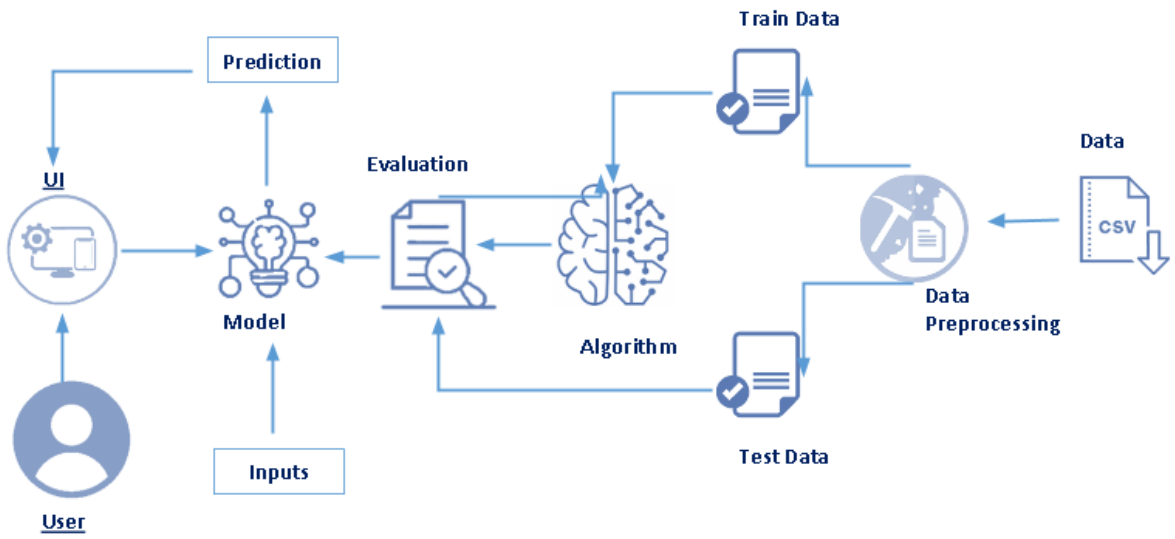
Rising Waters: A Machine Learning Approach to Flood Prediction

Floods are among the most destructive natural disasters worldwide. They cause significant damage to human life, infrastructure, agriculture, and the economy. Accurate flood prediction is essential for disaster management, early warning systems, and mitigation planning. Traditional flood forecasting methods rely on statistical techniques and manual monitoring. However, these approaches often fail to capture complex relationships among meteorological variables such as rainfall distribution, temperature, humidity, and cloud cover. This project presents a Machine Learning-based flood prediction system that analyzes historical weather data to classify whether there is a possibility of flood occurrence. The trained model is deployed using a Flask web application, enabling real-time flood risk prediction.

Technical Architecture Diagram



Pre-Requisites:

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

Python packages:

- Open command prompt as administrator

- Type “pip install numpy” and click enter.
- Type “pip install pandas” and click enter.
- Type “pip install scikit-learn” and click enter.
- Type ”pip install matplotlib” and click enter.
- Type ”pip install pickle-mixin” and click enter.
- Type ”pip install seaborn” and click enter.
- Type “pip install Flask” and click enter.

Refer this link to know about the above libraries.

Prior Knowledge

You must have prior knowledge of the following topics to complete this project.

ML Concepts

SUPERVISED LEARNING:https://youtu.be/QeKshry8pWQ

UNSUPERVISED LEARNING:https://youtu.be/D6gtZrsYi6c

METRICS:https://youtu.be/aWAnNHXIKww

FLASK: https://youtu.be/lj4I_CvBnt0

Project Objectives

Write what are all the technical aspects that students would get if they complete this project.

1. Knowledge of Machine Learning Algorithms.
1. Knowledge of Machine Learning Algorithms.
2. Knowledge of Python Language with Machine Learning
3. You’ll be able to understand the problem to classify if it is a regression or a classification kind of problem.
4. You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
5. Applying different algorithms according to the dataset and based on visualization.

6. Real-Time Analysis of Project
7. Building ease of User Interface (UI)
8. Navigation of ideas towards other projects(creativeness)
9. Knowledge of building ML models.
10. How to build web applications using the Flask framework.

PROJECT FLOW:

➤ The data is loaded and cleaning and feature extraction is performed on the data.

➤ Model is trained using machine learning models.

➤ Once model is trained, it is tested and evaluation is performed.

To accomplish this, we have to complete all the activities listed below:

1. Install Required Libraries.
2. Data Collection.
   - Collect the dataset or Create the dataset
3. Data Preprocessing.
   - Import the Libraries.
   - Importing the dataset.
   - Understanding Data Type and Summary of features.
   - Take care of missing data
   - Data Visualization.
   - Drop the column from DataFrame & replace the missing value.
   - Splitting the Dataset into Dependent and Independent variables
   - Splitting Data into Train and Test.

4. Model Building

- Training and testing the model
- Evaluation of Model
- Saving the Model

5. Application Building

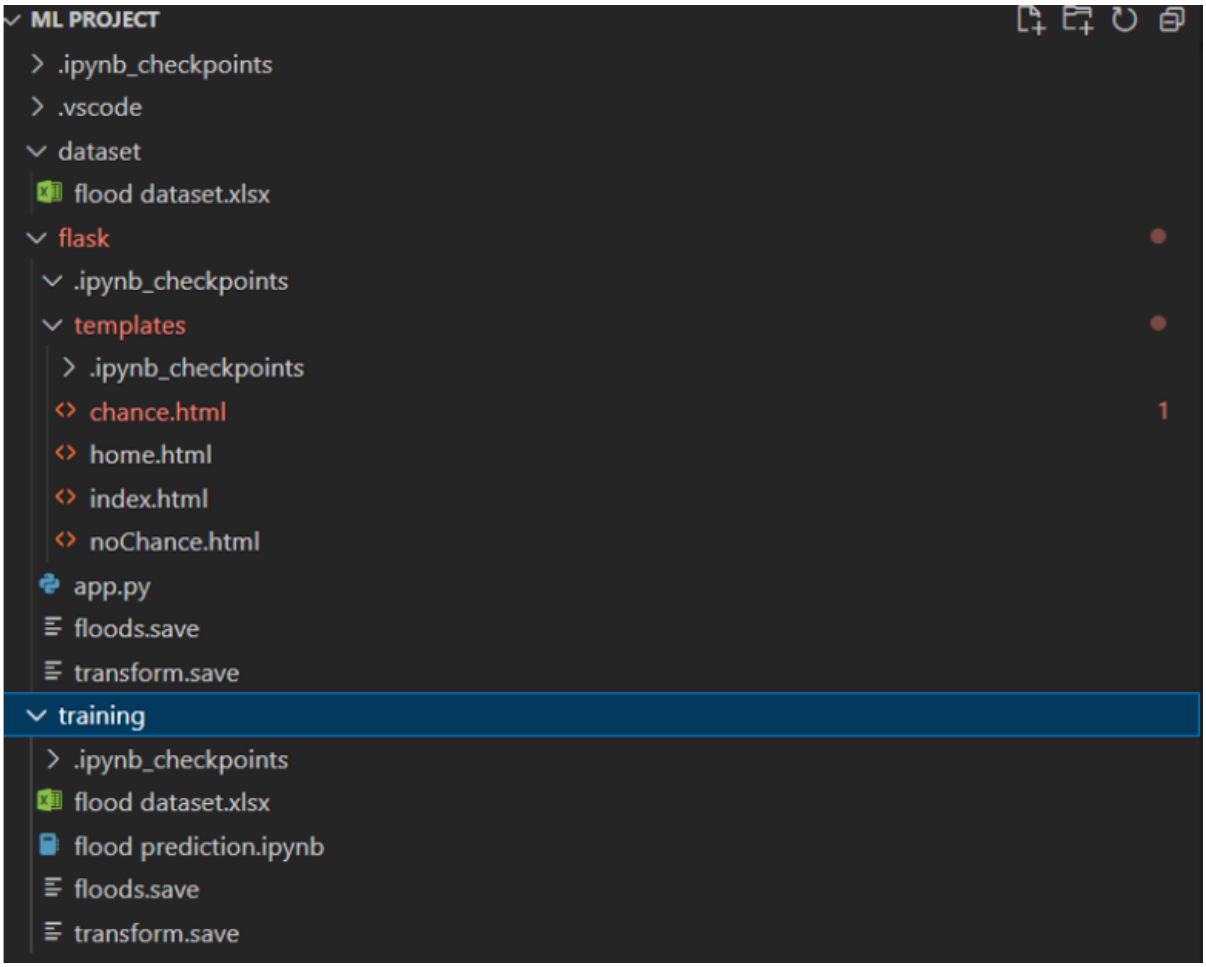
- Create an HTML file
- Build a Python Code

6. Final UI

- Dashboard Of the flask app

Project Structure

Create a Project folder that contains files as shown below



We are building a Flask Application that needs HTML pages “home.html”, “index.html” stored in the templates folder and a python script app.py for serve r side scripting

The model is built in the notebook floods.ipynb

> We need the model which is saved and the saved model in this content is f loods.save and transform.save

> The templates mainly used here are “home.html”, ”chance.html”, ”no chance.html”, “index.html” for showcasing the UI

> The flask app is denoted as app.py

Data Collection

ML depends heavily on data, without data, it is impossible for an “AI” to learn. It is themost crucial aspect that makes algorithm training possible. In Machine L earningprojects, we need a training data set. It is the actual data set used to train the model for performing various actions.

Download The Data Set

Download the dataset from the below link.

> There are many features that are responsible for predicting floods e.g. A nnual rain fall, Cloud visibility, June-Sep rain fall etc.

> You can collect datasets from different open sources like kaggle.com, d ata.gov, UCI machine learning repository, etc.

> Here we are using a data set which you can find in the below link and y ou can download it from the link: LINK

Visualizing And Analyzing The Data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.
Note: There is n number of techniques for understanding the data. But here we h ave used some of it. In an additional way, you can use multiple techniques.

Importing The Libraries

Import the necessary libraries as shown in the image
Import the required libraries for the model to run. The first step is usually impor ting the libraries that will be needed in the program.

Reading The Dataset

● Our dataset format might be in .csv, excel files, .txt, .json, etc.We can read the

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

● dataset with the help of pandas.
● In pandas, we have a function called read_csv() to read the dataset. As a para meter, we have to give the directory of CSV file.

```
dataset = pd.read_excel('flood dataset.xlsx')
```

Uni-Variate Analysis

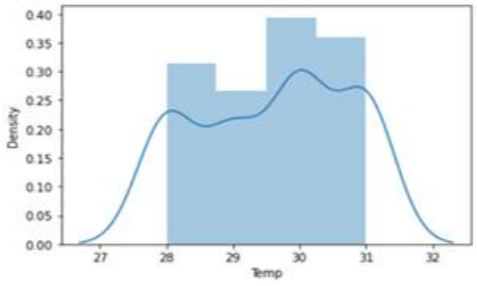
In simple words, univariate analysis is understanding the data with a single feat ure. Herewe have displayed two different graphs such as distribution plot, box p lot.

Distribution plot :

· The distribution plot is suitable for comparing range and distribution for grou ps of numerical data. Data are plotted as value points along an axis.
· it is used to identify, what kind of distribution does the data follow

```
print(sns.distplot(dataset["Temp"]))
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be rem
oved in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
AxesSubplot(0.125,0.125;0.775x0.755)
```



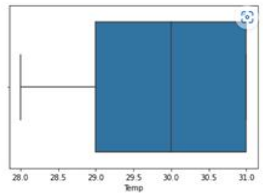
From the above graph, we can infer that the column temp follows some kind of normal distribution, it means the data is normal is almost normal

Boxplot:

· Box plot is used on the length of service and average training score feature. Le ngthof services feature has more outliers. The model should not be built without handling the outliers. Here, outliers are handled by the capping method. Cappin g will be discussed on data pre-processing.

```
In [6]: print(sns.boxplot(dataset['Temp']))
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will re
sult in an error or misinterpretation.
  warnings.warn(
AxesSubplot(0.125,0.125;0.775x0.755)
```



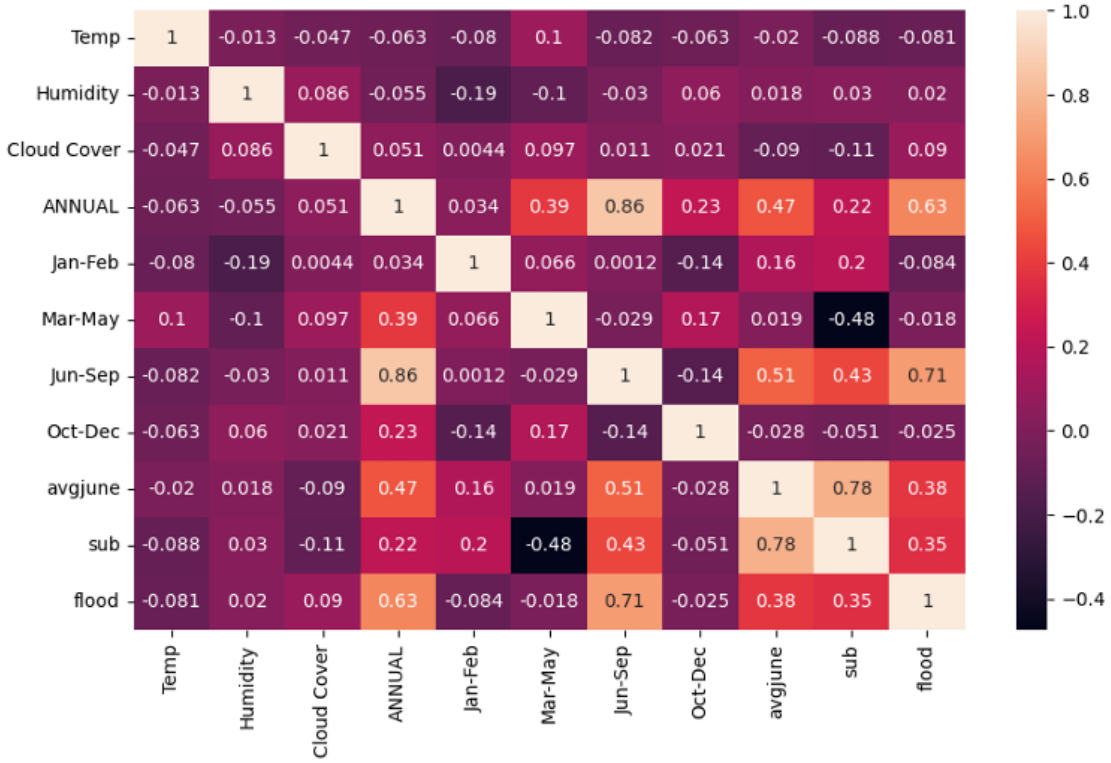
Multivariate Analysis

In simple words, multivariate analysis is to find the relation between multiple fe atures.

Heat map :

A heat map is a data visualization technique that shows magnitude of a phenom enon ascolour in two dimensions. The variation in colour may be by hue or inte nsity, givingobvious visual cues to the reader about how the phenomenon is clus tered or varies over space

```
#heat map
import seaborn as sns
fig=plt.gcf()
fig.set_size_inches(15,15)
fig=sns.heatmap(dataset.corr(),annot=True,cmap='summer',
                linewidths=1,linecolor='k',square=True,
                mask=False, vmin=-1, vmax=1,
                cbar_kws={"orientation": "vertical"},cbar=True)
```



Descriptive Analysis

To check the first five rows of the dataset, we have a function called head( ).

Descriptive Analysis

```
dataset.drop(["Oct-Dec"],axis=1,inplace=True)
```

```
dataset.head(10)
```

	Temp	Humidity	Cloud Cover	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	avgjune	sub	flood
0	29	70	30	3248.6	73.4	386.2	2122.8	274.866667	649.9	0
1	28	75	40	3326.6	9.3	275.7	2403.4	130.300000	256.4	1
2	28	75	42	3271.2	21.7	336.3	2343.0	186.200000	308.9	0
3	29	71	44	3129.7	26.7	339.4	2398.2	366.066667	862.5	0
4	31	74	40	2741.6	23.4	378.5	1881.5	283.400000	586.9	0
5	30	70	38	2708.0	34.1	230.0	1943.1	138.300000	254.1	0
6	29	74	40	3671.1	23.7	328.0	2737.8	256.966667	669.5	1
7	30	78	36	2648.3	28.8	283.7	2023.6	197.533333	450.0	0
8	30	71	40	3050.2	65.9	628.3	1940.4	234.900000	231.5	0
9	30	70	34	2848.6	28.4	296.7	1886.5	226.666667	531.2	0

Understanding Data Type and Summary of features

How the information is stored in a DataFrame or Python object affects what we can do with it and

the outputs of calculations as well. There are two main types of data: numeric a nd text data types. ● Numeric data types include integers and floats.

● Text data type is known as Strings in Python, or Objects in Pandas. Strings ca n contain numbers and / or characters.

● For example, a string might be a word, a sentence, or several sentences.

● Will see how our dataset is, by using info() method.

● info() method provides the summary of data.

```
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
```

As you can see in our dataset there is no textual data, all the set of data is in floa t and integer type.

Describe () functions are used to compute values like count, mean, standard dev iation give a summary type of data.

```
dataset.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Temp	115.0	29.600000	1.122341	28.0	29.000000	30.000000	31.000000	31.000000
Humidity	115.0	73.852174	2.947623	70.0	71.000000	74.000000	76.000000	79.000000
Cloud Cover	115.0	36.286957	4.330158	30.0	32.500000	36.000000	40.000000	44.000000
ANNUAL	115.0	2925.487826	422.112193	2068.8	2627.900000	2937.500000	3164.100000	4257.800000
Jan-Feb	115.0	27.739130	22.361032	0.3	10.250000	20.500000	41.600000	98.100000
Mar-May	115.0	377.253913	151.091850	89.9	276.750000	342.000000	442.300000	915.200000
Jun-Sep	115.0	2022.840870	386.254397	1104.3	1768.850000	1948.700000	2242.900000	3451.300000
Oct-Dec	115.0	497.636522	129.860643	166.6	407.450000	501.500000	584.550000	823.300000
avgjune	115.0	218.100870	62.547597	65.6	179.666667	211.033333	263.833333	366.066667
sub	115.0	439.801739	210.438813	34.2	295.000000	430.600000	577.650000	982.700000
flood	115.0	0.139130	0.347597	0.0	0.000000	0.000000	0.000000	1.000000

Data Pre-Processing

As we have understood how the data is. Let's pre-process the collected data.
The download data set is not suitable for training the machine learning model as it might
have so much randomness so we need to clean the dataset properly in order to fe tch good results. This activity includes the following steps.

> Handling missing values

> Handling categorical data

> Handling outliers Splitting the dependent and independent variables

> Splitting dataset into training and test set Feature scaling

Note: These are the general steps of preprocessing the data before using it for m achinelearning. Depending on the condition of your dataset, you may or may no t have to go through all these steps.

Handling Missing Values

● Sometimes you may find some data missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously, you cou ld
remove the entire line of data but what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common

ideas to handle the problem is to take a mean of all the values for continuous an d for categorical we make use of mode values and replace the missing data. Word “True” that the particular column has missing values, we can also see the
● count of missing values in each column by using isnull().sum function.

Handling the missing values

```
dataset.isnull().any()
```

```
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
```

As you can see that, our data do not contain any null values. Here the function i s null.any()return the Boolean values False. When that return True, which mean s that particular in dataset has missing values. So we can skip this step

Handling Outliers

With the help of boxplot, outliers are visualized (refer activity 3 univariate analy sis). Andhere we are going to find upper bound and lower bound of Na_to_K fe ature with some mathematical formula.

● To find the upper bound we have to multiply IQR (Interquartile range) with 1. 5 and
add it with 3rd quantile. To find lower bound instead of adding, subtract it with 1st quantile. Take image attached below as your reference.

● If outliers are removed, we lose more data. It will impact model performance.

● Here removing outliers is impossible. So, the capping technique is used on ou tliers.

● Capping: Replacing the outliers with upper bound values.

Note: In our Dataset all the values are in the same range, so outliers replacing is not necessary.

Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical d ata to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding tech niques.
There are several techniques but in our project, we are using feature mapping an d label encoding.
Note: In our dataset, there is no categorical data type, so we can skip this step

Splitting The Dataset Into Dependent And Independent Variables.

In machine learning, the concept of the dependent variable (y) and independent variables(x) is important to understand. Here, the Dependent variable is nothing butoutput in the dataset and the independent variable is all inputs in the dataset. We candenote with any symbol (alphabets). In our dataset, we can say that class is thedependent variable and all other columns are independent. But in order to select theindependent columns, we will be selecting only those columns which a re highly correlated and some value to our dependent column.
With this in mind, we need to split our dataset into the matrix of independent variables and the vector or dependent variable. Mathematically, Vector is define d as a matrix that has just one column.
Let's create out independent and dependent variables

splitting the dataset into dependent and independent variables

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

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

·In the above code we are creating a DataFrame of the independent variable x with ourselected columns and for the dependent variable y, we are only taking the class column.
Where DataFrame is used to represent a table of data with rows and columns.

Split The Dataset Into Train Set And Test Set

· Now split our dataset into a train set and test using train_test_split class from s ci-kit learn library.
· Train_test_split: used for splitting data arrays into training data and for testing data.

split the Dataset into Train set and Test set

```
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)
```

Feature Scaling

● Standard Scaler: Sklearn its main scaler, the StandardScaler, uses a strict defin ition ofstandardization to standardize data. It purely centers the data by using th e following formula, where u is the mean and s is the standard deviation

Feature scaling

```
#import Standardscaler
from sklearn.preprocessing import StandardScaler
#create object to standardScaler class
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.fit_transform(X_test)
```

```
from joblib import dump
dump(sc,"transform.save")
```

```
['transform.save']
```

Model Building

Now our data is cleaned and it's time to build the model. We can train our data o ndifferent algorithms. For this project we are applying four classification algorit hms. Thebest model is saved based on its performance. To evaluate the perform ance confusion matrix and classification report is used.

Decision Tree Model

A function named decision tree is created and train and test data are passed as th eparameters. Inside the function, the DecisionTreeClassifier algorithm is initiali zed andtraining data is passed to the model with the .fit() function. Test data is p redicted with thepredict() function and saved in a new variable. For evaluating t he model, a confusion matrix and classification report are done

Evaluating Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

```
model = DecisionTreeClassifier()
```

```
model.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```
from sklearn.metrics import accuracy_score,classification_report
```

```
y_predict = model.predict(X_test)
y_predict_train=model.predict(X_train)
```

Accuracy score

```
print('Test data',accuracy_score(y_test,y_predict))
print('Train data',accuracy_score(y_train,y_predict_train))
```

```
Test data 0.9655172413793104
Train data 1.0
```

Accuracy score

```
print('Test data',accuracy_score(y_test,y_predict))
print('Train data',accuracy_score(y_train,y_predict_train))
```

```
Test data 0.9655172413793104
Train data 1.0
```

Confusion matrix

pd.crosstab(y_test,y_predict)

Classification report ¶

```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        26
           1       0.75      1.00      0.86         3

    accuracy                           0.97        29
   macro avg       0.88      0.98      0.92        29
weighted avg       0.97      0.97      0.97        29
```

Random Forest Model

A function named randomForest is created and train and test data are passed as t heparameters. Inside the function, the RandomForestClassifier algorithm is initi alized andtraining data is passed to the model with the .fit() function. Test data i s predicted with thepredict() function and saved in a new variable. For evaluatin g the model, a confusion matrix and classification report are done

Random forest model

```
from sklearn.ensemble import RandomForestClassifier
```

```
model = RandomForestClassifier()
model.fit(X_train,y_train.ravel())
```

```
RandomForestClassifier()
```

```
from sklearn.metrics import accuracy_score,classification_report
```

```
y_predict = model.predict(X_test)
y_predict_train=model.predict(X_train)
```

Accuracy score

```
print('Test data',accuracy_score(y_test.ravel(),y_predict))
print('Train data',accuracy_score(y_train.ravel(),y_predict_train))
```

```
Test data 0.9655172413793104
Train data 1.0
```

Confusion matrix

```
pd.crosstab(y_test.ravel(),y_predict)
```

col_0	0	1
row_0		
0	25	1
1	0	3

Classification report

```
print(classification_report(y_test.ravel(), y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        26
           1       0.75      1.00      0.86         3

    accuracy                           0.97        29
   macro avg       0.88      0.98      0.92        29
weighted avg       0.97      0.97      0.97        29
```

KNN Model

A function named KNN is created and train and test data are passed as the para meters.Inside the function, the KNeighborsClassifier algorithm is initialized and training data ispassed to the model with the .fit() function. Test data is predicted with the .predict()function and saved in a new variable. For evaluating the mod el, a confusion matrix and classification report are done.

KNN model

```
from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier()

knn.fit(X_train,y_train.ravel())
```

```
KNeighborsClassifier()
```

```
y_predict=knn.predict(X_test)
y_pred=knn.predict(X_train)
```

```
from sklearn.metrics import accuracy_score,classification_report

print("Test accuracy=", accuracy_score(y_test.ravel(),y_predict))
print("Train accuracy=", accuracy_score(y_train.ravel(),y_pred))
```

```
Test accuracy= 0.896551724137931
Train accuracy= 0.9418604651162791
```

```
pd.crosstab(y_test.ravel(),y_predict)
```

col_0	0	1
row_0		
0	23	3
1	0	3

```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.88      0.94        26
           1       0.50      1.00      0.67         3

    accuracy                           0.90        29
   macro avg       0.75      0.94      0.80        29
weighted avg       0.95      0.90      0.91        29
```

Xgboost Model

A function named xgboost is created and train and test data are passed as the parameters. Inside the function, the GradientBoostingClassifier algorithm is initi alized andtraining data is passed to the model with the .fit() function. Test data i s predicted with thepredict() function and saved in a new variable. For evaluatin g the model, a confusion matrix and classification report are done.

Fit the model using x_train, y_train data

Xgboost model

```
from xgboost import XGBClassifier
```

```
xgb = XGBClassifier()
```

```
xgb.fit(X_train,y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

```
y_predict=xgb.predict(X_test)
y_pred=xgb.predict(X_train)
```

```
from sklearn.metrics import accuracy_score,classification_report

print("Test accuracy=", accuracy_score(y_test,y_predict))
print("Train accuracy=", accuracy_score(y_train,y_pred))
```

```
Test accuracy= 0.9655172413793104
Train accuracy= 1.0
```

```
pd.crosstab(y_test.ravel(),y_predict)
```

col_0	0	1
row_0		
0	25	1
1	0	3

```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        26
           1       0.75      1.00      0.86         3

    accuracy                           0.97        29
   macro avg       0.88      0.98      0.92        29
weighted avg       0.97      0.97      0.97        29
```

```
from sklearn import tree
from sklearn import ensemble
from sklearn import neighbors
import xgboost
```

```
dtree = tree.DecisionTreeClassifier()
Rf = ensemble.RandomForestClassifier()
knn = neighbors.KNeighborsClassifier()
xgb = xgboost.XGBClassifier()
```

```
dtree = tree.DecisionTreeClassifier()
Rf.fit(X_train,y_train.ravel())
knn.fit(X_train,y_train.ravel())
xgb.fit(X_train,y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

Compare The Model

For comparing the above four models compare model function is defined.

compare the model

```
: from sklearn import metrics

: print(metrics.accuracy_score(y_test,y_predict))
  print(metrics.accuracy_score(y_test,y_predict))
  print(metrics.accuracy_score(y_test,y_predict))
  print(metrics.accuracy_score(y_test,y_predict))

  0.9655172413793104
  0.9655172413793104
  0.9655172413793104
  0.9655172413793104
```

After calling the function, the results of models are displayed as output. From th e fourmodel Decision tree, random forest and xgboost are performing well. Fro m the belowimage, we can see the accuracy of the models. All three models hav e 96.55% accuracy

Evaluating Performance Of The Model

● After comparing all the three models with different attributes ,xgboost is the b etter model,so we will save this mode

evaluating the performance of the model

```
: metrics.confusion_matrix(y_test,y_predict)

: array([[25,  1],
        [ 0,  3]], dtype=int64)

: print(metrics.accuracy_score(y_test,y_predict))

  0.9655172413793104

: print(metrics.precision_score(y_test,y_predict))

  0.75

: print(metrics.recall_score(y_test,y_predict))

  1.0
```

Saving The Model

● Joblib of save is used for serializing and deserializing Python object structures , alsocalled marshalling or flattening. Serialization refers to the process of conve rting anobject in memory to a byte stream that can be stored on dissk or sent ove r a network. Later on, this character stream can then be retrieved and de-

serialized back to a Python object.
● Save our model by importing joblib dump class.

saving the model

```
from joblib import dump
```

```
dump(xgb,'floods.save')
```

```
['floods.save']
```

Build Flask Application

In this section, we will be building a web application that is integrated into the model webuilt. A UI is provided for the uses where he has to enter the values fo r predictions. Theenter values are given to the saved model and prediction is sho wcased on the UI. This section has the following tasks

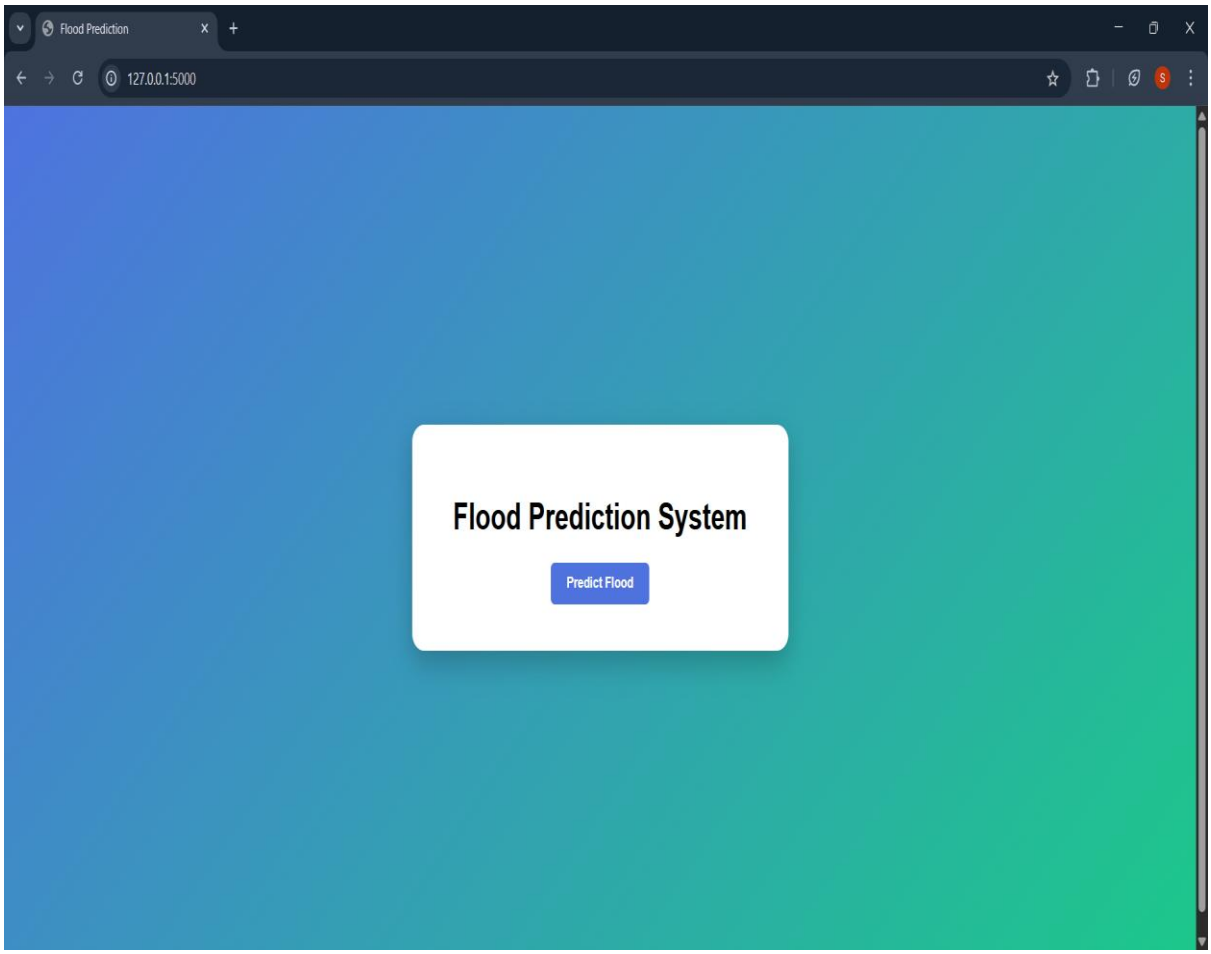
· Building HTML Pages Building serverside script Building HTML Pages
· Flask Frame Work with Machine Learning Model In this section we will be b uilding aweb application which is integrated to the model we built.

An UI is provided for the useswhere he has to enter the values for predictions. The enter values are given to the saved
model and prediction is showcased on the UI.
· Previously we are saved this file as “floods.save”.

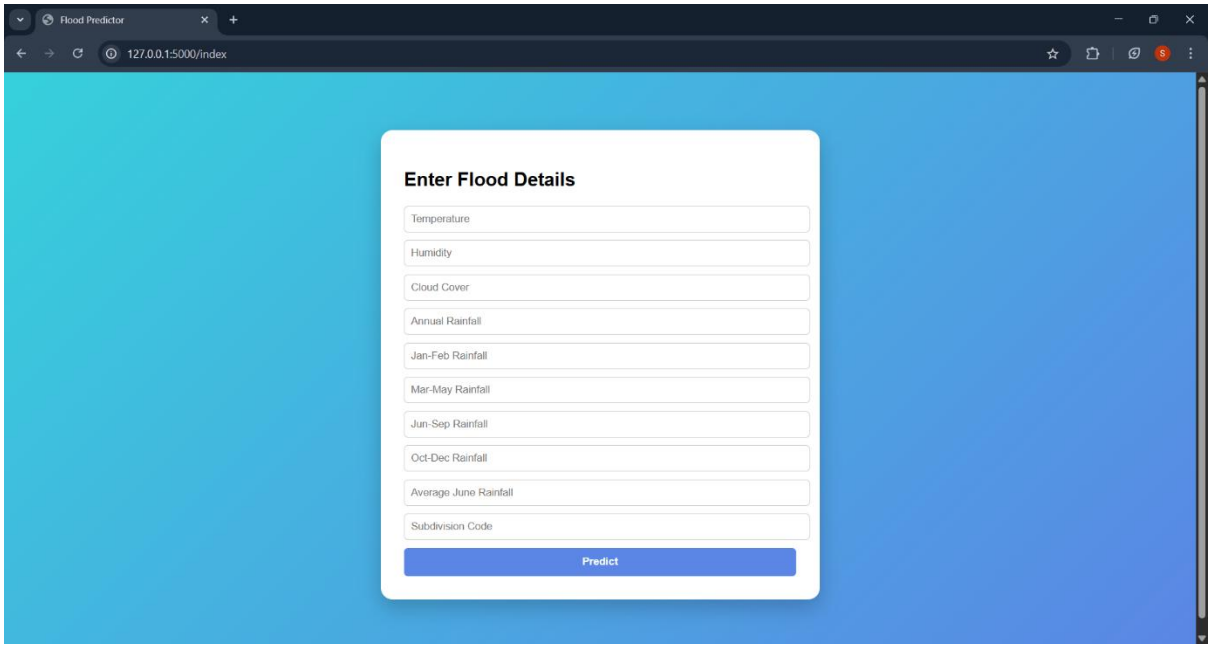
We have 5 independent variables and one dependent variable for this model.
· To build this you should know the basics of “HTML, CSS, Bootstrap, flask fra mework and python” Create a project folder that should contain.
· A python file called app.py. Model file (floods.save).
Templates folder which contains index.HTML file.
Static folder which contains CSS folder which contains styles.css.
We use HTML to create the front-end part of the web page.

Home Page Figure :

Home Page of Flood Prediction System



User Input Page



Build Python Code

● Let us build flask file ‘app.py’ which is a web framework written in python fo r server-side scripting. Let’s see step by step procedure for building the backend application.

● App starts running when “__name__” constructor is called in main.

● render_template is used to return HTML file.

● “GET” method is used to take input from the user.

● “POST” method is used to display the output to the user.

● We will be using python for server side scripting. Let’s see step by step proces s for writing backend code. Importing Libraries

● Importing the flask module in the project is mandatory. An object of the Flask class

is our WSGI application. Flask constructor takes the name of the current modul e (__name__) as argument Pickle library to load the model file.

```
from flask import Flask, render_template, request
from joblib import load
```

Create Flask app and Load our model file

```
app = Flask(__name__)

model = load('floods.save')
sc = load('transform.save')
```

Routing to the HTML Page

● Here we will be using the declared constructor to route to the HTML page tha t we have created earlier.

● In the above example, the ‘/’ URL is bound with the home.html function. Hen ce,

when the home page of the webserver is opened in the browser, the HTML page is rendered. Whenever you click on the Intro button from the home page, the intro.html page will be rendered

```
@app.route('/')
def home():
    return render_template('home.html')
@app.route('/predict')
def index():
    return render_template("index.html")
```

We are routing the app to the HTML templates which we want to render. Firstly we arerendering the “home.html” template which is the home page to our web UI. Where it will display two options, one is Home and Predict Floods.
When you click on Predict Floods, it redirects to the next page which is bounde d with URL/predict. At that time index.html page will be rendered. Where to ha ve to fill in all the details and get the result on the prediction page.

Route the prediction on UI

predict() – is taking the values from the prediction page and storing it into a vari able and then we
are creating a DataFrame along with the values and 5 independent features and f inally, we are
predicting the values using or loaded model which we build and storing the outp ut in a variable and returning it to the result page

```
# Prediction logic
@app.route('/data_predict', methods=['POST'])
def predict():
    try:
        temp = float(request.form['temp'])
        humidity = float(request.form['humidity'])
        cloud = float(request.form['cloud'])
        annual = float(request.form['annual'])
        janfeb = float(request.form['janfeb'])
        marmay = float(request.form['marmay'])
        junsep = float(request.form['junsep'])
        octdec = float(request.form['octdec'])
        avgjune = float(request.form['avgjune'])
        sub = float(request.form['sub'])
```

Main Function

- This is used to run the application in localhost

Run The Application

- Open anaconda prompt from start menu.
- Navigate to the folder where your app.py resides.

● Now type the “python app.py” command.
● It will show the local host where your app is running on http://127.0.0.1.5000/
● Copy that local host URL and open that URL in browser. It does navigate you to the where you can view your web page.

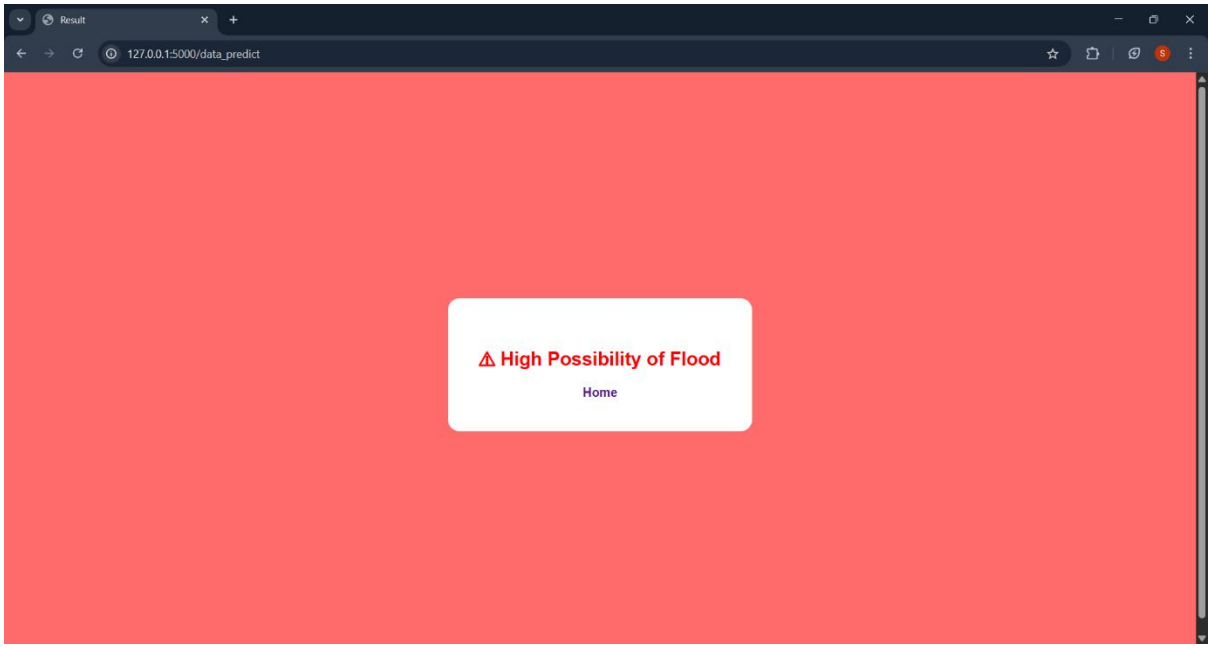
```
PS C:\Users\yogesh\OneDrive\Desktop> cd smartinterzproject
PS C:\Users\yogesh\OneDrive\Desktop\smartinterzproject> python app.py
C:\Users\yogesh\AppData\Local\Programs\Python\Python310\lib\pickle.py:1718: UserWarning: [22:30:59] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\gbm
\../common/error_msg.h:83: If you are loading a serialized model (like pickle in Python, RDS in R) or
configuration generated by an older version of XGBoost, please export the model by calling
`Booster.save_model` from that version first, then load it back in current version. See:

    https://xgboost.readthedocs.io/en/stable/tutorials/saving_model.html

for more details about differences between saving model and serializing.

  setstate(state)
C:\Users\yogesh\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:442: InconsistentVersionWarning: Trying to unpickle estimator Stan
dardScaler from version 1.6.1 when using version 1.7.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refe
r to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Navigate to the localhost (http://127.0.0.1:5000/) where you can view your web page.
Click on Predict Floods and submit the values to get the floods prediction result on the web page

output:

High Flood Risk Prediction

Figure : High Flood Risk Prediction Output



No Flood Risk Prediction Figure :

No Flood Risk Prediction Output