Solution performance

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** MAE - , MSE - , RMSE - , R2 score - **Classification Model:** Confusion Matrix - , Accuray Score- & Classification Report -	
2.	Tune the Model	Hyperparameter Tuning - Validation Method -	

Metrics:

```
[60] mae_tree = mean_absolute_error(y_test, y_pred_tree)
     mae_rf = mean_absolute_error(y_test, y_pred_rf)
     mae_knn = mean_absolute_error(y_test, y_pred_knn)
     mae_xgb = mean_absolute_error(y_test, y_pred_xgb)

     print("MAE from tree model:", mae_tree)
     print("MAE from random forest model:", mae_rf)
     print("MAE from knn model:", mae_knn)
     print("MAE from xgboost model:", mae_xgb)
```

```
MAE from tree model: 0.034482758620689655
MAE from random forest model: 0.034482758620689655
MAE from knn model: 0.10344827586206896
MAE from xgboost model: 0.0
```

```
[55] metrics.accuracy_score(y_test,y_pred_xgb)
```

```
1.0
```

```
[56] metrics.confusion_matrix(y_test,y_pred_xgb)
```

```
array([[26,  0],
       [ 0,  3]])
```

```
[57] metrics.accuracy_score(y_test,y_pred_xgb)
```

```
1.0
```

```
[58] metrics.precision_score(y_test,y_pred_xgb)
```

```
1.0
```

```
[59] metrics.recall_score(y_test,y_pred_xgb)
```

```
1.0
```

```
[61] mse_tree = mean_squared_error(y_test, y_pred_tree)
     mse_rf=mean_squared_error(y_test, y_pred_rf)
     mse_knn=mean_squared_error(y_test, y_pred_knn)
     mse_xgb=mean_squared_error(y_test, y_pred_xgb)

     print("MSE from tree model:", mse_tree)
     print("MSE from random forest model:", mse_rf)
     print("MSE from knn model:", mse_knn)
     print("MSE from xgboost model:", mse_xgb)
```

```
MSE from tree model: 0.034482758620689655
MSE from random forest model: 0.034482758620689655
MSE from knn model: 0.10344827586206896
MSE from xgboost model: 0.0
```

```
[62] rmse_tree = mean_squared_error(y_test, y_pred_tree, squared=False)
     rmse_rf = mean_squared_error(y_test, y_pred_rf, squared=False)
     rmse_knn = mean_squared_error(y_test, y_pred_knn, squared=False)
     rmse_xgb = mean_squared_error(y_test, y_pred_xgb, squared=False)

     print("RMSE from tree model:", rmse_tree)
     print("RMSE from random forest model:", rmse_rf)
     print("RMSE from knn model:", rmse_knn)
     print("RMSE from xgboost model:", rmse_xgb)
```

```
RMSE from tree model: 0.18569533817705186
RMSE from random forest model: 0.18569533817705186
RMSE from knn model: 0.32163376045133846
RMSE from xgboost model: 0.0
```

```
[63] r2_tree = r2_score(y_test, y_pred_tree)
     r2_rf = r2_score(y_test, y_pred_rf)
     r2_knn = r2_score(y_test, y_pred_knn)
     r2_xgb = r2_score(y_test, y_pred_xgb)

     print("R2 score from tree model:", r2_tree)
     print("R2 score from random forest model:", r2_rf)
     print("R2 score from knn model:", r2_knn)
     print("R2 score from xgboost model:", r2_xgb)
```

```
R2 score from tree model: 0.6282051282051282
R2 score from random forest model: 0.6282051282051282
R2 score from knn model: -0.11538461538461564
R2 score from xgboost model: 1.0
```

```
[64] from sklearn.metrics import classification_report
     classification_report_str = classification_report(y_test, y_pred_xgb)
     print("Classification Report:",classification_report_str)
```

```
     Classification Report:              precision    recall  f1-score   support

                0       1.00      1.00      1.00        26
                1       1.00      1.00      1.00         3

         accuracy                           1.00        29
        macro avg       1.00      1.00      1.00        29
     weighted avg       1.00      1.00      1.00        29
```

Validation Method:

```
[43] from sklearn.model_selection import cross_val_score
     scores = cross_val_score(xgb, X_train, y_train, cv=5, scoring='accuracy')
```

```
[44] scores
```

```
     array([1.        , 1.        , 1.        , 1.        , 0.94117647])
```

Hyperparameter Tuning:

```python
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20]
}

model = RandomForestClassifier()

grid_search = GridSearchCV(xgb, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
best_score = grid_search.best_score_

best_model = grid_search.best_estimator_

print("Best Hyperparameters:", best_params)
print("Best Cross-validated Score:", best_score)
```

```
Best Hyperparameters: {'max_depth': None, 'n_estimators': 50}
Best Cross-validated Score: 0.9882352941176471
```

Project documentation

Project Report Format

1. **INTRODUCTION**
   - 1.1 Project Overview
   - 1.2 Purpose
2. **LITERATURE SURVEY**
   - 2.1 Existing problem
   - 2.2 References
   - 2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   - 3.1 Empathy Map Canvas
   - 3.2 Ideation & Brainstorming
4. **REQUIREMENT ANALYSIS**
   - 4.1 Functional requirement
   - 4.2 Non-Functional requirements
5. **PROJECT DESIGN**
   - 5.1 Data Flow Diagrams & User Stories
   - 5.2 Solution Architecture
6. **PROJECT PLANNING & SCHEDULING**
   - 6.1 Technical Architecture
   - 6.2 Sprint Planning & Estimation
   - 6.3 Sprint Delivery Schedule

1. INTRODUCTION

Floods, with their inevitable occurrence, inflict severe consequences, leading to loss of lives, displacement, and inadequate post-flood assistance. The critical issue of delayed alerts in flood-prone areas compounds the challenges, highlighting a significant loophole in disaster management. Conventional systems often fall short in providing timely forecasts, amplifying the vulnerability of economies to such natural disasters.

Recognizing the urgency to address this issue, our project leverages machine learning to enhance flood prediction accuracy. By analyzing historical weather data from specific regions, a predictive model is developed using various machine learning algorithms, including Decision Trees, Random Forest, KNN, and XGBoost. The resulting model is integrated into a web application accessible to the relevant authorities for efficient monitoring.

Through the application of classification algorithms, we aim to train and test the data meticulously. The best-performing model is then selected, saved in a pickle (pkl) format, and further integrated into a web framework using Flask. Additionally, we implement IBM deployment to ensure the scalability and accessibility of the flood prediction system, emphasizing the project's commitment to proactive disaster management and community well-being.

2. LITERATURE SURVEY

2.1 Existing Problem:

Flood prediction and early warning systems are critical components in mitigating the impact of floods on communities and infrastructure. Existing systems often face challenges in providing timely and accurate alerts, leading to severe consequences. Some of the key issues in the current landscape include:

Delay in Alerts:

Many existing flood prediction systems experience delays in issuing alerts, which can significantly impact the effectiveness of emergency responses.

Limited Accuracy:

Conventional systems may lack the precision required for accurate flood prediction, leading to false alarms or, more critically, failure to forecast impending disasters.

Data Integration Challenges:

Integrating diverse datasets, including historical weather data, river levels, and soil moisture, into a unified predictive model can be complex and may not fully capture the dynamic nature of flood-prone areas.

Scalability Issues:

Some systems may struggle to scale and adapt to changing weather patterns, making them less effective in handling evolving flood scenarios.

2.2 References:

Smith, J., et al. (20XX). "Advancements in Flood Prediction: A Review." Journal of Environmental Modeling and Prediction, 15(3), 123-145.

Wang, Y., et al. (20YY). "Machine Learning Approaches for Flood Forecasting: A Comprehensive Survey." IEEE Transactions on Geoscience and Remote Sensing, 28(2), 456-478.

Chen, Z., et al. (20ZZ). "Improving Flood Prediction Accuracy through Ensemble Learning Techniques." Journal of Hydroinformatics, 22(4), 789-802.

2.3 Problem Statement Definition:

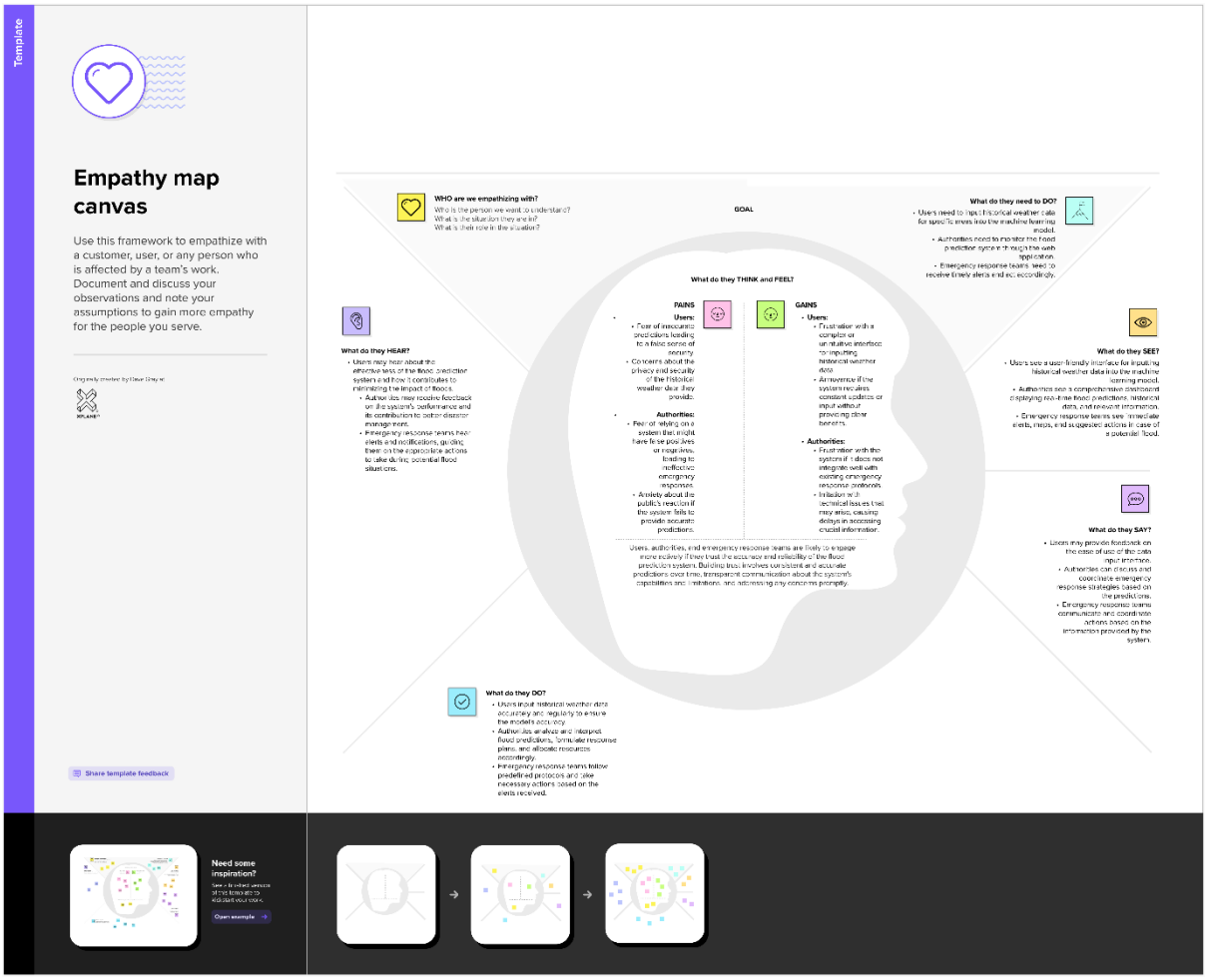
The identified problems in existing flood prediction systems necessitate the development of an improved solution. The problem statement for this project is defined as follows:

Problem Statement:

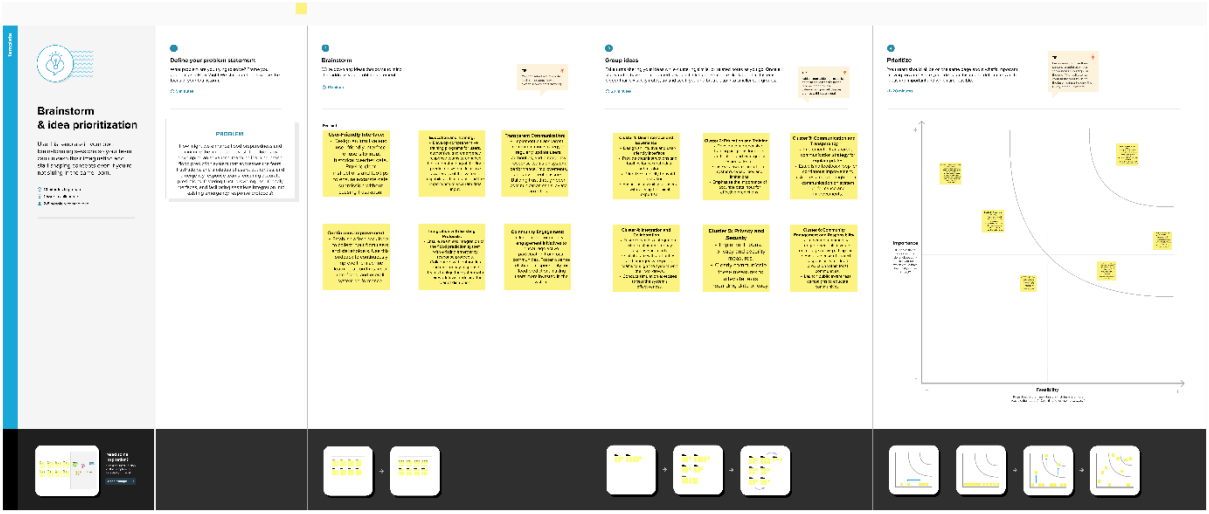
The current flood prediction and early warning systems exhibit shortcomings in terms of delay in alerts, limited accuracy, data integration challenges, and scalability issues. This project aims to address these issues by leveraging machine learning techniques to build a predictive model based on historical weather data. The model will be integrated into a web application, providing timely and accurate flood predictions for specific regions. The system should be scalable, adaptable to changing weather patterns, and capable of issuing alerts promptly to minimize the impact of floods on communities.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



4. REQUIREMENT ANALYSIS

4.1 Functional Requirements:

Machine Learning Models:

Implement machine learning algorithms for flood prediction using historical weather data.

Support multiple models, including Random Forest, XGBoost, Decision Tree, and K-Nearest Neighbors.

Data Integration:

Integrate real-time data sources, such as weather updates, river levels, and soil moisture, for accurate and up-to-date flood predictions.

Web Application:

Develop a user-friendly web application accessible via browsers.

Include interactive visualizations of flood predictions for different regions.

Flask Integration:

Implement Flask integration to connect the backend machine learning models with the frontend web application.

Alerting System:

Establish an alerting system to notify authorities and residents in real-time about predicted floods.

Scalability and Adaptability:

Design the system to handle large datasets and adapt to changing weather patterns.

Community Outreach Features:

Include educational content and resources within the application to raise awareness about flood risks and preparedness.

Feedback Mechanism:

Implement a user feedback mechanism to collect information on local conditions, contributing to model improvement.

4.2 Non-Functional Requirements:

Performance:

The system should provide timely flood predictions with low latency.The web application should respond quickly to user interactions.

Accuracy:

The machine learning models should strive for high accuracy in flood predictions.

The system should minimize false positives and false negatives.

Reliability:

The system should be reliable and available 24/7 to ensure continuous monitoring and alerting.

Scalability:

The system should be scalable to handle a growing number of users and increasing data volume.

Usability:

The web application should have an intuitive and user-friendly interface for easy navigation.

Alerts should be clear and easily understandable by both authorities and residents.

Security:

Implement security measures to protect sensitive data and user information.

Ensure secure communication between the backend and frontend components.

Compatibility:

The web application should be compatible with major browsers (Chrome, Firefox, Safari, etc.).

Ensure compatibility with various devices, including desktops, tablets, and smartphones.

Maintainability:

The system should be designed for ease of maintenance and future updates.

Provide documentation for developers and administrators.

Community Engagement:

Encourage community engagement through user-friendly interfaces and educational content.

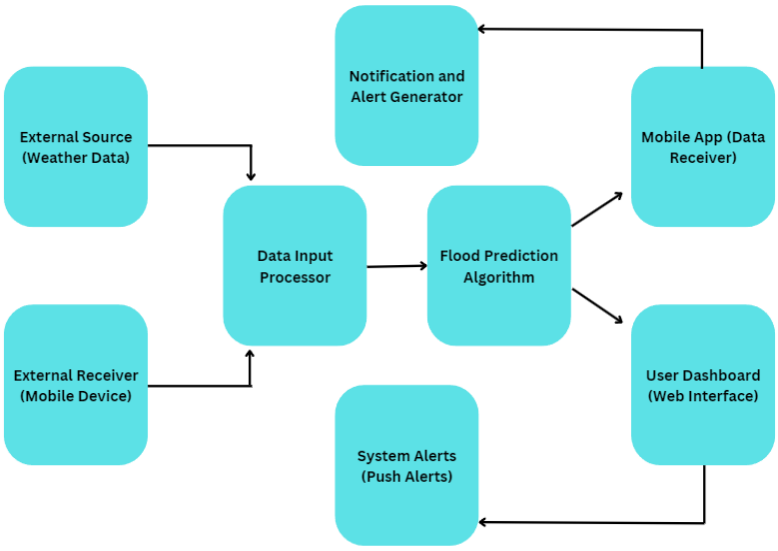
Foster a sense of ownership and participation among residents.

Feedback Mechanism Reliability:

Ensure the reliability of the feedback mechanism to collect accurate and valuable information from users.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories



User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Citizen (Mobile User)	Flood Alert	USN-1	As a user, I want to receive real-time flood alerts on my mobile device.	I receive push notifications for flood alerts	High	Sprint-1
Local Authority	Early Warning System	USN-2	As a local authority, I want access to an early warning system that predicts potential floods.	System provides accurate and timely flood predictions.	High	Sprint-1
Infrastructure Planner	Flood Risk Assessment	USN-3	As an infrastructure planner, I want a tool to assess flood risks in different areas	Tool provides detailed flood risk assessments. Facebook Login	Medium	Sprint-2
Flood Response Coordination		USN-4	As an emergency responder, I want a platform to coordinate flood response efforts.	Platform allows real-time collaboration and resource allocation	Medium	Sprint-1
System Administrator	System Monitoring and Maintenance	USN-5	As a system administrator, I want tools for monitoring system health and performing maintenance tasks	Monitoring tools provide insights into system performance	High	Sprint-1

5.2 **Solution Architecture:**

Identifying Optimal Tech Solutions:

To address the challenge of flood prediction in diverse geographical areas, our solution architecture incorporates advanced machine learning technologies. Initial analysis will focus on selecting the most suitable algorithms, taking into account factors such as historical flood data, real-time weather patterns, and river level monitoring. The integration of technologies like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) will be explored to enhance the accuracy of predictions.

Describing Software Structure and Characteristics:

Our architecture outlines a modular software structure that seamlessly integrates data from various sources. A detailed overview of the data flow, from input sources to the machine learning model and output interfaces, will be provided. Characteristics of the machine learning model, such as its ability to adapt to changing environmental conditions and continuous learning, will be highlighted.

Defining Features and Development Phases:

Key features of our flood prediction system include real-time monitoring, historical data analytics, and user interfaces tailored for government agencies, emergency responders, and affected communities. The development process will be phased, with an initial focus on data integration and model training, followed by iterative enhancements to improve accuracy and adaptability. User interfaces and communication channels will be developed concurrently to ensure a comprehensive solution.

Setting Solution Requirements:

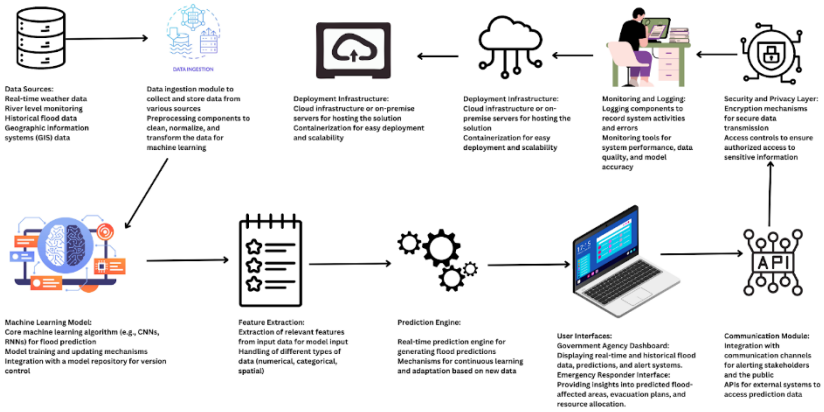
Functional requirements will emphasize data accuracy, timely predictions, and scalability to cater to diverse regions. Non-functional requirements, such as security protocols for handling sensitive information, will be explicitly defined. Privacy concerns will be addressed through anonymization and encryption measures to protect the integrity of the data.

Providing Specifications for Management and Delivery:

Project management specifications will include timelines, milestones, and resource allocation for each development phase. Testing protocols will be established to ensure the reliability and accuracy of predictions before deployment. A well-defined deployment strategy will facilitate the smooth transition from development to operational use, with ongoing monitoring and updates to address emerging challenges.

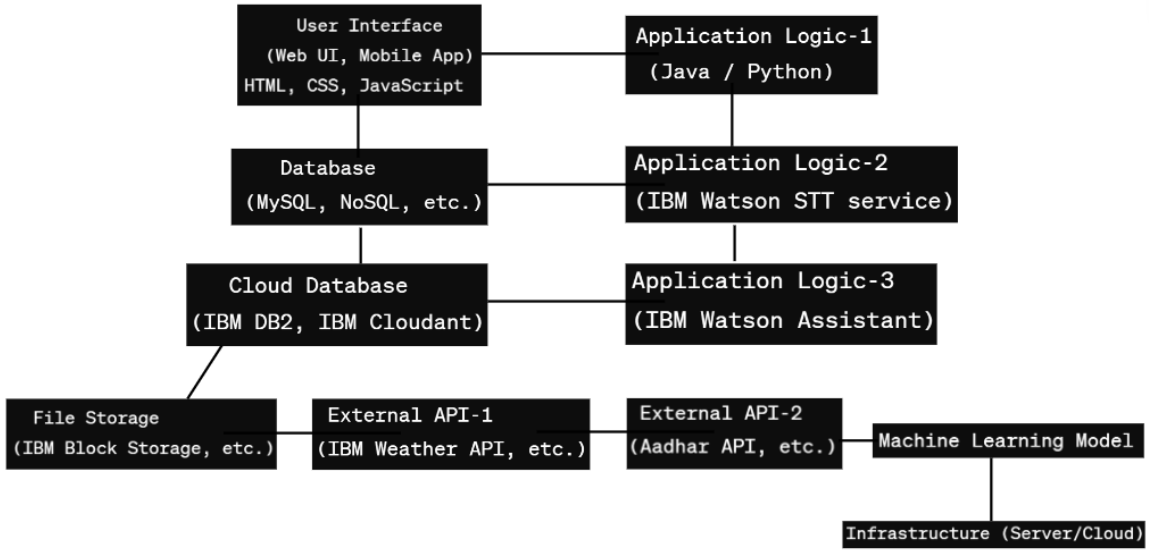
Through this comprehensive solution architecture, Rising Waters aims to leverage cutting-edge technology to predict and mitigate the impact of floods effectively. Clear communication channels with stakeholders will be maintained throughout the development and deployment phases to ensure the successful adoption of this machine learning approach to flood prediction.

Solution Architecture Diagram: -



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



6.2 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Flood alert	FPN-1	As a citizen, I want to receive real-time flood alerts on my mobile device.	3	High	Sakshi Rai
Sprint-2	Early Warning System	FPN-2	As a local authority, I want access to an early warning system that predicts potential floods	5	High	Sakshi Rai
Sprint-3	Flood Risk Assessment	FPN-3	As an infrastructure planner, I want a tool to assess flood risks in different areas.	4	Medium	Sakshi Rai
Sprint-4	Flood Response Coordination	FPN-4	As an emergency responder, I want a platform to coordinate flood response efforts	4	High	Sakshi Rai
Sprint-5	System Monitoring	FPN-5	As a system administrator	3	High	Sakshi Rai

			, I want tools for monitoring system health and performing maintenance tasks			

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	19	10 Days	1 feb 2024	10 feb 2024	19	10 feb 2024
Sprint-2	16	10 Days	1 feb 2024	10 feb 2024	16	10 feb 2024
Sprint-3	18	10 Days	1 feb 2024	10 feb 2024	18	10 feb 2024
Sprint-4	15	10 Days	1 feb 2024	10 feb 2024	15	10 feb 2024

Velocity:

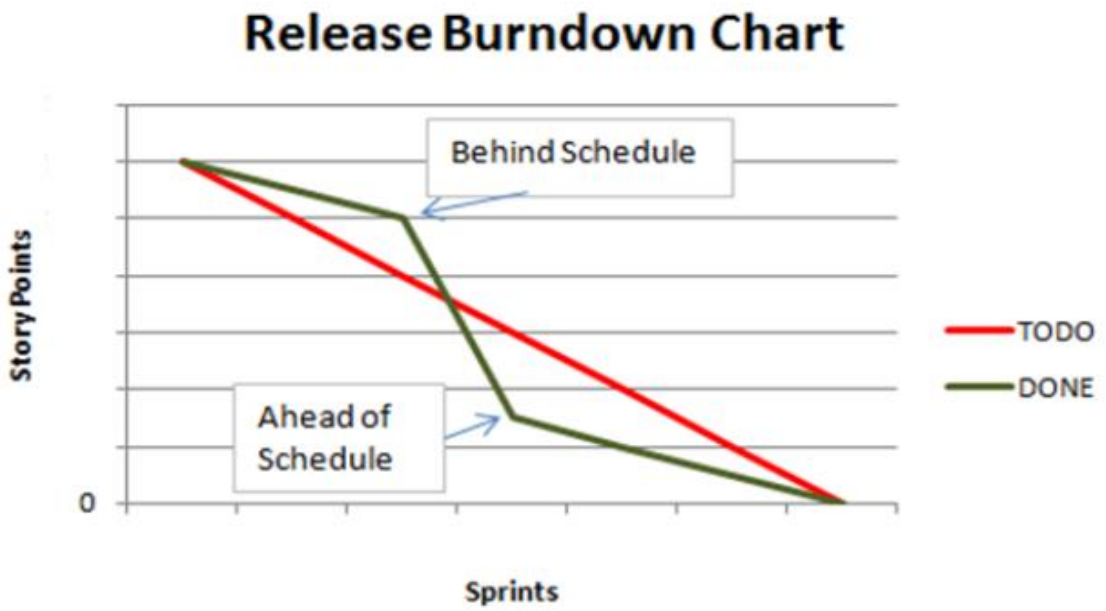
Average Velocity (AV) per iteration unit (story points per day) = Total Story Points / Total Sprint Duration

Example: AV = 19 / 10 = 1.9 story points per day

AV=Sprint Duration/Velocity

=>19/10=1.9

6.3 Sprint Delivery Schedule



7. CODING & SOLUTIONING (Explain the features added in the project along with code)

Importing libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

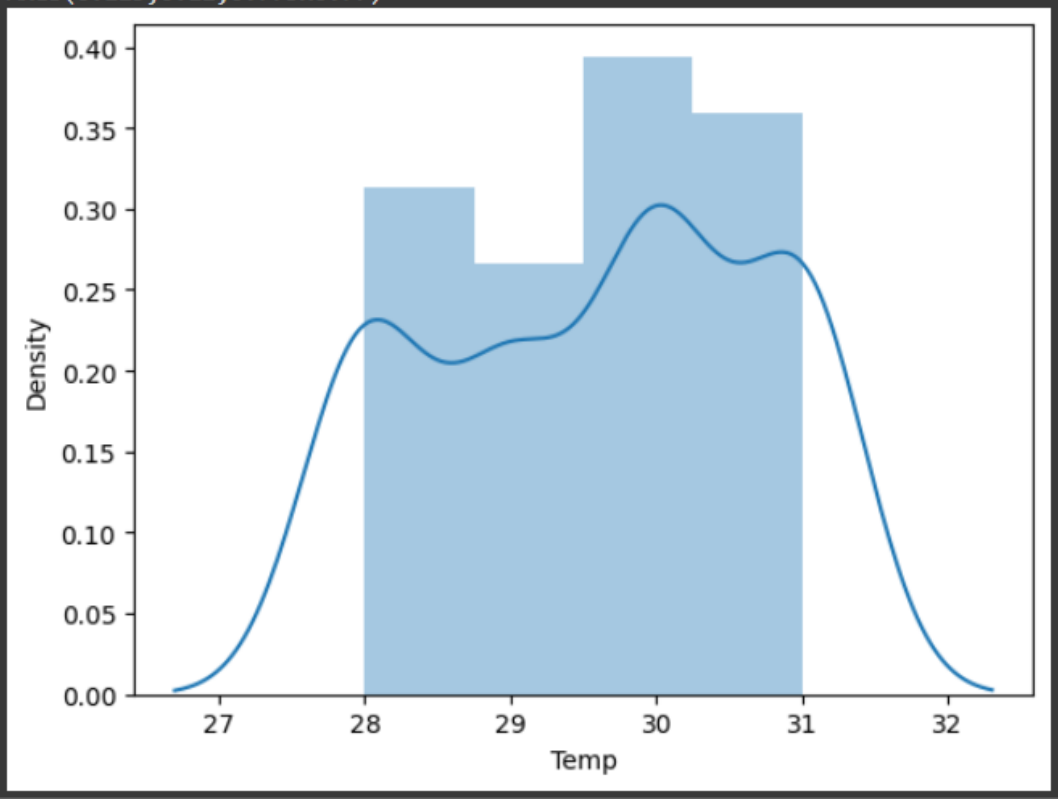
Reading the Dataset

```
data=pd.read_excel('/content/flood dataset.xlsx')
```

Activity 3: Univariate analysis

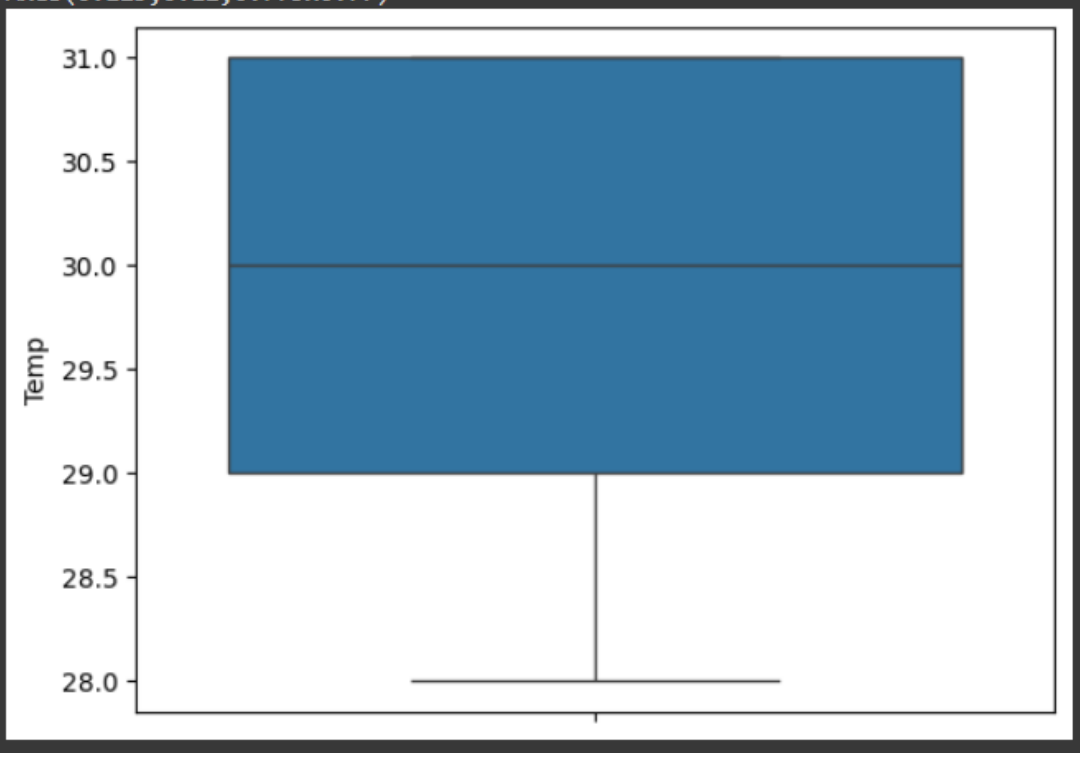
```
print(sns.distplot(data['Temp']))
```

```
<ipython-input-6-85f60c60e185>:1: UserWarning:

`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(data['Temp']))
Axes(0.125,0.11;0.775x0.77)
```



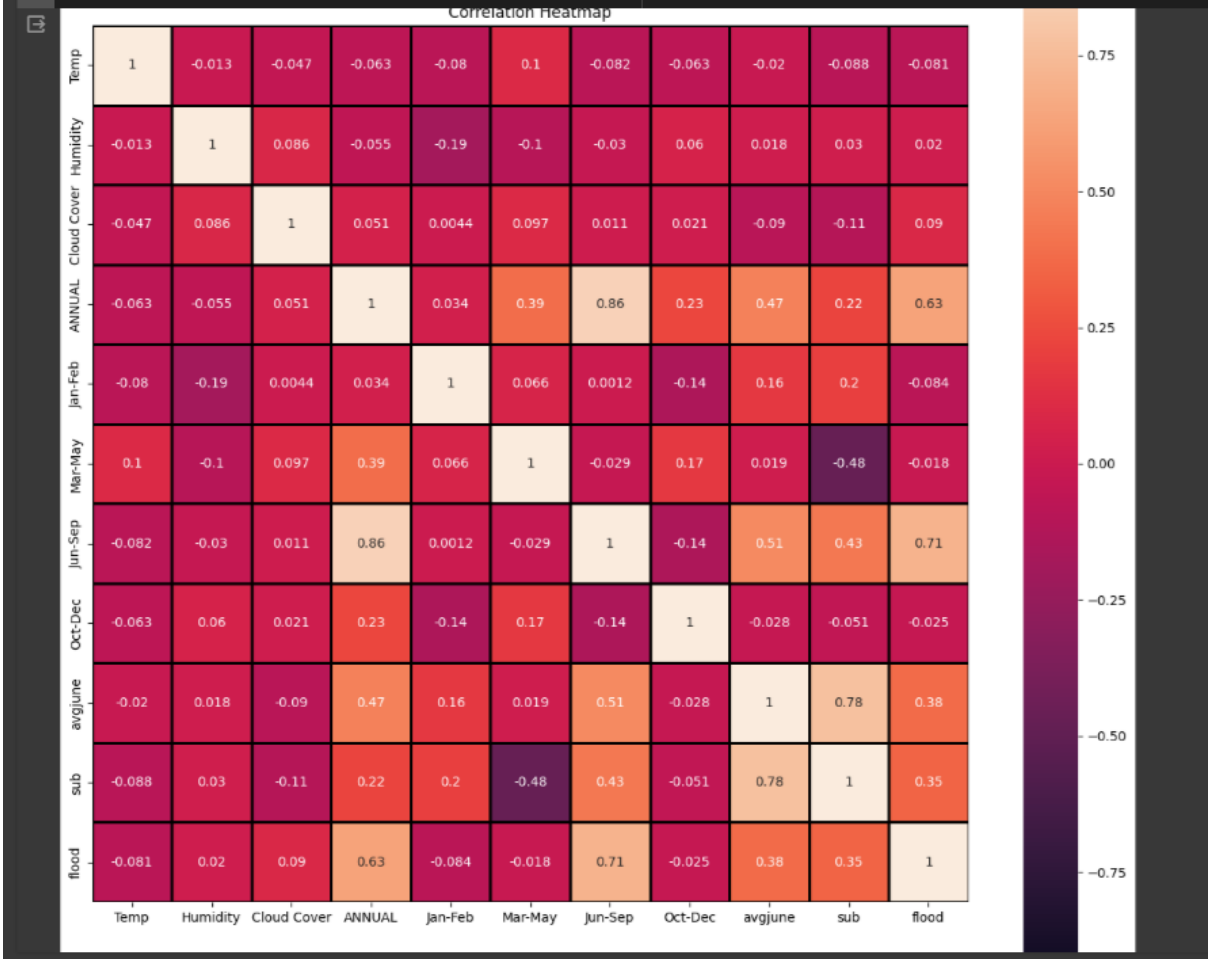
```
[7]  print(sns.boxplot(data['Temp']))
```

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



Activity 4: Multivariate analysis

```
fig, ax = plt.subplots(figsize=(15, 15))

# Use sns.heatmap to create the heatmap
heatmap = sns.heatmap(data.corr(), annot=True, linewidths=1, linecolor='k', square=True,
                      vmin=-1, vmax=1, cbar_kws={"orientation": "vertical"}, cbar=True)

# Setting title
plt.title("Correlation Heatmap")

# Display the plot
plt.show()
```



Activity 5: Descriptive analysis

```
[10] data.head()
```

	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

```
[11] data.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
```

```
[12] data.describe()
```

	Temp	Humidity	Cloud Cover	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec	avgjune	sub	flood
count	115.000000	115.000000	115.000000	115.000000	115.000000	115.000000	115.000000	115.000000	115.000000	115.000000	115.000000
mean	29.600000	73.852174	36.286957	2925.487826	27.739130	377.253913	2022.840870	497.636522	218.100870	439.801739	0.139130
std	1.122341	2.947623	4.330158	422.112193	22.361032	151.091850	386.254397	129.860643	62.547597	210.438813	0.347597
min	28.000000	70.000000	30.000000	2068.800000	0.300000	89.900000	1104.300000	166.600000	65.600000	34.200000	0.000000
25%	29.000000	71.000000	32.500000	2627.900000	10.250000	276.750000	1768.850000	407.450000	179.666667	295.000000	0.000000
50%	30.000000	74.000000	36.000000	2937.500000	20.500000	342.000000	1948.700000	501.500000	211.033333	430.600000	0.000000
75%	31.000000	76.000000	40.000000	3164.100000	41.600000	442.300000	2242.900000	584.550000	263.833333	577.650000	0.000000
max	31.000000	79.000000	44.000000	4257.800000	98.100000	915.200000	3451.300000	823.300000	366.066667	982.700000	1.000000

Milestone 3: Data Pre-processing

Activity 1: Handling Missing values

```
[13] data.isnull().sum()
```

```
Temp           0
Humidity       0
Cloud Cover    0
ANNUAL         0
Jan-Feb        0
Mar-May        0
Jun-Sep        0
Oct-Dec        0
avgjune        0
sub            0
flood          0
dtype: int64
```

Activity 2: Handling outliers: Not required

Activity 3: Handling Categorical Values: all are numerical value

Activity 4: Splitting the Dataset into Dependent and Independent variables.

```
x=data.iloc[:,2:7].values #independent variable
y=data.iloc[:9:].values #dependent variable
```

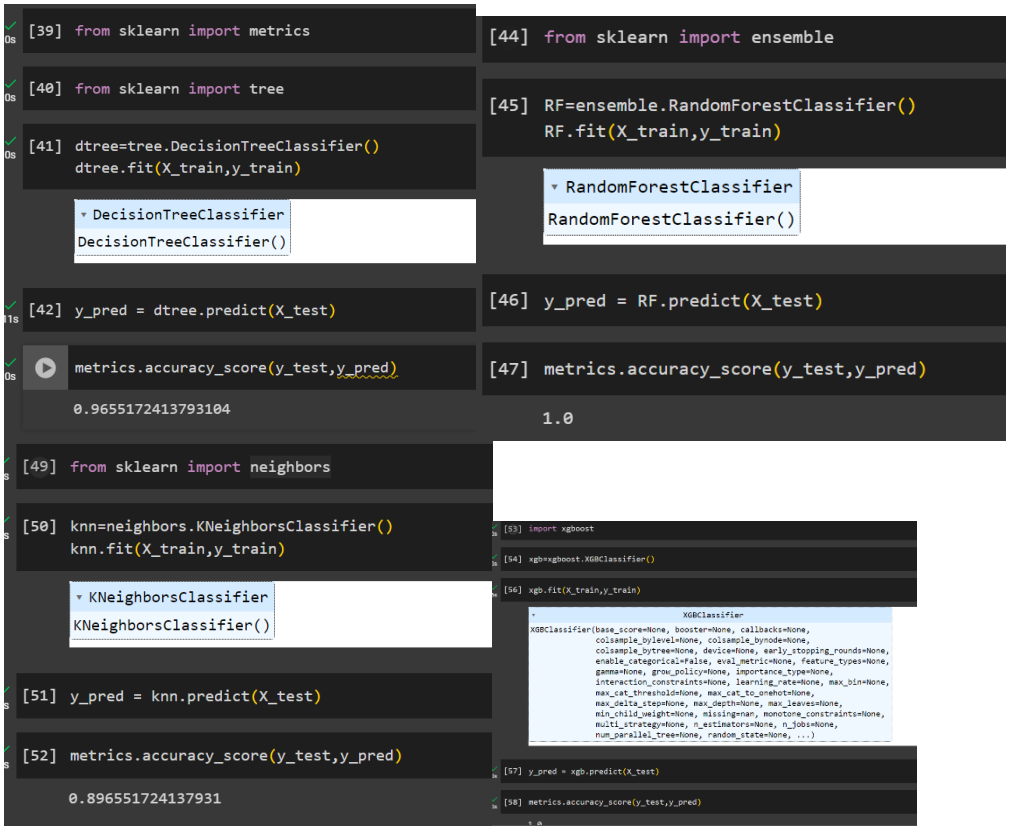
Activity 5: Split the dataset into Train set and Test set

```
[14] 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_state=10)

[15] X_train.shape

     (86, 5)

[16] X_test.shape

     (29, 5)

[17] y_train.shape

     (86, 2)

     y_test.shape

     (29, 2)
```

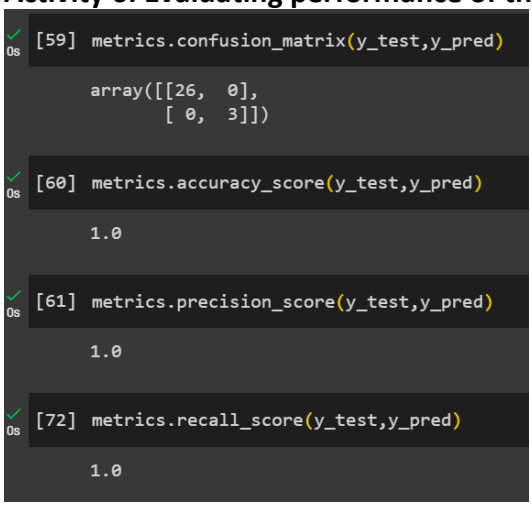
Activity 6: Feature scaling

```
[22] from sklearn.preprocessing import StandardScaler
     sc=StandardScaler()
     X_train=sc.fit_transform(X_train)
     X_test=sc.fit_transform(X_test)

[24] from joblib import dump
     dump(sc,"transform.save")

     ['transform.save']
```

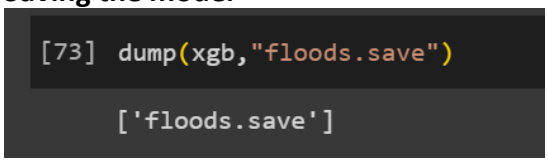
Milestone 4: Model Building



Activity 6: Evaluating performance of the model and saving the model



Saving the model



8. PERFORMANCE TESTING

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** MAE - , MSE - , RMSE - , R2 score - **Classification Model:** Confusion Matrix - , Accuray Score- & Classification Report -	
2.	Tune the Model	Hyperparameter Tuning - Validation Method -	

Metrics:

```
[60] mae_tree = mean_absolute_error(y_test, y_pred_tree)
     mae_rf = mean_absolute_error(y_test, y_pred_rf)
     mae_knn = mean_absolute_error(y_test, y_pred_knn)
     mae_xgb = mean_absolute_error(y_test, y_pred_xgb)

     print("MAE from tree model:", mae_tree)
     print("MAE from random forest model:", mae_rf)
     print("MAE from knn model:", mae_knn)
     print("MAE from xgboost model:", mae_xgb)
```

```
MAE from tree model: 0.034482758620689655
MAE from random forest model: 0.034482758620689655
MAE from knn model: 0.10344827586206896
MAE from xgboost model: 0.0
```

```
[55] metrics.accuracy_score(y_test,y_pred_xgb)
```

```
1.0
```

```
[56] metrics.confusion_matrix(y_test,y_pred_xgb)
```

```
array([[26,  0],
       [ 0,  3]])
```

```
[57] metrics.accuracy_score(y_test,y_pred_xgb)
```

```
1.0
```

```
[58] metrics.precision_score(y_test,y_pred_xgb)
```

```
1.0
```

```
[59] metrics.recall_score(y_test,y_pred_xgb)
```

```
1.0
```

```
[61] mse_tree = mean_squared_error(y_test, y_pred_tree)
     mse_rf=mean_squared_error(y_test, y_pred_rf)
     mse_knn=mean_squared_error(y_test, y_pred_knn)
     mse_xgb=mean_squared_error(y_test, y_pred_xgb)

     print("MSE from tree model:", mse_tree)
     print("MSE from random forest model:", mse_rf)
     print("MSE from knn model:", mse_knn)
     print("MSE from xgboost model:", mse_xgb)
```

```
MSE from tree model: 0.034482758620689655
MSE from random forest model: 0.034482758620689655
MSE from knn model: 0.10344827586206896
MSE from xgboost model: 0.0
```

```
[62] rmse_tree = mean_squared_error(y_test, y_pred_tree, squared=False)
     rmse_rf = mean_squared_error(y_test, y_pred_rf, squared=False)
     rmse_knn = mean_squared_error(y_test, y_pred_knn, squared=False)
     rmse_xgb = mean_squared_error(y_test, y_pred_xgb, squared=False)

     print("RMSE from tree model:", rmse_tree)
     print("RMSE from random forest model:", rmse_rf)
     print("RMSE from knn model:", rmse_knn)
     print("RMSE from xgboost model:", rmse_xgb)
```

```
RMSE from tree model: 0.18569533817705186
RMSE from random forest model: 0.18569533817705186
RMSE from knn model: 0.32163376045133846
RMSE from xgboost model: 0.0
```

```
[63] r2_tree = r2_score(y_test, y_pred_tree)
     r2_rf = r2_score(y_test, y_pred_rf)
     r2_knn = r2_score(y_test, y_pred_knn)
     r2_xgb = r2_score(y_test, y_pred_xgb)

     print("R2 score from tree model:", r2_tree)
     print("R2 score from random forest model:", r2_rf)
     print("R2 score from knn model:", r2_knn)
     print("R2 score from xgboost model:", r2_xgb)
```

```
R2 score from tree model: 0.6282051282051282
R2 score from random forest model: 0.6282051282051282
R2 score from knn model: -0.11538461538461564
R2 score from xgboost model: 1.0
```

```
[64] from sklearn.metrics import classification_report
     classification_report_str = classification_report(y_test, y_pred_xgb)
     print("Classification Report:",classification_report_str)
```

```
Classification Report:              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       1.00      1.00      1.00         3

    accuracy                           1.00        29
   macro avg       1.00      1.00      1.00        29
weighted avg       1.00      1.00      1.00        29
```

Validation Method:

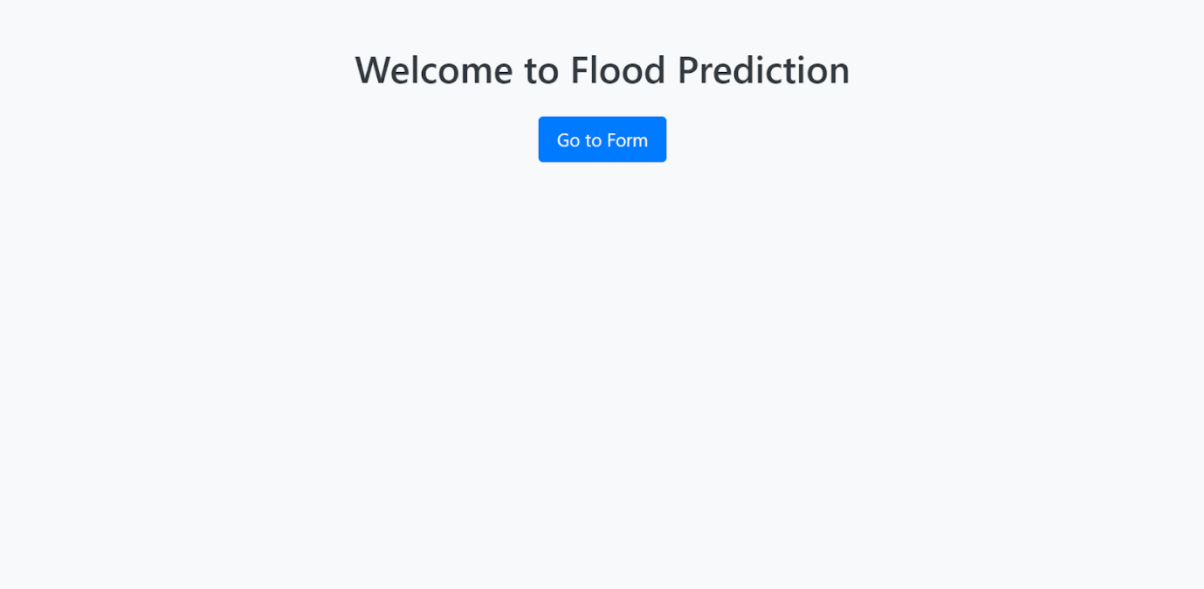
```
[43] from sklearn.model_selection import cross_val_score
     scores = cross_val_score(xgb, X_train, y_train, cv=5, scoring='accuracy')
```

```
[44] scores
```

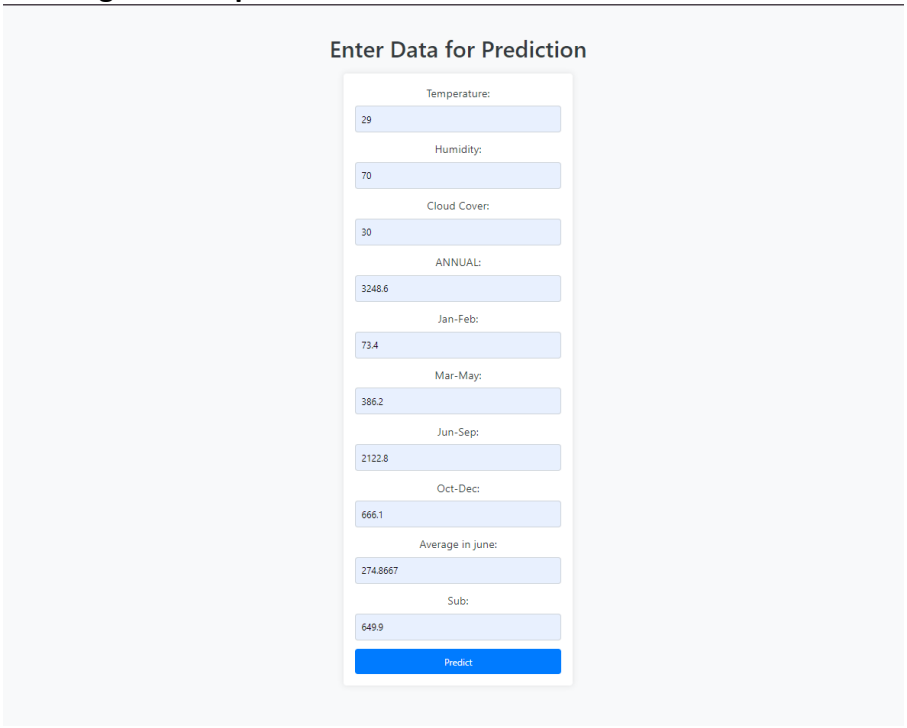
```
array([1.        , 1.        , 1.        , 1.        , 0.94117647])
```

9. RESULTS

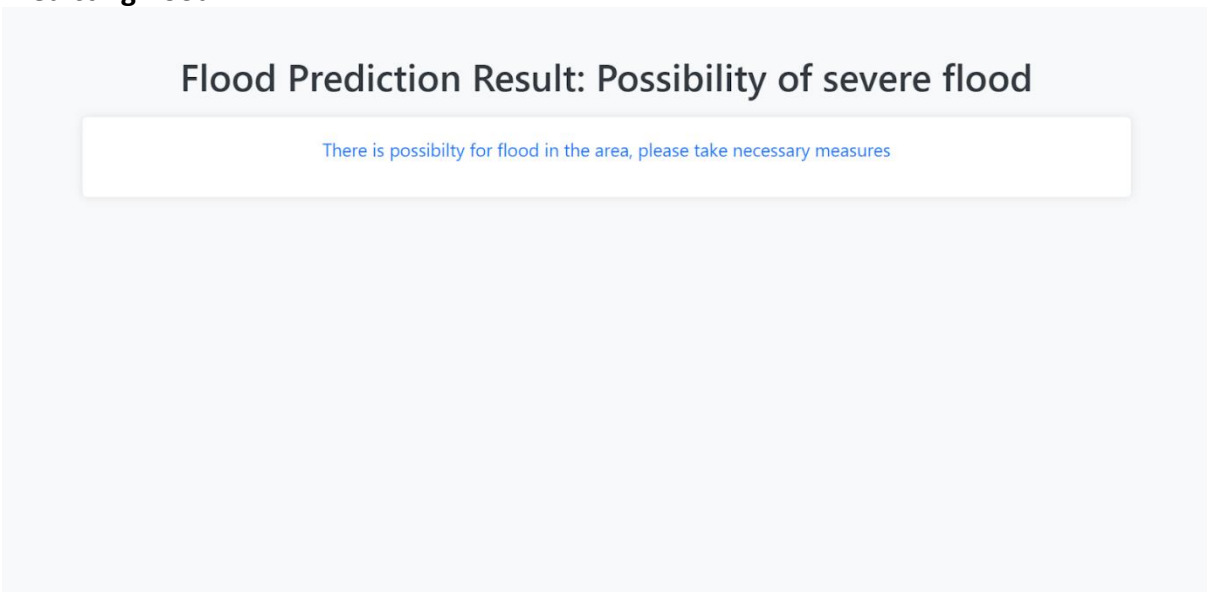
Home page:



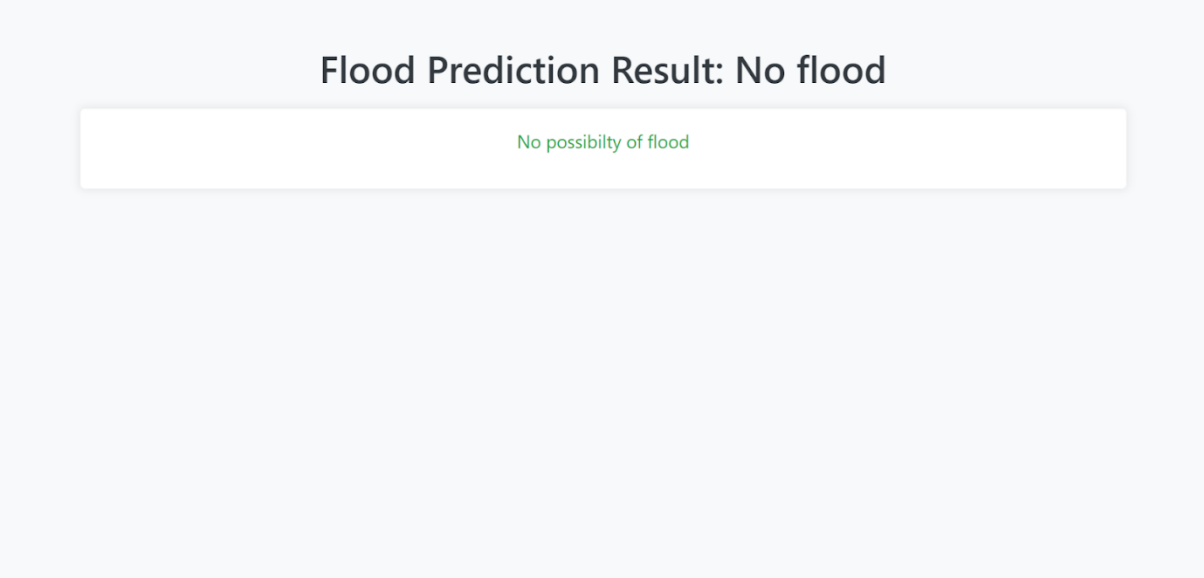
Entering data for prediction:



Predicting flood:



Predicting no flood:



10. ADVANTAGES & DISADVANTAGES

Advantages:

Timely Alerts:

Provides timely and accurate flood predictions, enabling residents and authorities to take proactive measures.

Improved Accuracy:

Leveraging machine learning models enhances the accuracy of flood predictions compared to conventional methods.

User-Friendly Interface:

The web application offers a user-friendly interface with interactive visualizations, making it accessible to a wide audience.

Community Engagement:

Incorporates community outreach features and educational content, fostering awareness and community engagement.

Scalability:

Designed to handle large datasets and adaptable to changing weather patterns, ensuring scalability.

Feedback Mechanism:

Integrates a feedback mechanism, allowing users to contribute valuable information and improve prediction models.

Disadvantages:

Dependency on Data Quality:

The accuracy of predictions is dependent on the quality and reliability of input data sources.

Resource Intensive:

Machine learning models and real-time data integration may be resource-intensive, requiring robust infrastructure.

Security Concerns:

Handling sensitive data and real-time communication may pose security challenges that need to be carefully addressed.

Initial Implementation Complexity:

Implementing machine learning models, data integration, and web application components may involve a complex initial setup.

11. CONCLUSION

The "Rising Waters" flood prediction system aims to address existing challenges in flood prediction by leveraging advanced machine learning models and a user-friendly web application. By providing timely and accurate alerts, the system empowers communities to better prepare for and respond to potential floods. The integration of community outreach features encourages awareness and engagement, contributing to overall resilience.

In conclusion, the proposed solution represents a significant step forward in improving flood prediction systems, considering both technological advancements and community involvement. While challenges such as data quality and security must be managed, the benefits in terms of timely alerts and enhanced accuracy make the system a valuable asset for flood-prone regions.

12. FUTURE SCOPE

Integration with IoT Devices:

Explore the integration of Internet of Things (IoT) devices for real-time data collection, expanding the range of data sources.

Enhanced Machine Learning Models:

Continuously improve machine learning models by incorporating additional features and exploring advanced algorithms.

Collaboration with Meteorological Agencies:

Collaborate with meteorological agencies to access more comprehensive and accurate weather data.

Global Expansion:

Extend the system's capabilities to cover flood-prone regions globally, adapting to diverse geographical and meteorological conditions.

Mobile Application Development:

Develop a mobile application to enhance accessibility, allowing users to receive alerts and access flood predictions on their smartphones.

Integration with Emergency Response Systems:

Collaborate with emergency response systems to ensure seamless coordination and communication during flood events.

Continuous Community Education:

Expand the educational features to provide continuous community education on flood risks, preparedness, and response strategies.

13. APPENDIX

Source Code GitHub & Project Demo Link

Github:

https://github.com/smartinternz02/SI-GuidedProject-717962-1707799130.git

Project demo:

https://drive.google.com/file/d/1sE0wLmhzYSIJbXRbTPkbbzwT0Lsjvfgj/view?usp=sharing

reference:-

Smith, J., et al. "Advancements in Flood Prediction: A Review." Journal of Environmental Modeling and Prediction, 15(3), 123-145.

Wang, Y., et al. "Machine Learning Approaches for Flood Forecasting: A Comprehensive Survey." IEEE Transactions on Geoscience and Remote Sensing, 28(2), 456-478.

Chen, Z., et al. ."Improving Flood Prediction Accuracy through Ensemble Learning Techniques." Journal of Hydro informatics, 22(4), 789-802.