FLOOD MONITORING AND EARLY WARNING SYSTEM
BATCH MEMBER
911721104068:NASREEN BANU.S

FLOOD MONITORING:

Flood monitoring involves the systematic observation, collection, analysis, and interpretation of data related to water levels, rainfall, weather conditions, and other factors that can indicate or contribute to flooding. Here's a detailed explanation of flood monitoring.

1.INTRODUCTION & DETAILS,DIAGRAM,OBJECTIVES,PROGRAM,ADVANTAGES,BENEFITS ARE ALL INCLUDE:

Flooding is the most common natural hazard and results worldwide in the most damaging disasters [1,2,3,4]. Recent studies associate the increasing frequency and severity of flood events with a change in land use (e.g., deforestation and urbanization) and climate [2,5,6,7]. This particularly holds for the tropical Andes region, where complex hydro-meteorological conditions result in the occurrence of intense and patchy rainfall events [8,9,10].

According to the flood generation mechanism, floods can be classified into long- and short-rain floods [11,12]. A key for building resilience to short-rain floods is to anticipate in a timely way the event, in order to gain time for better preparedness. The response time between a rainfall event and its associated flood depends on the catchment properties and might vary from minutes to hours [13]. In this study special attention is given to flash-floods, which are floods that develop less than 6 h after a heavy rainfall with little or no forecast lead time [14].

Flood anticipation can be achieved through the development of a flood early warning system (FEWS). FEWSs have proved to be cost-efficient solutions for life preservation, damage mitigation, and resilience enhancement [15,16,17,18]. However, although crucial, flood forecasting remains a major challenge in mountainous regions due to the difficulty to effectively record the aerial distribution of precipitation due to the sparse density of the monitoring network and the absence of high-tech equipment by budget constraints [8,9].

To date, there has been no report of any operational FEWS in the Andean region for scales other than continental [17,19,20]. An alternative attempt in Peru aimed to derive daily maps of potential floods based on the spatial cumulated precipitation in past days [21]. Other endeavors in Ecuador and Bolivia focused on the monitoring of the runoff in the upper parts of the catchment to predict the likelihood of flood events in the downstream basin area [19,22]. However, such attempts are unsatisfactory as countermeasures against floods and especially flash-floods, where it is required to have reliable and accurate forecasts with lead times shorter than the response time between the farthest precipitation station and runoff control point.

There are two paradigms that drive the modeling of the precipitation-runoff response. First, the physically-based paradigm includes knowledge of the physical processes by using physical process

equations [23]. This approach requires extensive ground data and, in consequence, intensive computation that hinders the temporal forecast window [24]. Moreover, it is argued that physically based models are inappropriate for real-time or short-term flood forecasting due to the inherent uncertainty of river-catchment dynamics and over-parametrization of this type of model [25]. The second data-driven paradigm assumes floods as stochastic processes with an occurrence distribution probability derived from historical data. Here, the idea is to exploit relevant input information (e.g., precipitation, past runoff) to find relations to the target variable (i.e., runoff) without requiring knowledge about the underlying physical processes. Among the traditional data-driven approaches, statistical modeling has proven to be unsuitable for short-term prediction due to lack of accuracy, complexity, model robustness, and even computational costs [24]. Previous encouraged the use of advanced data-driven models, e.g., machine learning (ML), to overcome the aforementioned shortcomings [7,24,26,27]. Particularly during the last decade, ML approaches have gained increasing popularity among hydrologists [24].

Different ML strategies for flood forecasting are implemented, generating either quantitative or qualitative runoff forecasts [18,28,29,30,31,32,33,34,35,36,37,38]. Qualitative forecasting consists of classifying floods into distinct categories or river states according to their severity (i.e., runoff magnitude), and use this as a base for flood class prediction [30,37,39]. The advantage of developing a FEWS is the possibility to generate a semaphore-like warning system that is easy to understand by decision-makers and the public (non-hydrologists). The challenge of FEWSs is the selection of the most optimal ML technique to obtain reliable and accurate forecasts with sufficient lead time for decision making. To date, the problem has received scant attention in the research literature, and as far as our knowledge extends no previous work examined and compared the potential and efficacy of different ML techniques for flood forecasting.

The present study compares the performance of five ML classification techniques for short-rain flood forecasting with special attention to flash floods. ML models were developed for a medium-size mountain catchment, the Tomebamba basin located in the tropical Andes of Ecuador. The ML models were tested with respect to their capacity to forecast three flood warning stages *(No-alert*, *Pre-alert* and *Alert*) for varying forecast lead times of 1, 4, and 6 h (flash-floods), but also 8 and 12 h to further test whether the lead time can be satisfactorily extended without losing the models' operational value.

This paper has been organized into four sections. The first section establishes the methodological framework for developing a FEWSs using ML techniques. It will then go on to describe the performance metrics used for a proper efficiency assessment. The second section presents the findings of the research following the same structure as the methodological section. Finally, the third and fourth sections presents the discussion and a summary of the main conclusions of the study, respectively.

2. MATERIALS AND METHODS:

2.1 Study Area and Dataset:

The study area comprises the Tomebamba catchment delineated upstream of the Matadero-Sayausí hydrological station of the Tomebamba river (**Figure 1**), where the river enters the city. The Tomebamba is a tropical mountain catchment located in the southeastern flank of the Western Andean Cordillera, draining to the Amazon River. The drainage area of the catchment is approximately 300 km^2,

spanning from 2800 to 4100 m above sea level (m a.s.l.). Like many other mountain catchments of the region, it is primarily covered by a páramo ecosystem, which is known for its important water regulation function [8].

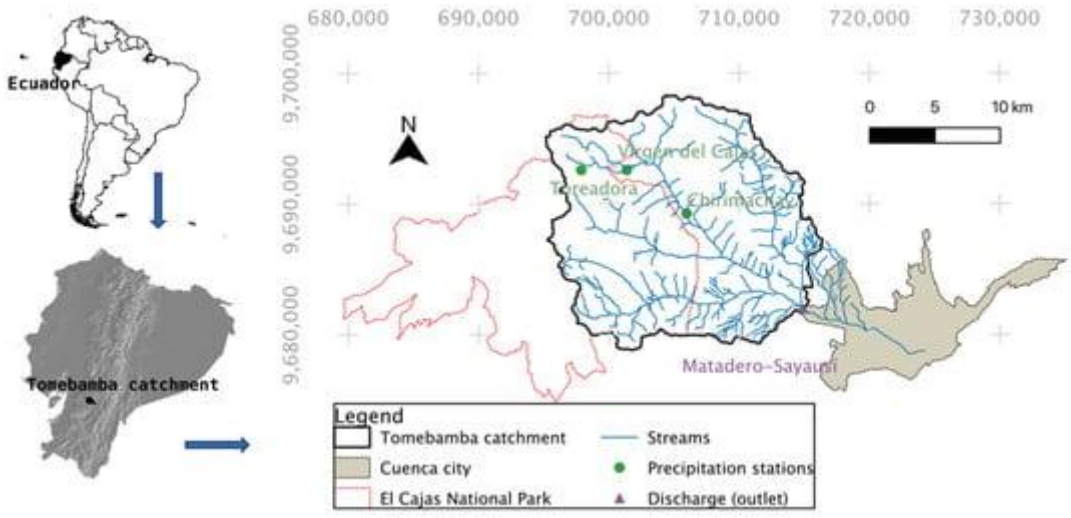


Figure 1. The Tomebamba catchment located at the Tropical Andean Cordillera of Ecuador, South America (UTM coordinates).

The Tomebamba river plays a crucial role as a drinking water source for the city of Cuenca (between 25% to 30% of the demand). Other important water users are agricultural and industrial activities. Cuenca, which is the third-largest city of Ecuador (around 0.6 million inhabitants), is crossed by four rivers that annually flood parts of the city, causing human and significant economic losses.

The local water utility, the Municipal Public Company of Telecommunications, Water, Sewerage and Sanitation of Cuenca (ETAPA-EP), defined three flood alert levels associated with the Matadero-Sayausí station for floods originating in the Tomebamba catchment: (i) *No-alert* of flood occurs when the measured runoff is less than 30 m^3/s, (ii) *Pre-alert* when runoff is between 30 and 50 m^3/s, and (iii) the flood *Alert* is triggered when discharge exceeds 50 m^3/s. With these definitions, and as shown in **Figure 2**, the discharge label for the *No-alert* class represents the majority of the data, whereas the *Pre-alert* and *Alert* classes comprise the minority yet the most dangerous classes.

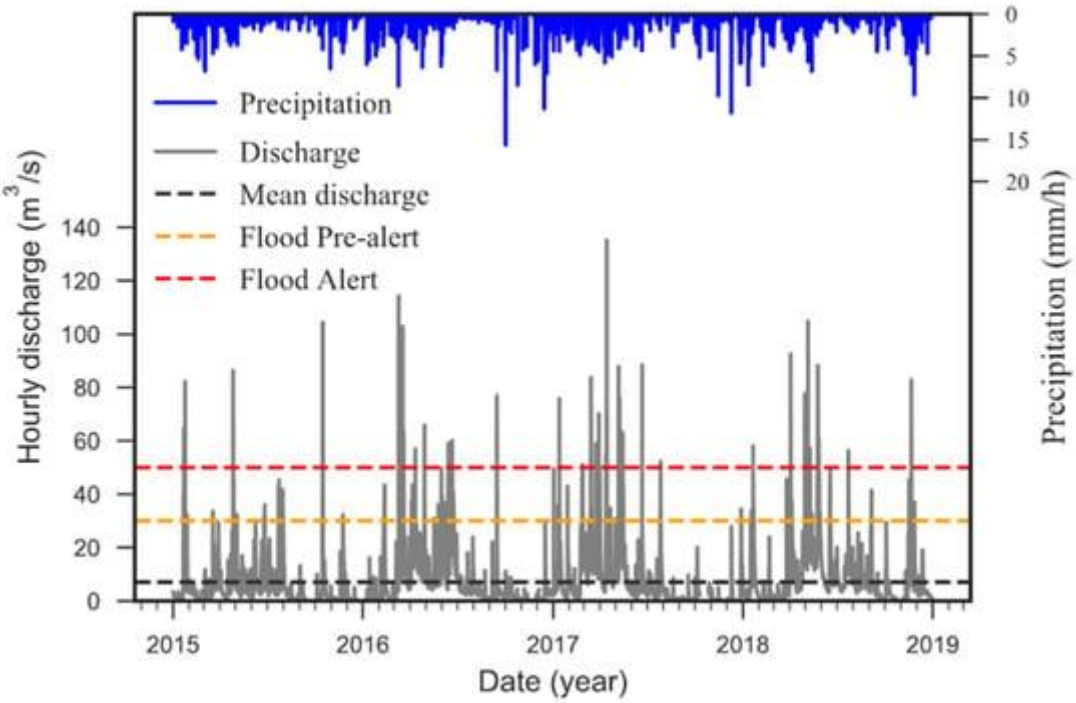


Figure 2. Time series of precipitation (Toreadora) and discharge (Matadero-Sayausí). Horizontal dashed lines indicate the mean runoff and the currently employed flood alert levels for labeling the *Pre-alert* and *Alert* flood warnings classes.

To develop and operate forecasting models, we use data of two variables: precipitation in the catchment area and river discharge at a river gauge. For both variables, the available dataset comprises 4 years of concurrent hourly time series, from Jan/2015 to Jan/2019 (Figure 2). Precipitation information was derived from three tipping-bucket rain gauges: Toreadora (3955 m a.s.l.), Virgen (3626 m a.s.l.), and Chirimachay (3298 m a.s.l.) installed within the catchment and along its altitudinal gradient. Whereas for discharge, we used data of the Matadero-Sayausí station (2693 m a.s.l., Figure 1). To develop the ML modes, we split the dataset into training and test subsets. The training period ran from 2015 to 2017, whereas 2018 was used as the model testing phase.

2.2. Machine Learning (ML) Methods for Classification of Flood Alert Levels

ML classification algorithms can be grouped in terms of their functionality. According to Mosavi et al. (2018), five of the worldwide most-popular statistical method groups are commonly used for short-term flood prediction (extreme runoff), and include:

i.Regression algorithms modeling the relationships between variables (e.g., logistic regression, linear regression, multivariate adaptive regression splines, etc.) [18,40].

ii.Instance-based algorithms that rely on memory-based learning, representing a decision problem fed with data for training (e.g., K-nearest neighbor, learning vector quantification, locally weighted learning, etc.) [30].

iii.Decision tree algorithms, which progressively divide the whole data set into subsets based on certain feature values, until all target variables are grouped into one category (e.g., classification and regression tree, M5, random forest, etc.) [18,28,30,31,37].

iv.Bayesian algorithms based on Bayes' theorem on conditional probability (e.g., naive Bayes, Bayesian network, Gaussian naïve Bayes, etc.) [18,31,35].

v.Neural Network algorithms inspired by biological neural networks convert input(s) to output(s) throughspecified transient states that enable the model to learn in a sophisticated way (e.g., perceptron, multi layer perceptron, radial basis function network, etc.) [18,31,36].

For this study, we selected five ML algorithms, one from each group, respectively, a logistic regression, K-nearest neighbor, random forest, naive Bayes, and a multi-layer perceptron.

2.2.1 Logistic Regression:

Logistic Regression (LR) is a discriminative model, modeling the decision boundary between classes. In a first instance, linear regressions are applied to find existent relationships between model features. Thereafter, the probability (conditional) of belonging to a class is identified using a logistic (sigmoid) function that effectively deals with outliers (binary classification). From these probabilities, the LR classifies, with regularization, the dependent variables into any of the created classes. However, for multiclass classification problems are all binary classification possibilities considered, it is *No-alert* vs. *Pre-alert*, *No-alert* vs. *Alert*, and *Pre-alert* vs. *Alert*. Finally, the solution is the classification with the maximum probability (multinomial LR) using the *softmax* function Equation (1). With this function is the predicted probability of each class defined [41]. The calculated probability for each class is positive with the logistic function and normalized across all classes.

$$softmax(z)i=ezi\sum kl=1ezl()=\sum=1$$

(1)

where z_i is the *i*th input of the *softmax* function, corresponding to class i from the k number of classes.

2.2.2.K-Nearest Neighbors:

K-Nearest Neighbors (KNN) is a non-parametric statistical pattern recognition algorithm, for which no theoretical or analytical background exist but an intuitive statistical procedure (memory-based learning) for the classification. KNN classifies unseen data based on a similarity measure such as a distance function (e.g., Euclidean, Manhattan, Chebyshev, Hamming, etc.). The use of multiple neighbors instead of only one is recommended to avoid the wrong delineation of class boundaries caused by noisy features. In the end, the majority vote of the nearest neighbors (see the formulation in [41]) determines the classification decision. The number of nearest neighbors can be optimized to reach a global minimum avoiding longer computation times, and the influence of class size. The major advantage of the KNN is its simplicity. However, the drawback is that KNN is memory intensive, all training data must be stored and compared when added information is to be evaluated.

2.2.3 Random Forest:

Random Forest (RF) is a supervised ML algorithm that ensembles a multitude of decorrelated decision trees (DTs) voting for the most popular class (classification). In practice, a DT (particular model) is a hierarchical analysis based on a set of conditions consecutively applied to a dataset. To assure decorrelation, the RF algorithm applies a bagging technique for a growing DT from different randomly resampled training subsets obtained from the original dataset. Each DT provides an independent output (class) of the phenomenon of interest (i.e., runoff), contrary to numerical labels for regression applications. The popularity of RF is due to the possibility to perform random subsampling and bootstrapping which minimizes biased classification [42]. An extended description of the RF functioning is available in [43,44].

The predicted class probabilities of an input sample are calculated as the mean predicted class probabilities of the trees in the forest. For a single tree, the class probability is computed as the fraction of samples of the same class in a leaf. However, it is well-known that the calculated training frequencies are not accurate conditional probability estimates due to the high bias and variance of the frequencies [45]. This deficiency can be resolved by controlling the minimum number of samples required at a leaf node, with the objective to induce a smoothing effect, and to obtain statistically reliable probability estimates.

2.2.4.Naive Bayes:

Naïve Bayes (NB) is a classification method based on Bayes' theorem with the "naive" assumption that there is no dependence between features in a class, even if there is dependence [46]. Bayes' theorem can be expressed as:

$$P(y|X)=P(X|y)\ P(y)P(X)\ (|)=(|)\ ()()$$

(2)

where $P(A|B)$(|) is the probability of y (hypothesis) happening, given the occurrence of X (features), and X can be defined as $X=x_1,x_2,\ldots,x_n$=1,2,…,. Bayes' theorem can be written as:

$$P(y|x_1,\ x_2,\ldots,x_n)=P(x_1|y)\ P(x_2|y)\ldots\ P(x_n|y)\ P(y)P(x_1)\ P(x_2)\ldots\ P(x_n)(|1,\ 2,\ldots,)=(1|)\ (2|)\ldots\ (|)\ ()(1)\ (2)\ldots\ ()$$

(3)

There are different NB classifiers depending on the assumption of the distribution of $P(x_i\ |y)$.(|). In this matter, the study of Zhang [46] proved the optimality of NB under the Gaussian distribution even when the assumption of conditional independence is violated (real application cases). Additionally, for multiclass problems, the outcome of the algorithm is the class with the maximum probability. For the Gaussian NB algorithm no parameters have to be tuned.

2.2.5.Multi-Layer Perceptron:

The Multi-Layer Perceptron (MLP) is a class of feedforward artificial neural networks (ANN). A perceptron is a linear classifier that separates an input into two categories with a straight line and produces a single outcome. Input is a feature vector multiplied by specific weights and added to a bias. Contrary to a single-layer case, the MLP can approximate non-linear functions using additional so-called hidden layers. Prediction of probabilities of belonging to any class is calculated through the *softmax* function. The MLP consists of multiple neurons in fully connected multiple layers. Determination of the number of neurons in the layers with a trial-and-error approach remains widely used [47]. Neurons in the first layer correspond to the input data, whereas all other nodes relate inputs to outputs by using linear combinations with certain weights and biases together with an activation function. To measure the performance of the MLP, the logistic loss function is defined with the limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) method as the optimizer for training the network. A detailed and comprehensive description of ANN can be found in [48].

2.3.Methodology:

Figure 3 depicts schematic the methodology followed in this study. The complete dataset for the study consists, as mentioned before, of precipitation and labeled discharge time-series (see Figure 2). The dataset was split in two groups, respectively, for training and testing purposes, and training and test feature spaces were composed for each lead time for the tasks of model hyperparameterization and model assessment. This procedure is repeated for each of the ML techniques studied. Finally, the ranking of the performance quality of all ML methods for every lead time, based on performance metrics and a statistical significance test, were determined.

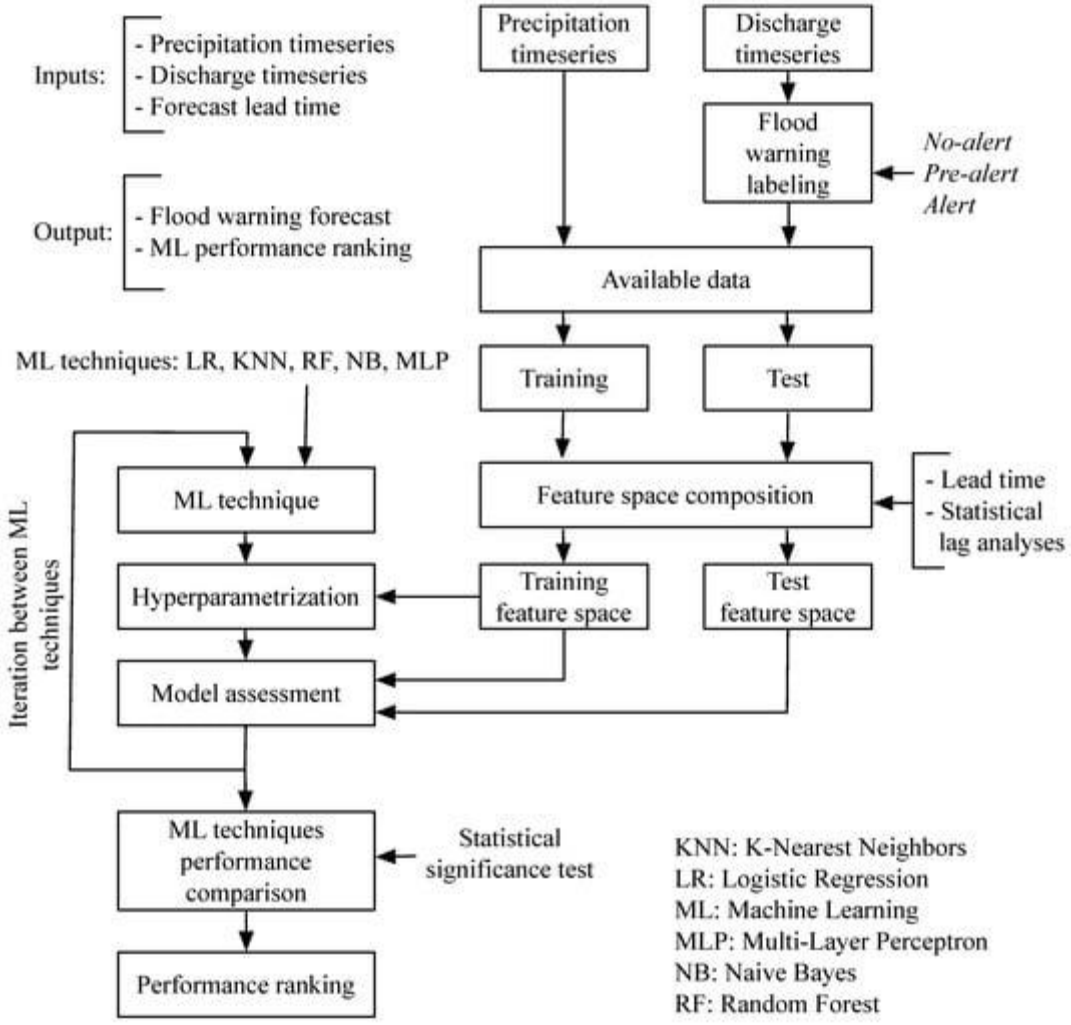


Figure 3. Work schedule for the development and evaluation of the machine learning (ML) flood forecasting models.

2.3.1.Feature Space Composition:

For each lead time, we used single training and testing feature spaces for all ML techniques. A feature space is composed by features (predictors) coming from two variables: precipitation and discharge. The process of feature space composition starts by defining a specific number of precipitation and discharge features (present time and past hourly lags) according to statistical analyses relying on Pearson's cross-, auto and partial-auto-correlation functions [49]. The number of lags from each station was selected by setting up a correlation threshold of 0.2 [28].

Similarly, for discharge, we used several features coming from past time slots of discharge selected for the analysis. It is worth noting that the number of discharge features triples since we replace each discharge feature with three features (one per flood warning class) in a process known as one-hot-encoding or binary encoding. Therefore, each created feature denotes 0 or 1 when the correspondent alarm stage is false or true, respectively. Finally, we performed a feature standardization process before

the computation stage of the KNN, LR, NB, and NN algorithms. Standardization was achieved by subtracting the mean and scaling it to unit variance, resulting in a distribution with a standard deviation equal to 1 and a mean equal to 0.

2.3.2.Model Hyperparameterization:

After the composition of the feature space the optimal architectures for each ML forecasting model, and for each lead time was set up. The optimal architectures were defined by the combination of hyperparameters under the concept of balance between accuracy, and computational cost, and speed. However, finding optimal architectures requires an exhaustive search of all combinations of hyperparameters. To overcome this issue, we relied on the randomized grid search (RGS) with a 10-fold cross-validation scheme. The RGS procedure randomly explores the search space for discretized continuous hyperparameters based on a cross-validation evaluation. Moreover, we selected the f1-macro score (see **Section 2.3.4**) as objective function.

2.3.3. Principal Component Analysis

ML applications require in general the analysis of high-dimension and complex data, involving substantial amounts of memory and computational costs. Reduction of the dimensionality was realized through the application of principal component analysis (PCA) enabling exclusion of correlating features that do not add information to the model. PCA was applied after feature scaling and normalization.

This method enables finding the dimension of maximum variance and the reduction of the feature space to that dimension so that the model performance remains as intact as possible when compared to performance with the full feature space. But considering that each ML technique assimilates data differently, we did not define the number of principal components to keep a fixed threshold of variance explanation (e.g., 80–90%), but performed an exploratory analysis to evaluate its influence on each model. As such, the number of PCAs was treated as an additional hyperparameter, and we optimized the number of principal components for each specific model (lead time and ML technique) with the objective to find the best possible model for each case.

All ML techniques and the RGS procedure were implemented through the scikit-learn package for ML in Python® [**50**]. **Table 1** presents the relevant hyperparameters for each ML technique and their search space for tuning [**38**]. We employed default values for the hyperparameters which are depicted in **Table 1**.

Table 1. Model hyperparameters and their ranges/possibilities for tuning.

2.3.4.Model Performance Evaluation:

Forecasting hydrological extremes such as floods turns into an imbalanced classification problem, and becomes even more complex when the interest lies in the minority class of the data (flood alert). This is because most ML classification algorithms focus on the minimization of the overall error rate, it is

the incorrect classification of the majority class [51]. Resampling the class distribution of the data for obtaining an equal number of samples per class is one solution. In this study, we used another approach that relies on training ML models with the assumption of imbalanced data. The approach we used penalizes mistakes in samples belonging to the minority classes rather than under-sampling or over-sampling data. In practice, this implies that for a given metric efficiency, the overall score is the result of averaging each performance metric (for each class) multiplied by its corresponding weight factor. According to the class frequencies the weight factors for each class were calculated (inversely proportional), using Equation (4).

$$w_i = NC\ n_j =$$

(4)

where w_i is the weight of class i, N is the total number of observations, C is the number of classes, and n_j the number of observations in class i. This implies that higher weights will be obtained for minority classes.

Performance Metrics:

The metrics for the performance assessment were derived from the well-known confusion matrix, especially suitable for imbalanced datasets and multiclass problems, and are respectively the f1 score, the geometric mean, and the logistic regression loss score [51,52,53,54,55,56]. Since neither of the metrics is adequate it is suggested to use a compendium of metrics to properly explain the performance of the model. In addition, those metrics complement each other.

f1 Score

The *f* score is a metric that relies on precision and recall, which is an effective metric for imbalanced problems. When the f score as a weighted harmonic mean, we name this score f1. The latter score can be calculated with Equation (5).

$$\text{f1 score} = 2 \times \text{Precision} \times \text{Recall} \ (\text{Precision} + \text{Recall}) \text{f1 score} = 2 \times \text{Precision} \times \text{Recall} \ (\text{Precision} + \text{Recall})$$

(5)

where precision and recall are defined with the following equations:

$$\text{Precision} = \text{TPTP} + \text{FP} \text{Precision} = \text{TPTP} + \text{FP}$$

(6)

$$\text{Recall} = \text{TPTP} + \text{FN} \text{Recall} = \text{TPTP} + \text{FN}$$

(7)

where TP stands for true positives, FP for false positives, and FN for false negatives.

The f1 score ranges from 0 to 1, indicating perfect precision and recall. The advantage of using the f1 score compared to the arithmetic or geometric mean is that it penalizes models most when either the precision or recall is low. However, classifying a *No-Alert* flood warning as *Alert* might have a different impact on the decision-making than when the opposite occurs. This limitation scales up when there is an

additional state, e.g., *Pre-alert*. Thus, the interpretation of the f1 score must be taken with care. For multiclass problems, the f1 score is commonly averaged across all classes, and is called the f1-macro score to indicate the overall model performance.

Geometric Mean:

The geometric-mean (g-mean) measures simultaneously the balanced performance of TP and TN rates. This metric gives equal importance to the classification task of both the majority (*No-alert*) and minority (*Pre-alert* and *Alert*) classes. The g-mean is an evaluation measure that can be used to maximize accuracy to balance TP and TN examples at the same time with a good trade-off [53]. It can be calculated using Equation (8)

$$G-mean=(TP_{rate}\times TN_{rate})----------\sqrt{}G-mean=(TPrate\times TNrate)$$

(8)

where TP_{rate} TPrate and TN_{rate} TNrate are defined by:

$$TP_{rate}=Recall TPrate=Recall$$

(9)

$$TN_{rate}=TNTN+FP TNrate=TNTN+FP$$

(10)

The value of the g-mean metrics ranges from 0 to 1, where low values indicate deficient performance in the classification of the majority class even if the minority classes are correctly classified.

Logistic Regression Loss:

The metric logistic regression loss (log-loss) measures the performance of a classification model when the input is a probability value between 0 and 1. It accounts for the uncertainty of the forecast based on how much it varies from the actual label. For multiclass classification, a separate log-loss is calculated for each class label (per observation), and the results are summed up. The log-loss score for multi-class problems is defined as:

$$Log\ loss=-1N\sum i=1N\sum j=1My_{ij}\log(p_{ij})=-1\sum=1\sum=1\log()$$

(11)

where N is the number of samples, M the number of classes, y_{ij} equal to 1 when the observation belongs to class j; else 0, and p_{ij} is the predicted probability that the observation belongs to class j. Starting from 0 (best score), the log-loss magnitudes increase as the probability diverges from the actual label. It punishes worse errors more harshly to promote conservative predictions. For probabilities close to 1, the log-loss slowly decreases. However, as the predicted probability decreases, the log-loss increases rapidly.

Statistical Significance Test for Comparing Machine-Learning (ML) Algorithms

Although we can directly compare performance metrics of ML alternatives and claim to have found the best one based on the score, it is not certain whether the difference in metrics is real or the result of statistical chance. Different statistical frameworks are available allowing us to compare the performance of classification models (e.g., a difference of proportions, paired comparison, binomial test, etc.).

Among them, Raschka [57] recommends using the chi-squared test to quantify the likelihood of the samples of skill scores, being observed under the assumption that they have the same distributions. The assumption is known as the null hypothesis, and aims to prove whether there is a statistically significant difference between two models (error rates). If rejected, it can be concluded that any observed difference in performance metrics is due to a difference in the models and not due to statistical chance. In our study we used the chi-squared test to assess whether the difference in the observed proportions of the contingency tables of a pair of ML algorithms (for a given lead time) is significant.

For the model comparison, we defined the statistical significance of improvements/degradations for all lead times (training and test subsets) under a value of 0.05 (chi-squared test). In all cases, the MLP model was used as the base model to which the other models were compared.

3.RESULT:

This section presents the results of the flood forecasting models developed with the LR, KNN, RF, NB, and MLP techniques, and for lead times of 1, 4, 6, 8, and 12 h. For each model, we addressed the forecast of three flood warnings, *No-alert*, *Pre-alert* and *Alert*. First, we present the results of the feature space composition process, taking the 1 h lead time case as an example. Then, we show the results of the hyperparameterization for all models, followed by an evaluation and ranking of the performance of the ML techniques.

3.1.Feature Space Composition:

Figure 4 and Figure 5 show the results of the discharge and precipitation lag analyses for the flood forecasting model 1-h before the flood would occur. Figure 4a depicts the discharge autocorrelation function (ACF) and the corresponding 95% confidence interval from lag 1 up to 600 (h). We found a significant correlation up to a lag of 280 h (maximum correlation at the first lag) and, thereafter, the correlation fell within the confidence band. On the other hand, Figure 4b presents the discharge partial-autocorrelation function (PACF) and its 95% confidence band from lag 1 to 30 h. We found a significant correlation up to lag 8 h (first lags outside the confidence band). As a result, based on the interpretation of the ACF and PACF analyses, and according to Muñoz et al. [28] we decided to include 8 discharge lags (hours) for the case of 1 h flood forecasting in the Tomebamba catchment.

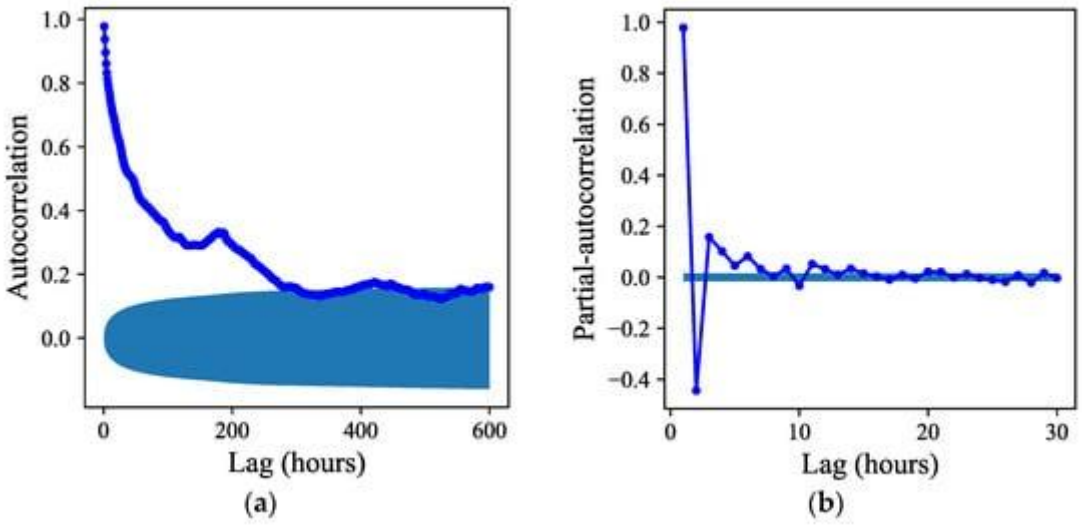


Figure 4. (**a**) Autocorrelation function (ACF) and (**b**) partial-autocorrelation function (PACF) of the Matadero-Sayausí (Tomebamba catchment) discharge series. The blue hatch indicates in each case the correspondent 95% confidence interval.

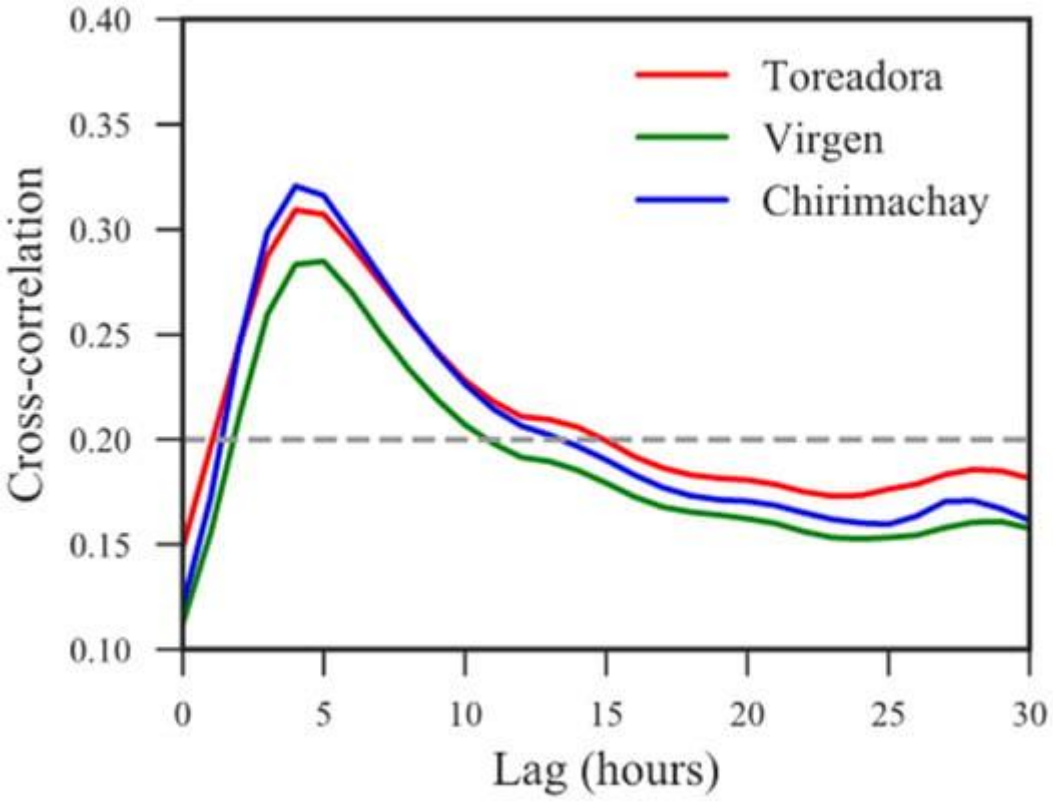
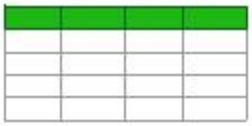


Figure 5. Pearson's cross-correlation comparison between the Toreadora (3955 m a.s.l.), Virgen (3626 m a.s.l.), and Chirimachay (3298 m a.s.l.) precipitation stations and the Matadero-Sayausí discharge series. Note the blue horizontal line at a fixed correlation of 0.2 for determining past lags.

Figure 5 plots the Pearson's cross-correlation between the precipitation at each rainfall station and the discharge at the Matadero-Sayausí stream gauging station. For all stations, we found a maximum correlation at lag 4 (maximum 0.32 for Chirimachay). With the fixed correlation threshold of 0.2, we included 11, 14, and 15 lags for Virgen, Chirimachay, and Toreadora stations, respectively.

Similarly, the same procedure was applied for the remaining lead times (i.e., 4, 6, 8, and 12 h). In Table 2, we present the input data composition and the resulting total number of features obtained from the lag analyses for each forecasting model. For instance, for the 1 h case, the total number of features in the feature space equals 67, from which 43 are derived from precipitation (past lags and one feature from present time for each station), and 24 from discharge (one-hot-encoding).

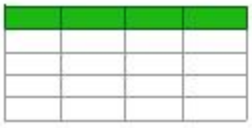
Table 2. Input data composition (number of features) for all ML models of the Tomebamba catchment.



3.2 Model Hyperparameterization:

The results of the hyperparameterization including the number of PCA components employed for achieving the best model efficiencies are presented in Table 3. No evident relation between the number of principal components and the ML technique nor the lead time was found. In fact, for some models we found differences in the f1-macro score lower than 0.01 for a low and high number of principal components. See for instance the case of the KNN models where the optimal number of components significantly decayed for lead times greater than 4 h. For the 1 h lead time, 96% of the components were used, whereas for the rest of the lead times only less than 8%.

Table 3. Model hyperparameters and number of principal components used for each specific model (ML technique and lead time).



If we turn to the evolution of models' complexity with lead time (Table 3) more complex ML architectures are needed to forecast greater lead times. This is underpinned by the fact that the corresponding optimal models require for greater lead times a stronger regularization (lower values of *C*) for LR, a greater number of neighbors (n_neighbors) for KNN, more specific trees (lower values of min_samples_split) for RF and more hidden layers (hidden_layers) for MLP.

3.3.Model Performance Evaluation:

As mentioned before, model performances calculated with the f1-score, g-mean, and log-loss score were weighted according to class frequencies. Table 4 presents the frequency distribution for the complete dataset, respectively, for the training and test subsets. Here, the dominance of the *No-alert* flood class is evident, with more than 95% of the samples in both subsets. With this information, the class weights for the training period were calculated as $w_{No-alert}=0.01$wNo−alert=0.01, $w_{Pre-alert}=0.55$wPre−alert=0.55 and $w_{Alert}=0.51$wAlert=0.51.

Table 4. The number of samples and relative percentage for the entire dataset and the training and test subsets.

The results of the model performance evaluation for all ML models and lead times (test subset) are summarized in Table 5. We proved for all models that the differences in performance metrics for a given lead time were due to the difference in the ML techniques rather than to the statistical chance. As expected, ML models' ability to forecast floods decreased for a longer lead time. For instance, for the case of 1 h forecasting, we found a maximum f1-macro score of 0.88 (MLP) for the training and 0.82 (LR) for the test subset. Whereas, for the 12 h case, the maximum f1-macro score was 0.71 (MLP) for the training and 0.46 (MLP) for the test subset.

Table 5. Models' performance evaluation on the test subset. Bold fonts indicate the best performance for a given lead time.

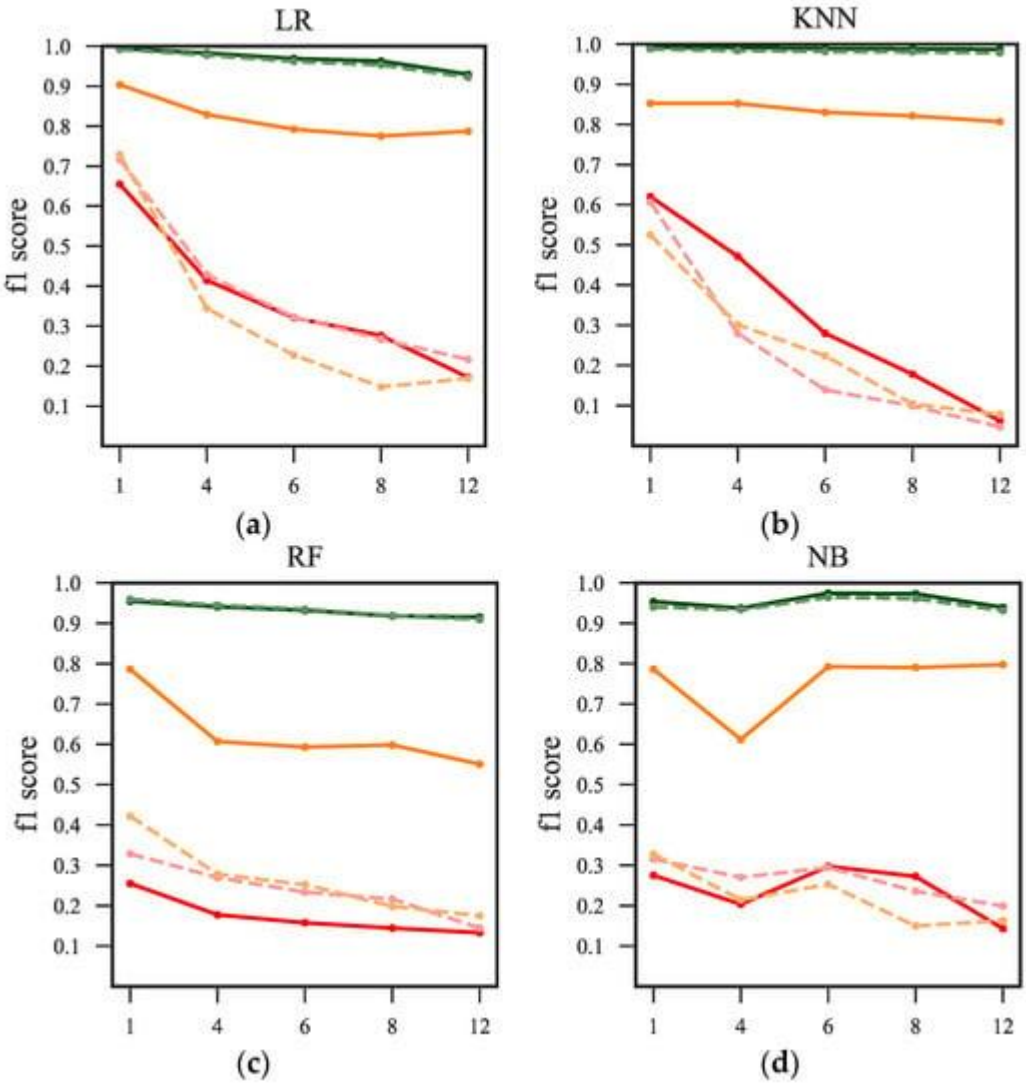
The extensive hyperparameterization (RGS scheme) powered by 10-fold cross-validation served to assure robustness in all ML models and reduced overfitting. We found only a small difference between the performance values by using the training and the test subsets. For all models, maximum differences in performances were lower than 0.27 for the f1-macro score and 0.19 for the g-mean.

In general, for all lead times, the MLP technique obtained the highest f1-macro score, followed by the LR algorithm. This performance dominance was confirmed by the ranking of the models according to the log-loss score. The ranking of the remaining models was highly variable and, therefore, not conclusive. For instance, the results of the KNN models obtained the second-highest score for the training subset, but the lowest for the test subset, especially for longer lead times. This is because the KNN is a memory-based algorithm and therefore more sensitive to the inclusion of information different

to the training subset in comparison to the remaining ML techniques. This can be noted in **Table 4**, where the training and test frequency distributions are different for the *Pre-alert* and *Alert* classes.

On the other hand, for the g-mean score, we obtained a different ranking of the methods. We found the highest scores for the LR algorithm, followed by the RF and the MLP models. Despite this behavior, the values of the g-mean were superior to the f1-macro scores for all lead times and subsets. This is because the f1 score relies on the harmonic mean. Therefore, the f1 score penalizes more a low precision or recall in comparison with a metric based on a geometric or arithmetic mean. Results of the g-mean served to identify that the LR is the most stable method in terms of correctly classifying both the majority *(No-alert*) and the minority (*Pre-alert* and *Alert*) flood warning classes, while the MLP technique could be used to focus on the minority (flood alert) classes.

To extend the last idea, we analyzed the individual f1 scores of each flood warning class. This unveils the ability of the model to forecast the main classes of interest, i.e., *Pre-alert* and *Alert*. **Figure 6** presents the evolution of the f1-score of each ML algorithm at the corresponding lead time. We found that for all ML techniques, the *Alert* class is clearly the most difficult to forecast when the f1-macro score was selected as the metric for the hyperparameterization task. An additional exercise consisted in choosing the individual f1-score for the *Alert* class as the target for hyperparameterization of all models. However, although we obtained comparable results for the *Alert* class, the scores of the *Pre-alert* class had significantly deteriorated, even reaching scores near zero. The most interesting aspect in **Figure 6** is that the most efficient and stable models across lead times (test subset) were the models based on MLP and LR techniques. It is also evident that for all forecasting models, a lack of robustness for the *Pre-alert* warning class was found, and there were major differences between the f1-scores for the training and test subsets. An explanation for this might be that the *Alert* class implies a *Pre-alert* warning class, but not the opposite. Consequently, this might mislead the learning process causing overfitting during training leading to poor performances when assessing unseen data during the test phase.


LR
KNN
RF
NB
f1 score
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1
4
6
8
12
(a)
(b)
(c)
(d)

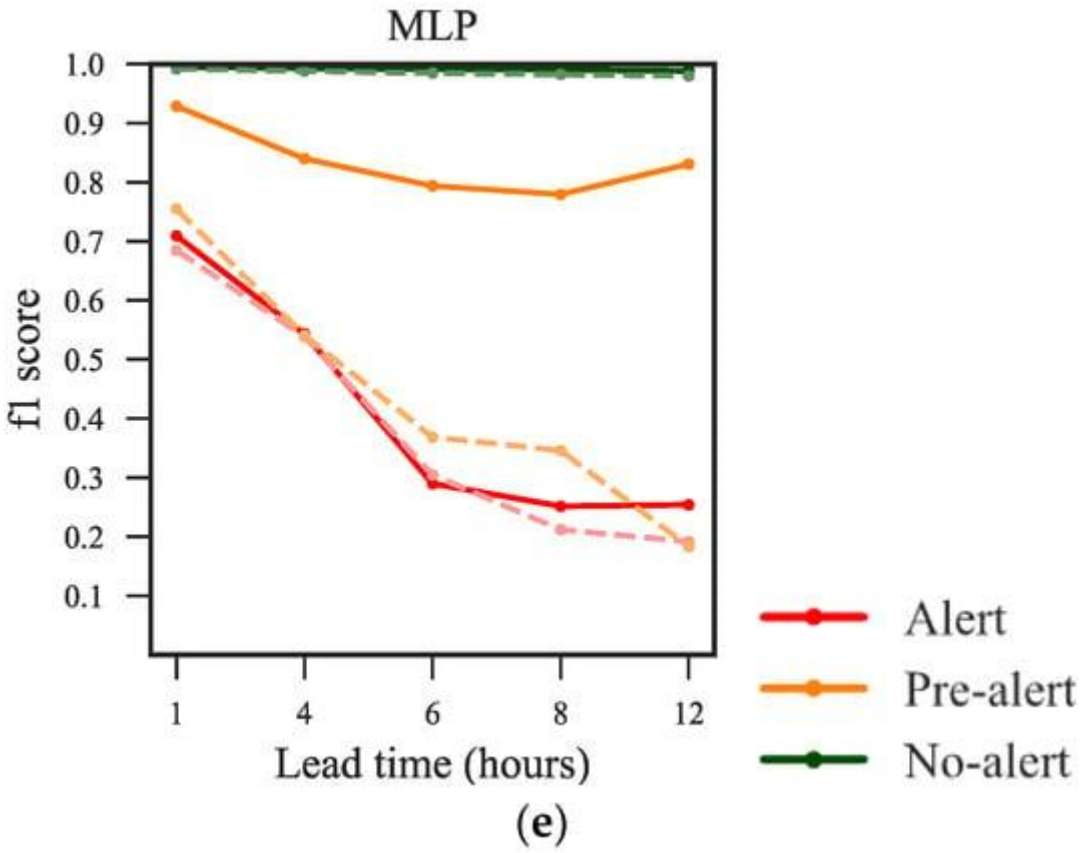


Figure 6. f1 scores per flood warning state (*No-alert*, *Pre-alert* and *Alert*) for all combinations of ML techniques across lead times. (**a**), Logistic Regression (**b**), K-Nearest Neighbors, (**c**) Random Forest, (**d**) Naïve Bayes, and (**e**) Multi-layer Perceptron. The brightest and dashed lines in each subfigure (color coding) represent the scores for the test subset.

Moreover, although we added a notion of class frequency distribution (weights) to the performance evaluation task, it can be noted that for all models, the majority class is most perfectly classified. This is because the *No-alert* class arises from low-to-medium discharge magnitudes. This eases and simplifies the learning process of the ML techniques since these magnitudes can be related to normal conditions (present time and past lags) of precipitation and discharge.

4. Discussion

In this study, we developed and evaluated five different FEWSs relying on the most common ML techniques for flood forecasting, and for short-term lead times of 1, 4, and 6 h for flash-floods, and 8 and 12 h to assess models' operational value for longer lead times. Historical runoff data were used to define and label the three flood warning scenarios to be forecasted *(No-alert*, *Pre-alert* and *Alert*). We constructed the feature space for the models according to the statistical analyses of precipitation and discharge data followed by a PCA analysis embedded in the hyperparameterization.

This was aimed to better exploit the learning algorithm of each ML technique. In terms of model assessment, we proposed an integral scheme based on the f1-score, the geometric mean, and the log-loss score to deal with data imbalance and multiclass characteristics. Finally, the assessment was complemented with a statistical analysis to provide a performance ranking between ML techniques. For all lead times, we obtained the best forecasts for both, the majority and minority classes from the models based on the LR, RF, and MLP techniques (g-mean). The two most suitable models for the dangerous warning classes (*Pre-Alert* and *Alert*) were the MLP and LR (f1 and log-loss scores). This finding has important implications for developing FEWSs since real-time applications must be capable of dealing with both the majority and minority classes. Therefore, it can be suggested that the most appropriate forecasting models are based on the MLP technique.

The results on the evolution of model performances across lead times suggest that the models are acceptable for lead times up to 6 h, i.e., the models are suitable for flash-flood applications in the Tomebamba catchment. For lead times greater than 6 h, we found a strong decay in model performance. In other words, the utility of the 8 and 12 h forecasting models is limited by the models' operational value. This is because, in the absence of rainfall forecasts, the assumption of future rain is solely based on runoff measurements at past and present times. This generates forecasts that are not accurate enough for horizons greater than the concentration-time of the catchment. The concentration-time of the Tomebamba catchment was estimated between 2 and 6 h according to the equations of Kirpich, Giandotti, Ven Te Chow, and Temez, respectively. A summary of the equations can be found in Almeida et al. [58]. This results in an additional performance decay for the 8 and 12 h cases in addition to the error in modeling.

The study of Furquim et al. [31] is comparable. These authors analyzed the performance of different ML classification algorithms for flash-flood nowcasting (3 h) in a river located in an urban area of Brazil. They found that models based on neural networks and decision trees outperformed those based on the NB technique. In addition, the study of Razali et al. [30] proved that decision tree-based algorithms perform better than KNN models, which agrees with our findings. However, such studies only evaluated the percentage of correctly classified instances which is a simplistic evaluation. Thus, we recommend a more integral assessment of model performances, like the one in the current study, which allows for better support in decision making.

Other studies related to quantitative forecasting revealed that neural network-based models usually outperform the remaining techniques proposed in our study [32,33,34]. Similarly, the study of Khalaf et al. [37] proved the superiority of the RF algorithm when compared to the bagging decision trees and HyperPipes classification algorithms. Thus, in certain cases, the use of less expensive techniques regarding the computational costs produces comparable results as in [36]; this is also the case in our short-rain and flash-flood flood classification problem.

As a further step, we propose the development of ensemble models for improving the performance results of individual models. This can be accomplished by combining the outcomes of the ML models with weights obtained, for instance, from the log-log scores. Another alternative that is becoming popular is the construction of hybrid models as a combination of ML algorithms for more accurate and efficient models [24,35,36]. Moreover, as stated by Solomatine and Xue [36], inaccuracies in forecasting floods are mainly due to data-related problems. In this regard, Muñoz et al. [9] reported a

deficiency in precipitation-driven models due to rainfall heterogeneity in mountainous areas, where orographic rainfall formation occurs. In most cases, rainfall events are only partially captured by punctual measurement, and even the entire storm coverage can be missing.

In general precipitation-runoff models will reach at a certain point an effectiveness threshold that cannot be exceeded without incorporating new types of data such as soil moisture [59,60]. In humid areas, the rainfall–runoff relationship also depends on other variables such as evapotranspiration, soil moisture, and land use, which leads to significant spatial variations of water storage. However, these variables are difficult to measure or estimate.

5. Conclusions:

The current study set out to propose a methodology and integral evaluation framework for developing optimal short-rain flood warning forecasting models using ML classification techniques. The proposed analyses were applied to forecast three flood warnings, *No-alert*, *Pre-alert* and *Alert* for the Tomebamba catchment in the tropical Andes of Ecuador. For this, the five most common ML classification techniques for short-term flood forecasting were used. From the results, the following conclusion can be drawn: results related to model comparison are statistically significant. This is important because this is not usually performed in other studies and it validates the performance comparison and ranking hereby presented.

For all lead times, the most suitable models for flood forecasting are based on the MLP followed by the LR techniques. From the integral evaluation (i.e., several performance metrics), we suggest LR models as the most efficient and stable option for classifying both the majority (*No-alert*) and the minority (*Pre-alert* and *Alert*) classes whereas we recommend MLP when the interest lies in the minority classes.

The forecasting models we developed are robust. Differences in the averaged f1, g-mean and log-loss scores between training and test are consistent to all models. However, we limit the utility of the models for flash-flood applications (lead times up to 6 h). For longer lead times, we encourage improvement in precipitation representation, and even forecasting this variable for lead times longer than the concentration-time of the catchment.

A more detailed model assessment (individual f1 scores) demonstrated the difficulties of forecasting the *Pre-alert* and *Alert* flood warnings. This was evidenced when the hyperparameterization was driven for the optimization of the forecast for the alert class and this, however, did not improve the model performance of this specific class. This study can be extended with a deep exploration of the effect of input data composition, precipitation forecasting, and the feature engineering strategies for both the MLP and LR techniques. Feature engineering pursues the use of data representation strategies that could, for example, provide spatial and temporal information of the precipitation in the study area. This can be done by spatially discretizing precipitation in the catchments with the use of remotely sensed imagery. With this additional knowledge, it would be possible to improve the performance of the models hereby developed at longer lead times.

We recommend that future efforts should be put into applying the methodology and assessment framework proposed here in other tropical Andean catchments, and/or benchmarking the results obtained in this study with the outputs of physically based forecasting models. This was not possible for this study due to lack of data.

Finally, for FEWSs, the effectiveness of the models is strongly linked to the speed of communication to the public after a flood warning is triggered. Therefore, future efforts should focus on the development of a web portal and/or mobile application as a tool to boost the preparedness of society against floods.

Objectives & Components:

Objectives:

It will run continuously to monitor the variety of water levels and will give an alert to the authorities using varieties of platform such as using Twilio platform by using GSM messaging.

Components:

1. IoT Sensors:

- Deploy various sensors (water level, weather, rainfall, etc.) in flood-prone areas.
- Sensors should be capable of real-time data collection and transmission.

2. Data Transmission:

- Use wireless communication protocols like LoRaWAN, NB-IoT, or MQTT for sensor data transmission to the central server.

3. Central Server:

- Receive, store, and process the incoming sensor data.
- Implement data validation and authentication mechanisms to ensure data integrity.
- Utilize a database system (like MySQL, PostgreSQL, or NoSQL databases) to store sensor readings.

4. Data Processing and Analysis:

- Implement algorithms to analyze sensor data for flood risk assessment.
- Utilize historical data to identify patterns and predict potential flooding events.

5. Early Warning System:

- Set up predefined thresholds for sensor readings.

- When sensor data crosses these thresholds, trigger automatic alerts.
- Alerts can be in the form of SMS, emails, push notifications, or sirens in the affected areas.

6. User Interface:
- Develop a web-based or mobile application interface for users to access the platform.
- Display real-time sensor data, historical trends, and flood risk assessments.
- Implement intuitive visualizations (charts, maps) for easy understanding of data.
- Provide customization options for users to set their alert preferences.

7. GIS Integration:
- Integrate Geographic Information System (GIS) for mapping flood-prone areas.
- Overlay sensor data on maps to provide a spatial context to the information.

8. Maintenance and Monitoring:
- Implement a system for monitoring sensor health and battery status.
- Set up regular maintenance schedules for sensor calibration and replacement if necessary.

PROGRAM FOR CREATE A PLATFORM TO DISPLAY FLOOD MONITORING SYSTEM:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Flood Monitoring and Early Warning System</title>
  <style>
        Body {
```

```
        Font-family: Arial, sans-serif;
    Margin: 0;
    Padding: 0;
}
     Header {
    Background-color: #4CAF50;
    Color: white;
    Text-align: center;
    Padding: 1em;
}
.container {
    Margin: 20px;
}
.sensor-data {
    Border: 1px solid #ccc;
    Padding: 10px;
    Margin-bottom: 10px;
}
Footer {
   Background-color: #4CAF50;
   Color: white;
    Text-align: center;
    Padding: 1em;
    Position: fixed;
    Bottom: 0;
    Width: 100%;
    }
```

```
  </style>
</head>
<body>
  <header>
    <h1>Flood Monitoring and Early Warning System</h1>
  </header>

  <div class=”container”>
    <h2>Sensor Data</h2>
    <div class=”sensor-data”>
      <h3>Sensor 1</h3>
      <p>Water Level: XX meters</p>
      <p>Last Updated: XX:XX AM/PM</p>
    </div>

    <div class=”sensor-data”>
      <h3>Sensor 2</h3>
      <p>Water Level: XX meters</p>
      <p>Last Updated: XX:XX AM/PM</p>
    </div>
    <!—Add more sensor data sections as needed →

    <h2>Early Warning System</h2>
    <p>Status: <strong>Normal</strong></p>
    <!—You can add more information about the early warning system here →
  </div>
```

```
  <footer>
    <p>&copy; 2023 Flood Monitoring and Early Warning System. All rights reserved.</p>
 </footer>
</body>
</html>
```

Java script(script.js):

```
// Initialize map using Leaflet.js library
Var map = L.map('map').setView([0, 0], 2);
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png').addTo(map);

// Dummy data for flood monitoring (latitude, longitude, and flood status)
Var floodData = [
  { lat: 12.9716, lon: 77.5946, status: 'Flooded' },
  { lat: 40.7128, lon: -74.0060, status: 'Not Flooded' },
  // Add more data points as needed
];

// Add markers to the map based on flood status
For (var I = 0; I < floodData.length; i++) {
  Var marker = L.marker([floodData[i].lat, floodData[i].lon]).addTo(map);
  Marker.bindPopup('Flood Status: ' + floodData[i].status);
}
```

Advantages:

1.Timely detection of possible flood risks and floods.

2.Highly reliable and available real-time data.

3.Tailored solution that can be integrated with external developments at any level (device, connectivity, cloud or user application).

4.Total adaptation and integration with emergency plans.

5.Creation of historic data for Administrations.

6.Low energy consumption.

7.An unlimited number of devices can be included in future extensions.

8.Far-reaching bidirectional communications.

9.Long working life of the equipment.