Building Predictive Models for Flood Level

Serena Shah and Osvaldo Salinas

Background

Following the floods brought by Tropical Storm Allison in June 2001, the Harris County Flood Control District, together with the Federal Emergency Management Agency, initiated the Tropical Storm Allison Recovery Project. A pivotal component of this project involved conducting control surveys leading to the establishment of a comprehensive network of benchmarks known as Floodplain Reference Marks. This network encompasses more than 1,600 reference marks strategically placed across the county. Each reference mark, which may be an engraved disk, rod, or other types of permanent monument, is meticulously affixed to bridges or other stable structures. They are crucial for accurately determining the vertical elevation, thereby providing essential reference points for locating structures within a floodplain. The benchmarks established by the Harris County Flood Control District serve as crucial reference points for determining high water marks, which are key indicators of flood levels during significant storm events.

High water marks are observed indicators of the peak water levels during flood events, often visible as lines of debris, stains, or damage on structures that have been submerged. Once the benchmark data is used to calibrate measurement tools, surveyors can precisely determine the elevation of these high water marks relative to the benchmark's elevation. This involves physically measuring the vertical distance from the benchmark to the point where the high water mark is observed. The dataset is published on an ArcGIS public database as a series of shapefiles for each recorded flood event from 2006 to 2019. The goal of this project is to use this data to build predictive models for flood levels in Harris County.

Preprocessing Data

Shapefiles serve as datasets to map spatial data within ArcGIS software, each accompanied by an attribute table. In this project, multiple shapefiles were merged into a single one. Subsequently, a new field, 'Water_Level_Change', was computed by subtracting benchmark elevation from high water mark elevation, quantifying flooding levels at each benchmark. Following this process, the attribute table of the merged shapefile was exported as a .csv file, facilitating its use in Python through conversion to a dataframe.

Initially, the data loading process involves importing the necessary Python libraries, `pandas` and `numpy`, which are fundamental tools for data manipulation and numerical computation, respectively. The data is then loaded into a Pandas DataFrame from a CSV file. This is a critical first step as it transitions the raw data into a structured format that allows for more efficient and powerful data manipulation:

Once the data is loaded into a DataFrame, the `.shape` and `.size` attributes are used to understand the dimensions of the dataset, specifically, how many records (rows) and features (columns) are included, and the total number of data points:

- `flooding.shape` provides the tuple (2240, 47), indicating 2240 entries and 47 features.
- `flooding.size` calculates the total data points by multiplying rows by columns, yielding 105280.

The `info()` method extends this initial inspection by listing each column along with its data type and count of non-null values, revealing a detailed breakdown of the dataset's structure. A lot of the columns in the dataset are not useful for our analysis. They contain nominal data like the IDs of the benchmarks, the names of the people who made measurements, and some comments they made on the benchmark site.

To refine the dataset for specific analysis needs, predicting flood levels in this case, columns that are not relevant to this objective are omitted. This step involves selecting columns believed to influence flood levels, such as geographic coordinates, storm event data, and benchmark elevation.

Missing values can skew analysis and model training; thus, handling them appropriately is crucial. Median filling is used here, which replaces missing values with the median of their respective column. This method is robust to outliers and does not drastically shift the distribution of the data. Several features in the dataset are categorical (non-numeric) and must be converted to numeric formats to facilitate machine learning algorithms. This process involves mapping each unique category to a specific numeric value. The variable of focus is continuous data but is converted into categories to suit classification models. This is done using quantile-based binning, which ensures that each category has approximately the same number of data points, thereby addressing any potential imbalances. Each bin represents high, medium, and low flooding (low=0, medium=1, high=2).

Model Development and Evaluation

The first step in our predictive modeling process involved partitioning our dataset into training and testing sets. This split ensures that we can train our models on a subset of the data (70%) and then test their performance on an unseen subset (30%) to evaluate their predictive accuracy. The variables related to flood characteristics and geographical markers were designated as independent variables (X), while the flood/water change level was set as the target variable (y).

K-Nearest Neighbor Classifier

The K-Nearest Neighbor (KNN) Classifier was chosen as one of the primary models for this analysis due to its efficacy in handling classification problems through a simple yet powerful approach that involves identifying the 'k' nearest data points. To optimize our KNN model, we did not rely on arbitrary guesses for the 'k' value (number of neighbors). Instead, we employed a systematic approach using cross-validation and grid search techniques to find the most suitable 'k'. The range of 'k' values tested spanned from 1 to 100, and the model's performance was evaluated based on accuracy. The cross-validation process provided a clear indication of the best performing 'k' value. For our dataset, the optimal number of neighbors was found to be 16. This parameter provided the highest accuracy during the testing phase, indicating that it is the most effective at predicting flood levels given the current data.

After training, the KNN model was tested against the unseen data in the test set to predict flood levels. The performance of the model was quantitatively assessed using several key metrics: precision, recall, and the F1-score. These metrics are derived from the confusion matrix and provide insights into the accuracy of the model’s predictions across different flood risk categories (low, medium, high).

K-Nearest Neighbor Classifier - Model Evaluation

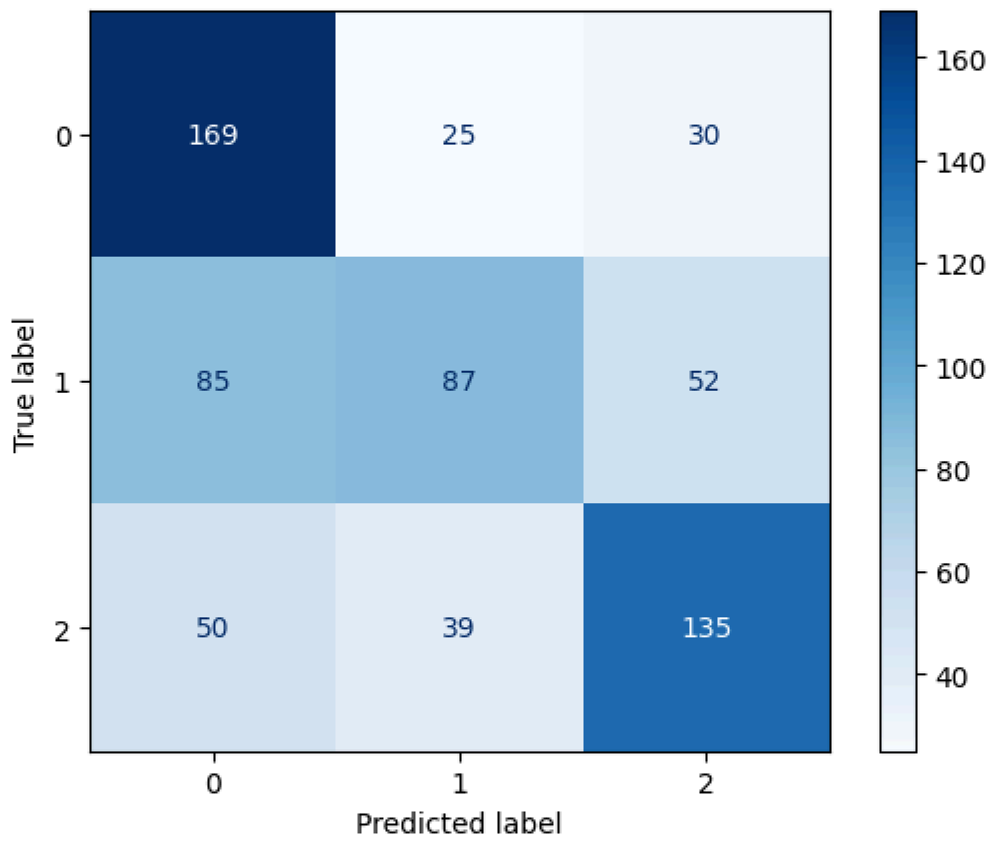
Flood Level	precision	recall	f1-score	support
0	0.56	0.75	0.64	224

1	0.58	0.39	0.46	224
2	0.62	0.60	0.61	224
Accuracy			0.58	672
Macro avg	0.58	0.58	0.57	672
Weighted avg	0.58	0.58	0.57	672

The results show a moderate level of accuracy with an overall accuracy of 58%. Notably, the model performs best in correctly predicting low flood risk (Category 0), with a precision of 0.56 and a recall of 0.75, indicating good sensitivity for this category. The performance on medium and high risk categories shows room for improvement, particularly in increasing the recall for the medium risk category.

To further analyze the model's performance, a confusion matrix was generated. This matrix provides a visual and numerical representation of the predictive performance, with the true labels on the y-axis and the predicted labels on the x-axis.

KNN Confusion Matrix



This matrix shows that the model is relatively better at identifying the low flood risk (169 correct predictions out of 224) but often confuses medium risk (Category 1) for low risk (85 incorrect predictions as low risk).

Finally, to ensure the usability of the KNN model in future applications without retraining, the model is exported and saved to disk using the joblib library. This allows the model to be deployed in different environments or integrated into decision-making tools.

Gaussian Naive Bayes

The Gaussian Naive Bayes model is particularly suited for features that follow a normal distribution and can be effective for classification tasks even with the assumption of independence among features. To evaluate the performance of the Naive Bayes model, we utilized the same metrics as in our previous model assessment (precision, recall, and the F1-score) which are crucial for understanding the effectiveness of the model across different flood risk categories.

Gaussian Naive Bayes - Model Evaluation

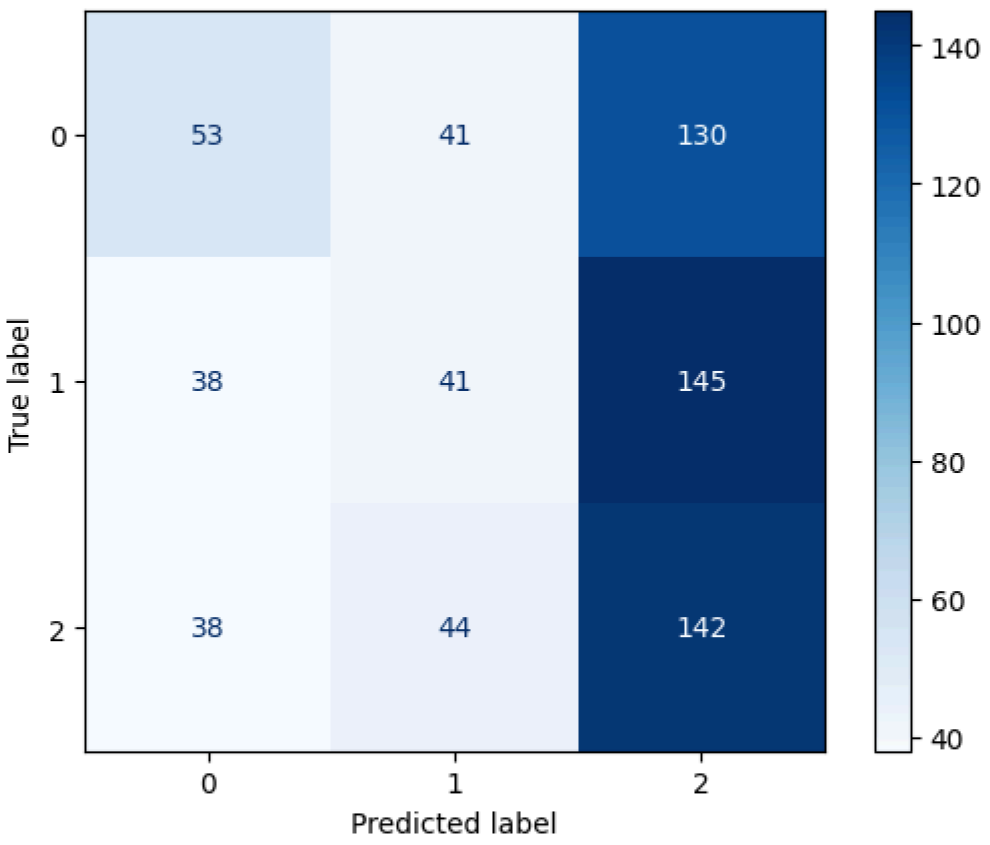
Flood Level	precision	recall	f1-score	support

0	0.41	0.24	0.30	224
1	0.33	0.18	0.23	224
2	0.34	0.63	0.44	224
Accuracy			0.35	672
Macro avg	0.36	0.35	0.33	672
Weighted avg	0.36	0.35	0.33	672

The results indicate modest performance with an overall accuracy of 35%. The model shows a relatively better capability in classifying high-risk floods (Category 2) with a recall of 0.63 but lacks precision across all categories, suggesting a tendency to overpredict higher flood levels.

To visually assess the model's performance, a confusion matrix was generated, illustrating the distribution of predicted versus actual classifications:

Gaussian Naive Bayes Confusion Matrix



This matrix highlights significant misclassification, especially with many actual low and medium risk cases being predicted as high risk, reflecting the model's tendency to favor the high-risk category. While the Gaussian Naive Bayes model provides a baseline for understanding flood risks, its performance suggests there is considerable scope for improvement.

Decision Tree Classifier

Next, we employ a Decision Tree Classifier to predict high flooding levels, utilizing metrics such as accuracy, recall, precision, and F1 score to evaluate the model's performance. The Decision Tree, a fundamental machine learning algorithm, is known for its simplicity and effectiveness in handling nonlinear data relationships through a hierarchical structure of decisions.

Decision Tree Classifier - Model Evaluation

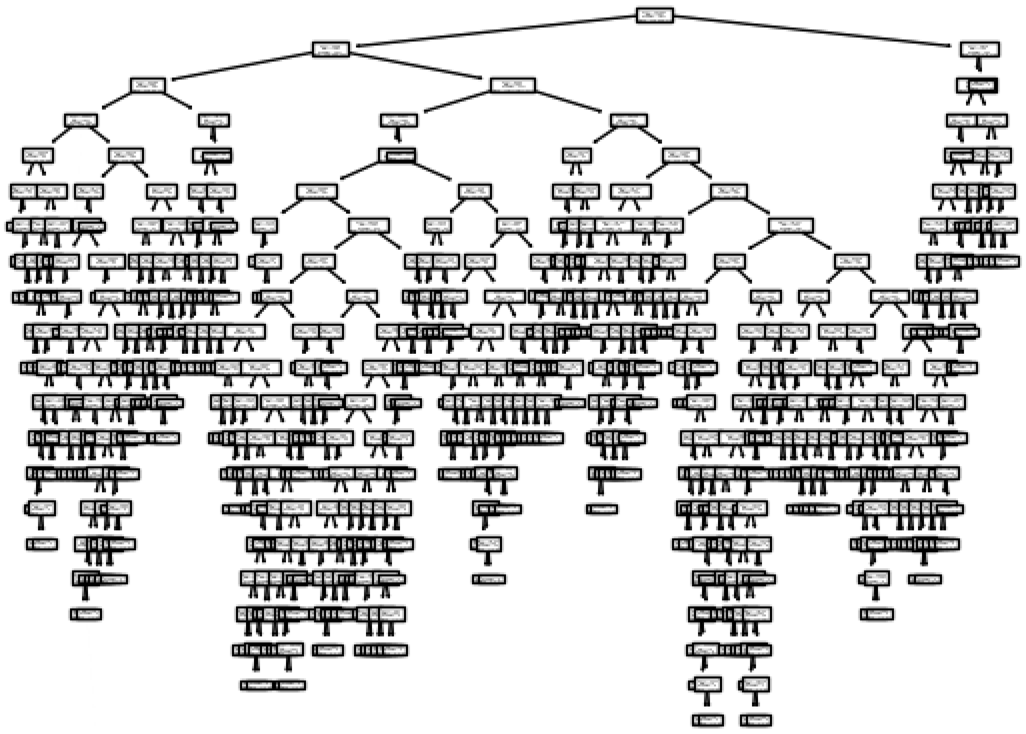
Flood Level	precision	recall	f1-score	support
0	0.58	0.73	0.65	224
1	0.56	0.50	0.53	224
2	0.61	0.53	0.57	224

Accuracy			0.58	672
Macro avg	0.59	0.58	0.58	672
Weighted avg	0.59	0.58	0.58	672

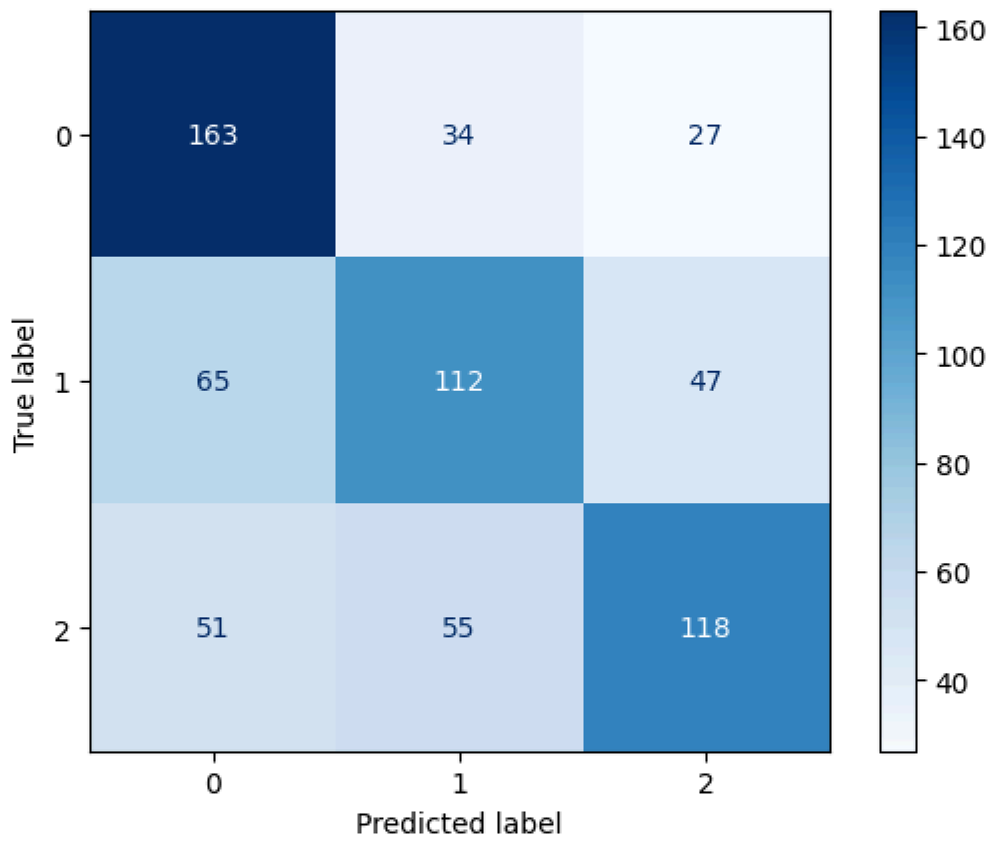
This output highlights moderate precision and recall across the classes, indicating a fair classification of flood levels. The model tends to predict class 0 (no flood or low flood level) more accurately than the higher flood levels.

The structure of the decision tree can be visualized to understand how different features influence the prediction. This is achieved using plot_tree from sklearn.tree. This visual representation helps in identifying the decision-making process of the model, showcasing the splits based on feature importance and thresholds that lead to different flood level predictions.

Decision Tree



Decision Tree Classifier Confusion Matrix



The decision tree classifier shows varying performance across the classes. Class 0 (low level flooding) has the highest number of true positives, indicating a relatively better recognition by the model compared to the other classes. Class 1 (medium flood) and Class 2 (high flood) have more balanced false positives and negatives but show a moderate level of misclassification, suggesting possible areas for model improvement.

Gradient Boosting Machine

In our effort to find the best model for predictive accuracy of high flooding levels, we have implemented a Gradient Boosting Machine (GBM) using XGBoost. To utilize XGBoost, the library was first installed using pip and imported into our Python environment. The model was instantiated as an XGBoost classifier with default parameters, allowing us to benchmark its out-of-the-box performance before any hyperparameter tuning.The performance of the XGBoost model was quantitatively assessed using key metrics such as precision, recall, F1-score, and overall accuracy. Here's a breakdown of the classification report:

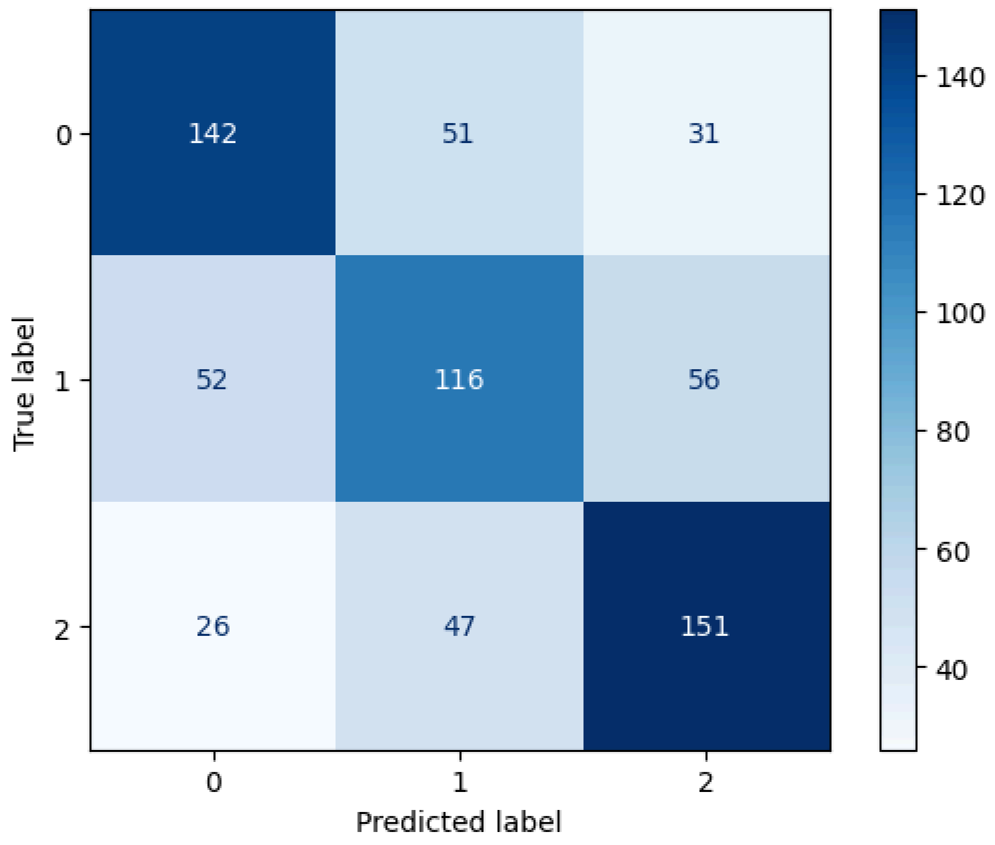
Gradient Boosting Machine - Model Evaluation

Flood Level	precision	recall	f1-score	support

0	0.65	0.63	0.64	224
1	0.54	0.52	0.53	224
2	0.63	0.67	0.65	224
Accuracy			0.61	672
Macro avg	0.61	0.61	0.61	672
Weighted avg	0.61	0.61	0.61	672

The XGBoost classifier's performance for predicting flooding levels shows varied effectiveness across classes. Class 0 (low flooding) has a precision of 0.65 and recall of 0.63, indicating a fairly reliable prediction of low flooding events, reflected by an F1-score of 0.64. For Class 1 (medium flooding), precision drops to 0.54 and recall to 0.52, capturing just over half of true instances, with an F1-score of 0.53, suggesting the model struggles with moderate flooding predictions. In contrast, Class 2 (high Flooding) shows better performance with a precision of 0.63 and a recall of 0.67, highlighting its relative reliability in severe conditions, supported by the highest F1-score of 0.65.

Gradient Boosting Machine Confusion Matrix



Notably, for low flooding (True 0), the model accurately predicted 142 instances, but misclassified 51 as moderate and 31 as high flooding. In the case of moderate flooding (True 1), the model correctly recognized 116 instances, yet misclassified 52 as low and 56 as high flooding. For high flooding (True 2), the model exhibited superior performance, accurately predicting 151 instances, with fewer errors, 26 classified as low and 47 as moderate. This analysis underscores the model's effectiveness in identifying high flooding scenarios with fewer misclassifications compared to its performance with moderate flooding, where it tends to confuse a significant number of events with other categories.

Containerization and API

The Dockerfile specified for the `serenashah/proj04-api` container encapsulates all essential components required to deploy an inference server for predicting flood levels based on location, elevation, and storm event data. It begins by selecting the Python 3.11 base image and installs necessary dependencies like Flask, XGBoost, and scikit-learn. Furthermore, it incorporates the `api.py` script responsible for handling HTTP requests and the `models` directory containing trained classification models. Upon container initialization, the `api.py` script is executed, serving as the entry point for interacting with the flood level prediction API. Utilizing this container involves pulling the Docker image and running it with port mapping to enable communication with the API. Interacting with the API allows users to retrieve model information, understand the expected input format, and obtain flood level predictions based on specified parameters, facilitating informed decision-making and flood risk assessment processes.

The Python Flask API serves as a platform for predicting flood levels using machine learning models, each accessible through specific routes. The /models endpoint, accessed via a GET request, gives essential information about the API, including its version, description, and expected output format. Another GET endpoint, /input_example, gives a JSON example illustrating the required input data format for flood level predictions, encompassing elevation, storm event data, and geographical coordinates. Meanwhile, the /models/knn, /models/dt, /models/nb, and /models/xgb endpoints, accessed through POST requests, facilitate flood level predictions using K-Nearest Neighbors, Decision Tree, Naive Bayes, and XGBoost models, respectively. These endpoints preprocess input data, execute model predictions, and return flood level forecasts categorized as low, medium, or high. Robust error handling mechanisms ensure graceful handling of faulty or missing data, enhancing the API's reliability and user experience for flood risk assessment in diverse scenarios.

Conclusions

In conclusion, our analysis of various machine learning models for predicting flood levels based on location, elevation, and storm event data provides valuable insights into their performance and potential applications. The K-Nearest Neighbor (KNN) classifier demonstrated moderate accuracy, with a notable ability to predict low flood risk accurately but room for improvement in identifying medium and high-risk scenarios. Gaussian Naive Bayes exhibited modest performance, particularly in classifying high-risk floods, albeit with a tendency to overpredict. The Decision Tree classifier showed balanced performance across flood categories, with relatively better accuracy in predicting low flooding levels. Finally, the Gradient Boosting Machine model, implemented using XGBoost, showcased varied effectiveness across flood classes, with superior performance in identifying high flooding scenarios. Notably, the XGBoost model displayed fewer misclassifications in high flooding scenarios compared to moderate flooding, highlighting its effectiveness in severe conditions. The models could be improved by using a larger dataset that includes additional characteristics for the benchmark locations. For example, an updated version of this project would include new calculations like the distance of each marker to a body of water, drainage density, ruggedness number, bifurcation ratio, and so on. Additionally, the models would benefit from using a dataset that has this data for multiple counties. Overall, our analysis underscores the importance of machine learning techniques in predicting and mitigating flood risks, offering valuable tools for decision-making and disaster management in flood-prone areas. By leveraging these models and the containerized API, stakeholders can make informed decisions, implement proactive measures, and enhance resilience against the adverse impacts of flooding events.