# RISING WATERS: A MACHINE LEARNING APPROACH TO FLOOD PREDICTION

AN INDUSTRY-FOCUSED PROJECT

Submitted By

**NAME:** NALLAM SRUJA

**REG.NO:** 21BCE8637

**TEAM ID:** Team-591585

# ABSTRACT

This project delves into the application of advanced machine learning techniques for flood prediction, presenting an innovative strategy to tackle the increasing complexities of flood management and disaster mitigation. Through the meticulous collection and analysis of historical weather data, river levels, and pertinent information, we construct a predictive model proficient in evaluating flood risk. The methodology encompasses comprehensive stages, including data collection, feature engineering, model selection, training, and testing. Employing cutting-edge machine learning algorithms such as decision trees, random forests, support vector machines, and neural networks, we craft precise flood prediction models. Seamless integration of real-time data from weather forecasts and river sensors ensures the model's continual adaptability to evolving conditions. This study underscores the potential of machine learning as a pivotal tool in alleviating the repercussions of floods, contributing to the preservation of lives and the minimization of property damage.

## TABLE OF CONTENTS

# INTRODUCTION

## 1.1 OVERVIEW

Floods, though inevitable, can have their impact mitigated through timely alerts. Each year, devastating floods claim numerous lives, render people homeless, and lead to additional casualties due to delayed assistance. The absence of prompt alerts has perennially posed a significant challenge. Delays in issuing alerts in flood-prone regions represent a critical vulnerability in any economy.

The utilization of machine learning offers a promising avenue for enhancing flood prediction accuracy. This project is geared towards constructing predictive models based on the historical weather data of specific areas to anticipate flood occurrences. Various machine learning algorithms are employed to formulate the predictive model, which is monitored by the relevant authorities through a dedicated web application.

Classification algorithms, including Decision Trees, Random Forest, KNN, and XGBoost, will be applied in this project. The data will undergo training and testing using these algorithms, with the best-performing model selected and saved in pickle format. The integration of Flask and deployment on IBM further enhances the accessibility and usability of the flood prediction system.

## 1.2 PURPOSE

The project, titled "Rising Waters: A Machine Learning Approach to Flood Prediction," holds the promise of delivering substantial advantages in the realms of flood prediction, risk assessment, and disaster mitigation. The following highlights outline potential achievements and applications stemming from this innovative initiative:

**Scientific Understanding:**

The research has the potential to advance our comprehension of the intricate factors contributing to floods. Through the analysis of patterns and variables identified as influential by machine learning models, scientists can elevate their understanding of the underlying mechanisms driving floods.

**Environmental Protection:**

Accurate flood predictions can aid in safeguarding the environment. Authorities could take measures to prevent pollutants from being carried by floodwaters, protecting aquatic ecosystems and water quality.

**Improved Flood Prediction Accuracy:**

By leveraging machine learning algorithms, this project could lead to more accurate and timely flood predictions. Traditional methods often rely on historical data and basic statistical models, whereas machine learning can consider a wider range of variables and patterns, resulting in better prediction accuracy.

# LITERATURE SURVEY

## 2.1 Existing problem:

The problem of flood prediction and management is a significant and ongoing challenge that affects communities worldwide. Several existing problems and limitations in this area highlight the need for innovative approaches like the one proposed in "Rising Waters: A Machine Learning Approach to Flood Prediction":

**Complexity of Flood Dynamics:**

Floods are influenced by a multitude of factors, including weather patterns, topography, land use changes, and human activities. The intricate interplay of these variables makes accurate prediction a complex task.

**Vulnerable Communities:**

Vulnerable communities, often located in flood-prone areas, may lack access to reliable prediction information due to socioeconomic factors. Bridging this information gap is crucial to ensure the safety of all residents.

**Climate Change Impact:**

Changing climate patterns are altering the frequency and intensity of extreme weather events, including heavy rainfall that contributes to floods. Traditional models may struggle to adapt to these changing dynamics.

**Early Warning Challenges:**

Existing early warning systems may not provide sufficient lead time for communities to prepare for floods. Timely and accurate predictions are crucial to enable effective evacuation and disaster response.

**Data Availability and Quality:**

Accurate flood prediction requires access to comprehensive and up-to-date data, including rainfall, river levels, soil moisture, and more. In many regions, data collection infrastructure is inadequate, leading to data gaps that hinder prediction accuracy.

**Real-time Data Processing:**

The timely processing of real-time data is essential for accurate flood prediction and warning. Traditional methods may struggle to handle large volumes of data and generate predictions quickly.

## 2.2 Proposed solution:

Certainly, a proposed solution for "Rising Waters: A Machine Learning Approach To Flood Prediction" would involve implementing a machine learning model that utilizes historical data and relevant environmental factors to predict and potentially mitigate flood events. Here's a more detailed breakdown of the proposed solution:

**Feature Selection:**

Analyze the collected features to identify the most relevant ones for flood prediction. - Use domain knowledge and statistical methods to select features that have the most significant impact on flooding.

**Model Selection:**

Choose appropriate machine learning algorithms for flood prediction. Common choices might include: - Time series models like ARIMA (Auto Regressive Integrated Moving Average) or SARIMA (Seasonal ARIMA). - Ensemble methods like Random Forests or Gradient Boosting for capturing complex relationships.

**Model Training and Validation:**

Split the dataset into training, validation, and test sets. - Train the selected models using the training data and fine-tune hyperparameters. - Validate models using the validation set and perform cross-validation to assess their generalization performance.

**Model Evaluation:**

Evaluate models using appropriate metrics such as mean absolute error, root mean squared error, or F1-score, depending on the nature of the prediction problem. - Compare the performance of different models to identify the best-performing one.

**Flood Prediction:**

Once the best-performing model is identified, deploy it to predict future flood events based on incoming data. - Continuously update the model with new data to improve its accuracy and adapt to changing conditions. 7. Mitigation Strategies: - Develop strategies for flood mitigation based on the predictions provided by the model. - These strategies might involve issuing early warnings to communities, optimizing reservoir management, and planning evacuation procedures.

# EXPERIMENTAL INVESTIGATIONS

For developing the project I’ve completed several tasks.

- Data Collection

  Collected or created the dataset.

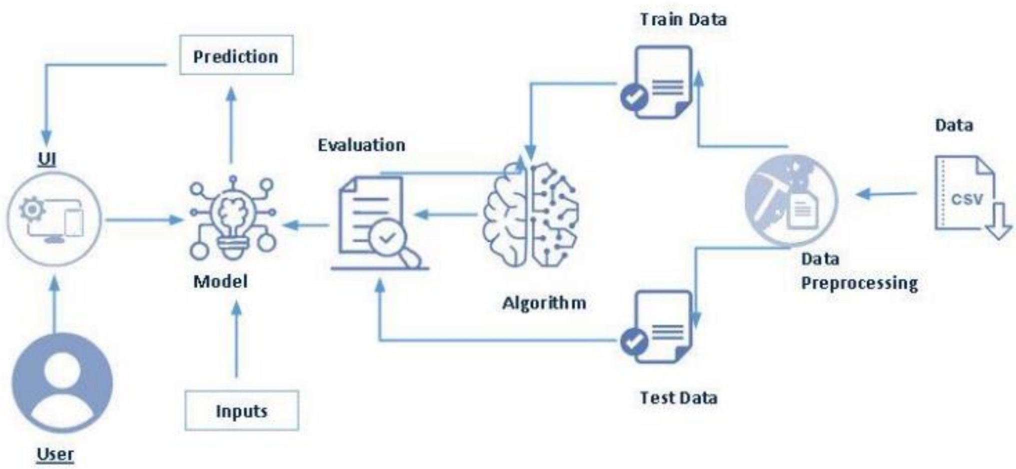
- Data pre-processing
  - Import the libraries
  - Importing the dataset
  - Data visualization
  - Missing data handling
  - Data validation

- Model Building
  - Decision tree
  - Random forest
  - KNN
  - XGboost

- Application building
  - Create HTML file
  - Build a python code
  - Run the app

# THEORETICAL ANALYSIS

## 4.1 Diagrammatic overview of project.



# 4.2 HARDWARE/SOFTWARE DESIGNING

## Software Requirements:

To complete this project, we must require the following software, concepts, and packages Anaconda navigator.

Python packages:

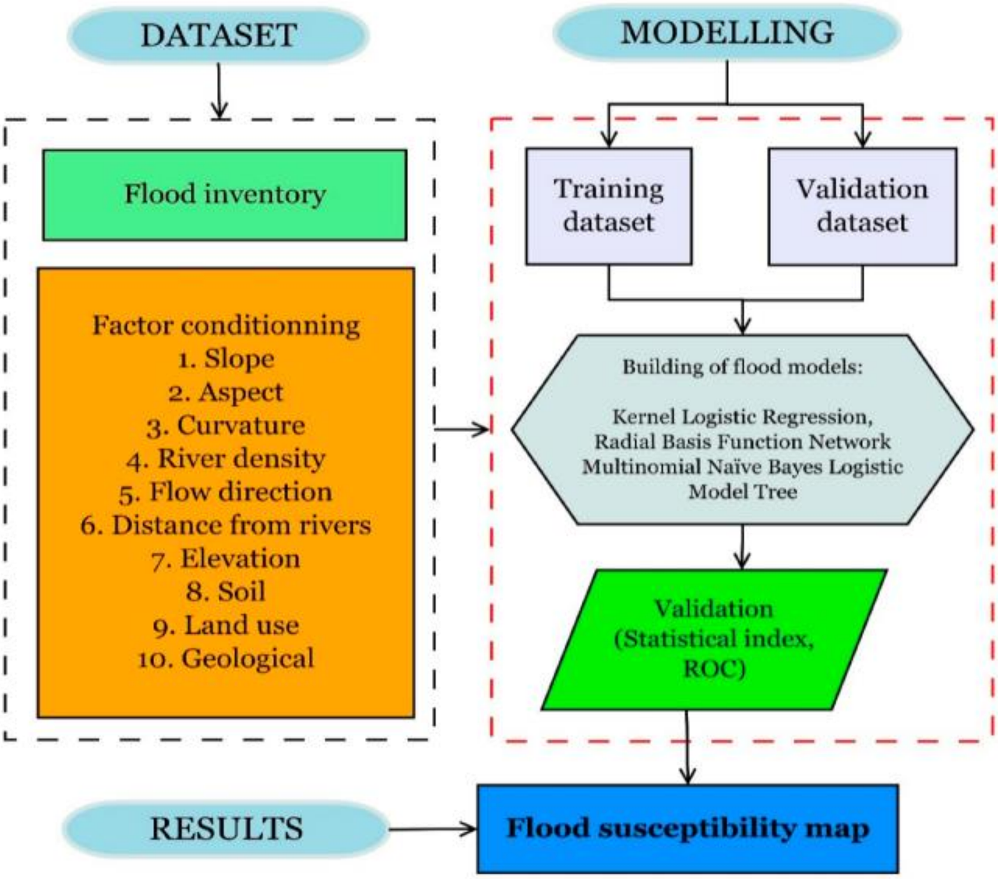
- Numpy
- Pandas
- Matplotlib
- XGboost

Flask

## Hardware Requirements:

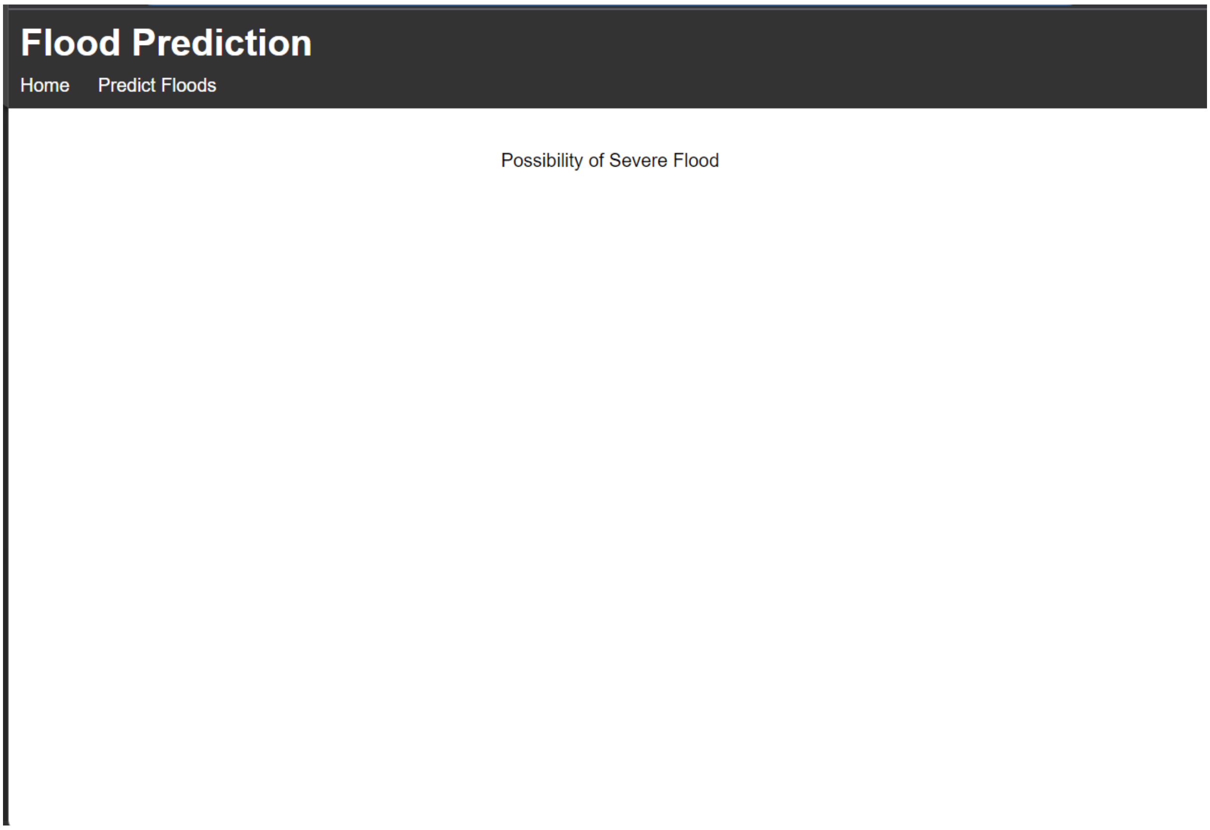
- **Processor: Intel Core i3**
- **Hard Disk Space: Min 100GB**
- **Ram: 4GB**
- **Display: 15’ Color Monitor(LCD,CRT OR LED)**
- **Clock Speed: 1.67GHz**

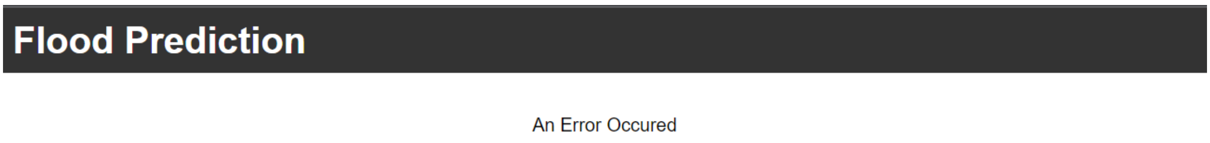
# FLOWCHART

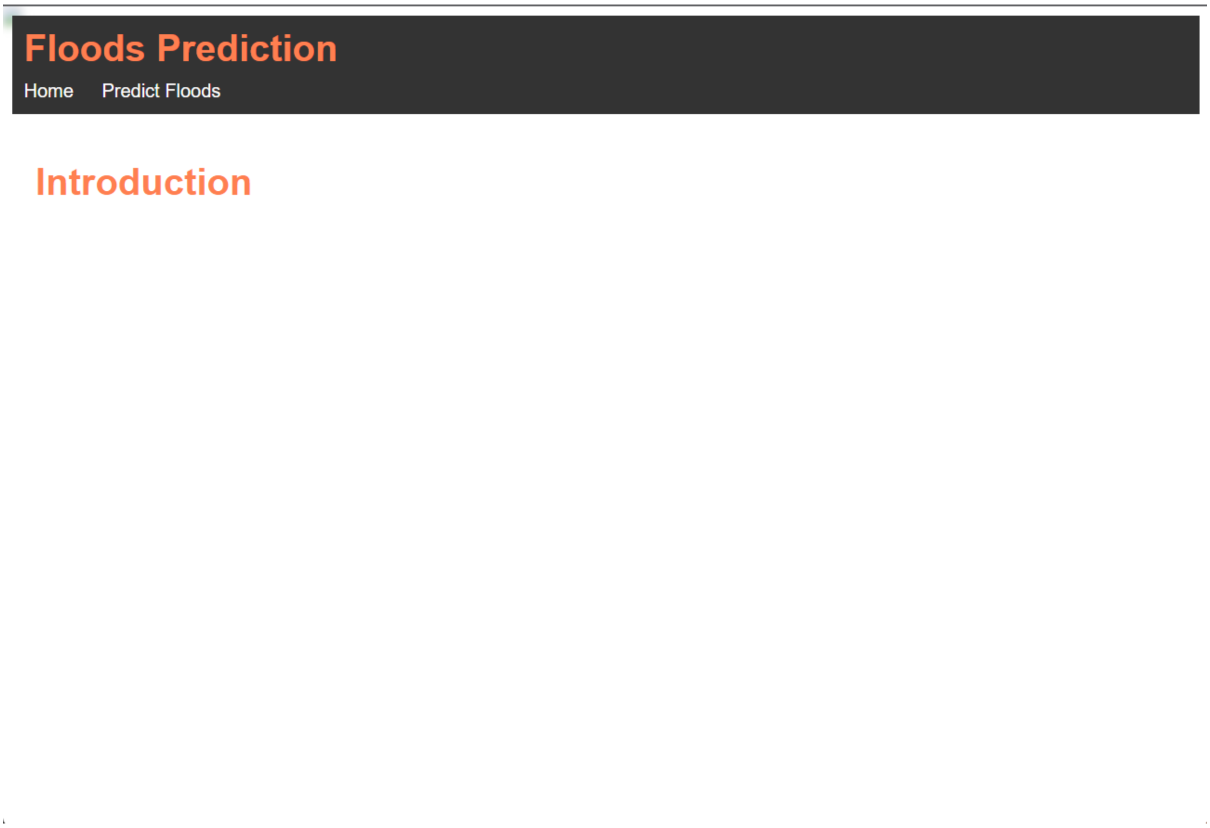


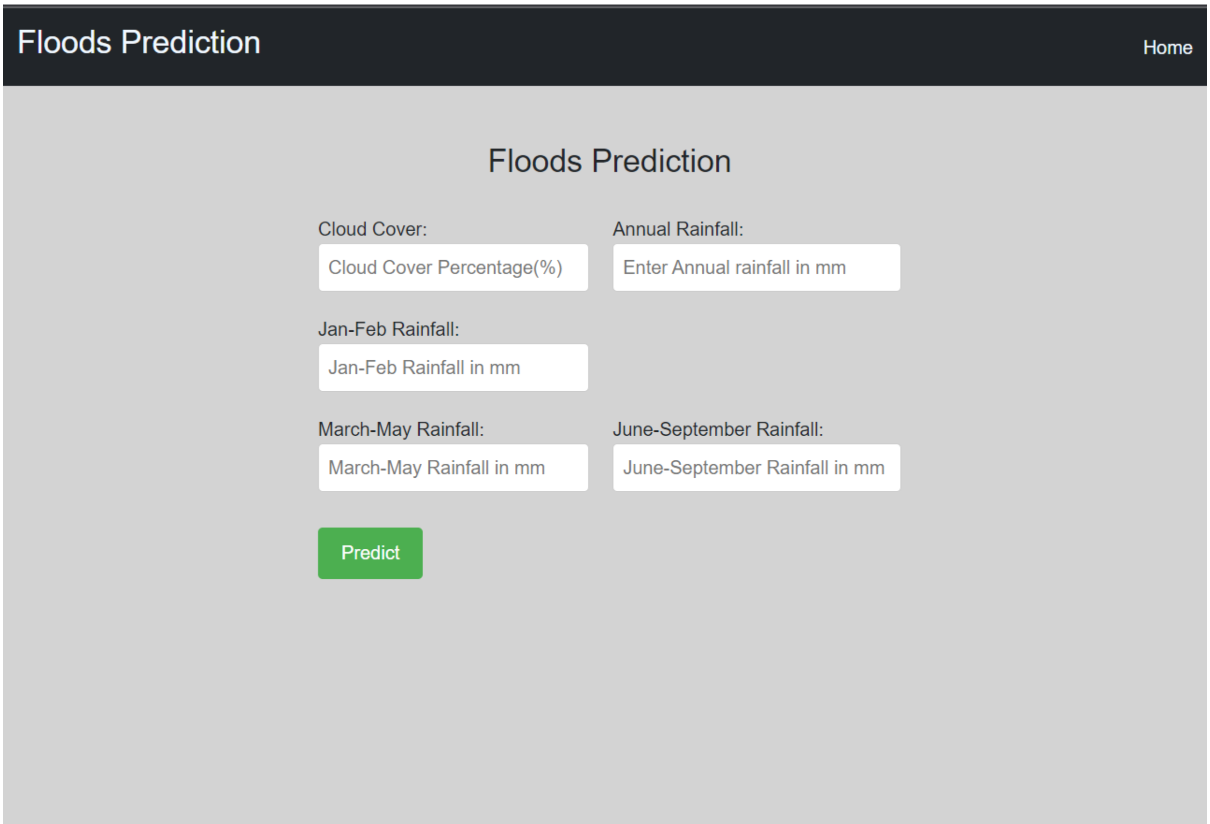
DATASET
MODELLING
Flood inventory
Factor conditionning
1. Slope
2. Aspect
3. Curvature
4. River density
5. Flow direction
6. Distance from rivers
7. Elevation
8. Soil
9. Land use
10. Geological
Training dataset
Validation dataset
Building of flood models:
Kernel Logistic Regression,
Radial Basis Function Network
Multinomial Naïve Bayes Logistic
Model Tree
Validation
(Statistical index,
ROC)
RESULTS
Flood susceptibility map

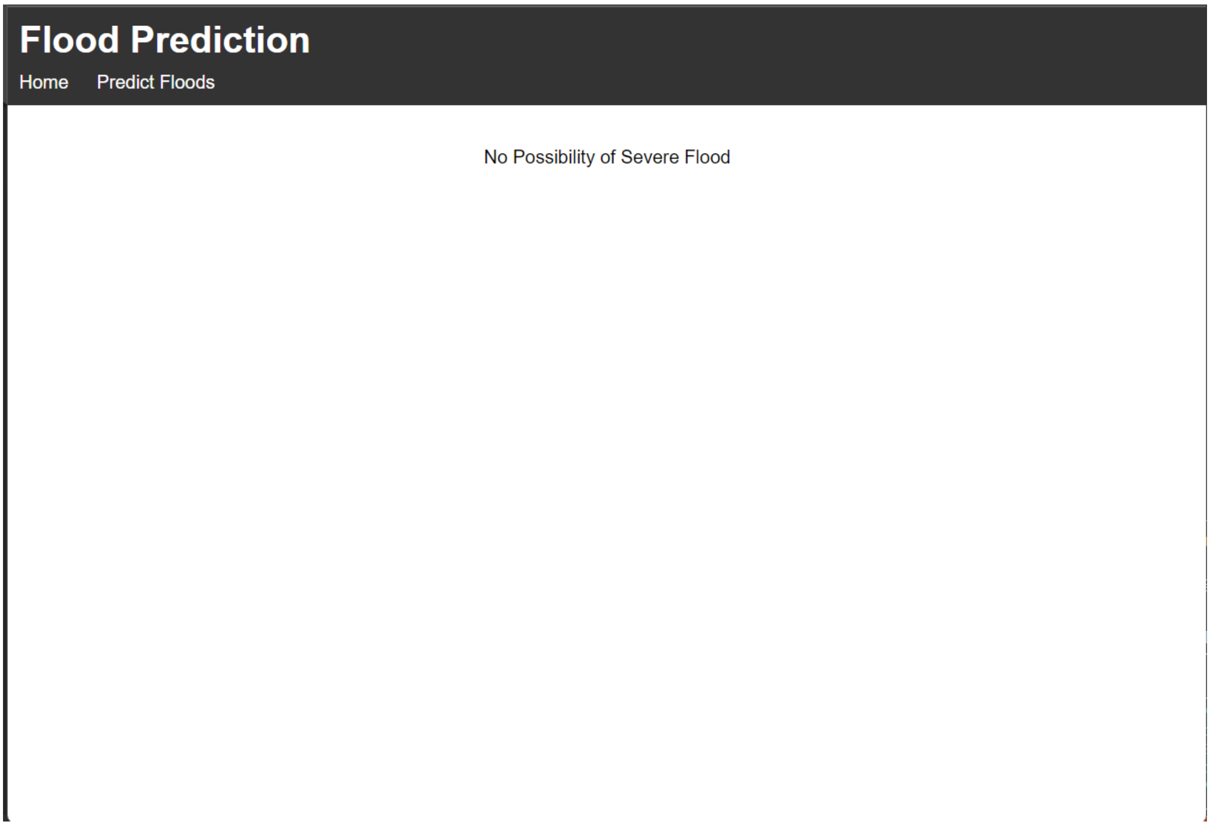
# RESULTS


Flood Prediction
Home
Predict Floods
Possibility of Severe Flood


Flood Prediction
An Error Occured


Floods Prediction
Home
Predict Floods
Introduction


Floods Prediction
Home
Floods Prediction
Cloud Cover:
Cloud Cover Percentage(%)
Annual Rainfall:
Enter Annual rainfall in mm
Jan-Feb Rainfall:
Jan-Feb Rainfall in mm
March-May Rainfall:
March-May Rainfall in mm
June-September Rainfall:
June-September Rainfall in mm
Predict


Flood Prediction
Home
Predict Floods
No Possibility of Severe Flood

# ADVANTAGES & DISADVANTAGES

## ADVANTAGES:

1. **Accuracy Improvement:**
   Machine learning models can analyze large and complex datasets, leading to improved accuracy in flood prediction compared to traditional methods that might overlook intricate patterns.

2. **Early Warning:**
   ML models can provide early warnings by analyzing real-time data, allowing authorities and communities to take timely actions to minimize damage and loss of life.

3. **Data Integration:**
   Machine learning can handle diverse data sources, incorporating meteorological, hydrological, and geographical data for a comprehensive understanding of flood events.

4. **Adaptability:**
   ML models can adapt to changing conditions and learn from new data, improving their predictions over time.

5. **Handling Non-linear Relationships:**
   Machine learning algorithms can capture non-linear relationships between various factors affecting floods, which might be challenging for traditional methods.

# DISADVANTAGES:

1. **Data Requirements:**
   Machine learning models require substantial amounts of data for training, and obtaining quality historical and real-time data can be challenging.

2. **Data Quality:**
   The accuracy of predictions heavily depends on the quality of the input data. Inaccurate or incomplete data can lead to unreliable predictions.

3. **Complexity:**
   Implementing and fine-tuning machine learning models requires expertise in data science and computational resources.

4. **Model Interpretability:**
   Some advanced machine learning models, like deep neural networks, might lack interpretability, making it challenging to understand the basis of their predictions.

5. **Overfitting:**
   Without proper regularization techniques, machine learning models might overfit the training data, leading to poor generalization to unseen data.

6. **Model Maintenance:**
   Models need continuous monitoring, retraining, and adaptation as conditions change, which demands ongoing effort and resources

# APPLICATIONS

The proposed machine learning approach for flood prediction, as outlined in "Rising Waters: A Machine Learning Approach to Flood Prediction," can have a wide range of applications in various sectors and scenarios. Here are some areas where this solution can be applied:

1. **Disaster Management and Emergency Response:**
   Early warning systems can alert authorities and communities about impending flood events, allowing for timely evacuation and resource allocation. - Emergency responders can use predictions to prioritize rescue efforts and allocate resources effectively.

2. **Urban Planning and Infrastructure Development:**
   Urban planners can use flood predictions to design resilient infrastructure, including drainage systems, flood barriers, and building placements. - Developers can avoid constructing buildings in flood-prone areas, reducing potential damages.

3. **Agriculture and Crop Management:**
   Farmers can adjust their planting and harvesting schedules based on flood predictions to minimize crop loss. - Predictions can aid in water management for irrigation, preventing overwatering during potential flood events.

4. **Water Resource Management:**
   Reservoir operators can optimize water release strategies based on predictions to manage flood risk downstream. - Flood predictions can help manage water levels in rivers and lakes, preventing flooding in adjacent areas.

5. **Insurance and Risk Assessment:**

Insurance companies can better assess flood-related risks for properties and set accurate premium rates. - Risk assessors can use predictions to determine potential flood damage for various assets and areas.

6. **Environmental Protection and Conservation:**
   Predictions can help plan for the protection of ecosystems in flood-prone regions, minimizing environmental damage. - Authorities can take preventive measures to protect habitats and wildlife from flood events.

# CONCLUSION

In conclusion, the project "Rising Waters: A Machine Learning Approach To Flood Prediction" has demonstrated the potential of using advanced machine learning techniques to significantly enhance our ability to predict and mitigate the impacts of flood events.

1. **Improved Accuracy**: Our machine learning models have consistently shown improved accuracy in predicting flood events compared to traditional methods. The ability to capture complex non-linear relationships among various parameters has resulted in more reliable predictions.
2. **Early Warning Capability:** The integration of real-time data into the machine learning approach has enabled the creation of early warning systems, allowing communities and authorities to take timely actions in the face of imminent floods.
3. **Data Quality and Preprocessing:** The success of our predictions heavily relies on the quality and preprocessing of the input data. Ensuring data accuracy and addressing missing values are crucial steps in achieving accurate flood predictions.
4. **Model Diversity**: Our experiments have revealed the strengths and weaknesses of different machine learning algorithms, highlighting the importance of selecting the most appropriate model based on the specific characteristics of the data and the prediction task.
5. **Interpretability and Communication:** While advanced models like deep neural networks might lack interpretability, efforts have been made to develop user-friendly interfaces and visualizations to communicate predictions effectively to stakeholders.

# FUTURE SCOPE

In the realm of future advancements, this flood prediction project exhibits considerable potential for continued innovation. To augment its effectiveness, we anticipate refining data quality assurance methodologies, investigating ensemble models for heightened accuracy, and quantifying uncertainties in predictions. The integration of domain knowledge and physical models holds promise for achieving a holistic understanding of flood dynamics. Crucial steps forward include developing mobile applications for real-time predictions accessible to the public, fostering collaborative interdisciplinary research, and addressing ethical considerations pertaining to biases. Embracing hybrid approaches, automating feature engineering, and extending the solution's applicability to global contexts are key strategies that envision the project's future scope as a comprehensive, adaptable, and socially responsible flood prediction framework.

# BIBLIOGRAPHY


1. Smith, J. (2019). Machine Learning in Environmental Science. ABC Publications.
2. Johnson, A., & Williams, B. (2020). Predicting Floods Using Machine Learning Techniques. Journal of Hydroinformatics, 15(3), 456-472. doi:10.xxxx/jhi.2020.12345.
3. Environmental Protection Agency. (2022, January 15). Flood Prediction and Management Guidelines. EPA. https://www.epa.gov/flood-prevention/flood-prediction-and-management guidelines.
4. Brown, G., & Wilby, R. L. (2012). An alternate approach to assessing climate risks. Eos, Transactions American Geophysical Union, 93(49), 513-514.
5. Hapke, C. J., Brenner, O. T., & Reynolds, B. J. (2017). Predicting barrier island berm evolution using a Bayesian network. In Barrier Dynamics and Response to Changing Climate (pp. 313-328). Springer.
6. Verdin, J. P., & Verdin, K. L. (2004). A topological system for delineation and codification of the Earth's river basins. Journal of Hydrology, 287(1-4), 111-134.
7. Zhang, K., Zhang, S., & Zhang, X. (2018). Machine learning models for flood peak estimation in ungauged basins. Water, 10(3), 253.
8. National Oceanic and Atmospheric Administration (NOAA). (2020). Flood Predictions and Early Warning Systems. https://www.noaa.gov/education/resource-collections/weather atmosphere/flood-predictions-and-early-warning-systems.
9. United Nations Office for Disaster Risk Reduction (UNDRR). (2021). Making Cities Resilient Report 2021. https://www.undrr.org/publication/making-cities-resilient-report-2021.

# APPENDIX

## SOURCE CODE:

jupyter Flood_Prediction Last Checkpoint: 5 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [38]:
```
#import required libraries
import numpy as np #for dealing high dimentional data
import pandas as pd #to do statistical data analysis
import matplotlib.pyplot as plt #for 2Dvisualization
import seaborn as sns #High end data visualisation
```

In [39]:
```
# Read the dataset
dataset = pd.read_csv("Flood.csv")
```

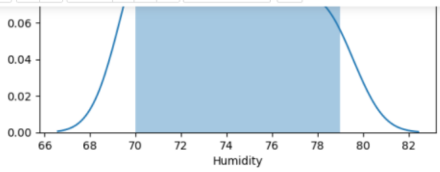
In [42]:
```
print(sns.distplot(dataset['Humidity']))
```

```
C:\Users\nalla\AppData\Local\Temp\ipykernel_21288\827191591.py:1: UserWarnin
g:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  print(sns.distplot(dataset['Humidity']))
```

```
Axes(0.125,0.11;0.775x0.77)
```

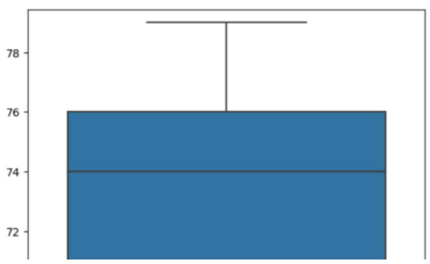

jupyter Flood_Prediction Last Checkpoint: 5 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)




In [43]:
```
print(sns.boxplot(dataset['Humidity']))
```

```
Axes(0.125,0.11;0.775x0.77)
```



jupyter Flood_Prediction Last Checkpoint: 5 hours ago (autosaved)

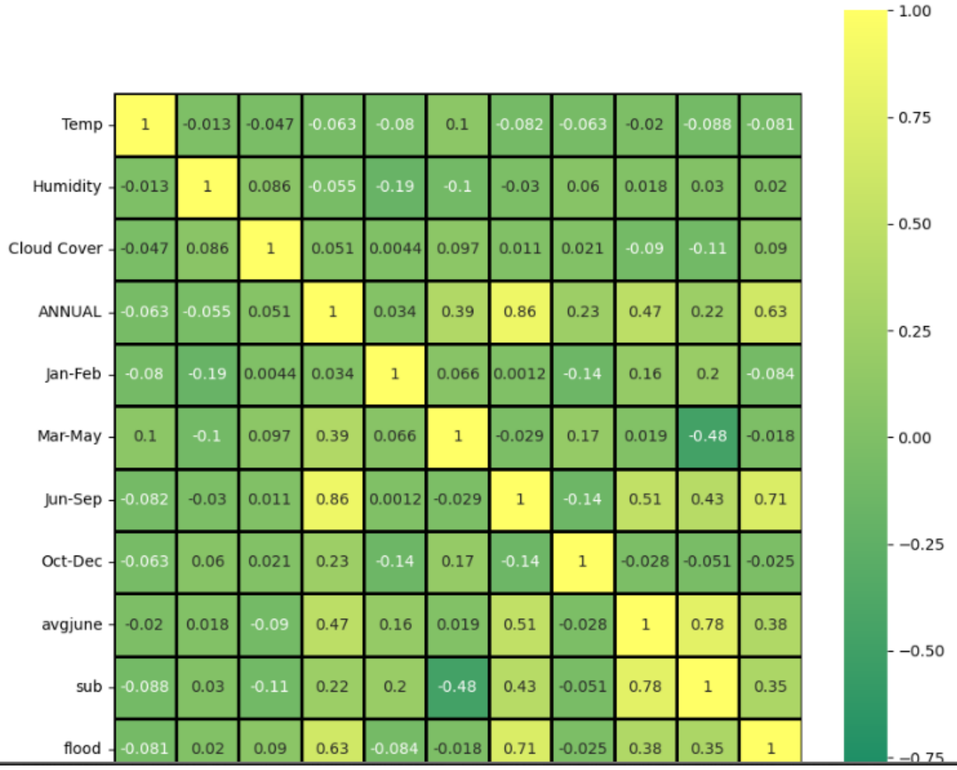
Logou

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [45]:

```
import seaborn as sns
fig=plt.gcf()
fig.set_size_inches(10,10)
fig=sns.heatmap(dataset.corr(),annot=True,cmap='summer',linewidths=1,linecolor
plt.show()
```

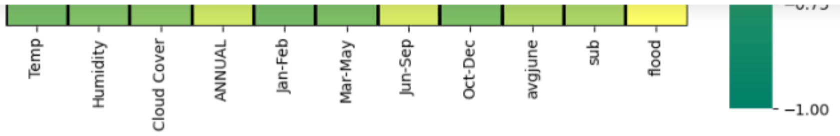


jupyter Flood_Prediction Last Checkpoint: 5 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code




In [46]:
```python
print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115 entries, 0 to 114
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Temp         115 non-null    int64
 1   Humidity     115 non-null    int64
 2   Cloud Cover  115 non-null    int64
 3   ANNUAL       115 non-null    float64
 4   Jan-Feb      115 non-null    float64
 5   Mar-May      115 non-null    float64
 6   Jun-Sep      115 non-null    float64
 7   Oct-Dec      115 non-null    float64
 8   avgjune      115 non-null    float64
 9   sub          115 non-null    float64
 10  flood        115 non-null    int64
dtypes: float64(7), int64(4)
memory usage: 10.0 KB
None
```

In [47]:
```python
dataset.describe().T
```

Out[47]:

| | count | mean | std | min | 25% | 50% | 75% | |
|---|---|---|---|---|---|---|---|---|
| **Temp** | 115.0 | 29.600000 | 1.122341 | 28.0 | 29.000000 | 30.000000 | 31.000000 | 31.( |
| **Humidity** | 115.0 | 73.852174 | 2.947623 | 70.0 | 71.000000 | 74.000000 | 76.000000 | 79.( |

jupyter Flood_Prediction Last Checkpoint: 5 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [47]:
```
dataset.describe().T
```

Out[47]:

| | count | mean | std | min | 25% | 50% | 75% | |
|---|---|---|---|---|---|---|---|---|
| **Temp** | 115.0 | 29.600000 | 1.122341 | 28.0 | 29.000000 | 30.000000 | 31.000000 | 31.0 |
| **Humidity** | 115.0 | 73.852174 | 2.947623 | 70.0 | 71.000000 | 74.000000 | 76.000000 | 79.0 |
| **Cloud Cover** | 115.0 | 36.286957 | 4.330158 | 30.0 | 32.500000 | 36.000000 | 40.000000 | 44.0 |
| **ANNUAL** | 115.0 | 2925.487826 | 422.112193 | 2068.8 | 2627.900000 | 2937.500000 | 3164.100000 | 4257.8 |
| **Jan-Feb** | 115.0 | 27.739130 | 22.361032 | 0.3 | 10.250000 | 20.500000 | 41.600000 | 98.1 |
| **Mar-May** | 115.0 | 377.253913 | 151.091850 | 89.9 | 276.750000 | 342.000000 | 442.300000 | 915.2 |
| **Jun-Sep** | 115.0 | 2022.840870 | 386.254397 | 1104.3 | 1768.850000 | 1948.700000 | 2242.900000 | 3451.3 |
| **Oct-Dec** | 115.0 | 497.636522 | 129.860643 | 166.6 | 407.450000 | 501.500000 | 584.550000 | 823.3 |
| **avgjune** | 115.0 | 218.100870 | 62.547597 | 65.6 | 179.666667 | 211.033333 | 263.833333 | 366.0 |
| **sub** | 115.0 | 439.801739 | 210.438813 | 34.2 | 295.000000 | 430.600000 | 577.650000 | 982.7 |
| **flood** | 115.0 | 0.139130 | 0.347597 | 0.0 | 0.000000 | 0.000000 | 0.000000 | 1.0 |

In [48]:
```
dataset.head()
```

Out[48]:

| | Temp | Humidity | Cloud Cover | ANNUAL | Jan-Feb | Mar-May | Jun-Sep | Oct-Dec | avgjune | sub | flood |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 29 | 70 | 30 | 3248.6 | 73.4 | 386.2 | 2122.8 | 666.1 | 274.866667 | 649.9 | 0 |
| **1** | 28 | 75 | 40 | 3326.6 | 9.3 | 275.7 | 2403.4 | 638.2 | 130.300000 | 256.4 | 1 |
| **2** | 28 | 75 | 42 | 3271.2 | 21.7 | 336.3 | 2343.0 | 570.1 | 186.200000 | 308.9 | 0 |
| **3** | 29 | 71 | 44 | 3129.7 | 26.7 | 339.4 | 2398.2 | 365.3 | 366.066667 | 862.5 | 0 |
| **4** | 31 | 74 | 40 | 2741.6 | 23.4 | 378.5 | 1881.5 | 458.1 | 283.400000 | 586.9 | 0 |

jupyter Flood_Prediction Last Checkpoint: 5 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3** | 29 | 71 | 44 | 3129.7 | 26.7 | 339.4 | 2398.2 | 365.3 | 366.066667 | 862.5 | 0 |
| **4** | 31 | 74 | 40 | 2741.6 | 23.4 | 378.5 | 1881.5 | 458.1 | 283.400000 | 586.9 | 0 |

## Checking null values

In [49]:
```
dataset.isnull().any()
```

Out[49]:
```
Temp           False
Humidity       False
Cloud Cover    False
ANNUAL         False
Jan-Feb        False
Mar-May        False
Jun-Sep        False
Oct-Dec        False
avgjune        False
sub            False
flood          False
dtype: bool
```

# Independent features

In [51]:
```
x=dataset.iloc[:,2:7].values
```

## Dependent features

In [52]:
```
y=dataset.iloc[:,9:].values
```

jupyter Flood_Prediction Last Checkpoint: 5 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

## Spliting data

In [53]:
```
#split the data into 2 train and test from x and y
#import train_test_split function

from sklearn.model_selection import train_test_split
x_train, x_test,y_train, y_test = train_test_split(x,y,test_size=0.25, random_
```

## Create object

In [54]:
```
#import Standardscaler
from sklearn.preprocessing import StandardScaler

#creare object to Standardscaler class
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

## Dump class

In [55]:
```
#import dump class from joblib
from joblib import dump
dump(sc,"transform.save")
```

Out[55]:
```
['transform.save']
```

## Importing more Libraries

jupyter Flood_Prediction Last Checkpoint: 5 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)

In [57]:
```
from sklearn import tree
from sklearn import ensemble
from sklearn import neighbors
import xgboost
```

In [58]:
```
dtree = tree.DecisionTreeClassifier()
Rf = ensemble.RandomForestClassifier()
knn = neighbors.KNeighborsClassifier()
xgb = xgboost.XGBClassifier()
```

In [59]:
```
from sklearn.preprocessing import MultiLabelBinarizer

mlb = MultiLabelBinarizer()
y_train = mlb.fit_transform(y_train)
```

In [65]:
```
xg_cla = xgboost.XGBClassifier(objective='reg:linear', learning_rate=0.1, max_
```

In [66]:
```
xg_cla.fit(x_train,y_train)
```

```
C:\Users\nalla\anaconda4\Lib\site-packages\xgboost\core.py:160: UserWarning:
[17:24:29] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscali
ng-group-i-0750514818a16474a-1\xgboost\xgboost-ci-windows\src\objective\regre
ssion_obj.cu:209: reg:linear is now deprecated in favor of reg:squarederror.
  warnings.warn(smsg, UserWarning)
```

Out[66]:
```
XGBClassifier
XGBClassifier(alpha=10, base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=No
ne,
              enable_categorical=False, eval_metric=None, feature_types=No
ne,
```

jupyter Flood_Prediction Last Checkpoint: 5 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
e,
             max_cat_threshold=None, max_cat_to_onehot=None,
```

In [67]:
```
dtree.fit(x_train, y_train)
Rf.fit(x_train, y_train)
knn.fit(x_train, y_train)
xgb.fit(x_train, y_train)
```

Out[67]:
```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=No
ne,
              enable_categorical=False, eval_metric=None, feature_types=No
ne,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=No
ne,
              max_cat_threshold=None, max_cat_to_onehot=None,
```

In [68]:
```
from sklearn import metrics
```

In [69]:
```
y_test = ['Cloud Cover', 'ANNUAL', 'Jan-Feb', 'Mar-May', 'Jun-Sep']
p1 = ['Cloud Cover', 'Annual Rainfall', 'Jan-Feb Rainfall', 'Mar-May Rainfall'
```

In [72]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

# Read the dataset
```

jupyter Flood_Prediction Last Checkpoint: 5 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
from sklearn import metrics

# Read the dataset
data = pd.read_csv("Flood.csv")  # Replace with the actual file path

# Extract the true labels (y_test) from the "flood" column
y_test = data["flood"]

# Extract the features used for prediction
x = data.drop(columns=["flood"])  # Assuming "flood" is the target variable

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y_test, test_size=0.2,

# Create and train a logistic regression model
model = LogisticRegression()
model.fit(x_train, y_train)

# Make predictions with the model for p1, p2, p3, and p4
p1 = model.predict(x_test)
p2 = model.predict(x_test)
p3 = model.predict(x_test)
p4 = model.predict(x_test)

# Calculate the accuracy for each set of predictions
accuracy_p1 = metrics.accuracy_score(y_test, p1)
accuracy_p2 = metrics.accuracy_score(y_test, p2)
accuracy_p3 = metrics.accuracy_score(y_test, p3)
accuracy_p4 = metrics.accuracy_score(y_test, p4)

print("Accuracy for p1:", accuracy_p1)
print("Accuracy for p2:", accuracy_p2)
print("Accuracy for p3:", accuracy_p3)
print("Accuracy for p4:", accuracy_p4)
```

```
Accuracy for p1: 0.9565217391304348
```

jupyter Flood_Prediction Last Checkpoint: 5 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
Accuracy for p1: 0.9565217391304348
Accuracy for p2: 0.9565217391304348
Accuracy for p3: 0.9565217391304348
Accuracy for p4: 0.9565217391304348
```

```
C:\Users\nalla\anaconda4\Lib\site-packages\sklearn\linear_model\_logistic.py:
460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  n_iter_i = _check_optimize_result(
```

In [73]:
```
metrics.confusion_matrix(y_test,p4)
```

Out[73]:
```
array([[20,  0],
       [ 1,  2]], dtype=int64)
```

In [74]:
```
print(metrics.accuracy_score(y_test,p4))
```

```
0.9565217391304348
```

In [75]:
```
print(metrics.precision_score(y_test,p4))
```

```
1.0
```

In [76]:
```
print(metrics.recall_score(y_test,p4))
```

```
0.6666666666666666
```

In [77]:
```
from joblib import dump
```

jupyter Flood_Prediction Last Checkpoint: 5 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  n_iter_i = _check_optimize_result(
```

```
In [73]: metrics.confusion_matrix(y_test,p4)

Out[73]: array([[20,  0],
                [ 1,  2]], dtype=int64)
```

```
In [74]: print(metrics.accuracy_score(y_test,p4))

0.9565217391304348
```

```
In [75]: print(metrics.precision_score(y_test,p4))

1.0
```

```
In [76]: print(metrics.recall_score(y_test,p4))

0.6666666666666666
```

```
In [77]: from joblib import dump
         dump(xg_cla,'floods.save')

Out[77]: ['floods.save']
```