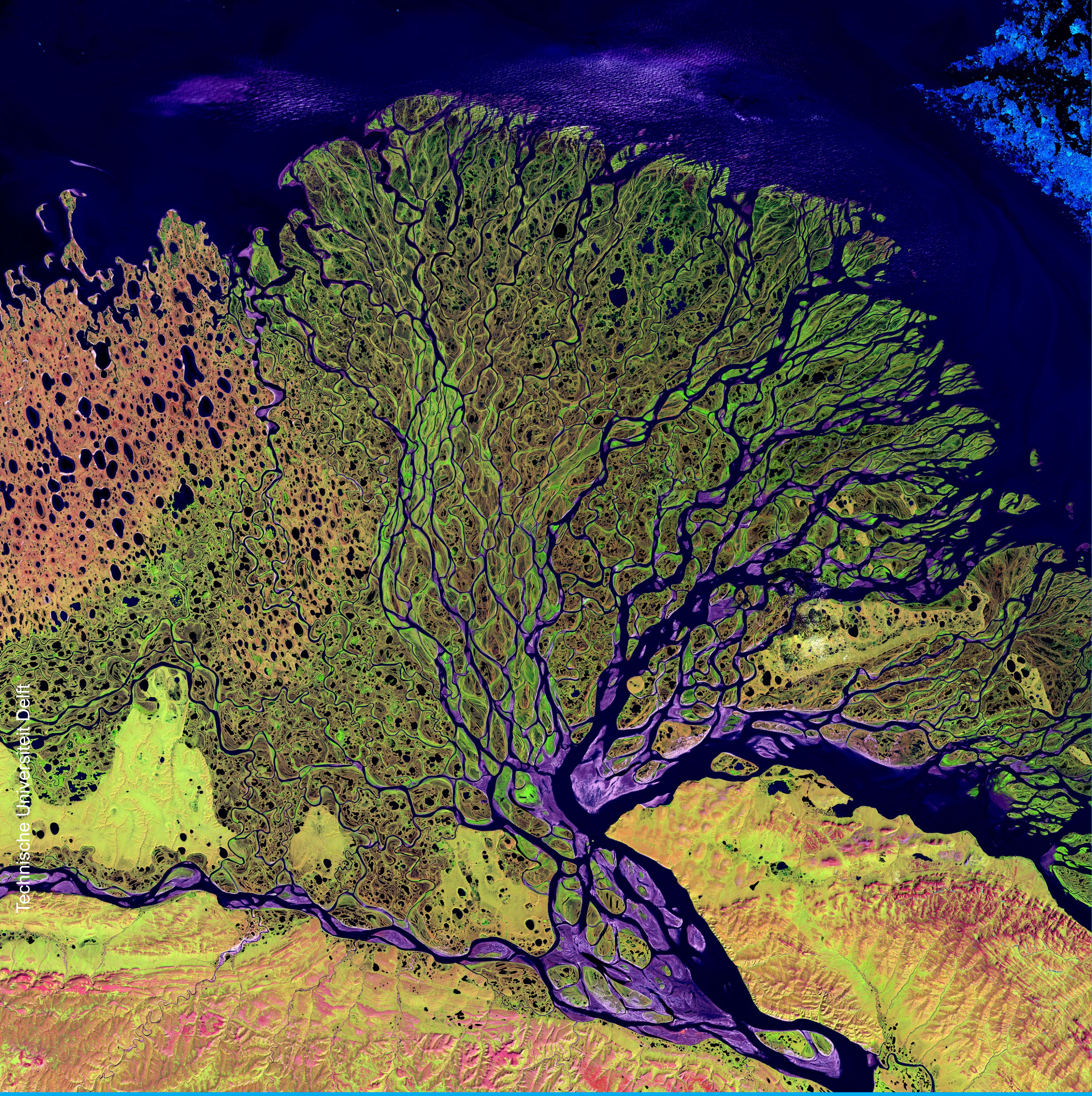


MSc Thesis

Matching streamflow river gauges with hydrologic models

M. van der Ven

TUDelft

Matching streamflow river gauges with hydrologic models

MSc Thesis

by

M. van der Ven

to obtain the degree of Master of Science
at the Delft University of Technology,

Student number:	4280997	
Project duration:	September 1, 2020 - July 30, 2021	
Thesis committee:	Dr. Ir. R. Hut	TU Delft, Chair & Daily supervisor
	MSc J. Aerts,	TU Delft, Daily supervisor
	Dr. R. Taormina,	TU Delft
	Prof. Dr. C. Prudhomme	ECMWF
	Dr. C. Mazzetti	ECMWF
	Dr. B. Weel	eScience Center
	S. M. Saia, PhD	State Climate Office of NC

Preface

Before you lies the thesis report ”Matching streamflow river gauges with hydrologic models”. This report has been written to complete the graduation requirements of the master Environmental Engineering at the Delft University of Technology.

The combination of the fields of hydrology and machine learning sparked my interest for this project. I would like to thank my daily supervisors, R. Hut and J. Aerts, for inspiring me with their enthusiastic guidance during this process. For providing me with a theoretical basis of machine learning, I would like to thank R. Taormina. I also wish to thank S. Saia and B. Weel for their availability to answer any of my questions and for their interesting suggestions. Finally, I would like to thank C. Prudhomme and C. Mazzetti for their counsel and for providing me with essential data resources for this research.

The thesis work was conducted under unique circumstances. I would like to thank my roommates for making these circumstances bearable with their positivity, the many walks around the block for ice cream, and coffee breaks with (home-made) sweets. I want to thank my boyfriend for supporting me with good food, his patience and understanding, and providing me with some space every once in a while. Finally, I would like to thank my parents and sister for their relentless support and safe space they have always provided.

M. van der Ven
Delft, July 2021

Abstract

Hydrologic model performance evaluation depends on streamflow observations that are accurately positioned in the landscape. For distributed hydrologic models, this means that the streamflow observation need to be mapped to a location along the model streamflow network that represents the location of the observation station in a hydrologic system. However, the gridded representation of the modelled area causes a spatial mismatch between the hydrologic system and hydrologic model.
In this study we aimed to develop a Machine learning-based method to improve matching between streamflow observations and streamflow simulations. The setup of this method was implemented in two steps: (1) a dataset was created consisting of streamflow characteristics of simulations and and observations and (2) a Machine learning algorithm was trained with the created dataset.
Three data sources were used for the creation of the dataset: (1) 595 streamflow observations were retrieved from the Global Runoff Database Centre (GRDC), (2) streamflow simulations were extracted from the European Flood Alert System (EFAS) and (3) we were provided with a manually created and checked dataset by European Centre for Medium-Range Weather Forecasts linking each GRDC observation to the correct EFAS grid cell.
To link 60% of the observations in the dataset with the correct grid cells, the observations required to be moved away from the cell corresponding to the geo-location of the observations. The method developed in this study anticipated this by creating a search window around the initial location of each observation. The streamflow simulations were extracted from the grid cells in the search window and compared with the streamflow observation. The algorithm aimed to select the streamflow simulation that best reflected the characteristics of the streamflow observation. The characteristics were described with streamflow signatures.
Four Machine learning algorithms, a Logistic Regression, Random Forest, Support Vector Machine and K-Nearest Neighbours algorithm, were trained with a K-fold Cross Validation procedure to match streamflow simulations with streamflow observations based on streamflow signatures. Their performance was compared with four benchmark algorithms: a Center Cell benchmark which places the observations on their initial location, and the Root Mean Squared Error, Kling-Gupta Efficiency and Nash-Sutcliffe Efficiency benchmarks that compare the streamflow observation with the streamflow simulations.
We identified the Logistic Regression and Random Forest algorithms as the best performing algorithms. However, neither outperformed all benchmarks. Despite these results, we show the potential to automate matching between streamflow observations and streamflow simulations with a ML-based approach in this study.

Contents

List of Figures

List of Tables

Glossary

Deep learning	A set of advanced Machine learning algorithms based on Artificial Neural Networks (Shen, 2018) .
Global water cycle	The movement of water on planet Earth in all phases (liquid, gas, solid). Consists of the Oceanic, Atmospheric and Terrestrial water cycle .
Hydrological model	A mathematical representation of a real-life hydrological system using hydrological conservation equations to compute the state of system.
Hydrology	The scientific discipline that studies the movement and distribution of water on planet Earth.
K-fold Cross Validation	A validation method where a dataset is split into K parts. K iterations follow, where K-1 parts are used as training set and one part is held out as validation set. This way, the entire dataset is used to train an algorithm.
LISFLOOD	Physically based spatially distributed hydrological model (De Roo et al., 2000).
Machine learning (ML)	A range of algorithms that have the ability to learn from experience and do not need to be explicitly programmed beforehand (Mitchell, 1997, Samuel, 1959).
Terrestrial water cycle	The movement of water under or directly above the land surface. Combined with the Oceanic and Atmospheric water cycle, the complete global water cycle is formed.

Acronyms

ASTER Advanced Spaceborne Thermal Emission and Reflection Radiometer.

CC Center Cell.

CCM Catchment Characterisation and Modelling.

CDS Climate Data Store.

DEM Digital Elevation Model.

ECMWF European Centre for Medium-Range Weather Forecasts.

EFAS European Flood Alert System.

FDC Flow Duration Curve.

GHM Global Hydrological Model.

GRDC Global Runoff Database Centre.

JRC Joint Research Centre.

K-NN K-Nearest Neighbours.

KGE Kling-Gupta Efficiency.

LDD Local Drainage Direction.

LR Logistic Regression.

ML Machine learning.

NSE Nash-Sutcliffe Efficiency.

PC Principal Component.

PCA Principal Component Analysis.

RF Random Forest.

RMSE Root Mean Squared Error.

SVM Support Vector Machine.

1 Introduction

Hydrology is the scientific discipline that studies the movement and distribution of water. Monitoring and understanding the terrestrial water cycle is vital for society: it enables provision of safe drinking water, flood predictions and sustainable water resources management. Next to human interventions in the water cycle (Montanari et al., 2013), anthropologically-induced climate change is expected to further alter the global water cycle (Prudhomme et al., 2002, 2014). Hydrological models help assess the impact of the changing water cycle for society.

Hydrological models are a mathematical simplification of hydrological systems that aim to mimic the system's streamflow response to inputs such as precipitation, solar radiation and land use changes (Döll and Lehner, 2002). The simplest hydrological model represents a system as a bucket. The bucket depicts the storage of the system. The storage level in these buckets is influenced by input fluxes, such as precipitation and evaporation. Streamflow, the model output, is determined by the rate of flow out of the bucket. This outgoing flux is determined by parameters linked to geo-physical properties of the system, such as elevation, slope, soil type, vegetation cover and land use. By increasing the number of buckets in the model and adapting the buckets, a more accurate representation of different hydrological processes can be represented (for more background information, see Appendix A, p.36).

An additional element to modelling is achieved through spatial representation. Lumped models simulate a hydrological system as a single unit, with uniform forcing and parameters, whereas distributed models divide the system in separate grid cells that are interconnected. The hydrological response in each cell is calculated separately. This enables the model to capture the spatial variability of the hydrological system (Ajami et al., 2004, Krysanova et al., 1999, Singh and Woolhiser, 2002) (for more background information, see Appendix A, p.36). Linked to spatial representation are the connections of the sub-systems in distributed models. The most common method of linking subsystems is performed with a Local Drainage Direction (LDD) map derived from a Digital Elevation Model (DEM) (also see: Figure 1.1a, p.3 and Appendix A, p.36). With the linking of subsystems, the accumulated streamflow from upstream cells forms an additional incoming flux for each cell besides forcing data. The output of each cell is directed to another downstream cell by the LDD until the outlet point is reached. Here the accumulated streamflow leaves the hydrological system captured by the model. In some models the direction towards an outlet point is done in two phases by including a streamflow network (see: Figure 1.1c, p.3). In this case, the LDD is used to direct streamflow from cell to cell until a streamflow network is reached. From here, flow direction is determined by a streamflow network.

Besides meteorological input and model structure, the simulated runoff depends on model parameters. Model parameters can either be derived from additional data sources (e.g., soil maps, land cover maps) or determined by calibration against observed (historical) states of the model. After calibration, model performance is evaluated by comparing simulated streamflow timeseries with streamflow observations. The degree of similarity determines to what extent the simulated output of the model is able to match observations. Model performance is quantified by objective functions that numerically compare observations and simulations, such as the Nash-Sutcliffe Efficiency (NSE) (Nash and Sutcliffe, 1970), Kling-Gupta Efficiency (KGE) (Gupta et al., 2009) or (Root) Mean Squared Error ((R)MSE). During the calibration and evaluation procedure, a dataset with meteorological observations and streamflow observations is split into a calibration and evaluation set. After determining model parameters with help of the calibration set, the model performance is evaluated with the evaluation set. Alternatively, with cross-calibration, the meteorological and streamflow dataset can be split into multiple parts. With each iteration, a part is hold out as evaluation dataset, while the remaining parts are used for the model calibration.

The gridded representation of a hydrological system in a distributed model introduces an additional source of uncertainty: it is a cause for spatial mismatches between the physical system and the model (Graham et al., 1999). This complicates matching streamflow simulations at the locations of observations in the hydrological system. Because of this, issues arise during model calibration and evaluation. Without streamflow simulations at the location of the observations in the hydrological system, a structural uncertainty during model calibration and evaluation is introduced. This uncertainty cannot be reduced by improving the input data, calibration data, model parameters nor model structure.

Therefore, it is important that simulations come from a location in the hydrological model that matches with the location of the observations in the hydrological system.

Methods have been developed to overcome the spatial mismatch between the actual hydrological system and hydrological model. An overview of these methods can be found in Figure 1.1 (p.3). In an attempt to force the correct location of the streamflow network, a pre-existing and validated streamflow network can be "burned" into a DEM (Figure 1.1b, p.3): the elevation difference between the streamflow network and surrounding area is increased, so that the LDD is adjusted accordingly (Graham et al., 1999, Lehner et al., 2006, 2008, Renssen and Knoop, 2000). However, this does not always yield adequate results (Renssen and Knoop, 2000).
After streamflow network derivation, iterative and manual correction procedures include editing based on digitized high resolution maps and vector data such as the Digital Chart of the World (Costa et al., 2002, Danko, 1991, Döll and Lehner, 2002, USGS, 2000, Vörösmarty et al., 2000) and the comparison of upstream areas of observations locations (Costa et al., 2002). Döll and Lehner (2002) adjust the location of major confluences in the network to structurally match with the real life network. This implies that the location of the representative cell of major confluences does not have to be the cell corresponding to the actual geographical location. In case of multiple confluences located in the same cell, one of the confluences may be shifted to a neighbouring cell for correct representation of sub-basins.
Another method to incorporate a pre-existing and validated streamflow network in a hydrological model, is to execute streamflow routing in two phases (Figure 1.1c, p.3): streamflow is directed based on a LDD until it reaches a pre-existing and validated streamflow network that further guides the streamflow through the model (Thielen et al., 2009). An advantage of this method is that high resolution and high quality streamflow networks can be implemented. However, the spatial resolution of the streamflow network needs to be adapted to match the model grid. The resampling of the resolution of a streamflow network can introduce additional spatial mismatches (Yamazaki et al., 2014) as illustrated in Figure 1.2 (p.4).

Evaluation of the streamflow network location is conducted by visual comparison against high resolution datasets (Döll and Lehner, 2002, USGS, 2000) and by comparing upstream area information from the network with data from literature and observations (Coe, 2000, Costa et al., 2002, David et al., 2011, Döll and Lehner, 2002, Irving et al., 2018, Lehner and Grill, 2013, Lehner et al., 2006, 2008, Li et al., 2015, Renssen and Knoop, 2000, Sutanudjaja et al., 2018, Vörösmarty et al., 2000). A common method to evaluate both the location of the streamflow network and streamflow simulations in the model uses location information and streamflow observations of gauging stations. Based on the station coordinates, streamflow observations are placed in the model and snapped to the nearest cell. The estimated upstream area, provided with the station metadata, is compared to the upstream area in the streamflow network. If the discrepancy between the two values falls below below 5% to 15% (Costa et al., 2002, Döll and Lehner, 2002, Sutanudjaja et al., 2018), the observations are used for streamflow evaluation. In case of a bigger difference in upstream area, either the evaluation data is omitted from the study, or an inspection and manual adjustment are required in an attempt to find a cell representing the observations. It is often found that the evaluation data is represented by a neighbouring cell belonging to a tributary or neighbouring catchment (Döll and Lehner, 2002, US Geological Survey, 2015) or that uncertainties in evaluation data exists (Döll and Lehner, 2002, Vörösmarty et al., 2000). In cases where the observations were shifted from their initial position, shifts from 1 to 3 cells were needed to find representative simulations (Coe, 2000, Irving et al., 2018, Kuentz et al., 2017, Li et al., 2015, Sutanudjaja et al., 2018, Wu et al., 2014).

By improving the efficiency of matching model output with observations, requiring a number of iterations and manual corrections, time and energy could be spent more efficiently and more data would remain for the improvement of hydrological modelling. As the representative simulations for observations are selected, a source of error

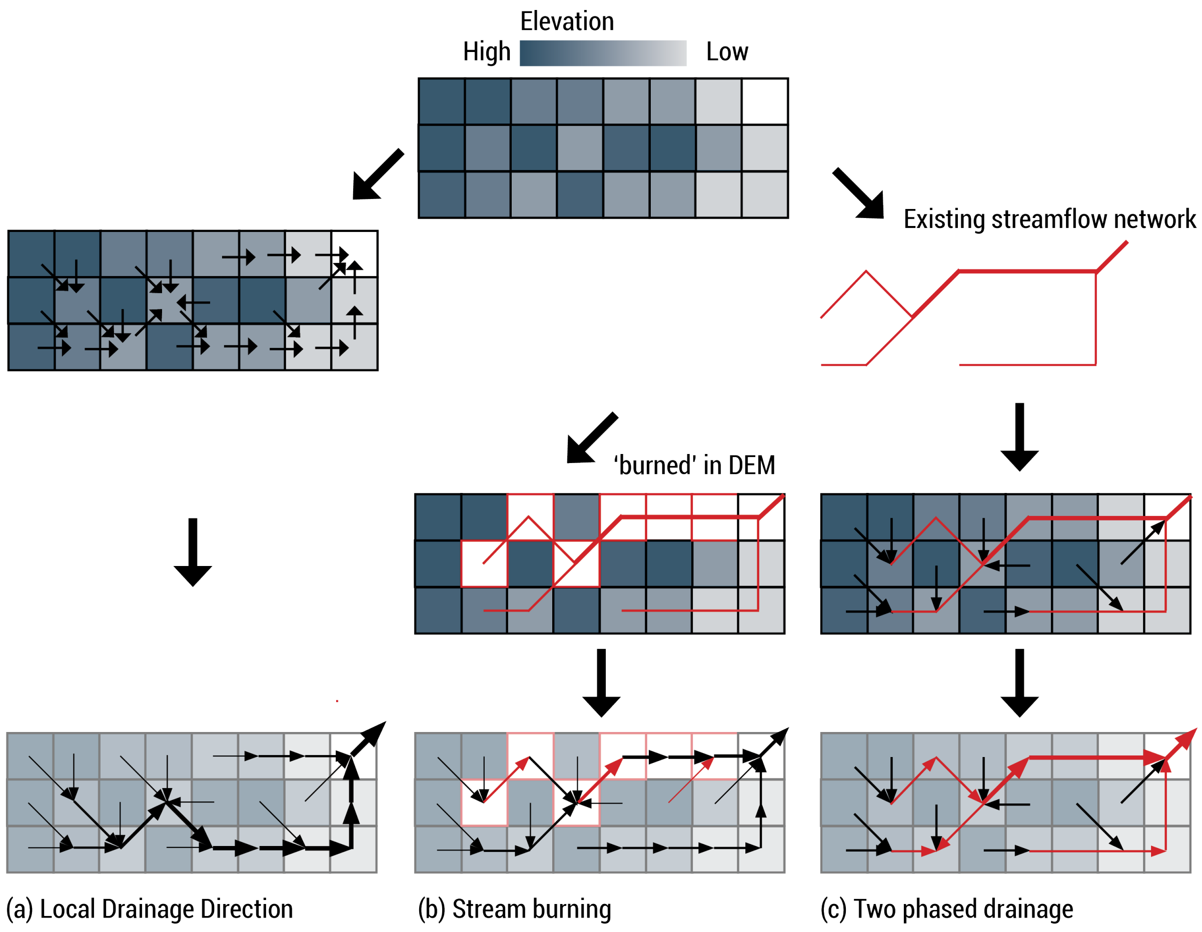


Figure 1.1: This figure depicts an overview of methods to determine flow direction in a spatially distributed model. At the top an impression of a Digital Elevation Model (DEM) is given with high elevation on the left side and lower elevations on the right side, as indicated by the colour of the grid cell. Case (a) depicts the derivation of a Local Drainage Direction (LDD) from the DEM: flow in each cell is directed into one of the eight surrounding cells with the lowest elevation. Case (b) depicts the derivation of a LDD after an existing streamflow network was 'burned' into the DEM. The grid cells and network sections that have been altered by this step are indicated in red. Case (c) shows the results of a two-phased LDD. The black part of the LDD matches with that of Case (a). The red parts of the of the LDD follow the direction of a pre-existing streamflow network. All LDDs, derived with different methods, show differences from one another.

would be eliminated and full focus can be directed towards refining model structures and our understanding of hydrological processes. To date, an approach for matching streamflow observations with streamflow simulations that uses streamflow characteristics of observations and simulations to select a representative cell has not been explored.

Therefore, this study aims to find a widely applicable algorithm that improves matching between distributed model output and observations for a range of hydrological models with varying spatial resolutions.
In contrast to previous studies that map streamflow observations based on proximity to the streamflow network and upstream drainage area, the algorithm developed in this study is designed to compare streamflow observations with streamflow simulations. Since shifts of 1 to 3 cells from the initial observation location in the model are often required to find representative simulations, this study proposes to create a search window around the initial grid cell. The characteristics of streamflow simulations, retrieved from grid cells in the search window, are compared with characteristics of the streamflow simulations. The algorithm is trained to select a representative streamflow simulation among all simulations in the search window.

For the development and evaluation of the algorithm, the objective is defined as *matching streamflow observations and streamflow simulations with high confidence*. This implies a primary focus on predicting *true* matches, followed by a focus on *maximizing* the number of true matches while

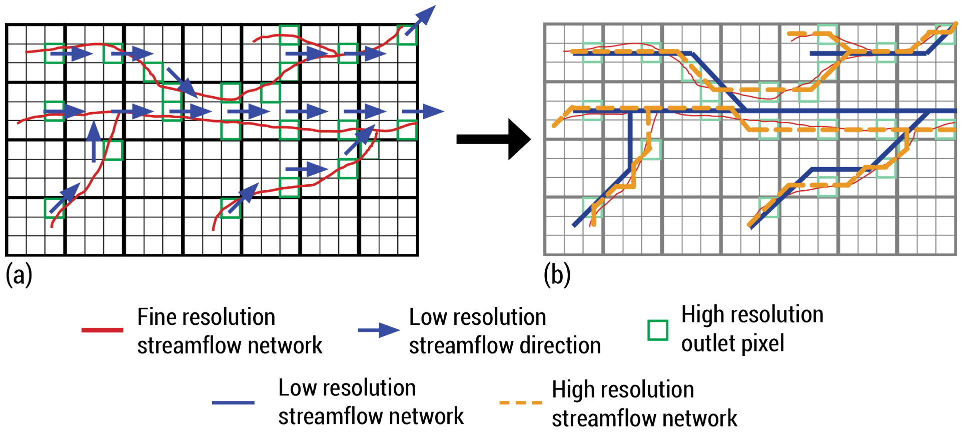

(a)
(b)
Fine resolution streamflow network
Low resolution streamflow direction
High resolution outlet pixel
Low resolution streamflow network
High resolution streamflow network

Figure 1.2: In this figure, issues that arise when a fine resolution streamflow network (indicated with red in (a) and (b)) is upscaled to a low or high resolution streamflow in network (b). The low resolution streamflow network is unable to capture the two separate streams of the original fine resolution streamflow network. The upstream part of the smaller stream (at the top part of the network) is merged into the stream at the lower part of the network. The illustration was adapted from Yamazaki et al. (2014)

maintaining a low number of false predictions.

Machine learning (ML) has proven to have the potential to extract patterns from large datasets and predict the response of complex and non-linear processes as occur in the field of hydrology (Shen, 2018). Therefore, a ML-based approach is applied in this study to match streamflow observations with streamflow simulations.
As streamflow signatures are focused on capturing hydrological behaviour, this study proposes to research the potential of streamflow signatures as features to match observations with model output.

The European Flood Alert System (EFAS) has been developed for flood predictions and warnings in European continental basins. The streamflow simulations in EFAS are produced with a hydrological model called LISFLOOD. The hydrological model produces streamflow simulations in continental Europe in a 5 km grid (Thielen et al., 2009). To calibrate and evaluate the model, a set of streamflow observations from the Global Runoff Database Centre (GRDC) (GRDC, 2015) have been placed onto the streamflow network of the model. The mapping of the total dataset of around 3200 gauge observations took a single person nearly 2.5 months of full time work (Mazzetti, 2021).
A subset of around 1000 stations was used for calibration of the hydrological model. Since these observations had to be mapped before the start of the calibration procedure, thus before the initial model run, no streamflow simulations were yet produced. This means that these stations were placed in the model based on coordinates and station information only, without information derived from streamflow characteristics. The remaining part of the dataset was used to evaluate the model performance after calibration.

In this study, EFAS simulations in combination with observations from the GRDC and a database of matching streamflow observations and simulations will be used to setup a dataset of streamflow signatures.
With this dataset, four supervised classification algorithms (Logistic Regression, Random Forest, Support Vector Machine and K-Nearest Neighbours) will be trained to match streamflow observations and streamflow simulations.
To find a most adequate algorithm setup, the following sub-objectives are defined: (1) find the most adequate supervised classification algorithm; (2) find a set of metrics that can distinguish differences in streamflow in a streamflow network; (3) find the optimal database formatting for an algorithm to train and predict with and (4) evaluate the matching algorithm compared to existing matching approaches.

After the problem definition in Chapter 1, the method to address the problem are described in Chapter 2 (p.5). Chapter 3 (p.14) reports the results obtained with the described approach. These are discussed in Chapter 4 (p.27), followed by drawn conclusions (Chapter 5, p.29) and recommendations (Chapter 6, p.30). More background information regarding hydrological modelling can be found in Appendix A (p.36). Background information on Machine learning can be found in Appendix B (p.40).

2 Methodology

In this chapter the setup of the algorithm is described (see Figure 2.1, p.6), starting with a description of data sources in section 2.1, that will be used for the creation of the dataset in section 2.2 (p.6). After this, the setup of a Machine learning (ML) algorithm in section 2.3 is described (p.9). In section 2.4 (p.10), dataset formatting options are discussed. Benchmark algorithms are described in section 2.5 (p.12). At last, the performance evaluation metrics are defined in section 2.6 (p.12).

2.1. Data sources

The goal of this research is to find or design an algorithm that optimally links streamflow observations to corresponding cells (pixels) of the streamflow simulations as calculated by the model. To achieve this goal, we used three key datasets: (1) gridded streamflow simulations from EFAS which will be addressed as *streamflow simulations* in this study; (2) streamflow observations from the Global Runoff Database Centre (GRDC) including their coordinates and upstream area. In this report, these will be called *streamflow observations*; (3) a manually created and checked dataset linking each GRDC streamflow observation location to the correct EFAS grid cell. In the remainder of this work, we will call these verified GRDC streamflow locations *representative locations*.

We obtained streamflow simulations from the European Flood Alert System (EFAS), developed flood for flood predictions and warnings in European continental basins (Thielen et al., 2009). The distribution of EFAS is managed by the Joint Research Centre (JRC) and executed in four centres: (1) the European Centre for Medium-Range Weather Forecasts (ECMWF) is the computational center, executing forecasts and hosting the system's platform; (2) the Swedish Meteorological and Hydrological Institute, Rijkswaterstaat and the Slovak Hydro-Meteorological Institute together form the dissemination centre that analyses and interprets the daily EFAS output; (3) the hydrological data collection centre consists of the Environment and Water Agency of the Regional Ministry for the Environment and Spatial Planning, and Soologic. This centre is responsible for the collection of historic and real-time river discharge and water level observations; (4) the meteorological data collection centre, responsible for the collection of historic and real-time meteorological data, is formed by KISTERS AG and Deutscher Wetterdienst (EFAS, 2012, Mazzetti et al., 2021).
In the latest version, the distributed model provides streamflow simulations on a 5 x 5 km grid with a 6-hourly timestep (Mazzetti et al., 2020). While the flood forecasts are only available to EFAS partners, historical data is publicly available via the Climate Data Store (CDS). Historical streamflow simulations are available from January 1st 1991 and are published with a 30-day delay. In this study, we used EFAS historical streamflow simulations from January 1st 1991 to the 31st of December 2020 will to calculate simulated streamflow signatures.
EFAS simulations are obtained by forcing the LISFLOOD model with meteorological data (Thielen et al., 2009). Distributed data layers such as land use, land cover and soil type determine cell properties. The Local Drainage Direction (LDD) map was derived by ”burning” the CCM River and Catchment Database (version 2.1) (Vogt et al., 2007) into a high resolution (100m) Digital Elevation Model (DEM) (Mazzetti, 2021). Both the CCM streamflow network and LDD were spatially up-sampled with the tracing method ”Flexible Location of Waterways” (FLOW) developed by Yamazaki et al. (2009) (Arnal et al., 2019). Streamflow is routed along the river network using a four-point implicit finite-difference solution of the kinematic wave equations (Arnal et al., 2019, Van Der Knijff et al., 2010).

We retrieved the streamflow observations used for calculation of observed streamflow signatures from the GRDC (Global Runoff Data Centre, 2020). The streamflow observations are provided with a daily timestep. Additional station information, such as station coordinates and an estimate of the upstream area are included in the metadata.

We made one requirement for an observation to be included in the study: stations were selected if the location in the EFAS grid was available in a database provided by ECMWF. This database contains observations that have been used for

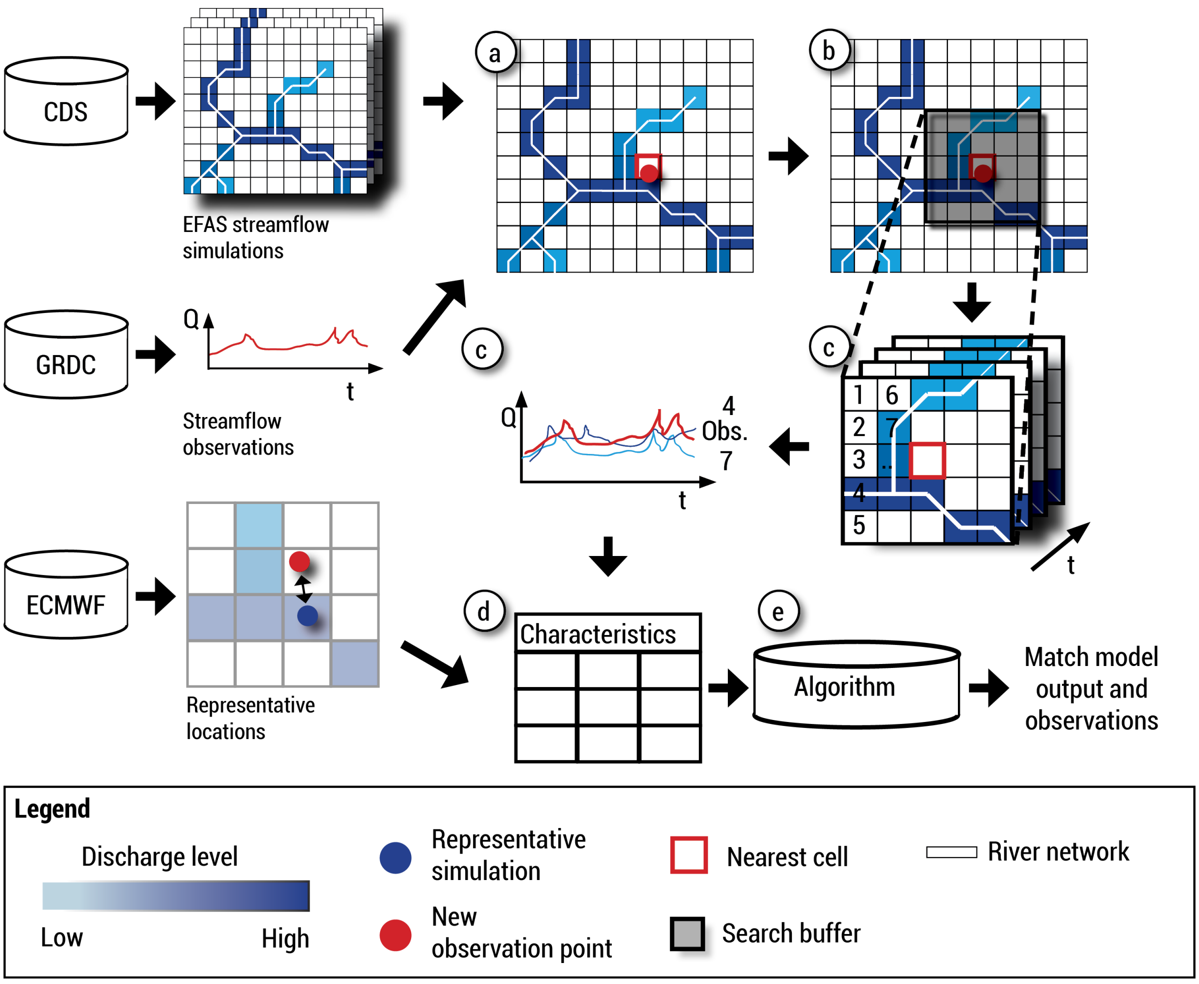


Figure 2.1: Global overview of processing steps required to setup the algorithm developed in this study. With steps a-d, the dataset is compiled: (a) each observation was placed in a cell the model grid closest to the provided coordinates; (b) a search window was specified around the initial location; (c) the streamflow simulations were extracted for each cell in the search window and (d) streamflow signatures were calculated based on the observations and simulations. The simulated streamflow signatures were labelled and added to a dataset with the observed streamflow signatures. With the created dataset, (e) an algorithm was trained to match model output and observations. The three sources used to compile the dataset are : Climate Data Store (CDS), Global Runoff Database Centre (GRDC) and European Centre for Medium-Range Weather Forecasts (ECMWF).

model performance evaluation and therefore already have been matched with their representative location in the EFAS model. The spatial distribution of the selected gauges in the model domain is depicted in Figure 2.2 (p.7).

A potential limitation of the algorithm developed in this study is the reliance on the availability of streamflow simulations. The dataset in this study was created with streamflow simulations from a calibrated model. However, calibration requires that a number of stations already have been mapped in the model before streamflow simulations have been produced. Matching un-calibrated streamflow simulations with observations is outside the scope of this study.

2.2. Dataset creation

We compiled the dataset by combining streamflow simulations, observations and representative locations with the steps depicted in Figure 2.1a (p.6): (a) observations were placed within the model grid based on their provided geo-location, and (b) a search window was defined around the initial location. All streamflow simulations within the search window were considered as potential representative simulations to match with the observations. Therefore, (c) the streamflow simulations were extracted and streamflow signatures were calculated for both the observed and simulated streamflow timeseries. The representative streamflow simulation, i.e. the grid cell in which the observation actually is according to the representative locations dataset, was identified in the search window. Rep-

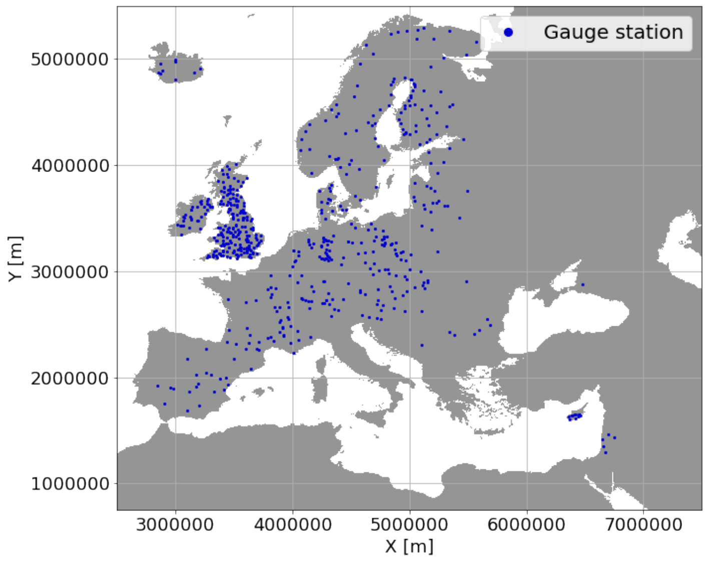


Figure 2.2: Location of 595 selected gauging stations retrieved from a database provided by European Centre for Medium-Range Weather Forecasts (ECMWF). The gauges have been placed within the EFAS model domain. The boundaries of the EFAS model domain correspond to the edges of this figure.

resentative streamflow simulations were labelled as 1 and non-representative streamflow simulations were labelled as 0. The streamflow signatures of all simulations and the observations were combined to (d) compile the dataset. In this study, a 9 by 9 search window was specified around the initial location of each observation for streamflow simulation extraction.

The streamflow signatures in the dataset have been selected to capture a wide range of streamflow characteristics. The selected signatures are depicted in Table 2.1 (p.8). Due to uniqueness of place (Beven, 2000), it is assumed that a unique set of streamflow signatures will characterize streamflow along a section in the streamflow network between confluences. By comparing streamflow simulations with streamflow observations based on streamflow signatures, a match of characteristic streamflow representation can be sought for.

Given that streamflow response on spatial scales of 10 to 15 km will vary little - given that no confluences or other disturbances occur - streamflow signatures will also be very similar. Therefore, a maximum of two streamflow simulations in a search window were labelled as additional acceptable matches.
We accepted simulations if (1) the mean streamflow was within a 1% range of the mean streamflow of the initial matching simulation, and (2) if the streamflow simulation was located directly upstream or downstream of the initial matching location.

Before calculation of the streamflow signatures, the 6-hourly EFAS simulations were resampled to match the daily timestep of the observations. Streamflow simulations in a search window were cropped to match the length of the corresponding streamflow observations.

Next, the streamflow signatures were calculated based on the whole timeseries and with seasonal time windows.

After calculation of the streamflow signatures was completed, the dataset was checked for missing values. For 1556 streamflow simulations that were included in the search windows, no streamflow signatures could be calculated as there was no streamflow simulation data available. This was attributed to the fact that the corresponding grid cells were located outside of the modelling area, so no streamflow simulations were produced for these cells. For about 15% of the streamflow observations, elevation data was not provided in the station's metadata. Missing values were extracted from the EU-DEM (Bashfield and Keim, 2011) and ASTER GDEM (Abrams et al., 2020).
Before extraction of elevation data from the DEMs, the RMSE between the available elevation data and elevation of the DEM was calculated to evaluate the quality of the additional DEM data sources. Of the total set of available elevation data, 100 samples were randomly selected. Based on the

Table 2.1: Overview of streamflow signatures used as features in Machine learning application in this study

Statistical distribution	**ID**	**Description**	**Formula**	**Reference**
Normal distribution parameters $[\mu, \sigma]$	Nm, Ns		$f_x(x) = \frac{1}{\sqrt{2\pi}}\frac{1}{\sigma}e^{\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]}$, $\mu = \frac{1}{n}\Sigma_{i=1}^{n} x_i$, $\sigma = \sqrt{\frac{\Sigma_{i=1}^{n}(x_i-\mu)^2}{n}}$	Bennett et al. (2013), Zhang et al. (2018)
Log-normal distribution parameters $[\mu, \sigma]$	Lm, Ls		$f_x(x) = \frac{1}{x\sqrt{2\pi}}\frac{1}{\sigma_Y}e^{\left[-\frac{(ln(x)-\mu_Y)^2}{2\sigma_Y^2}\right]}$ for $Y = ln(X)$ with: $\mu = exp(\mu_Y)$ and $\sigma = \mu\sigma_Y$	
Gumbel distribution parameters $[u, a]$	Gu, Ga		$f_x(x) = a * exp\left[-a(x-u) - e^{-a(x-u)}\right]$, $\mu = u + \frac{0.577}{a} \rightarrow u = \mu - \frac{0.577}{a}$, $\sigma = \frac{\pi}{a\sqrt{6}} \rightarrow a = \frac{\pi}{\sqrt{6}\sigma}$	
Poisson distribution parameters $[\lambda]$	Pl		$f_x(x) = exp(-x)\frac{\lambda^x}{x!}$ for $x = 0, 1, 2, ..$	
Gamma distribution parameters $[k, \theta]$	Gk, Gt		$f_x(x) = \frac{1}{\theta\Gamma(k)}\left(\frac{x}{\theta}\right)^{k-1} e^{\left(-\frac{x}{\theta}\right)}$ with $\Gamma(k) = \int_{z=0}^{\infty} z^{k-1}e^{-z}dz$, $\mu = k\theta \rightarrow k = (\mu/\sigma)^2$, $\sigma = \theta\sqrt{k} \rightarrow \theta = \sigma/(\mu/\sigma)$	
Correlation				
Pearson's n-lag auto-correlation	alag-n	Smoothness of hydrograph	$\rho_{auto} = \frac{cov(X,Y)}{\sigma_X\sigma_Y}$, with: $cov(X,Y) = \frac{\Sigma_{i=1}^{n}(x_i-\mu_x)(y_i-\mu_y)}{n-1}$	Winsemius et al. (2009), Euser et al. (2013)
Pearson's n-lag cross-correlation	clag-n	Cross-correlation between observation and simulation	$\rho_{cross} = \frac{cov(X,Y)}{\sigma_X\sigma_Y}$	Bennett et al. (2013)
Flow duration curve (FDC)				
FDC n-th percentile	fdc-q-n	Flow exceeded n% of the time	$FDC(Q_n)$	Yadav et al. (2007), Smakhtin (2001), Vogel and Fennessey (1994)
FDC Slope	fdc-s	Slope of the FDC between 33^{rd} and 66^{th} percentile	$FDC_{slope} = \frac{Q_{33}-Q_{66}}{0.33-0.66}$	Sawicz et al. (2011)
Peak Distribution	pkd	Slope of FDC, constructed from peaks only, between 10^{th} and 50^{th} percentile	$pkd = \frac{FDC(Q_{peaks,10})-FDC(Q_{peaks,50})}{0.9-0.5}$	Euser et al. (2013)
Low Flow Ratio	lf	Ratio of low flow to median flow	$LF = Q_{90}/Q_{50}$	Nijzink et al. (2018)
Hydrological Inidices				
Baseflow Index	bfi	Ratio of long-term baseflow to total streamflow	$BI = \sum Q_B/Q$	Sawicz et al. (2011)
Rising Limb Density	rld	Number of peaks divided by total duration of rising limbs in hydrograph	$RLD = N_{peaks}/T_{risinglimbs}$	Morin et al. (2002)
Declining Limb Density	dld	Number of peaks divided by total duration of declining limbs in hydrograph	$DLD = N_{peaks}/T_{declininglimbs}$	Shamir et al. (2005)
Richard-Baker Flashiness	rbf	Sum of absolute values of day-to-day changes in mean daily flows divided by sum of all daily flows	$RBF = \sum \lvert Q - Q_{mean} \rvert / \sum Q$	Baker et al. (2004), Holko et al. (2011), Kuentz et al. (2017)
Recession Curve Slope	rcs		$ln(-dQ) = ln(a) + rds * ln(Q)$	Vogel and Kroll (1992), Stoelzle et al. (2013)
High Flow Events	hfe, hfd	High flow event frequency and duration	$Q_{hf} \geq 9Q_{median}$	Westerberg and McMillan (2015)
Low Flow Events	lfe, lfd	Low flow spell frequency and duration	$Q_{lf} \leq 0.2Q_{mean}$	Westerberg and McMillan (2015)
Elevation	h	Elevation of streamflow observation station or simulation location		
Upstream area	upArea	Size of area contributing to streamflow at observation or simulation location		Toth (2013)

sample coordinates, the elevation was extracted from the DEM and compared with the elevation provided in the station's metadata. The RMSE of the EU-DEM was slightly lower than the ASTER GDEM with RMSE values of 76.8m and 78.2m respectively.
For some streamflow simulations the upstream area was not retrieved during the initial timeseries extraction from the EFAS model. The missing values were retrieved from the separate EFAS upstream area output layer (Mazzetti et al., 2020).

After data cleaning, the final step of the dataset creation consisted of expressing similarity between the streamflow simulations and streamflow observations. In this study, similarity was expressed by subtracting the observation signature values from the simulation signature values. Only the cross-correlation signatures were left out of this procedure, as they already are an expression for similarity between the streamflow observations and streamflow simulations.

2.3. Algorithm setup

After combination of streamflow simulations, streamflow observations and representative locations, we had created a dataset where streamflow observations have been linked with representative streamflow simulations. The similarities between representative and non-representative streamflow simulations and streamflow observations are described with streamflow signatures. With this dataset, a Machine learning (ML) algorithm can be trained to link streamflow observations to corresponding simulations. In ML terminology, an algorithm makes predictions based on characteristics called *features*. In this study, these features correspond to the streamflow signature values that express the similarity between streamflow simulations and streamflow observations. The algorithm is trained to predict a target value or *label*: in this study, the algorithm is trained to predict whether a streamflow simulation is representative for a streamflow observation.
As the goal is to train an algorithm to predict labels based on feature values, a supervised classification problem is defined. A subset of Machine learning (ML) algorithms is suited to solve supervised classification problems.

We selected four ML algorithms for exploration of their applicability to the supervised classification problem: (1) a Logistic Regression (LR), (2) a Random Forest (RF), (3) a Support Vector Machine (SVM) and (4) a K-Nearest Neighbours (K-NN) classifier. These algorithms have been selected for their simplicity and quick setup with the Python module Scikit-learn (Pedregosa et al., 2011). More information about the algorithms can be found in Appendix B.2.1 (p.41).
This study researches the potential to match streamflow observations with streamflow simulations and is, by the author's best knowledge, the first one to do so. Therefore, relatively simple algorithms are deployed to evaluate the potential in this exploratory phase.
Additionally, when comparing the size of the current dataset, with 595 matches between streamflow observations and simulations, it becomes apparent that the dataset size is relatively small (Ajiboye et al., 2015, Prusa et al., 2015). Generally, ML algorithm perform better than Deep learning (e.g. neural networks) algorithms when trained with small datasets. This is another argument to deploy ML algorithms in this exploratory study. Despite the limited dataset size, it is believed that - similar to other studies - the potential of this method can be evaluated (Kratzert et al., 2018).

ML algorithms require training to generalize relations from a dataset and make predictions based on unseen data. For performance evaluation, a dataset is randomly split in three parts: a training set (60-80% of the total dataset), a validation set and a test set (both 20-10% of the total dataset). After the algorithm is trained with the first set, the performance is evaluated with a set of unseen data, the validation set. Then, after multiple rounds of adjusting the algorithm based on training and evaluation performance, the algorithm performance is evaluated one final time with never before seen data, the test set.

Before features in the training, validation and test set can be used as input for an algorithm, they need to be preprocessed.
All features have been scaled and normalized between a range from zero to one. By scaling, each feature is equally weighed by the algorithm and the algorithm can converge faster during training. Additionally, normalization can improve the numerical stability of an algorithm (Kuhn et al., 2013). The scaling and normalisation functions are fitted on the training set and applied to the validation and test set.

In this study, the training and performance evaluation of each ML algorithm was conducted with a K-fold Cross Validation procedure, with $K = 5$. The dataset is split in two parts instead of three: a training and validation set (80-90% of the total dataset)

and a test set (20-10% of the total dataset). In this study, the training and validation set consisted of 85% of the total dataset (503 observations). The remaining 15% (92 observations) were assigned to the test set.
Due to the imbalanced nature of the of the displacements required to map streamflow observations on their representative location (see Figure 3.3), we conducted an intermediate sorting step before splitting the training and validation set and test set. The samples in the dataset were sorted based on the absolute number of cell shifts required to place observations on the representative location. Then, each sub-group was randomly split into the training and validation set, and the test set. We conducted this extra step to guarantee an equal representation of different numbers of cell shifts in both the training and validation and test set.
To start the K-fold Cross Validation procedure (see Figure 2.3, p.11), the training and validation set was split into K parts. Then, K iterations followed where the training set consists of K-1 parts. The remaining part was hold out as validation set. With each iteration, first the training and validation data was preprocessed. Then, an instance of an algorithm was trained with a unique combination of the split dataset. After K iterations, an ensemble of K algorithms have been trained and evaluated with every part of the training and evaluation set.

Each algorithm produces an output of the probability of a simulation being a match with an observation. The simulation corresponding to the maximum probability is predicted to be a representative for the observations by the algorithm.

Additionally, a minimum probability threshold can be set to accept a prediction. In this study, we set the probability threshold to exceed 0.5.

The K-fold Cross Validation procedure was repeated ten times. After ten repetitions, the performance was averaged. We used the variation between different performances to evaluate the stability of each algorithm.

For the final performance evaluation each algorithm in the ensemble made a prediction based on the features in the test set. The predicted probabilities of the ensemble were summed to obtain a soft ensemble vote: the simulation with the highest summed probability was predicted to be the representative simulation for the observation by the ensemble.

More details regarding Machine learning can be found in Appendix B (p.40).

2.4. Dataset formatting

Besides searching for the most adequate ML algorithm, we varied three factors that influence the shape of the dataset to evaluate the impact on the algorithm performance.
The first factor regards how spatial cohesion between streamflow simulations in a single search window is reflected in the input and output of the algorithm. Two options are considered.

The first option is the *Single Cell* format, where all simulations in a search window are considered individually - based on similarity between a single simulation and observation the algorithm predicts the probability of that simulation being representative for the observation, without comparison to other simulations. The prediction for each simulation consists of the probability that the simulation is not a match and the probability that the simulation is a match. The sum of probabilities in each prediction is equal to one. The simulation with the maximum probability of matching with the observation, optionally exceeding a threshold, is returned as representative for the observation.
The second option is the *Window* format, where the simulations in a search window are considered as a single input - the similarity of all simulations and the observations are interpreted at once by the algorithm so that spatial relations are preserved. The algorithm returns predictions in the shape of the search window. For each simulation in the search window, the probability of being representative for the observations is returned, so that the simulation with the maximum probability, optionally exceeding a threshold, is labelled as being representative for the observation.
With the *Window* formatting a binary multi-label classification problem is created. The RF and K-NN algorithms could be implemented for this type of problem without any alterations. The LR and SVM algorithm were adapted for the binary multi-class classification problem with a One-versus-Rest method.

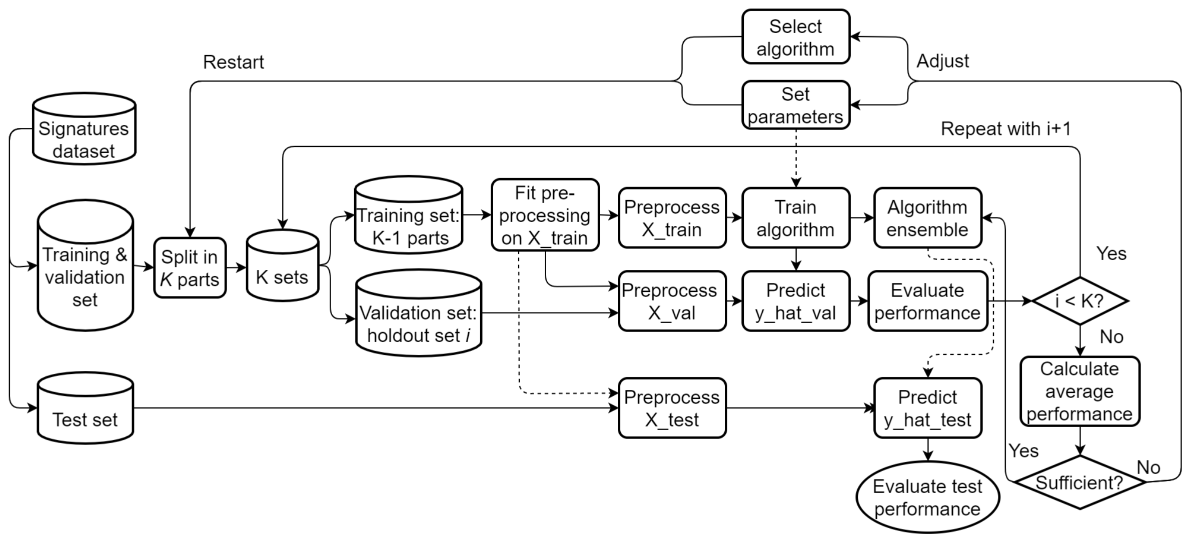


Figure 2.3: This illustration presents an overview of the K-fold Cross Validation: the signatures dataset is split into a training & validation set and a test set. The training & validation set if split into K parts. With K iterations, a training set and validation set are created from the K parts. The pre-processing steps - consisting of sub-sampling (for the Single Cell format), normalization, scaling and optional PCA transformation - are fitted onto the training set and applied to the validation and test set. After training of an algorithm, the performance is evaluated and the evaluation statistics are saved. After K iteration, the all performance evaluation statistics are combined for overall performance evaluation. The conclusion to adjust the algorithm is based on the average performance. If the performance is sufficient, all K algorithms trained during the iterations can form an ensemble that makes predictions on the pre-processed test set. The test performance can than be evaluated.

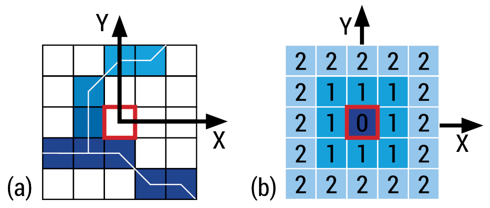


Figure 2.4: This illustration shows how the search window is defined. (a) depicts the the initial placement of an observation within the cell with the red edges. This cell becomes the origin of the X and Y axis. (b) depicts how many cells are located within a shift distance of 0, corresponding to a search window of 1 by 1 cells, to a shift of 2 cells, corresponding with a search window of 5 by 5 cells.

The second factor influencing the shape of the dataset, in terms of the number of samples, is the size of the search window (see Figure 2.4, p.11). The search window is centered around the initial location of the observation based on the station coordinates. The size of the search window corresponds to the number of streamflow simulations that are considered as potential matches for the observation.
In other words, in each search window a single representative streamflow simulation is sought for while the remaining streamflow simulations are labelled as non-representative. Dependent on the size of the search window, in a three-by-three window, one sample is labelled as representative while eight samples are labelled as non-representative. Increasing the search window with one in each direction, in a five-by-five window, the ratio of representative to non-representative simulations will have changed to one to twenty-four. This means that the dataset will be more imbalanced for larger search windows due to an increasing number of non-representative samples.
In this study, the performance sensitivity of algorithms for varying search window sizes was evaluated with a minimum search window size of three by three, and a maximum search window size of nine by nine.

Since positive samples of matching simulations are under-represented in the imbalanced dataset, it will be difficult for an ML algorithm to equally generalize relations between features of representative and non-representative samples. Therefore, in the *Single Cell* format, the training dataset is sub-sampled to create a balanced dataset for an algorithm to train with.

The final factor that influences the shape of the dataset, in terms of the number of features, is the application of a dimensionality reduction, specifically a Principal Component Analysis (PCA) (also see Section B.3.2, p.44).
A PCA is re-projects feature values in the dataset onto new axes, so that feature values lose their physical meaning. Each axis or Principal Component (PC) corresponds to a certain amount of information or variance of the original dataset. The

first PC will contain the largest amount of variance, followed by the second PC, and third PC, etc (Wold et al., 1987). Therefore, a certain percentage of the information stored by the features in the original dataset can be captured by a number of PCs that is less than the original number of features. The reduction in features can remove noise in the dataset, enhance relations between features and labels, and speed up training of the algorithm as simpler relations need to be established (Kuhn et al., 2013).
However, a large feature reduction can result in loss of essential information and negatively affect the algorithm performance. Therefore, a range of feature reduction levels, expressed as the percentage variance to be preserved, was applied to the dataset and the effect of the feature reduction on the algorithm performance is evaluated.

2.5. Benchmark performance

This study aims to find the most adequate ML algorithm setup to match streamflow observations with streamflow simulations by evaluating the performance of four different supervised classification algorithms - Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbours (K-NN) - with the dataset formatting options described in section 2.4 (p.10). Besides evaluating the different setups mutually, the added value of the setup is evaluated by checking if the setup can outperform benchmark algorithms.

The *Center Cell* (CC) benchmark selects the representative cell based on the provided observation station coordinates. In other words: the streamflow simulations closest to the initial location based on station coordinates, or the center of the search window, are predicted to be representative of the observations

Three benchmark algorithms are based on objective functions: the *NSE* benchmark calculates the Nash-Sutcliffe Efficiency (NSE) for every streamflow simulation in the search window. The representative simulation is selected as the simulation with the highest NSE score (Nash and Sutcliffe, 1970).
Similarly, the *KGE* benchmark calculates the Kling-Gupta Efficiency (KGE) for every streamflow simulation in search window and predicts the representative simulation based on the highest KGE score (Gupta et al., 2009).
At last, the *RMSE* benchmark calculates the Root Mean Squared Error (RMSE) for every streamflow simulation in the search window and predicts the streamflow simulation with the lowest RMSE as the representative streamflow simulation for the given streamflow observation.

2.6. Performance evaluation

An algorithm makes predictions based on features in the validation and test set. The predictions from algorithms can be compared with the labels of the validation and test set to evaluate the quality of the predictions. The focus of the performance evaluation is determined by the objective of this study, which is to *match streamflow simulations and observations with high confidence*.

The primary focus derived from this objective is that an algorithm makes *true* predictions. This can be measured by calculating the *precision* score of the predictions (Equation 2.1):

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

with TP the number of True Positives and FP the number of False Positives. The number of True Positives equals the number of correctly predicted matches by an algorithm. In contrast, the number of False Positives is equal to the number of incorrectly predicted matches by an algorithm. The precision score is a metric to express how many of matches predicted by the algorithm are correct and helps establish confidence in the output. The maximum precision score is equal to 1.

The secondary focus is to *maximize* the number of true predictions while maintaining a high precision score. This can be measured using the *recall* score (Equation 2.2):

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

with TP the number of True Positives and FN the number of False Negatives, which equals the number of matches that have not been identified by an algorithm. A high recall score indicates that the algorithm is able to classify a large portion of the total matches in the dataset.

A metric that balances precision and recall is the *F1 score* (Equation 2.3):

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.3}$$

and can help find a setup with the best overall trade-off between both metrics.

A final metric to support the performance evaluation is the *Balanced Accuracy* score (Equation 2.4):

$$\text{Balanced Accuracy} = 0.5 * \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right) \quad (2.4)$$

with TP the number of True Positives, FP the number of False Positives, TN the number of True Negatives, equal to the number of correctly identified non-representative streamflow simulations by an algorithm, and FN the number of False Negatives, that equals the number of incorrectly predicted non-representative streamflow simulations. This metric is developed to establish the accuracy for imbalanced datasets, as used in this study.

2.7. Data and Code availability

Code to reproduce all steps described above can be found in `https://github.com/mvanderven/matching_machine`.

3 Results

First, the created dataset (section 2.2, p.6) is analysed to establish the problem the algorithm needs to solve in section 3.1 (p.14). Second, the results of a Principal Component Analysis (PCA) on the training and validation set are presented in section 3.2 (p.14). Third, the performance of different test setups, based on the test and validation set, is compared in terms of precision (section 3.3, p.16), recall (section 3.4, p.18) and F1 scores (section 3.5, p.20). The trade-off between precision and recall is further analysed in section 3.6 (p.20). After that, the most adequate algorithm setups are evaluated in section 3.7 (p.22) and are applied to the test set. The test performance is analysed in section 3.8 (p.23).

3.1. Match analysis

The dataset created in this study contains the representative locations of 595 observations (see (3) in section 2.1, p.5). The displacements needed from the initial location of the streamflow observations in the model to the representative streamflow simulations were analysed.
Figure 3.1 (p.14) depicts an overview of displacements of streamflow observations from the initial location in the model to the representative streamflow simulations of the set of observations in the training and validation dataset. This dataset consists of 505 observations. In this dataset, 224 of the observations matched with the initial location.
Figure 3.2 (p.15) shows the displacements in X and Y directions. The peak for displacements in X and Y direction were both located around zero. The outliers were directed in both negative X and Y direction.
Figure 3.3 (p.15) shows the absolute shifts in X and Y direction. Here, can be seen that the number of observations requiring one cell displacement was higher than the number of observations that could be placed on the initial location in the model.
It can be concluded that 40% of the observations are located at the initial location in the model. Nearly 50% of the observations require one cell displacement, which can be a displacement in the X-direction, Y-direction or both. 90% of the matches are located within a 3 by 3 search window around the initial location in the model which corresponds to displacements observed in other studies (Coe, 2000, Irving et al., 2018, Kuentz et al., 2017, Li et al., 2015, Sutanudjaja et al., 2018, Wu et al., 2014). The remaining 10% of the observations required larger displacements towards a representative location in the model and could be caused by both uncertainties in the streamflow network of the model or uncertainties in the observation metadata (Döll and Lehner, 2002).

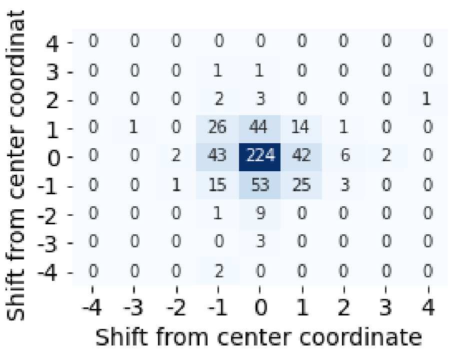


Figure 3.1: Displacements in test and validation set in search window view. The center, located at (0,0) represents the initial placement of each observation. A major part of the streamflow observations can be matched with representative streamflow simulations at the initial location. Most of the observations are located within one cell distance of the initial locations, or within a 3 by 3 search window around the initial location.

3.2. PCA

The application of a Principal Component Analysis (PCA) analysis is one of the variables in the dataset setup (see Section 2.4, p.10). The output of a PCA varies depends on the amount of variance that is preserved in the transformed dataset and influences the number of Principal Components (PC) in the transformed dataset.

Figure 3.4 (p.16) shows the results of a PCA transformation on the training and validation dataset. The bars show the percentage of variance explained per PC. The cumulative variance is indicated by the red line.

After removal of missing data, 76 streamflow signatures remained to be used as features for an

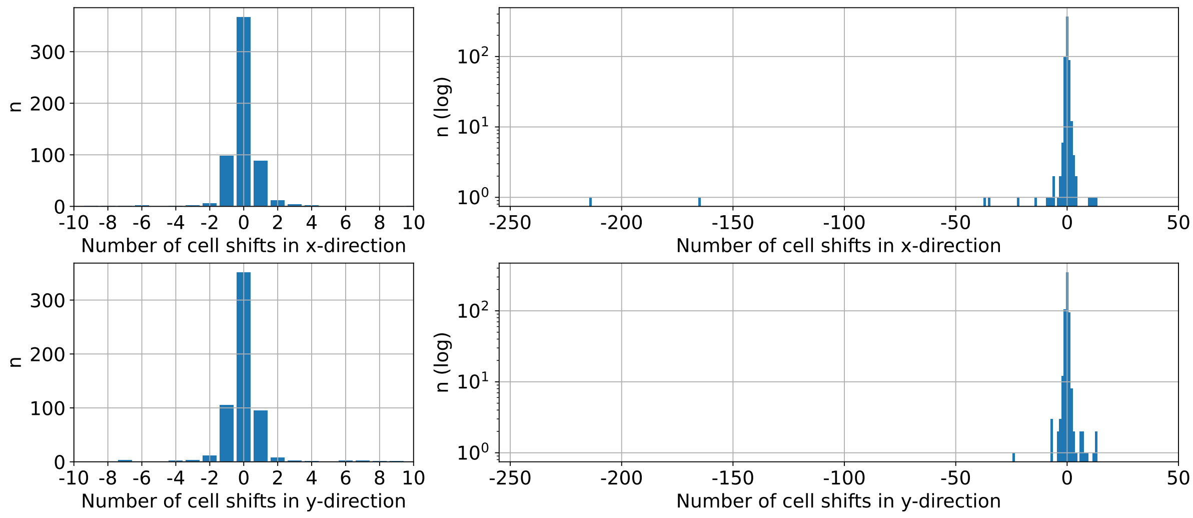


Figure 3.2: Distribution of displacements from an initial location needed to match streamflow observations with streamflow simulations in the model in X and Y direction. The sub-graphs on the left display the distribution between -10 and +10 cell shifts in X and Y direction. The sub-graphs on the right display the total distribution, including outliers at the left side of the graphs. The majority of required displacements fall between -1 and +1.

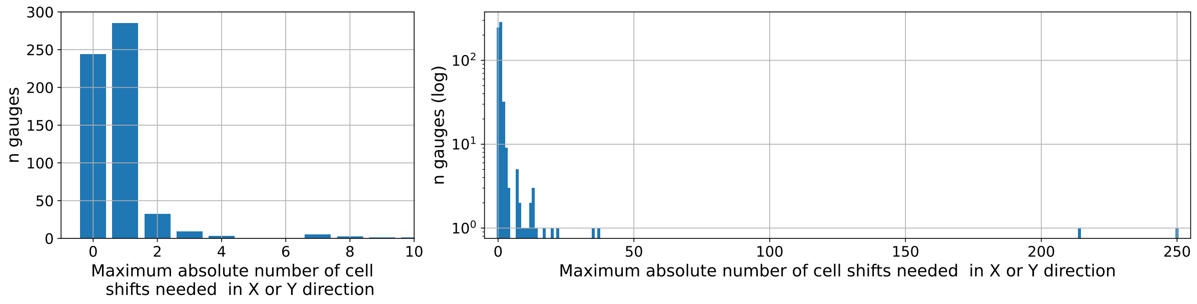


Figure 3.3: Distribution of absolute displacement from an initial location needed to match streamflow observations iwth streamflow simulations in combined X and Y direction. From the left sub-graph it can be observed that the number of observations requiring at least a one cell displacement in any direction is larger than the number of observations that can be matched with an initial location.

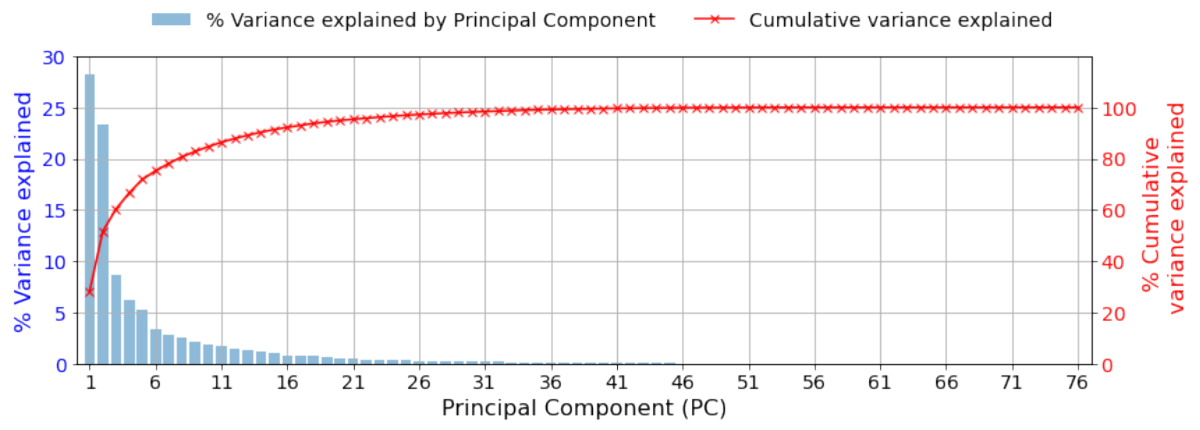


Figure 3.4: Results of an Principal Component Analysis (PCA) transformation on the training and validation set. On the X-axis the number of Principal Components (PCs) is displayed. The left Y-axis corresponds to the variance explained by each PC, displayed by a blue bar. The right Y-axis corresponds to the cumulative percentage of variance explained. The first two PCs contribute to over 50% of the variance explained of the original dataset.

algorithm. Following the PCA, the transformed dataset still contained 76 features. However, the 76 features now corresponded to 76 PCs and can no longer be interpreted as the signature values in Table 2.1 (p.8).
Based on Figure 3.4, it could be estimated that 26 PCs are needed to capture the variance of 99% of the original dataset with 76 features.
Over 50% of the variance can be captured by PC1 and PC2. These PCs are the main contributors, contributing 28% and 24% variance respectively. The variance explained from PC3 and other PCs drops below 10%.

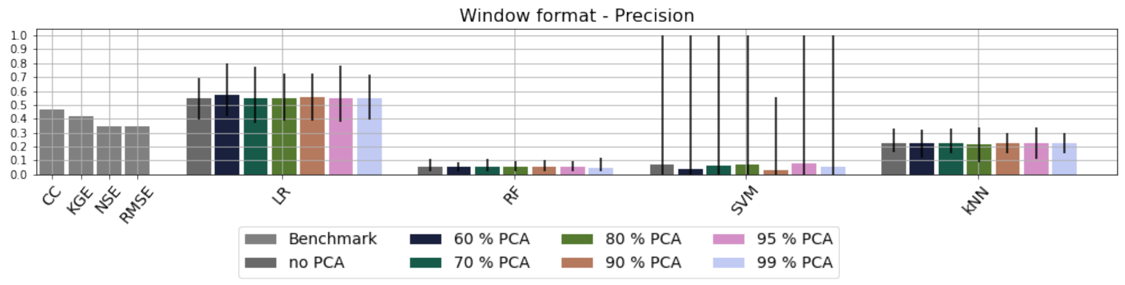
3.3. Precision

As defined in Chapter 1 (p.1) and section 2.6 (p.12), the primary objective for performance evaluation is to have high confidence in algorithm predictions. This can be expressed through the precision score, as a measure to express how many of the predicted matches are true matches (see Equation 2.1, p.12). Figures 3.5 and 3.6 (p.17) display the precision scores of different algorithm setups for a 3 by 3 and a 9 by 9 search window respectively.

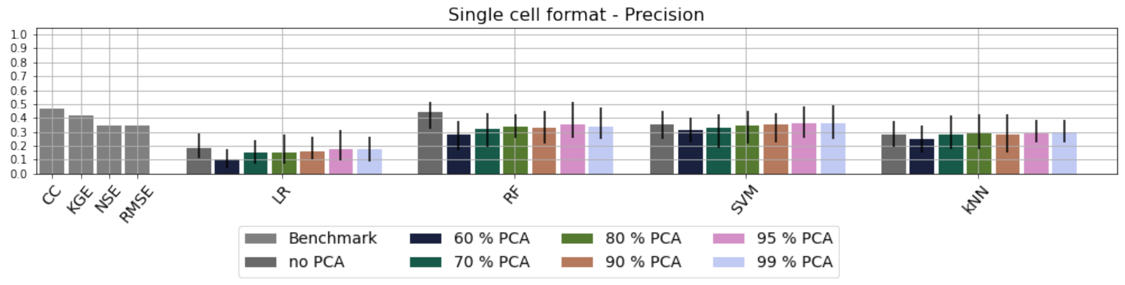
All figures display the benchmark scores on the left side. To the right, the precision scores for the Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM) and K-Nearest Neighbours (K-NN) algorithms are shown. These scores are the average performance after repeating K-fold Cross Validation procedure (with $K = 5$) ten times. The vertical black lines indicate the minimum and maximum performance of the ten repetitions. The experiment was repeated seven times with different PCA settings on the training and validation sets. The initial performance without PCA transformation can be compared with performances where a PCA transformation was applied preserving 60% to 99% variance explained with respect to the original dataset.

3.3.1. Minimum search window

Figure 3.5a (p.17) displays the precision scores for the Window formatted algorithm setup (see section 2.4, p.10) with a 3 by 3 search window.
The LR algorithm depicted the overall highest precision scores, varying around a precision score of 0.55. The LR algorithm showed the highest precision when trained and evaluated with a dataset with a strong feature reduction, with 60% variance explained after application of a PCA. For every experiment, the LR algorithm outperformed all benchmarks in terms of precision (CC=0.47, KGE = 0.42, NSE & RMSE = 0.346). The increased precision could be attributed to removal of noise after the PCA transformation, highlighting relations between features and target values.
The second best performing algorithm was the K-NN algorithm. The precision score for all experiments varied between 0.2 and 0.3. This was not high enough to exceed the benchmark scores.
The RF algorithm showed the lowest precision scores together with the SVM algorithm. For neither, the precision exceeded over 0.10.
The largest difference between minimum and maximum precision scores was observed for the SVM algorithm. In all test setups but one, the minimum and maximum precision score varied between 0 and 1.

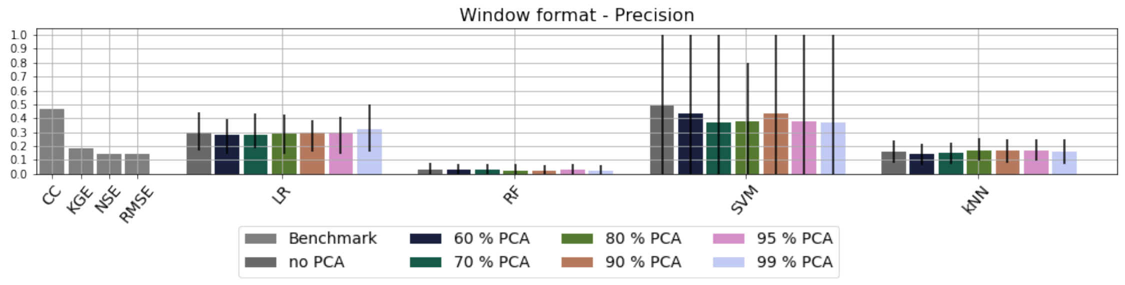


(a) Precision scores of algorithm setups with Window format in a 3 by 3 search window

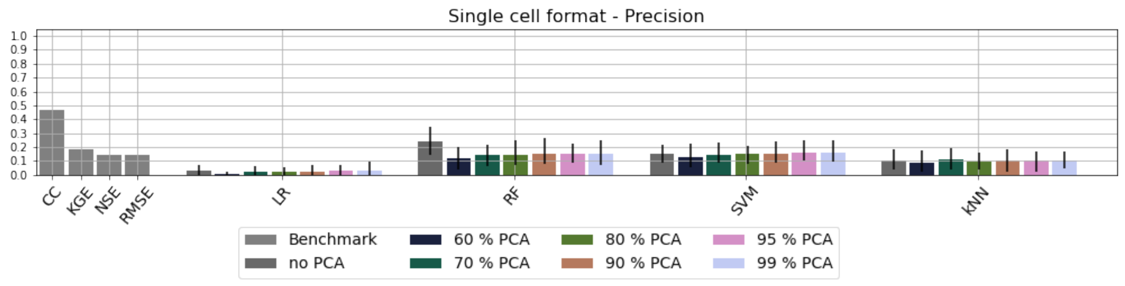


(b) Precision scores of algorithm setups with Single Cell format in a 3 by 3 search window

Figure 3.5: Precision scores with a 3 x 3 search window - On the left, the benchmark precision scores are displayed in light gray. In dark gray, the performance with the original dataset is displayed. The black, navy and green green bars correspond to the strongest feature reductions, with PCA transformation preserving 60%, 70% and 80% of the original variance respectively. The lighter colored bars - in brown, pink and mint - display the performance of algorithms trained with datasets after a PCA transformation with 90%, 95% and 99% of variance explained.



(a) Precision scores of algorithm setups with Window format in a 9 by 9 search window



(b) Precision scores of algorithm setups with Single Cell format in a 9 by 9 search window

Figure 3.6: Precision scores with a 9 x 9 search window - On the left, the benchmark precision scores are displayed in light gray. In dark gray, the performance with the original dataset is displayed. The black, navy and green green bars correspond to the strongest feature reductions, with PCA transformation preserving 60%, 70% and 80% of the original variance respectively. The lighter colored bars - in brown, pink and mint - display the performance of algorithms trained with datasets after a PCA transformation with 90%, 95% and 99% of variance explained.

Figure 3.5b (p.17) displays the precision scores for algorithms in a Single Cell format (see section 2.4, p.10) with a 3 by 3 search window.

The LR algorithm in Single Cell format displayed the lowest precision scores, varying between 0.1 and 0.18. A strong feature reduction improved the performance of the LR algorithm with Window formatting. However, in the Single Cell format the precision score was lowest (0.10) with a strong feature reduction.

The RF algorithm displayed the overall highest results of all test setups. The maximum precision score was obtained when the RF algorithm was trained with the original dataset without feature reduction (0.45). In this setup, the algorithm outperformed the NSE and RMSE benchmarks (0.35) and competed with the performances of the CC (0.47) and KGE (0.42) benchmarks.

The second best performing algorithm in Single Cell formatting was the SVM algorithm, with precision scores between 0.32 and 0.37. The SVM algorithm performed best when trained with the original dataset, or with a PCA transformed dataset with over 90% of the variance preserved.

The K-NN algorithm was the third best performing algorithm. The lowest score was obtained when the algorithm was trained with a PCA transformed dataset with 60% of variance explained (0.25). All other test setups showed precision scores varying around 0.28.

3.3.2. Maximum search window

Figure 3.6a (p.17) depicts the precision scores for algorithms in a Window format with a 9 by 9 search window.

The performance of the LR algorithm decreased with respect to the performance in Figure 3.5a, varying around 0.30. This score was high enough to outperform the KGE (0.18), NSE (0.15) and RMSE (0.15) benchmarks, but not enough to outperform the CC benchmark with a score of 0.47. The highest LR algorithm performance was obtained after training with a PCA transformed dataset preserving 99% of the variance.

The average precision scores of the SVM had increased with respect to the performance in the smaller search window, but still showed a large difference between the minimum and maximum scores.

The RF algorithm precision scores were the highest scores among the different test setups, varying around 0.20.

The K-NN algorithm performance varied around 0.16, which was enough to outperform the NSE and RMSE benchmarks (0.15) and to compete with the KGE benchmark (0.18). The CC benchmark was not outperformed in this setup.

Figure 3.6b (p.17) displays the precision scores for algorithm setups in a Single Cell format and a 9 by 9 search window. None of the algorithm setups were able to outperform all benchmarks. The RF trained with the original dataset was able to outperform the KGE (0.18), NSE (0.15) and RMSE (0.15) benchmarks with a precision score of 0.24. This was not high enough to compete with the CC benchmark (0.47).

3.4. Recall

As described in in Chapter 1 (p.1) and section 2.6 (p.12), the secondary objective is to maximize the number of true predictions. Recall (Equation 2.2, p.12) is a measure to express how many of the matches have been found. Figures 3.7 and 3.8 (p.19) depict the recall scores of different algorithm setups for a 3 by 3 and a 9 by 9 search window respectively.

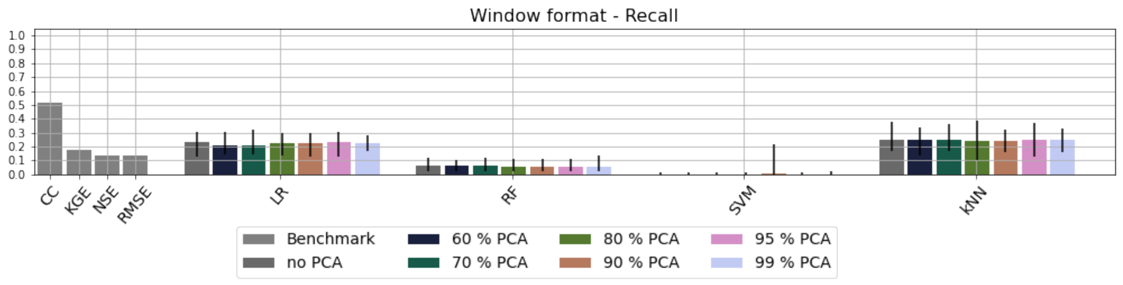
Each figure displays the benchmark recall scores on the left. On the right side, the recall scores of the algorithms - Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbours (K-NN) - are displayed. They are the averaged results of experiments where a K-fold Cross Validation procedure (with $K = 5$) was repeated ten times. The experiments were conducted seven times with various PCA applications.

3.4.1. Minimum search window

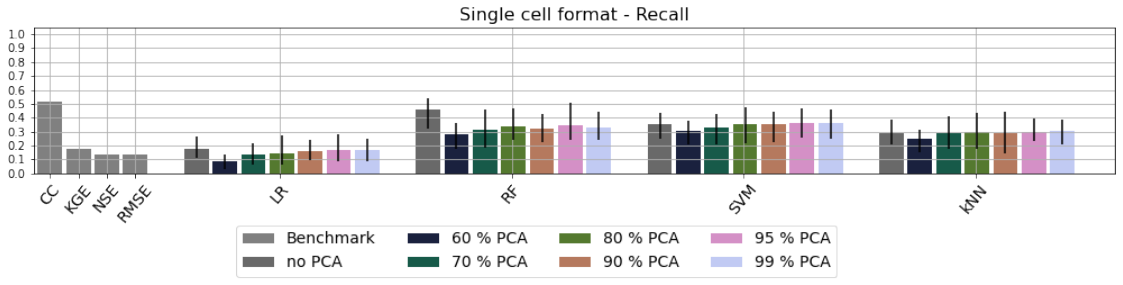
Figure 3.7a (p.19) depicts the recall scores for the algorithms with a Window format. None of the algorithms outperformed all benchmarks. The LR and K-NN algorithm were both able to outperform the KGE (0.18), the NSE (0.14) and the RMSE (0.14) benchmarks with recall scores around 0.22 and 0.25 respectively. The CC benchmark displayed the highest recall score equal to 0.51.

The recall scores of the SVM algorithm approximated 0. The recall scores of the RF algorithm were higher, with recall scores around 0.06. This was not high enough to outperform any of the benchmarks.

Figure 3.7b (p.19) displays the recall scores for the algorithms with a Single Cell format. The RF, SVM and K-NN algorithms outperformed the KGE (0.18), NSE (0.14) and RMSE (0.14) benchmarks. The RF algorithm trained with the original dataset had a competitive recall score (0.46) with respect to the CC benchmark (0.47).

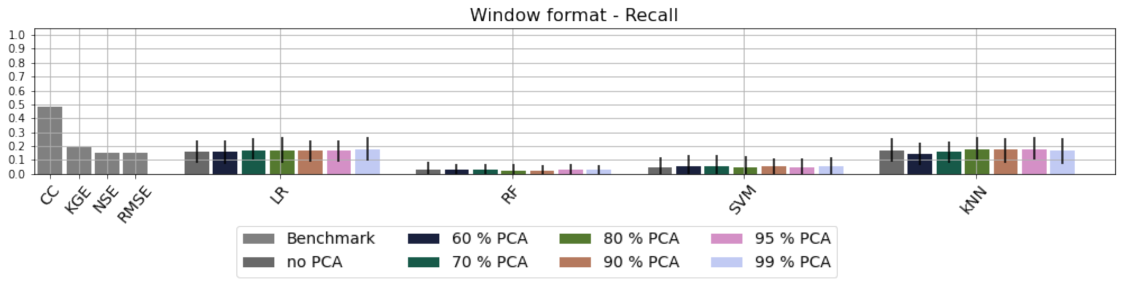


(a) Recall scores of algorithm setups with Window format in a 3 by 3 search window

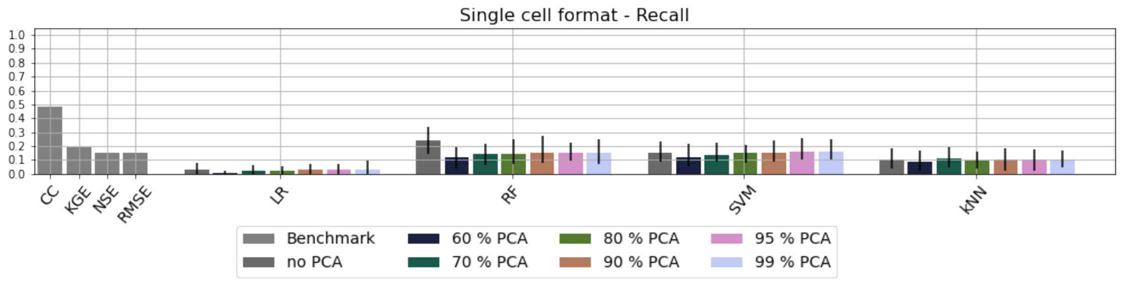


(b) Recall scores of algorithm setups with Single Cell format in a 3 by 3 search window

Figure 3.7: Recall scores with a 3 x 3 search window - On the left, the benchmark recall scores are displayed in light gray. In dark gray, the performance with the original dataset is displayed. The black, navy and green green bars correspond to the strongest feature reductions, with PCA transformation preserving 60%, 70% and 80% of the original variance respectively. The lighter colored bars - in brown, pink and mint - display the performance of algorithms trained with datasets after a PCA transformation with 90%, 95% and 99% of variance explained



(a) Recall scores of algorithm setups with Window format in a 9 by 9 search window



(b) Recall scores of algorithm setups with Single Cell format in a 9 by 9 search window

Figure 3.8: Recall scores with a 9 x 9 search window - On the left, the benchmark recall scores are displayed in light gray. In dark gray, the performance with the original dataset is displayed. The black, navy and green green bars correspond to the strongest feature reductions, with PCA transformation preserving 60%, 70% and 80% of the original variance respectively. The lighter colored bars - in brown, pink and mint - display the performance of algorithms trained with datasets after a PCA transformation with 90%, 95% and 99% of variance explained

3.4.2. Maximum search window

Figure 3.8a (p.19) depicts the recall scores for the algorithm with a Window format. The recall scores of the LR and K-NN algorithms varied around 0.16. Both algorithms showed competitive scores with the NSE and RMSE benchmark (0.15), but were not able to outperform the KGE (0.19) and CC (0.48) benchmarks.

Figure 3.8b (p.19) displays the recall scores for the algorithms with a Single Cell format. The RF algorithm (0.24) was able to outperform the KGE (0.19), NSE (0.15) and RMSE (0.15) benchmarks.

3.5. F1-score

To optimize the the balance between precision and recall, the F1-score (Equation 2.3, p.12) can be used since this is an expression based on both scores. Figures 3.9 and 3.10 (p.21) display the F1 scores for all test setups within a 3 by 3 and a 9 by 9 search grid.

At the left side of the figures, the F1 scores of the benchmarks are displayed. To the right, the F1 scores of the algorithms in different test setups can be compared.

3.5.1. Minimum search window

Figure 3.9a (p.21) displays the F1 scores of the algorithms with a Window formatting. It can be observed that both the LR (0.16) and K-NN (0.12) algorithms showed competitive results when compared with the KGE (0.12), the NSE (0.10) and the RMSE (0.10) benchmarks. The CC benchmark obtained the highest F1 score (0.24).

Figure 3.9b (p.21) shows the F1 scores of the algorithms with a Single Cell format. With a F1 score of 0.23, the RF algorithm showed competitive results compared with CC benchmark (0.24). The RF algorithm, trained with the original dataset, outperformed the KGE (0.12), NSE (0.10) and RMSE (0.10) benchmarks. The SVM (0.17) and K-NN algorithm (0.15) were able to outperform the KGE, NSE and RMSE benchmarks as well.

3.5.2. Maximum search window

Figure 3.10a (p.21) depicts the F1 scores of the algorithm setup with a Window format. None of the algorithms competed with the CC benchmark (0.24). The highest F1 scores among the algorithm setups could be observed for the LR algorithm (0.11) and K-NN algorithm (0.09). Both algorithms competed with the KGE (0.09), NSE (0.07) and RMSE (0.09) benchmarks. The F1 scores that were obtained with the RF algorithm were the lowest observed scores, varying around 0.01.

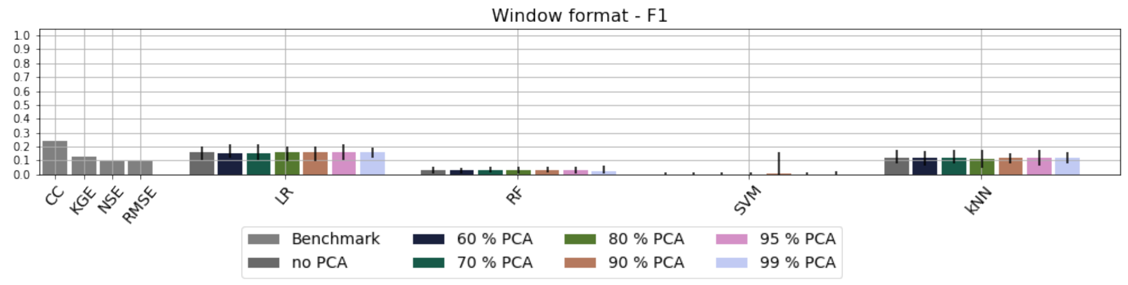
Figure 3.10b (p.21) displays the F1 scores of the algorithm setup with a Single Cell format. The RF showed the highest F1 scores of all algorithms (0.12). This was not enough to outperform the CC benchmark (0.24). This algorithm setup was the only one able to outperform the KGE (0.09), NSE (0.07) and RMSE (0.07) benchmarks.

3.6. Recall & precision trade-off

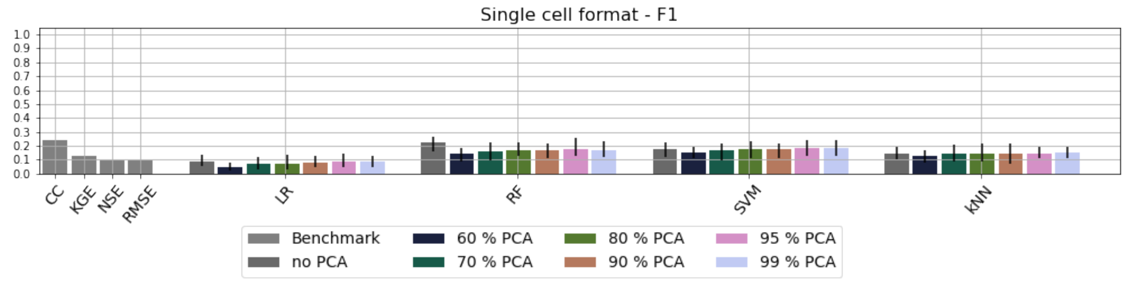
Besides finding a trade-off between precision and recall with help of the F1 score, an optimum can be found with a Pareto-dominance inspired analysis as displayed in Figure 3.11 and Figure 3.12 (p.22). In these figures, the precision scores of all algorithms are plotted on the X-axis, with the recall score plotted on the Y-axis.

Figure 3.11a (p.22) displays the trade-off between precision and recall for all algorithm setups in a 3 by 3 (left) and 9 by 9 (right) search window with a Window format respectively.
In the minimum search window, the the CC benchmark showed a combination of the highest performance and recall. In terms of recall, this benchmark dominated all other test setups. A higher precision score could be observed for the LR algorithm, at the cost of a lower recall score.
Moving to a larger search window on the right side, the KGE, RMSE and NSE benchmarks scored lower recall and precision scores. The LR has lost precision, but the recall score remained nearly equal. The CC benchmark remained in place. The SVM showed an increase in both precision and recall.

Figure 3.12 (p.23) depicts the trade-off between precision and recall for all algorithm setups in a 3 by 3 (left) and a 9 by 9 (right) search window with a Single Cell format. In both setups, the precision and recall scores had a linear relation. The CC benchmark dominated in precision and recall, closely followed by the RF and SVM algorithms. In case of the smallest search window, the K-NN algorithm was competitive with the SVM algorithm. The difference between a smaller and larger search window resulted in a decrease of both precision and recall scores for all algorithms. The CC benchmark remained in place. The KGE, NSE and RMSE performance reduced in both precision and recall when applied in the maximum search window.

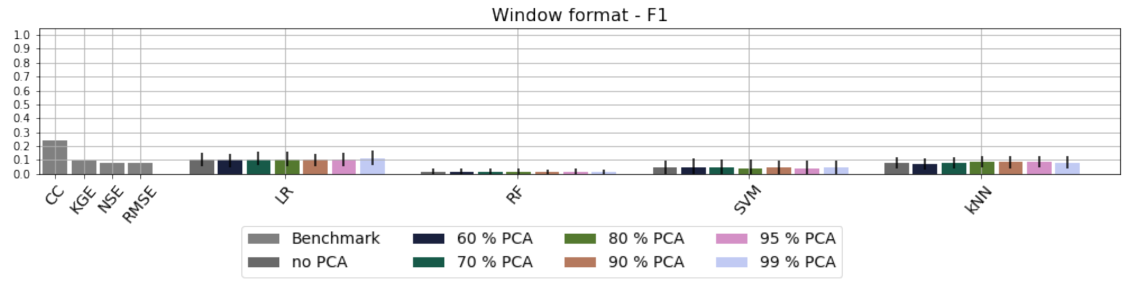


(a) F1 scores of algorithm setups with Window format in a 3 by 3 search window

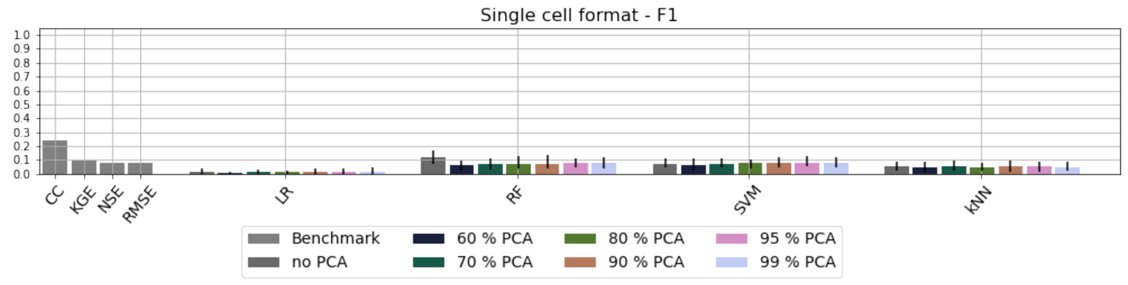


(b) F1 scores of algorithm setups with Single Cell format in a 3 by 3 search window

Figure 3.9: F1 scores with a 3 x 3 search window - On the left, the benchmark F1 scores are displayed in light gray. In dark gray, the performance with the original dataset is displayed. The black, navy and green green bars correspond to the strongest feature reductions, with PCA transformation preserving 60%, 70% and 80% of the original variance respectively. The lighter colored bars - in brown, pink and mint - display the performance of algorithms trained with datasets after a PCA transformation with 90%, 95% and 99% of variance explained.



(a) F1 scores of algorithm setups with Window format in a 9 by 9 search window



(b) F1 scores of algorithm setups with Single Cell format in a 9 by 9 search window

Figure 3.10: F1 scores with a 9 x 9 search window - On the left, the benchmark F1 scores are displayed in light gray. In dark gray, the performance with the original dataset is displayed. The black, navy and green green bars correspond to the strongest feature reductions, with PCA transformation preserving 60%, 70% and 80% of the original variance respectively. The lighter colored bars - in brown, pink and mint - display the performance of algorithms trained with datasets after a PCA transformation with 90%, 95% and 99% of variance explained.

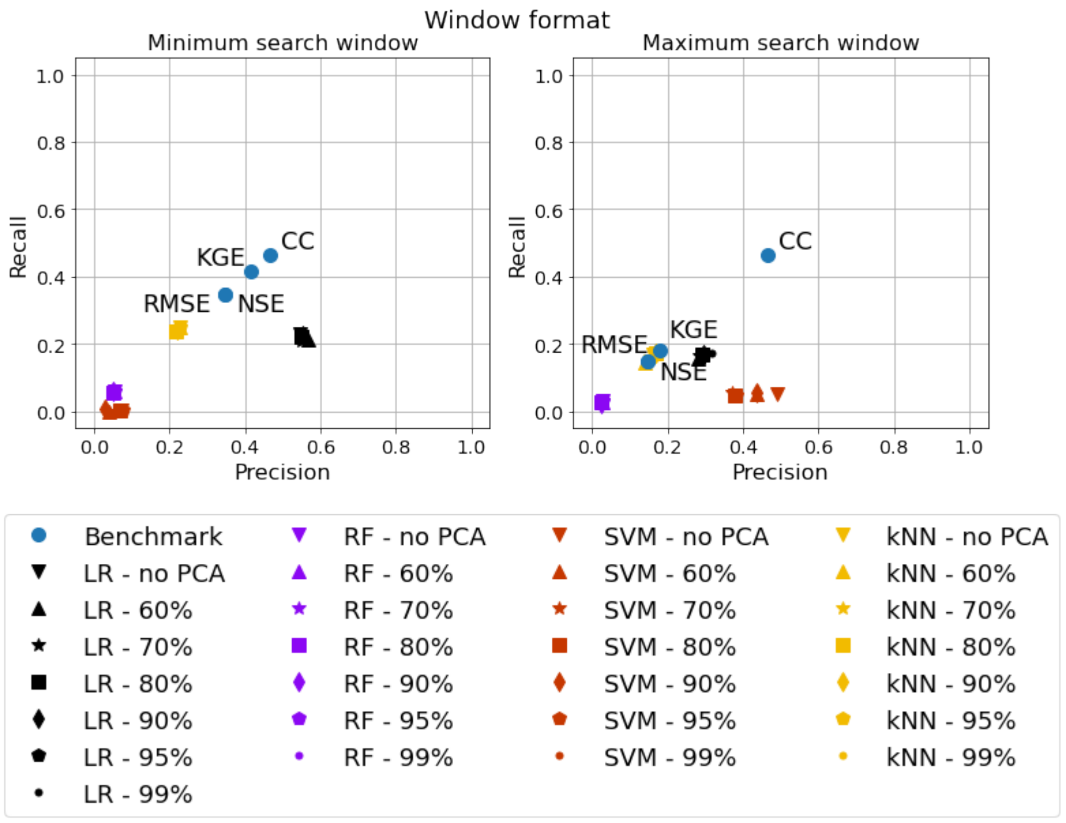


Figure 3.11: The illustration displays Pareto dominance inspired analysis for the relation between precision - displayed in the X-axis - and recall - displayed on the Y-axis. On the left, the algorithm scores with Window format in the 3 by 3 search window are displayed. On the right, the scores for the 9 by 9 search window are shown. The scores of each algorithm type is depicted in a different colour. Different Principal Component Analysis (PCA) applications are indicated with different marker shapes. The performance of the Logistic Regression (LR) and Support Vector Machine (SVM) algorithms are depicted in clusters. The Random Forest (RF) and K-Nearest Neighbours (K-NN) performances are spread over the X-axis.

3.7. Evaluation

The results presented prior in this chapter are based on the average performance of a ten times repeated K-fold Cross Validation with the training and validation set, where the formatting, search window size and feature reduction were varied. Here, the training and validation performances are evaluated to find the most adequate algorithm setups to match streamflow observations with simulations. The performances are evaluated based on the defined evaluation criteria: the primary evaluation criterion being precision, and the secondary objective being recall, or the best trade-off between precision and recall.

Observation of the trade-off between precision and recall in Figure 3.12 (p.23) leads to the conclusion that the LR algorithm is the least suited matching algorithm with Single Cell formatting of the algorithms considered in this study.
The SVM algorithm with Window formatting showed high precision scores in the maximum search window (Figure 3.6a, p.17), but also large differences between the minimum and maximum scores. In combination with low recall scores (Figures 3.7a & 3.6a, p.19), it can be concluded that the SVM algorithm makes predictions that are either correct, so that the precision is equal to 1, or incorrect, with precision is 0, and that the predictions have a low rate of True Positives. This leads to the conclusion that the SVM algorithm with Window format returns unstable results and is therefore unable to meet the research objectives in this study.

Generally, the performance of algorithms is highest in the 3 by 3 search window. Only the performance of the CC benchmark is not impacted with varying window sizes. The algorithm always classifies the cell based on the coordinates of the gauging station. This initial location is the center of the search window and is therefore not affected with varying search window sizes.
The average performance of the SVM algorithm also increases with respect to the minimum and maximum search window. This could be explained

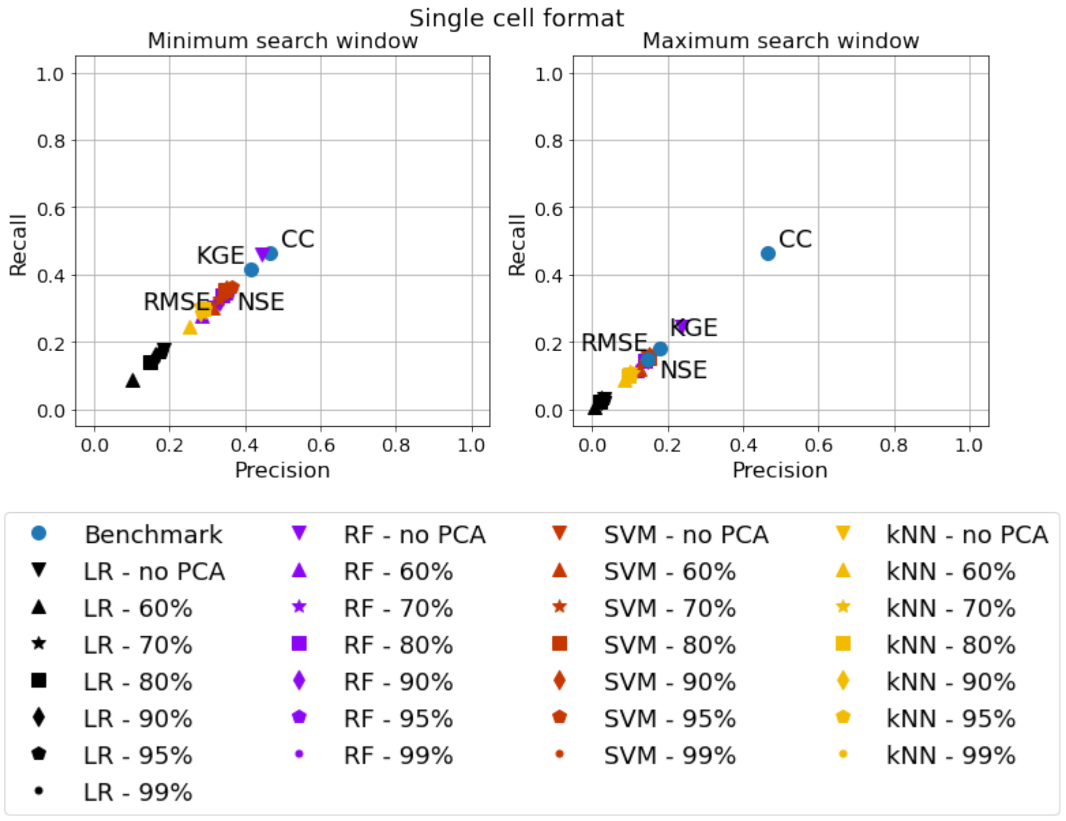


Figure 3.12: The illustration displays Pareto dominance inspired analysis for the relation between precision - displayed in the X-axis - and recall - displayed on the Y-axis. On the left, the algorithm scores with Single Cell format in the 3 by 3 search window are displayed. On the right, the scores for the 9 by 9 search window are shown. The scores of each algorithm type is depicted in a different colour. Different Principal Component Analysis (PCA) applications are indicated with different marker shapes. A linear relation between precision and recall is observed.

by the fact that the algorithm will either score very high or very low in terms of precision and that this has averaged to precision scores around 0.3.

Besides the size of the search window, the formatting of the dataset impacts the algorithm performance as well. In the 3 by 3 search window, the performance of the LR algorithm in Window format is competitive with the performance of the RF algorithm. However, when the search window was increased to 9 by 9, the performance of the Single Cell formatted algorithms showed a larger decline than the Window formatted algorithms.

From all algorithms with Window formatting, the LR algorithm is the most adequate. The performance of the algorithm is stable for various levels of feature reductions and search window sizes. The best performance is observed when the algorithm is trained with a dataset that has been transformed with a PCA preserving 60% of the variance of the original dataset. After the PCA transformation, the dataset is reduced to 3 PCs. This corresponds to the level of cumulative variance in Figure 3.4 (p.16) for 3 PCs.
In the Single Cell format, the RF algorithm trained with the original dataset shows the highest overall performance. Neither algorithms are able to outperform the all benchmarks in terms of both precision and recall. The RF algorithm in the 3 by 3 search window is the only algorithm that comes close to compete with the CC benchmark, as displayed in Figure 3.12 (p.23).

3.8. Test results

Based on the previous performance evaluation, two algorithm setups were selected to be applied on the test set: (1) the Logistic Regression algorithm with Window format and a PCA transformation preserving 60% variance and (2) the RF algorithm with Single Cell format and the original dataset. Predictions with these algorithm setups have been made with an ensemble of trained algorithms with after a single cycle in a 3 by 3 and a 9 by 9 search window.

The performance of all algorithm setups is displayed in Table 3.1 (p.24). The test set consisted of 92 observations. According to *n* in the table, in a 3 by 3 search window 82 of the 92 observations were located in the search window. In the 9 by 9 search window, 88 observations could be linked to a grid cell.

The highest performing benchmark was the CC model: nearly 47% of the matching simulations in the test set were located in the initial location. In both search windows, the second-best and identical performing benchmarks were the NSE and RMSE algorithms, followed by the KGE benchmark.

In the 3 by 3 search window, the precision of the RF algorithm was slightly higher than the precision score obtained by the LR algorithm. The RF algorithm was able to find a higher number of True Positives (39 vs. 37) and identified less False Positives (43 vs. 55) and False Negatives (43 vs. 45).
In terms of precision, both algorithms were competitive with the KGE score (0.4 and 0.42 vs. 0.4), but were outperformed by the NSE (0.47), RMSE (0.47) and CC (0.47) benchmarks.
In terms of recall, the RF (0.48) and LR (0.45) algorithms outperformed the KGE (0.17), NSE (0.21) and RMSE (0.21) benchmark, but not the CC benchmark (0.52).
The precision and recall scores were balanced with the F1 score. Based on this score, the algorithms (RF: 0.22, LR: 0.21) were competitive with the CC benchmark (0.25).

In the 9 by 9 search window, the Window formatted LR proved to be more robust for varying window sizes: the LR algorithm outperformed the RF algorithm in terms of precision (0.38 vs. 0.24), recall (0.40 vs. 0.25) and F1 (0.19 vs 0.12) scores. Both algorithms could outperform the KGE, NSE and RMSE benchmarks. The LR algorithm is closest to competing with the CC benchmark. By increasing the size of the search window, the CC benchmark performance is affected in terms of recall (0.52 vs 0.49) and F1 (0.25 vs 0.24) scores.

The precision scores of all algorithms and benchmarks were below 0.5. Looking at Equation 2.1 (p.12), this implies that of the predicted matches, less than half of the matches was correct. This means that little confidence can be attributed to the predicted matches of all algorithms and benchmarks.
In terms of recall, it can be concluded that the CC benchmark, with recall scores of 0.52 and 0.49 for the 3 by 3 and 9 by 9 search windows, was able to identify the highest number of True Positives (see Equation 2.2, p.12). Neither the RF and LR could compete with the CC benchmark in terms of recall, though these algorithms were able to outperform the KGE, NSE and RMSE benchmarks.

The classification results can be found in Appendix D (p.52). Figure 3.13 (p.26) displays an overview of classification results of the LR algorithm with Window format. The top row of the image displays the classification result indicated with an orange X. The true location is indicated with a red dot. The shade of blue in the background corresponds to the level of average simulated streamflow in each grid cell. On the second row, the shade of blue corresponds to the probability of each grid cell to matching with the observation.
From Figures D.3 and D.4 it can be concluded that the LR algorithm had a tendency to predict the center cell as matching cell. The probability distributions observed in the grid resemble the distribution

Table 3.1: Results test set with 3 by 3 and 9 by 9 search window

Search Window	Model	Precision	Recall	F1	Balanced Accuracy	n	TP	TN	FP	FN
3	CC	0.467	0.524	0.247	0.729	82	43	697	49	39
	KGE	0.400	0.171	0.120	0.571	82	14	725	21	68
	NSE	0.472	0.207	0.144	0.591	82	17	727	19	65
	RMSE	0.472	0.207	0.144	0.591	82	17	727	19	65
	LR - Window	0.402	0.451	0.213	0.689	82	37	691	55	45
	RF - Single cell	0.424	0.476	0.224	0.702	82	39	693	43	43
9	CC	0.467	0.489	0.239	0.741	88	43	7197	49	45
	KGE	0.174	0.182	0.089	0.586	88	16	7170	76	72
	NSE	0.196	0.205	0.1	0.597	88	18	7172	74	70
	RMSE	0.196	0.205	0.1	0.597	88	18	7172	74	70
	LR - Window	0.380	0.398	0.194	0.695	88	35	7307	57	53
	RF - Single cell	0.239	0.250	0.122	0.620	88	22	7176	70	66

displayed in Figure 3.1 (p.14).
The false classification results were assigned to two classes: incorrect (Figure 3.13a) and close (Figure 3.13c) classifications. Around 50% of the incorrect classification could be assigned to the close class, where the algorithm was able identify the correct streamflow segment, but predicted the matching grid cell to be too far upstream or downstream.
Figure 3.14 (p.26) displays a similar overview of the classification results of the RF algorithm. Again, correct, incorrect and close classes were identified. Here it can be observed that for the correct and close classifications, the probability patterns followed the average discharge patterns.

It can be concluded that the LR and RF algorithms are currently the most adequate algorithm setups that have been applied to the test set. Based on the performances described in Table 3.1 (p.24), and Figures 3.13 and 3.14 (p.26), the following can be concluded: the most adequate algorithm setups in this study (1) are not able to outperform the CC benchmark in terms of precision and recall; nor (2) meet the research objectives of predicting matches with high confidence. However, the high percentage of incorrect predictions that are close to the target cells shows potential for the method designed in this study.

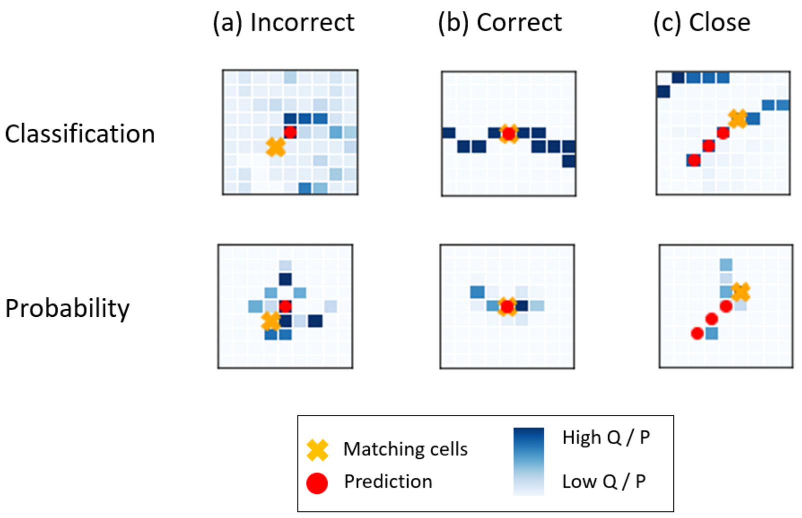


Figure 3.13: Summary of classification results of the LR classifier in Window format. The shade of blue in the plots corresponds to the level of average discharge in each grid cell on the top row and the probability of matching with an observation on the bottom row. The orange X indicates a cell marked as matching streamflow simulation for a set of streamflow observations. The red circle indicates the streamflow simulation marked as match by the algorithm.

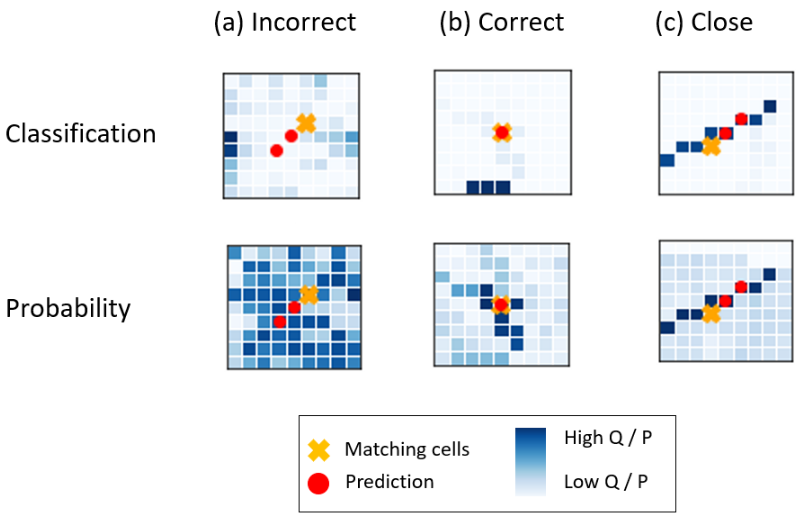


Figure 3.14: Summary of classification results of the RF classifier in Single Cell format. The shade of blue in the plots corresponds to the level of average discharge in each grid cell on the top row. In the bottom row the probability of matching with an observation on the bottom row. The orange X indicates a cell marked as matching streamflow simulation for a set of streamflow observations. The red circle indicates the streamflow simulation marked as match by the algorithm.

4 Discussion

The objective of this study was to improve the matching between hydrological model output and streamflow observations. To meet this objective, we explored the following: (1) four different types of Machine learning (ML) algorithms - Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM) and K-Nearest Neighbours (K-NN) (see section 2.3, p.9) - have been fitted on a dataset which varied in (2) two formatting types (Single Cell or Window), (3) two search window sizes (3 by 3 and 9 by 9) and (4) seven feature reduction approaches using a Principal Component Analysis (PCA) (see section 2.4, p.10).

The dataset created in this study was composed of streamflow observations, streamflow simulations and matches between observations and representative simulations (see section 2.1, p.5). The algorithm developed in this study was trained with this dataset to match observations with simulations.
A dataset influences the performance of a ML algorithm with both the quality of features in the dataset and the size of the dataset (Banko and Brill, 2001). Due to the relatively small number of matching samples (n=595) (Ajiboye et al., 2015, Prusa et al., 2015), the algorithm may have had too limited opportunities to generalize how characteristic similarities between observations and simulations determine whether a simulation is representative for an observation. It is believed that this limits the algorithm performance.
Just over half of the incorrect predictions made by the RF and LR algorithm are considered to be close errors (Figures 3.13 & 3.14, p.26). Here the algorithm is able to identify the correct section of the streamflow network, but makes a prediction too far upstream or downstream to be accepted as correct. This indicates that relations can be generalized from the dataset, but they may be too complicated for the algorithms that have been deployed in this study.

The search window size determines the number of non-matching samples and the degree of imbalance in the dataset. In the Single Cell format, the imbalance was mitigated by sub-sampling the training dataset: a single non-matching sample is randomly selected in every search window. Regardless of search window size, a new dataset is created where the number of non-matching samples equals the number of matching samples (n=595). This means that the size of the dataset is reduced and that non-matching samples are discarded, altering their representation in the resulting dataset.
An alternative approach could be to *up-sample* the dataset by duplicating the number of positive samples until a balanced dataset is created. In case of a 3 by 3 search window, a single matching sample is found in the search window together with eight non-matching samples. To create a balanced dataset, the number of positive samples should be multiplied by eight. With a 5 by 5 search window, a multiplication factor of twenty-four is needed for a balanced dataset. So, with up-sampling, all non-matching samples could be preserved in the training dataset. Additionally, the size of the dataset is at least increased by a factor eight (minimum search window size is 3 by 3) by duplicating matching samples.

Another option to make the Single Cell formatted algorithms more robust, could be by filtering streamflow simulations in the search window. By applying a filter, for example a threshold value for the average streamflow based on the observations, the number of potential matches is reduced. This could remove some of the noise in the set of potential matching simulations and therefore benefit the algorithm performance.
To prevent the Window formatted algorithms to predict center cells only (Figure D.3, p.55 and Figure D.4, p.56), the dataset could be altered: by removing the samples where the initial location of the streamflow observations is the representative simulation from the training dataset, the training is forced towards the samples where adjustments in location are necessary. This would also reduce the size of the dataset by 40 % which could further limit training opportunities for the algorithm.

Besides dataset size, the other factor influencing algorithm performance are the features. In this study, the features are based on streamflow data and metadata from streamflow observations and streamflow simulations only, consisting

of streamflow timeseries, elevation and upstream area. From the streamflow timeseries, streamflow signatures have been calculated (see Table 2.1, p.8).
Uncertainties in the observations and metadata of observations stations exist (Döll and Lehner, 2002, Vörösmarty et al., 2000). These uncertainties have been transferred to the dataset, which may limited the algorithm performance.
The algorithm's dependency on the availability of streamflow signatures limits the application of the algorithm in situations where no simulations are available. For example: prior to model calibration, no streamflow simulations have been generated. Therefore, mapping of observations used for calibration lies beyond the algorithm's capabilities. Potentially, the observations could be mapped in the model using the output from an initial pre-calibration model run. However, in this study, the performance of the algorithm is only evaluated based on the output of a calibrated hydrological model.
Another option to broaden the application of the algorithm could be achieved by including features derived from additional layers unrelated to streamflow timeseries. For example, information retrieved from the model's streamflow network could add valuable information to the dataset. As the EFAS streamflow network was not publicly available at a spatial resolution matching the EFAS model and constructing the streamflow network from the upstream area map was beyond the scope of this study, retrieving information from the streamflow network was omitted from this study.

Feature reduction was implemented as result of a PCA transformation. After PCA transformation, feature values have been transformed onto new axes and have lost physical meaning. Therefore, relating the PCs and original streamflow signature values is inconvenient.
An alternative method for feature reduction could be Random Forest Feature Importance. Based on the importance of features, the features with least importance could be removed from the set of features. As the feature values are not transformed, the preserved features could more easily be interpreted from a hydrological point of view.

The performance of the algorithms have been evaluated with data from a single hydrological model. However, finding a widely applicable solution for the matching problem could be further complicated by varying resolutions of hydrological models and streamflow networks. Contributing to this issue is that with the development of high resolution models (grid size < 1 km), a larger range of streamflow orders can be simulated, increasing the number of stream sections that can match with a single river station. Applying the algorithms on streamflow simulations with varying spatial resolutions remained outside the scope of this study.

Additionally, the algorithms in this study have been trained make a match between streamflow observations and a set of streamflow simulations.
In the Single Cell format, the match between a streamflow simulation and streamflow observations is evaluated for all individual streamflow simulations in a search window. With the independent evaluation, spatial relations in the search window are discarded: the match is evaluated based on features only. To preserve spatial cohesion, Single Cell predictions could be transformed to a Softmax classification, where the probabilities of all predictions sum to one.
Predictions made with a Window formatted algorithm could become more robust with a Softmax classification approach as well.

It should also be noted that predictions made by the algorithms during training and validation were subject to a threshold. Only matches predicted with a probability exceeding a threshold of $p \geq 0.5$ were accepted.
However, no threshold was enforced for the predictions made by the ensemble on the test set. By establishing a confidence level for ensemble predictions, the ensemble performance could become more robust.

Finally, in this study an algorithm was developed to match streamflow observations with streamflow simulations. An alternative approach to the matching problem could be to train models to predict whether or not the initial location of the observation is representative for the model, or whether the representative location for the observation is located within a search window or not. Based on algorithm predictions, part of the matching could be automated and efforts could be focused on difficult to locate matches.

5 Conclusions

We trained and applied four different ML algorithms to match streamflow observations with streamflow simulations: the Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM) and K-Nearest Neighbours (K-NN) algorithms. The algorithm performances were affected by the dataset formatting, where three parameters were varied: (1) a Single Cell versus Window formatting, (2) the size of the search window and (3) the application of a feature reduction with a Principal Component Analysis (PCA) transformation.
The performance of the algorithms have been compared with the benchmark performance of existing matching approaches. These approaches include matching based on geolocation and on comparison of streamflow observations and streamflow simulations.

In both the Window and Single Cell format, the overall highest performance was attributed to the SVM algorithm. In the Window format, the LR algorithm had a similar performance to the SVM algorithm. The RF algorithm scored the highest performance in the Single Cell format.

However, in the Window format, the algorithm always marked the initial location of an observation as matching simulation. An alternative application of an algorithm in Window formatting could be accomplished by establishing a probability threshold. This could help indicate whether the match at the initial location in the model should be rejected or accepted. By being able to automatically accept the streamflow simulation at the initial location in the model as a representative for a streamflow observation, the procedure to match observations with model output could become more efficient. This way, efforts can be focused to find the matching location of the observations whose initial location was rejected.

The performance of the Single Cell formatted algorithms was lower than the Window formatted algorithms. However, the matches predicted by the Single Cell formatted SVM and RF algorithm displayed a spatial variation where the Window formatted algorithms did not. About 25% of the matches predicted by the algorithms were correct. Additionally, around 50% of the incorrect predictions were located along the correct branch in the streamflow network, indicating that the algorithms have the potential to distinguish streamflow characteristics based on streamflow signatures.

Most streamflow observations were matched with streamflow simulations within a 3 by 3 search window around the initial location of the observations in the model. This was also the size of the search window that resulted in the highest performance for algorithms with Window and Single Cell formatting.

Based on the results of this study, we recommend the SVM and RF algorithms in Single Cell format as the most adequate algorithm setups to match streamflow observations with streamflow simulations. However, neither of these approaches was able to outperform the benchmarks in terms of recall and precision. Nevertheless, due to the the large number of near misses, it is believed that the method developed in this study does have potential to serve as a first pass in matching streamflow observations to ECMWF streamflow simulations, ultimately, lessening the burden of this time consuming task on ECMWF staff working on pressing hydrologic applications. Recommendations for further research are presented in the next section.

6 Recommendations

The research objective in this study was to improve matching between model output and observations. An algorithm has been developed, with the objective to predict matches with high confidence. The Logistic Regression (LR) and Random Forest (RF) algorithms have displayed potential to match streamflow observations with streamflow simulations. However, the current performance does not outperform benchmark performances.

It is believed that the performance of the algorithms could be increased by an increase in dataset size, the addition of features to the dataset and application of more advanced algorithms.

The Machine learning (ML) algorithms deployed in this study generalize relations from a dataset to produce predictions on new data. The dataset in this study consists of 595 matching samples. If the dataset size could be increased by collecting more matching samples, the algorithm would have more opportunity to generalize how characteristic similarities between streamflow simulations and streamflow observations determine when a simulations can be representative for an observation.
Also, the predictions of the algorithm are the interpretation of ensemble predictions. For this prediction, no confidence level is yet established. It is believed that finding an optimum confidence level could improve the robustness of the predictions.

The features in the dataset are based on streamflow timeseries. Streamflow signatures were calculated from streamflow observations and streamflow simulations. The upstream area and elevation, extracted from the simulation and observation metadata, were included as features as well.
Since the features mainly consist of streamflow signatures, the application of the algorithm is limited to scenarios where streamflow simulations are available. For example, in a situation before a model has been calibrated, no streamflow simulations are available. This prevents observations that are to be used during calibration to be mapped with use of the algorithm. It is therefore recommended to evaluate the performance of the algorithm while deriving streamflow signatures from streamflow simulations generated with an uncalibrated model.
The dependency on the availability of streamflow signatures could also be reduced by including features from other data sources such as the model's streamflow network.

It is also recommended to evaluate the performance of other ML algorithms, such as Boosted trees. The application of Deep learning could also prove to be beneficial for the algorithm performance, as Deep learning algorithms (e.g., neural networks) are generally able to capture more complex relations than ML algorithms.

The method in this study has been applied on a single hydrological model. For future research, it is recommended to research the limitations of the developed approach in terms of minimum and maximum spatial resolution by applying the algorithm on distributed models with varying spatial resolutions.

A final adaptation for developed algorithm could be to adjust the algorithm's application. In this study, the algorithm has shown potential to select representative streamflow simulations for streamflow observations.
Alternative approaches to improve matching between streamflow simulations and streamflow observations could increase the practical applicability of the algorithm. For example, the algorithm could be deployed to accept or reject the initial placement of an observation. Another option could be to determine whether a representative simulation is located within a search window to narrow down the manual matching procedure.

In this study a first exploration has been conducted for the applicability of streamflow signatures to match streamflow observations with streamflow simulations with a ML-based approach. Though the algorithm performances in this study do not meet the research objectives, the potential of this approach has been demonstrated. It is believed that the performance could be improved by adapting advanced ML or Deep learning algorithms, or by adjusting the application objective.

Bibliography

Michael Abrams, Robert Crippen, and Hiroyuki Fujisada. Aster global digital elevation model (gdem) and aster global water body dataset (astwbd). *Remote Sensing*, 12(7):1156, 2020.

Newsha K Ajami, Hoshin Gupta, Thorsten Wagener, and Soroosh Sorooshian. Calibration of a semi-distributed hydrologic model for streamflow estimation along a river system. *Journal of hydrology*, 298(1-4):112–135, 2004.

AR Ajiboye, Ruzaini Abdullah-Arshah, and Q Hongwu. Evaluating the effect of dataset size on predictive model using supervised learning technique. 2015.

L Arnal, SS Asp, C Baugh, A De Roo, J Disperati, F Dottori, R Garcia, M GarciaPadilla, E Gelati, G Gomes, et al. Efas upgrade for the extended model domain–technical documentation. Technical report, JRC Technical Reports, EUR 29323 EN, Publications Office of the European …, 2019.

David B Baker, R Peter Richards, Timothy T Loftus, and Jack W Kramer. A new flashiness index: Characteristics and applications to midwestern rivers and streams 1. *JAWRA Journal of the American Water Resources Association*, 40(2):503–522, 2004.

Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, pages 26–33, 2001.

A Bashfield and A Keim. Continent-wide dem creation for the european union. In *34th International Symposium on Remote Sensing of Environment. The GEOSS Era: Towards Operational Environmental Monitoring. Sydney, Australia*, pages 10–15. Citeseer, 2011.

Neil D Bennett, Barry FW Croke, Giorgio Guariso, Joseph HA Guillaume, Serena H Hamilton, Anthony J Jakeman, Stefano Marsili-Libelli, Lachlan TH Newham, John P Norton, Charles Perrin, et al. Characterising performance of environmental models. *Environmental Modelling & Software*, 40:1–20, 2013.

Keith Beven. Changing ideas in hydrology—the case of physically-based models. *Journal of hydrology*, 105(1-2):157–172, 1989.

Keith J Beven. Uniqueness of place and process representations in hydrological modelling. *Hydrology and earth system sciences*, 4(2):203–213, 2000.

Marc FP Bierkens, Victoria A Bell, Peter Burek, Nathaniel Chaney, Laura E Condon, Cédric H David, Ad de Roo, Petra Döll, Niels Drost, James S Famiglietti, et al. Hyper-resolution global hydrological modelling: what is next? “everywhere and locally relevant”. *Hydrological processes*, 29(2):310–320, 2015.

Michael T Coe. Modeling terrestrial hydrological systems at the continental scale: Testing the accuracy of an atmospheric gcm. *Journal of Climate*, 13(4):686–704, 2000.

Marcos Heil Costa, Carlos Henrique C Oliveira, Ricardo G Andrade, Thiago R Bustamante, Fabrício A Silva, and Michael T Coe. A macroscale hydrological data set of river flow routing parameters for the amazon basin. *Journal of Geophysical Research: Atmospheres*, 107(D20):LBA–6, 2002.

David M Danko. The digital chart of the world project. In *Proceedings of the Eleventh Annual ESRI User Conference*, volume 1, page 169. Environmental Systems Research Institute, 1991.

Cédric H David, David R Maidment, Guo-Yue Niu, Zong-Liang Yang, Florence Habets, and Victor Eijkhout. River network routing on the nhdplus dataset. *Journal of Hydrometeorology*, 12(5):913–934, 2011.

Christian W Dawson and Robert Wilby. An artificial neural network approach to rainfall-runoff modelling. *Hydrological Sciences Journal*, 43(1):47–66, 1998.

APJ De Roo, CG Wesseling, and WPA Van Deursen. Physically based river basin modelling within a gis: the lisflood model. *Hydrological Processes*, 14(11-12):1981–1992, 2000.

Martine G de Vos, Wilco Hazeleger, Driss Bari, Jörg Behrens, Sofiane Bendoukha, Irene Garcia-Marti, Ronald van Haren, Sue Ellen Haupt, Rolf Hut, Fredrik Jansson, et al. Open weather and climate science in the digital era. *Geoscience Communication*, 3(2):191–201, 2020.

Petra Döll and Bernhard Lehner. Validation of a new global 30-min drainage direction map. *Journal of Hydrology*, 258(1-4):214–231, 2002.

EFAS. European flood awareness system – efas, 2012. URL `https://www.efas.eu/en/european-flood-awareness-system-efas`.

Tanja Euser, HC Winsemius, Markus Hrachowitz, Fabrizio Fenicia, Stefan Uhlenbrook, and HHG Savenije. A framework to assess the realism of model structures using hydrological signatures. *Hydrology and Earth System Sciences*, 17(5):1893–1912, 2013.

Simone Fatichi, Enrique R Vivoni, Fred L Ogden, Valeriy Y Ivanov, Benjamin Mirus, David Gochis, Charles W Downer, Matteo Camporese, Jason H Davison, Brian Ebel, et al. An overview of current applications, challenges, and future trends in distributed process-based models in hydrology. *Journal of Hydrology*, 537:45–60, 2016.

Global Runoff Data Centre. Major River Basins of the World 2nd, rev. ext. ed., 2020. Koblenz, Germany: Federal Institute of Hydrology (BfG).

ST Graham, JS Famiglietti, and DR Maidment. Five-minute, 1/2°, and 1° data sets of continental watersheds and river networks for use in regional and global hydrologic and climate system modeling studies. *Water Resources Research*, 35(2):583–587, 1999.

GRDC. Global Runoff Database, 2015. Koblenz.

Hoshin V Gupta, Harald Kling, Koray K Yilmaz, and Guillermo F Martinez. Decomposition of the mean squared error and nse performance criteria: Implications for improving hydrological modelling. *Journal of hydrology*, 377(1-2):80–91, 2009.

Scott D Hamshaw, Mandar M Dewoolkar, Andrew W Schroth, Beverley C Wemple, and Donna M Rizzo. A new machine-learning approach for classifying hysteresis in suspended-sediment discharge relationships using high-frequency monitoring data. *Water Resources Research*, 54(6):4040–4058, 2018.

Ladislav Holko, Juraj Parajka, Zdeno Kostka, Peter Škoda, and Günter Blöschl. Flashiness of mountain streams in slovakia and austria. *Journal of Hydrology*, 405(3-4):392–401, 2011.

Markus Hrachowitz and Martyn Clark. Hess opinions: The complementary merits of top-down and bottom-up modelling philosophies in hydrology. *Hydrol. Earth Syst. Sci. Discuss., https://doi.org/10.5194/hess-2017-36, in review*, 2017.

Markus Hrachowitz, HHG Savenije, G Blöschl, JJ McDonnell, M Sivapalan, JW Pomeroy, Berit Arheimer, Theresa Blume, MP Clark, U Ehret, et al. A decade of predictions in ungauged basins (pub)—a review. *Hydrological sciences journal*, 58(6):1198–1255, 2013.

Rolf Hut, Niels Drost, Willem Van Hage, and Nick Van De Giesen. eWaterCycle II. In *2018 IEEE 14th International Conference on e-Science (e-Science)*, pages 379–379. IEEE, 2018.

Katie Irving, Mathias Kuemmerlen, Jens Kiesel, Karan Kakouei, Sami Domisch, and Sonja C Jähnig. A high-resolution streamflow and hydrological metrics dataset for ecological modeling using a regression model. *Scientific data*, 5(1):1–14, 2018.

Andrey Kolmogorov. Sulla determinazione empirica di una lgge di distribuzione. *Inst. Ital. Attuari, Giorn.*, 4:83–91, 1933.

Frederik Kratzert, Daniel Klotz, Claire Brenner, Karsten Schulz, and Mathew Herrnegger. Rainfall–runoff modelling using long short-term memory (lstm) networks. *Hydrology and Earth System Sciences*, 22(11):6005–6022, 2018.

V Krysanova, A Bronstert, and D-I Müller-Wohlfeil. Modelling river discharge for large drainage basins: from lumped to distributed approach. *Hydrological Sciences Journal*, 44(2):313–331, 1999.

Anna Kuentz, Berit Arheimer, Yeshewatesfa Hundecha, and Thorsten Wagener. Understanding hydrologic variability across europe through catchment classification. *Hydrology and Earth System Sciences*, 21(6):2863–2879, 2017.

Max Kuhn, Kjell Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.

Bernhard Lehner and Günther Grill. Global river hydrography and network routing: baseline data and new approaches to study the world's large river systems. *Hydrological Processes*, 27(15):2171–2186, 2013.

Bernhard Lehner, Kris Verdin, and Andy Jarvis. Hydrosheds technical documentation, version 1.0. *World Wildlife Fund US, Washington, DC*, pages 1–27, 2006.

Bernhard Lehner, Kristine Verdin, and Andy Jarvis. New global hydrography derived from spaceborne elevation data. *Eos, Transactions American Geophysical Union*, 89(10):93–94, 2008.

Hong-Yi Li, L Ruby Leung, Augusto Getirana, Maoyi Huang, Huan Wu, Yubin Xu, Jiali Guo, and Nathalie Voisin. Evaluating global streamflow simulations by a physically based routing model coupled with the community land model. *Journal of Hydrometeorology*, 16(2):948–971, 2015.

George Marsaglia, Wai Wan Tsang, Jingbo Wang, et al. Evaluating kolmogorov's distribution. *Journal of statistical software*, 8(18):1–4, 2003.

C. Mazzetti. Private Communication, 2021.

C. Mazzetti, D. Decremer, C. Blick, M. Carton de Wiart, F. Wetterhall, and C. Prudhomme. River discharge and related historical data from the european flood awareness system, v4.0, copernicus climate change service (c3s) climate data store (cds), 2020. Accessed: (17-06-2021).

Cinzia Mazzetti, Damien Decremer, and Christel Prudhomme. Major upgrade of the european flood awareness system, Jan 2021. URL `https://www.ecmwf.int/en/newsletter/166/meteorology/major-upgrade-european-flood-awareness-system`.

TM Mitchell. Machine learning, mcgraw-hill higher education. *New York*, 1997.

Alberto Montanari, G Young, HHG Savenije, D Hughes, T Wagener, LL Ren, D Koutsoyiannis, Christophe Cudennec, E Toth, S Grimaldi, et al. "panta rhei—everything flows": change in hydrology and society—the iahs scientific decade 2013–2022. *Hydrological Sciences Journal*, 58(6):1256–1275, 2013.

Efrat Morin, Konstantine P Georgakakos, Uri Shamir, Rami Garti, and Yehouda Enzel. Objective, observations-based, automatic estimation of the catchment response timescale. *Water Resources Research*, 38(10):30–1, 2002.

J Eamonn Nash and Jonh V Sutcliffe. River flow forecasting through conceptual models part i—a discussion of principles. *Journal of hydrology*, 10(3):282–290, 1970.

RC Nijzink, Susana Almeida, IG Pechlivanidis, Rene Capell, D Gustafssons, Berit Arheimer, Juraj Parajka, Jim Freer, Dawei Han, Thorsten Wagener, et al. Constraining conceptual hydrological models with multiple information sources. *Water Resources Research*, 54(10):8332–8362, 2018.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Christel Prudhomme, Nick Reynard, and Sue Crooks. Downscaling of global climate models for flood frequency analysis: where are we now? *Hydrological processes*, 16(6):1137–1150, 2002.

Christel Prudhomme, Ignazio Giuntoli, Emma L Robinson, Douglas B Clark, Nigel W Arnell, Rutger Dankers, Balázs M Fekete, Wietse Franssen, Dieter Gerten, Simon N Gosling, et al. Hydrological droughts in the 21st century, hotspots and uncertainties from a global multimodel ensemble experiment. *Proceedings of the National Academy of Sciences*, 111(9):3262–3267, 2014.

Joseph Prusa, Taghi M Khoshgoftaar, and Naeem Seliya. The effect of dataset size on training tweet sentiment classifiers. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 96–102. IEEE, 2015.

H Renssen and JM Knoop. A global river routing network for use in hydrological modeling. *Journal of Hydrology*, 230(3-4):230–243, 2000.

Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

K Sawicz, T Wagener, Murugesu Sivapalan, Peter A Troch, and G Carrillo. Catchment classification: empirical analysis of hydrologic similarity based on catchment function in the eastern usa. *Hydrology & Earth System Sciences Discussions*, 8(3), 2011.

James Kincheon Searcy. *Flow-duration curves*. US Government Printing Office, 1959.

Eylon Shamir, Bisher Imam, Efrat Morin, Hoshin V Gupta, and Soroosh Sorooshian. The role of hydrograph indices in parameter estimation of rainfall–runoff models. *Hydrological Processes: An International Journal*, 19(11):2187–2207, 2005.

Chaopeng Shen. A transdisciplinary review of deep learning research and its relevance for water resources scientists. *Water Resources Research*, 54(11):8558–8593, 2018.

Jalal Shiri, Amir Hossein Nazemi, Ali Ashraf Sadraddini, Gorka Landeras, Ozgur Kisi, Ahmad Fakheri Fard, and Pau Marti. Global cross-station assessment of neuro-fuzzy models for estimating daily reference evapotranspiration. *Journal of hydrology*, 480:46–57, 2013.

Vijay P Singh and David A Woolhiser. Mathematical modeling of watershed hydrology. *Journal of hydrologic engineering*, 7(4):270–292, 2002.

Vladimir U Smakhtin. Low flow hydrology: a review. *Journal of hydrology*, 240(3-4):147–186, 2001.

Dimitri P Solomatine and Durga Lal Shrestha. A novel method to estimate model uncertainty using machine learning techniques. *Water Resources Research*, 45(12), 2009.

Michael Stoelzle, Kerstin Stahl, and Markus Weiler. Are streamflow recession characteristics really characteristic? *Hydrology and Earth System Sciences*, 17(2):817–828, 2013.

Edwin H Sutanudjaja, Rens Van Beek, Niko Wanders, Yoshihide Wada, Joyce HC Bosmans, Niels Drost, Ruud J Van Der Ent, Inge EM De Graaf, Jannis M Hoch, Kor De Jong, et al. Pcr-globwb 2: a 5 arcmin global hydrological and water resources model. *Geoscientific Model Development*, 11(6): 2429–2453, 2018.

Jutta Thielen, J Bartholmes, M-H Ramos, and A de Roo. The european flood alert system–part 1: concept and development. *Hydrology and Earth System Sciences*, 13(2):125–140, 2009.

E Toth. Catchment classification based on characterisation of streamflow and precipitation time series. *Hydrology and Earth System Sciences*, 17(3):1149–1159, 2013.

US Geological Survey. Gage Locations (GageLoc.shp) indexed to the NHDPlus Version 2.1 stream network. *US Environmental Protection Agency*, 2015.

USGS. HYDRO1k Elevation Derivative Database, 2000.

JM Van Der Knijff, Jalal Younis, and APJ De Roo. Lisflood: a gis-based distributed model for river basin scale water balance and flood simulation. *International Journal of Geographical Information Science*, 24(2):189–212, 2010.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Richard M Vogel and Neil M Fennessey. Flow-duration curves. i: New interpretation and confidence intervals. *Journal of Water Resources Planning and Management*, 120(4):485–504, 1994.

Richard M Vogel and Charles N Kroll. Regional geohydrologic-geomorphic relationships for the estimation of low-flow statistics. *Water Resources Research*, 28(9):2451–2458, 1992.

Jürgen Vogt, Pierre Soille, A De Jager, E Rimaviciute, W Mehl, S Foisneau, K Bodis, J Dusart, ML Paracchini, P Haastrup, et al. A pan-european river and catchment database. *Report EUR*, 22920:Ispra, 2007.

CJ Vörösmarty, Balazs M Fekete, M Meybeck, and Richard B Lammers. Global system of rivers: Its role in organizing continental land mass and defining land-to-ocean linkages. *Global Biogeochemical Cycles*, 14(2):599–621, 2000.

Wen-chuan Wang, Kwok-wing Chau, Dong-mei Xu, and Xiao-Yun Chen. Improving forecasting accuracy of annual runoff time series using arima based on eemd decomposition. *Water Resources Management*, 29(8):2655–2675, 2015.

IK Westerberg and Hilary K McMillan. Uncertainty in hydrological signatures. *Hydrology and Earth System Sciences*, 19(9):3951–3968, 2015.

HC Winsemius, Bettina Schaefli, A Montanari, and HHG Savenije. On the calibration of hydrological models in ungauged basins: A framework for integrating hard and soft hydrological information. *Water Resources Research*, 45(12), 2009.

Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

Eric F Wood, Joshua K Roundy, Tara J Troy, LPH Van Beek, Marc FP Bierkens, Eleanor Blyth, Ad de Roo, Petra Döll, Mike Ek, James Famiglietti, et al. Hyperresolution global land surface modeling: Meeting a grand challenge for monitoring earth's terrestrial water. *Water Resources Research*, 47(5), 2011.

Huan Wu, Robert F Adler, Yudong Tian, George J Huffman, Hongyi Li, and JianJian Wang. Real-time global flood estimation using satellite-based precipitation and a coupled land surface and routing model. *Water Resources Research*, 50(3):2693–2717, 2014.

Maitreya Yadav, Thorsten Wagener, and Hoshin Gupta. Regionalization of constraints on expected watershed response behavior for improved predictions in ungauged basins. *Advances in water resources*, 30(8):1756–1774, 2007.

D Yamazaki, T Oki, and S Kanae. Deriving a global river network map and its sub-grid topographic characteristics from a fine-resolution flow direction map. *Hydrology and Earth System Sciences*, 13 (11):2241–2251, 2009.

Dai Yamazaki, Fiachra O'Loughlin, Mark A Trigg, Zachary F Miller, Tamlin M Pavelsky, and Paul D Bates. Development of the global width database for large rivers. *Water Resources Research*, 50 (4):3467–3480, 2014.

Yongqiang Zhang, Francis HS Chiew, Ming Li, and David Post. Predicting runoff signatures using regression and hydrological modeling approaches. *Water Resources Research*, 54(10):7859–7878, 2018.

A
Theoretical Background: Hydrology

Since this study aims to combine the disciplines of hydrology and machine learning, this chapter intends to provide a theoretical background for hydrologic modelling.

A.1. Hydrologic Modelling

Hydrologic models are a simplification of real-life hydrologic systems that describe the movement and distribution of water within a predefined system or area.

A hydrologic system is defined as an area where, through various states and fluxes, all incoming and stored water will eventually flow through a point where it leaves the system, defined as the outlet point, which is often located in a stream. Water can enter the system as precipitation. Dependent on where it lands, the precipitated water can enter various states and fluxes, but will eventually all pass the same outlet point. Dependent on topography, land cover, temperature, e.g., the precipitated water could, for example, be intercepted and leave the system through evaporation, infiltrate the ground and enter the groundwater storage, infiltrate the ground and be transpired through vegetation, freeze, or directed into a stream as runoff, or a combination of all of the above. The path through different states and fluxes determine the time it takes for precipitation to reach the outlet point. For example, runoff that reaches streamflow quickly, will leave the system much faster than precipitation that is infiltrated and reaches a stream through subsurface flow. The combination of states and fluxes determine the response behavior of the system, which is characterized as changes in discharge level at the outlet point as reaction to precipitation or lack of it.

The model can be built with a bottom-up or top-down approach (Hrachowitz and Clark, 2017).
The micro-scale bottom-up approach applies the hydrologic community's physical understanding of small scale processes. However, spatial heterogeneity makes it difficult to describe all processes at a global scale (Beven, 1989, Hut et al., 2018).
The large-scale top-down approach results in a conceptual model, describing the system as a set of states and fluxes. With increasing model components, model complexity increases as well: not only more input data is required, the set of parameters also increases. This introduces the problem of equifinality, where different parameter sets result in the same model output that are not physically correct, causing prediction uncertainty (Hrachowitz et al., 2013).
Roughly three types of hydrological models can be defined (Figure A.1): (1) a lumped model describes a hydrologic system in a conceptual manner, as a single box with in- and outputs and model properties (states and fluxes) that are representative of the entire system; (2) a distributed model divides the hydrologic system in grid cells, where each cell can have different properties (states and fluxes) and different in- and outputs as well; (3) a semi-distributed model splits a hydrologic system in smaller sub-systems with in- and outputs and model properties that are representative of the sub-system.

Since the water cycle is a global system, the need for hydrological models that span the globe has originated. This has resulted in the development of a number of distributed Global Hydrological Models

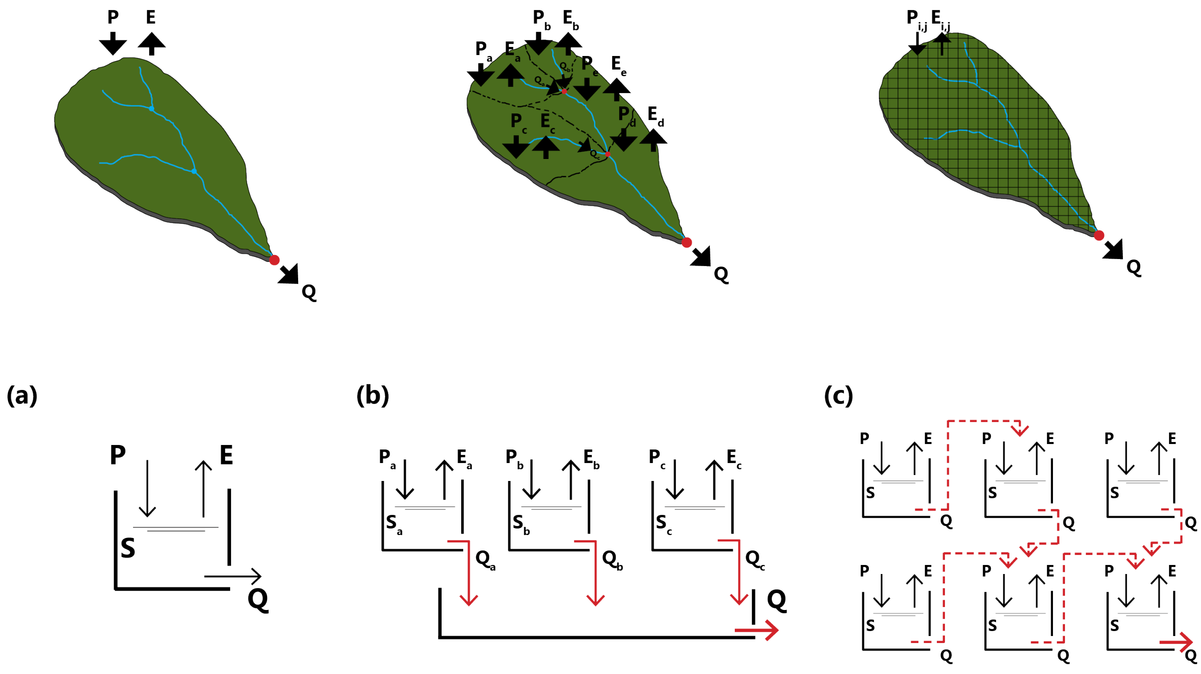


Figure A.1: Schematic overview of the three types of hydrologic models: (a) represents a lumped model that generalizes an entire catchment as one heterogeneous system; (b) depicts a semi-distributed model where the catchment is sub-divided in sub-systems where each sub-system is generalized and (c) displays a distributed model that represents the catchment in a grid, where each grid cell is generalized and connected with upstream and downstream cells.

(GHMs) at spatial resolutions varying from 10 to 100 km, as well as the need to further increase spatial resolutions up to ”hyper-resolution” scale over 1 km (Bierkens et al., 2015, Wood et al., 2011).
Recent advances in computational capabilities, data storage capacity and open availability of high-resolution global datasets have brought the realization of ”hyper-resolution” models closer to reality. Still, challenges remain: when moving to higher spatial resolutions, parameterized sub-scale processes now need to be resolved, introducing additional calculations and extra parameters; the increasing availability of global datasets contain inaccuracies introducing uncertainties; with increasing spatial resolution, calculation and storage needs increase exponentially, so model structures should be adapted for parallel calculations to fully take advantage of new computational capabilities (Bierkens et al., 2015, de Vos et al., 2020, Fatichi et al., 2016).
As this study focuses on matching observations with distributed hydrologic model output, the following sections will go further into the details of distributed hydrologic models.

A.2. Distributed hydrologic models

Distributed hydrologic models are grid-based data structures, and thus built upon multiple spatially distributed data layers. These types of layers vary from gridded meteorological input data to gridded layers describing surface and soil properties for each grid cell. A unique property of distributed models is that all fluxes and states are calculated in every cell, instead of simulating discharge at in single outlet point only.
The output of a distributed model is not only dependent on what happens in each cell, it is also determined by how fluxes are directed between cells. Besides meteorological input, a grid cell will receive runoff or discharge from an upstream cell and direct it to a downstream cell. A common method to determine the direction of flow between cells is to derive a local drainage direction map from a DEM: for each cell in the model the direction of flow is determined by finding the surrounding cell with the lowest elevation.
After derivation of a drainage direction map, the number of upstream cells can be determined for each cell, the flow accumulation. This is a measure of magnitude of flow in the cell and can be used for derivation of a streamflow or river network. For the derivation of a river network, a threshold value of flow accumulation determines which grid cells are part of the network.

With this streamflow network, fluxes between cells can be divided into two phases of flow: (i) the surface flow phase, where the grid cell is not part of the streamflow network; fluxes can consist of surface runoff and subsurface flow, for example, and follow the drainage direction map with low flow velocities compared to when flow has entered the (ii) streamflow phase, where the flow has reached the streamflow network and flow velocities have increased.
Since many distributed hydrologic models work with a spatial resolution varying from 1 to 100 km (Bierkens et al., 2015, Wood et al., 2011), deriving streamflow networks with DEMs at those resolutions result in coarse streamflow networks that may not match with the real world (Graham et al., 1999). To improve the quality of the derived networks and the model's spatial resolution, a pre-existing streamflow network can be burned into the DEM before the drainage direction and streamflow network derivation (Graham et al., 1999, Lehner et al., 2006, 2008, Renssen and Knoop, 2000).
Similarly, a pre-existing streamflow network can be combined with the drainage direction map two simulate fluxes between cells in the two phases mentioned above, as two separate elements: surface flow is directed with the drainage direction map until it reaches the streamflow network, which is then used to determine the direction of streamflow.

A.3. Performance evaluation

To evaluate the model performance, simulations are compared with observations. Hydrographs can visually be compared, or, (preferably a set of) objective functions can be used to express the model performance. Differences between simulations and observations are bound to occur in hydrologic modelling as a result of uncertainties in input and calibration data, model parameters and imperfect model structure that propagate into model output. Therefore, investigation in the source of errors between observations and simulations can lead to improvement of the model and assessing the reliability of the model. Commonly used objective functions are the Nash-Sutcliffe Efficiency (NSE) (Eq. A.1), Kling-Gupta Efficiency (KGE) (Eq. A.2) and Root Mean Squared Error (RMSE) (Eq. A.3) (Gupta et al., 2009, Nash and Sutcliffe, 1970)

$$NSE = 1 - \frac{\sum_{t=0}^{T}\left(Q_{sim}^{t} - Q_{obs}^{t}\right)}{\sum_{t=0}^{T}\left(Q_{obs}^{t} - \overline{Q}_{obs}\right)} \tag{A.1}$$

$$KGE = 1 - \sqrt{(r-1)^2 + \left(\frac{\sigma_{sim}}{\sigma_{obs}} - 1\right)^2 + \left(\frac{\mu_{sim}}{\mu_{obs}} - 1\right)^2} \tag{A.2}$$

with r the correlation between observation and simulation, σ the standard deviation and μ the mean of the streamflow timeseries.

$$RMSE = \sqrt{\sum_{t=1}^{T}\frac{\left(Q_{sim}^{t} - Q_{obs}^{t}\right)^2}{T}} \tag{A.3}$$

Another calibration and evaluation approach is based on similarity of streamflow characteristics, or streamflow signatures. Streamflow signatures are characteristics derived from streamflow time-series. They quantify streamflow properties and can be linked to physical catchment functioning (Sawicz et al., 2011). A reason to consider including streamflow signatures in the calibration and/or evaluation procedure, is that they offer another view to how well simulations approximate observations. They characterize streamflow processes and responses by only looking at specific parts of the hydrograph of time-series. They can tell how well simulations can imitate the observed streamflow behavior in high and low flow regimes, in terms of response to high intensity precipitation events or baseflow contribution to the total streamflow. This also helps in identifying necessary elements in the model structure. In Appendix Appendix C a description of various streamflow signatures can be found.

While comparing simulations and observations, one of the unique properties of the distributed hydrologic model can introduce another source of uncertainty: due to the spatially distributed information in the model, it is essential that observations are compared with simulations representative of the observation location in the model. Otherwise, another structural source of uncertainty is introduced, that cannot be solved by improvement of the input or calibration data, or adaptation of the model structure.

B Theoretical Background: Machine Learning

Since this study aims to combine the disciplines of hydrology and machine learning, this chapter intends to provide a theoretical background for Machine learning.
"Machine learning" (ML) is a term used to describe a range of algorithms that have the ability to learn from experience and do not need to be explicitly programmed beforehand (Mitchell, 1997, Samuel, 1959). In the field of hydrology, ML applications have not only been used as predictors of model outputs - for example of flood forecasts (Dawson and Wilby, 1998), reference evaporation prediction (Shiri et al., 2013), annual runoff forecasting (Wang et al., 2015) and rainfall-runoff forecasting (Kratzert et al., 2018) - but also to estimate model uncertainty (Solomatine and Shrestha, 2009), to classify hystereses (Hamshaw et al., 2018) and to predict runoff signatures (Zhang et al., 2018) among others. In these studies, amongst many more, ML has proven to have the potential to extract patterns from large datasets and predict the response of complex and non-linear processes as occur in the field of hydrology (Shen, 2018).
The following sections give an overview about the types of algorithms and applications, data requirements, training algorithms, performance evaluation and common pitfalls that are often encountered while working in the field of machine learning. Then, a link is made to the field of hydrology.

B.1. Setup

Two types of ML problems exist: (1) supervised learning problems and (2) unsupervised learning problems. ML algorithms learn with a dataset existing of samples and features. Each row consists of a single sample, whose explanatory features (properties, descriptors or attribute) can be found in the columns of the dataset. This table is referenced as X.
When solving a supervised learning problem, this dataset also contains labels or values belonging to each sample, indicated with y. By training the algorithm, it generalizes the relations between X and y. After training, the algorithm should be able to predict $\hat{y}$ when given a new set of samples X.
Unsupervised learning problems work with an unlabelled dataset, only containing features X. Unsupervised algorithms are trained to find structures in X.

B.2. Supervised learning

Supervised learning problems can be split up into classification and regression problems, where a discrete label or continuous value is to be predicted.
To solve a supervised learning problem, a dataset is split into a set of features X and corresponding target values y. A supervised learning algorithm is trained to find relations between X and y, so that when it is given new values of X, it can predict the corresponding values of y, called $\hat{y}$.

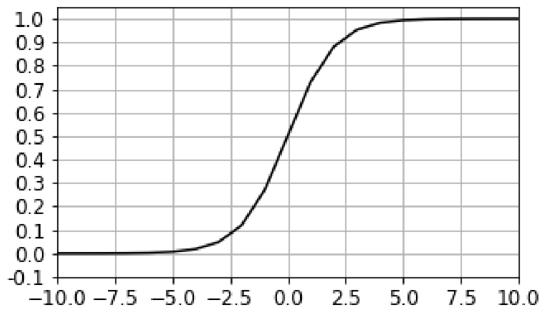


Figure B.1: Example of sigmoid function being bound on the Y-axis between 0 and 1 and having a value of 0.5 at X = 0.

B.2.1. Classification

A classification problem sorts data into predefined classes. The simplest classification problem is a binary classification, that distinguishes between two classes, 0 and 1. Examples of binary classification are the prediction of an event occurrence, or whether a tumor is malign or benign.
For multiple classes, multiple binary classifiers can be combined with a one-versus-all approach. In case of three classes (1,2,3), three algorithms are trained. Algorithm 1 calculates the probabilities of samples belonging to either class 1, or class 2 & 3. Algorithm 2 calculates the probability of a sample belonging to class 2, or class 1 & 3, and so on. A sample will eventually be assigned to the class with the highest probability.
Below, a short overview of common classifiers is discussed:

- Logistic Regression
  Logistic regression is a binary classifier that deploys a *sigmoid* function for predictions. A sigmoid functions is an 'S' shaped function bound on the y-axis between 0 and 1 (Figure B.5):

  $$h(x) = \frac{1}{1+e^{-z}} \tag{B.1}$$

  with z being an activation function in the shape of $\beta_0 + \beta_1 x + .. + \beta_n x$. The output of $h(x)$ can be interpreted as sample having a probability of being 1 or 0, whether $h(x)$ is above or below a certain threshold. This determines the value of predicted label $\hat{y}$:

  $$\hat{y} = \begin{cases} 1, & \text{if } h(x) \geq 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{B.2}$$

  with a threshold equal to 0.5. The weights of coefficients $\beta_0..\beta_n$ are estimated by looking for a best fit between y and $\hat{y}$, usually with the help of maximum likelihood or a cost function. Parameter fitting is discussed in Section B.5;

- K-Nearest Neighbours
  The K-Nearest Neighbours (K-NN) algorithm classifies a sample based on distances to and label values of their *k* nearest neighbouring samples. The class of each sample is determined by a major vote of *k* nearest neighbours; the influence of each vote can be adjusted by weighing the distance between sample and neighbour.

- Decision Trees
  A decision tree is a flowchart-like structure consisting of decision nodes and leaf nodes (see Figure B.2). In each decision node, a partition of samples is made based on a feature value, until a leaf node is reached, which represents a class or label. A decision tree is optimized by finding the best data split in each decision node, based on maximum information gain or split purity. The complexity of a decision tree is determined by the depth of the three, how many decision nodes it contains and the number of features considered. DTs are easy to setup and interpret, but a single tree is prone to *overfitting*, where its classifies well on the data it is trained upon, but less well on new data. Overfitting is further discussed in section B.5.4.

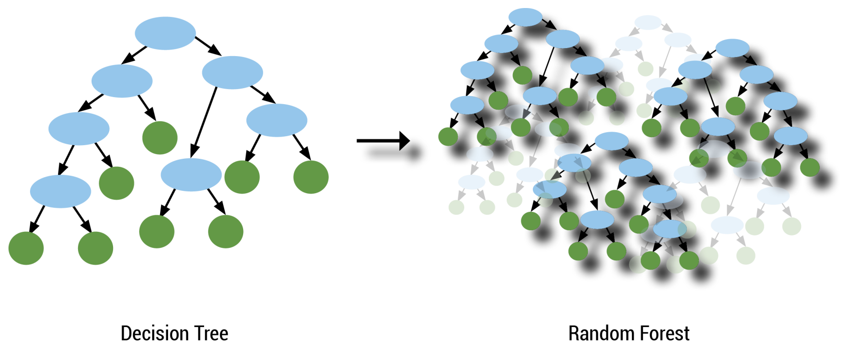


Figure B.2: Visualisations of a Decision Tree and Random Forest. On the left a single Decision Tree is displayed. On the right, a forest of Decision Trees is depicted. The decision nodes are displayed in blue, the leaf nodes, representing the output classes or predictions, are displayed in green. In Random Forests the votes of all the Decision Trees are assembled to produce a robust prediction.

- Random Forests
  Random Forests are an ensemble of Decision Trees, aimed to tackle the problem of overfitting. Instead of following the predictions of a single Decision Tree, multiple trees (e.g. a forest) are trained on the same dataset (see Figure B.2). For the training of each tree, a subset of samples is taken form the dataset, that may include doubles. This is called bootstrapping. Then, a random subset of features is selected to be used in the tree. The predictions of all trees are aggregated to for the classification of the whole forest.
  Due to bootstrapping, a portion of the dataset is not used for training: the ”out-of-bag” set (OOB). This set of samples can be used to evaluate the classification performance of the forest (OOB error). By varying the depth of trees and number of features included, a minimum value for the OOB error can be found to maximize the RF performance.
  Another attempt to improve Decision Trees are the Boosted Trees. Here, a series of Decision Trees are trained and tweaked to improve the predictive abilities. After training a Decision Tree, the correctly classified samples are left and a new Tree is trained using only the mis-classified samples. This results a series of classifiers that are focused on harder to classify features;

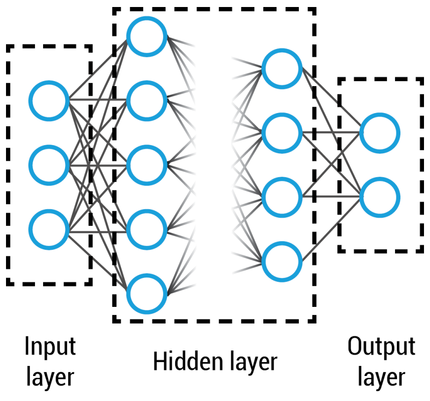


Figure B.3: Visualisation example of a neural network consisting of three layers: an input layer with three neurons, a hidden layer and an output layer with two neurons. The hidden layer can consist of 1 or more layers of neurons. All neurons are connected with activation functions and individual weights

- Neural Network (NN)
  Neural Networks (NN) are Deep learning algorithms that have been designed to imitate neurons in the brain. A NN is built with three elements: an input layer consisting of *n* nodes corresponding to *n* features in *X*; a hidden layer, whose properties depend on the type of neural network; and an output layer which consists of one or more output nodes, depending on the application and number of output classes(see Figure B.3). For a binary classification, the output layer consists of one node, which can have a value of 0 or 1.

Nodes or neurons in the network are connected through simple relations called activation functions. These functions take various forms to be able to capture linear and non-linear relations. By adding more hidden layers, more complex relations can be approximated. With more advanced training techniques developed in recent years, the performance of NNs has strongly increased, nearing human-level performances in fields of natural language translation, image recognition and game play for example.

- Support Vector Machine (SVM)
  SVMs make predictions based on the *Large Margin intuition* resulting in robust classifications. Similar to K-means, an SVM is able to classify samples into clusters. The algorithm aims to find boundaries between classes with a maximum distance between samples of different classes (see Figure B.4). This margin ensures that new samples will be classified with large confidence.

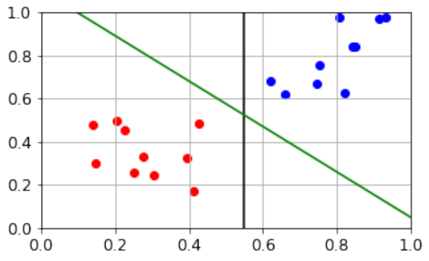


Figure B.4: Example of SVM decision boundary. The black and green line display two different decision boundaries between the red and blue clusters. The green line represents the Large Margin decision boundary fitted by an SVM algorithm: the perpendicular distance between the decision boundary and nearest sample of each cluster is maximized for robust classification.

B.2.2. Regression

A regression problem will try to predict a continuous value based on given features.

- Linear Regression
  A simple example of regression, is linear regression. A linear relation between *X* and *y* exists, which is captured by fitting a straight line:

$$h(x) = \beta_0 + \beta_1 x_1 + ...\beta_n x_n \tag{B.3}$$

  By adjusting the bias β_0 and coefficients β_1 .. β_n, the fit can be adjusted and improved;

- Polynomial Regression
  A form of regression where the relation between *X* and *y* can be described with a polynomial relation:

$$h(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + ...\beta_n x_n^m \tag{B.4}$$

- Non-linear Regression
  A variation of regression where the relation between *X* and *y* cannot be described by a linear function. The relation can instead be described by exponential or logarithmic functions e.g.

B.3. Unsupervised learning

Unsupervised learning problems extract relationships between parameters in and unlabelled dataset *X*, without predicting an output value *y*. Instead, unsupervised learning problems can be broken down into clustering and dimensionality reduction problems: the first one tries to discover clusters in a given dataset; the second one tries to reduce the number of features in a dataset while preserving the variance.

B.3.1. Clustering

Clustering algorithms look for coherent subsets or a cluster structure a given dataset. A widely used algorithm is the K-means algorithm that searches for *k* clusters in the data. The number *k* can be predefined (depending on the purpose of the classification), determined by a visualisation of the data, or estimated with an optimisation algorithm. The k clusters centers are found by minimizing the total distance of all samples to their respective cluster centers:

$$I = \sum_{i=0}^{n} \min_{\mu_j \in K}(|x_i - \mu_j|^2) \tag{B.5}$$

The simplest shape of cluster is circular, but more advanced clustering algorithms can also organize non-circular clusters.

B.3.2. Dimensionality reduction

A dimensionality reduction algorithm decreases the size of the dataset while keeping the data structure intact and preserving a percentage of the total variance. The Principal Component Analysis (PCA) is a widely used algorithm to reduce the number of features or samples of a dataset using Singular Value Decomposition. The algorithm can enhance relations between features, but as the values are transformed onto new axes, any physical meaning of parameter values is lost after the transformation. Dimensionality reduction can be used to save data storage, help visualize data and to speed up machine learning algorithm by reducing the number of features.
The transformation is best calculated based onto *normalized and standardized data*. By calculating a covariance matrix of all features, strongly correlated features can be identified as sources of redundant information. Then, the eigenvectors and -values of the covariance matrix are calculated to identify the Principal Components of the data. Principal Components are variables calculated based on linear transformation of the original dataset, summarizing information from the dataset. From a number of n features, a number of n Principal Components can be calculated, where most information, or explained variance, is put in the first principal component, then the maximum amount of remaining information is put in the second principal component, and so on. When plotting the percentage of variance explained per principal component, it will exponentially decrease, so that the main contributors to the total explained variance can be assigned to the first number of principal components, containing the same information with less features (see Figure B.5).

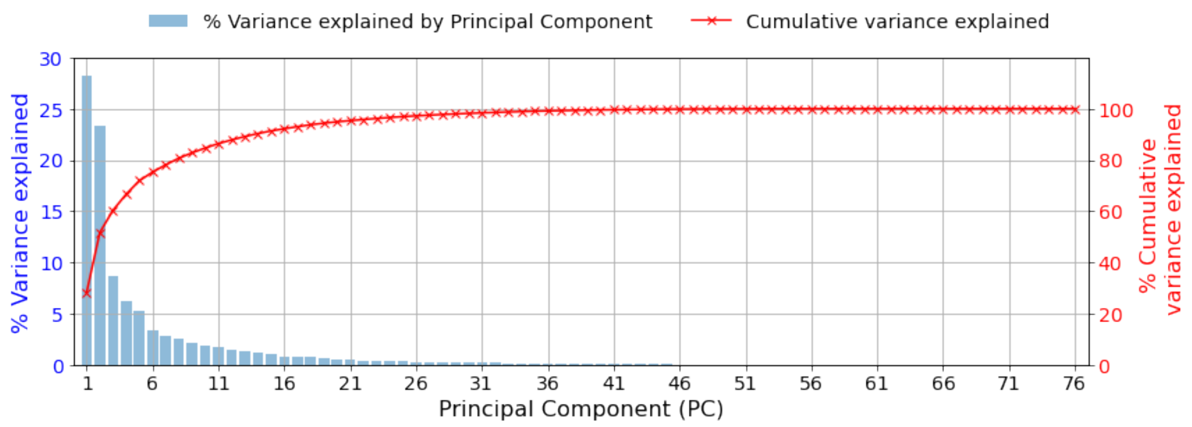


Figure B.5: This skree plot displays the results of an Principal Component Analysis (PCA) transformation. On the X-axis the number of Principal Components (PCs) is displayed. The left Y-axis corresponds to the variance explained by each PC, displayed by a blue bar. The right Y-axis corresponds to the cumulative percentage of variance explained. The first two PCs contribute to over 50% of the variance explained of the original dataset.

Another option for feature reduction can be achieved with recursive feature elimination using Random Forests. After training, weights are assigned to features. By eliminating the features with the smallest weights, having the smallest impact on the prediction, the robustness of the algorithm can be preserved while using a reduced number of features.

B.4. Data pre-processing

While it is important that that an appropriate algorithm and parameters are selected, the quality and amount of data has an equally or even bigger impact on ML algorithm performance (Banko and Brill, 2001). While collecting data and determining the type of features use, a number of things should be kept in mind which will be discussed in this section.

B.4.1. Data exploration

One of the first recommended actions to do when starting to work with a new dataset is exploration: open the file and see what is in there. How is the data formatted? Are there only numerical values in the dataset, or also labelled values? How are missing values indicated? By plotting some features, a feeling for the data distribution can be obtained, or any inconsistencies or outliers in the data can be noticed. After this initial exploration, an approach can be formulated to solve issues that may exist in the dataset.

B.4.2. Missing data

One if these issues may be the occurrence of missing data. A quick and dirty way to get rid of missing data is by removing the rows or column of data containing missing data, with the risk of losing valuable information.

Instead, it can be useful to spend a little more time to try and fill in the missing data. To this end, it can be helpful to try and guess why the data is missing: because it has not been included (failure of measurement instrument), or because it does not exist (measurement is not applicable).

In the first case, if a feature was failed to be included, a value could be copied from the previous or following observation for instance, this is called imputation.

In the second case, if the feature does not exist, it does not make sense to replace the value. The best would be to either keep the missing value or choose another appropriate filler value. While selecting an appropriate filler value, keep in mind that some ML algorithms are able to handle missing values, while others are not.

Dependent on the portion of data missing in a column, a decision should be made on how to deal with missing values. If less than 10% of the column is filled with missing values, it can be more advantageous to impute the missing values than to either drop rows with missing data (reducing the number of samples) or drop the entire column (reducing the number of features). However, if a larger portion is missing, it might be better to either drop the column or rows with missing values instead. Otherwise, too much uncertainty is introduced to the dataset which will limit an algorithm in its training.

B.4.3. Categorized data

As ML algorithms are equipped to work with numerical data, non-numerical categorical features should be converted to numerical. An option is to replace all unique labels with integer: label encoding. The downside is that this can unintentionally introduce a hierarchy in the specific feature. An approach to avoid this could be to add a new column for each unique label and fill the feature column with 0's and 1' in the corresponding rows: one-hot encoding. However, this increases the number of features, which could be unfavorable if a large number of categories exist.

B.4.4. Scaling & normalization

A final step of preprocessing the dataset for training is to adjust the scaling and distribution of values. While exploring the data, a histogram plot of a feature can be a good indicator of the range of data values and distribution.

It is not unusual for various features to have different ranges of data. To give each feature an equal weight for the ML algorithm training, it is recommended to scale all features between values of 0 and 1 for example. This can be achieved by dividing all values by the range of values. This division does not change the shape of the data, but the range of the data only.

Looking at the histogram plot, the distribution of the data becomes apparent as well. Since many ML algorithms assume that the feature data is normally distributed, it is recommended to normalize the data if the distribution appears to be skewed for instance.

B.4.5. Feature selection

While selecting features to train the algorithm with, there are two things to keep in mind: (1) to evaluate the results and make the working of the algorithm understandable, it is recommended to select features with which an expert on the subject would be able to work with as well; and (2) to keep in mind what features will be available in real life when deploying the algorithm in real time. Meaning that, if the features contain foreknowledge, the algorithm may yield good evaluation results but poor results when applied in practice, since some information is not yet available. This is called data leakage and should be avoided. What features are good features will be determined by the algorithm: if the feature is explanatory it will be assigned larger weights during training of the algorithm. Other options to help feature selection is by analysing results of PCA or recursive feature elimination with RF.

B.4.6. Subsampling

For highly imbalanced datasets, the *training dataset only* can be subsampled to equal the percentage of true en false samples. By doing this, the algorithm can equally learn about all classes. To this end, all samples of the minority class are taken from the training dataset, together with an equal number of randomly selected samples from the remaining majority classes. The number of features is reduced but the number of labels are distributed equally over all classes.

B.4.7. Upsampling

Another option to deal with imbalanced datasets is through upsampling. Instead of removing samples belonging to the majority classes, samples from the minority class(es) are duplicated until a balanced dataset is created.

B.5. Algorithm training & evaluation

Training a Machine learning (ML) is an iterative process, since usually multiple tries are needed to find the best parameter set. The goodness-of-fit, or quality of predictions, is quantified using a scoring function, such as a maximum likelihood or cost function, that scores the performance by penalizing correct and incorrect predictions. Based on the score of this function, weights are adjusted with each iteration. Below, training and evaluation is further discussed.
Best performance found with use of maximum likelihood or cost functions, which penalize correct and incorrect predictions in the dataset. Training started with initial values (guesses of parameter values) and with each iteration.
Training, evaluating and testing an algorithm is done using different datasets - the performance evaluation and testing of the algorithm should be done using never seen data, to really check if a trained model has learned about relations in the data, or has learned about specific relation occurring in the training dataset. A common practice is as follows:

- Randomly split data into a training, validation and test set with a ratio from 60-20-20 to 80-10-10 for training-validation-test set.
- Develop a number of models with the following steps:
  - Preprocess training data (scale / normalize / subsample in case of imbalanced data)
  - Optional: perform a feature reduction transformation (PCA)
  - Train an algorithm of choice
  - Pre-process the validation set with identical procedure used for the training data
  - Evaluate the performance of the trained algorithm
  - Adjust parameters of the algorithm until evaluation performance deemed acceptable
- After optimization, select a number of adequate algorithms
- Pre-process the test with the same procedure used for the training data and evaluation data
- Evaluate the model performances based on the never before seen data in the test set

Special care should be taken to separate the test set from the training and evaluation set. While training and evaluating the algorithm, it is likely that the algorithm is set to best fit particularities of those datasets, introducing a bias. For fair evaluation of the model performance, all evaluations of the final performance should be based on the never before seen test set, for which no bias is yet introduced, as it has not been seen by the model. In the following sections, training and evaluation procedures are discussed.

B.5.1. Training

During training, an algorithm tries to generalize relations between X_{train} and y_{train}, so that is can make a prediction $\hat{y}$ for new instances of X. With y_{train} and $\hat{y}_{train}$, the progress of training the algorithm is evaluated with a function that measures the error between y_{train} and $\hat{y}_{train}$. These functions are cost functions $J(\theta)$, with θ corresponding to the weights in the algorithm for each feature in the dataset. Ideally, $J(\theta)$ is a convex function with a single global minimum, but in practice, where θ has a length of $j \geq 2$, local minima occur. Equation B.6 is an example cost function for a binary classifier.

$$(\theta) = -\frac{1}{n}\left[\sum_{i=1}^{n}(y_i log(h(x_i,\theta)) + (1-y_i)log(1-h(x_i,\theta)))\right] \tag{B.6}$$

The aim is to minimize the $J(\theta)$ by finding to global minimum.
This is done by *gradient descent*: using the training dataset and an initial set of weights θ, cost $J(\theta)$ is calculated. With the derivative $\frac{d}{d\theta_j}J(\theta)$ (Equation B.7), all *j* weights are updated, creating a new set of θ (Equation B.8):

$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{m}\sum_{i=1} n(h(x_i,\theta) - y_i)x_i^j \tag{B.7}$$

$$\theta_j = \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta) \tag{B.8}$$

for the next iteration where a new set of θ is used to calculate the updated cost $J(\theta)$. If after a number of iterations the derivative of $\frac{dJ(\theta_n)}{d\theta}$ falls below a threshold level, meaning that the cost remains near constant, it is assumed that a (global) minimum for $J(\theta)$ is found and training is completed. The rate of convergence is influenced by learning rate α in Equation B.8. If α is too small, convergence will take a long time, while a large value of α may cause the cost function to overshoot minima, so no convergence occurs. Different cost functions and solvers, to minimize $J(\theta)$ exist, each with other approaches for finding the global minima and ways to speed up convergence, by having a varying value for α example. Another solving method is *stochastic gradient descent*, which shuffles the training set and only looks at a subset at every iteration, *estimating* or *approximating* the optimum. This can speed up convergence for faster datasets, instead of looking for an *exact* solution.
To further stabilize convergence, a regularization term can be added to $J(\theta)$, with additional benefit of reducing overfitting. The strength of regularization is determined by regularization term λ, also represented as regularization factor C, which equals $1/\lambda$.

An alternative approach for training and evaluation is K-fold Cross Validation. Here, the training and validation set is combined and randomly split in K parts. With K iterations, a different set is held out as validation set. The remaining K-1 sets are used combined into the training set. After K iterations, K algorithms have been trained on all different parts of the training and validation set. This can be useful for training algorithms with small datasets: an ensemble of algorithms is created that have been trained with the whole training and validation set. Additionally, stability of the algorithm can be evaluated.

B.5.2. Evaluation

During training, the error of the model is calculated with cost functions. To assess the predictive quality of the algorithm, a *new* set of features, that was not part of the training set but whose target values are known as well, is given to the algorithm. With this evaluation set, the quality of predictions on never before seen data is assessed. Likely, since during training the algorithm settings are adapted for better training performance, the performance on the set will decrease, but if the algorithm was able to learn

Table B.1: Explanation of a confusion matrix. The number of correctly predicted labels is established on the diagonal, represented by the number of true positives (TP) and true negatives (TN). The incorrectly classified samples can be classified as false positives (FP) or false negatives (FN). With the help of a confusion matrix, the type of error and performance of classifier can be evaluated.

	Predicted: 1	Predicted: 0
Actual: 1	TP	FN
Actual: 0	FP	TN

good generalized equations, the performance will not decrease drastically. Classification performance can be expressed with, for example a confusion matrix (see Table B.1). The values found in this table can belong to four classes:

- True Positives (TP): number of correctly predicted positives
- True Negatives (TN): number of correctly predicted negatives
- False Positives (FP): number of falsely predicted positives
- False Negatives (FN): number of falsely predicted negatives

Whose output can be used to calculate the following metrics:

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
  Evaluates how much of the samples are correctly labelled
- Precision = $\frac{TP}{TP+FP}$
  How many predicted positives are actually positive
- Recall = $\frac{TP}{TP+FN}$
  How many positives are labelled as negatives
- F1-score = $\frac{precision*recall}{precision+recall}$
  Balanced score
- Balanced = $0.5*\left(\frac{TP}{TP+FN}+\frac{TN}{TN+FP}\right)$
  Adapted accuracy metric for imbalanced datasets

B.5.3. Benchmarking

Besides evaluating the algorithm performance with the help of metrics, added value of the algorithms can be assessed by comparing the performance with simple predictive algorithms: benchmarks. When the ML algorithms is able to outperform the benchmark, it can be concluded that the trained ML has truly been able to *learn* generalized relations from the dataset which can also be applied on new data.

B.5.4. Pitfalls

Despite best efforts while conducting the data pre-processing and algorithm training steps described above, the performance of the algorithm can still be below minimal requirements. This can be due to lack of a sufficient number of samples, need for more informative features, non-optimal algorithm settings or an inadequate algorithm structures. Interpretation of the mentioned metrics and the classification matrix, and the occurrence of overfitting or underfitting, give away clues for actions needed to improve the algorithm performance.

B.5.5. Underfitting

Underfitting occurs when the algorithm is not able to capture the complexity of the relations in the data. This manifests in low training and evaluation performances. If even after many iterations, for both the training and validation set, the model bias remains high while the variance is low, this can be an indicator that underfitting occurs. If this is detected, it is recommended to choose a more advanced algorithm as the current algorithm is not able to grasp the complexity of the data.

B.5.6. Overfitting

Overfitting occurs when the algorithm is training too long and starts to develop training set specific relations instead of generalized relations for the entire dataset. This becomes evident when the algorithm performance for the training set keeps increasing while the evaluation performance decreases. The bias for the training set is low, while the variance is high. By calculating and tracking the training and evaluation cost at each learning iteration and stopping when the learning once the training cost is stable, or once the evaluation cost starts increasing, it can be prevented that the algorithm overfits on the training dataset. This is called early stopping.

C
Streamflow signatures

In this study, streamflow signatures are used to compare properties of streamflow observations with streamflow simulations. The signatures used in this study, see Table 2.1, have been chosen to capture a range of flow characteristics are illustrated below.

C.1. Statistical distribution

The first group of signatures can be described as descriptors of the distribution of streamflow. The parameters and the goodness-of-fit of the Normal, Log-Normal, Gumbel, Poisson and Gamma distributions are calculated based on streamflow simulations and streamflow observations and used as features. The goodness-of-fit is calculated with the Kolmogorov-Smirnov test (Kolmogorov, 1933, Marsaglia et al., 2003, Virtanen et al., 2020).

C.2. Correlation

The second group of features are signatures related to cross correlation. The first one is the Pearson's 1-lag auto-correlation. Here, the correlation of a streamflow timeseries is calculated with the same timeseries, with a lag of one timestep. In this study, this means that the streamflow of 1 day is compared with the streamflow of the day before (Euser et al., 2013). The 1-lag auto-correlation signature is a measure for the smoothness of the hydrograph and response time.
With the cross-correlation signature, the similarity between a streamflow observation and streamflow simulation is calculated (Bennett et al., 2013). The cross-correlation feature is calculated with a lag of 0 and 1.

C.3. Flow duration curve

A Flow Duration Curve (FDC) depicts the relation between magnitude of streamflow and the frequency of occurrence (Searcy, 1959, Vogel and Fennessey, 1994). To construct an FDC, all streamflow values in a timeseries are sorted and ranked. The probability of exceedance can then be calculated by dividing the rank by the length of the timeseries $+1$.
The streamflow values corresponding to several percentiles ([1, 2, 5, 10, 50, 90, 95, 99]) are calculated as features. Also, the slope and low flow ratio are derived from the FDC (Nijzink et al., 2018, Sawicz et al., 2011, Smakhtin, 2001, Yadav et al., 2007).
A secondary FDC is calculated based on peaks in the streamflow timeseries only. Peaks are defined as a point in the hydrograph that is higher than the previous and next point. From this FDC, the slope is calculated to find the peak distribution (Euser et al., 2013). The slope of the FDC functions as a measure for the variability of flow response or variability of peaks.

C.4. Hydrological indices

The final group of features consists of streamflow signatures describing various aspects of streamflow. The baseflow index is the ratio of baseflow to total flow. A high value of the baseflow index indicates

that baseflow is a large contributor to the total streamflow. This implies that long flowpaths are used to reach the streamflow network (Sawicz et al., 2011).
The Rising Limb Density is the ratio of total number of peaks to the total duration of rising limbs and functions as a measure for the smoothness of the hydrograph (Morin et al., 2002). Similarly, the Declining Limb Density is the ratio of the total number of peaks versus the duration of declining limbs (Shamir et al., 2005).
Another measure for the smoothness of the hydrograph is through the Richard-Baker Flashiness, as the sum of absolute values of day-to-day changes divided by the sum of all daily flows.
Recession curve characteristics can be used to quantify the storage-outflow relationship of a catchment (Stoelzle et al., 2013, Vogel and Kroll, 1992).
The duration and frequency of high and low flow events provide insights in the flow regimes observed in the streamflow timeseries.
Finally, the elevation and upstream area are location descriptors. The upstream area is a commonly used metric to accept or reject the location of a placed gauge. It also provides information about the streamflow magnitude (Toth, 2013).

D
Results

D.1. LR - Window format

D.1.1. Classification

3 by 3 search window

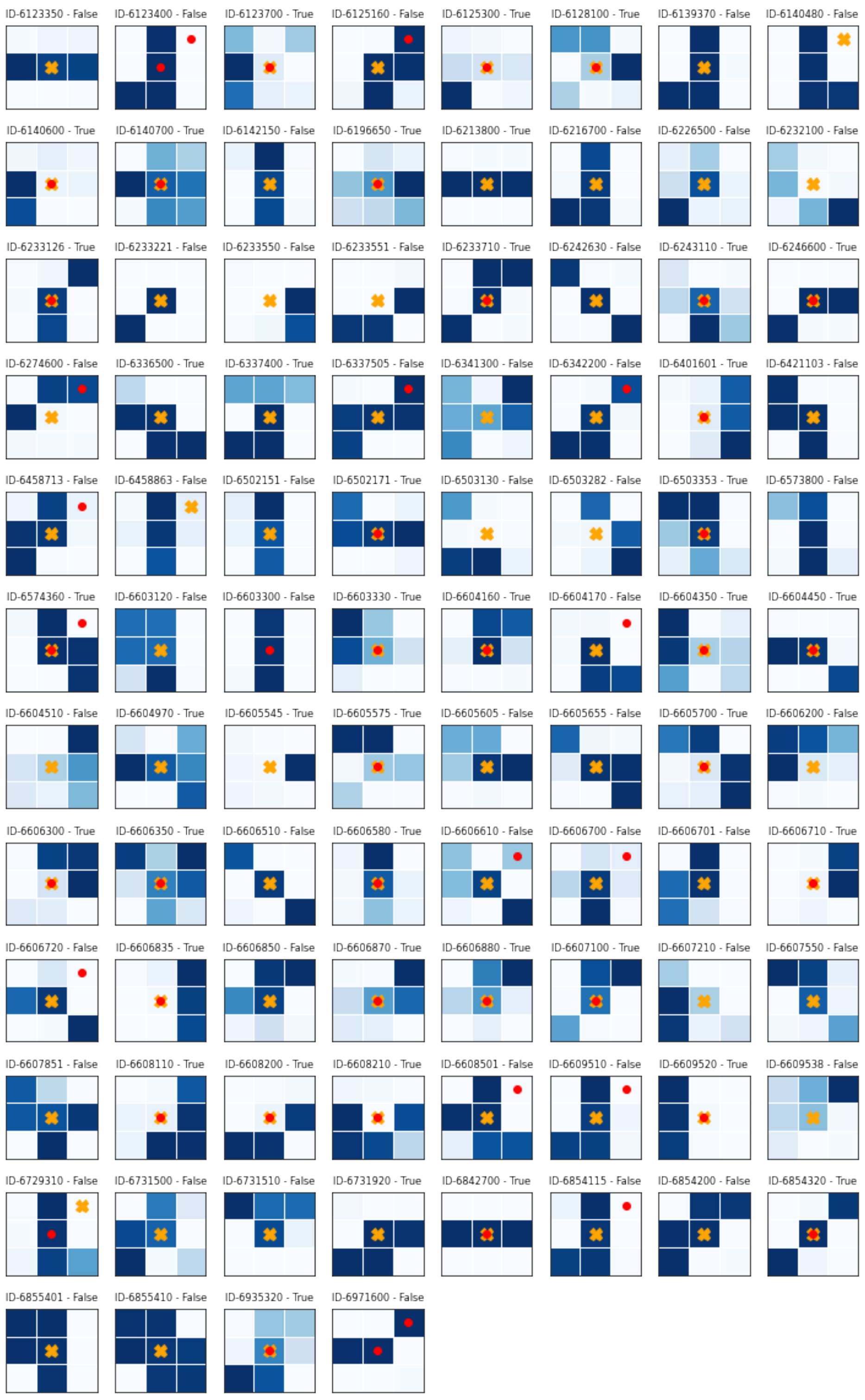


Figure D.1: This figure displays the classification of the test set with the LR algorithm with Window format. Each square is displays a 3 by 3 search window around a station whose ID is shown at the top. Each grid cell in the search window is displayed with a shade of blue that is related to the average streamflow: darker shades of blue indicate a higher level of average streamflow. The matches from the dataset provided by ECMWF are displayed in orange. The matches predicted by the algorithm are depicted in red.

9 by 9 search window

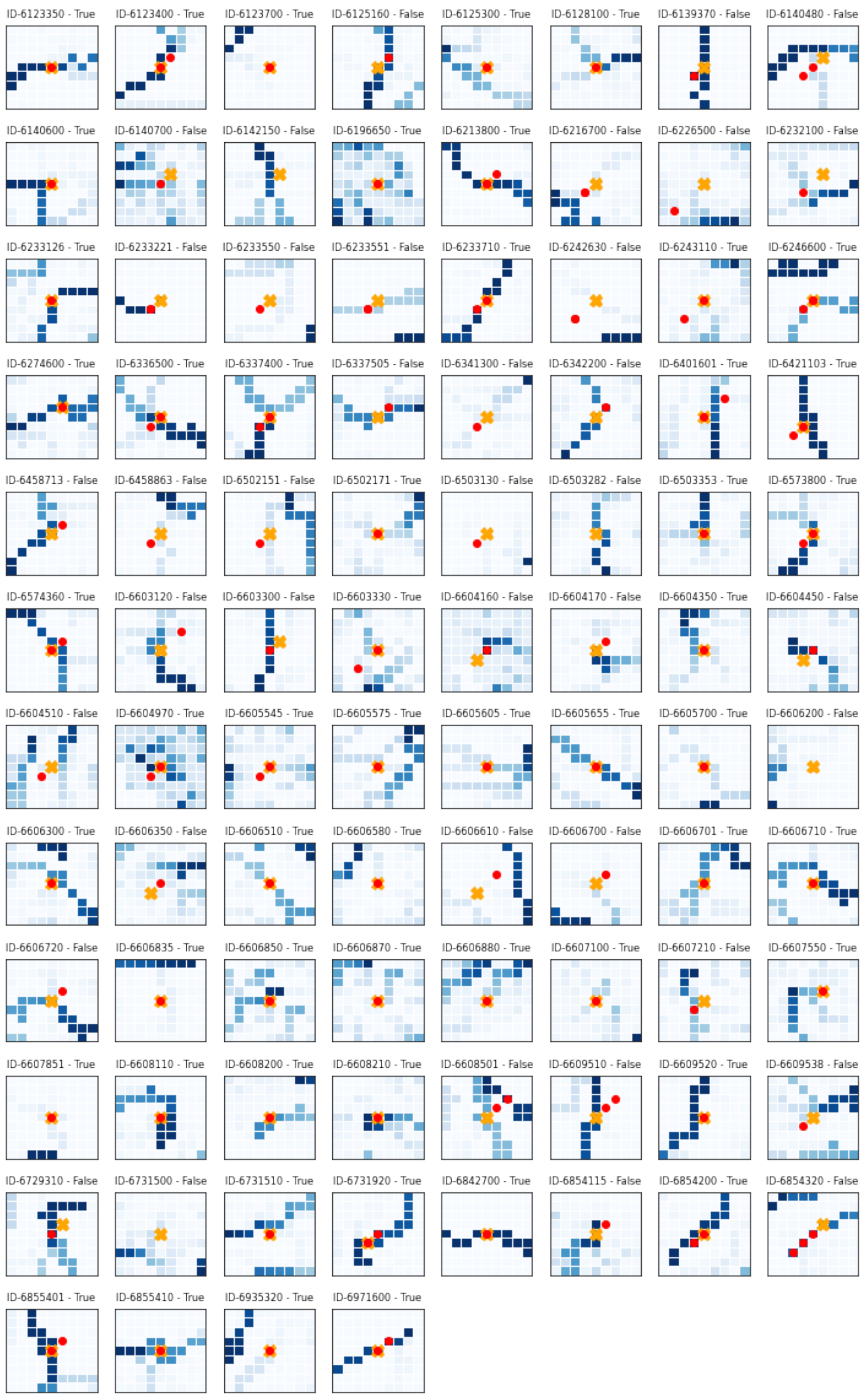


Figure D.2: This figure displays the classification of the test set with the LR algorithm with Window format. Each square is displays a 9 by 9 search window around a station whose ID is shown at the top. Each grid cell in the search window is displayed with a shade of blue that is related to the average streamflow: darker shades of blue indicate a higher level of average streamflow. The matches from the dataset provided by ECMWF are displayed in orange. The matches predicted by the algorithm are depicted in red.

D.1.2. Probability

3 by 3 search window

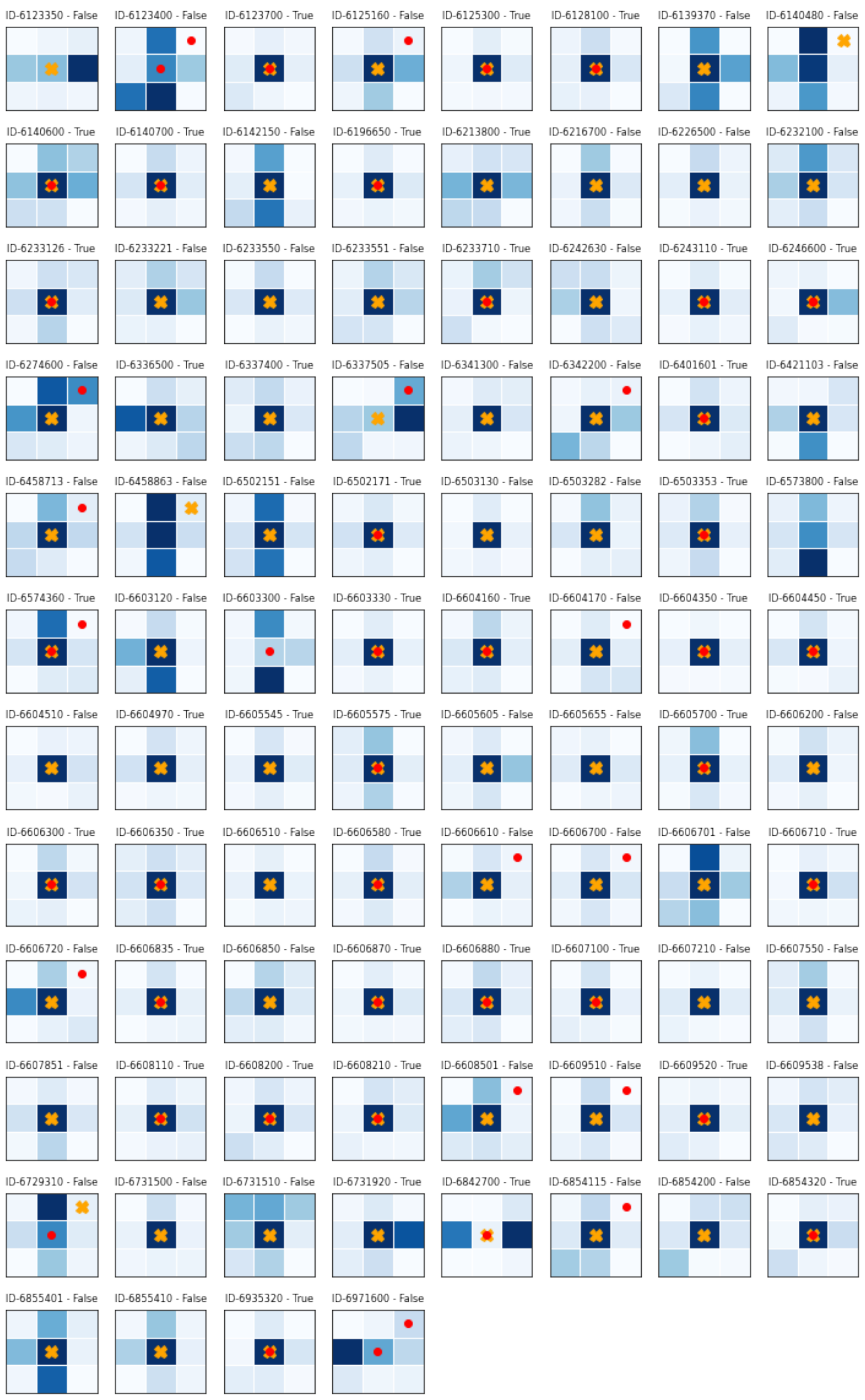


Figure D.3: This figure displays the probabilities levels that determine the classification of the test set with the LR algorithm with Window format. Each square is displays a 3 by 3 search window around a station whose ID is shown at the top. Each grid cell in the search window is displayed with a shade of blue that is related to the probability of each cell being representative of the observation. Darker shades of blue indicate a higher probability. The matches from the dataset provided by ECMWF are displayed in orange. The matches predicted by the algorithm are depicted in red.

9 by 9 search window

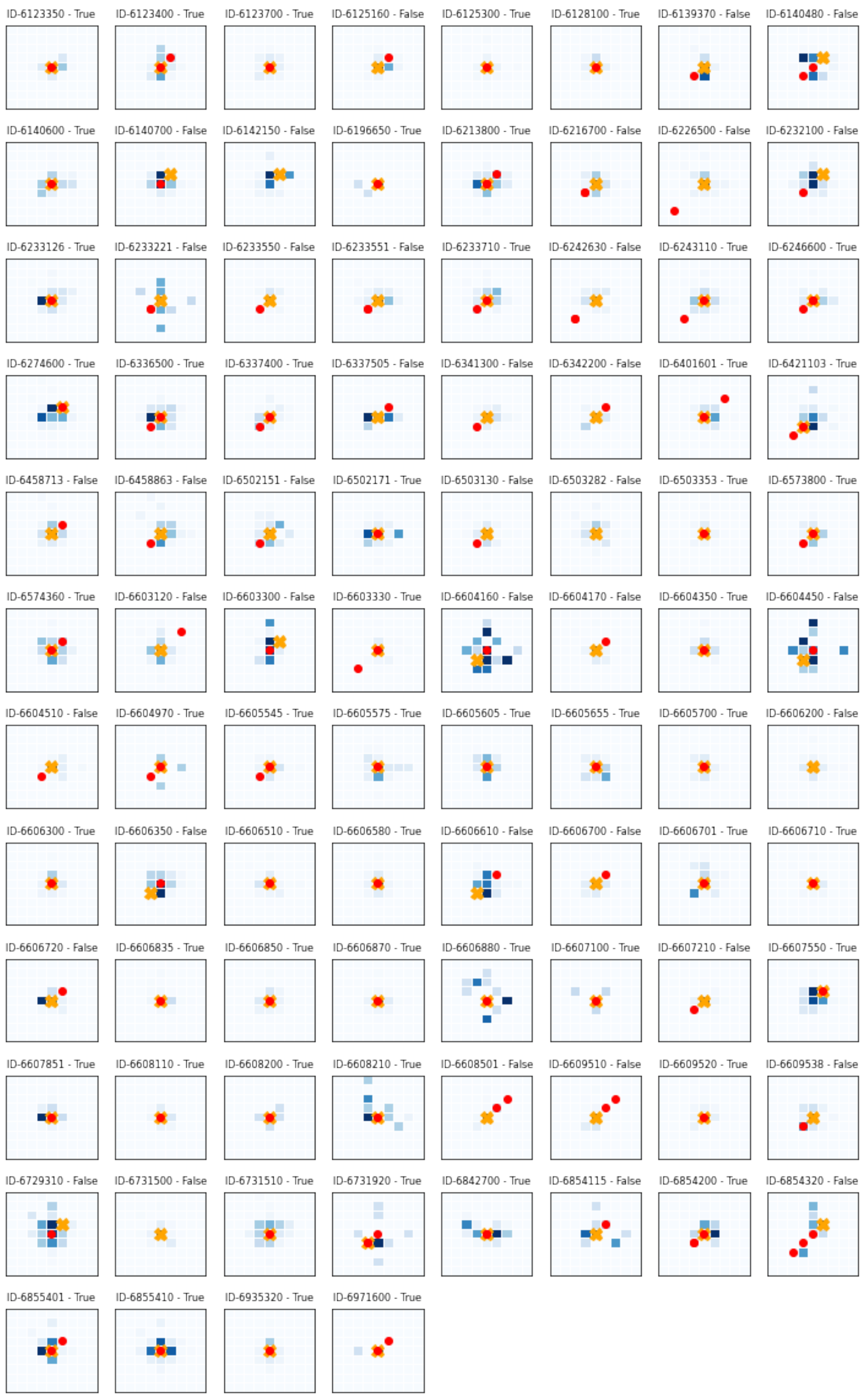


Figure D.4: This figure displays the probabilities levels that determine the classification of the test set with the LR algorithm with Window format. Each square is displays a 9 by 9 search window around a station whose ID is shown at the top. Each grid cell in the search window is displayed with a shade of blue that is related to the probability of each cell being representative of the observation. Darker shades of blue indicate a higher probability. The matches from the dataset provided by ECMWF are displayed in orange. The matches predicted by the algorithm are depicted in red.

D.2. RF - Single Cell format

D.2.1. Classification

3 by 3 search window

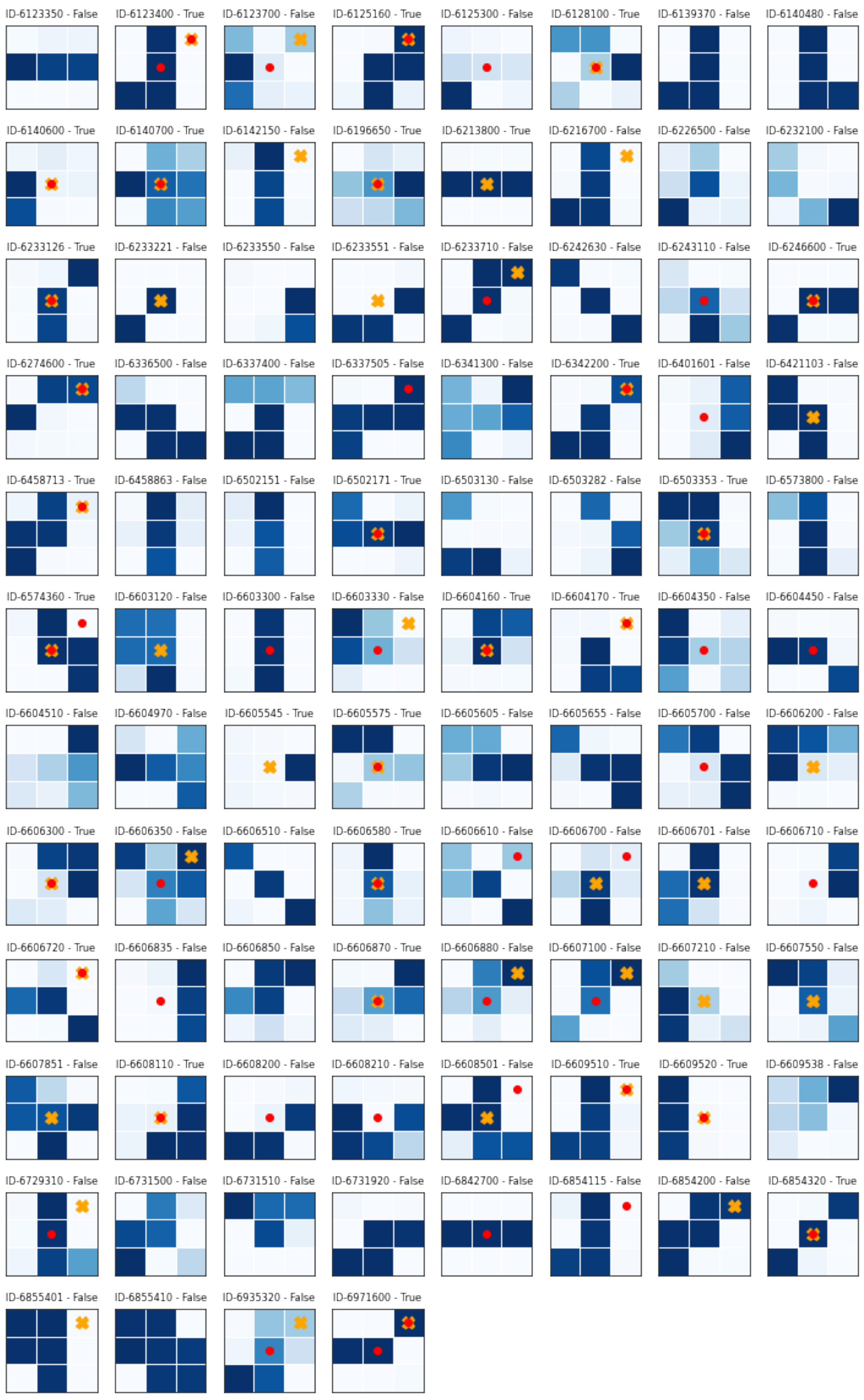


Figure D.5: This figure displays the classification of the test set with the RF algorithm with Single Cell format. Each square is displays a 3 by 3 search window around a station whose ID is shown at the top. Each grid cell in the search window is displayed with a shade of blue that is related to the average streamflow: darker shades of blue indicate a higher level of average streamflow. The matches from the dataset provided by ECMWF are displayed in orange. The matches predicted by the algorithm are depicted in red.

9 by 9 search window

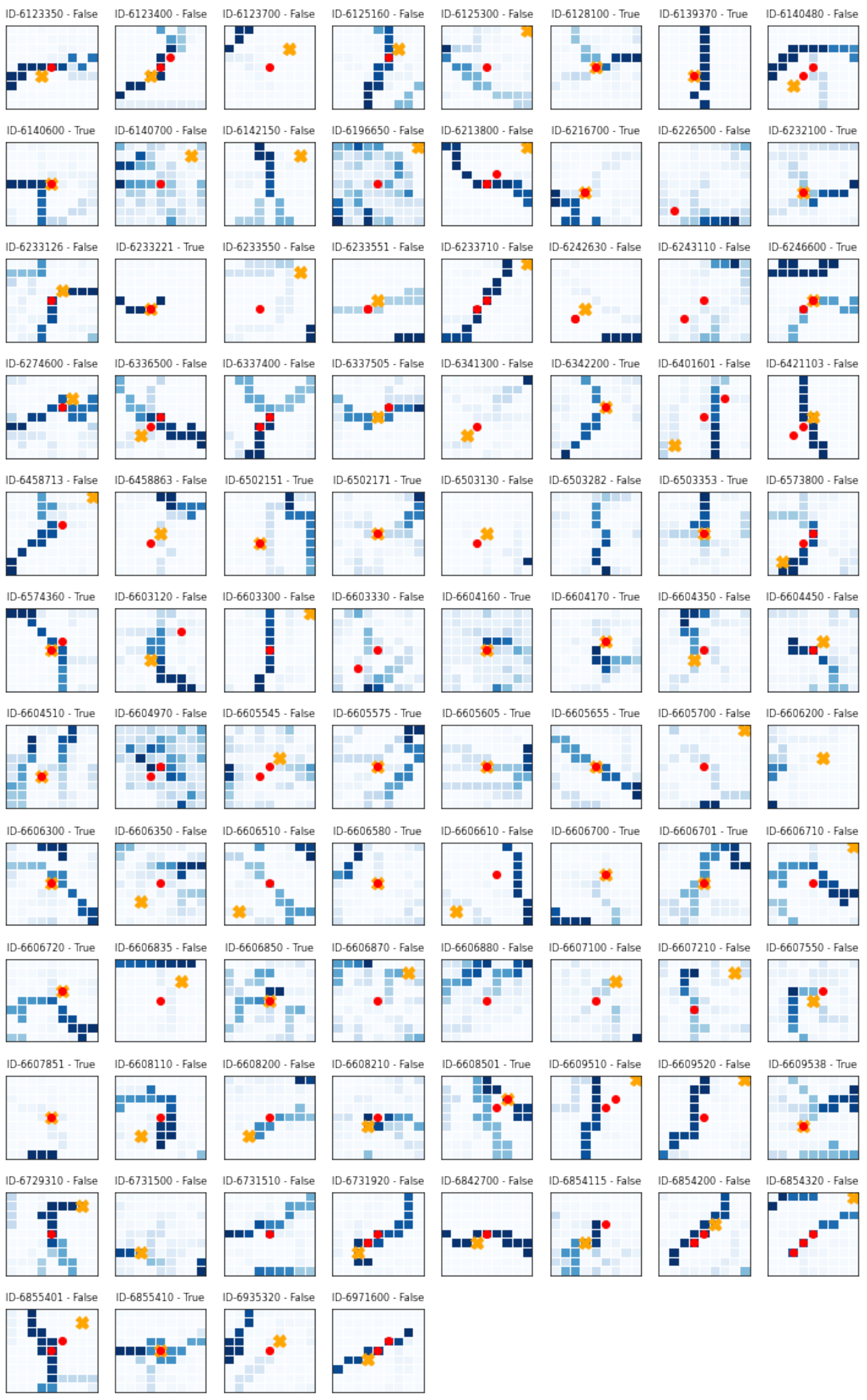


Figure D.6: This figure displays the classification of the test set with the RF algorithm with Single Cell format. Each square is displays a 9 by 9 search window around a station whose ID is shown at the top. Each grid cell in the search window is displayed with a shade of blue that is related to the average streamflow: darker shades of blue indicate a higher level of average streamflow. The matches from the dataset provided by ECMWF are displayed in orange. The matches predicted by the algorithm are depicted in red.

D.2.2. Probability

3 by 3 search window

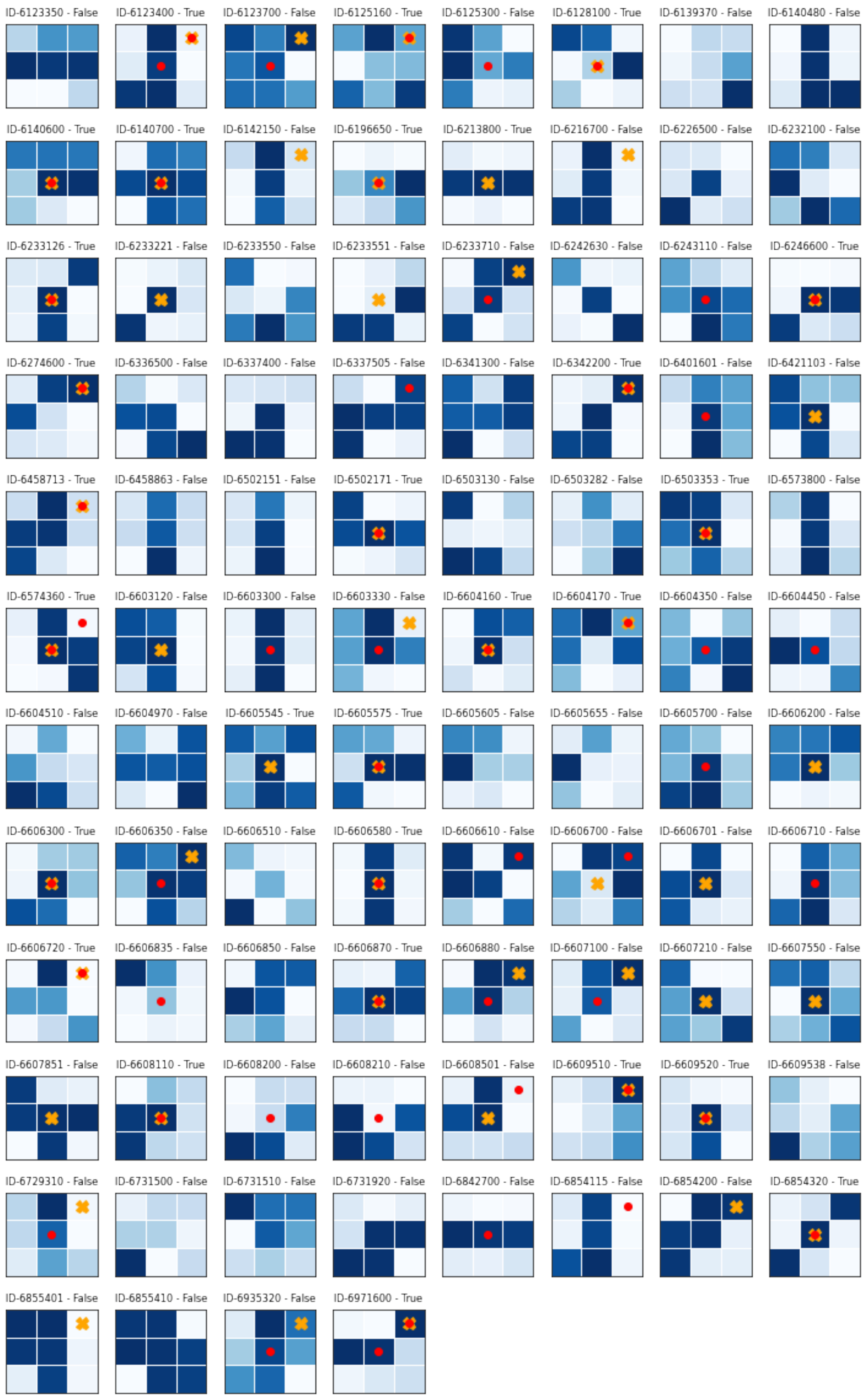


Figure D.7: This figure displays the probabilities levels that determine the classification of the test set with the RF algorithm with Single Cell format. Each square is displays a 3 by 3 search window around a station whose ID is shown at the top. Each grid cell in the search window is displayed with a shade of blue that is related to the probability of each cell being representative of the observation. Darker shades of blue indicate a higher probability. The matches from the dataset provided by ECMWF are displayed in orange. The matches predicted by the algorithm are depicted in red.

9 by 9 search window

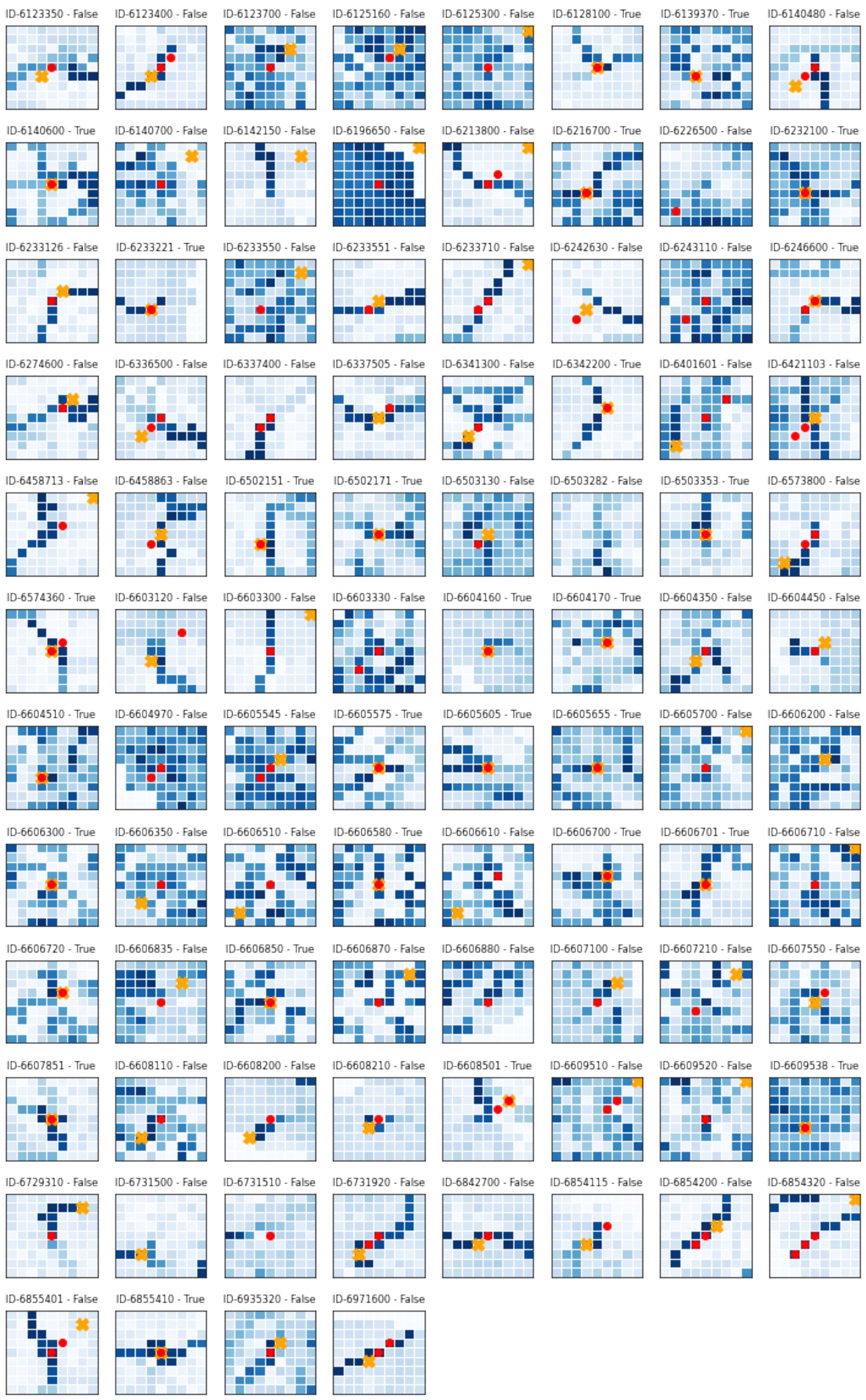


Figure D.8: This figure displays the probabilities levels that determine the classification of the test set with the RF algorithm with Single Cell format. Each square is displays a 9 by 9 search window around a station whose ID is shown at the top. Each grid cell in the search window is displayed with a shade of blue that is related to the probability of each cell being representative of the observation. Darker shades of blue indicate a higher probability. The matches from the dataset provided by ECMWF are displayed in orange. The matches predicted by the algorithm are depicted in red.