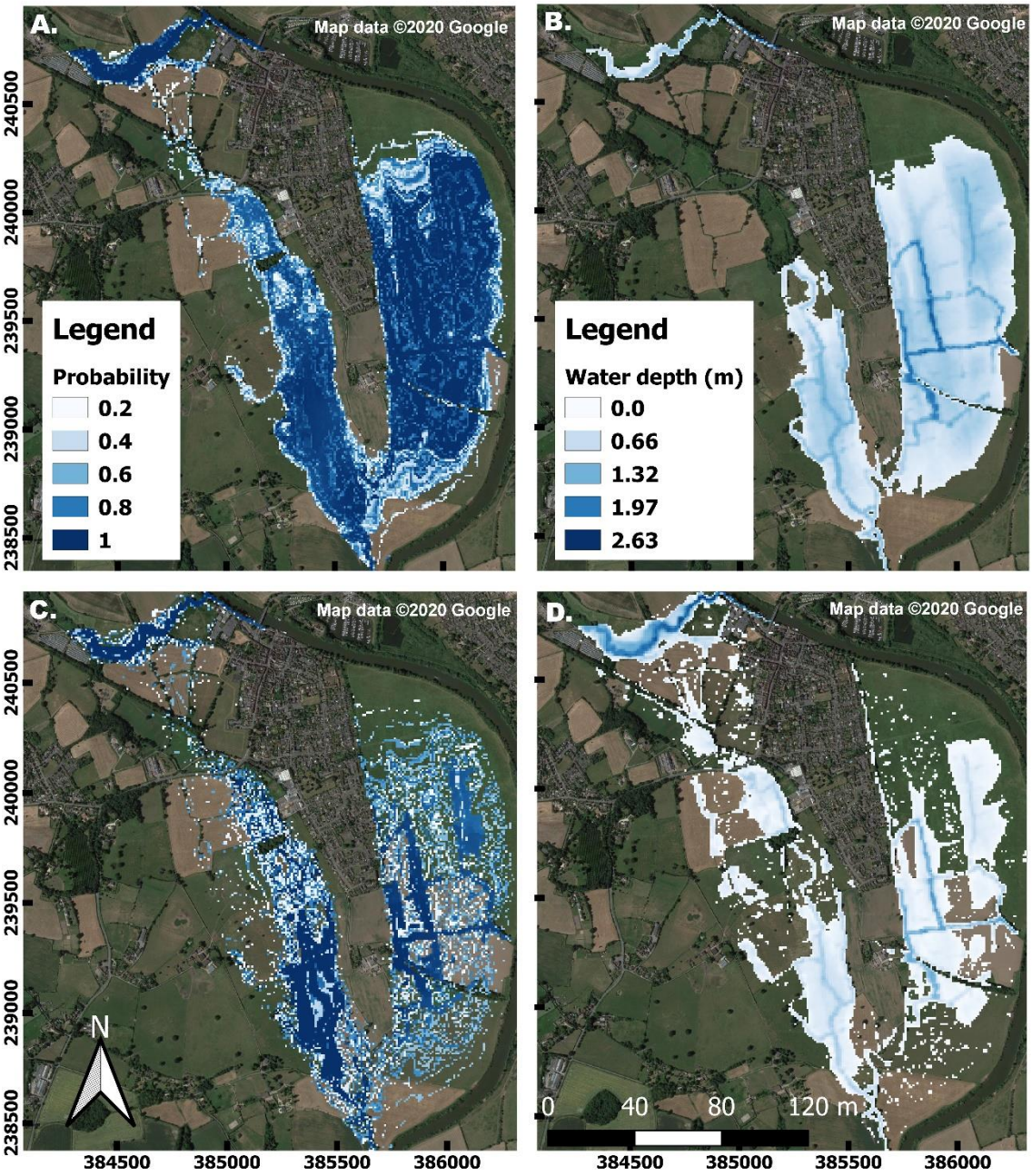


Figure 7.9: (A) Probabilistic flood inundation map forecasted during growing phase using the proposed data-driven modelling framework; (B) deterministic flood map generated by FM software during growing phase; (C) probabilistic flood inundation map forecasted during receding phase using the proposed data-driven modelling framework; (D) deterministic flood map generated by FM software during receding phase.

7.11 Conclusion

This chapter presented the development of a data-driven modelling framework for operational flood forecasting and inundation visualisation. The test results obtained for the proposed model from the Upton case study reveal promising outcomes in terms of forecasting inundated areas in real-time. A key finding of this research is that the

classifiers can be trained on samples generated from coarse grid 2D hydrodynamic model runs and then be applied to high-resolution DEM for dynamic mapping of flood inundation extents.

Major advantages of the proposed modelling framework are that it provides greater computational efficiency than fully distributed 2D models (in the context of real-time application) and it does not require large storage capacity compared to the offline method. Model training process does not take longer, and models can be re-trained in case of any significant changes in the topography of the floodplain. In addition, it requires very little expertise to run the trained models. Furthermore, it could also be possible to add/assimilate external information from satellite images, social sites or a multitude of data coming from other sensors in the same modelling platform. This study has also shown that the models can be trained using synthetic datasets if there are not enough observed data available.

The quantification of model uncertainty was limited to arrival time, the next stage of development of the proposed real-time inundation modelling system could be a systematic uncertainty quantification. Further research is also required to improve forecasting and visualization capabilities of the model (e.g. training on data generated from a calibrated 2D model, using observed inflow hydrographs instead of synthetic flow series). For urban flood mapping, Digital Surface Model (DSM) could be used for spatial interpolation for a street level visual representation.

Chapter 8 **Concluding Remarks and Future Developments**

8.1 Research summary

The impact of human actions and amplified anthropogenic emission on the climate system is clear and are the highest in history IPCC (2014). These unprecedented climate changes have had prevalent impacts on our natural systems. For the UK, most of the research has now concluded that climate change has contributed to increasing the frequency and severity of extreme flood events. Extreme flood events are often associated with devastating impacts on the society, economy, environment, and infrastructure, including disruption to critical services and people in flood prone areas. To ensure sustainability, in light of the projected future climate change, a considerable amount of research has been done and is on-going that is aiming towards the development of a comprehensive flood forecasting and standardized flood risk management scheme (Schumann et al., 2008). Bearing this subject matter in mind, the research reported in this thesis has investigated and developed an exclusive ML based real-time flood inundation extent mapping system.

To begin with, a large number of literatures were reviewed which have applied ML techniques in flood forecasting and inundation modelling. The summary of these reviewed articles is documented in Chapter 3. Most of these published articles have predominantly studied rainfall-streamflow/discharge, rainfall-runoff modelling. However, the application of ML in inundation modelling was found to be very limited. Therefore, in this thesis a novel framework has been developed that integrates the rainfall-streamflow and streamflow-inundation models to generate sequential inundation extents from rainfall data.

Different ML techniques were implemented to construct the rainfall-streamflow (Chapter 4) and streamflow-inundation models (Chapter 5 and 6). Finally, suitable candidate models were combined in Chapter 7 to present the holistic rainfall-inundation modelling framework. Summary of these chapters are given in the following:

Chapter 4 attempted to identify a simple, but robust ML based model which can be applied to forecast multi-step ahead streamflow from rainfall data. The chapter presented a systematic investigation into modelling capacities of three data-driven modelling techniques, namely, WANN, SVR and DBN for multi-step ahead stream flow forecasting.

To evaluate the effectiveness of these modelling techniques, hydro-meteorological hourly datasets from three case-study rivers, located in the UK, were used. The results showed that the SVR-based model could forecast quite accurately up to one to three hours ahead, but its performance deteriorated gradually from three hours onwards. Furthermore, it was found that the WANN-based model performed better when the overall non-linearity of the system increased, whereas the DBN-based model appeared to show consistently poor predictive capabilities when compared to the other models. The results suggest that, for any selected model, it is possible to use an identical model structure for up to two time-steps ahead forecasting and that models need to be re-configured beyond that limit. The study also identifies the SVR-based model as the best-suited model for short term forecasting.

The objective of Chapter 5 was to develop and assess the potential of ML based models to map the inundation extent for a multi-channel river basin. This chapter presented SVR and ANN based modelling schematics, developed, and trained using the outputs from a 2D hydraulic model. Multiyear flood extent maps for the River Niger catchment in Mali were used as a case study and modelled using a LISFLOOD-FP hydrodynamic model. The data-driven models were validated to assess the reliability and accuracy of their predictions. Results showed that both data-centric modelling engines could efficiently simulate the outcomes of the hydrodynamic model with considerably high accuracy, measured in terms of error rate, during the high flow periods. However, for probabilistic extent mapping, MLP-based models could be preferred. The methodological approach presented herein can be adapted widely to substantially reduce the computational cost and time required in operational flood forecasting and mapping.

Chapter 6 intended to evaluate the performance of multiple ML algorithms in generating flood extent maps using higher spatial and temporal resolution (30 m and 1 hour, respectively). A semi-urban area (the town of Upton-Upon-Severn) was chosen for training MLP-based classifier models to generate probabilistic flood mapping, while RF and SVR were applied for regression-based flood extent mapping. The results from the SVR-based models appeared to be less efficient and less stable compared to the RF-based models. The results showed that the MLP and RF-based models could predict the arrival of flood water within -2 to +3 hours and -1 to +6.4 hours, respectively, with 95% confidence. The predicted outputs, i.e. presence/absence of water, during a flood event at the sampled locations were interpolated using the RK method to produce flood extent

maps. These maps were compared against FM generated maps during three flood stages (during the flood growing period, peak flood, and during the receding period). The results indicated that the predicted maps underestimated the number of inundated cells during the flood growing stage, but results improved slowly and produced a very good match during peak flood with an accuracy of 98.1% for the MLP and 97.4% for the RF based models. However, prediction accuracy reduced drastically when the flow started to decrease from the peak, possibly due to the selection of threshold values used to produce binary maps and the limited number of sampled locations used. The chapter also addressed factors, such as the number of ground sampling points and cut off threshold for binary mapping, that limiting the model performance during flood growing and receding phases.

The objective of Chapter 7 was to improve upon the limitations from previous studies and propose a novel data-driven modelling framework to forecast probabilistic flood inundation maps for real-time applications. The proposed end-to-end (rainfall-inundation) method integrated a suite of ML algorithms to forecast discharge and subsequently delivered probabilistic flood inundation maps. The RF based rainfall-discharge models were integrated with MLP based classifiers to classify wet/dry cells. For visualisation, statistical interpolation method was applied that generated dynamic inundation maps. The performance of the novel forecasting hybrid model was assessed using two subsets of data created from an observed flood event. The results showed that the model could effectively simulate the outcomes of the hydrodynamic model, specifically of the Flood modeller, with considerably high accuracy measured in terms of flood arrival time error and classification accuracy. The mean arrival time error for the difference in outcomes of the proposed model and the hydrodynamic model was noted 1 h 53 min, while classification accuracy was estimated to be 98.51%.

8.2 Conclusions

8.2.1 General discussion

The thesis presented a novel data-driven modelling framework for visualizing real-time flood inundation information. The concept of deriving an ML-based flood extent mapping system was successfully formulated and tested. During the model development and testing stage, *R* and *Python* programming languages were used in conjunction. A *Python*

implementation of the integrated dynamic system can be found in the end of the thesis (Appendix D). The simulation time for generating the 9m maps were only 14 seconds (approximately), which is significantly lower than the 2D model simulation time.

However, the proposed framework was configured and tested for fluvial flooding conditions. The spatial domain of the study area was relatively small. In the future, similar methods could be developed for larger spatial domains. For a large domain (for streamflow-inundation modelling), it may not be feasible to train a separate ML-based model for each sampled location and therefore, a global classifier approach should be adopted. Although an ANN can predict multiple outputs (a single ANN for all the cells in the domain), due to computer memory limitations, it is also not possible to make predictions for an entire domain containing millions of cells. Therefore, a balance between computer memory and the number of sampled locations (GCPs) is required. The predicted values at the GCPs can then be simply interpolated using a higher resolution DEM. In this research, inundation maps were produced that only demonstrate the extent; however, deterministic depth maps can also be generated. To do so, one can skip the binarising steps in the development of the hydraulic classifier database and fit regression-based models.

The performance of ML-based models depends on the quality and quantity of the training data. An integral part of developing data-driven models is to make sure the training data samples are sufficient and encompass all possible hydrological scenarios. In this study, the classifiers were trained using four synthetic hydrographs. The number of hydrographs can be increased to provide more training samples. For large domains, topographical features (e.g. land cover data), rainfall and so on can also be included in the training dataset.

Uncertainty in outputs can severely limit the application of such a data-driven modelling framework. This is because every step of the modelling chain – from the input of the 2D hydrodynamic/hydraulic model simulation to the output of the RK method – leads to a cascade of uncertainty. While it is not possible to eliminate uncertainties completely, they can be diminished and quantified. To reduce model uncertainty, it is essential to generate training samples from a fully calibrated 2D model and to optimise the hyperparameters of the ML-based model. However, a comprehensive quantification of propagated uncertainty was not part of this research and the quantification of uncertainty was limited to flood arrival time.

The primary objective of this work was to develop a data-driven end-to-end inundation mapping system that can be used for operational forecasting. Specific to this research, the proposed method reduced the computational time from minutes to seconds. The FM software takes about 40 min to generate raster files containing water depths. (Note, velocity files were not generated in this study: the model simulation time would be significantly longer if velocity values were also produced). In contrast, once the training samples are prepared, training of the RF-based models can take only a few seconds. The training of MLP-based models, however, depends on the selected approach. Naturally, training a separate model for each GCP (local MLP) will take longer than training a global MLP. Both local and global approaches were tested in this study. The global model took ~9 sec to complete the training process while the local approach took ~25 min. Once the models are trained, it takes ~10 sec to sequentially generate probabilistic inundation maps. An Intel Core i5- 250 GHz CPU with 8 GB RAM system was used to execute both the FM and data-driven model runs.

8.2.2 Contribution to knowledge

The research activities conducted in the thesis has contributed to knowledge through two peer reviewed journal articles (i.e. Kabir et al. (2019) and Kabir et al. (2020a)). The possibility of transferring the model parameters across time and space for streamflow forecasting has been demonstrated in Kabir et al. (2019) and the potential of the proposed integrated modelling framework in estimating multi-step ahead inundation extent has been presented in Kabir et al (2020a).

Although the present thesis is focused on flood extent modelling, the author has shown that regression-based models could be used to estimate water depths as well. This knowledge has led to a recently published research article (i.e. Kabir et al. (2020b)). In this study, the authors have successfully implemented a regression-based deep convolutional neural network model to predict the water depths for two flood events in the UK (Carlisle 2005 and 2015).

8.2.3 Key limitations

Challenges and limitations are an integral facet of any research work. The key constraints of this research and how these were tackled are listed as below:

- *Challenges*

Data scarcity has been one of the key challenges for this research. There was a limited option for choosing case study areas, partly due to data unavailability and the data required to conduct the research activities were provided in parts by third parties. Therefore, the case studies were selected for the areas for which required hydro-meteorological and topographical data were available. For example, in Chapter 4, hourly rainfall and streamflow for three study catchments were provided by the School of Geographical Sciences, University of Bristol. Datasets for ML based flood extent mapping in Niger Inland Delta were also provided by the School of Geographical Sciences, University of Bristol (Chapter 5). On the other hand, Upton-Upon-Severn case study data sets were acquired from the School of Energy, Geosciences, Infrastructure and Society (EGIS) database and covered only one flood event. Therefore, multiple synthetic hydrographs were created and used during the model development and validation process. It should also be noted that hourly rainfall data for the Upton study catchment was not available from the archive. Only two MIDAS stations were found that had historical records of hourly rainfall data. Hence, rainfall from these two stations was used as a proxy for the entire catchment. It should also be noted that in this research a free version of FM software was used which has limited access to functions. Therefore, the model could not be simulated for a larger domain with high spatial resolution.

- *Limitations*

One of the key limitations of this modelling framework is that the output does not provide depth and velocity information. However, some studies have shown that it is possible to apply ML-based models for depth estimation (e.g. Bermúdez et al., 2019; Liu and Pender, 2015) and therefore, in future studies, a depth estimation module could be embedded to the system. However, developing a model to estimate velocity can be challenging and, so far, recent studies have reported that ML-based models do not perform well in predicting velocities (Bermúdez et al., 2019). In addition, the current approach has only been tested for fluvial flooding. Further studies are required to develop and test the capacity of data-driven models for pluvial flooding.

Another limitation of this study is that the performance of the integrated model could not be compared against any other ML based modelling system. For example, the methods proposed by Chang et al. (2018a), Chang et al. (2014b) would require hundreds of hydrological conditions (rainfall - inundation) to do the clustering and training the recurrent networks. It was, therefore, not possible to develop their model for comparison

purposes. The proposed integrated modelling framework rather offers an alternative and time efficient approach to those above-mentioned studies because it does not require hundreds of hydrographs to run 2D hydraulic models.

8.3 Future work

One of the biggest advantages of the ML based models is that they can be scaled and easily modified to add new components. The proposed framework for real-time flood inundation can be further improved by plugging auto-updating features for refining the forecasted maps. For example, fusing live social media information utility can be used to update flood maps. Rosser et al. (2017) and Fohringer et al. (2015) have used social media information to produce rapid inundation maps.

The proposed methodology only generates dynamic flood maps. A comprehensive uncertainty analysis of the proposed integrated modelling schematics and water level extraction were not conducted as a part of the PhD study (mainly due to the time constraints). However, it can be more beneficial to generate water depths and velocities at the same time for full-scale inundation modelling. This can be further investigated in the future studies.

The LSTM networks have shown to perform well in flow forecasting in the snow-dominated catchments (Kratzert et al., 2018). This module could add value to the model and may significantly improve flood forecasting and inundation mapping in such areas. Therefore, a future study could potentially use the LSTM network as a hydrological forecasting module instead of RF. It should also be mentioned that the implemented hydrological model did not consider forecasted rainfall as an input. This could potentially enhance the multi-step ahead streamflow forecasting accuracy. In addition, further research will be required to develop the ML-based surface water flooding module and investigate the effect of hysteresis. Future studies may yield many more fascinating and comprehensive insights into the development of AI-based flood inundation modelling that will hopefully be of interest to many disciplines studying disaster risk management.

Appendix A

Variogram models

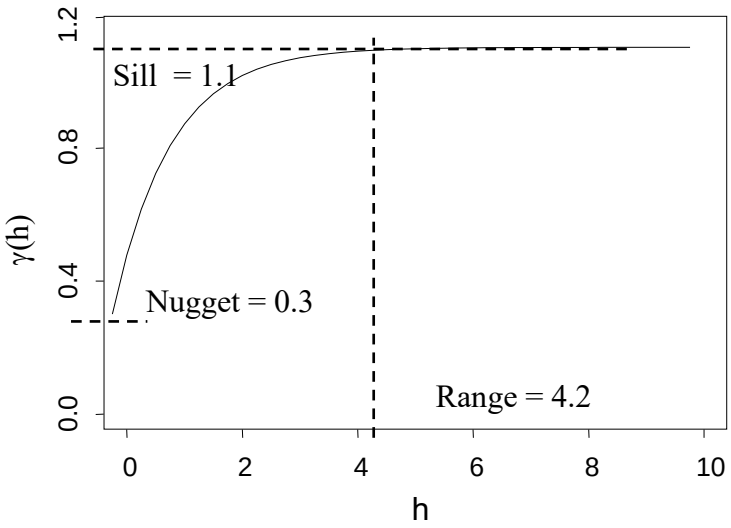
Variogram models are used to describe how the spatial data are related with distance. It is defined in terms of variation in data values between two locations separated by a distance *h*. A widely used robust variogram estimator is given by Cressie and Hawkins (1980):

$$\hat{\gamma}(h) = \frac{1}{2N(h)} \frac{(\sum |z(s_i) - z(s_i + h)|^{1/2})^4}{0.457 + 0.494/N(h)}$$

Where, $z(.)$ is a geostatistical model, h is the distance separating sample locations s_i and $s_i + h$, $N(h)$is the number of distinct data pairs. Note that, the terms variogram and semivariogram are often used interchangeably. By definition $\gamma(h)$ is semivariogram and the variogram is $2\gamma(h)$.

Variogram parameters

The objective of fitting a variogram it to best estimate the autocorrelation structure of the underlying stochastic process. A variogram is described using three parameters:



- Nugget – It is the measurement error estimated from the empirical variogram $\hat{\gamma}(h)$ at $h = 0$.
- Range – It represents the distance at which data are no longer correlated.
- Sill – It represents the variance where the variogram reaches at the range.

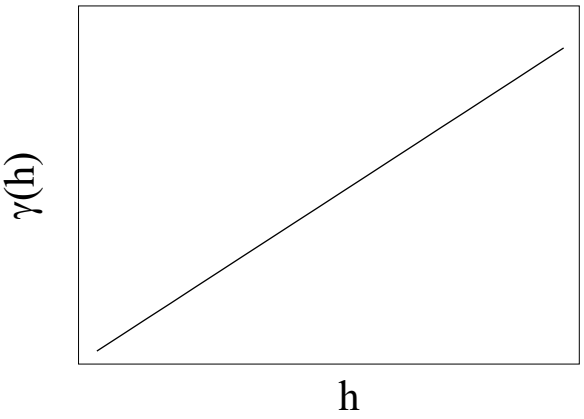
We need to fit a model to the empirical variogram because kriging requires estimates of the variogram, $\gamma(h)$, for those distances, *h*, which are not available in the data.

Various variogram models are available:

Linear mode

$$\gamma(h) = c + bh$$

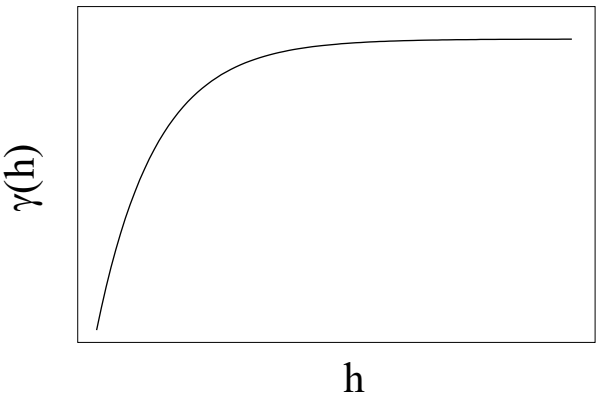
Where, c is the nugget and h is the distance. The linear variogram has no sill and suggests a trend in the data.



Exponential model

$$\gamma(h) = c + c_0(1 - e^{-h/\alpha})$$

Here c is the nugget and sill is $c + c_0$. The range is defined as $3a$ at which the variogram is of 95% of the sill.



Spherical model

$$\gamma(h) = c + c_0(\frac{3}{2}\frac{h}{a} - \frac{1}{2}(\frac{h}{a})^3)$$

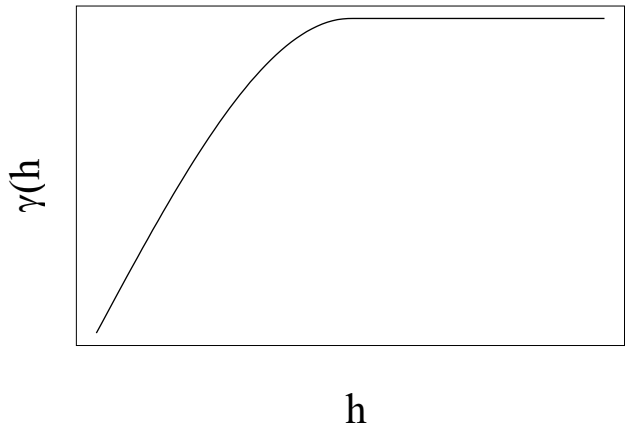
for $0 < h \leq a$

and

$$\gamma(h) = c + c_0$$

for $h \geq a$

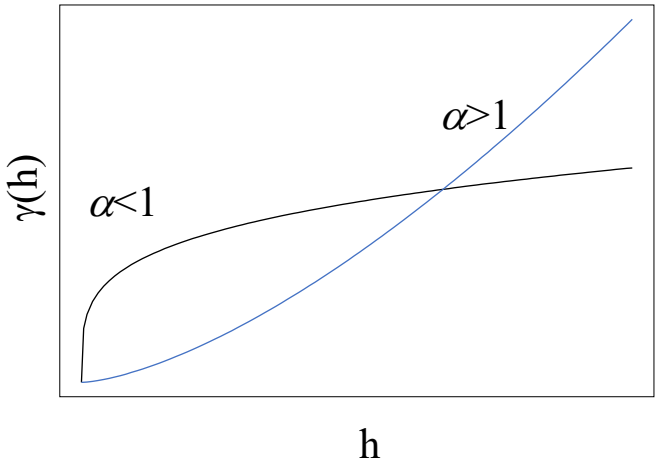
Here c is the nugget and sill is $c + c_0$. The range for the spherical model can be computed by setting $\gamma(h)$ $0.95(c + c_0)$.



Power model

$\gamma(h) = c + bh^{\alpha}$

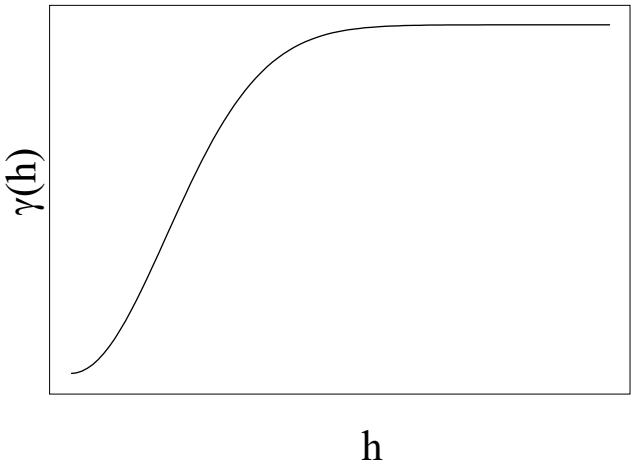
where c is the nugget. Similar to the linear model, power model also does not have a sill. This means that the variance of the process is infinite.



Gaussian model

$\gamma(h) = c + c_0(1 - e^{-(h/\alpha)^2})$

Here c is the nugget, $c + c_0$ is the sill and the range is $3a$.



Appendix B

Grid search algorithm for SVR parameter estimation

```
from sklearn.svm import NuSVR

from sklearn.svm import SVR

from sklearn.model_selection import GridSearchCV

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

##import data

train= pd.read_csv("train.csv")

test= pd.read_csv("test.csv")

X_train = train.iloc[:,1:8]

Y_train = train.iloc[:,8]

X_test = test.iloc[:,1:8]

Y_test = test.iloc[:,8]

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

def nusvr_param_select(X_train, Y_train, nfolds):

    Cs = [0.001,0.01,0.1,1,5,10,15,20,100,1000]

    gammas = [0.001,0.01,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]

    NUs = [0.001,0.01,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]

    param_grid = {'C': Cs, 'gamma': gammas, 'nu': NUs}

    model_param = GridSearchCV(NuSVR(kernel='rbf'), param_grid, cv =
nfolds)

    model_param.fit(X_train, Y_train)

    model_param.best_params_

    return model_param.best_params_

results = nusvr_param_select(X_train, Y_train, 5)

svr_rbf = NuSVR(kernel='rbf', C=1e3, gamma=0.01, nu=1)
```

```
y_rbf = svr_rbf.fit(X_train, Y_train)

pred_svr = y_rbf.predict(X_test)

Q = pd.DataFrame(pred_svr)

Q.to_csv("pred_1.csv", encoding='utf-8', index=False)
```

The ‘results’ variable would display the parameters:

```
{'C': 1000, 'gamma': 0.01, 'nu': 1}
```

Appendix C

Selecting variogram model

The following *Python* script was used to find the best variogram model for RK task (Chapter 8).

```
from sklearn.linear_model import LinearRegression
from pykrige.rk import RegressionKriging
from pykrige.compat import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np

#load presence of water samples
df = pd.read_csv("RK_database_k2.csv")
hydro = pd.read_csv("MLP_Model_DataBase.csv") #load coordinate and
elevation values
x = hydro.iloc[:,:-2]
x = x.iloc[:,-2:]
x = x.to_numpy()   #2D array of coordinates
x = np.float64(x)

#convert and reshape the predictor (elevation) variable
p = hydro['Bed'].values
p = np.reshape(p, (-1, 1))
target = df['Hour_62'].values

#split samples 80% for fitting the model and 20% for validation
p_train, p_test, x_train, x_test, target_train, target_test =
train_test_split(p, x, target, test_size=0.2, random_state=42)

# linear regression model
lr_model = LinearRegression(normalize=True, copy_X=True,
fit_intercept=False)

vg_models = ["spherical", "exponential", "linear", "power",
"gaussian"]

for m in vg_models:
```

```
    print('=' * 40)
    print('vg model:', m)
    #fit the regression kriging model
    m_rk        =        RegressionKriging(regression_model=lr_model,
variogram_model= m)
    m_rk.fit(p_train, x_train, target_train)
    pred = m_rk.predict(p_test, x_test)
    waterdf = pd.DataFrame({'Flood_Probability': pred})

    waterdf[waterdf >= 0.5] =1.0
    waterdf[waterdf < 0.5] =0.0
    print('RK score: ', accuracy_score(waterdf, target_test))
```

Appendix D

Dynamic inundation modelling

The *python* script loads previously trained RF and MLP-based models. Forecasts hourly inundation maps.

Specific instructions: to reproduce the results please change the path to the directories of the necessary files and saved models.

```
####

### This python script loads previously neural network based

### trained models from the disk and dynamically generates flood

##inundation maps

import pandas as pd
import numpy as np
import time
import matplotlib.pyplot as plt
import sys
from sklearn.ensemble import RandomForestRegressor
from pykrige.rk import RegressionKriging
from shapely.geometry import Point
import geopandas
import rasterio
from rasterio import features
from pykrige.compat import train_test_split
from rasterio.plot import show
import pickle
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import load_model, Sequential
from tensorflow.keras.layers import Dense
#import keras
import h5py
from IPython import display


###import training data for the classifiers to scale input data
train_H= pd.read_csv("Training_data_study_6.csv")

X_flow = train_H.iloc[:, 0:7]

# Define the scaler
scaler = StandardScaler().fit(X_flow)

#import MLP based hydraulic classifier database
model= []
```

```
hydro = pd.read_csv("MLP_Model_DataBase.csv")

for index, row in hydro.iterrows():
    hydro_mod = (hydro['Mod_Loc'][index])
    model1 = load_model(hydro_mod)
    model.append(model1)

##background images
r = rasterio.open('UUS.tif')
red = r.read(1)
bounds = (r.bounds.left, r.bounds.right, \
          r.bounds.bottom, r.bounds.top)



##Load rainfall-flow data
mc010 = pd.read_csv("hydrograph_k2_MC010_3hr_ahead.csv")
mc033 = pd.read_csv("hydrograph_k2_MC033_3hr_ahead.csv")

mc010_x = mc010.iloc[:,4:11]
mc033_x = mc033.iloc[:,4:11]
mc010_x.head(), mc033_x.head()

##put initial flow values with lags (Q1, Q2, Q3)
inflow_mc010 = [121.55, 120.23, 116.45]
inflow_mc033 = [121.55, 120.23, 116.45]


##time 3 hours ahead (in seconds)
t = 10800


for index, row in mc010_x.iterrows():
    ###start calculate timing
    start_time = time.time()
    hydroDB = pd.read_csv("RF_HydroModel_DB.csv")

    ##load saved model
    mod_mc010 = pickle.load(open(hydroDB['Hydro_Model_Loc'][0],"rb"))
    mod_mc033 = pickle.load(open(hydroDB['Hydro_Model_Loc'][1],"rb"))

    prd_mc010 = mod_mc010.predict(mc010_x.iloc[index].to_frame().T)
    prd_mc033 = mod_mc033.predict(mc033_x.iloc[index].to_frame().T)

    inflow_mc010.append(prd_mc010)
    inflow_mc033.append(prd_mc033)

    del inflow_mc010[0]
    del inflow_mc033[0]

    dt                                                              =
{'T':[t],'mc010q1':[inflow_mc010[0]],'mc010q2':[inflow_mc010[1]],
         'mc010q3':[inflow_mc010[2]],'mc033q1':[inflow_mc033[0]],
         'mc033q2':[inflow_mc033[1]],'mc033q3':[inflow_mc033[2]]}
```

```
flow = pd.DataFrame(data=dt)

# Scale the data
flow = scaler.transform(flow)

####import Hydraulic model database

water= []

#plt.ion()

for i in model:
    mdl = i
    water_pred = mdl.predict(flow)
    water_pred = np.float64(water_pred)
    water.append(water_pred)

    waterdf = pd.DataFrame({'Flood_Probability': water)

    waterdf[waterdf >= 0.5] =1.0
    waterdf[waterdf < 0.5] =0.0   #binarized data

    target= waterdf['Flood_Probability'].values

    op = index+3 #outputs are t+3h lead

if (target == 1).any() == True:

    ###RK step

    x = hydro.iloc[:,:-2]
    x = x.iloc[:,-2:]
    x = x.to_numpy()   #2D array of coordinates


    #convert and reshape the predictor (elevation) variable
    p = hydro['Bed'].values
    p = np.reshape(p, (-1, 1))

    #constract a randomforest model
    rf_model = RandomForestRegressor(n_estimators=100)

    #fit the regression kriging model
    m_rk = RegressionKriging(regression_model=rf_model,
           n_closest_points=10, method= 'ordinary',
           variogram_model= 'exponential')
    m_rk.fit(p, x, target)

    #import gridded 9m xyz values for prediction
    #grd = pd.read_csv("9m_dem_xyz.csv")
    grd = pd.read_csv("9m_dem_xyz.csv")

    ##numpy array of coordinates
```

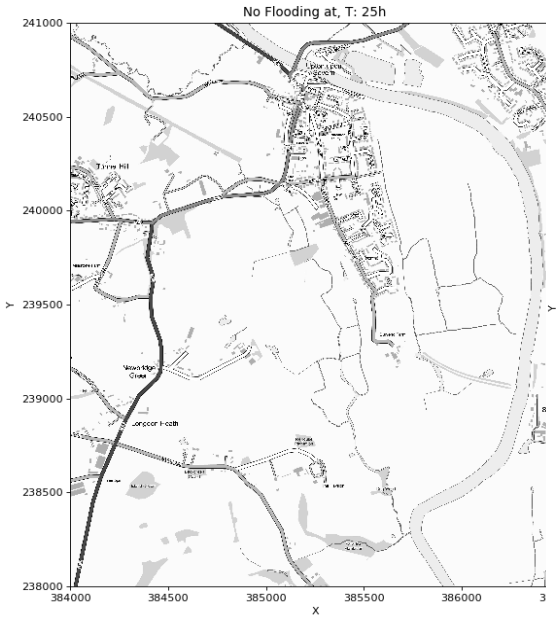
```
        x_grd = grd.to_numpy()
        x_y = x_grd[:, :-1]

        ##numpy array of surface elevation
        z = x_grd[:, -1:]

        ##get X and Y as a separate array for geopandas dataframe

        xi = grd['x']
        yi = grd['y']

        pred = m_rk.predict(z, x_y)


        #filter values
        pred[pred > 1] = 1
        pred[pred < 0] = 0

        d = {'X': xi,'Y': yi, 'Prob': pred}
        df = pd.DataFrame(data=d)
        #df.to_csv("PyKrig_Ioutput_9m.csv")

        ##create geodataframe (similar to spatial pixeldataframe in R)

        geom = [Point(xy) for xy in zip(df.X, df.Y)]
        df1 = df.drop(['X', 'Y'], axis=1)
        crs = {'init': 'epsg:27700'}
        gdf = geopandas.GeoDataFrame(df1, crs=crs, geometry=geom)
        gdf.Prob[gdf.Prob < 0.2]= np.nan


        fig,    ax    =    plt.subplots(figsize=(10,    10),    dpi=80,
facecolor='w', edgecolor='k')
        ax.imshow(red, extent=bounds, cmap = 'gray')
        gdf.plot(column='Prob',ax=ax,                        alpha=0.5,
cmap='Blues',legend=True, marker='*', markersize=7)
        plt.xlabel('X')
        plt.ylabel('Y')
        plt.title('Flood Extent: T=%i h'%op)
        display.display(plt.gcf())
        plt.savefig('Hour_{}.png'.format(op))
        display.clear_output(wait=True)
        time.sleep(2)

    else:
        plt.figure(num=None, figsize=(10, 10), dpi=80, facecolor='w',
edgecolor='k')
        plt.imshow(red, extent=bounds, cmap = 'gray')
        plt.xlabel('X')
        plt.ylabel('Y')
        plt.title("No Flooding at, T: {}h".format(op))
        display.display(plt.gcf())
        plt.savefig('Hour_{}.png'.format(op))
        display.clear_output(wait=True)
```

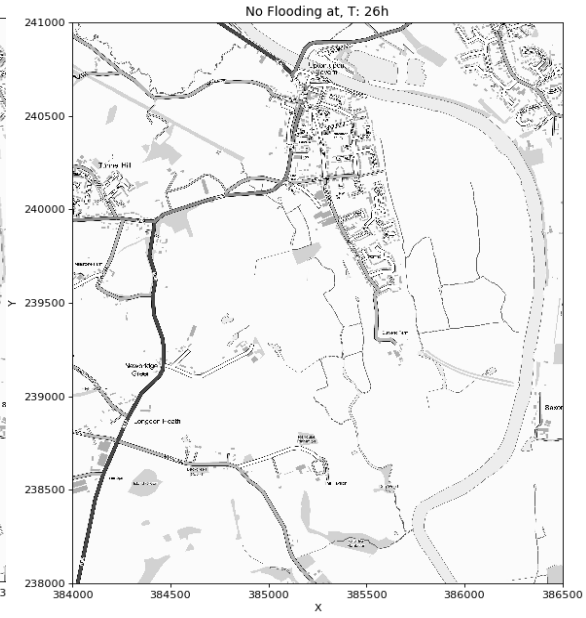
```
        time.sleep(2)

    t = t+3600

    elapsed_time = time.time() - start_time

    print("Total time required to complete this simulation: {} seconds".format(elapsed_time))
```

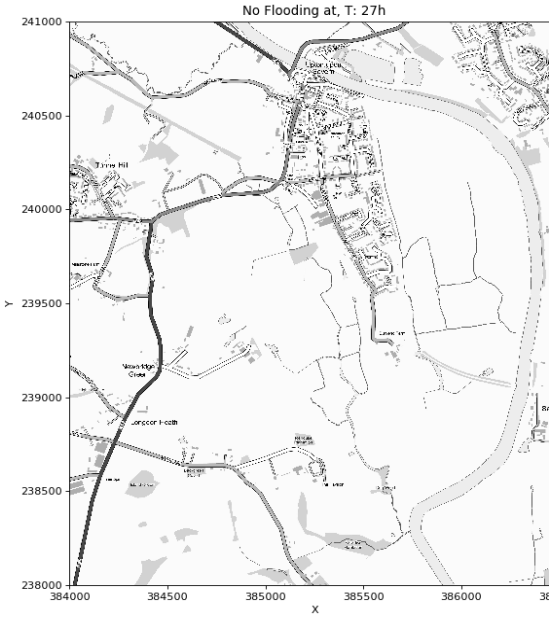
Upon running the script, loading the trained model will take significant amount of time. Once loaded the system will start generating inundation map and stepwise simulation time to generate maps will be displayed:

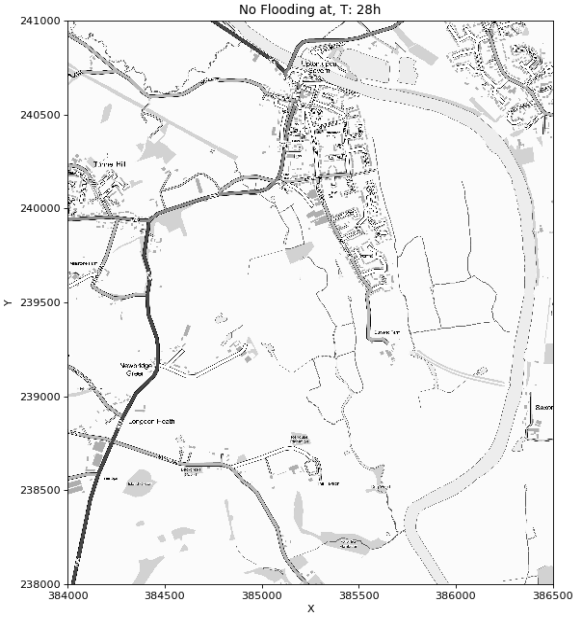
Example:

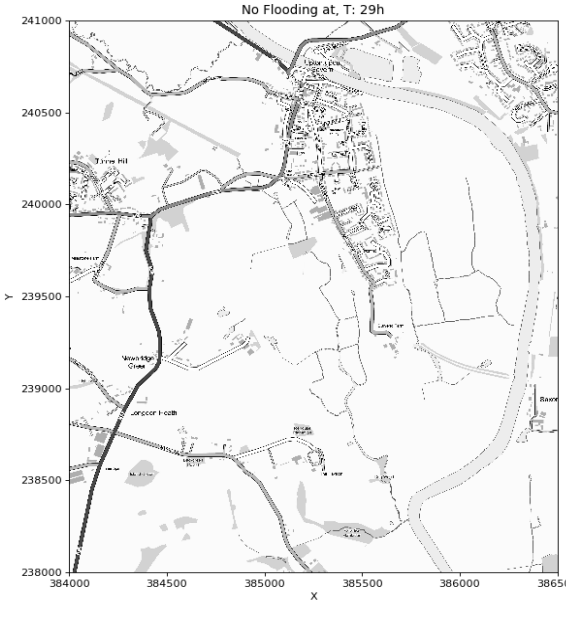
```
Total time required to complete this simulation: 13.7574145793914 seconds
Finished learning regression model
```

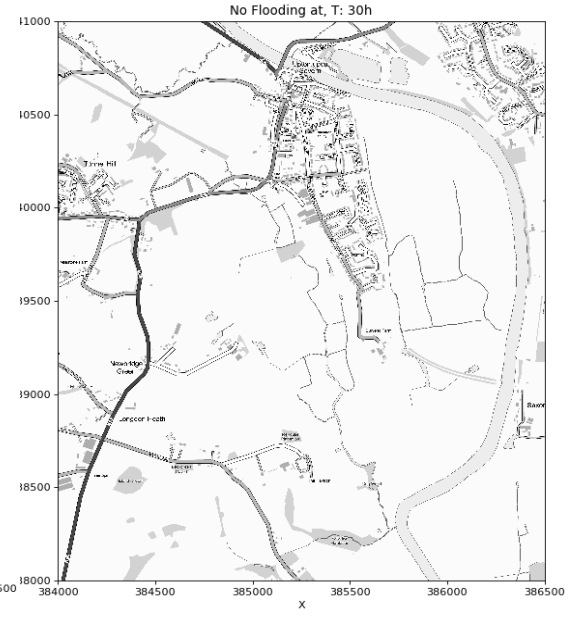

No Flooding at, T: 25h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
X

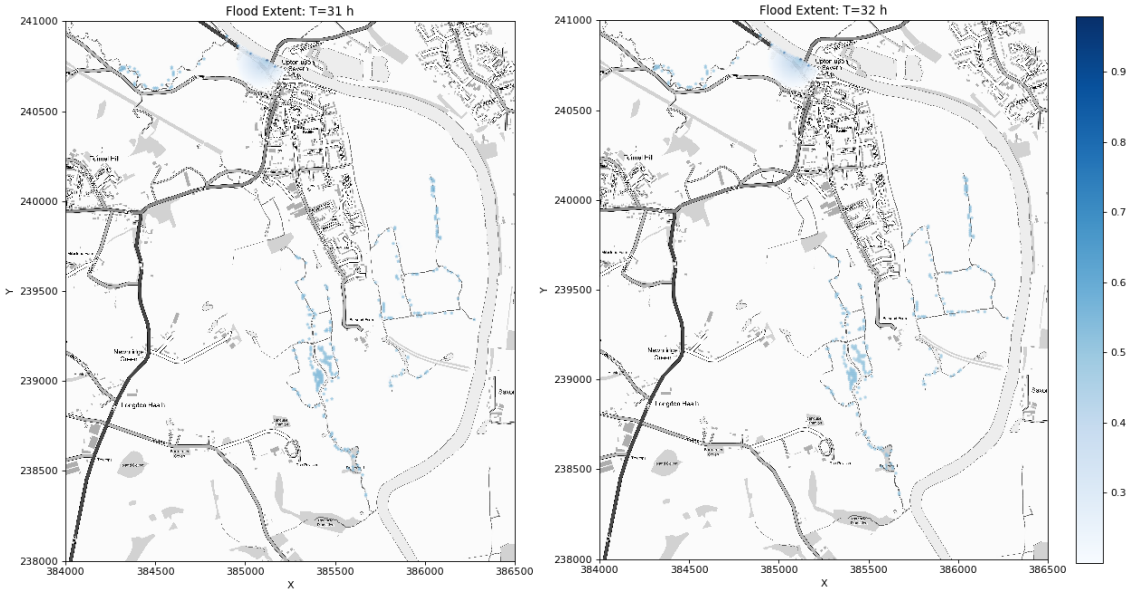

No Flooding at, T: 26h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X

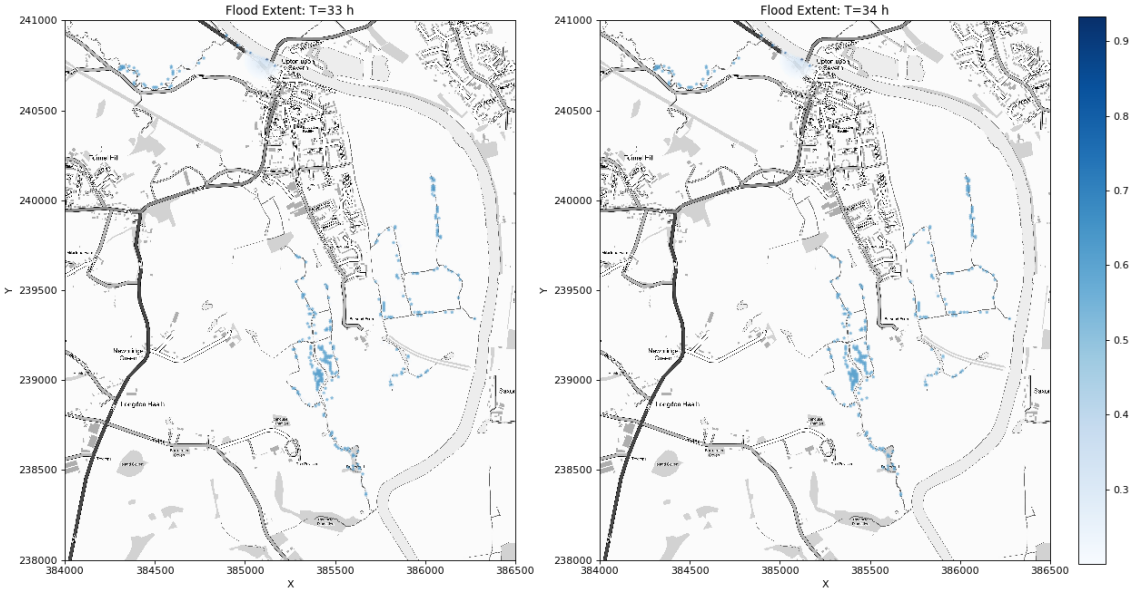

No Flooding at, T: 27h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
X

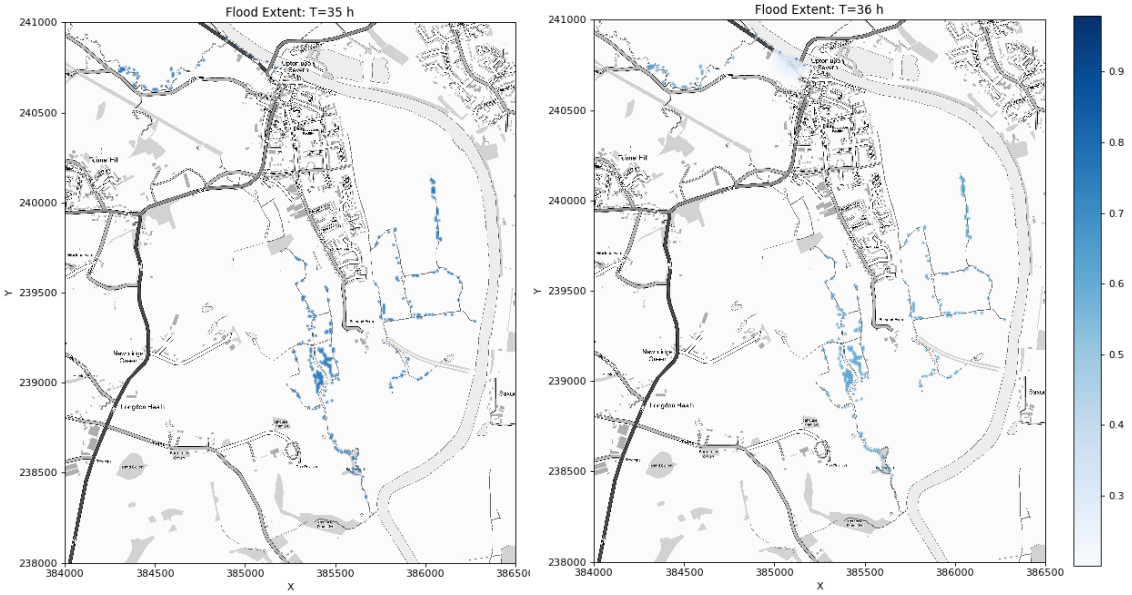

No Flooding at, T: 28h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X

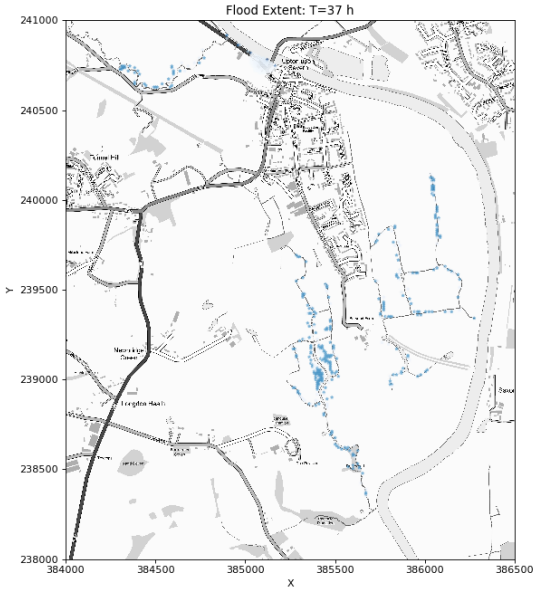

No Flooding at, T: 29h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X

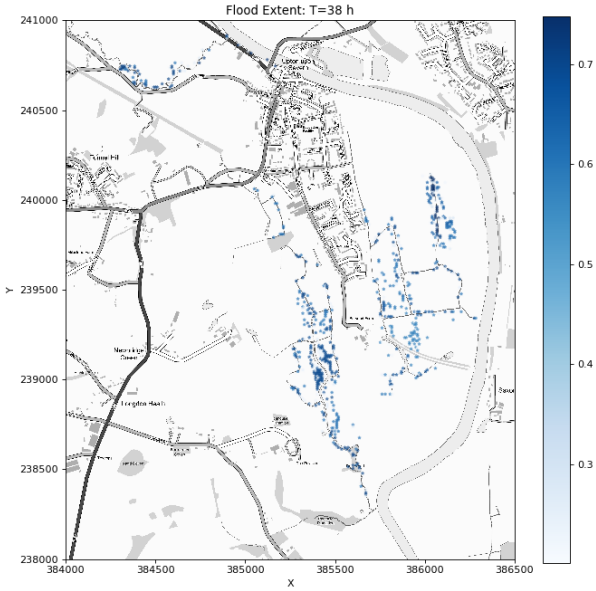

No Flooding at, T: 30h
384000
384500
385000
385500
386000
386500
X

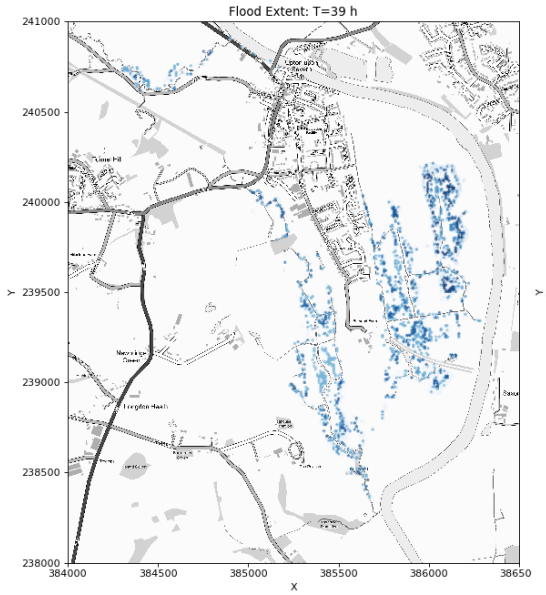

Flood Extent: T=31 h
Flood Extent: T=32 h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X
0.9
0.8
0.7
0.6
0.5
0.4
0.3

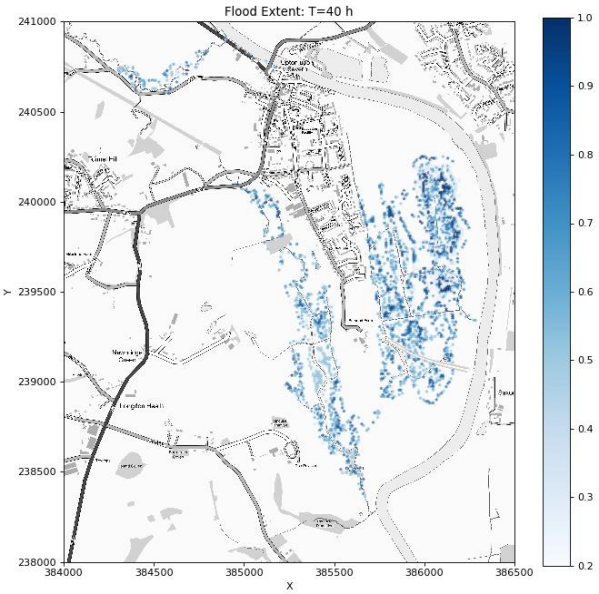

Flood Extent: T=33 h
Flood Extent: T=34 h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X
0.9
0.8
0.7
0.6
0.5
0.4
0.3

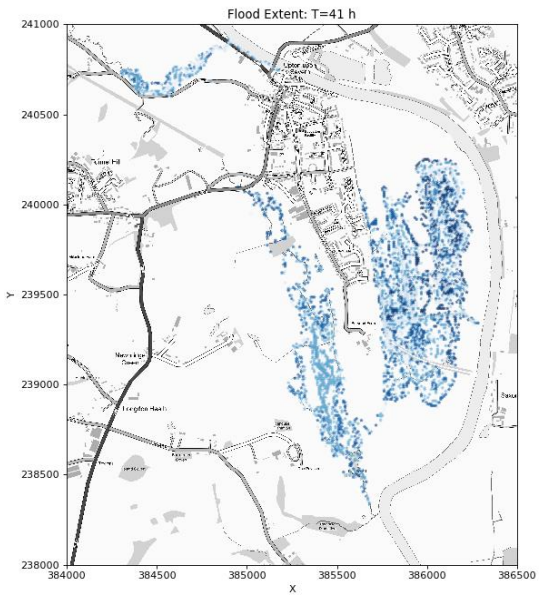

Flood Extent: T=35 h
Flood Extent: T=36 h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X
0.9
0.8
0.7
0.6
0.5
0.4
0.3

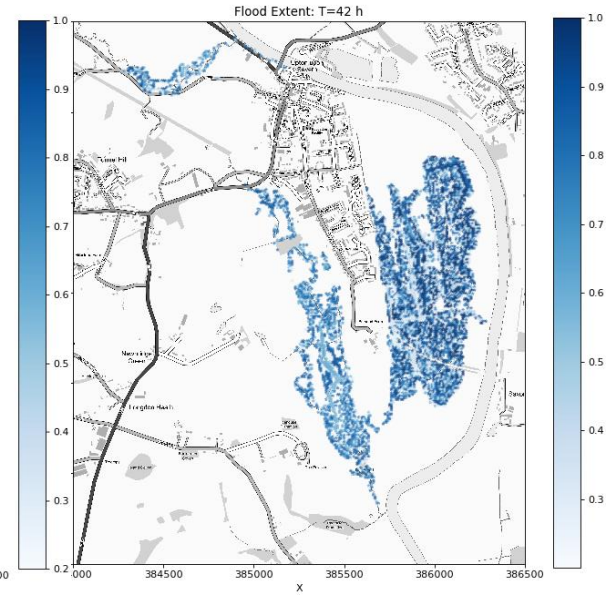

Flood Extent: T=37 h
Y
X

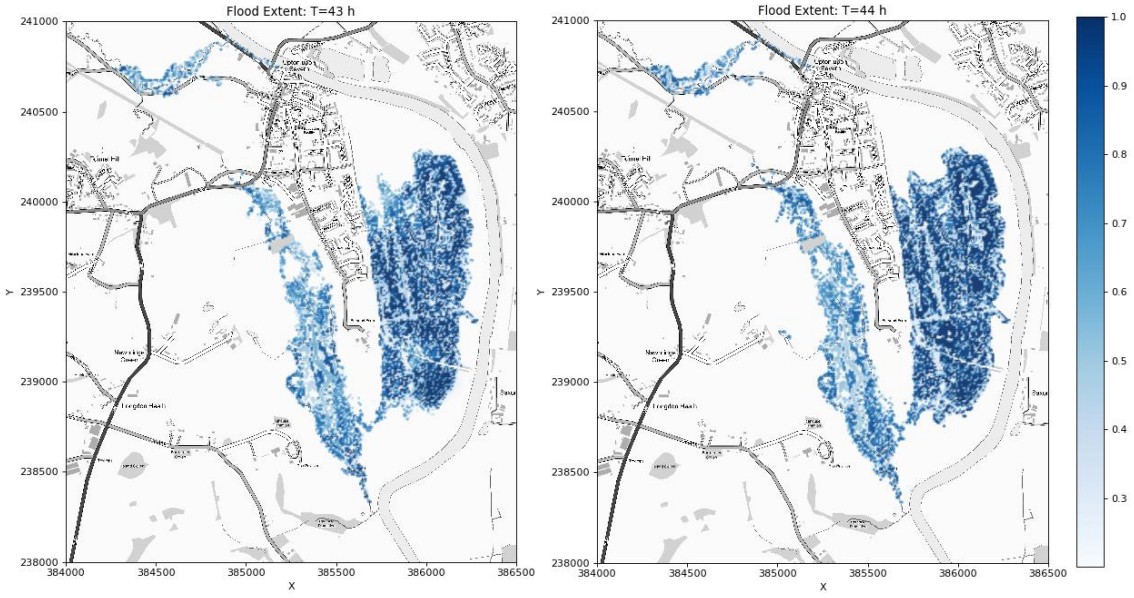

Flood Extent: T=38 h
Y
X

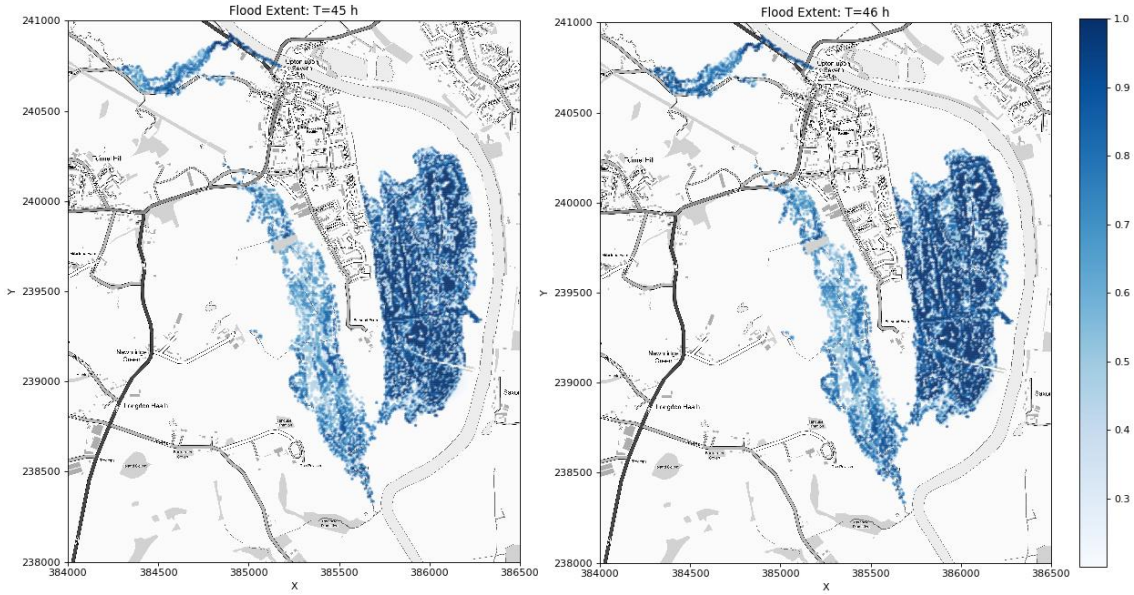

Flood Extent: T=39 h
Y
X

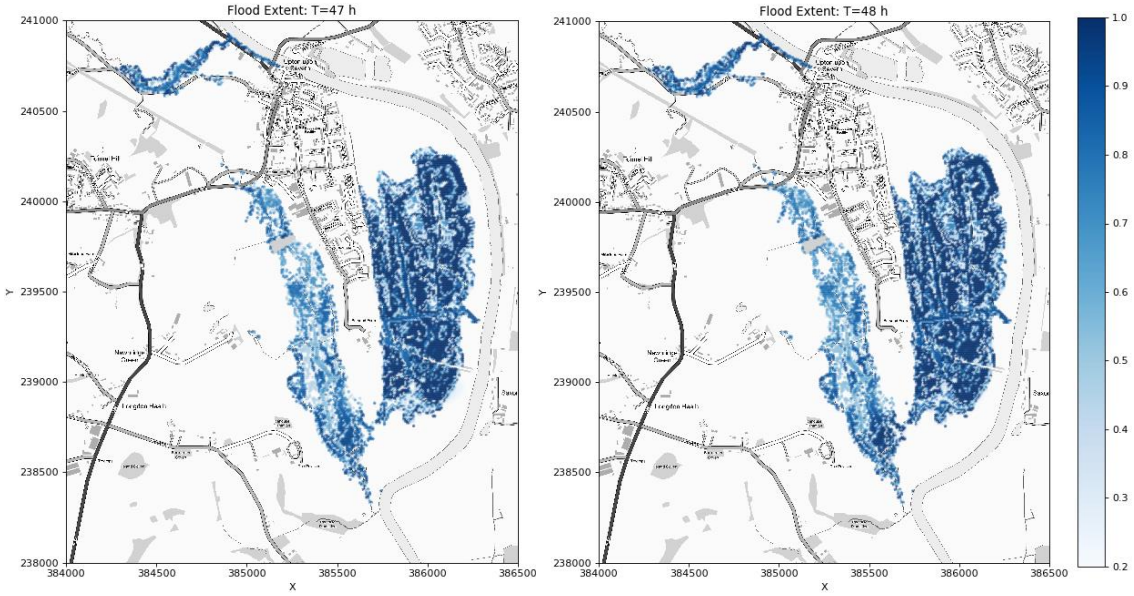

Flood Extent: T=40 h
Y
X

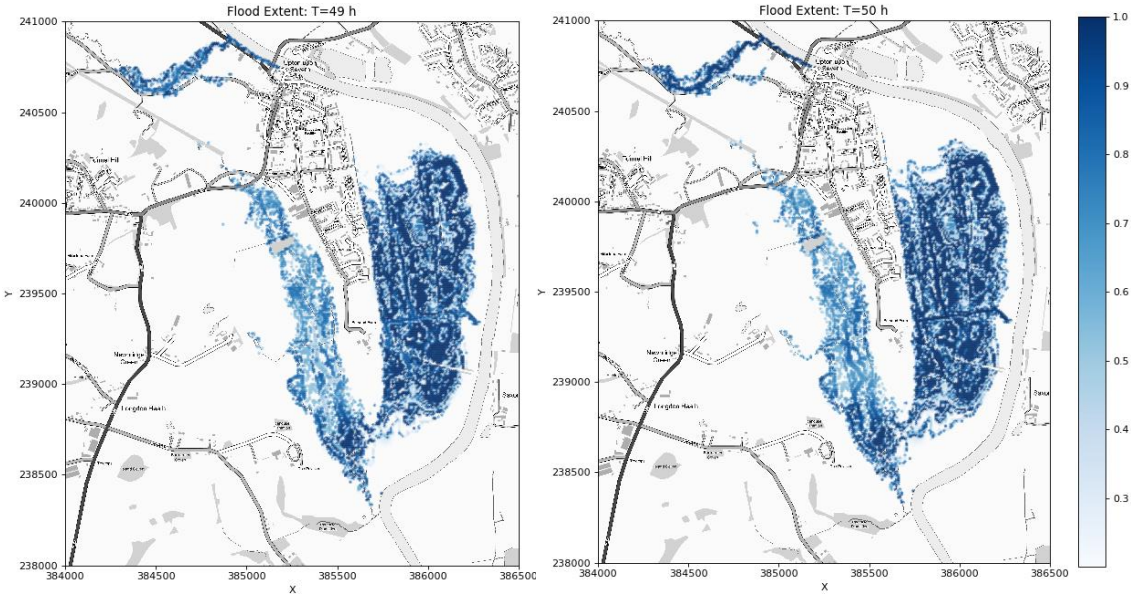

Flood Extent: T=41 h
Y
X

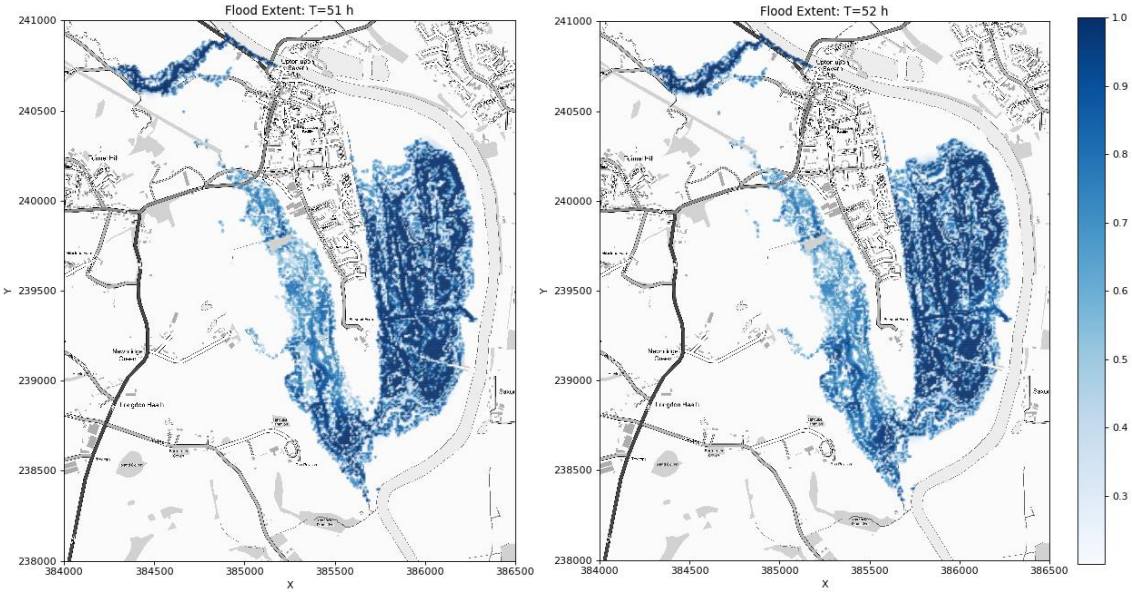

Flood Extent: T=42 h
Y
X

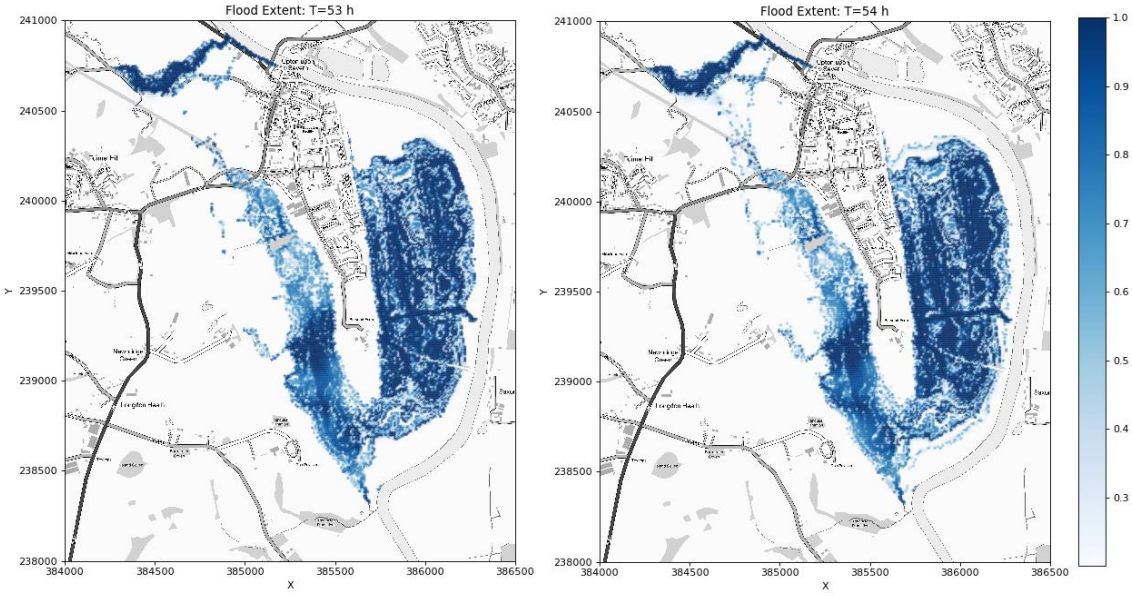

Flood Extent: T=43 h
Flood Extent: T=44 h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

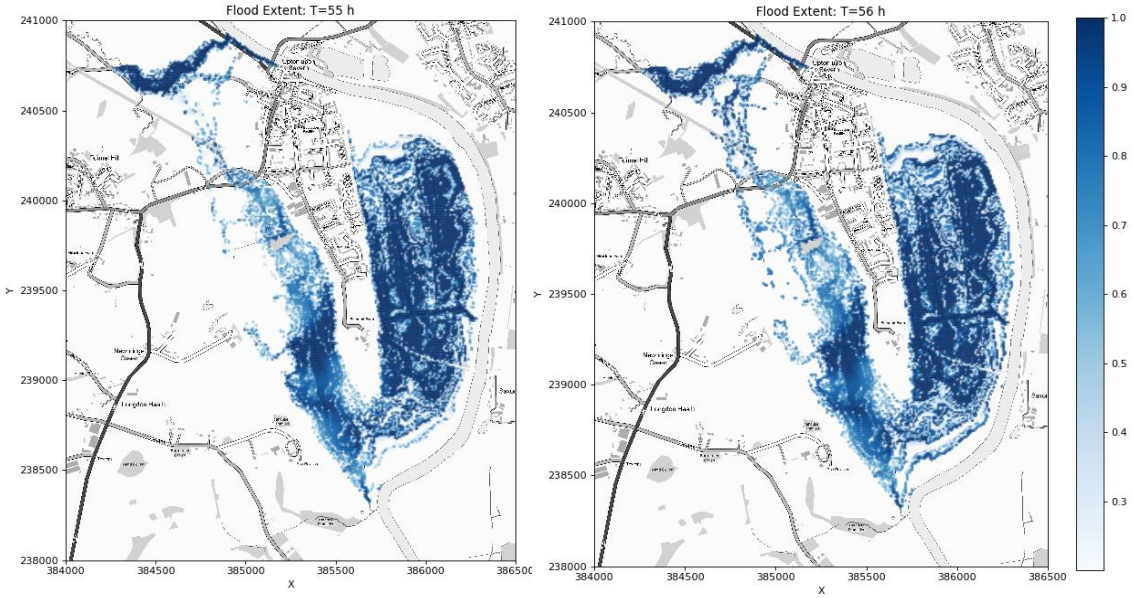

Flood Extent: T=45 h
Flood Extent: T=46 h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

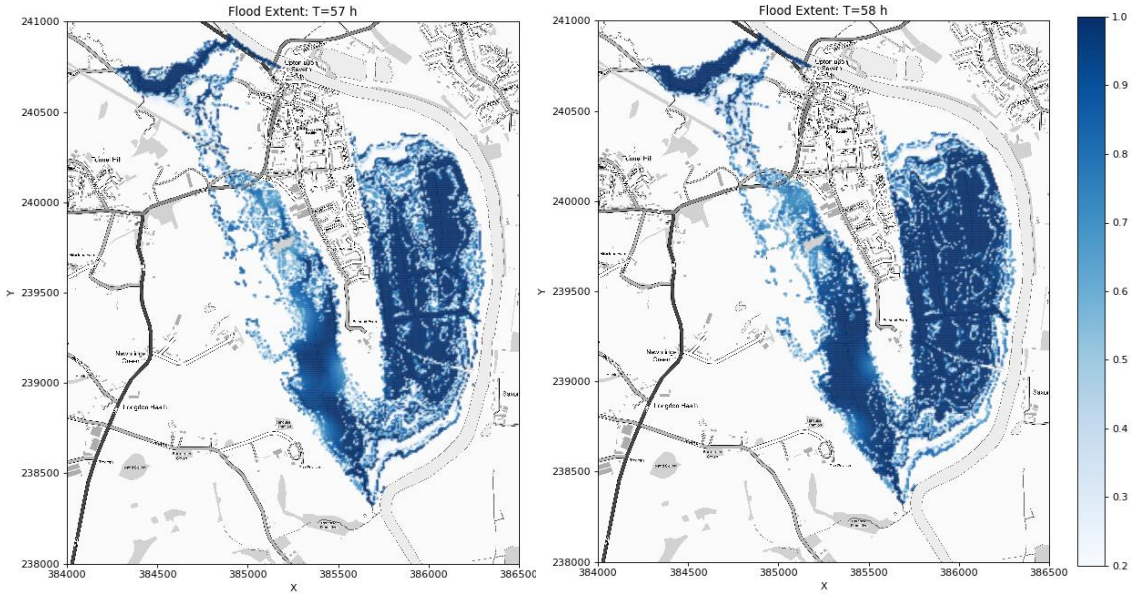

Flood Extent: T=47 h
Flood Extent: T=48 h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

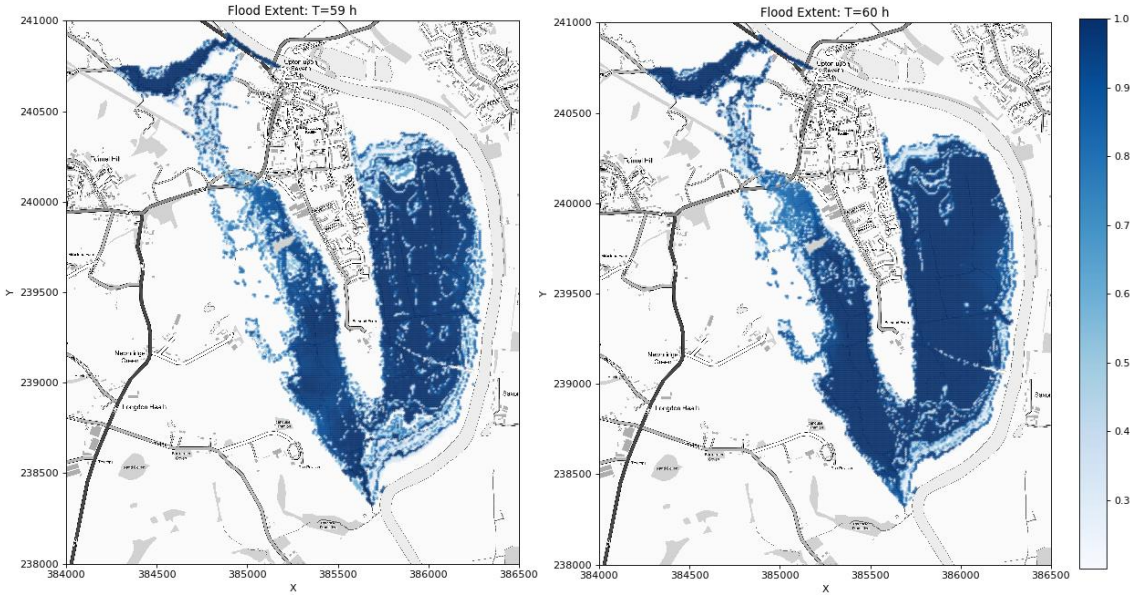

Flood Extent: T=49 h
Flood Extent: T=50 h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

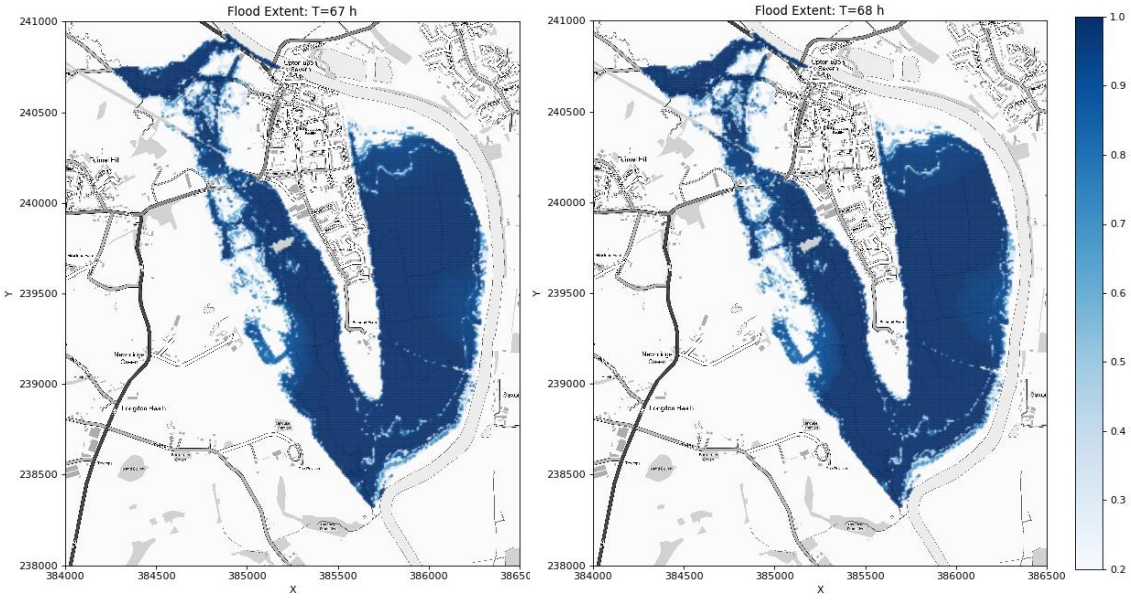

Flood Extent: T=51 h
Flood Extent: T=52 h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

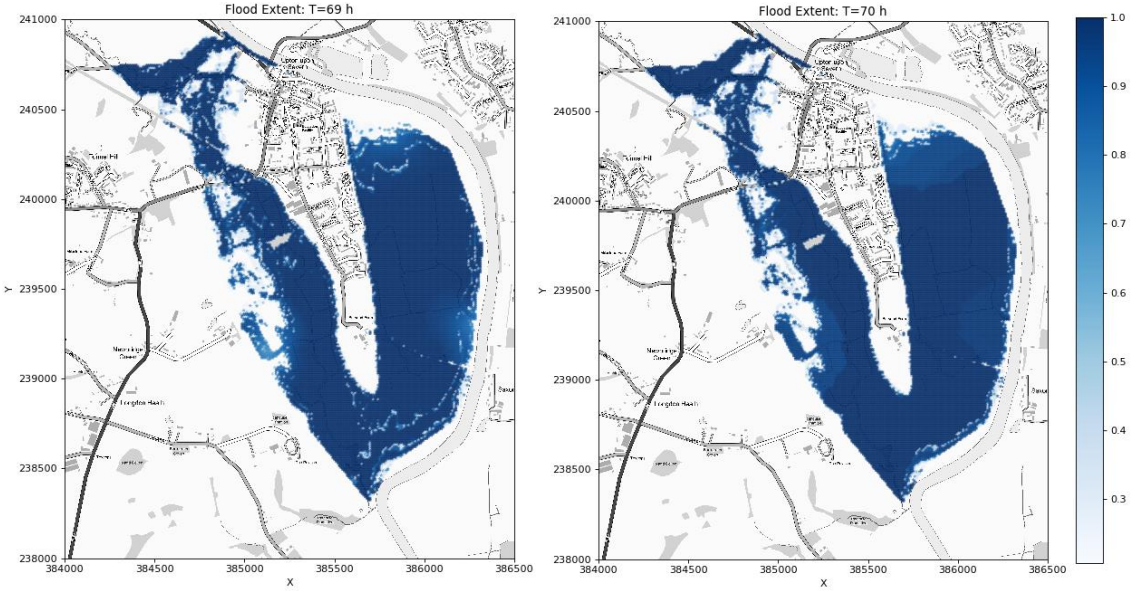

Flood Extent: T=53 h
Flood Extent: T=54 h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

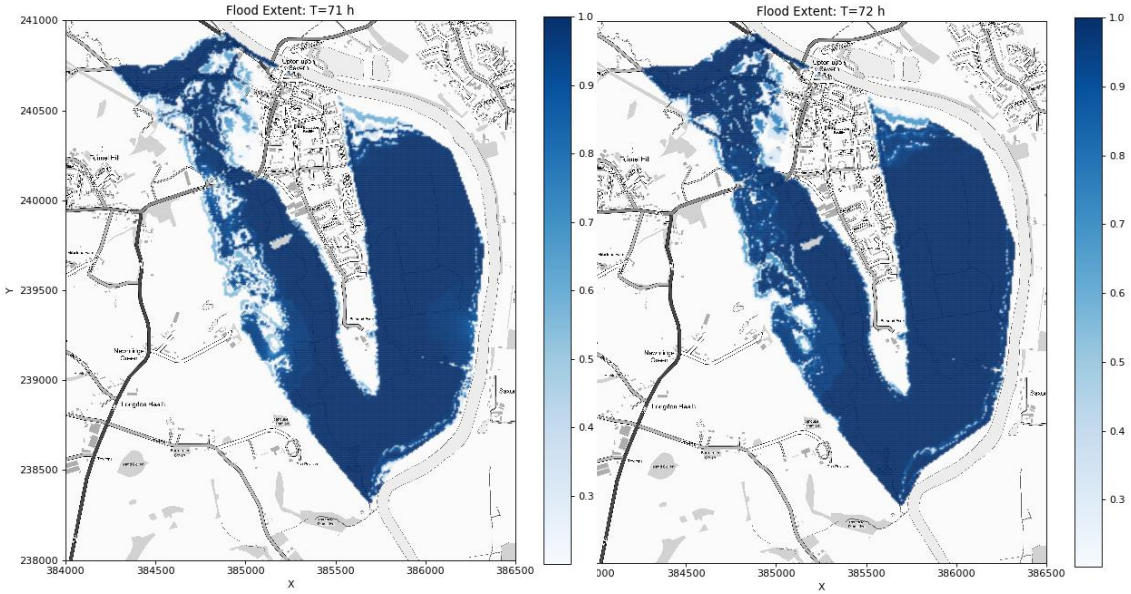

Flood Extent: T=55 h
Flood Extent: T=56 h
241000
240500
240000
239500
239000
238500
238000
384000
384500
385000
385500
386000
386500
X
Y
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3


Flood Extent: T=57 h
Flood Extent: T=58 h
241000
240500
240000
239500
239000
238500
238000
384000
384500
385000
385500
386000
386500
X
Y
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2


Flood Extent: T=59 h
Flood Extent: T=60 h
241000
240500
240000
239500
239000
238500
238000
384000
384500
385000
385500
386000
386500
X
Y
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

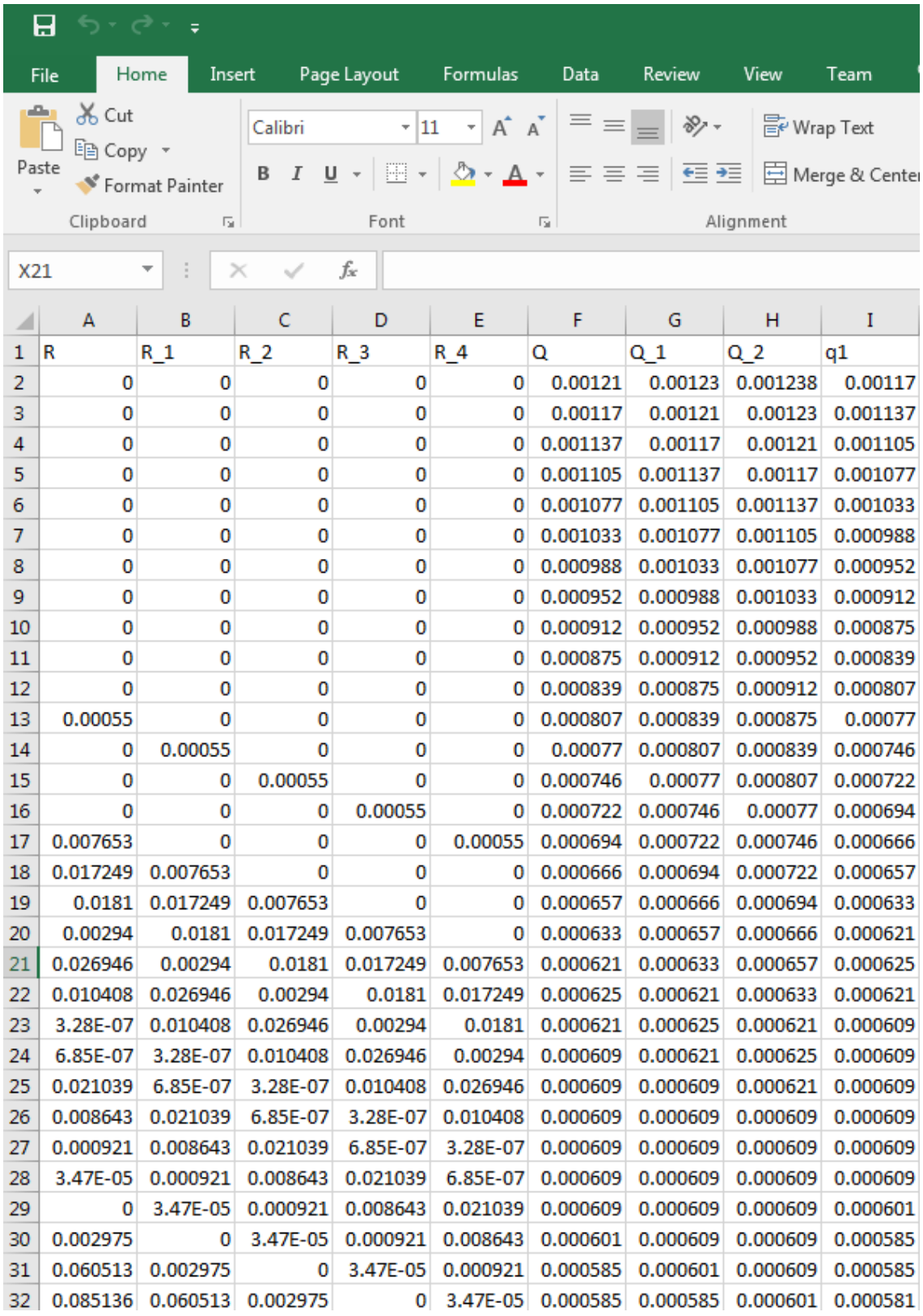

Flood Extent: T=67 h
Flood Extent: T=68 h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
X
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2


Flood Extent: T=69 h
Flood Extent: T=70 h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3


Flood Extent: T=71 h
Flood Extent: T=72 h
241000
240500
240000
239500
239000
238500
238000
Y
384000
384500
385000
385500
386000
386500
X
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

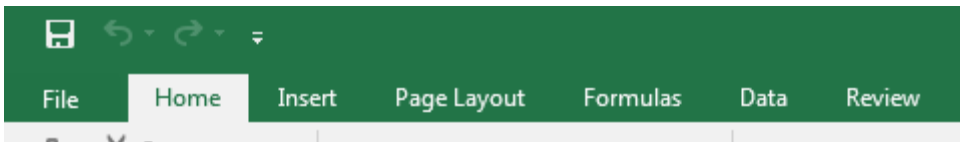
Appendix E

Sample training data- Chapter 5



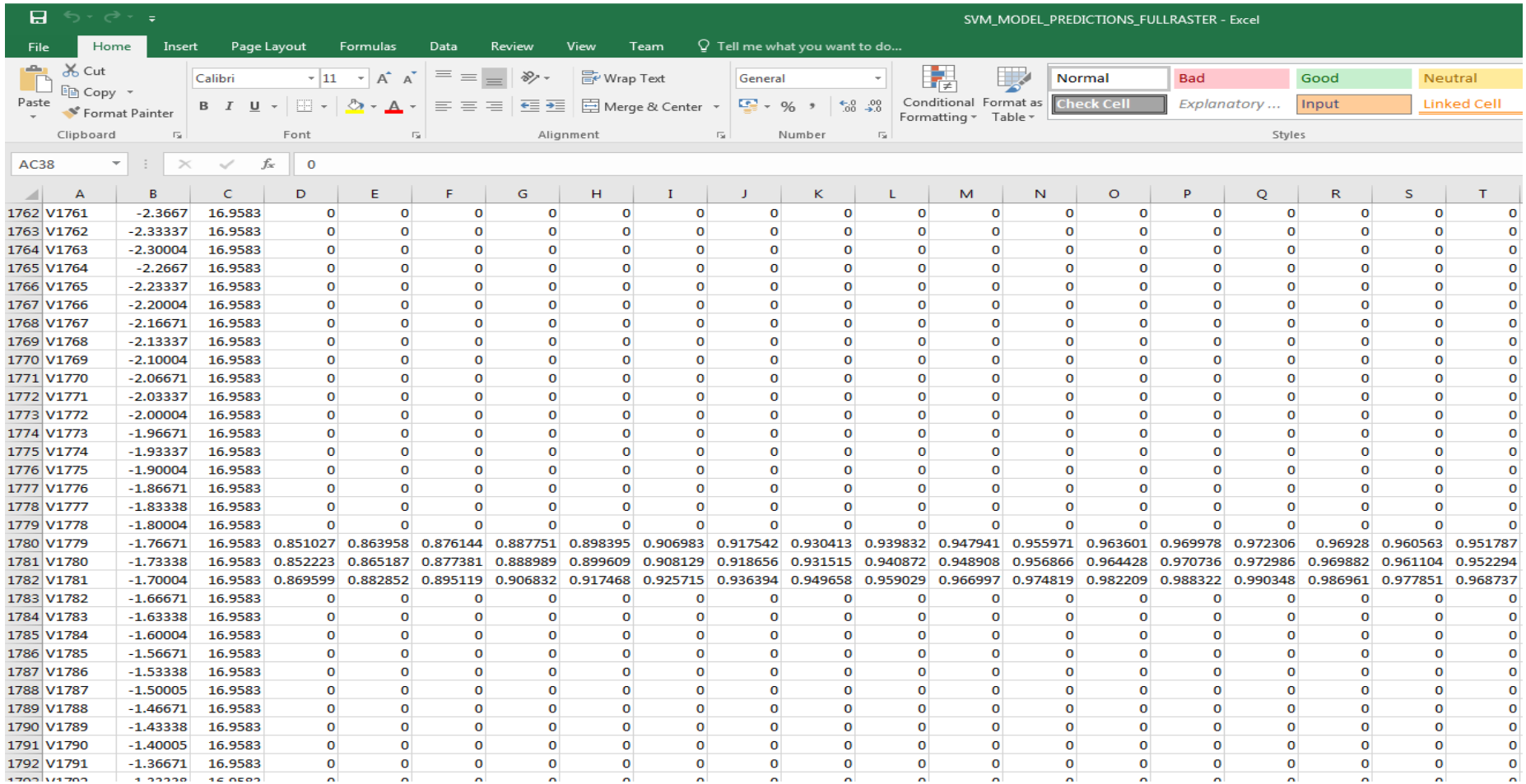
	A	B	C	D	E	F	G	H	I
1	R	R_1	R_2	R_3	R_4	Q	Q_1	Q_2	q1
2	0	0	0	0	0	0.00121	0.00123	0.001238	0.00117
3	0	0	0	0	0	0.00117	0.00121	0.00123	0.001137
4	0	0	0	0	0	0.001137	0.00117	0.00121	0.001105
5	0	0	0	0	0	0.001105	0.001137	0.00117	0.001077
6	0	0	0	0	0	0.001077	0.001105	0.001137	0.001033
7	0	0	0	0	0	0.001033	0.001077	0.001105	0.000988
8	0	0	0	0	0	0.000988	0.001033	0.001077	0.000952
9	0	0	0	0	0	0.000952	0.000988	0.001033	0.000912
10	0	0	0	0	0	0.000912	0.000952	0.000988	0.000875
11	0	0	0	0	0	0.000875	0.000912	0.000952	0.000839
12	0	0	0	0	0	0.000839	0.000875	0.000912	0.000807
13	0.00055	0	0	0	0	0.000807	0.000839	0.000875	0.00077
14	0	0.00055	0	0	0	0.00077	0.000807	0.000839	0.000746
15	0	0	0.00055	0	0	0.000746	0.00077	0.000807	0.000722
16	0	0	0	0.00055	0	0.000722	0.000746	0.00077	0.000694
17	0.007653	0	0	0	0.00055	0.000694	0.000722	0.000746	0.000666
18	0.017249	0.007653	0	0	0	0.000666	0.000694	0.000722	0.000657
19	0.0181	0.017249	0.007653	0	0	0.000657	0.000666	0.000694	0.000633
20	0.00294	0.0181	0.017249	0.007653	0	0.000633	0.000657	0.000666	0.000621
21	0.026946	0.00294	0.0181	0.017249	0.007653	0.000621	0.000633	0.000657	0.000625
22	0.010408	0.026946	0.00294	0.0181	0.017249	0.000625	0.000621	0.000633	0.000621
23	3.28E-07	0.010408	0.026946	0.00294	0.0181	0.000621	0.000625	0.000621	0.000609
24	6.85E-07	3.28E-07	0.010408	0.026946	0.00294	0.000609	0.000621	0.000625	0.000609
25	0.021039	6.85E-07	3.28E-07	0.010408	0.026946	0.000609	0.000609	0.000621	0.000609
26	0.008643	0.021039	6.85E-07	3.28E-07	0.010408	0.000609	0.000609	0.000609	0.000609
27	0.000921	0.008643	0.021039	6.85E-07	3.28E-07	0.000609	0.000609	0.000609	0.000609
28	3.47E-05	0.000921	0.008643	0.021039	6.85E-07	0.000609	0.000609	0.000609	0.000609
29	0	3.47E-05	0.000921	0.008643	0.021039	0.000609	0.000609	0.000609	0.000601
30	0.002975	0	3.47E-05	0.000921	0.008643	0.000601	0.000609	0.000609	0.000585
31	0.060513	0.002975	0	3.47E-05	0.000921	0.000585	0.000601	0.000609	0.000585
32	0.085136	0.060513	0.002975	0	3.47E-05	0.000585	0.000585	0.000601	0.000581

Sample training data for the MLP-based model- Chapter 6



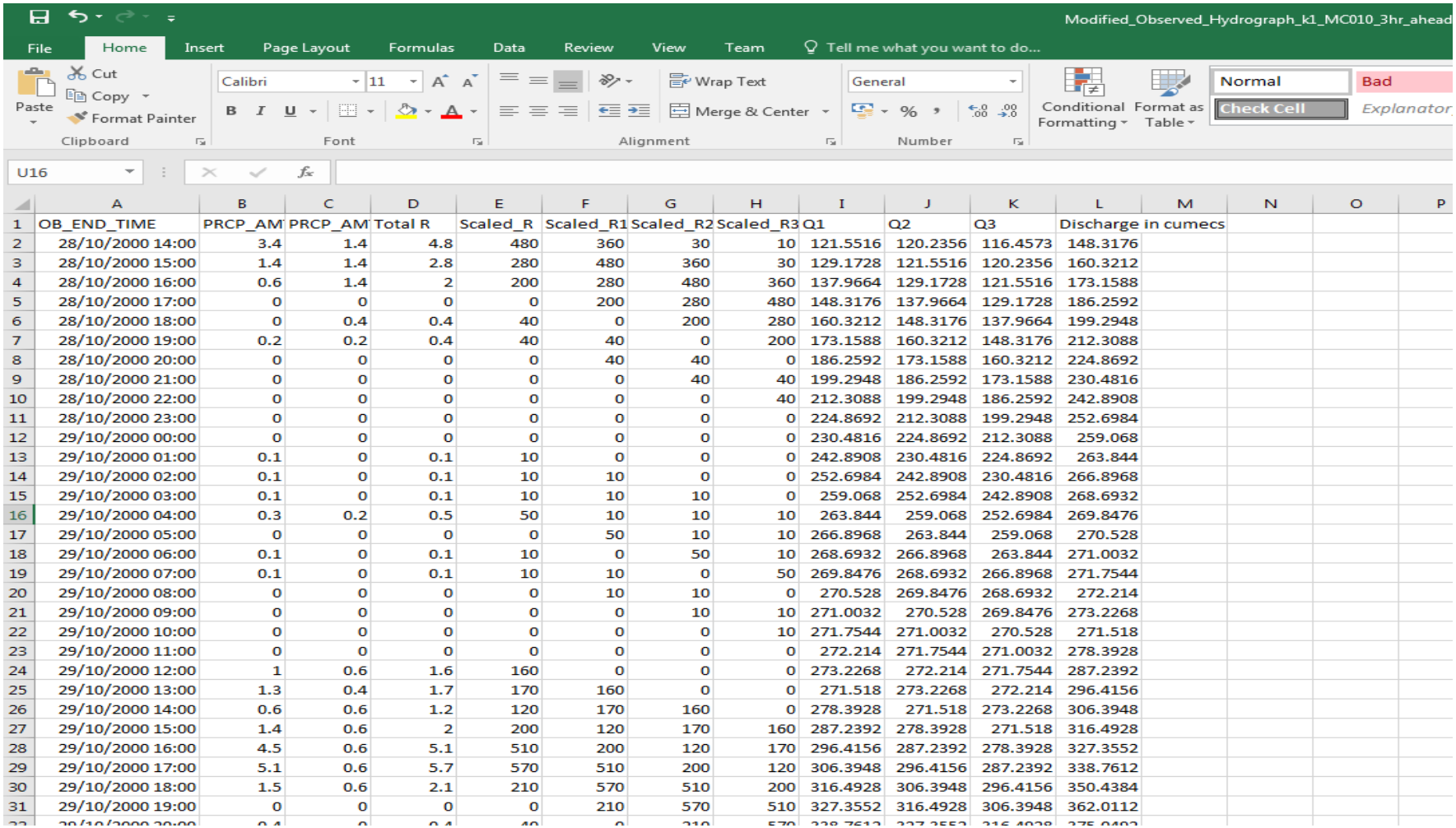
	A	B	C	D	E	F	G
1	Q1	Q11	Q12	Q2	Q21	Q22	Day
2	40.72	41.32	42.79	59.59	61.71	64.17	1
3	40.69	40.72	41.32	58.09	59.59	61.71	2
4	39.95	40.69	40.72	57.27	58.09	59.59	3
5	39.65	39.95	40.69	57.22	57.27	58.09	4
6	39.65	39.65	39.95	56.93	57.22	57.27	5
7	39.68	39.65	39.65	54.85	56.93	57.22	6
8	40.42	39.68	39.65	52.18	54.85	56.93	7
9	40.72	40.42	39.68	50.42	52.18	54.85	8
10	40.65	40.72	40.42	48.52	50.42	52.18	9
11	39.18	40.65	40.72	47.3	48.52	50.42	10
12	40.06	39.18	40.65	45.72	47.3	48.52	11
13	72.66	40.06	39.18	44.8	45.72	47.3	12
14	85.33	72.66	40.06	43.66	44.8	45.72	13
15	70.7	85.33	72.66	42.16	43.66	44.8	14
16	66.06	70.7	85.33	40.28	42.16	43.66	15
17	49.79	66.06	70.7	39.39	40.28	42.16	16
18	41.26	49.79	66.06	38.53	39.39	40.28	17
19	39.18	41.26	49.79	37.48	38.53	39.39	18
20	38.58	39.18	41.26	36.19	37.48	38.53	19
21	38.58	38.58	39.18	34.79	36.19	37.48	20
22	38.29	38.58	38.58	33.34	34.79	36.19	21
23	32.58	38.29	38.58	31.97	33.34	34.79	22
24	44.58	32.58	38.29	31.37	31.97	33.34	23
25	54.14	44.58	32.58	30.78	31.37	31.97	24
26	55.72	54.14	44.58	30.06	30.78	31.37	25
27	57.55	55.72	54.14	29.25	30.06	30.78	26
28	58.05	57.55	55.72	28.46	29.25	30.06	27
29	52.56	58.05	57.55	27.78	28.46	29.25	28
30	49.95	52.56	58.05	27.17	27.78	28.46	29
31	41.88	49.95	52.56	26.69	27.17	27.78	30

Output of the SVM model (Chapter 6)- 1st column represents the location, 2nd and 3rd column represent coordinates of the location. 4th column onwards represent water level at daily time-step.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1762	V1761	-2.3667	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1763	V1762	-2.33337	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1764	V1763	-2.30004	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1765	V1764	-2.2667	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1766	V1765	-2.23337	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1767	V1766	-2.20004	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1768	V1767	-2.16671	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1769	V1768	-2.13337	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1770	V1769	-2.10004	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1771	V1770	-2.06671	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1772	V1771	-2.03337	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1773	V1772	-2.00004	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1774	V1773	-1.96671	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1775	V1774	-1.93337	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1776	V1775	-1.90004	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1777	V1776	-1.86671	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1778	V1777	-1.83338	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1779	V1778	-1.80004	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1780	V1779	-1.76671	16.9583	0.851027	0.863958	0.876144	0.887751	0.898395	0.906983	0.917542	0.930413	0.939832	0.947941	0.955971	0.963601	0.969978	0.972306	0.96928	0.960563	0.951787
1781	V1780	-1.73338	16.9583	0.852223	0.865187	0.877381	0.888989	0.899609	0.908129	0.918656	0.931515	0.940872	0.948908	0.956866	0.964428	0.970736	0.972986	0.969882	0.961104	0.952294
1782	V1781	-1.70004	16.9583	0.869599	0.882852	0.895119	0.906832	0.917468	0.925715	0.936394	0.949658	0.959029	0.966997	0.974819	0.982209	0.988322	0.990348	0.986961	0.977851	0.968737
1783	V1782	-1.66671	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1784	V1783	-1.63338	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1785	V1784	-1.60004	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1786	V1785	-1.56671	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1787	V1786	-1.53338	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1788	V1787	-1.50005	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1789	V1788	-1.46671	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1790	V1789	-1.43338	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1791	V1790	-1.40005	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1792	V1791	-1.36671	16.9583	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sample training data for the RF based hydrological model (Chapter 8). Average rainfall data is scaled up by a factor of 10.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	OB_END_TIME	PRCP_AM	PRCP_AM	Total R	Scaled_R	Scaled_R1	Scaled_R2	Scaled_R3	Q1	Q2	Q3	Discharge in cumecs				
2	28/10/2000 14:00	3.4	1.4	4.8	480	360	30	10	121.5516	120.2356	116.4573	148.3176				
3	28/10/2000 15:00	1.4	1.4	2.8	280	480	360	30	129.1728	121.5516	120.2356	160.3212				
4	28/10/2000 16:00	0.6	1.4	2	200	280	480	360	137.9664	129.1728	121.5516	173.1588				
5	28/10/2000 17:00	0	0	0	0	200	280	480	148.3176	137.9664	129.1728	186.2592				
6	28/10/2000 18:00	0	0.4	0.4	40	0	200	280	160.3212	148.3176	137.9664	199.2948				
7	28/10/2000 19:00	0.2	0.2	0.4	40	40	0	200	173.1588	160.3212	148.3176	212.3088				
8	28/10/2000 20:00	0	0	0	0	40	40	0	186.2592	173.1588	160.3212	224.8692				
9	28/10/2000 21:00	0	0	0	0	0	40	40	199.2948	186.2592	173.1588	230.4816				
10	28/10/2000 22:00	0	0	0	0	0	0	40	212.3088	199.2948	186.2592	242.8908				
11	28/10/2000 23:00	0	0	0	0	0	0	0	224.8692	212.3088	199.2948	252.6984				
12	29/10/2000 00:00	0	0	0	0	0	0	0	230.4816	224.8692	212.3088	259.068				
13	29/10/2000 01:00	0.1	0	0.1	10	0	0	0	242.8908	230.4816	224.8692	263.844				
14	29/10/2000 02:00	0.1	0	0.1	10	10	0	0	252.6984	242.8908	230.4816	266.8968				
15	29/10/2000 03:00	0.1	0	0.1	10	10	10	0	259.068	252.6984	242.8908	268.6932				
16	29/10/2000 04:00	0.3	0.2	0.5	50	10	10	10	263.844	259.068	252.6984	269.8476				
17	29/10/2000 05:00	0	0	0	0	50	10	10	266.8968	263.844	259.068	270.528				
18	29/10/2000 06:00	0.1	0	0.1	10	0	50	10	268.6932	266.8968	263.844	271.0032				
19	29/10/2000 07:00	0.1	0	0.1	10	10	0	50	269.8476	268.6932	266.8968	271.7544				
20	29/10/2000 08:00	0	0	0	0	10	10	0	270.528	269.8476	268.6932	272.214				
21	29/10/2000 09:00	0	0	0	0	0	10	10	271.0032	270.528	269.8476	273.2268				
22	29/10/2000 10:00	0	0	0	0	0	0	10	271.7544	271.0032	270.528	271.518				
23	29/10/2000 11:00	0	0	0	0	0	0	0	272.214	271.7544	271.0032	278.3928				
24	29/10/2000 12:00	1	0.6	1.6	160	0	0	0	273.2268	272.214	271.7544	287.2392				
25	29/10/2000 13:00	1.3	0.4	1.7	170	160	0	0	271.518	273.2268	272.214	296.4156				
26	29/10/2000 14:00	0.6	0.6	1.2	120	170	160	0	278.3928	271.518	273.2268	306.3948				
27	29/10/2000 15:00	1.4	0.6	2	200	120	170	160	287.2392	278.3928	271.518	316.4928				
28	29/10/2000 16:00	4.5	0.6	5.1	510	200	120	170	296.4156	287.2392	278.3928	327.3552				
29	29/10/2000 17:00	5.1	0.6	5.7	570	510	200	120	306.3948	296.4156	287.2392	338.7612				
30	29/10/2000 18:00	1.5	0.6	2.1	210	570	510	200	316.4928	306.3948	296.4156	350.4384				
31	29/10/2000 19:00	0	0	0	0	210	570	510	327.3552	316.4928	306.3948	362.0112				

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Kaiser, L., Kudlur, M., Levenberg, J. & Zheng, X. (2015) TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems.

Abe, S. (2010) Function Approximation. In *Support Vector Machines for Pattern Classification*.) Springer London, London, pp. 395-442.

Abrahart, R. J., See, L. M. & Dawson, C. W. (2008) Neural Network Hydroinformatics: Maintaining Scientific Rigour. In *Practical Hydroinformatics: Computational Intelligence and Technological Developments in Water Applications*. (Abrahart, R. J., See, L. M., andSolomatine, D. P. (eds)) Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 33-47.

Akhtar, M. K., Corzo, G. A., Van Andel, S. J. & Jonoski, A. (2009) River Flow Forecasting With Artificial Neural Networks Using Satellite Observed Precipitation Pre-processed With Flow Length and Travel Time Information: Case Study of the Ganges River Basin. *Hydrol. Earth Syst. Sci.* **13(9)**:1607-1618.

Albertbup (2017) A Python implementation of Deep Belief Networks Built Upon NumPy and TensorFlow With scikit-learn compatibility, https://github.com/albertbup/deep-belief-network.

Aldridge, T., Gunawan, O., Moore, R.J., Cole, S.J., and Price, D. (2016) A surface water flooding impact library for flood risk assessment. *E3S Web of Conferences*, 7, 18006.

Almeida, G. a. M. & Bates, P. (2013) Applicability of the Local Inertial Approximation of the Shallow Water Equations to Flood Modeling. *Water Resources Research* **49(8)**:4833-4844.

Altenau, E. H., Pavelsky, T. M., Bates, P. D. & Neal, J. C. (2017) The Effects of Spatial Resolution and Dimensionality on Modeling Regional-scale Hydraulics in a Multichannel River. *Water Resources Research* **53(2)**:1683-1701.

Alvisi, S., Mascellani, G., Franchini, M. & Bárdossy, A. (2006) Water Level Forecasting Through Fuzzy Logic and Artificial Neural Network Approaches. *Hydrol. Earth Syst. Sci.* **10(1)**:1-17.

Anctil, F., Perrin, C. & Andréassian, V. (2004) Impact of the Length of Observed Records on the Performance of ANN and of Conceptual Parsimonious Rainfall-Runoff Forecasting Models. *Environmental Modelling & Software* **19(4)**:357-368.

Anusree, K. & Varghese, K. O. (2016) Streamflow Prediction of Karuvannur River Basin Using ANFIS, ANN and MNLR Models. *Procedia Technology* **24**:101-108.

Arnold, J., Moriasi, D., Gassman, P., Abbaspour, K., White, M., Srinivasan, R., Santhi, C., Harmel, R., Van Griensven, A., Van Liew, M., Kannan, N. & Jha, M. (2012) SWAT: Model Use, Calibration, and Validation **54.**

Aronica, G., Bates, P. D. & Horritt, M. S. (2002) Assessing the Uncertainty in Distributed Model Predictions Using Observed Binary Pattern Information Within GLUE. *Hydrological Processes* **16(10)**:2001-2016.

ASCE (2000a) Artificial Neural Networks in Hydrology. I: Preliminary Concepts. *Journal of Hydrologic Engineering* **5(2)**:115-123.

ASCE (2000b) Artificial Neural Networks in Hydrology. II: Hydrologic Applications. *Journal of Hydrologic Engineering* **5(2)**:124-137.

Asefa, T., Kemblowski, M., Mckee, M. & Khalil, A. (2006) Multi-time Scale Stream Flow Predictions: The Support Vector Machines Approach. *Journal of Hydrology* **318(1)**:7-16.

Badrzadeh, H., Sarukkalige, R. & Jayawardena, A. W. (2015) Hourly Runoff Forecasting for Flood Risk Management: Application of Various Computational Intelligence Models. *Journal of Hydrology* **529, Part 3**:1633-1643.

Bai, Y., Chen, Z., Xie, J. & Li, C. (2016) Daily Reservoir Inflow Forecasting Using Multiscale Deep Feature Learning With Hybrid Models. *Journal of Hydrology* **532(Supplement C)**:193-206.

Bates, P., Trigg, M., Neal, J. & Dabrowa, A. (2013) LISFLOOD-FP User Manual. School of Geographical Sciences, University of Bristol, University Road, Bristol, BS8 1SS, UK.

Bates, P. D. & De Roo, A. P. J. (2000) A Simple Raster-based Model for Flood Inundation Simulation. *Journal of Hydrology* **236(1–2)**:54-77.

Bates, P. D., Horritt, M. S. & Fewtrell, T. J. (2010) A Simple Inertial Formulation of the Shallow Water Equations for Efficient Two-dimensional Flood Inundation Modelling. *Journal of Hydrology* **387(1)**:33-44.

Bates, P. D., Pappenberger, F. & Romanowicz, R. J. (2014) Uncertainty in Flood Inundation Modelling. In *Applied Uncertainty Analysis for Flood Risk Management*.) IMPERIAL COLLEGE PRESS, pp. 232-269.

Bell, V. A., Carrington, D. S. & Moore, R. J. (2001) Comparison of Rainfall-Runoff Models for Flood Forecasting. Part 2: Calibration and Evaluation of Models. Bristol, UK, Report R&D Technical Report W242, pp. 239pp.

Bengio, Y., Simard, P. & Frasconi, P. (1994) Learning Long-term Dependencies With Gradient Descent is Difficult. *IEEE Transactions on Neural Networks* **5(2)**:157-166.

Bergström, S. (1992) *THE HBVMODEL-Its Structure and Applications*

Berkhahn, S., Fuchs, L., and Neuweiler, I. (2019) An ensemble neural network model for real-time prediction of urban floods. *Journal of Hydrology*, 575, 743–754.

Bermúdez, M., Cea, L., and Puertas, J. (2019) A rapid flood inundation model for hazard mapping based on least squares support vector machine regression. *Journal of Flood Risk Management*, 12 (S1), e12522.

Beven, K. (2012) Evolution of Rainfall–Runoff Models: Survival of the Fittest? In *Rainfall-Runoff Modelling*, pp. 25-50.

Beven, K. J. (1996) A Discussion of Distributed Hydrological Modelling. In *Distributed Hydrological Modelling*. (Abbott, M. B., andRefsgaard, J. C. (eds)) Springer Netherlands, Dordrecht, pp. 255-278.

Bhola, P. K., Leandro, J. & Disse, M. (2018) Framework for Offline Flood Inundation Forecasts for Two-Dimensional Hydrodynamic Models. *Geosciences* **8(9)**:346.

Birikundavyi, S., Labib, R., Trung, H. T. & Rousselle, J. (2002) Performance of Neural Networks in Daily Streamflow Forecasting. *Journal of Hydrologic Engineering* **7(5)**:392-398.

Bishop, T. F. A. & Mcbratney, A. B. (2001) A Comparison of Prediction Methods for the Creation of Field-extent Soil Property Maps. *Geoderma* **103(1)**:149-160.

Boser, B. E., Guyon, I. M. & Vapnik, V. N. (1992) A Training Algorithm for Optimal Margin Classifiers In *Proceedings of Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 130401, pp. 144-152.

Breiman, L. (2001) Random Forests. *Machine Learning* **45(1)**:5-32.

Brown, K. M., Hambidge, C. H. & Brownett, J. M. (2016) Progress in Operational Flood Mapping Using Satellite Synthetic Aperture Radar (SAR) and Airborne

Light Detection and Ranging (LiDAR) Data. *Progress in Physical Geography: Earth and Environment* **40(2)**:196-214.
Campolo, M., Soldati, A. & Andreussi, P. (2003) Artificial Neural Network Approach to Flood Forecasting in the River Arno. *Hydrological Sciences Journal* **48(3)**:381-398.
Carriere, P., Mohaghegh, S. & Gaskari, R. (1996) Performance of a Virtual Runoff Hydrograph System. *Journal of Water Resources Planning and Management* **122(6)**:421-427.
Chandwani, V., Vyas, S. K., Agrawal, V. & Sharma, G. (2015) Soft Computing Approach for Rainfall-Runoff Modelling: A Review. *Aquatic Procedia* **4**:1054-1061.
Chang, F.-J., Chen, P.-A., Lu, Y.-R., Huang, E. & Chang, K.-Y. (2014a) Real-time Multi-step-ahead Water Level Forecasting by Recurrent Neural Networks for Urban Flood Control. *Journal of Hydrology* **517**:836-846.
Chang, F.-J., Chiang, Y.-M. & Chang, L.-C. (2007) Multi-step-ahead Neural Networks for Flood Forecasting. *Hydrological Sciences Journal* **52(1)**:114-130.
Chang, F.-J. & Tsai, M.-J. (2016) A Nonlinear Spatio-temporal Lumping of Radar Rainfall for Modeling Multi-step-ahead Inflow Forecasts by Data-driven Techniques. *Journal of Hydrology* **535**:256-269.
Chang, L.-C., Amin, M. Z. M., Yang, S.-N. & Chang, F.-J. (2018a) Building ANN-Based Regional Multi-Step-Ahead Flood Inundation Forecast Models. *Water* **10(9)**:1283.
Chang, L.-C., Chang, F.-J., Yang, S.-N., Kao, I.-F., Ku, Y.-Y., Kuo, C.-L. & Amin, I. M. Z. B. M. (2018b) Building an Intelligent Hydroinformatics Integration Platform for Regional Flood Inundation Warning Systems. *Water* **11(1)**:9.
Chang, L.-C., Shen, H.-Y. & Chang, F.-J. (2014b) Regional Flood Inundation Nowcast Using Hybrid SOM and Dynamic Neural Networks. *Journal of Hydrology* **519**:476-489.
Chang, L.-C., Shen, H.-Y., Wang, Y.-F., Huang, J.-Y. & Lin, Y.-T. (2010) Clustering-based Hybrid Inundation Model for Forecasting Flood Inundation Depths. *Journal of Hydrology* **385(1)**:257-268.
Chang, M.-J., Chang, H.-K., Chen, Y.-C., Lin, G.-F., Chen, P.-A., Lai, J.-S. & Tan, Y.-C. (2018c) A Support Vector Machine Forecasting Model for Typhoon Flood Inundation Mapping and Early Flood Warning Systems. *Water* **10(12)**:1734.
Chen, J., Jin, Q. & Chao, J. (2012) Design of Deep Belief Networks for Short-Term Prediction of Drought Index Using Data in the Huaihe River Basin. *Mathematical Problems in Engineering* **2012**:16.
Chen, P.-A., Chang, L.-C. & Chang, F.-J. (2013) Reinforced Recurrent Neural Networks for Multi-step-ahead Flood Forecasts. *Journal of Hydrology* **497**:71-79.
Chollet, F. & Others, A. (2015) *Keras*. https://keras.io.
Chow, V. T., Maidment, D. R. & Mays, L. W. (1988) Applied Hydrology. New York, McGraw-Hill.
Cilss (2016) Landscapes of West Africa – A Window on a Changing World. U.S. Geological Survey Eros, N. S., Garretson, Sd 57030, United States.
Clement, M. A., Kilsby, C. G. & Moore, P. (2018) Multi-temporal Synthetic Aperture Radar Flood Mapping Using Change Detection. *Journal of Flood Risk Management* **11(2)**:152-168.
Coulibaly, P. & Baldwin, C. K. (2005) Nonstationary Hydrological Time-series Forecasting Using Nonlinear Dynamic Methods. *Journal of Hydrology* **307(1)**:164-174.

Cressie, N. & Hawkins, D. M. (1980) Robust Estimation of the Variogram: I. *Journal of the International Association for Mathematical Geology* **12(2)**:115-124.
Cutler, A. (2014) Random Forests. In *Wiley StatsRef: Statistics Reference Online*. (Balakrishnan, N., Colton, T., Everitt, B., Piegorsch, W., Ruggeri, F., andTeugels, J. L. (eds)).
Cutler, A., Cutler, D. & Stevens, J. (2011) Random Forests. vol. 45, pp. 157-176.
Dadson, S. J., Ashpole, I., Harris, P., Davies, H. N., Clark, D. B., Blyth, E. & Taylor, C. M. (2010) Wetland Inundation Dynamics in a Model of Land Surface Climate: Evaluation in the Niger Inland Delta Region. *Journal of Geophysical Research: Atmospheres* **115(D23)**.
Dawson, C. W., Abrahart, R. J., Shamseldin, A. Y. & Wilby, R. L. (2006) Flood Estimation at Ungauged Sites Using Artificial Neural Networks. *Journal of Hydrology* **319(1–4)**:391-409.
Dawson, C. W. & Wilby, R. (1998) An Artificial Neural Network Approach to Rainfall-Runoff Modelling. *Hydrological Sciences Journal* **43(1)**:47-66.
Dawson, C. W. & Wilby, R. L. (2001) Hydrological Modelling Using Artificial Neural Networks. *Progress in Physical Geography* **25(1)**:80-108.
De Paiva, R. C. D., Buarque, D. C., Collischonn, W., Bonnet, M.-P., Frappart, F., Calmant, S. & Bulhões Mendes, C. A. (2013) Large-scale Hydrologic and Hydrodynamic Modeling of the Amazon River Basin. *Water Resources Research* **49(3)**:1226-1243.
De Vos, N. J. & Rientjes, T. H. M. (2005) Constraints of Artificial Neural Networks for Rainfall-Runoff Modelling: Trade-offs in Hydrological State Representation and Model Evaluation. *Hydrol. Earth Syst. Sci.* **9(1/2)**:111-126.
Dhamge, N. R., Atmapoojya, S. L. & Kadu, M. S. (2012) Genetic Algorithm Driven ANN Model for Runoff Estimation. *Procedia Technology* **6**:501-508.
Dibike, Y. B. & Solomatine, D. P. (2001) River Flow Forecasting Using Artificial Neural Networks. *Physics and Chemistry of the Earth, Part B: Hydrology, Oceans and Atmosphere* **26(1)**:1-7.
Dibike, Y. B., Velickov, S., Solomatine, D. & Abbott, M. B. (2001) Model Induction With Support Vector Machines: Introduction and Applications. *Journal of Computing in Civil Engineering* **15(3)**:208-216.
Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D. & Weingessel, A. (2011) e1071: Misc Functions of the Department of Statistics (e1071), TU Wien. R package version 1.5-24.), Vienna, Austria.
Durrant, P. (2001) winGamma TM : A Non-linear Data Analysis and Modelling Tool With Applications to Flood Prediction. In *Department of Computer Science*.) Cardiff University.
EA (2015) Upton Upon Severn Flood Risk Management Scheme, See https://www.gov.uk/government/publications/upton-upon-severn-flood-risk-management-scheme/upton-upon-severn-flood-risk-management-scheme (accessed 19 September 2019).
Elman, J. L. (1990) Finding Structure in Time. *Cognitive Science* **14(2)**:179-211.
Elshorbagy, A., Corzo, G., Srinivasulu, S. & Solomatine, D. P. (2010a) Experimental Investigation of the Predictive Capabilities of Data Driven Modeling Techniques in Hydrology - Part 1: Concepts and methodology. *Hydrol. Earth Syst. Sci.* **14(10)**:1931-1941.
Elshorbagy, A., Corzo, G., Srinivasulu, S. & Solomatine, D. P. (2010b) Experimental Investigation of the Predictive Capabilities of Data Driven Modeling Techniques in Hydrology - Part 2: Application. *Hydrol. Earth Syst. Sci.* **14(10)**:1943-1961.

Fausett, L. (1994) Fundamentals of Neural Networks: Architectures, Algorithms, and Applications. Prentice-Hall, Inc.

Feldman, A. D. (1981) Hec Models for Water Resources System Simulation: Theory and Experience. In *Advances in Hydroscience*. (Chow, V. T. (ed)) Elsevier, vol. 12, pp. 297-423.

Fohringer, J., Dransch, D., Kreibich, H. & Schröter, K. (2015) Social Media as an Information Source for Rapid Flood Inundation Mapping. *Nat. Hazards Earth Syst. Sci.* **15(12)**:2725-2738.

Fryirs, K. A. & Brierley, G. J. (2012) Catchment Hydrology. In *Geomorphic Analysis of River Systems*. (Fryirs, K. A., andBrierley, G. J. (eds)).

Ghimire, B., Chen, A.S., Djordjevic, S., and Savic, D. (2011) Application of cellular automata approach for fast flood simulation. *In*: *CCWI 2011: Computing and Control for the Water Industry*. Exeter, UK: Centre for Water Systems, University of Exeter, 265–270.

Ghose, D. K., Panda, S. S. & Swain, P. C. (2013) Prediction and Optimization of Runoff via ANFIS and GA. *Alexandria Engineering Journal* **52(2)**:209-220.

Granata, F., Gargano, R. & De Marinis, G. (2016) Support Vector Regression for Rainfall-Runoff Modeling in Urban Drainage: A Comparison With the EPA's Storm Water Management Model. *Water* **8(3)**:69.

Grayson, R. B., Moore, I. D. & Mcmahon, T. A. (1992a) Physically Based Hydrologic Modeling: 1. A Terrain-based Model for Investigative Purposes. *Water Resources Research* **28(10)**:2639-2658.

Grayson, R. B., Moore, I. D. & Mcmahon, T. A. (1992b) Physically Based Hydrologic Modeling: 2. Is the Concept Realistic? *Water Resources Research* **28(10)**:2659-2666.

Guo, J., Zhou, J., Qin, H., Zou, Q. & Li, Q. (2011) Monthly Streamflow Forecasting Based on Improved Support Vector Machine Model. *Expert Systems with Applications* **38(10)**:13073-13081.

Hagan, M. T., Demuth, H. B., Beale, M. H. & Jesús, O. D. (2002) Neural Network Design. 2nd edn. Oklahoma State University.

Han, D., Chan, L. & Zhu, N. (2007) Flood Forecasting Using Support Vector Machines. *Journal of Hydroinformatics* **9(4)**:267-276.

Hassan, Z., Harun, S. & Malek, M. A. (2012) Application of ANNs Model With the SDSM for the Hydrological Trend Prediction in the Sub-catchment of Kurau River, Malaysia. *Journal of Environmental Science & Engineering* **Vol. 1B(Issue 6)**:p577.

Hastie, T., Tibshirani, R. & Friedman, J. (2009) Boosting and Additive Trees. In *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*.) Springer New York, New York, NY, pp. 337-387.

Hengl, T., Bajat, B., Blagojević, D. & Reuter, H. I. (2008) Geostatistical Modeling of Topography Using Auxiliary Maps. *Computers & Geosciences* **34(12)**:1886-1899.

Hengl, T., Heuvelink, G. B. M. & Rossiter, D. G. (2007) About Regression-kriging: From Equations to Case Studies. *Computers & Geosciences* **33(10)**:1301-1314.

Hengl, T., Heuvelink, G. B. M. & Stein, A. (2004) A Generic Framework for Spatial Prediction of Soil Variables Based on Regression-kriging. *Geoderma* **120(1)**:75-93.

Henonin, J., Russo, B., Mark, O. & Gourbesville, P. (2013) Real-time Urban Flood Forecasting and Modelling – A State of the Art. *Journal of Hydroinformatics* **15(3)**:717-736.

Hinton, G. E. (2002) Training Products of Experts by Minimizing Contrastive Divergence. *Neural Comput.* **14(8)**:1771-1800.
Hinton, G. E. (2012) A Practical Guide to Training Restricted Boltzmann Machines. In *Neural Networks: Tricks of the Trade: Second Edition*. (Montavon, G., Orr, G. B., andMüller, K.-R. (eds)) Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 599-619.
Hinton, G. E., Osindero, S. & Teh, Y.-W. (2006) A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation* **18**:1527-1554.
Hosseini, S. M. & Mahjouri, N. (2016) Integrating Support Vector Regression and a Geomorphologic Artificial Neural Network for Daily Rainfall-Runoff Modeling. *Applied Soft Computing* **38**:329-344.
Hsu, K., Gupta, H. & Sorooshian, S. (1997) Application of a Recurrent Neural Network to Rainfall-Runoff Modeling. In *Proceedings of the Annual Water Resources Planning and Management Conference*. (Merritt, D. (ed)) ASCE, Houston, TX, USA, pp. 68-73.
Huth, J., Leinenkugel, P. & Künzer, C. (2015) Surface Water Derivation With WaMaPro to Support Hydrological Applications In *Proceedings of Third Space for Hydrology Workshop*.
Ipcc (2014) Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, R.K. Pachauri and L.A. Meyer (eds.)]. Ipcc, G., Switzerland, 151 Pp.
Jacobs (2017) ISIS User Manual Cost Effective, Integrated Software Solutions. See http://help.floodmodeller.com/isis/ISIS.htm#Tutorials_and_Model_Building/Tutorials_and_Model_Building.htm (accessed 17 July 2019).
Jain, A. & Kumar, A. M. (2007) Hybrid Neural Network Models for Hydrologic Time-series Forecasting. *Applied Soft Computing* **7(2)**:585-592.
Jain, A. & Srinivasulu, S. (2004) Development of Effective and Efficient Rainfall-Runoff Models Using Integration of Deterministic, Real-coded Genetic Algorithms and Artificial Neural Network Techniques. *Water Resources Research* **40(4)**:n/a-n/a.
Jenkins, D. P., Patidar, S. & Simpson, S. A. (2014) Synthesising Electrical Demand Profiles for UK Dwellings. *Energy and Buildings* **76**:605-614.
Jeong, D.-I. & Kim, Y.-O. (2005) Rainfall-Runoff Models Using Artificial Neural Networks for Ensemble Streamflow Prediction. *Hydrological Processes* **19(19)**:3819-3834.
Jhong, Y.-D., Chen, C.-S., Lin, H.-P. & Chen, S.-T. (2018) Physical Hybrid Neural Network Model to Forecast Typhoon Floods. *Water* **10(5)**:632.
Jones, A. J., Margetts, S. & Durrant, P. (2001) The winGamma User Guide.
Jordan, M. I. (1997) Chapter 25 - Serial Order: A Parallel Distributed Processing Approach. In *Advances in Psychology*. (Donahoe, J. W., andPackard Dorsel, V. (eds)) North-Holland, vol. 121, pp. 471-494.
Kabir, S., Patidar, S., Xia, X., Liang, Q., Neal, J., and Pender, G. (2020b) A deep convolutional neural network model for rapid prediction of fluvial flood inundation. *Journal of Hydrology*, 125481.
Kaur, R. & Singh, V. (2014) Time-Frequency Domain Characterization of Stationary and Non stationary Signals. *International Journal of Computer Science and Mobile Computing* **3(5)**:936 – 947.
Kennedy, J. & Eberhart, R. (1995) Particle Swarm Optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*.), vol. 4, pp. 1942-1948 vol.4.

Khan, M. S. & Coulibaly, P. (2006) Bayesian Neural Network for Rainfall-Runoff Modeling. *Water Resources Research* **42(7)**:W07409.

Kirkby, M., Bracken, L. & Reaney, S. (2002) The Influence of Land Use, Soils and Topography on the Delivery of Hillslope Runoff to Channels in SE Spain. *Earth Surface Processes and Landforms* **27(13)**:1459-1473.

Kourgialas, N. N., Dokou, Z. & Karatzas, G. P. (2015) Statistical Analysis and ANN Modeling for Predicting Hydrological Extremes Under Climate Change Scenarios: The Example of a Small Mediterranean Agro-watershed. *Journal of Environmental Management* **154**:86-101.

Kratzert, F., Klotz, D., Brenner, C., Schulz, K. & Herrnegger, M. (2018) Rainfall–Runoff Modelling Using Long Short-Term Memory (LSTM) Networks. *Hydrol. Earth Syst. Sci.* **22(11)**:6005-6022.

Krishna, B., Rao, Y. R. S. & Nayak, P. C. (2011) Time-series Modeling of River Flow Using Wavelet Neural Networks. *Journal of Water Resource and Protection* **Vol.03No.01**:10.

Krupka, M., Wallis, S., Pender, S. & Neelz, S. (2007) Some Practical Aspects of Flood Inundation Modelling, Transport Phenomena in Hydraulics. *Publ. Inst. Geophys. Pol. Acad. Sci.* **7 E(401)**:129-134.

Kumar, V. (2011) Hysteresis BT - Encyclopedia of Snow, Ice and Glaciers. (V. P. Singh, P. Singh & U. K. Haritashya, eds.), 554–555. Dordrecht: Springer Netherlands. doi:10.1007/978-90-481-2642-2_252

Le, X.-H., Ho, H. V., Lee, G. & Jung, S. (2019) Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting. *Water* **11(7)**:1387.

Leedal, D., Neal, J., Beven, K., Young, P. & Bates, P. (2010) Visualization Approaches for Communicating Real-time Flood Forecasting Level and Inundation Information. *Journal of Flood Risk Management* **3(2)**:140-150.

Lhomme, J., Sayers, P. B., Gouldby, B. P., Samuels, P. G., Wills, M. & Mulet-Marti, J. (2008) Recent Development and Application of a Rapid Flood Spreading Method In *Proceedings of FLOODrisk 2008*.

Li, X., Du, Z. & Song, G. (2018) A Method of Rainfall Runoff Forecasting Based on Deep Convolution Neural Networks. In *2018 Sixth International Conference on Advanced Cloud and Big Data (CBD).*), pp. 304-310.

Lin, J.-Y., Cheng, C.-T. & Chau, K.-W. (2006) Using Support Vector Machines for Long-term Discharge Prediction. *Hydrological Sciences Journal* **51(4)**:599-612.

Liong, S.-Y. & Sivapragasam, C. (2002) Flood Stage Forecasting With Support Vector Machines. *JAWRA Journal of the American Water Resources Association* **38(1)**:173-186.

Liu, J., Gong, M., Zhao, J., Li, H. & Jiao, L. (2016) Difference Representation Learning Using Stacked Restricted Boltzmann Machines for Change Detection in SAR Images. *Soft Computing* **20(12)**:4645-4657.

Liu, L., Liu, Y., Wang, X., Yu, D., Liu, K., Huang, H., and Hu, G. (2015) Developing an effective 2-D urban flood inundation model for city emergency management based on cellular automata. *Nat. Hazards Earth Syst. Sci.*, 15 (3), 381–391.

Liu, Y. & Pender, G. (2013) Automatic Calibration of a Rapid Flood Spreading Model Using Multiobjective Optimisations. *Soft Computing* **17(4)**:713-724.

Liu, Y. & Pender, G. (2015) A Flood Inundation Modelling Using v-Support Vector Machine Regression Model. *Engineering Applications of Artificial Intelligence* **46, Part A**:223-231.

Liu, Y., Pender, G. & Neélz, S. (2009) Improving the Performance of Fast Inundation Models Using v-Support Vector Regression and Particle Swarm Optimisation. *In the 33rd IAHR 2009 Congress*:1436–1443.

Machado, F., Mine, M., Kaviski, E. & Fill, H. (2011) Monthly Rainfall–Runoff Modelling Using Artificial Neural Networks. *Hydrological Sciences Journal* **56(3)**:349-361.

Makkeasorn, A., Chang, N. B. & Zhou, X. (2008) Short-term Streamflow Forecasting With Global Climate Change Implications – A Comparative Study Between Genetic Programming and Neural Network Models. *Journal of Hydrology* **352(3)**:336-354.

Mallikarjuna, P., Suresh Babu, C. H. & Reddy, A. J. M. (2009) Rainfall—Runoff Modelling Using Artificial Neural Networks. *ISH Journal of Hydraulic Engineering* **15(1)**:24-33.

Martinis, S., Kuenzer, C., Wendleder, A., Huth, J., Twele, A., Roth, A. & Dech, S. (2015) Comparing Four Operational SAR-based Water and Flood Detection Approaches. *International Journal of Remote Sensing* **36(13)**:3519-3543.

Martinis, S., Twele, A. & Voigt, S. (2009) Towards Operational Near Real-time Flood Detection Using a Split-based Automatic Thresholding Procedure on High Resolution TerraSAR-X Data. *Nat. Hazards Earth Syst. Sci.* **9(2)**:303-314.

Mason, D. C., Giustarini, L., Garcia-Pintado, J. & Cloke, H. L. (2014) Detection of Flooded Urban Areas in High Resolution Synthetic Aperture Radar Images Using Double Scattering. *International Journal of Applied Earth Observation and Geoinformation* **28**:150-159.

Mccullagh, P. & Nelder, J. A. (1989) Generalized Linear Models. 2 edn. London, Chapman & Hall / CRC.

Melesse, A. M., Ahmad, S., Mcclain, M. E., Wang, X. & Lim, Y. H. (2011) Suspended Sediment Load Prediction of River Systems: An Artificial Neural Network Approach. *Agricultural Water Management* **98(5)**:855-866.

Minns, A. W. & Hall, M. J. (1996) Artificial Neural Networks as Rainfall-Runoff Models. *Hydrological Sciences Journal* **41(3)**:399-417.

Misra, D., Oommen, T., Agarwal, A., Mishra, S. K. & Thompson, A. M. (2009) Application and Analysis of Support Vector Machine Based Simulation for Runoff and Sediment Yield. *Biosystems Engineering* **103(4)**:527-534.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. (2013) Playing Atari With Deep Reinforcement Learning.

Moore, R. J. (2007) The PDM rainfall-Runoff model. *Hydrol. Earth Syst. Sci.* **11(1)**:483-499.

Moradkhani, H., Hsu, K.-L., Gupta, H. V. & Sorooshian, S. (2004) Improved Streamflow Forecasting Using Self-organizing Radial Basis Function Artificial Neural Networks. *Journal of Hydrology* **295(1)**:246-262.

Mosavi, A., Ozturk, P. & Chau, K.-W. (2018) Flood Prediction Using Machine Learning Models: Literature Review. *Water* **10(11)**:1536.

Mujumdar, P. P. & Kumar, D. N. (2012) Introduction, Floods in a Changing Climate: Hydrologic Modeling. Cambridge University Press.

Müller, K.-R., Smola, A. J., Rätsch, G., Schölkopf, B., Kohlmorgen, J. & Vapnik, V. (1997) Predicting Time-series With Support Vector Machines. In *Artificial Neural Networks — ICANN'97: 7th International Conference Lausanne, Switzerland, October 8–10, 1997 Proceeedings*. (Gerstner, W., Germond, A., Hasler, M., andNicoud, J.-D. (eds)) Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 999-1004.

Mulligan, M. & Wainwright, J. (2013) Modelling Catchment and Fluvial Processes and their Interactions. In *Environmental Modelling: Finding Simplicity in Complexity*.) Second edn. John Wiley & Sons, Ltd, pp. 183-204.

Nagesh Kumar, D., Srinivasa Raju, K. & Sathish, T. (2004) River Flow Forecasting Using Recurrent Neural Networks. *Water Resources Management* **18(2)**:143-161.

Nanda, T., Sahoo, B., Beria, H. & Chatterjee, C. (2016) A Wavelet-based Non-linear Autoregressive With Exogenous Inputs (WNARX) Dynamic Neural Network Model for Real-time Flood Forecasting Using Satellite-based Rainfall Products. *Journal of Hydrology* **539**:57-73.

Nash, J. E. & Sutcliffe, J. V. (1970) River Flow Forecasting Through Conceptual Models Part I — A Discussion of Principles. *Journal of Hydrology* **10(3)**:282-290.

Nayak, P. C., Venkatesh, B., Krishna, B. & Jain, S. K. (2013) Rainfall-Runoff Modeling Using Conceptual, Data Driven, and Wavelet Based Computing Approach. *Journal of Hydrology* **493**:57-67.

Neal, J., Schumann, G. & Bates, P. (2012) A Subgrid Channel Model for Simulating River Hydraulics and Floodplain Inundation Over Large and Data Sparse Areas. *Water Resources Research* **48(11)**.

Néelz, S., Hall, J. & Pender, G. (2007) Improving the Performance of Fast Flood Inundation Models by Incorporating Results From Very High Resolution Simulations. In *Proceedings of Flood Risk Assessment II Conference*.), Institute of Mathematics & Its Applications, Southend on Sea, pp. 1-9.

Nourani, V., Kisi, Ö. & Komasi, M. (2011) Two Hybrid Artificial Intelligence Approaches for Modeling Rainfall–Runoff Process. *Journal of Hydrology* **402(1–2)**:41-59.

Nkwunonwo, U.C., Whitworth, M., and Baily, B., 2019. Urban flood modelling combining cellular automata framework with semi-implicit finite difference numerical formulation. *Journal of African Earth Sciences*, 150, 272–281.

Okkan, U. (2012) Wavelet Neural Network Model for Reservoir Inflow Prediction. *Scientia Iranica* **19(6)**:1445-1454.

Pan, T. Y., Lai, J. S., Chang, T. J., Chang, H. K., Chang, K. C. & Tan, Y. C. (2011) Hybrid Neural Networks in Rainfall-inundation Forecasting Based on a Synthetic Potential Inundation Database. *Nat. Hazards Earth Syst. Sci.* **11(3)**:771-787.

Parlos, A. G., Rais, O. T. & Atiya, A. F. (2000) Multi-step-ahead Prediction Using Dynamic Recurrent Neural Networks. *Neural Networks* **13(7)**:765-786.

Patidar, S., Allen, D., Haynes, R. & Haynes, H. (2018) Stochastic Modelling of Flow Sequences for Improved Prediction of Fluvial Flood Hazards. *Geological Society, London, Special Publications* **488**:SP488.4.

Pebesma, E. & Bivand, R. (2005) Classes and Methods for Spatial Data: the sp Package.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, D. (2011) Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **12**:2825-2830.

Pender, D., Patidar, S., Pender, G. & Haynes, H. (2015) Stochastic Simulation of Daily Streamflow Sequences Using a Hidden Markov Model. *Hydrology Research*.

Pilling, C., Price, D., Wynn, A., Lane, A., Cole, S. J., Moore, R. J. & Aldridge, T. (2014) From Drought to Floods in 2012: Operations and Early Warning Services in the UK. In *Hydrology in a Changing World: Environmental and Human*

Dimensions Proceedings of FRIEND-Water) IAHS Publ., Montpellier, France, vol. 363.
Price, D., Hudson, K., Boyce, G., Schellekens, J., Moore, R. J., Clark, P., Harrison, T., Connolly, E. & Pilling, C. (2012) Operational Use of a Grid-based Model for Flood Forecasting. *Proceedings of the Institution of Civil Engineers - Water Management* **165(2)**:65-77.
Raghavendra, N., Sujay & Deka, P. C. (2014) Support Vector Machine Applications in the Field of Hydrology: A Review. *Applied Soft Computing* **19**:372-386.
Rajurkar, M. P., Kothyari, U. C. & Chaube, U. C. (2002) Artificial Neural Networks for Daily Rainfall—Runoff Modelling. *Hydrological Sciences Journal* **47(6)**:865-877.
Rajurkar, M. P., Kothyari, U. C. & Chaube, U. C. (2004) Modeling of the Daily Rainfall-Runoff Relationship With Artificial Neural Network. *Journal of Hydrology* **285(1–4)**:96-113.
Raschka, S. & Mirjalili, V. (2017) Python Machine Learning, *2nd Ed.* 2 edn., Packt Publishing. Birmingham, UK.
Rasouli, K., Hsieh, W. W. & Cannon, A. J. (2012) Daily Streamflow Forecasting by Machine Learning Methods With Weather and Climate Inputs. *Journal of Hydrology* **414-415**:284-293.
Refsgaard, J. & Storm, B. (1995) MIKE SHE. In *Computer Models of Watershed Hydrology*. (Singh, V. (ed)) Water Resource Publications, Colorado, pp. 809–846.
Riad, S., Mania, J., Bouchaou, L. & Najjar, Y. (2004) Rainfall-Runoff Model Using an Artificial Neural Network Approach. *Mathematical and Computer Modelling* **40(7–8)**:839-846.
Rodríguez, E., Morris, C. S. & Belz, J. E. (2006) A Global Assessment of the SRTM Performance. *Photogrammetric Engineering & Remote Sensing* **72(3)**:249-260.
Romanowicz, R. J., Young, P. C. & Beven, K. J. (2006) Data Assimilation and Adaptive Forecasting of Water Levels in the River Severn Catchment, United Kingdom. *Water Resources Research* **42(6)**.
Romanowicz, R. J., Young, P. C., Beven, K. J. & Pappenberger, F. (2008) A Data Based Mechanistic Approach to Nonlinear Flood Routing and Adaptive Flood Level Forecasting. *Advances in Water Resources* **31(8)**:1048-1056.
Rosser, J. F., Leibovici, D. G. & Jackson, M. J. (2017) Rapid Flood Inundation Mapping Using Social Media, Remote Sensing and Topographic Data. *Natural Hazards* **87(1)**:103-120.
Rubio, G., Pomares, H., Rojas, I. & Herrera, L. J. (2011) A Heuristic Method for Parameter Selection in LS-SVM: Application to Time-series Prediction. *International Journal of Forecasting* **27(3)**:725-739.
Sahoo, S., Russo, T. A., Elliott, J. & Foster, I. (2017) Machine Learning Algorithms for Modeling Groundwater Level Changes in Agricultural Regions of the U.S. *Water Resources Research* **53(5)**:3878-3894.
Sazal, M. M. R., Biswas, S. K., Amin, M. F. & Murase, K. (2014) Bangla Handwritten Character Recognition Using Deep Belief Network. In *2013 International Conference on Electrical Information and Communication Technology (EICT)*.), pp. 1-4.
Schumann, G., Pappenberger, F. & Matgen, P. (2008) Estimating Uncertainty Associated With Water Stages From a Single SAR Image. *Advances in Water Resources* **31(8)**:1038-1047.
Segal, M. & Xiao, Y. (2011) Multivariate Random Forests. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **1(1)**:80-87.

Shen, H. Y. & Chang, L. C. (2013) Online Multistep-ahead Inundation Depth Forecasts by Recurrent NARX Networks. *Hydrol. Earth Syst. Sci.* **17(3)**:935-944.
Shiri, J. & Kisi, O. (2010) Short-term and Long-term Streamflow Forecasting Using a Wavelet and Neuro-fuzzy Conjunction Model. *Journal of Hydrology* **394(3)**:486-493.
Shoaib, M., Shamseldin, A. Y. & Melville, B. W. (2014) Comparative Study of Different Wavelet Based Neural Network Models for Rainfall–Runoff Modeling. *Journal of Hydrology* **515**:47-58.
Shortridge, J. E., Guikema, S. D. & Zaitchik, B. F. (2016) Machine Learning Methods for Empirical Streamflow Simulation: A Comparison of Model Accuracy, Interpretability, and Uncertainty in Seasonal Watersheds. *Hydrol. Earth Syst. Sci.* **20(7)**:2611-2628.
Singh, V. P. & Woolhiser, D. A. (2002) Mathematical Modeling of Watershed Hydrology. *Journal of Hydrologic Engineering* **7(4)**:270-292.
Solomatine, D. P. & Dulal, K. N. (2003) Model Trees as an Alternative to Neural Networks in Rainfall—Runoff Modelling. *Hydrological Sciences Journal* **48(3)**:399-411.
Solomatine, D. P. & Ostfeld, A. (2008) Data-driven Modelling: Some Past Experiences and New Approaches. *Journal of Hydroinformatics* **10(1)**:3-22.
Stefánsson, A., Končar, N. & Jones, A. J. (1997) A Note on the Gamma Test. *Neural Computing & Applications* **5(3)**:131-133.
Suarez, M. & Morton, A. (2001) Kinds of Models. In *Model Validation: Perspectives in Hydrological Science*. (Bates, P. D., andAnderson, M. G. (eds)) John Wiley & Sons, Ltd., pp. 11-21.
Sutskever, I., Martens, J. & Hinton, G. (2011) Generating Text With Recurrent Neural Networks. In *Proceedings of Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress, 3104610, pp. 1017-1024.
Teh, Y. W. & Hinton, G. E. (2000) Rate-coded Restricted Boltzmann Machines for Face Recognition In *Proceedings of Proceedings of the 13th International Conference on Neural Information Processing Systems*. MIT Press, 3008878, pp. 872-878.
Todini, E. (2007) Hydrological Catchment Modelling: Past, Present and Future. *Hydrol. Earth Syst. Sci.* **11(1)**:468-482.
Tokar, A. S. & Markus, M. (2000) Precipitation-Runoff Modeling Using Artificial Neural Networks and Conceptual Models. *Journal of Hydrologic Engineering* **5(2)**:156-161.
Toth, E. & Brath, A. (2007) Multistep Ahead Streamflow Forecasting: Role of Calibration Data in Conceptual and Neural Network Modeling. *Water Resources Research* **43(11)**.
Tyralis, H., Papacharalampous, G. & Langousis, A. (2019) A Brief Review of Random Forests for Water Scientists and Practitioners and Their Recent History in Water Resources. *Water* **11(5)**:910.
Uysal, G., ſorman, A. A. & ſensoy, A. (2016) Streamflow Forecasting Using Different Neural Network Models With Satellite Data for a Snow Dominated Region in Turkey. *Procedia Engineering* **154**:1185-1192.
Vapnik, V. N. (1997) The Support Vector Method. In *Artificial Neural Networks — ICANN'97: 7th International Conference Lausanne, Switzerland, October 8–10, 1997 Proceeedings*. (Gerstner, W., Germond, A., Hasler, M., andNicoud, J.-D. (eds)) Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 261-271.

Vijayashanthar, V., Qiao, J., Zhu, Z., Entwistle, P. & Yu, G. (2018) Modeling Fecal Indicator Bacteria in Urban Waterways Using Artificial Neural Networks. *Journal of Environmental Engineering* **144(6)**:05018003.

Wang, J., Wang, J., Fang, W. & Niu, H. (2016) Financial Time-series Prediction Using Elman Recurrent Random Neural Networks. *Computational Intelligence and Neuroscience* **2016**:14.

Williams, R. J. & Zipser, D. (1989) A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation* **1(2)**:270-280.

Witten, I. H., Frank, E. & Hall, M. A. (2011) Chapter 1 - What's It All About? In *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*.) Morgan Kaufmann, Boston, pp. 3-38.

Wu, Z., Zhou, Y., Wang, H., and Jiang, Z. (2020). Depth prediction of urban flood under different rainfall return periods based on deep learning and data warehouse. *Science of The Total Environment*, 716, 137077.

Wu, C. L. & Chau, K. W. (2011) Rainfall–Runoff Modeling Using Artificial Neural Network Coupled With Singular Spectrum Analysis. *Journal of Hydrology* **399(3–4)**:394-409.

Wu, C. L., Chau, K. W. & Li, Y. S. (2008a) River Stage Prediction Based on a Distributed Support Vector Regression. *Journal of Hydrology* **358(1)**:96-111.

Wu, W., Wang, X., Xie, D. & Liu, H. (2008b) Soil Water Content Forecasting by Support Vector Machine in Purple Hilly Region. In *Computer And Computing Technologies In Agriculture, Volume I: First IFIP TC 12 International Conference on Computer and Computing Technologies in Agriculture (CCTA 2007), Wuyishan, China, August 18-20, 2007*. (Li, D. (ed)) Springer US, Boston, MA, pp. 223-230.

Yamazaki, D., Kanae, S., Kim, H. & Oki, T. (2011) A Physically Based Description of Floodplain Inundation Dynamics in a Global River Routing Model. *Water Resources Research* **47(4)**.

Yaseen, Z. M., El-Shafie, A., Jaafar, O., Afan, H. A. & Sayl, K. N. (2015) Artificial intelligence based models for stream-flow forecasting: 2000–2014. *Journal of Hydrology* **530**:829-844.

Young, A. R., Grew, R. & Holmes, M. G. R. (2003) Low Flows 2000: A National Water Resources Assessment and Decision Support Tool. *Water Science and Technology* **48(10)**:119-126.

Yu, P.-S., Chen, S.-T. & Chang, I. F. (2006) Support Vector Regression for Real-time Flood Stage Forecasting. *Journal of Hydrology* **328(3–4)**:704-716.