IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Citizen Alert System for Flood Prediction Using Machine Learning and Edge Computing

Rashmi Ingle
Computer Department
Smt. Kashibai Navale College
Of Engineering, Pune, India
rashmiingle2206@gmail.com

Pratiksha Ghorpade
Computer Department
Smt. Kashibai Navale College
Of Engineering, Pune, India
pratikshaghorpade7@gmail.com

Radhika Raina
Computer Department
Smt. Kashibai Navale College
Of Engineering, Pune, India
radhikaraina183@gmail.com

Gargi Avadhani
Computer Department
Smt. Kashibai Navale College
Of Engineering, Pune, India
gargiavadhani@gmail.com

Prof. R. H. Borhade
Computer Department
Smt. Kashibai Navale College
Of Engineering, Pune, India
rhborhade.skncoe@sinhgad.edu

Abstract- Flooding incidents in India significantly threaten human lives and infrastructure. This research paper presents a comparative analysis of machine learning and IoT techniques integrated with edge computing for flood prediction. Our proposed cyber-physical system aims to provide timely alerts for water flooding, detect illegal debris dumping, and prevent sound pollution. By analyzing diverse models and methodologies, we evaluate their effectiveness in achieving accurate flood prediction. Leveraging machine learning algorithms and IoT sensor data, combined with edge computing capabilities, our system facilitates real-time decision-making and enhances the efficiency of flood management. The findings of this study contribute to informed flood management strategies, mitigating the impact of flooding incidents and safeguarding lives and property.

I. INTRODUCTION

Flooding is a recurring natural disaster in India, causing significant damage to lives and infrastructure. To improve flood management strategies, this research focuses on integrating machine learning, IoT, and edge computing for advanced flood prediction. By leveraging data from various sources and analysing it with machine learning algorithms, accurate flood prediction can be achieved. The integration of IoT devices and edge computing reduces latency and enables real-time decision-making. This research aims to conduct a comparative analysis of different machine-learning models and methodologies to identify the most effective approaches for flood prediction. The findings will enhance flood management and aid in minimizing the impact of flooding incidents in India and beyond.

We compared some algorithms to our Linear Discriminant Analysis (LDA) Model such as

1) Linear Regression:

Linear regression is a statistical modeling technique [5] used to establish a linear relationship between a dependent variable and one or more independent variables. In the context of flood prediction, linear regression can be employed to analyze the relationship between various predictors (such as rainfall, river levels, and temperature) and the likelihood of flooding. By fitting a line to the data points, linear regression can estimate the impact of each predictor on flood prediction and provide insights into the correlation between variables.

2) Support Vector Machines (SVMs):

Support Vector Machines are powerful machine learning algorithms commonly used for classification and regression tasks. SVMs aim to find a hyperplane in a high-dimensional feature space that optimally separates different classes or predicts a continuous variable. In flood prediction, SVMs [6] can be employed to classify areas into different flood risk categories or to estimate the severity of flooding based on input features. SVMs are particularly useful when dealing with non-linear relationships between predictors and flood prediction.

3) Decision Trees (DT):

Decision trees are non-parametric supervised learning models that partition the feature space into regions based on the input features. Each partition or leaf node represents a decision or outcome, while internal nodes correspond to conditions or criteria for splitting the data. In flood prediction, decision trees can be utilized to create a tree-like model that captures the relationships between predictors and flood occurrence. Decision trees are interpretable and can handle both categorical and continuous variables, making them useful for understanding the decision-making process in [7] flood prediction.

4) Artificial Neural Networks (ANNs):

Artificial Neural Networks are computational models [5] inspired by the structure and functioning of the human brain. ANNs consist of interconnected nodes or neurons organized in layers. Each neuron processes information and passes it to the next layer until a final output is obtained. In the context of flood prediction, ANNs can learn complex non-linear relationships between predictors and flood occurrence. By training on historical flood data, ANNs [5] can capture hidden patterns and make predictions based on new input features.

5) Ensemble Learning:

Ensemble learning combines multiple machine learning models to improve prediction accuracy and robustness. It involves training multiple individual models and aggregating their predictions to obtain a final result. In flood prediction [8], ensemble learning methods, such as Random Forests or Gradient Boosting, can be employed to combine the predictions of multiple decision trees or other base models. Ensemble methods often outperform individual models [9] by reducing bias, variance, and overfitting, thus enhancing the overall predictive capability.

6) K-Nearest Neighbors (KNNs):

K-Nearest Neighbors [10] is a simple yet effective algorithm for both classification and regression tasks. KNN operates by identifying the k nearest data points in the training set to a new input sample and using their labels or values to predict the outcome of the new sample. In flood prediction, [11] KNN can be utilized to identify similar historical flood events based on input features and predict the severity or likelihood of flooding. KNN is particularly useful when the underlying data distribution is non-linear and there is a local correlation between nearby data points.

Comparative Analysis-

When comparing these methods for flood prediction individually, several factors should be considered. Linear regression provides interpretability but assumes a linear relationship, while SVMs handle non-linear relationships but are less interpretable. Decision trees are easy to interpret and handle both categorical and continuous variables, but they may suffer from overfitting. ANNs can capture complex relationships but lack interpretability. Ensemble learning combines the strengths of individual models but may be computationally intensive. KNN is simple and effective but relies on the choice of k and suffers from the curse of dimensionality. The choice of the most suitable method depends on the specific requirements, available data, interpretability needs, and computational resources.

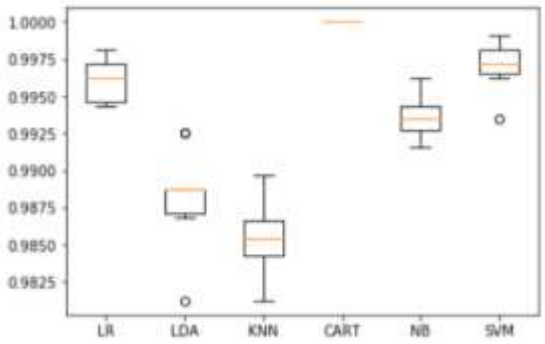

Figure 1: Algorithm Comparison

II. LITERATURE SURVEY

Rath, Subhashree, et al. [1] "IoT and ML based Flood Alert and Human Detection System." 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT). IEEE, 2022:

This paper presents a flood alert and human detection system based on the Internet of Things (IoT) and machine learning (ML). The authors propose an architecture that integrates various sensors, such as water level sensors and cameras, to monitor flood-prone areas. They employ ML techniques, specifically a combination of Support Vector Machines (SVM) and Convolutional Neural Networks (CNN), to detect humans and issue timely flood alerts. The proposed system aims to improve flood preparedness and response by providing real-time monitoring and accurate detection of human presence in flood-affected regions.

Diao, Chao, Guoqing Sang, and JunNuo Wang. "Research on the Application of Remote Sensing Monitoring to Flood Monitoring Based on Sentinel-1A in Linyi City." 2022 3rd International Conference on Geology, Mapping and Remote Sensing (ICGMRS). IEEE, 2022 [2]:

In this paper, the authors investigate the application of remote sensing monitoring, specifically using the Sentinel-1A satellite, for flood monitoring in Linyi City. They utilize Synthetic Aperture Radar (SAR) images to analyze flood extent and severity. The study focuses on the integration of SAR data with Geographic Information System (GIS) techniques to develop flood monitoring and mapping systems. The results demonstrate the effectiveness of using remote sensing technologies for flood monitoring and provide valuable insights for disaster management and response in Linyi City.

Karyotis, Charalampos, et al. [3] "Deep learning for flood forecasting and monitoring in urban environments." 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). IEEE, 2019:

This paper explores the application of deep learning techniques for flood forecasting and monitoring in urban environments. The authors propose a deep learning model that utilizes Long Short-Term Memory (LSTM) networks to predict flood events. The model incorporates various meteorological and hydrological data as input features, enabling accurate flood predictions. The study demonstrates the potential of deep learning in improving flood forecasting accuracy, especially in complex urban

settings. The findings contribute to the development of advanced flood monitoring systems for urban areas.

Rani, Dola Sheeba, G. N. Jayalakshmi, and Vishwanath P. Baligar. "Low-cost IoT-based flood monitoring system using machine learning and neural networks: flood alerting and rainfall prediction." 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). IEEE, 2020 [4]:

In this paper, the authors propose a low-cost IoT-based flood monitoring system that incorporates machine learning and neural networks for flood alerting and rainfall prediction. The system utilizes a network of sensors to collect real-time data on water levels, rainfall, and other environmental parameters. Machine learning algorithms, including Random Forest and Multilayer Perceptron neural networks, are employed to analyze the collected data and provide flood alerts and rainfall predictions. The proposed system aims to address the limitations of traditional flood monitoring approaches by leveraging IoT and ML technologies for cost-effective and accurate flood prediction and alerting.

III. PROPOSED SYSTEM

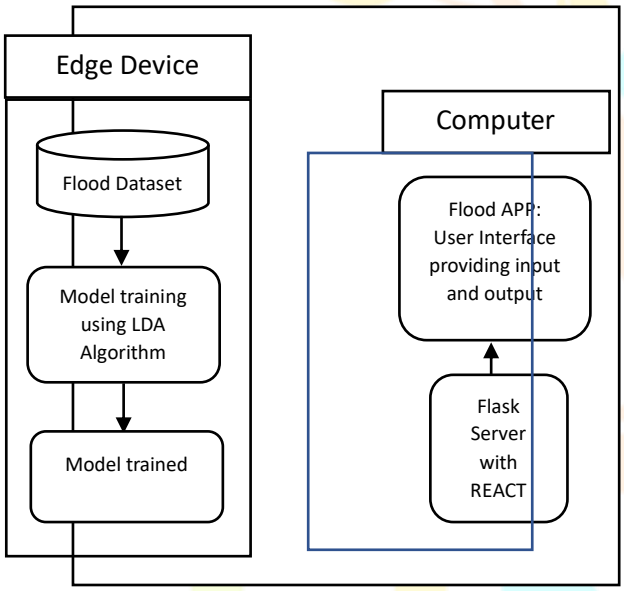

Figure 2. Proposed system architecture

There are three parts majorly used in proposed method:

A. Model Generation

1) Data Collection: Gather a comprehensive dataset related to rivers and their associated flood occurrences. Include relevant features such as river levels, rainfall data, water flow rates, geographical information, and any other variables that may contribute to flood prediction.

2) Data Pre-processing: Perform data cleaning to handle missing values, outliers, and inconsistencies in the dataset. Explore and analyze the dataset to understand the distribution and characteristics of the variables. Split the dataset into training and testing sets for model evaluation.

3) Feature Selection: Identify the most informative and relevant features for flood prediction. Consider features like river levels, rainfall, historical flood data, and any other factors known to influence flooding. Remove any irrelevant or redundant features that may not contribute significantly to the prediction task.

4) Feature Scaling: Normalize or standardize the selected features to ensure they are on a similar scale. Common techniques include min-max scaling or z-score normalization. Scaling helps prevent features with larger magnitudes from dominating the analysis.

5) LDA Model Training: Import the required libraries and modules for LDA, such as scikit-learn in Python. Initialize an LDA classifier object. Fit the LDA model to the training dataset, providing the selected features and corresponding flood labels. The LDA algorithm will learn the discriminant functions that maximize the separation between different flood categories.

6) Model Evaluation: Evaluate the performance of the trained LDA model using the testing dataset. Calculate metrics such as accuracy, precision, recall, or F1-score to assess the model's predictive capabilities. Use techniques like cross-validation to obtain more robust performance estimates and check for overfitting.

7) Model Optimization: Fine-tune the LDA model if necessary by adjusting hyperparameters. Experiment with hyperparameters like regularization parameters or prior probabilities to improve model performance. Employ techniques like grid search or randomized search to find the optimal combination of hyperparameters.

8) Model Deployment and Prediction: Deploy the trained LDA model for flood prediction tasks. Utilize the model to make predictions on new, unseen data related to rivers and their characteristics. Apply the model to classify or predict the likelihood or severity of flooding based on the selected features.

B. Building Android Application:

To build the web application, we utilized the Flask framework, a popular Python web framework for developing web applications. Flask provides a flexible and efficient environment for handling HTTP requests and building the application's functionality. The main.py file serves as the entry point for the application, containing the necessary routes and corresponding functions.

We started by importing the required dependencies, including Flask, Flask-CORS for enabling cross-origin resource sharing, and various modules for data processing and machine learning. Next, we created an instance of the Flask application using `app = Flask(_name_)`. This instance served as the central application object. Routes were defined using the `@app.route` decorator, allowing us to map specific URLs to their corresponding functions. For example, the `/floodHome` route renders the flood_entry.html template, displaying flood alert data. The `/floodResult` route handles form submissions, processing the data and rendering the flood_result.html template with the results.

The application leverages templates, stored in the templates directory, to generate dynamic HTML content. We used the `render_template` function provided by Flask to pass data to the templates and render them appropriately. The templates contain HTML and Jinja2 code to define the user interface and display data from the backend.

C. Application Deployment:

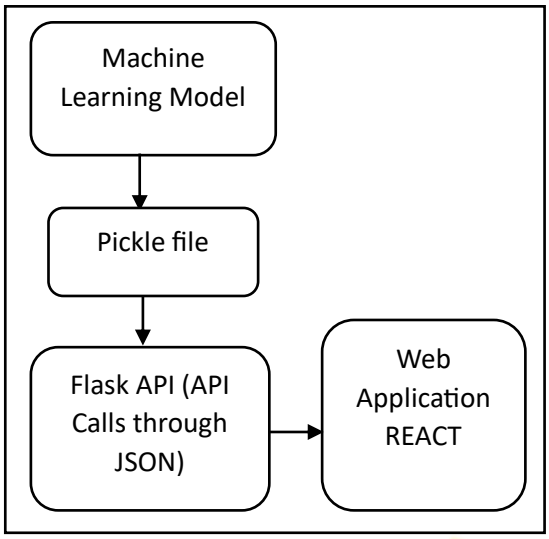

Figure 3. Communication Model between Application and ML Model

Deploying the web application involves making it accessible on a hosting environment or server, allowing users to access it remotely. Here is a general outline of the deployment process:

1. Setting up a Hosting Environment: We established a hosting environment or server to host the web application. This could be a local server or a cloud-based hosting platform.

2. Installing Dependencies: We ensured that all necessary dependencies, including Flask and other required libraries, were installed on the hosting environment. This allowed the application to run smoothly.

3. Transferring Code Files: The code files, including main.py and model.py, were transferred to the hosting environment. Additionally, any templates and static files required for the application's functionality were also transferred.

4. Configuring Server Settings: We configured the server settings to ensure proper handling of incoming requests. This involved setting up domain names, ports, and firewall rules to allow access to the web application.

5. Running the Application: We initiated the Flask application on the hosting environment using the appropriate commands or configurations specific to the hosting platform. This started the development server and made the web application accessible on a specified port.

6. Monitoring and Maintenance: Once the application was deployed, we monitored its performance and addressed any issues or errors that arose. Regular maintenance and updates were performed as necessary to ensure optimal functionality.

IV. EXPERIMENT AND RESULTS

We have defined a function named flood_classifier that implements a machine-learning model using the Linear Discriminant Analysis (LDA) algorithm for flood prediction. Here is an overview of the steps performed in the function:

Data Pre-processing: The function reads an Excel file using the specified filename and loads the data into a pandas Data Frame. Null values in the dataset are filled with the mean values of their respective columns. The target variable 'Flood' is transformed into a binary format (0 or 1) based on a threshold value of 0.1.

Feature Engineering and Visualization: The 'Date' column is processed to extract the day, month, and year information, stored in separate columns. Yearly trends of the 'Discharge' feature are visualized using a line plot.

Data Splitting: The dataset is split into training and testing sets, where the data up zto the end of 2015 is used for training, and the rest is used for testing. The 'Day', 'Months', and 'Year' columns are dropped from both the training and testing sets.

Data Scaling and Upsampling: The feature data is scaled using a normalizer or commented out MinMaxScaler. The training data is upsampled using the SMOTE (Synthetic Minority Over-sampling Technique) algorithm to address the class imbalance issue.

Model Training and Evaluation: The LDA classifier is initialized, trained on the upsampled training data, and saved to a file. The saved model is loaded, and predictions are made on the testing set. The accuracy, mean absolute error (MAE), and classification report are computed to evaluate the model's performance.

Future Data Prediction: There is a separate function named predicting that takes future data as input and returns predictions using the trained LDA model. The function returns the predictions, MAE value, and classification report (as a dictionary) for the testing set.

A. Machine Learning Algorithm Results

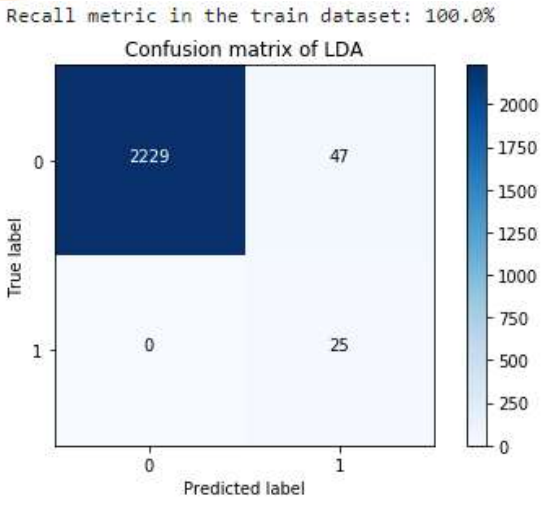

Figure 4. Confusion Matrix of LDA Algorithm

The confusion matrix when applied to Linear Discriminant Analysis (LDA) for flood data prediction, provides insights into the model's accuracy. It compares predicted and actual class labels, indicating true positives, true negatives, false positives, and false negatives. Analyzing the matrix, the calculated metrics like accuracy, precision, recall, and F1-score, giving a comprehensive understanding of the model's performance in predicting floods are presented below.

```
# Make predictions on validation dataset
knn = KNeighborsClassifier()
knn.fit(X_train_res_shuf, Y_train_res_shuf)
predictions = knn.predict(X_test)
print(accuracy_score(Y_test, predictions))
cnf_matrix_tra = confusion_matrix(Y_test, predictions)
print(cnf_matrix_tra )
print(classification_report(Y_test, predictions))
```

```
0.970013037809648
[[2211   65]
 [   4   21]]
              precision    recall  f1-score   support

         0.0       1.00      0.97      0.98      2276
         1.0       0.24      0.84      0.38        25

   micro avg       0.97      0.97      0.97      2301
   macro avg       0.62      0.91      0.68      2301
weighted avg       0.99      0.97      0.98      2301
```

Figure 5. Performance measures of Implemented LDA Algorithm

B. Deployment of Model

We have developed a simple and neat interface in which the home page contains basic information about the web application predicting floods for different rivers on the basis of the date.

Figure 6(a). Web Application deployment screenshot

The Web App consists of a panel on the left with two options to get to the home page and another clickable option of Flood prediction which redirects you to a new page consisting of the predict button where you can select one river among 5 options and select a date to get the status.

Figure 6(b). Web App Prediction page

Upon clicking the predict button we get the river stats as shown in the figure along with the water discharge represented in the figure and also the run-off visualized in the third graph hence alerting the citizens about the conditions resulting to alarming conditions.

Figure 6(c). Web App Flood Predicted values

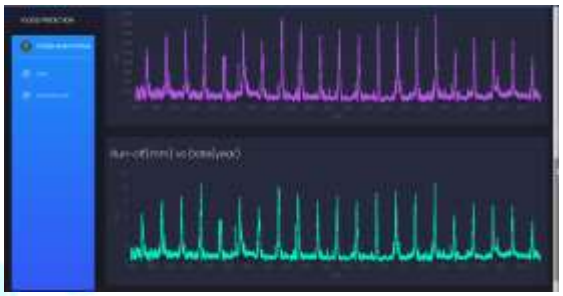

Figure 6(d). Web App water discharge & run-off

Figure 6(e). Web App select date for prediction.

V. CONCLUSION

In conclusion, the Citizen Alert System project has successfully developed a web application that enables users to predict flood alarming conditions for citizens. The application integrates machine learning and edge computing to enhance its performance and efficiency. By leveraging the LDA algorithm machine learning model, accurate flood predictions can be made based on selected rivers and dates.The web application, built using Flask and React, provides a user-friendly interface with options to select from five rivers and choose a specific date for flood prediction. This empowers citizens to proactively prepare for potential flood events and take necessary precautions.

Moreover, the implementation of edge computing technology in the project has significantly reduced latency. By processing data and performing computations locally on edge devices, such as IoT devices or edge servers, the system delivers faster response times and improves the overall user experience. The Citizen Alert System project highlights the potential of combining machine learning, web development, and edge computing to create valuable applications that address critical issues like flood prediction. By providing timely and accurate flood alerts, this system contributes to public safety and disaster management efforts.

Overall, this project demonstrates the power of technological innovations in harnessing data-driven insights and delivering actionable information to citizens, enabling them to make informed decisions during flood

events. The integration of machine learning, web application development, and edge computing showcases a holistic approach to addressing real-world challenges and improving the resilience of communities in the face of natural disasters.

VI. FUTURE SCOPE

In the near future, this system can be transformed into an integrated system that can solve multiple problems in a physical area hence alerting the citizens of admins to take precautionary measures to improve the quality of life and prevent any harm to humans. The objectives like implementing an IoT architecture of sensor nodes to detect sound frequencies, temperature, and illegal debris dumping detection. A message alerting system using a GSM module can be adapted in the future for better alerting.

VII. REFERENCES

[1] Rath, Subhashree, et al. "IoT and ML based Flood Alert and Human Detection System." 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT). IEEE, 2022.

[2] Diao, Chao, Guoqing Sang, and JunNuo Wang. "Research on the Application of Remote Sensing Monitoring to Flood Monitoring Based on Sentinel-1A in Linyi City." 2022 3rd International Conference on Geology, Mapping and Remote Sensing (ICGMRS). IEEE, 2022.

[3] Karyotis, Charalampos, et al. "Deep learning for flood forecasting and monitoring in urban environments." 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). IEEE, 2019.

[4] Rani, Dola Sheeba, G. N. Jayalakshmi, and Vishwanath P. Baligar. "Low cost IoT based flood monitoring system using machine learning and neural networks: flood alerting and rainfall prediction." 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). IEEE, 2020.

[5] Tsakiri, Katerina, Antonios Marsellos, and Stelios Kapetanakis, "Artificial neural network and multiple linear regression for flood prediction in Mohawk River, New York," Water 10.9 (2018): 1158

[6] Han, D., L. Chan, and N. Zhu, "Flood forecasting using support vector machines," Journal of hydroinformatics 9.4 (2007): 267-276.

[7] Khosravi, Khabat, et al, "A comparative assessment of decision trees algorithms for flash flood susceptibility modeling at Haraz watershed, northern Iran," Science of the Total Environment 627 (2018): 744-755.

[8] Sadler, J. M., et al, "Modeling urban coastal flood severity from crowdsourced flood reports using Poisson regression and Random Forest," Journal of hydrology 559 (2018): 43-55.

[9] Lee, Sunmin, et al, "Spatial prediction of flood susceptibility using random-forest and boosted-tree models in Seoul metropolitan city, Korea," Geomatics, Natural Hazards and Risk 8.2 (2017): 1185-1203.

[10] Liu, Moyang, et al, "The applicability of LSTM-KNN model for realtime flood forecasting in different climate zones in China," Water 12.2 (2020): 440.

[11] Ren, Juanhui, et al, "A Novel Hybrid Extreme Learning Machine Approach Improved by K Nearest Neighbor Method and Fireworks Algorithm for Flood Forecasting in Medium and Small Watershed of Loess Region," Water 11.9 (2019): 1848.

[12] Ren, Juanhui, et al, "A Novel Hybrid Extreme Learning Machine Approach Improved by K Nearest Neighbor Method and Fireworks Algorithm for Flood Forecasting in Medium and Small Watershed of Loess Region," Water 11.9 (2019): 1848.

[13] Yan, Jun, et al, "Urban flash flood forecast using support vector machine and numerical simulation," Journal of Hydroinformatics 20.1 (2018): 221-231.

[14] O'Connor, Jim E., and John E. Costa, The world's largest floods, past and present: their causes and magnitudes, No. 1254, Geological Survey (USGS), 2004

[15] Yan, Jun, et al, "Urban flash flood forecast using support vector machine and numerical simulation," Journal of Hydroinformatics 20.1 (2018): 221-231.

[16] Xiang Cao, Tyler Gizinski, "Design, Implementation and Performance of an Edge Computing Prototype Using Raspberry Pis". IEEE. 2022