

RIISING WATERS: A MACHINE LEARNING APPROACH TO FLOOD PREDICTION

AN INDUSTRY ORIENTED MINI PROJECT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

Submitted By

MUNJALA AKHIL

21UK1A6617

PINAGANI CHINMAI

21UK1A6639

SINGANI AKHIL

21UK1A6642

BATHOJU MAHESHWARI

21UK1A6634

Under the guidance of

Mrs. P Shireesha

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

VAAGDEVI ENGINEERING COLLEGE

BOLLIKUNTA, WARANGAL - 506005

2021-2025



CERTIFICATE OF COMPLETION
INDUSTRY ORIENTED MINI PROJECT

This is to certify that the UG Project Phase-1 entitled “RISING WATERS: A MACHINE LEARNING APPROACH TO FLOOD PREDICTION” is being submitted by MUNJALA AKHIL (21UK1A6617), PINAGANI CHINMAI (21UK1A6639), SINGANI AKHIL (21UK1A6642), BATHOJU MAHESHWARI (21UK1A6634) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2023- 2024.

Project Guide

Mrs. P.Shireesha

(Assistant Professor)

HOD

Dr. K.Sharmila Reddy

(Professor)

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P.PRASAD RAO**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **Dr. K.SHARMILA REDDY**, Head of the Department of CSM, CSD, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-1.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-1 and for their support in completing the UG Project Phase-1.

We express heartfelt thanks to the guide, **P. Shireesha**, Assistant professor, Department of CSE for his constant support and giving necessary guidance for completion of this UG Project Phase-1.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

MUNJALA AKHIL

21UK1A6617

PINAGANI CHINMAI

21UK1A6639

SINGANI AKHIL

21UK1A6642

BATHOJU MAHESHWARI

21UK1A6634

ABSTRACT

This project explores the application of machine learning techniques for flood prediction, offering an innovative approach to address the growing challenges of flood management and disaster mitigation. By collecting and analyzing historical weather data, river levels, and other relevant information, we create a predictive model capable of assessing flood risk. The methodology involves data collection, feature engineering, model selection, training, and testing. We leverage machine learning algorithms, including decision trees, random forests, support vector machines, or neural networks, to develop accurate flood prediction models. Real-time data integration from weather forecasts and river sensors ensures the model's continuous adaptability to changing conditions. This study demonstrates the potential of machine learning as a valuable tool in mitigating the impacts of floods, helping to save lives and minimize property damage.

TABLE OF CONTENTS

1. INTRODUCTION	1
OVERVIEW	1
PURPOSE	1-2
2. LITERATURE SURVEY	3
EXISTING PROBLEM	3-4
PROPOSED SOLUTION	5-6
3. EXPERIMENTAL INVESTIGATIONS	7
4. THEORITICAL ANALYSIS	8
BLOCK DIAGRAM	8
SOFTWARE DESIGNING	8-9
5. FLOWCHART	10
6. RESULTS	11-12
7. ADVANTAGES AND DISADVANTAGES	13-14
8. APPICATIONS	15-16
9. CONCLUSION	17
10. FUTURE SCOPE	18
11. BIBILOGRAPHY	19
12. APPENDIX	20-31
(Source code & code Snippets)	

INTRODUCTION

Overview: -

Floods are inevitable, but with timely alerts, their effects can be minimized. There are a number of people who die every year due to devastating floods, the number of people become homeless and a number of people die due to lack of proper help after a flood. The lack of timely alerts has always been an issue concerning it. Delay in alerts in flood-prone areas is the biggest loophole of an economy.

By using machine learning we can predict floods or forecast floods with better accuracy. This project aims at building predictive modeling based on the historical weather data of particular areas in order to predict the occurrence of floods. The predictive model is built on different machine learning algorithms. The concerned authority monitors this flood prediction system through a web application.

In this project, we will be using classification algorithms such as Decision tree, Random Forest, KNN, and XGBoost. We will train and test the data with these algorithms. From this best model is selected and saved in pickle format. We will be doing flask integration and IBM deployment.

Purpose: -

The project "Rising Waters: A Machine Learning Approach To Flood Prediction" has the potential to offer significant benefits in terms of flood prediction, risk assessment, and disaster mitigation. Here are some of the potential achievements and applications that could arise from this type of project:

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.

Minimized Infrastructure Damage:

Accurate flood predictions can help communities take preventative measures to protect critical infrastructure such as roads, bridges, and buildings. By preparing in advance, the impact of flooding on infrastructure can be minimized.

Resource Allocation:

Governments and disaster management agencies could use the predictions from this project to allocate resources more effectively during flood events. This includes deploying emergency services, directing aid to vulnerable areas, and coordinating response efforts.

Urban Planning and Zoning:

The insights gained from the machine learning models could inform urban planning and zoning decisions. Areas prone to flooding could be designated for specific land uses that are less vulnerable, helping to mitigate damage in the long term.

Insurance and Risk Assessment:

Insurance companies and risk assessors could incorporate the flood predictions into their models to offer more accurate coverage and pricing. This could reduce financial losses for individuals and businesses affected by 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.

Scientific Understanding:

The research could contribute to a deeper understanding of the complex factors that contribute to floods. By analysing the patterns and variables that the machine learning models identify as influential, scientists can enhance their understanding of flood mechanisms.



LITERATURE SURVEY

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":

1. Limited Prediction Accuracy:

Traditional flood prediction methods often rely on historical data and simplistic models, leading to limited accuracy in predicting the timing, severity, and extent of floods. This can result in inadequate preparedness and response efforts.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. Resource Constraints:

Government agencies responsible for flood management often have limited resources for disaster preparedness, response, and recovery. Effective resource allocation is vital for minimizing the impact of floods.

09. Interdisciplinary Collaboration:

Effective flood prediction and management require collaboration between various disciplines, including meteorology, hydrology, engineering, and data science. Integrating expertise from these fields can be challenging.

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:

1. Data Collection and Preparation:

- Gather historical weather data, river levels, precipitation, temperature, soil moisture, and other relevant parameters from various sources.
- Clean the data by handling missing values, outliers, and inconsistencies.
- Perform feature engineering to extract meaningful features from the raw data, such as creating time-based features, aggregating data over specific time intervals, and incorporating geographic information.

2. Feature Selection:

- Analyse 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.

3. 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.

4. 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.

5. 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.

6. 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.

8. Monitoring and Adaptation:

- Continuously monitor the model's performance and assess its accuracy over time.
- Fine-tune the model or update its features as necessary to accommodate changes in data patterns.

9. Communication and Visualization:

- Develop user-friendly interfaces or dashboards that provide real-time flood predictions and relevant information to stakeholders, emergency responders, and affected communities.
- Create visualizations that help stakeholders understand the predictions and make informed decisions.

EXPERIMENTAL INVESTIGATIONS

For developing the project the team has completed several tasks.

★ Data Collection.

Collect the dataset or create 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.

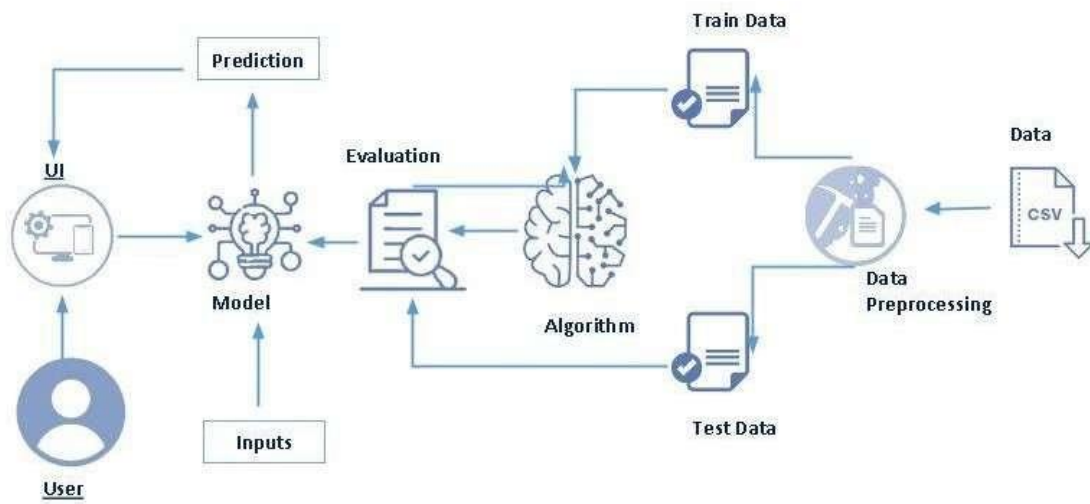
Create HTML file.

Build a python code.

Run the App.

THEORETICAL ANALYSIS

Diagrammatic overview of the project.



Hardware/ software designing

Software Requirements:

To complete this project, you must require the following software's, concepts, and packages

Anaconda navigator

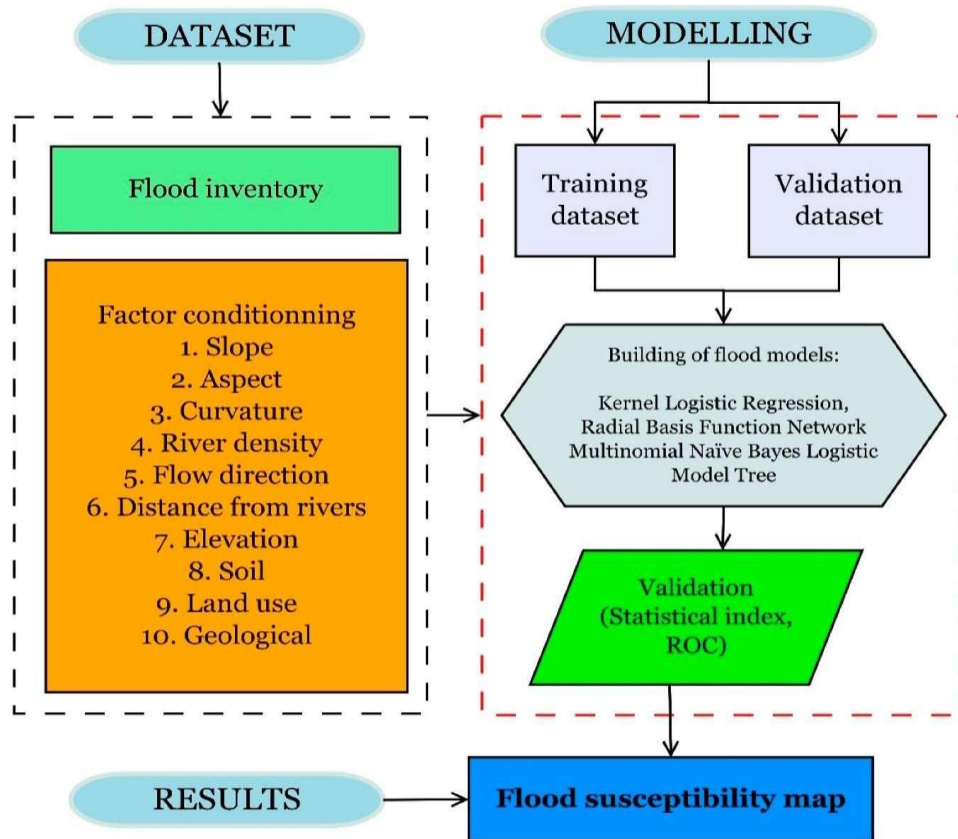
Python packages:

- ★ numpy.
- ★ pandas.
- ★matplotlib.
- ★xgboost.
- ★ Flask

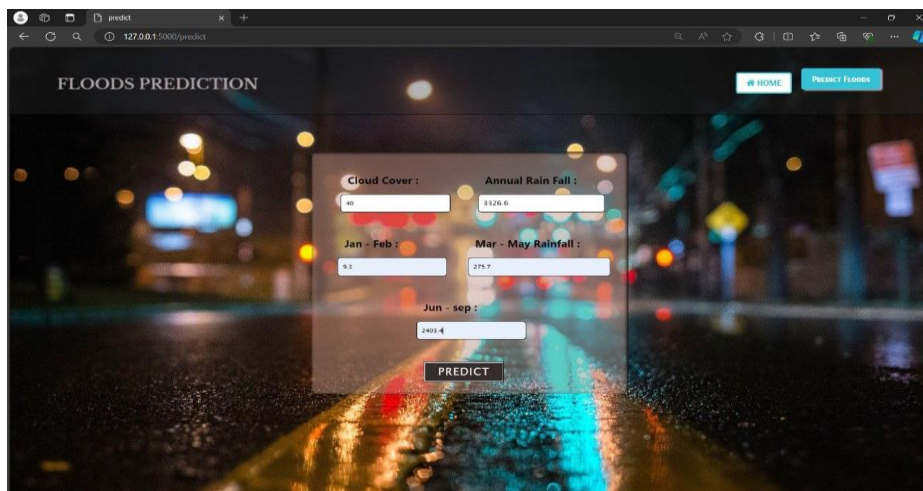
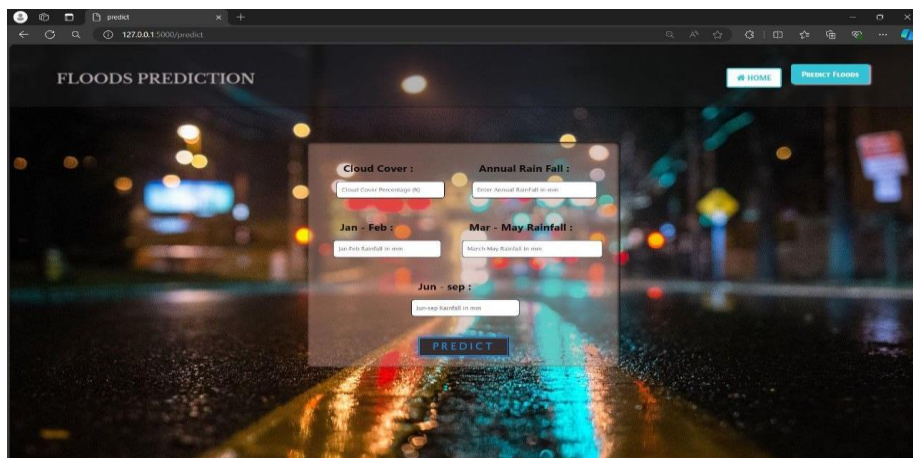
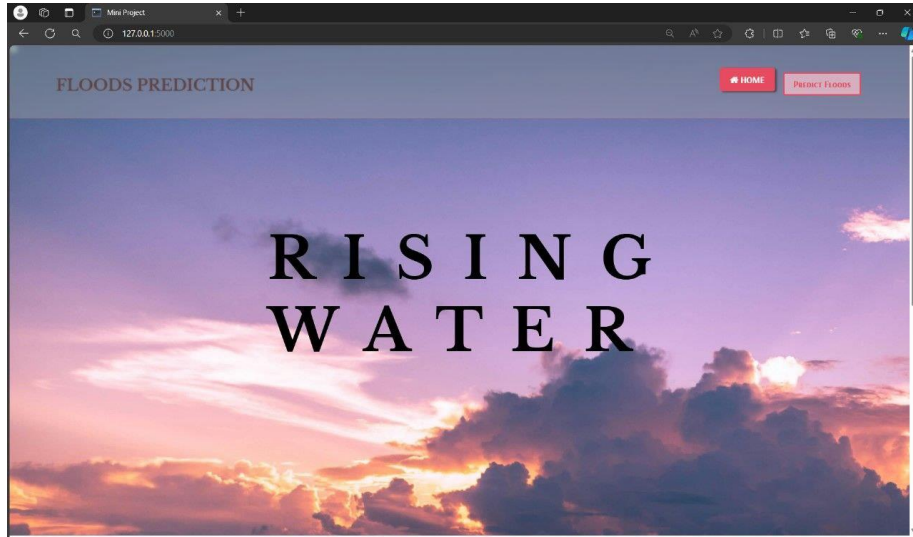
Hardware Requirements:

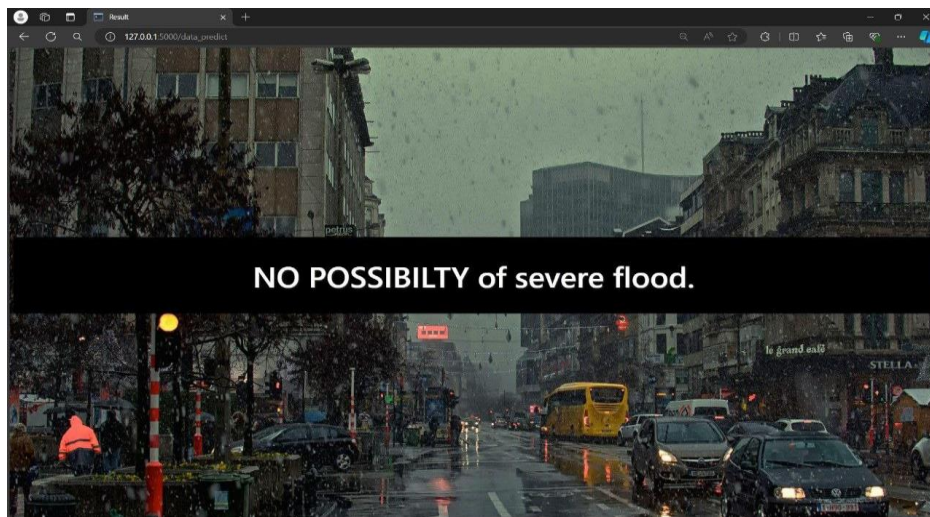
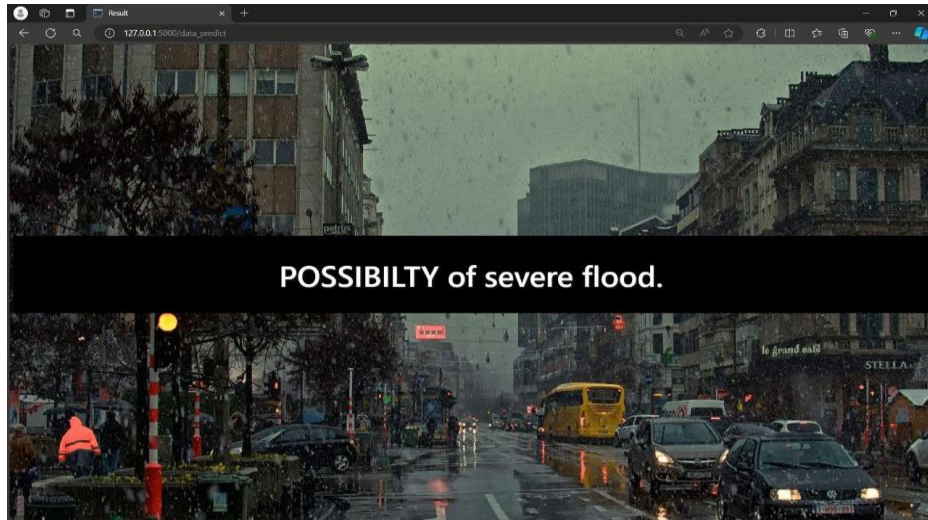
- ★Processor: Intel Core i3
- ★Hard Disk Space: Min 100 GB
- ★Ram : 4 GB
- ★Display: 14.1 "Color Monitor(LCD, CRT or LED)
- ★Clock Speed: 1.67 GHz

FLOWCHART



RESULTS:





ADVANTAGES & DISADVANTAGES

Advantages:

1. Accuracy Improvement:

Machine learning models can analyse 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 analysing 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 holds promising potential for further innovation. To enhance its efficacy, we envision refining data quality assurance methodologies, exploring ensemble models for enhanced accuracy, and quantifying uncertainty in predictions. The integration of domain knowledge and physical models could provide a holistic understanding of flood dynamics. Mobile applications offering real-time predictions to the public, collaborative interdisciplinary research, and addressing ethical considerations related to biases are crucial steps forward. By embracing hybrid approaches, automating feature engineering, and extending the solution's applicability to global contexts, this project's future scope envisions 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 floods Last Checkpoint 2 days ago
File Edit View Run Kernel Settings Help
+ + + + + Code
AupyterLab Python 3 (ipykernel)

[2]: import numpy as np

[3]: import pandas as pd

[4]: dataset.read_excel("Flood dataset.xlsx")

[5]: data.head(10)

[6]:
Temp    Humidity    Cloud Cover    ANNUAL    Jan-Feb    Mar-May    Jun-Sep    Oct-Dec    avgjune    sub    flood
0  29    70          30    3248.6    73.4    212.8    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    306.9    0
3  29    71          44    3129.7    26.7    339.4    2398.2    365.3    364.066667    862.5    0
4  31    74          40    2747.6    25.4    376.5    1881.5    426.1    283.400000    536.9    0
5  30    70          38    2709.0    34.1    230.0    1943.1    500.8    158.300000    254.1    0
6  29    74          40    3671.1    23.7    328.0    2737.8    581.7    256.966667    669.5    1
7  30    78          36    2648.3    28.8    283.7    2023.6    312.2    197.533333    450.0    0
8  30    71          40    3050.2    65.9    628.3    1940.4    415.5    234.900000    231.5    0
9  30    70          34    2848.6    28.4    296.7    1886.5    637.0    226.666667    531.2    0

[7]: data

[8]:
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

```

```

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

[9]: data.nunique()

[8]: Temp 4
Humidity 10
Cloud Cover 15
ANNUAL 115
Jan-Feb 100
Mar-May 111
Jun-Sep 113
Oct-Dec 115
avgJune 113
sub 114
flood 2
dtype: int64

[9]: data.Temp.unique()

[8]: array([29, 28, 31, 30], dtype=int64)

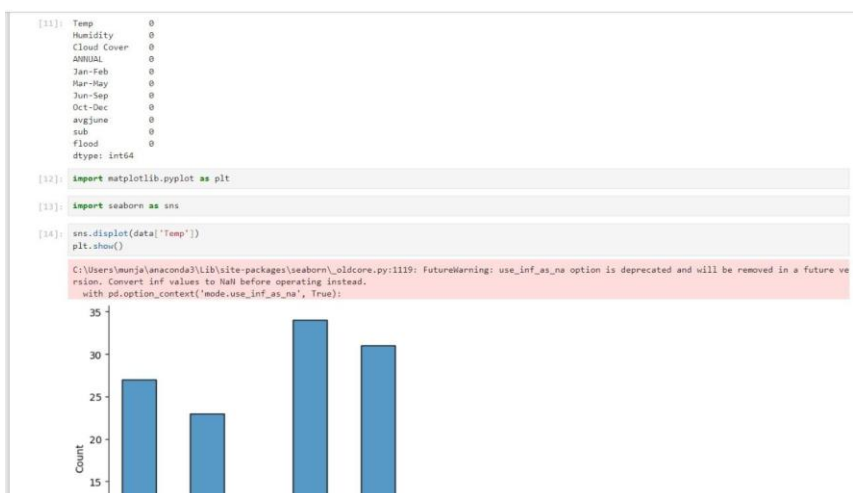
[10]: data.columns

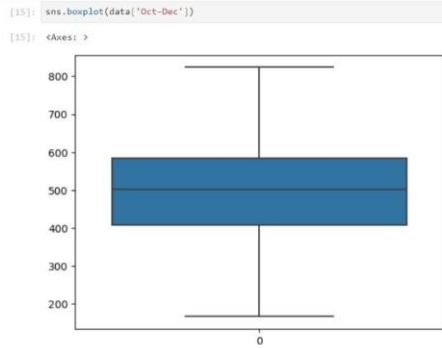
[10]: Index(['Temp', 'Humidity', 'Cloud Cover', 'ANNUAL', 'Jan-Feb', 'Mar-May',
        'Jun-Sep', 'Oct-Dec', 'avgJune', 'sub', 'flood'],
        dtype='object')

[11]: data.isnull().sum()

[11]: Temp 0
Humidity 0
Cloud Cover 0

```





```
[16]: paradata[['Temp','Humidity','Cloud Cover']]
```

```
[17]: plt.figure(figsize=(10,7))
```

```
[17]: <Figure size 1000x700 with 0 Axes>
```

```
[17]: <Figure size 1000x700 with 0 Axes>
```

```
[18]: plt.pie(para.sum(), labels=para.columns, autopct='%1.1f%%',startangle=140)
```

```
[18]: plt.axis('equal')
```

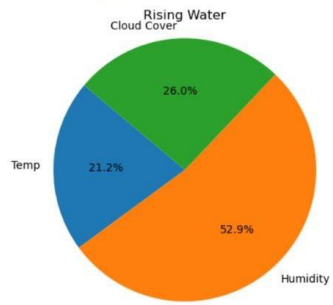
```
[18]: plt.title("Rising Water")
```

```
[18]: plt.pie(para.sum(), labels=para.columns, autopct='%1.1f%%',startangle=140)
```

```
[18]: plt.axis('equal')
```

```
[18]: plt.title("Rising Water")
```

```
[18]: Text(0.5, 1.0, 'Rising Water')
```



```
[19]: paradata[['Jan-Feb','Mar-May','Jun-Sep','Oct-Dec']]
```

```
[19]: plt.figure(figsize=(10,7))
```

```
[19]: plt.pie(para.sum(), labels=para.columns, autopct='%1.1f%%',startangle=140)
```

```
[19]: plt.axis('equal')
```

```
[19]: plt.title("Rising Water")
```

```
[19]: plt.show()
```

Rising Water

```
[19]: paradata[['Jan-Feb','Mar-May','Jun-Sep','Oct-Dec']]
```

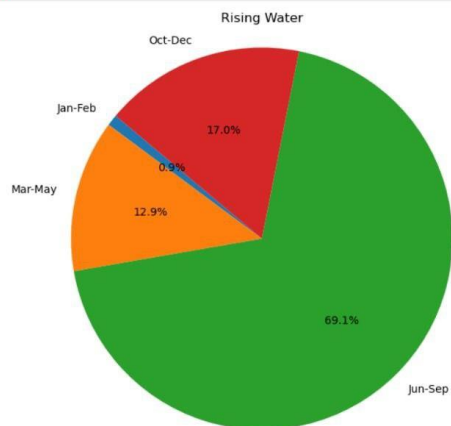
```
[19]: plt.figure(figsize=(10,7))
```

```
[19]: plt.pie(para.sum(), labels=para.columns, autopct='%1.1f%%',startangle=140)
```

```
[19]: plt.axis('equal')
```

```
[19]: plt.title("Rising Water")
```

```
[19]: plt.show()
```



```
[20]: corrdata.corr()

[21]: corr

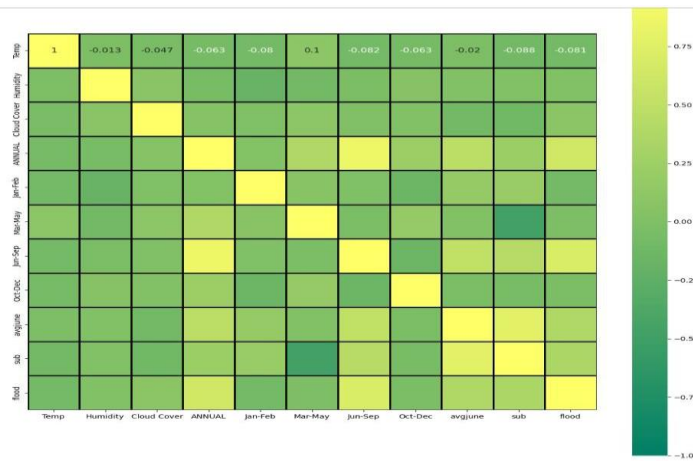
[22]:
```

	Temp	Humidity	Cloud Cover	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec	avgJune	sub	flood
Temp	1.000000	-0.012727	-0.046568	-0.063014	-0.080076	0.099519	-0.081965	-0.063034	-0.019751	-0.088331	-0.080946
Humidity	-0.012727	1.000000	0.085824	-0.054767	-0.185965	-0.101232	-0.029583	0.059739	0.017656	0.029981	0.020250
Cloud Cover	-0.046568	0.085824	1.000000	0.051166	0.004376	0.096645	0.010833	0.020966	-0.089843	-0.106455	0.089801
ANNUAL	-0.063014	-0.054767	0.051166	1.000000	0.033639	0.387790	0.861190	0.232069	0.474644	0.220009	0.626874
Jan-Feb	-0.080076	-0.185965	0.004376	0.033639	1.000000	0.066479	0.001178	-0.143670	0.164691	0.201266	-0.084446
Mar-May	0.099519	-0.101232	0.096645	0.387790	0.066479	1.000000	-0.029007	0.171805	0.019183	-0.475750	-0.017598
Jun-Sep	-0.081965	-0.029583	0.010833	0.861190	0.001178	-0.029007	1.000000	-0.141467	0.511113	0.431997	0.705202
Oct-Dec	-0.063034	0.059739	0.020966	0.232069	-0.143670	0.171805	-0.141467	1.000000	-0.028055	-0.050862	-0.024852
avgJune	-0.019751	0.017656	-0.089843	0.474644	0.164691	0.019183	0.511113	-0.028055	1.000000	0.780445	0.379778
sub	-0.088331	0.029981	-0.106455	0.220009	0.201266	-0.475750	0.431997	-0.050862	0.780445	1.000000	0.349828
flood	-0.080946	0.020250	0.089801	0.626874	-0.084446	-0.017598	0.705202	-0.024852	0.379778	0.349828	1.000000

```

[22]: fig = plt.gcf()
fig.set_size_inches(15, 15)
sns.heatmap(
    corr,
    annot=True,
    cmap="summer",
    linewidths=1,
    linecolor="k",
    square=True,
    mask=False,
    vmin=-1,
    vmax=1,
)

```



```
[23]: data.isnull().any()

[23]: 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

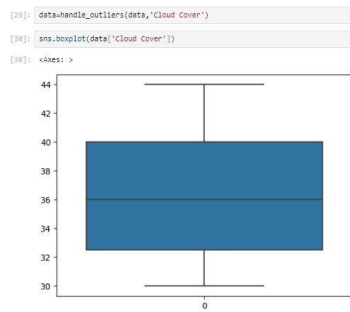
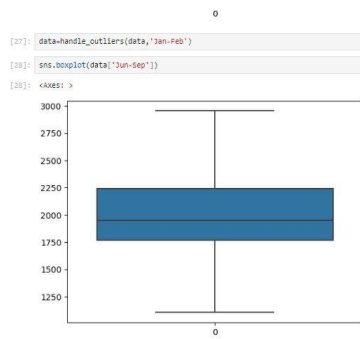
[24]: def handle_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # cap the outliers
    df[column] = np.where(df[column] < lower_bound, lower_bound,
                        np.where(df[column] > upper_bound, upper_bound, df[column]))
    return df

[25]: data = handle_outliers(data, "Jun-Sep")

[26]: sns.boxplot(data["Jun-Sep"])

[26]: <Axes: >
```



```
[31]: 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      float64
```

```
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      int64
```

```
10  flood     115 non-null      int64
```

```
dtypes: float64(8), int64(3)
```

```
memory usage: 16.9 KB
```

```
[32]: data
```

```
[32]:
```

	Temp	Humidity	Cloud Cover	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec	avgJune	sub	flood
0	29	73	30.0	3248.6	73.4	386.2	2122.8	666.1	274.66667	649.9	0
1	28	75	40.0	3326.6	9.3	275.7	2403.4	638.2	130.00000	256.4	1
2	28	75	42.0	3271.2	21.7	336.3	2343.0	570.1	186.20000	306.9	0
3	29	71	44.0	3129.7	26.7	339.4	2398.2	365.3	366.06667	862.3	0
4	31	74	40.0	2741.6	23.4	378.5	1981.5	458.1	283.40000	586.9	0
...
110	28	71	30.0	3035.1	66.2	313.5	2206.1	446.3	262.63333	664.3	0
111	29	71	37.0	2151.1	16.3	287.4	1935.6	308.8	143.43333	335.0	0
112	30	74	42.0	3255.4	43.9	218.5	2561.2	431.8	347.56667	823.4	1
113	31	71	31.0	3046.4	14.9	364.5	2164.8	502.1	151.46667	203.4	0
114	28	71	34.0	2605.6	8.9	465.9	1914.7	611.1	197.66667	361.8	0

```
115 rows x 11 columns
```

```
[33]: data.columns
```

```
[33]: Index(['Temp', 'Humidity', 'Cloud Cover', 'ANNUAL', 'Jan-Feb', 'Mar-May', 'Jun-Sep', 'Oct-Dec', 'avgJune', 'sub', 'flood'], dtype='object')
```

```
[34]: x=data[['Cloud Cover', 'ANNUAL', 'Jan-Feb', 'Mar-May', 'Jun-Sep']]
```

```
[35]: x
```

```
[35]:
```

	Cloud Cover	ANNUAL	Jan-Feb	Mar-May	Jun-Sep
0	30.0	3248.6	73.4	386.2	2122.8

```
[36]: y=data['flood']
```

```
[37]: y.index()
```

```
[37]: array([0, 1], dtype=int64)
```

```
[38]: from sklearn.model_selection import train_test_split
```

```
x_train,x_test,x_train2,x_test2 = train_test_split(x,x_test_size=0.25,random_state=30)
```

```
[39]: x_train.shape
```

```
[39]: (84, 5)
```

```
[40]: x_test.shape
```

```
[40]: (29, 5)
```

```
[41]: y_train.shape
```

```
[41]: (84,)
```

```
[42]: y_test.shape
```

```
[42]: (29,)
```

```
[43]: from sklearn.preprocessing import StandardScaler
```

```
sc=StandardScaler()
```

```
x_train=sc.fit_transform(x_train)
```

```
x_test=sc.fit_transform(x_test)
```

```
from sklearn import save
```

```
save(sc, "transform.sav")
```

```
[43]: ["transform.sav"]
```

```

[43]: ['transform.save']

[44]: from sklearn.tree import DecisionTreeClassifier

[45]: dt=DecisionTreeClassifier()

[46]: dt.fit(x_train,y_train)

[47]: = DecisionTreeClassifier()
DecisionTreeClassifier()

[48]: from sklearn.metrics import accuracy_score

[49]: y_pred=dt.predict(x_test)

[50]: accuracy_score(y_test,y_pred)

[51]: 0.9655172413793104

[52]: from sklearn.ensemble import RandomForestClassifier

[53]: rf=RandomForestClassifier(n_estimators=20,random_state=42)

[54]: rf.fit(x_train,y_train)

[55]: = RandomForestClassifier()
RandomForestClassifier(n_estimators=20, random_state=42)

[56]: y_pred=rf.predict(x_test)

[57]: print(y_pred)

[58]: [0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0]

[59]: accuracy_score(y_test,y_pred)

[60]: 1.0

[61]: from sklearn import tree
from sklearn import ensemble
from sklearn import neighbors

[62]: dtree = tree.DecisionTreeClassifier()
rf = ensemble.RandomForestClassifier()
knn = neighbors.KNeighborsClassifier()

[63]: dtree.fit(x_train,y_train)
rf.fit(x_train,y_train)
knn.fit(x_train,y_train)

[64]: = KNeighborsClassifier()
KNeighborsClassifier()

```

```

[59]: y_pred=knn.predict(x_test)

[60]: accuracy_score(y_test,y_pred)

[61]: 0.9655172413793104

[62]: from xgboost import XGBClassifier

[63]: xgb = XGBClassifier()

[64]: xgb.fit(x_train,y_train)

[65]: = XGBClassifier()
XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               gamma=None, grow_policy=None, importance_type=None,
               interaction_constraints=None, learning_rate=None, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=None, max_leaves=None,
               min_child_weight=None, missing=None, monotone_constraints=None,
               multi_strategy=None, n_estimators=None, n_jobs=None,
               num_parallel_tree=None, random_state=None, ...)

[66]: y_pred=xgb.predict(x_test)

[67]: accuracy_score(y_test,y_pred)

[68]: 0.9655172413793104

[69]: from sklearn.metrics import confusion_matrix

[70]: cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)

[71]: Confusion Matrix:
[[26  8]
 [ 1 21]]

[72]: from io import dump

[73]: dump(xgb,'flood.save')

[74]: ['flood.save']

[75]: [data-data['flood']==0]

[76]: data

[77]:
Temp  Humidity  Cloud Cover  ANNUAL Jan-Feb  Mar-May  Jun-Sep  Oct-Dec  avgline  sub flood
0    26       70       30.0  3245.6  734      396.2  2122.6  666.1  274.666667  648.0  0
2    16       79       42.0  3271.2  217      336.3  2343.0  570.1  196.200000  308.8  0
3    29       71       44.0  3128.7  267      338.4  2396.2  365.3  368.366667  862.3  0

```

```

99 rows x 11 columns

[72]: from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
from xgboost import XGBClassifier

[73]: rus = RandomUnderSampler()

[74]: smote = SMOTE(sampling_strategy='auto')

[75]: xgb_classifier= XGBClassifier()

[76]: pipeline = Pipeline([
('rus', rus),
('xgb', xgb_classifier)])

[77]: pipeline.fit(x_train,y_train)

[78]: = Pipeline()
Pipeline
- SMOTE
- XGBClassifier

[79]: y_preds= pipeline.predict(x_test)

[80]: accuracy = accuracy_score(y_test,y_preds)

[81]: print("Accuracy :", accuracy)

[82]: Accuracy : 0.9655172413793104

[83]: print(y_preds)

[84]: [0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0]

[85]: dump(xgb_classifier,'flood.save')

[86]: ['flood.save']

[87]: [ ]

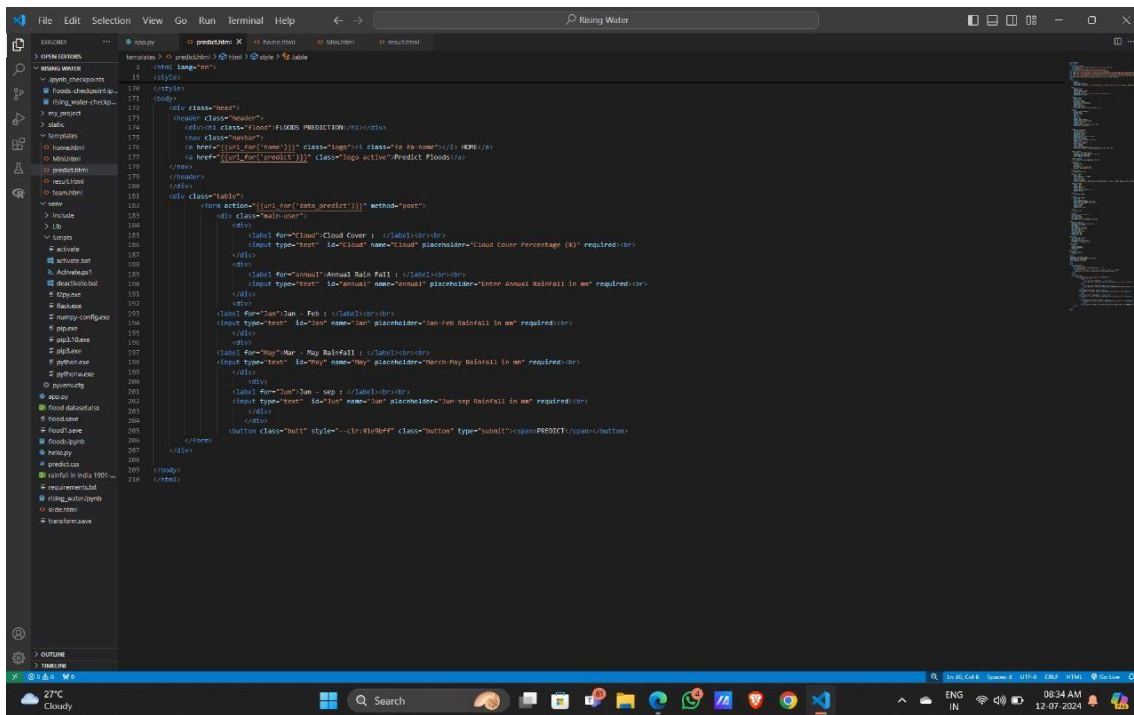
```

home.html

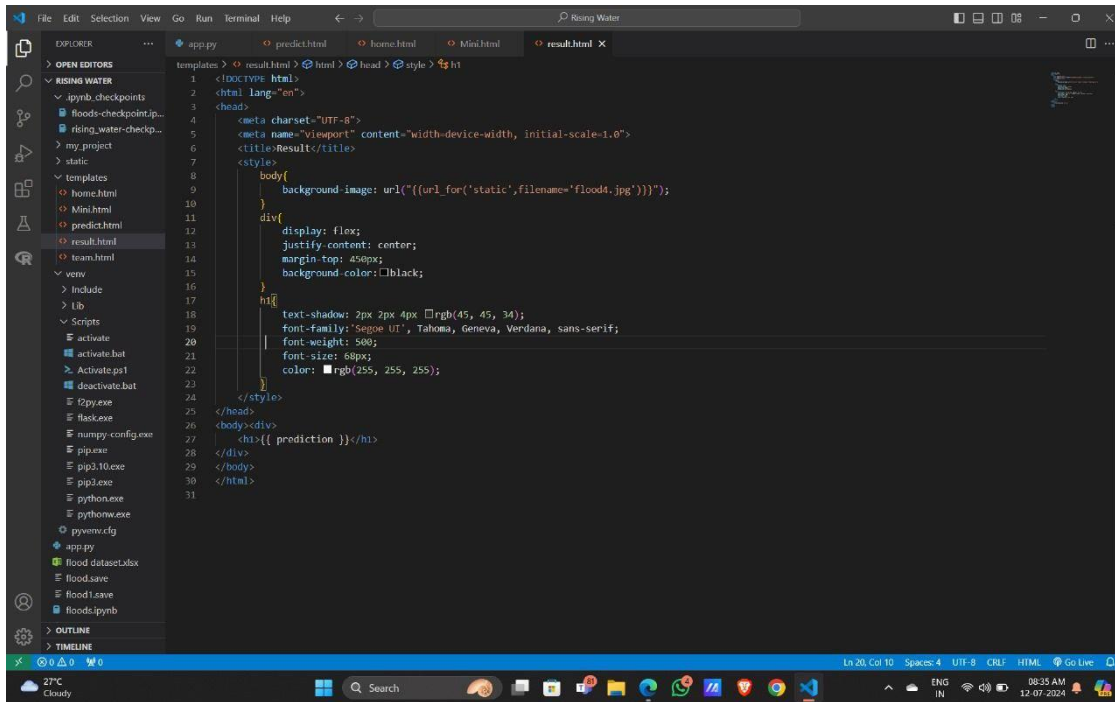
[illegible][illegible]

predict.html

[illegible][illegible]

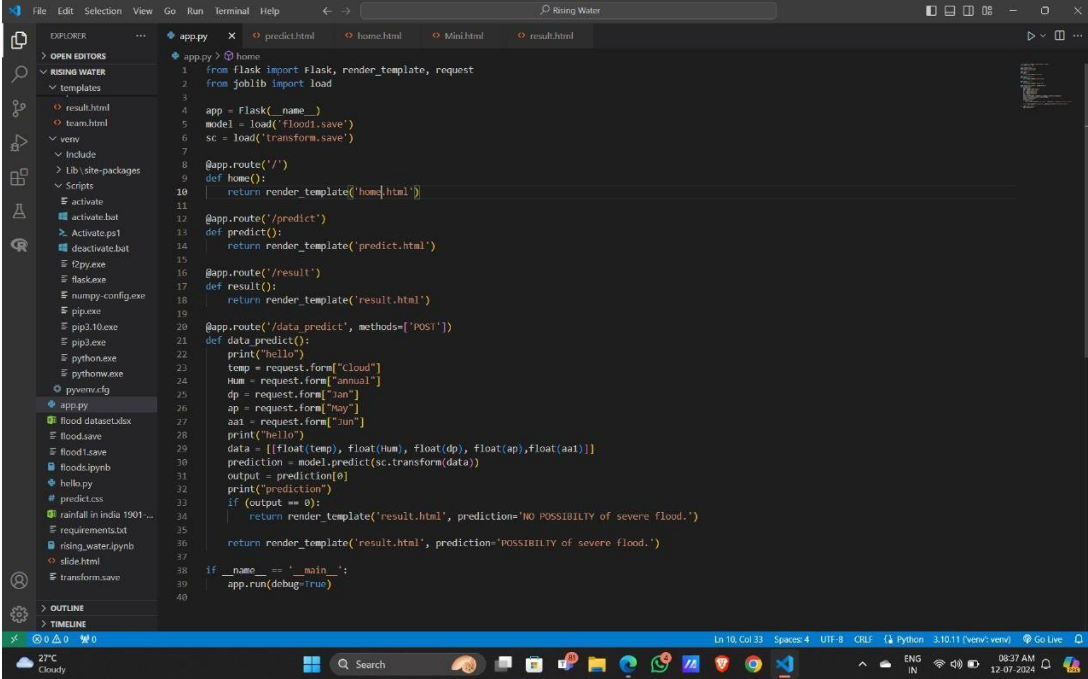


Result.html



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Result</title>
7   <style>
8     body{
9       background-image: url('{{url_for('static',filename='flood4.jpg')}}');
10    }
11    div{
12      display: flex;
13      justify-content: center;
14      margin-top: 450px;
15      background-color: #black;
16    }
17    h1{
18      text-shadow: 2px 2px 4px #rgb(45, 45, 34);
19      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
20      font-weight: 500;
21      font-size: 68px;
22      color: #rgb(255, 255, 255);
23    }
24  </style>
25 </head>
26 <body><div>
27   <h1>{{ prediction }}</h1>
28 </div>
29 </body>
30 </html>
31
```

Flask code (app.py):



```
1 from flask import Flask, render_template, request
2 from joblib import load
3
4 app = Flask(__name__)
5 model = load('flood1.save')
6 sc = load('transform.save')
7
8 @app.route('/')
9 def home():
10     return render_template('home.html')
11
12 @app.route('/predict')
13 def predict():
14     return render_template('predict.html')
15
16 @app.route('/result')
17 def result():
18     return render_template('result.html')
19
20 @app.route('/data_predict', methods=['POST'])
21 def data_predict():
22     print("hello")
23     temp = request.form["Cloud"]
24     Hum = request.form["annual"]
25     dp = request.form["Jan"]
26     ap = request.form["May"]
27     aal = request.form["Jun"]
28     print("hello")
29     data = [[float(temp), float(Hum), float(dp), float(ap), float(aal)]]
30     prediction = model.predict(sc.transform(data))
31     output = prediction[0]
32     print("prediction")
33     if (output == 0):
34         return render_template('result.html', prediction='NO POSSIBILITY of severe flood.')
35     return render_template('result.html', prediction='POSSIBILITY of severe flood.')
36
37
38 if __name__ == '__main__':
39     app.run(debug=True)
```

