In partial fulfilment for the award of "BACHELOR OF ENGINEERING".

GOVERNMENT COLLEGE OF ENGINEERING, THANJAVUR – 613402

BE - ELECTRONICS AND COMMUNICATION ENGINEERING

NATURAL DISASTER INTENSITY ANALYSIS AND CLASSIFICATION USING ARTIFICIAL INTELLIGENCE

A PROJECT REPORT

(SUBMITTED BY)

TEAM ID: PNT2022TMID47258

TEAM MEMBERS	REGISTER NUMBER
BALAJI M	822719106009
BALASUBRAMANIAM GOVINDHARAJ	822719106010
NAGARATHINAM M	822719106025
SANJAYVIKRAM M	822719106034

INDUSTRY MENTOR(S): Swathi

Faculty Mentor(s) : G R Annushakumar

TABLE OF CONTENTS

1. INTRODUCTION

- Improve the understanding of disaster risk, hazards, and vulnerabilities
- Strengthen disaster risk governance at all levels from local to centre
- Invest in disaster risk reduction for resilience through structural, non structural and financial measures, as well as comprehensive capacity development
- Enhance disaster preparedness for effective response
- Promote “Build Back Better” in recovery, rehabilitation and reconstruction
- Prevent disasters and achieve substantial reduction of disaster risk and losses in lives, livelihoods, health, and assets (economic, physical, social,cultural and environmental)

1.1 PROJECT OVERVIEW

Natural disasters are large-scale geological or meteorological events that have the potential to cause loss of life or property. A disaster is a result of a natural or man-made hazard impacting a vulnerable community. It is the combination of the hazard along with exposure of a vulnerable society that results in a disaster. The project aims at building a deep learning model that can classify and tell the intensity of a natural disaster based on images.The project uses a multilayered deep convolutional neural network as the main model architecture and also it uses various techniques to enhance the model

performance and robustness such as data augmentation, transfer learning, and ensemble methods. The project can have various applications and use cases for disaster management and response such as providing timely and accurate information, assessing the damage and impact, and facilitating the recovery and reconstruction.

1.2 PURPOSE

The purpose of natural disaster intensity analysis and classification using AI is to build a deep learning model that can classify and tell the intensity of a natural disaster based on images. This can help to overcome losses in ecosystems, human lives, and properties by providing timely and accurate information for disaster management and response. It can also be integrated with other technologies such as geographic information systems, remote sensing, and social media to provide a comprehensive and multidimensional view of the disaster situation and impact.

1.3 PREREQUISITES:

Hardware Specifications:

- Windows (minimum 10), Mac & Linux
- Ram - 4GB (minimum)
- Hard Disk - 100GB (minimum)
- Processor - Intel i3 (minimum), Mac M1

Software Specifications:

- Anaconda Navigator - https://www.anaconda.com/products/distribution
- Jupyter notebook.
- Google Colab - https://colab.research.google.com/
- Spyder / VS Code / Pycharm

IBM:

● IBM Account Creation - https://vimeo.com/742609168/1824d26a5b (Follow this video for IBM

Skill Build Account Creation)

● IBM Skill Build - https://www.ibm.com/academic/home

● Webmail - https://sg2plmcpnl492529.prod.sin2.secureserver.net:2096/

● IBM Cloud - https://cloud.ibm.com/login

2. LITERATURE SURVEY

2.1 EXISTING PROBLEMS

There is no standardized method for estimating tropical cyclone intensity. The low-pressure system developing over Bay of Bengal and South East Asian region makes a landfall and often these cyclone causes life loss, property loss. Due to flood many life losses occurs because of not giving any forecast or intimation about flood.

2.2 REFERENCES

TITLE: Tropical Cyclone Intensity Estimation Using Multidimensional Convolutional Neural Network from Multichannel Satellite Imaginary

AUTHOR: Wei Tian, Xinxin Zhou, Wei Huang, Yonghong Zhang, Pengfei Zhang, Shaofeng Hao

ABSTRACT:

Estimating tropical cyclone (TC) intensity is the first step in the processes of monitoring and predicting destructive TC disasters. Due to the dilemma of meteorological methods, accurate estimation of TC intensity is a

Longterm challenge. In recent years, while deep learning methods have been applied to TC intensity estimation, most of them fail to make full use of multichannel satellite imageries to consider the three-dimensional (3-D) structure of TC. In this letter, we propose a novel deep learning model (3DAttentionTCNet) to overcome this shortcoming. The model can automatically extract 3-D environment information related to TC intensity from multichannel satellite observation imageries such as infrared (IR), water vapor (WV), and passive microwave rain rate (PMW) satellite imageries by 3-D convolution. In addition, we employ the convolutional block attention module (CBAM) to simulate visual attention for strengthening the model's attention to core cloud structure and important channels. The experimental results show that the root-mean-square error (RMSE) of the proposed model is 9.48 kts, which is improved by 25% compared to that of the advanced Dvorak technique (ADT) and by 9.2% over that of the traditional deep learning method of TC intensity estimation.

TITLE: Vulnerability analysis of cyclone hazards and Dimension of disaster risk management in Odisha Along the east coast of India

AUTHOR: Jitendra Kumar Behera and Gopal Krishna Panda Dept. of Geography, Utkal University Vani Vihar, Bhubaneswar – 751004 Odisha India

ABSTRACT:

Odisha is one of the most vulnerable states for the hazards of the tropical cyclones along the east coast of India since time immemorial. The low-pressure systems developing over the Bay of Bengal and South East Asian region makes a landfall along the Odisha coast and travel inland. Very often

these cyclonic hazards had turned in to disasters affecting the life, livelihood and property of the people. Strong wind, torrential rain, flooding and unusual storm surges accompanied with the cyclones cause severe devastations with the destruction of dwellings, damage to infrastructure and standing crops besides loss of life along the track of its movement and adjacent areas. Odisha's exposure to these extreme events, people's perception and human response, adaptations, its risk mitigation and management has undergone a sea change in the twenty-first century keeping at pace with the scientific innovations and international guidelines. This study makes an attempt to assess the vulnerability of the state to the tropical cyclones based on a Disaster Risk Index. Time series and spatial analysis is used to study their trend and impacts. Content analysis is used to study the innovative strategies of disaster risk reduction of achieving the zero casualty as per the Sendai framework and community resilience. The findings of the study indicate an increasing vulnerability of the state to a greater number of severe cyclones. But however, the revised strategies in crisis management and community-based disaster preparedness have been the key to the success in reducing disaster risk in the state.

TITLE: Desigining Deep-Based Learning Flood Forecast Model with Comvest Hybrid Algorithm

AUTHOR: Mohammed Meishin School of Science, University of Southern Queensland Springfield, Springfield, QLD, Australia

ABSTRACT:

Efficient, robust, and accurate early flood warning is a pivotal decision support tool that can help save lives and protect the infrastructure in natural disasters. This research builds a hybrid deep learning (Comvest) algorithm integrating the predictive merits of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) Network to design and evaluate a flood forecasting model to forecast the future occurrence of flood events. Derived from precipitation dataset, the work adopts a Flood Index (IF), in form of a mathematical representation, to capture the gradual depletion of water resources over time, employed in a flood monitoring system to determine the duration, severity, and intensity of any flood situation. The newly designed predictive model utilizes statistically significant lagged I F, improved by antecedent and real-time rainfall data to forecast the next daily I F value. The performance of the proposed ConvLSTM model is validated against 9 different rainfall datasets in flood prone regions in Fiji which faces flood-driven devastations almost annually. The results illustrate the superiority of ConvLSTM-based flood model over the benchmark methods, all of which were tested at the 1-day, 3-day, 7-day, and the 14-day forecast horizon. For instance, the Root Mean Squared Error (RMSE) for the study sites were 0.101, 0.150, 0.211 and 0.279 for the four forecasted periods, respectively, using ConvLSTM model. For the next best model, the RMSE values were 0.105, 0.154, 0.213 and 0.282 in that same order for the four forecast horizons. In terms of the difference in model performance for individual stations, the Legate-McCabe Efficiency Index (LME) were 0.939, 0.898, 0.832 and 0.726 for the four forecast horizons, respectively. The results demonstrated practical utility of ConvLSTM in accurately forecasting I F and its potential use in disaster management and risk mitigation in the current phase of extreme weather events.

2.3 PROBLEM STATEMENT DEFINITION

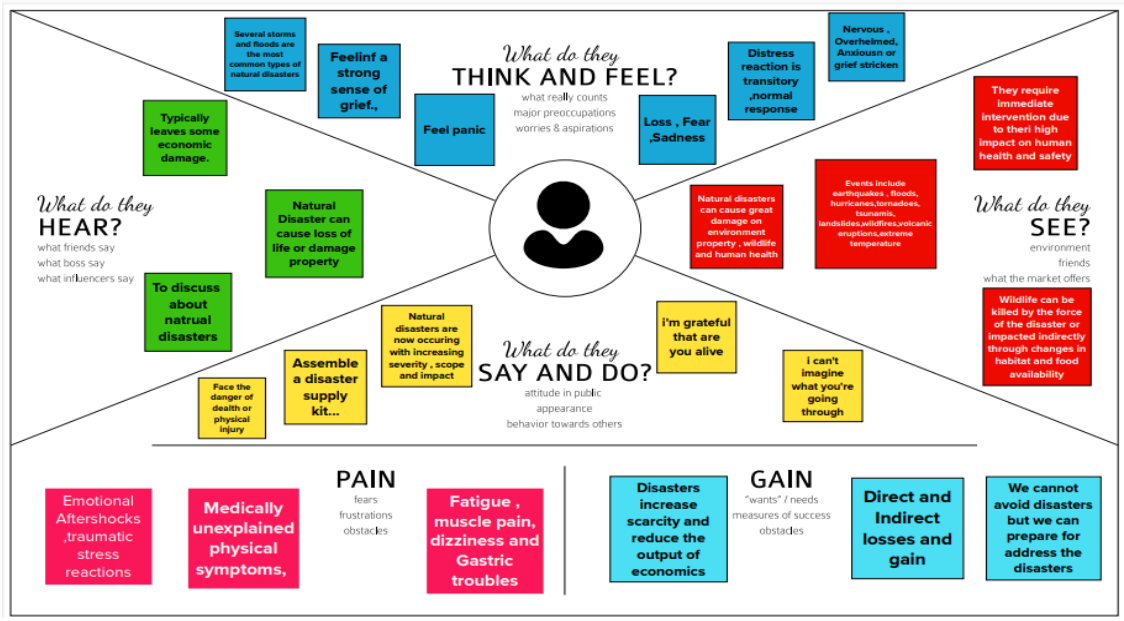
A problem statement is a concise description of an issue to be addressed or a condition to be improved upon. It identifies the gap between the current (problem) state and desired (goal) state of a process or product.

Problem Statement (PS)	Iam (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	An employee from NDRF	Classify the type of natural disaster	It is difficult to identify	The data is of images and images are of various forms	Tensed and frustrated
PS-2	A student	classify the type of natural disaster in order to learn about that disaster	It is difficult to identify	The data is of images and images are of various forms	Tensed and frustrated

3. IDEATION AND PROPOSED SOLUTION

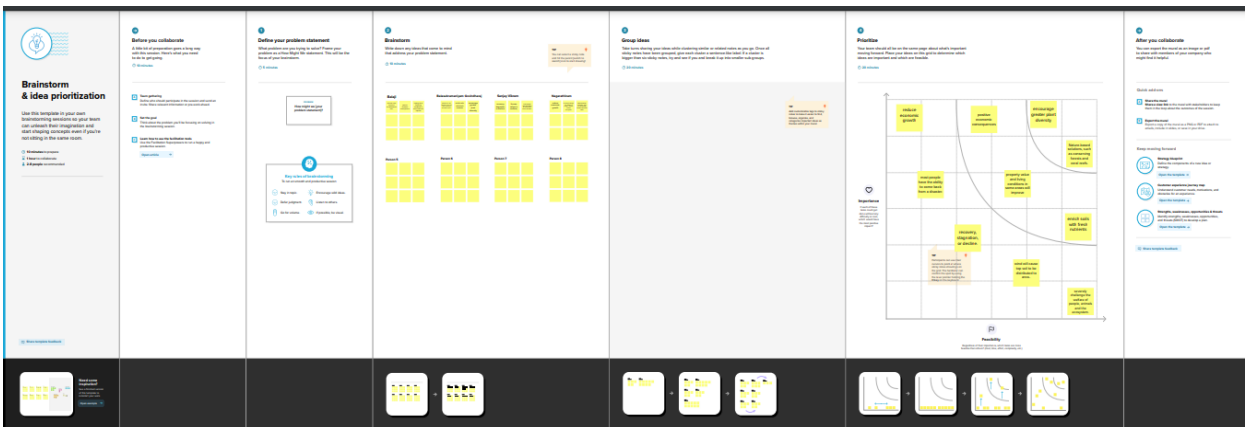
3.1 EMPATHY MAP CANVAS

An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment.



3.2 IDEATION AND BRAINSTORMING

Brainstorming is a group creativity technique by which efforts are made to find a conclusion for a specific problem by gathering a list of ideas spontaneously contributed by its members.



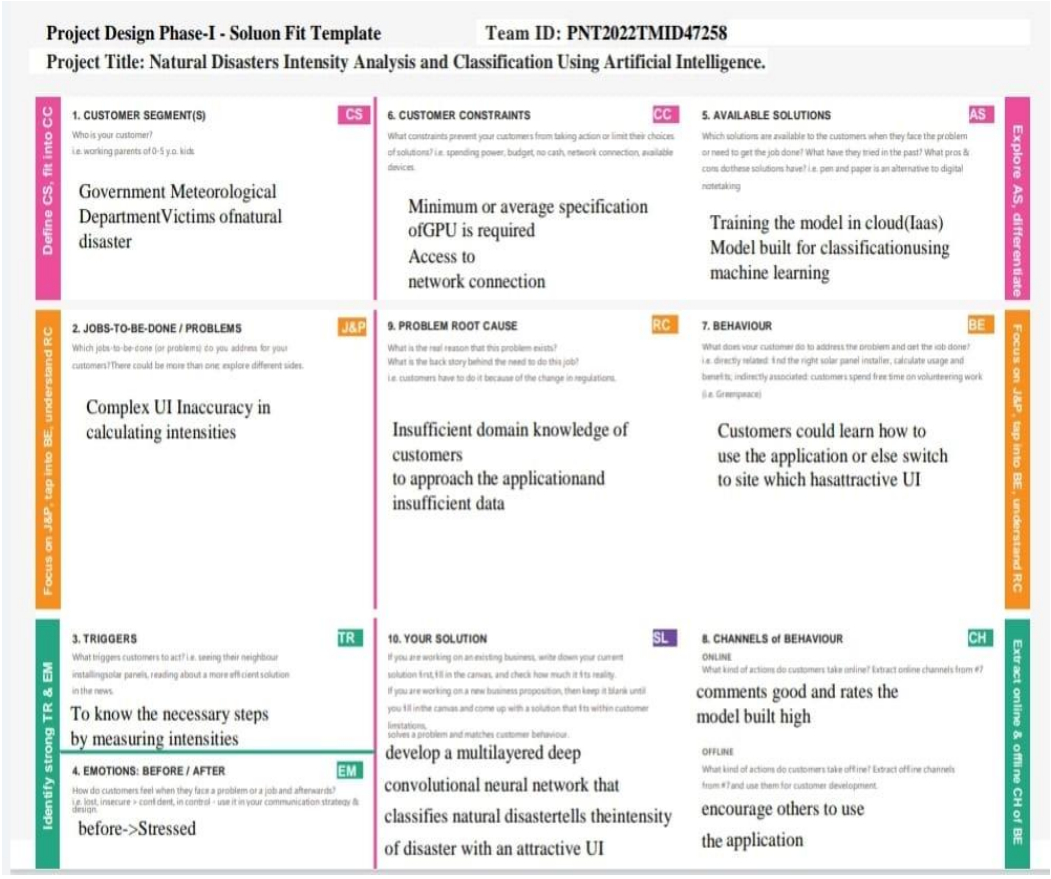
3.3 PROPOSED SOLUTION

Proposed solution should relate the current situation to a desired result and describe the benefits that will accrue when the desired result is achieved.

S.No.	Parameter	Description
1	Problem Statement (Problem to be solved)	The effects of cyclone are as it affects the agriculture, loss of livelihood of coastal areas, loss of communication.
2	Idea / Solution description	As of the intensity of cyclone and range of rainfall inform the people of coastal areas for their protect on and also for them agricultural crops.
3	Novelty / Uniqueness	Till now no indica on to Farmers, we are planning to indicate to Farmers about the climax c condions in simple way.
4	Social Impact / Customer Satisfaction	It will be helpful for farmers for protecting their crops.
5	Business Model (Revenue Model)	We will introduce an app to solve the problem (All the cyclone issues shown in the app) and keep posting ads for this app to earn source.
6	Scalability of the Solution	As the product we created is of user friendly and it will be very useful for farmers and agriculture.

3.4 PROBLEM SOLUTION FIT

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem.



4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

The following are the functional requirements of the proposed solution.

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Help Desk	User should be able to get guidance from the customer care
FR-4	Management	Administration must collect new datasets and keep the model trained
FR-5	User authentication	Verify the user

4 .2 NON-FUNCTIONAL REQUIREMENTS

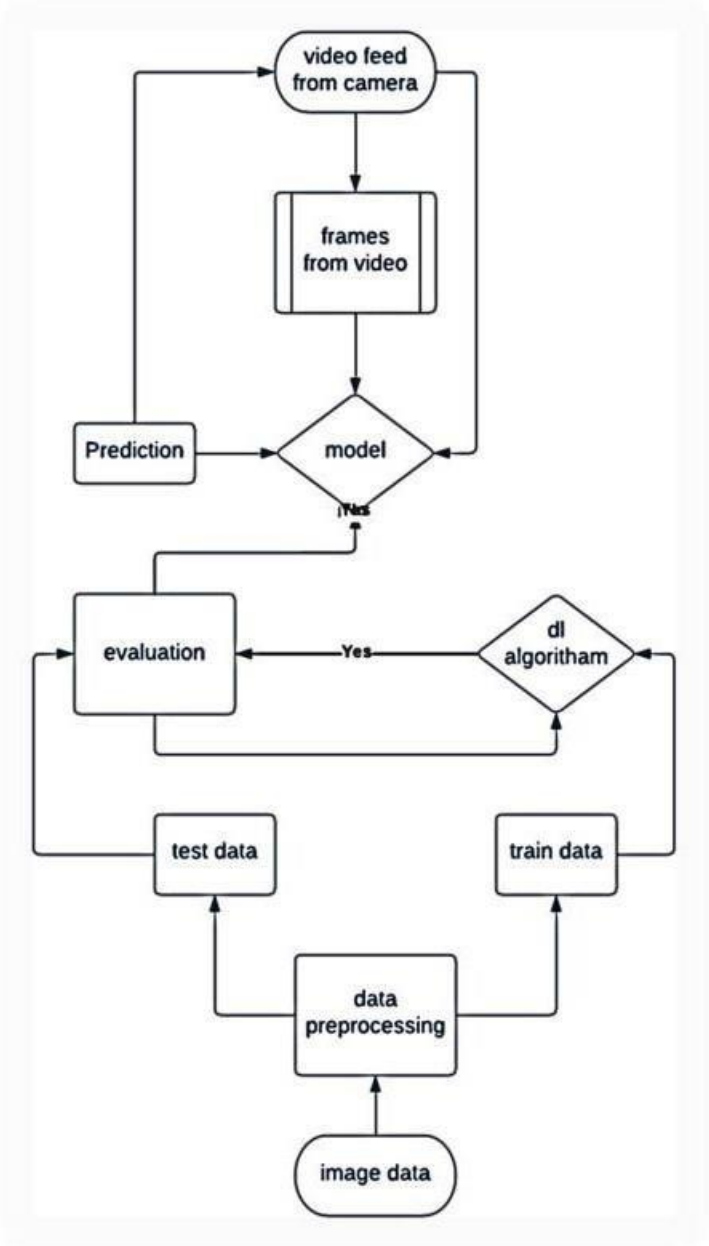
The following are the non-functional requirements of the proposed

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The system must be efficient and easy for the user to carry out tasks.
NFR-2	**Security**	User details must be secured.
NFR-3	**Reliability**	The output produced should be reliable to the users.
NFR-4	**Performance**	The system should be able to handle many userswithout performance deterioration.
NFR-5	**Availability**	The system should be accessible to a user at a given point in time
NFR-6	**Scalability**	The website pages should load fast with thetotal number of simultaneous users.

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAM

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself.



5.2 USER STORIES

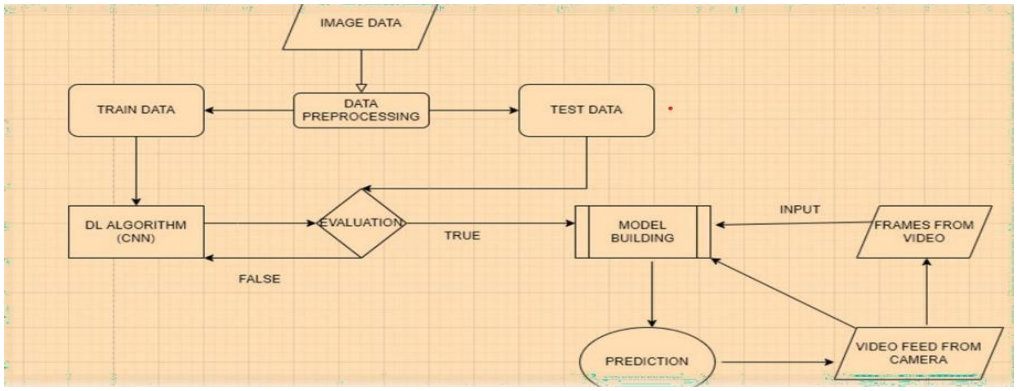
A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer. The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	LOGIN	USN-1	As a farmer, I can login by giving mobile number, gmail or google account and their location.	I can prepare myself from cyclone and storing enough food and essentials	High	Sprint- 1
	ALERT	USN-2	As a farmer, I can receive the alert message when the cyclone hits.	I can know about current climatic conditions and upcoming weather conditions	High	Sprint- 2
	MONITORING	USN-3	As a farmer, I can view the continuous monitoring of cyclone and climatic changes.	I can know where the cyclone hits and how much impacts it may create	High	Sprint- 3
	REPORTS	USN-4	As a farmer, I can keep the records of the previous cyclone and refer news from meteorologist for live updation.	I can receive the alert messages when the disaster occurs	High	Sprint- 4
	END USERS (farmers)	USN-5	As a farmer, I can receive the information from the database.	I should ensure that any stored seeds or harvested crops are carefully protected from wind and flooding	High	Sprint- 5

5.3 SOLUTION AND TECHNICAL ARCHITECTURE

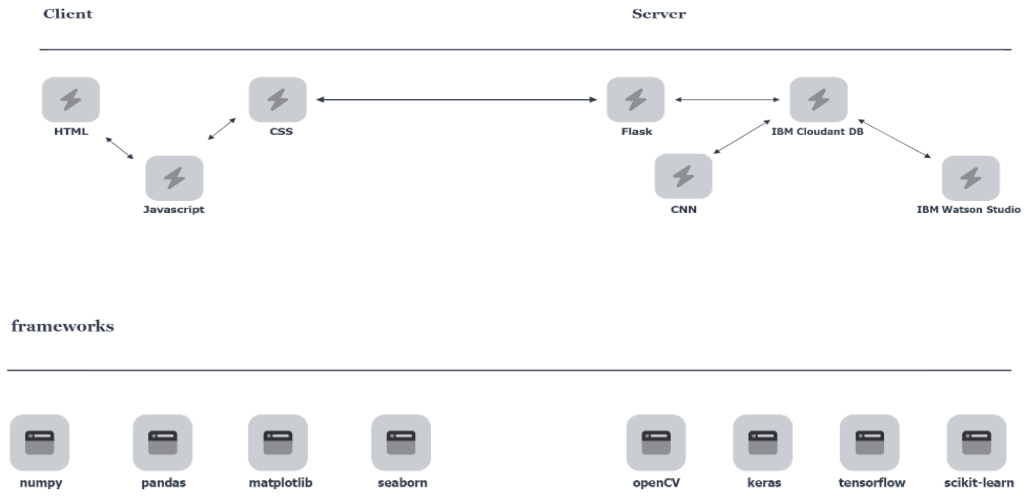
SOLUTION ARCHITECTURE :

A solution architecture (SA) is an architectural description of a specific solution. SAs combine guidance from different enterprise architecture viewpoints (business, information and technical), as well as from the enterprise solution architecture (ESA).



TECHNOLOGY STACK :

A tech stack is the combination of technologies a company uses to build and run an application or project. Sometimes called a “solutions stack,” a tech stack typically consists of programming languages, frameworks, a database, front-end tools, back-end tools, and applications connected via APIs.



6. PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I Collecting data from trusted sources, in addition to collecting analysis.	2	High	Balaji M SanjayVikram M BalaSubramaniam Govindharaj Nagarathinam M
Sprint-1		USN-2	As a user, I Filtering of demographic information, as well as filtering of countries , region, state ,or province with cases of disaster	1	High	Balaji M SanjayVikram M BalaSubramaniam Govindharaj Nagarathinam M
Sprint-2		USN-3	As a user, I Counting, globally or from a specific location ,of confirmed cases, Recovered and deaths by Disaster	2	Low	Balaji M SanjayVikram M BalaSubramaniam Govindharaj
Sprint-1		USN-4	As a user, I can register for the application through maps	2	Medium	Balaji M SanjayVikram M
Sprint-1	Login	USN-5	As a user, I can log into the application by entering geographic panel	1	High	BalaSubramaniam Govindharaj Nagarathinam M
Sprint-2	Dashboard	USN-6	As a user, I Display of maps, histograms, or an interactive geographic panel	1	High	SanjayVikram M BalaSubramaniam Govindharaj
Sprint-2	Importing and Exporting data	USN-7	As a user, I Exporting results, data, or information in CSV or JSON format, as well as importing data from CSV files	3	High	Balaji M SanjayVikram M BalaSubramaniam Govindharaj Nagarathinam M
Sprint-3	Show orientation	USN-8	As a user, I Displaying Disaster prevention tips, a page with information on how to protect itself , travel tips,	4	Low	Nagarathinam M SanjayVikram M
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
			emergency contacts ,link toweb sites with import an information about the AI			
Sprint-4	Data update	USN-9	As a user I Updating information, spreadsheets, list of recovered patients, news page, and daily statistics	3	Medium	BalaSubramaniam Govindharaj Balaji M
Sprint-4	Responsiveness	USN-10	As a user I , Terms of supporting the phases of disaster management, it was observed that the repositories focused only on the response phase.	4	High	Balaji M SanjayVikram M BalaSubramaniam Govindharaj Nagarathinam M
Sprint-2	Risk Management	USB-11	As a user I, Raise risk culture and awareness and avoid any risk situations by eliminating risky practices	5	High	Balaji M SanjayVikram M BalaSubramaniam Govindharaj Nagarathinam M
Sprint-2	Communication Management	USB-12	As a user I , Timely involvement of the community and sharing ideas , hands-on experiences	4	High	Balaji M SanjayVikram M BalaSubramaniam Govindharaj
Sprint-3	Time, Cost, Scope and Quality	USB-13	As a user I, Keep the balance of these project variables , taking into account that in emergency situation priorities shift lot from normal everyday project.	4	Low	SanjayVikram M BalaSubramaniam Govindharaj Nagarathinam
Sprint-4	Project Integration management	USB-14	As a user I , Coordinate and integrate several alternative initiatives .	6	Medium	Balaji M SanjayVikram M BalaSubramaniam Govindharaj Nagarathinam M

6.2 SPRINT DELIVERY SCHEDULE

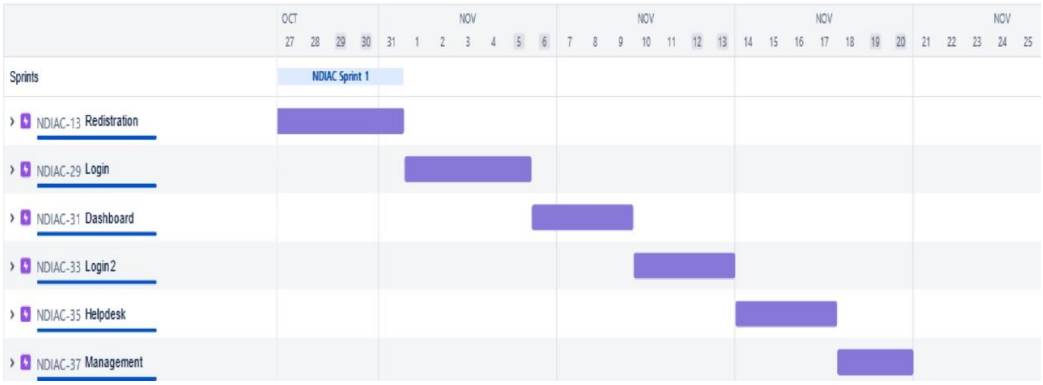
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	28 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	30	04 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	35	11 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	15	17 Nov 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

6.3 REPORTS FROM JIRA

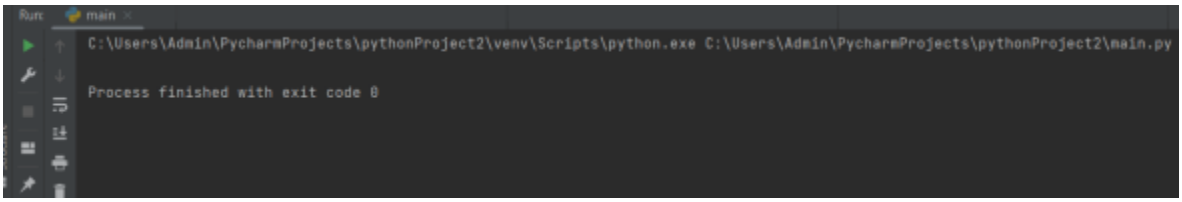


7. FEATURE

7.1: CYCLONE PYTHON

```
def get_nasa_eonet(count_search=int):
TEST_SPEC_TARGET_URL="https://eonet.gsfc.
nasa.gov/api/v3/events"
 try:
 READ_URL=requests.get(TEST_SPEC_TARGET_
URL)
 READ_JSON = READ_URL.json()
 EVENTS_JSON = READ_JSON["events"]
 for x_range in range(count_search):
 EVENT_TITLE = EVENTS_JSON[x_range]["title"]
 EVENT_DATE =
EVENTS_JSON[x_range]["geometry"][0]["date"
]
 EVENTS_LAT =
EVENTS_JSON[x_range]["geometry"][0]["coord
inates"][1]
 EVENTS_LON =
EVENTS_JSON[x_range]["geometry"][0]["coord
inates"][0]
 time.sleep(0.5)
 print("\n")
 print("TITLE: ",EVENT_TITLE)
 print("DATE: ",EVENT_DATE)
 print("LATITUDE: ",EVENTS_LAT)
 print("LONGITUDE: ",EVENTS_LON)
 print("--"*10)
 except:
 print("\n")
```

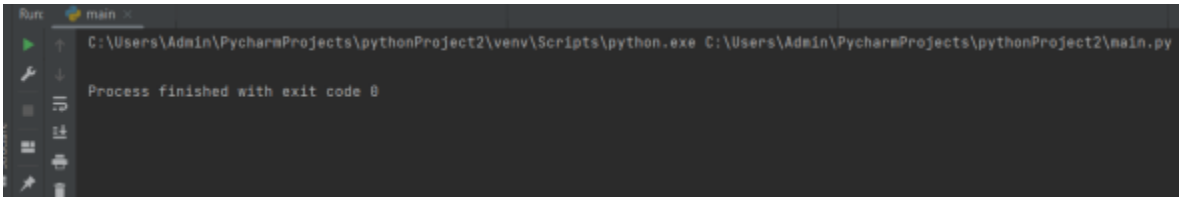
```
 print("THERE IS A CONNECTION PROBLEM, IT
MAY BE ABOUT YOUR INTERNET CONNECTION
OR DATABASE")
 time.sleep(0.5)
 print("PLEASE CHECK YOUR CONNECTION AND
TRY AGAIN")
 print("\n")
 time.sleep(0.5)
```



7.2 EARTHQUAKE

```
def check_earthquakes_location(lat_out=float,lon_out=float):
try:
 TARGET_REQ_URL = "https://www.emsc-csem.org/service/rss/rss.php?typ=emsc"
 REQ_TARGET = requests.get(TARGET_REQ_URL).text
 SOUP_TARGET = BeautifulSoup(REQ_TARGET,"html.parser")
 FIND_ALL_IT = SOUP_TARGET.find_all("item")
 checking_value = 0
 print("\n")
time.sleep(1.2)
 print("CONNECTED PORTAL I")
for x_loop in FIND_ALL_IT:
TITLE_OUT = x_loop.find("title")
LAT_OUT = x_loop.find("geo:lat")
LON_OUT = x_loop.find("geo:long")
DEP_OUT = x_loop.find("emsc:depth")
MAG_OUT = x_loop.find("emsc:magnitude")
TIME_OUT = x_loop.find("emsc:time")
ST_OUT = x_loop.find("status")
LAT_OUT = float(LAT_OUT.text)
```

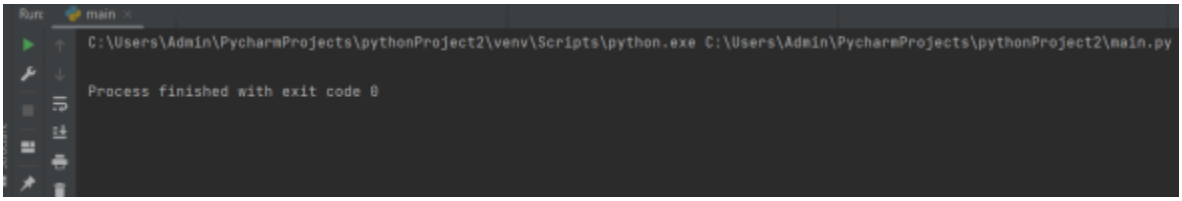
```
LON_OUT = float(LON_OUT.text)
if lat_out == LAT_OUT and lon_out == LON_OUT:
time.sleep(0.5)
print("\n")
print("TITLE: ",TITLE_OUT.text)
print("LATITUDE: ",LAT_OUT)
print("LONGITUDE: ",LON_OUT)
print("DEPTH: ",DEP_OUT.text)
print("MAGNITUDE: ",MAG_OUT.text)
print("DATE: ",TIME_OUT.text)
print("STATUS: ",ST_OUT.text)
print("\n")
checking_value += 1
elif "%.2f"%float(lat_out) == "%.2f"%float(LAT_OUT) and "%.2f"%float(lon_out) ==
"%.2f"%float(LON_OUT):
time.sleep(0.5)
print("\n")
print("TITLE: ",TITLE_OUT.text)
print("LATITUDE: ",LAT_OUT)
print("LONGITUDE: ",LON_OUT)
print("DEPTH: ",DEP_OUT.text)
print("MAGNITUDE: ",MAG_OUT.text)
print("DATE: ",TIME_OUT.text)
print("STATUS: ",ST_OUT.text)
print("\n")
checking_value += 1
```



FLOODS:

```
def get_flood(count_search=int):
try:
        GDACS_TARGET = requests.get("https://www.gdacs.org/default.aspx").text
        SOUP_GDACS = BeautifulSoup(GDACS_TARGET,"html.parser")
        ALL_F_DISASTER = SOUP_GDACS.find_all("div",id="mainListFl")
        CONTROL_VALUE_LIST = []
 i_count_stop = 0
 for x_att in ALL_F_DISASTER:
        ALERT_DETAIL_LINK = x_att.find_all("a")
 for x_detail_link in ALERT_DETAIL_LINK:
        LINK_AFTER_SITE = str(x_detail_link.get("href"))
        SUB_TARGET = requests.get(LINK_AFTER_SITE).text
        SOUP_GDACS_FUNCTION = BeautifulSoup(SUB_TARGET,"html.parser")
        SUB_TARGET_SOUP                                                    =
SOUP_GDACS_FUNCTION.find_all("div",id="alert_summary_left")
 if i_count_stop <= count_search:
 i_count_stop += 1
 for x_sub_target in SUB_TARGET_SOUP:
 FIND_TR_ALL = x_sub_target.find_all("td")
for x_sub_td in FIND_TR_ALL:
        ALL_INFO_TEXT = x_sub_td.text
       CONTROL_VALUE_LIST.append(ALL_INFO_TEXT.replace("\n","").replace("\n",""))
       DE_TAR = CONTROL_VALUE_LIST[3]
       DI_TAR = CONTROL_VALUE_LIST[5]
       LOC_TAR = CONTROL_VALUE_LIST[7]
       DATE_TAR_F = CONTROL_VALUE_LIST[9]
 time.sleep(0.5)
```

```
print("\n")
print("DEATH: ",DE_TAR)
print("DISPLACED: ",DI_TAR)
print("LOCATION: ",LOC_TAR)
print("DATE: ",DATE_TAR_F) print("--"*10)
CONTROL_VALUE_LIST = []
except:
print("\n")
print("THERE IS A CONNECTION PROBLEM, IT MAY BE ABOUT YOUR INTERNET CONNECTION OR DATABASE")
time.sleep(0.5)
print("PLEASE CHECK YOUR CONNECTION AND TRY AGAIN")
print("\n")
time.sleep(0.5)
```



HTML CODE

```
<!DOCTYPE html>
<html lang=”en”>
<title>Home – Natural Disasters Database</title>
<meta charset=”UTF-8”>
<meta name=”viewport” content=”width=device-width, initial-scale=1”>
<link rel=”stylesheet” href=https://www.w3schools.com/w3css/4/w3.css>
<link rel=”stylesheet” href=https://fonts.googleapis.com/css?family=Lato>
<link rel=”stylesheet” href=https://fonts.googleapis.com/css?family=Montserrat>
<link rel=”stylesheet” href=https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/fontawesome.min.css>
<style>
Body,h1,h2,h3,h4,h5,h6 {font-family: “Lato”, sans-serif}
```

```
.w3-bar,h1,button {font-family: “Montserrat”, sans-serif}
.fa-anchor,.fa-coffee {font-size:200px}
</style>
<body>
<!—Navbar →
<div class=”w3-top”>
 <div class=”w3-bar w3-black w3-card w3-left-align w3-large”>
 <a class=”w3-bar-item w3-button w3-hide-medium w3-hide-large w3-right w3-padding-large w3-
hover-white w3-large w3-red” href=”javascript:void(0);” onclick=”myFunction()” title=”Toggle
Navigation Menu”><i class=”fa fa-bars”></i></a>
 <a href=”{% url ‘home’ %}” class=”w3-bar-item w3-button w3-hide-small w3-padding-large w3-hoverwhite”>Home</a>
 <a class=”w3-bar-item w3-button w3-padding-large w3-white”>Earthquake</a>
 <a href=”{%url ‘tsunami’%}” class=”w3-bar-item w3-button w3-hide-small w3-padding-large w3-
hover-white”>Tsunami</a>
 <a href=”{%url ‘tornado’%}” class=”w3-bar-item w3-button w3-hide-small w3-padding-large w3-
hover-white”>Tornado</a>
 <a href=”{%url ‘volcano’%}” class=”w3-bar-item w3-button w3-hide-small w3-padding-large w3-hoverwhite”>Volcanic Activity</a>
 </div>
 <!—Navbar on small screens →
 <div id=”navDemo” class=”w3-bar-block w3-white w3-hide w3-hide-large w3-hide-medium w3-large”>
 <a href=”#” class=”w3-bar-item w3-button w3-padding-large”>Earthquake</a>
 <a href=”#” class=”w3-bar-item w3-button w3-padding-large”>Tsunami</a>
 <a href=”#” class=”w3-bar-item w3-button w3-padding-large”>Tornado</a>
 <a href=”#” class=”w3-bar-item w3-button w3-padding-large”>Volcanic Activity</a>
 </div>
</div>
<!—Header →
<header class=”w3-container w3-grey w3-center” style=”padding: 128px 16px”>
```

```
 <h1 class=”w3-margin w3-jumbo”>Earthquakes</h1>
 <p class=”w3-xlarge”>Natural Disasters Database</p>
</header>
<div class=”w3-container”>
 <h2>Earthquakes</h2>
 <table class=”w3-table-all”>
 <tr>
 <th>Earthquake_id</th>
 <th>Intensity</th>
 <th>Date</th>
 <th>Country</th>
 <th>Place</th>
 <th>Latitude</th>
 <th>Longitude</th>
 {% for quake in all_quakes %}
 <tr>
 <td>{{quake.earthquake_id}}</td>
 <td>{{quake.intensity}}</td>
 <td>{{quake.date}}</td>
 <td>{{quake.country}}</td>
 <td>{{quake.place}}</td>
 <td>{{quake.latitude}}</td>
 <td>{{quake.longitude}}</td>
 </tr>
 {% endfor %}
 </table>
</div>
<div class=”w3-container”>
 <h2>Damage caused by the quakes</h2>
 <table class=”w3-table-all”>
 <tr>
 <th>Earthquake_id</th>
 <th>Amount (in million)</th>
```

Train Test and Save Model:-

Table of Contents:-

Step 1 – Import the library

```
From sklearn import model_selection, datasets
From sklearn.tree import DecisionTreeClassifier
From sklearn.externals import joblib
Import pickle
```

We have imported model_selection, datasets, joblib, DecisionTreeClassifier and pickel which will be needed for the dataset.

Step 2 – Setting up the Data

We have loaded inbuilt wine dataset and stored data in x and target in y. We have used test_train_split to split the dataset such that 30% of data is for testing the model.

```
Dataset = datasets.load_wine()

X = dataset.data; y = dataset.target
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.3)
```

Master the Art of Classification in Machine Learning to Become a Pro

Step 3 – Training and Saving the Model

We are using DecisionTreeClassifier as a model. We have trained the model by training data. We can save the model by using joblib.dump in which we have passed the parameter as model and the filename.

Model = DecisionTreeClassifier()
Model.fit(X_train, y_train)

Filename = "Completed_model.joblib"
Joblib.dump(model, filename)

Step 4 – Loading the Saved Model

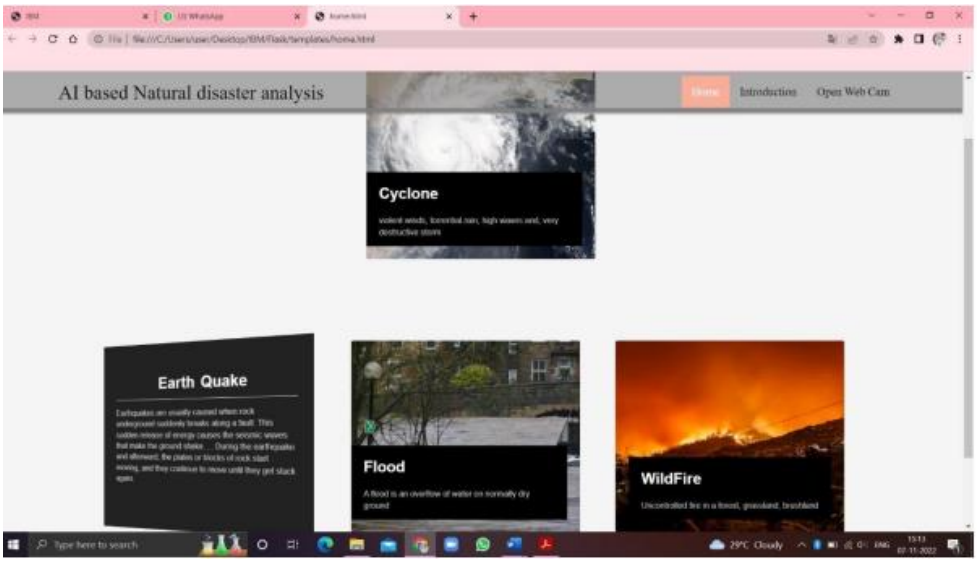
So here we are loading the saved model by using joblib.load and after loading the model we have
used score to get the score of the pretrained saved model.

Loaded_model = joblib.load(filename)
Result = loaded_model.score(X_test, y_test)
Print(result)

So the output comes as:
0.9444444444444444

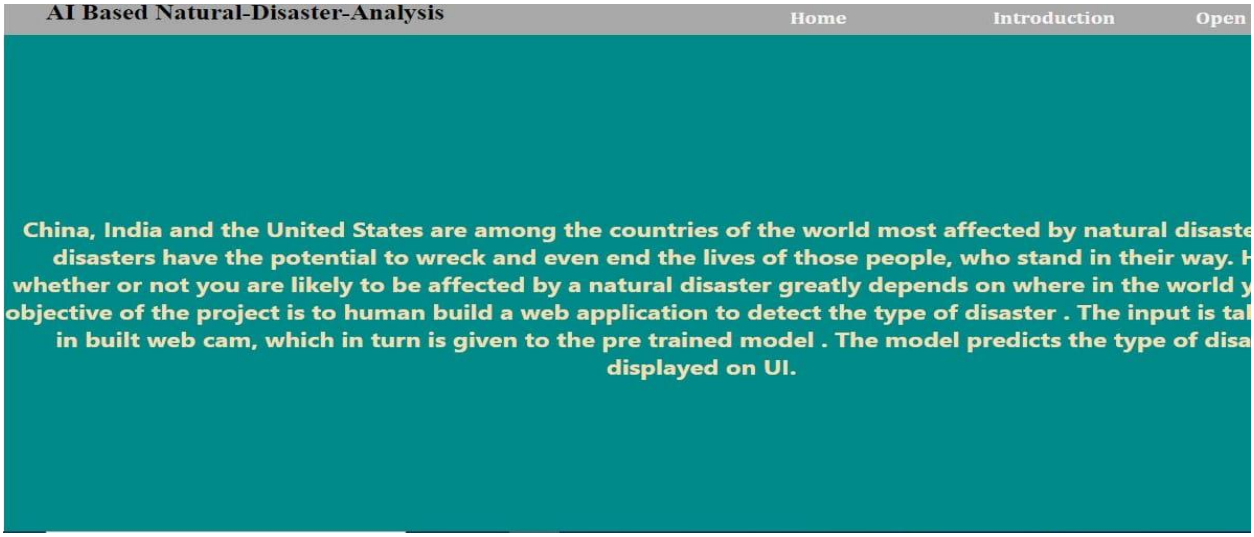
CREATION OF HOME PAGE:

Using HTML and CSS, the Home page is created. From the Home page the User can be able to know the basics of the frequently occurring Disasters. The home.html page is given below:



CREATION OF INTRO PAGE:

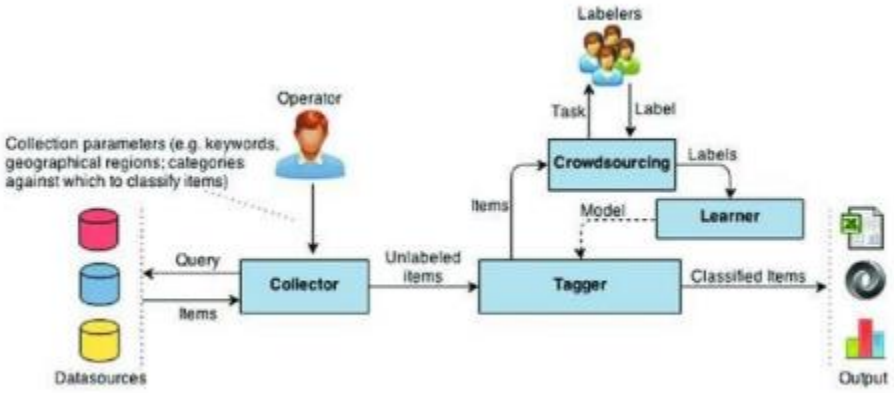
Using HTML and CSS, the intro page is created. From the intro page the user can be able to know about the project's introduction or abstract. The intro.html page is given below:



OPENING WEB CAM:

Using HTML and CSS, the upload.html page is created. Through this page User can be able to open the web cam to know about current disaster. The upload.html page is given below:

PROJECT FLOW:



COLLECTION OF DATASET:

The images of Disaster-prone areas are collected and organised into the subdirectories. The images of four types of Natural Disasters, Cyclone, Earthquake, Flood, Wildfire are collected and saved with the respective names. For more accuracy, Dataset with more images is selected and trained

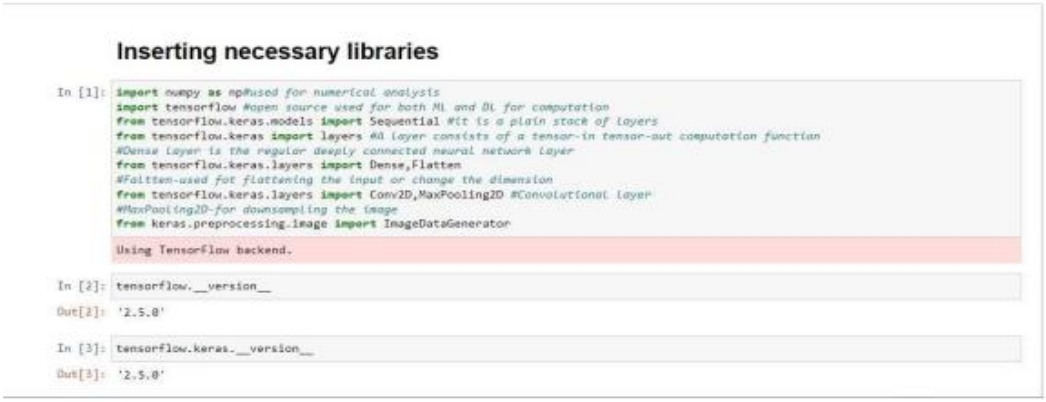
INSERTING NECESSARY LIBRARIES:

Numpy: It is an open source numerical python library.

Scikit-learn: It is a machine learning library for python.

OpenCV: OpenCV is a library of programming functions mainly aimed at real-time computer vision.

Flask: Web framework used for building web application.



Inserting necessary libraries

```
In [1]: import numpy as npNused for numerical analysis
        import tensorflow #open source used for both ML and DL for computation
        from tensorflow.keras.models import Sequential #it is a plain stack of layers
        from tensorflow.keras import layers #A layer consists of a tensor-in tensor-out computation function
        #Dense layer is the regular deeply connected neural network layer
        from tensorflow.keras.layers import Dense,Flatten
        #Faltten-used fot flattening the input or change the dimension
        from tensorflow.keras.layers import Conv2D,MaxPooling2D #Convolutional layer
        #MaxPooling2D-for downsampling the image
        from keras.preprocessing.image import ImageDataGenerator

        Using TensorFlow backend.

In [2]: tensorflow.__version__
Out[2]: '2.5.0'

In [3]: tensorflow.keras.__version__
Out[3]: '2.5.0'
```

LOADING DATA AND PERFORMING DATA AUGUMENTATION:

Loading the data into the Jupyter notebook by using RR dataset path.

Loading our data and performing Data Augumentation

```
In [5]: #performing data aguementation to train data
        x_train = train_datagen.flow_from_directory(r'C:\Users\ELCOT\Downloads\project\ibm\dataset\train_set',target_size=(64, 64),batch_
                                                    color_mode='rgb',class_mode='categorical')
        #performing data aguementation to test data
        x_test = test_datagen.flow_from_directory(r'C:\Users\ELCOT\Downloads\project\ibm\dataset\test_set',target_size=(64, 64),batch_si
                                                  color_mode='rgb',class_mode='categorical')

        Found 742 images belonging to 4 classes.
        Found 198 images belonging to 4 classes.

In [6]: print(x_train.class_indices)#checking the number of classes
        {'Cyclone': 0, 'Earthquake': 1, 'Flood': 2, 'Wildfire': 3}

In [7]: print(x_test.class_indices)#checking the number of classes
        {'Cyclone': 0, 'Earthquake': 1, 'Flood': 2, 'Wildfire': 3}

In [8]: from collections import Counter as c
        c(x_train .labels)
Out[8]: Counter({0: 220, 1: 156, 2: 198, 3: 168})
```

CREATING THE MODEL:

Creating the Model a Classifier Sequential. Classifier is a machine learning algorithm that determines the class of the input element based on the set of the feature. In this model using convolution 2D function. Convolution2D parameter is an number of filters that convolution layer will be learn from. Then we will be using MaxPooling 2D function. Then, using a Flatten() function that flatten the multidimensional input denser into the denser.

Creating the Model

```
In [9]: # Initializing the CNN
        classifier = Sequential()

        # First convolution layer and poolingo
        classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
        classifier.add(MaxPooling2D(pool_size=(2, 2)))
        classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
        # Second convolution layer and pooling
        classifier.add(Conv2D(32, (3, 3), activation='relu'))
        # input_shape is going to be the pooled feature maps from the previous convolution layer
        classifier.add(MaxPooling2D(pool_size=(2, 2)))
        classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))

        # Flattening the layers
        classifier.add(Flatten())

        # Adding a fully connected layer
        classifier.add(Dense(units=128, activation='relu'))
        classifier.add(Dense(units=4, activation='softmax')) # softmax for more than 2
```

Using classifier.summary() function summary of our model

COMPILING THE MODEL:

The model is compiled using the following code.

```
In [11]: # Compiling the CNN
         # categorical_crossentropy for more than 2
         classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

FITTING THE MODEL:

Fitting the Model with 70 epoch.

```
Epoch 65/70
149/149 [==============================] - 63s 421ms/step - loss: 0.0427 - accuracy: 0.9852 - val_loss: 1.8291 - val_accuracy: 0.7374
Epoch 66/70
149/149 [==============================] - 62s 415ms/step - loss: 0.0574 - accuracy: 0.9838 - val_loss: 1.7194 - val_accuracy: 0.7424
Epoch 67/70
149/149 [==============================] - 62s 415ms/step - loss: 0.0434 - accuracy: 0.9879 - val_loss: 1.8268 - val_accuracy: 0.7374
Epoch 68/70
149/149 [==============================] - 62s 415ms/step - loss: 0.0796 - accuracy: 0.9717 - val_loss: 1.7385 - val_accuracy: 0.7323
Epoch 69/70
149/149 [==============================] - 64s 427ms/step - loss: 0.0287 - accuracy: 0.9865 - val_loss: 2.0659 - val_accuracy: 0.7424
Epoch 70/70
149/149 [==============================] - 63s 424ms/step - loss: 0.0242 - accuracy: 0.9933 - val_loss: 3.4165 - val_accuracy: 0.6818
```

SAVING THE MODEL:

Saving the Model as disaster.h5. disaster.h5 file is used to find the image classification files. Model.json represents that Jason stands for JavaScript object rotation, Jason is a lite weight data format used for data inserting between multiple different language.

Saving the Model

```
In [13]: # Save the model
         classifier.save('disaster.h5')

In [14]: model_json = classifier.to_json()
         with open("model-bw.json", "w") as json_file:
             json_file.write(model_json)
```

PREDICTING RESULTS:

Loading model from the tensorflow keras models and loading the image then converting image into array. Then predicting our model.

```
In [15]: from tensorflow.keras.models import load_model
         from keras.preprocessing import image
         model = load_model("disaster.h5") #Loading the model for testing

In [ ]:

In [16]: img = image.load_img(r'C:\Users\ELCOT\Downloads\project\ibm\dataset\test_set\Cyclone\870.jpg',grayscale=False, target_size= (64,6
         x = image.img_to_array(img)#image to array\n",
         x = np.expand_dims(x,axis = 0)#changing the shape\n",
         pred = model.predict_classes(x)#predicting the classes\n",
         pred

         C:\Users\ELCOT\anaconda3\lib\site-packages\tensorflow\python\keras\engine\sequential.py:455: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` last-layer activation).
           warnings.warn('`model.predict_classes()` is deprecated and '

Out[16]: array([0], dtype=int64)

In [17]: index=['Cyclone','Earthquake','Flood','Wildfire']
         result=str(index[pred[0]])
         result

Out[17]: 'Cyclone'
```

DETECTION AND ANALYSIS OF DATA:

After Testing and Training the model, data which given in dataset are analysed and visualised effectively to detect the Disaster Type. Using webcam, it can capture image or video stream of Disaster, to detect and analyse the type of Disaster.

```
print(x_train.class_indices)#checking the number of classes
```

```
print(x_test.class_indices)#checking the number of classes
```

```
from collections import Counter as c
c(x_train .labels)
```

***IMAGE PREPROCESSING*:**

Image Pre-processing was done for Disaster intensity analysis and classification with three main tasks which includes for pre-processing of Images,

- Import ImageDataGenerator Library.
- Configure ImageDataGenerator Class.
- Applying ImageDataGenerator functionality to the trainset and test set.

Image Data Augumentation

```
In [ ]: #setting parameter for Image Data agumentation to the training data
        train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
        #Image Data agumentation to the testing data
        test_datagen=ImageDataGenerator(rescale=1./255)
```

IMPORTING THE IMAGEDATAGENERATOR LIBRARY:

1. By importing the ImageDataGenerator Library can expand the train_set data size using modified versions of dataset.
2. ImageDataGenerator class were importing from keras.

```
from keras.preprocessing.image import ImageDataGenerator
```

```
Using TensorFlow backend.
```

CONFIGURE IMAGEDATAGENERATOR CLASS:

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation.

An instance of the ImageDataGenerator class can be constructed for train and test dataset by ImageDataGenerator class.

Image Data Augumentation

In [ ]:
```
#setting parameter for Image Data agumentation to the training data
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
#Image Data agumentation to the testing data
test_datagen=ImageDataGenerator(rescale=1./255)
```

APPLYING IMAGEDATAGENERATOR FUNCTIONALITY TO TRAINSET AND TESTSET :

ImageDataGenerator functionality was applied to Trainset and Testset by using the following code,

"For Training set using flow_from_directory function".

Loading our data and performing Data Augumentation

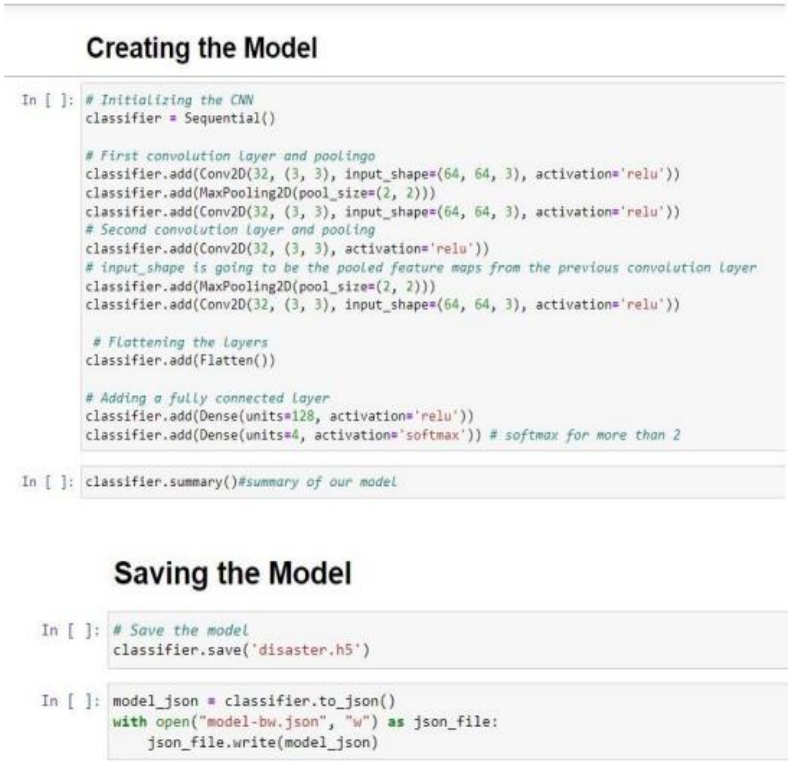
In [5]:
```
#performing data agumentation to train data
x_train = train_datagen.flow_from_directory(r'C:\Users\ELCOT\Downloads\project\ibm\dataset\train_set',target_size=(64, 64),batch_
                                            color_mode='rgb',class_mode='categorical')
#performing data agumentation to test data
x_test = test_datagen.flow_from_directory(r'C:\Users\ELCOT\Downloads\project\ibm\dataset\test_set',target_size=(64, 64),batch_si
                                            color_mode='rgb',class_mode='categorical')
```

```
Found 742 images belonging to 4 classes.
Found 198 images belonging to 4 classes.
```

MODEL BUILDING:

Building a Model with web application named “FLASK”, model building process consist several steps like,

- Import the model building Libraries
- Initializing the model
- Adding CNN Layers
- Adding Hidden Layer
- Adding Output Layer
- Configure the Learning Process
- Training and testing the model all the above processes are done and saved in a model.



Creating the Model

```
In [ ]: # Initializing the CNN
classifier = Sequential()

# First convolution layer and poolingo
classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
# Second convolution layer and pooling
classifier.add(Conv2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))

 # Flattening the layers
classifier.add(Flatten())

# Adding a fully connected layer
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=4, activation='softmax')) # softmax for more than 2
```

```
In [ ]: classifier.summary()#summary of our model
```

Saving the Model

```
In [ ]: # Save the model
classifier.save('disaster.h5')
```

```
In [ ]: model_json = classifier.to_json()
with open("model-bw.json", "w") as json_file:
    json_file.write(model_json)
```

CREATING app.py:

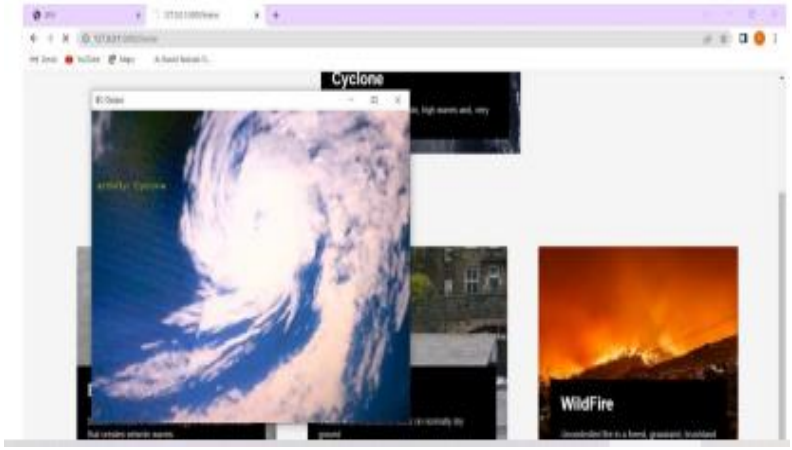
```
# import the necessary packages
from flask import Flask,render_template,request
# Flask-It is our framework which we are going to use to run/serve our application.
#request-for accessing file which was uploaded by the user on our application.
#import operator
import cv2 # opencv library
from tensorflow.keras.models import load_model#to load our trained model
import numpy as np
#import os
from werkzeug.utils import secure_filename
#from playsound import playsound
#from gtts import gTTS
'''
def playaudio(text):
    speech=gTTS(text)
    print(type(speech))
    speech.save("output1.mp3")
    playsound("output1.mp3")
    return
'''
app = Flask(__name__,template_folder="templates") # initializing a flask app
# Loading the model
model=load_model(r'C:\Users\user\Desktop\IBM\Flask\templates\disaster.h5')
print("Loaded model from disk")


app=Flask(__name__,template_folder="templates")
@app.route('/', methods=['GET'])
def index():
    return render_template('home.html')
@app.route('/home', methods=['GET'])
def home():
    return render_template('home.html')
@app.route('/intro', methods=['GET'])
def about():
    return render_template('intro.html')
@app.route('/upload', methods=['GET', 'POST'])
```

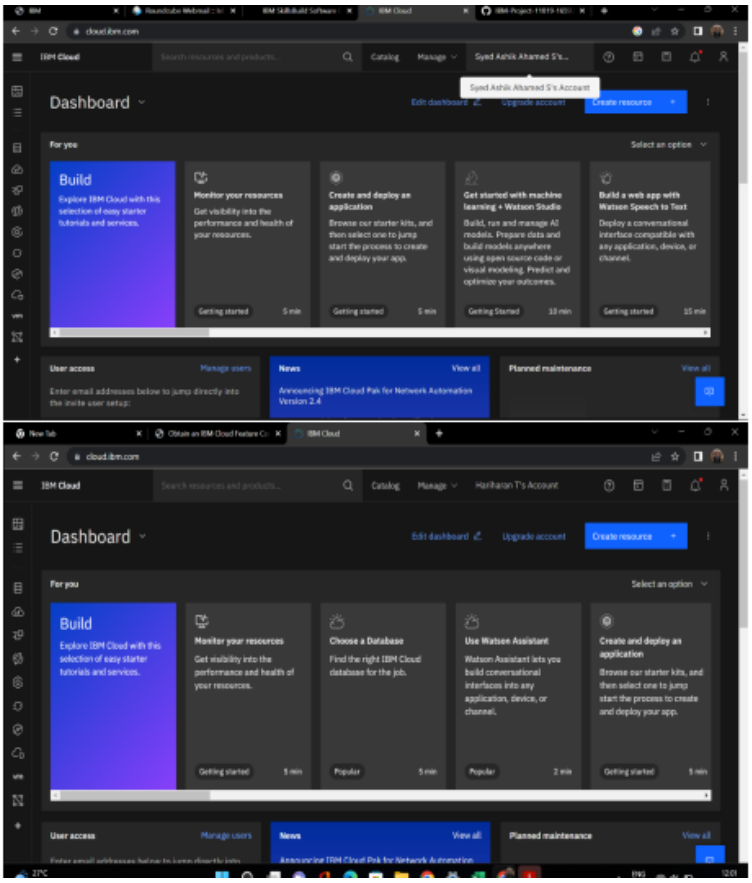
INTEGRATE THE WEB APP WITH AI MODEL:

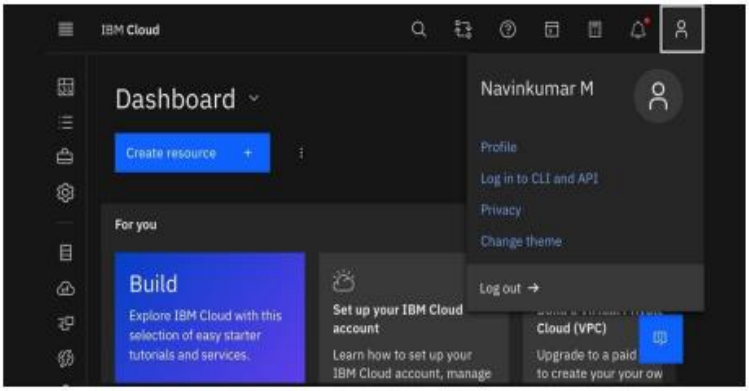
After creating the Model, the Model should be integrated with the web app using the Flask application. The coding part is named as app.py and it will be running in the localhost through the generated link. By navigating the localhost the webpage will be visible.

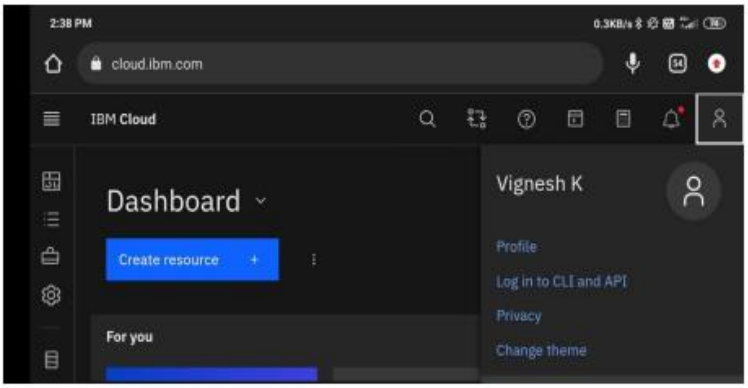
```
            output = frame.copy()
            #print("apple")
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            frame = cv2.resize(frame, (64, 64))
            #frame = frame.astype("float32")
            x=np.expand_dims(frame, axis=0)
            result = np.argmax(model.predict(x), axis=-1)
            index=['Cyclone','Earthquake','Flood','Wildfire']
            result=str(index[result[0]])
            #print(result)
            #result=result.tolist()

            cv2.putText(output, "activity: {}".format(result), (10, 120), cv2.FONT_HERSHEY_PLAIN,
                        1, (0,255,255), 1)
            #playaudio("Emergency it is a disaster")
            cv2.imshow("Output", output)
            key = cv2.waitKey(1) & 0xFF
        ----
        ----# if the `q` key was pressed, break from the loop
            if key == ord("q"):
                break

        # release the file pointers
        print("[INFO] cleaning up...")
        vs.release()
        cv2.destroyAllWindows()
        return render_template("upload.html")


if __name__ == '__main__':
    app.run(debug=False,threaded=True)
```


Output
activity: Earthquake


Cyclone
WildFire


Dashboard
Build
Explore IBM Cloud with this selection of easy starter tutorials and services.
Monitor your resources
Get visibility into the performance and health of your resources.
Create and deploy an application
Browse our starter kits, and then select one to jump start the process to create and deploy your app.
Get started with machine learning + Watson Studio
Build, run and manage AI models. Prepare data and build models anywhere using open source code or visual modeling. Predict and optimize your outcomes.
Build a web app with Watson Speech to Text
Deploy a conversational interface compatible with any application, device, or channel.
User access
News
Planned maintenance
Dashboard
Build
Explore IBM Cloud with this selection of easy starter tutorials and services.
Monitor your resources
Get visibility into the performance and health of your resources.
Choose a Database
Find the right IBM Cloud database for the job.
Use Watson Assistant
Watson Assistant lets you build conversational interfaces into any application, device, or channel.
Create and deploy an application
Browse our starter kits, and then select one to jump start the process to create and deploy your app.
User access
News
Planned maintenance


IBM Cloud
Dashboard
Create resource
For you
Build
Explore IBM Cloud with this selection of easy starter tutorials and services.
Navinkumar M
Profile
Log in to CLI and API
Privacy
Change theme
Log out
Set up your IBM Cloud account
Learn how to set up your IBM Cloud account, manage
Cloud (VPC)
Upgrade to a paid
to create your your ow


2:38 PM
0.3KB/s
cloud.ibm.com
IBM Cloud
Dashboard
Create resource
For you
Vignesh K
Profile
Log in to CLI and API
Privacy
Change theme

8. TESTING

8.1 TEST CASES

Section	Total Cases	Not Tested	Fail	Pass
Client Application	10	0	3	7
Security	2	0	1	1
Performance	3	0	1	2
Exception Reporting	2	0	0	2

8.2 USER ACCEPTANCE TESTING

DEFECT ANALYSIS

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Total
By Design	1	0	1	0	2
Duplicate	0	0	0	0	0
External	0	0	2	0	2
Fixed	4	1	0	1	6
Not Reproduced	0	0	0	1	1
Skipped	0	0	0	1	1
Won't Fix	1	0	1	0	2
Total	6	1	4	3	14

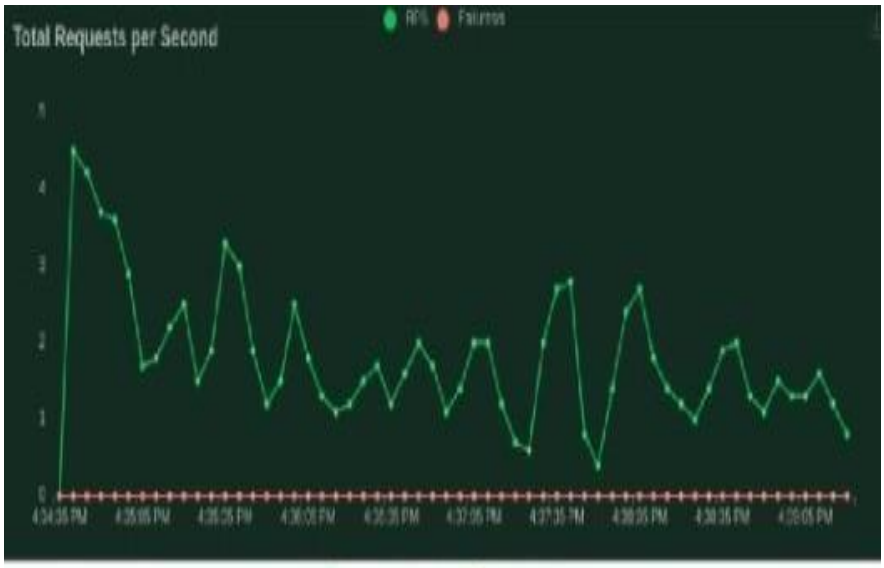
9. RESULTS

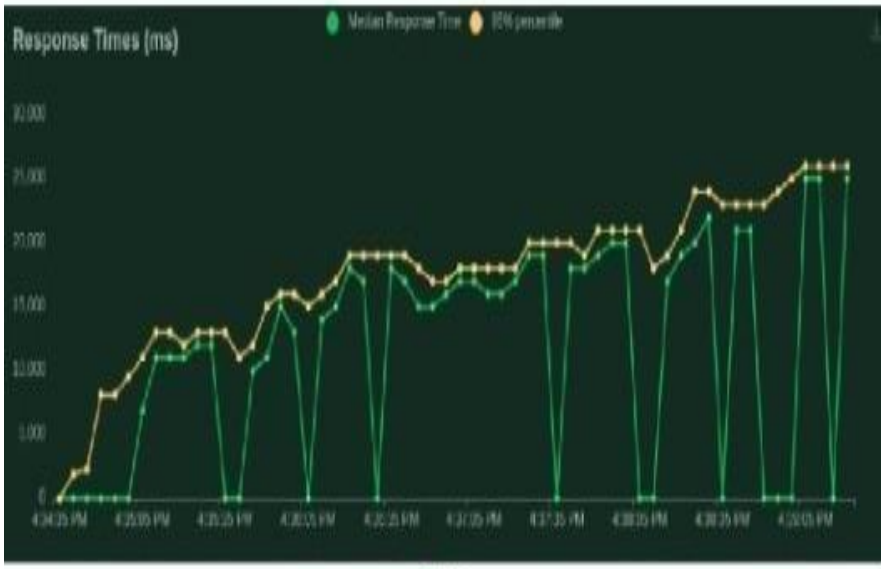
9.1 PERFORMANCE METRICS

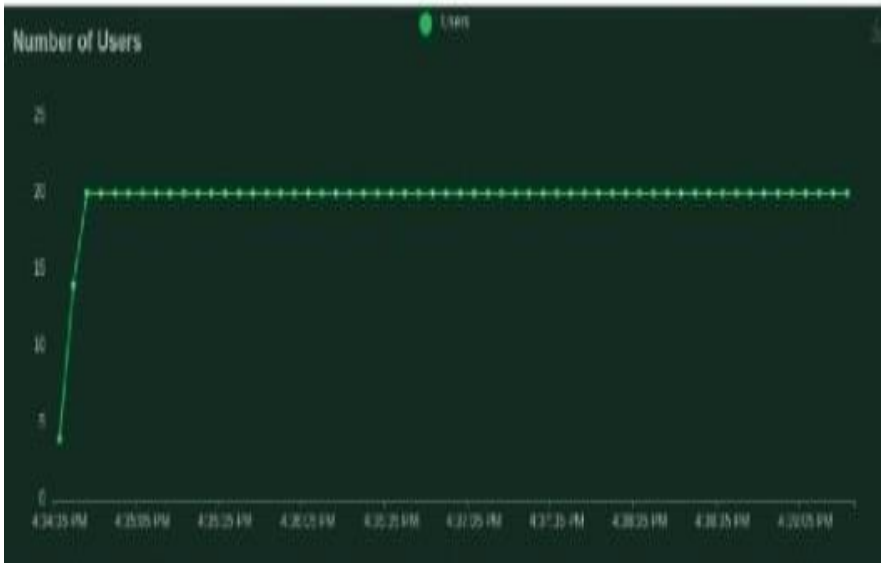
Locust Test Report :

During: 11/20/2022, 12:20:34 PM - 11/20/2022, 12:29:21 PM

Script: locustfile.py


Total Requests per Second
All
Failures


Response Times (ms)
Median Response Time
95% percentile


Number of Users
Users

10. ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

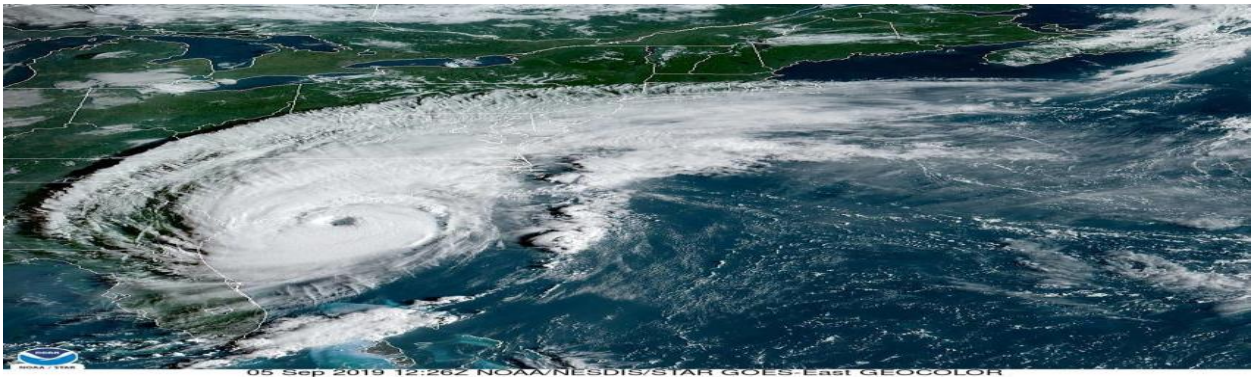
- It can help reduce the losses and damages caused by natural disasters by providing early warning and response systems.
- It can help improve the understanding and prediction of natural disaster patterns and trends by analyzing large amounts of data.
- It can help enhance the resilience and adaptation of human societies and ecosystems by providing information and guidance for disaster prevention and recovery.
- The advantage of natural disaster intensity analysis and classification is that it can use deep learning techniques to overcome the challenges of complex and imbalanced images. For example, a multilayered deep convolutional neural network can extract features and classify images of different natural disasters with high accuracy and efficiency.

DISADVANTAGES

- They can cause loss of life, injury, and displacement of people and animals.
- They can create humanitarian crises, such as food insecurity, water scarcity, disease outbreaks, and social unrest.
- They can damage or destroy properties, infrastructures, and ecosystems, resulting in economic losses and environmental degradation.

TEST SET:

CYCLONE



EARTHQUAKE

FLOODS:

WILDFIRES :

11. CONCLUSION

Artificial intelligence has the potential to enhance the detection and classificationof natural disasters, as well as the resilience and relief efforts of affected communities. By using deep learning techniques, AI can analyze complex and imbalanced images of disasters and provide accurate and timely information. However, AI also faces challenges such as data quality, ethical issues, and human-AI collaboration. Therefore, it is essential to develop robust and reliable AI systems that can complement human expertise and judgment in disaster management.

- AI can help predict the occurrence and impact of natural disasters by using historical data, satellite imagery, and weather models. This can enable early warning systems and preparedness plans for vulnerable areas. AI can also assist in the recovery and reconstruction of disaster-affected regions by providing insights into the needs and priorities of the survivors, as well as the best allocation of resources and funds.

- AI can also support the learning and improvement of disaster management practices by analyzing the lessons learned from past disasters and identifying the gaps and opportunities for future interventions.

12. FUTURE SCOPE

- To develop more advanced and efficient deep learning models that can handle the complexity and diversity of natural disaster images, and provide accurate and reliable results.

- To integrate multiple sources and types of data, such as text, audio, video, and sensor data, to enhance the analysis and classification of natural disasters and their impacts.

- To explore the ethical and social implications of using AI for natural disaster management, such as the privacy, security, and accountability of the data and the algorithms, and the potential biases and risks of the AI outputs.

- To evaluate the performance and impact of AI for natural disaster management, and compare it with other methods and tools, such as human experts, traditional models, and manual processes.

- To foster the collaboration and communication among different stakeholders, such as researchers, practitioners, policymakers, and communities, to share the best practices and challenges of using AI for natural disaster management, and to co-create solutions that meet the needs and expectations of the users.

- To promote the awareness and education of the public and the decision-makers on the benefits and limitations of AI for natural disaster management, and to encourage the participation and feedback of the affected people and groups.

13. APPENDIX SOURCE

```
[ ] software_spec_uid=wml_client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")

[ ] software_spec_uid

    'acd9c798-6974-5d2f-a657-ce06e986df4d'

[ ] model_details = wml_client.repository.store_model(model="naturaldisaster-classification-model.tgz",meta_props={
        wml_client.repository.ModelMetaNames.NAME : "CNN model",
        wml_client.repository.ModelMetaNames.TYPE : "tensorflow_2.7",
        wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
    })

[ ] model_id = wml_client.repository.get_model_id(model_details)

[ ] model_id

    'b859dab6-7364-4ad8-a417-559922bbefc2'

[ ] wml_client.repository.download(model_id,'naturaldisaster.tar.gb')

    Successfully saved model content to file: 'naturaldisaster.tar.gb'
    '/content/drive/MyDrive/dataset/naturaldisaster.tar.gb'
```

```
[ ] model_details

{'entity': {'hybrid_pipeline_software_specs': [],
  'software_spec': {'id': 'acd9c798-6974-5d2f-a657-ce06e986df4d',
   'name': 'tensorflow_rt22.1-py3.9'},
  'type': 'tensorflow_2.7'},
 'metadata': {'created_at': '2022-11-21T17:56:32.685Z',
  'id': 'b859dab6-7364-4ad8-a417-559922bbefc2',
  'modified_at': '2022-11-21T17:56:36.195Z',
  'name': 'CNN model',
  'owner': 'IBMid-6630043HCX',
  'resource_key': '211b15f2-fc2e-4c95-866c-45a87d520abf',
  'space_id': 'c147dde1-e498-4151-bcf8-7b82eebf2ee3'},
 'system': {'warnings': []}}
```