PROJECT REPORT

Emerging Methods For Early Detection Of Forest Fires

TEAM ID : PNT2022TMID27543

GROUP MEMBERS :

NAME	REG NO:
MITHRRA SREE RA	311119106031
SHRI VARSHA R	311119106049
VANITHA LAKSHMI M	311119106057
VEERALAKSHMI P	311119106059

1. INTRODUCTION

1.1 Project Overview

Forest fires are a major environmental issue, creating economic and ecological damage while endangering human lives. Fire in the forest can occur naturally or by humans. Naturally, fire takes place due to extreme drought, hot weather, lightning or combustion of dry leaves and scobs. Human activities like throwing cigarettes, especially in forest areas or using borne fire also lead to fires. Since when there is fire in any region then the temperature of the region will become high due to fire. So increase in temperature is one of the factors which can help in fire detection events. To increase the accuracy in predicting fire events, we will use various AI techniques to check if there is fire or not. We will also train the machine and test by providing custom input whether there is fire or not. Hence we can save our environment ,animals and livelihood from the adverse results of forest fires.

1.2 Purpose

Fire detection systems increase response times, as they are able to alert the correct people in order to extinguish the fire. This thus reduces the amount of damage to the property. The forest fires destroys the wildlife habitat, damages the environment, affects the climate, spoils the biological properties of the soil, etc. So the forest fire detection is a major issue in the present decade. At the same time the forest fire have to be detected as fast as possible.

2. LITERATURE SURVEY

2.1 Existing problem

- It is difficult to predict and detect Forest Fire in a sparsely populated forest area and it is more difficult if the prediction is done using ground-based methods like Camera or Video-Based approach.
- Using only IR images may lead to false alarms. The combination of IR images with visible range image together to reduce the false alarm rates of fire detection arm result.
- Weather data must be included as it plays a major role in the occurrence, growth, spread and the extinction of wildfires. It can impact on the strength and movement of fire, and thus burn more land which makes its extinction even moe difficult.

2.2 References

- Early Detection of Forest Fire Using Mixed Learning Techniques and UAV-Varanasi LVSKB Kasyap,et al (2022)
- Holistic approach of forest fire protection of split and Dalmatia country of croatia- Darko stipanicev Ranko vujic (2014)
- A review on early fire detection systems using optical remote sensing-Panagiotis barmpoutis Konsmas dimitropoulas Nikos grammalidis(2020)
- The influence of climate change on forest fires in Yunnan province, Southwest china detected by GRACE satellites-Lilu cui Chaolong yao Zhengbo zou (2022)
- Fire detection using infrared images for UAV-based forest fire surveillance- Chi Yuan, et al (2017)
- S-mart forest fires early detection sensory system: Another approach of utilizing wireless sensor and neutral networks-Hamdy soliman (2010)
- Adoption of image surface parameter under moving edge computing in the construction in the construction of mountain fire warning method-Che cheng Hui zhou Danning wang (2020)
- Natural hazards wildfires- Prof.David(E.Al exander)
- Predictive modeling of wildfires: A new dataset and machine learning approach-Younes OuladSayad, et al(2019)
- Forest Fire Detection Using IoT and Artificial Neural Network-Vinay Dubey, et al(2018)

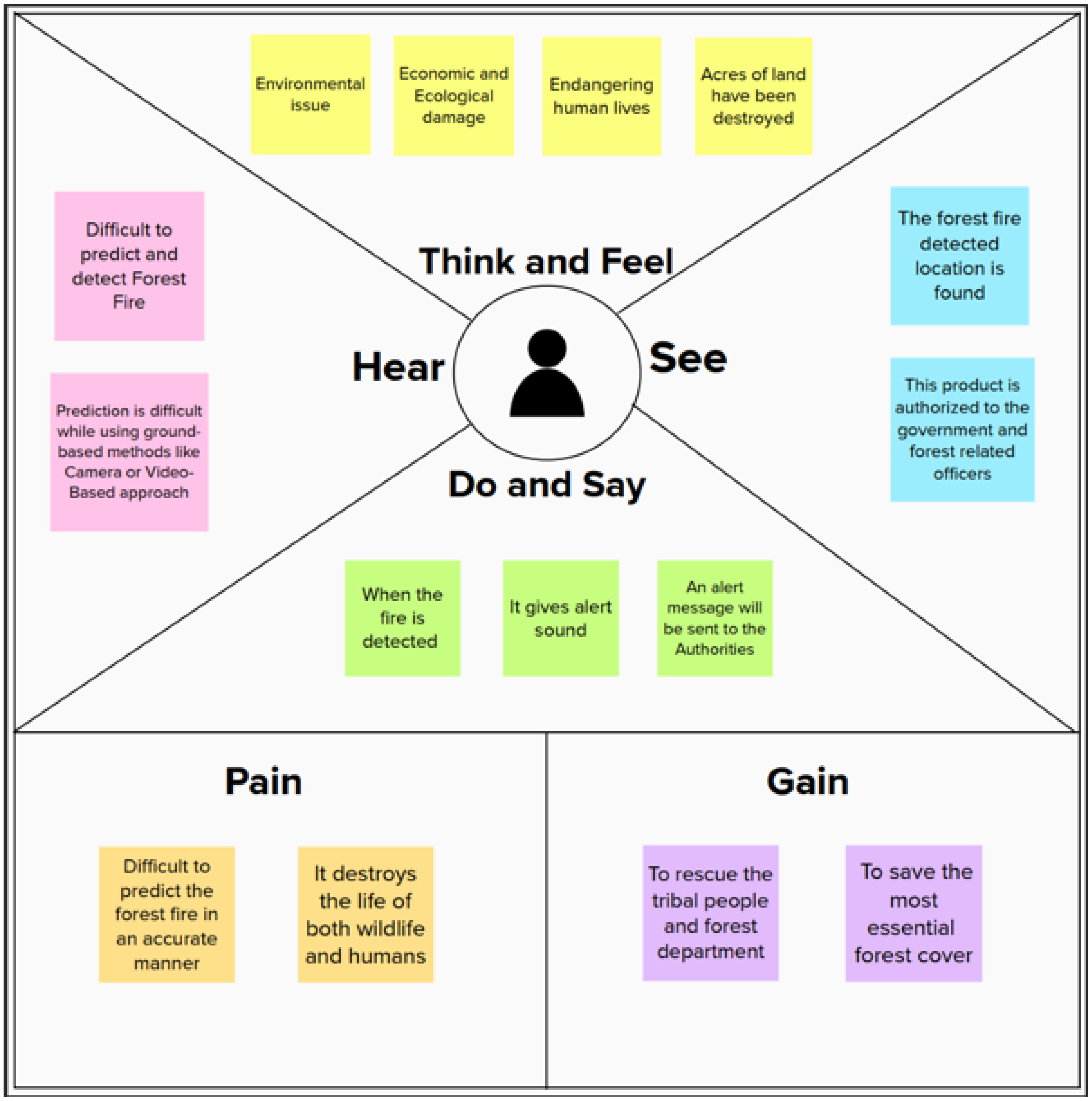
2.3 Problem Statement Definition

Forest fires are a major environmental issue, Over 9 million acres of land have been destroyed due to treacherous wildfires. Satellites can be an important source of data prior to and also during the Fire due to its reliability and efficiency. The various real-time forest fire detection and prediction approaches, with the goal of informing the local fire authorities.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

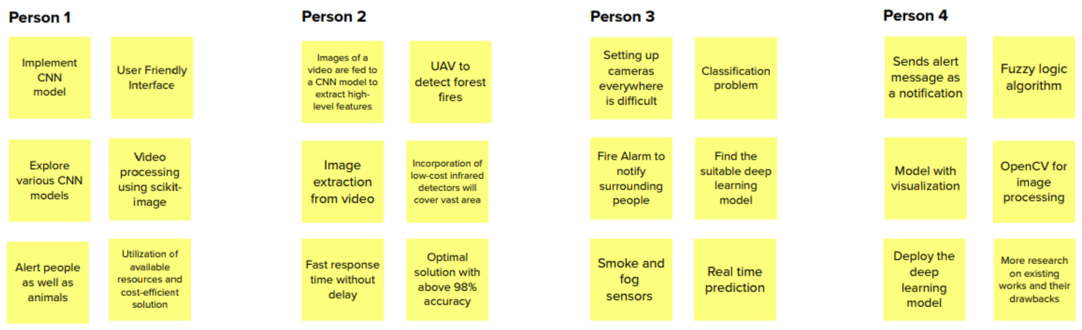
The empathy map canvas is used to capture the user Pains & Gains, Prepare list of problem statements. Here, it is used to explain the forest fire detection process.


Environmental issue
Economic and Ecological damage
Endangering human lives
Acres of land have been destroyed
Difficult to predict and detect Forest Fire
Prediction is difficult while using ground-based methods like Camera or Video-Based approach
Think and Feel
Hear
See
Do and Say
The forest fire detected location is found
This product is authorized to the government and forest related officers
When the fire is detected
It gives alert sound
An alert message will be sent to the Authorities
Pain
Gain
Difficult to predict the forest fire in an accurate manner
It destroys the life of both wildlife and humans
To rescue the tribal people and forest department
To save the most essential forest cover

3.2 Ideation & Brainstorming

Brainstorm

The brainstorming is used to get the consolidated ideas from each members and to choose the best idea presented and to implement that for the forest fire detection process.



Person 1

- Implement CNN model
- User Friendly Interface
- Explore various CNN models
- Video processing using scikit-image
- Alert people as well as animals
- Utilization of available resources and cost-efficient solution

Person 2

- Images of a video are fed to a CNN model to extract high-level features
- UAV to detect forest fires
- Image extraction from video
- Incorporation of low-cost infrared detectors will cover vast area
- Fast response time without delay
- Optimal solution with above 98% accuracy

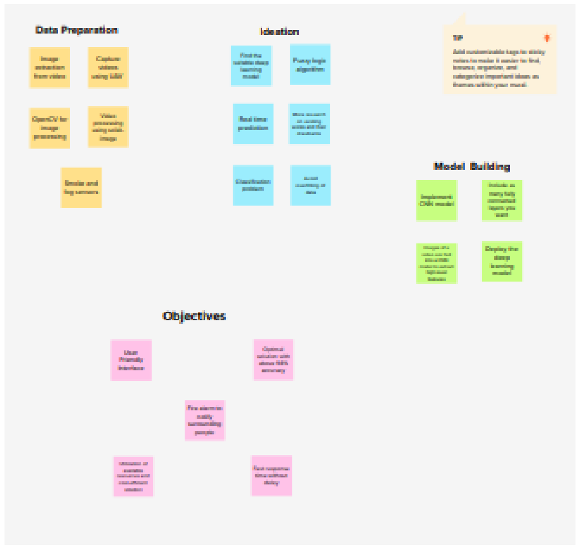
Person 3

- Setting up cameras everywhere is difficult
- Classification problem
- Fire Alarm to notify surrounding people
- Find the suitable deep learning model
- Smoke and fog sensors
- Real time prediction

Person 4

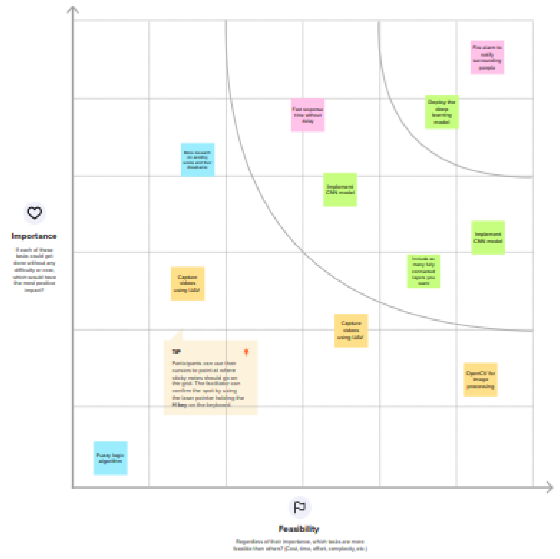
- Sends alert message as a notification
- Fuzzy logic algorithm
- Model with visualization
- OpenCV for image processing
- Deploy the deep learning model
- More research on existing works and their drawbacks

Group ideas



Data Preparation
Image extraction from video
Capture videos using UAV
OpenCV for image processing
Video processing using scikit-image
Smoke and fog sensors
Ideation
Find the suitable deep learning model
Fuzzy logic algorithm
Real time prediction
Classification problem
TIP
Model Building
Implement CNN model
Deploy the deep learning model
Objectives
User Friendly Interface
Optimal solution with above 98% accuracy
Fire alarm to notify surrounding people
Fast response time without delay

Prioritize



3.3 Proposed Solution

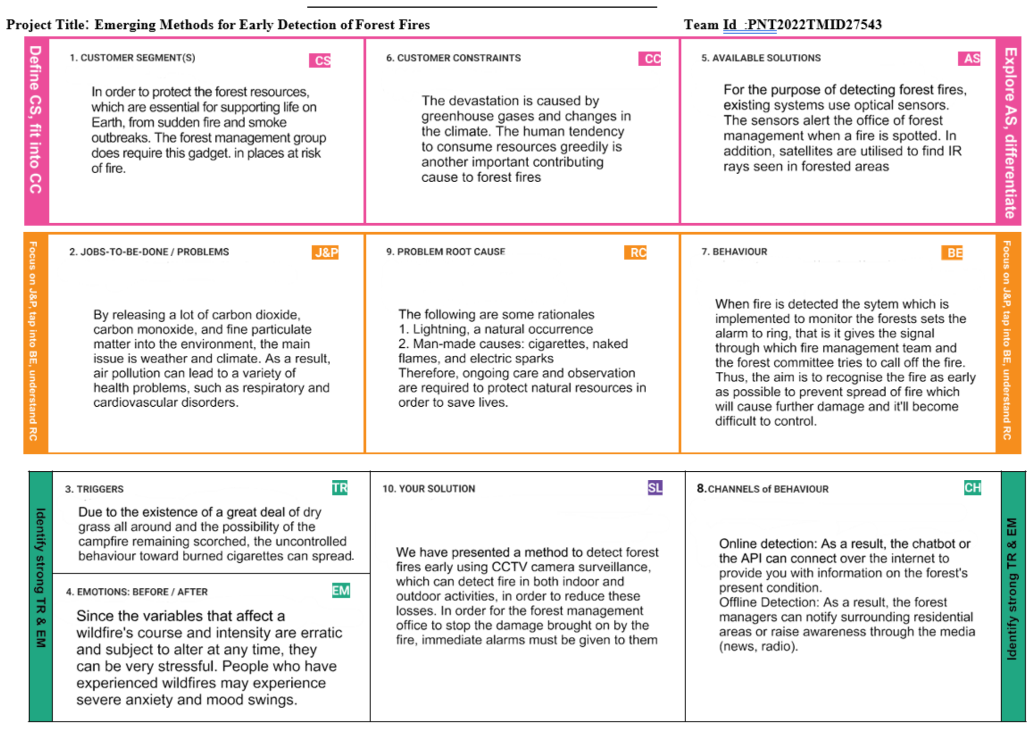
PROPOSED SOLUTION:

S.No.	**Parameter**	**Description**
1.	Problem Statement (Problem to be solved)	Forest fires pose a serious threat to the environment because they harm the economy, the ecosystem, and put people in danger. In a sparsely populated forest area, it is difficult to predict and detect forest fires, and it is even more challenging if the prediction is done using ground- based technologies such as camera or video-based approaches.
2.	Idea / Solution description	When there are any signs of a forest fire or other suspicious activity, the video of fire is streamed on the console, an alerting sound is produced, and an alert message is sent to the respective forest authorities.

3.	Novelty / Uniqueness	Due to their dependability and effectiveness, satellites can be a valuable source of data both before and during the Fire since ground-based techniques makes the prediction more challenging. Applying convolutional neural network (CNN) technology to image recognition can theoretically extract deeper features and minimize blindness and unpredictability to a substantial extent in the feature extraction process, which can significantly increase the accuracy of flame image recognition.
4.	Social Impact / Customer Satisfaction	A fire-detection system can limit the emission of harmful byproducts of combustion as well as global- warming gases produced by the fire by delivering early warning notification. Early fire detection helps to rescue countless acres of forest land, limits environmental damage caused by wildlife, and saves the loss of plants and animals.
5.	Business Model (Revenue Model)	This device can only be utilized by a large firm or by the government to monitor vast forest reserves.

6.	Scalability of the Solution	While controlling wildfires has advanced over the past few decades, there is still a need to increase disaster risk reduction capabilities, including early detection systems and real-time data transmission at all phases and stages of a forestsurveillance system. Monitoring the possible danger areas and early fire detection can considerably minimize the response time, potential damage, and firefighting expenses. Regardless of the geographical distance between resources and users, the system is regionally expandable and maintains its usability and usefulness.

3.4 Problem Solution fit



Project Title: Emerging Methods for Early Detection of Forest Fires **Team Id :PNT2022TMID27543**

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

In order to protect the forest resources, which are essential for supporting life on Earth, from sudden fire and smoke outbreaks. The forest management group does require this gadget. in places at risk of fire.

6. CUSTOMER CONSTRAINTS CC

The devastation is caused by greenhouse gases and changes in the climate. The human tendency to consume resources greedily is another important contributing cause to forest fires

5. AVAILABLE SOLUTIONS AS

For the purpose of detecting forest fires, existing systems use optical sensors. The sensors alert the office of forest management when a fire is spotted. In addition, satellites are utilised to find IR rays seen in forested areas

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

By releasing a lot of carbon dioxide, carbon monoxide, and fine particulate matter into the environment, the main issue is weather and climate. As a result, air pollution can lead to a variety of health problems, such as respiratory and cardiovascular disorders.

9. PROBLEM ROOT CAUSE RC

The following are some rationales
1. Lightning, a natural occurrence
2. Man-made causes: cigarettes, naked flames, and electric sparks
Therefore, ongoing care and observation are required to protect natural resources in order to save lives.

7. BEHAVIOUR BE

When fire is detected the sytem which is implemented to monitor the forests sets the alarm to ring, that is it gives the signal through which fire management team and the forest committee tries to call off the fire. Thus, the aim is to recognise the fire as early as possible to prevent spread of fire which will cause further damage and it'll become difficult to control.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

Due to the existence of a great deal of dry grass all around and the possibility of the campfire remaining scorched, the uncontrolled behaviour toward burned cigarettes can spread.

4. EMOTIONS: BEFORE / AFTER EM

Since the variables that affect a wildfire's course and intensity are erratic and subject to alter at any time, they can be very stressful. People who have experienced wildfires may experience severe anxiety and mood swings.

10. YOUR SOLUTION SL

We have presented a method to detect forest fires early using CCTV camera surveillance, which can detect fire in both indoor and outdoor activities, in order to reduce these losses. In order for the forest management office to stop the damage brought on by the fire, immediate alarms must be given to them

8. CHANNELS of BEHAVIOUR CH

Online detection: As a result, the chatbot or the API can connect over the internet to provide you with information on the forest's present condition.
Offline Detection: As a result, the forest managers can notify surrounding residential areas or raise awareness through the media (news, radio).

Identify strong TR & EM

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

- The system shall take training sets of fire images and recognize whether there is a fire or the beginning of a fire (smoke) or if there is no fire.
- The system shall send a notification to the admin when it recognizes a fire.
- The system shall take real inputs from the camera and determine whether the image contains a fire or not.
- The system shall be able to take images with a variety of sizes and convert it to one fixed image to be used throughout the application.
- Once the fire is detected , it sends the alert message to forest officer.

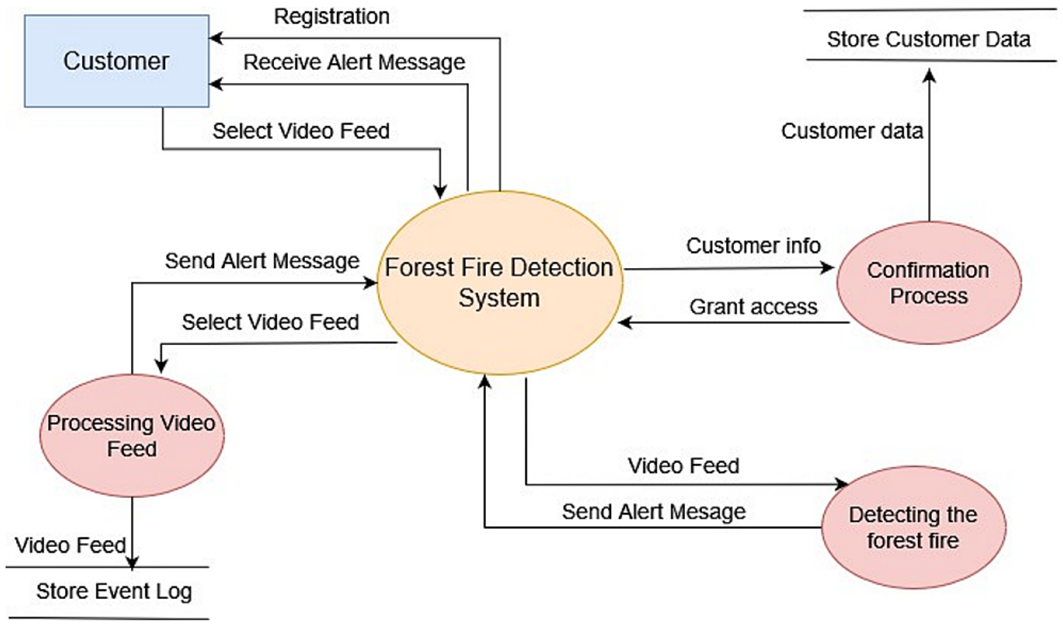
4.2 Non-Functional requirements

- Security
- Localization
- Reliability, maintainability, availability
- Probability and compatability
- Performance and scalability

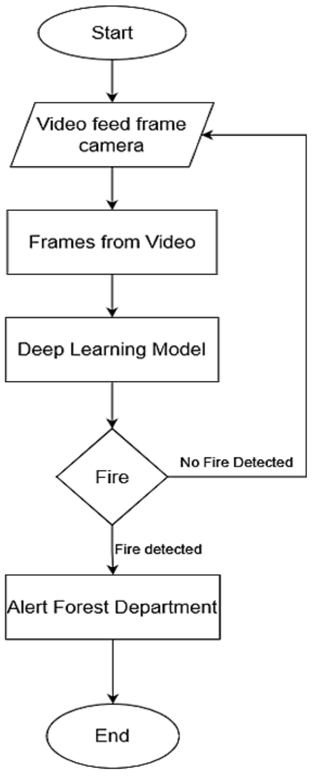
5. PROJECT DESIGN

5.1 Data Flow Diagrams

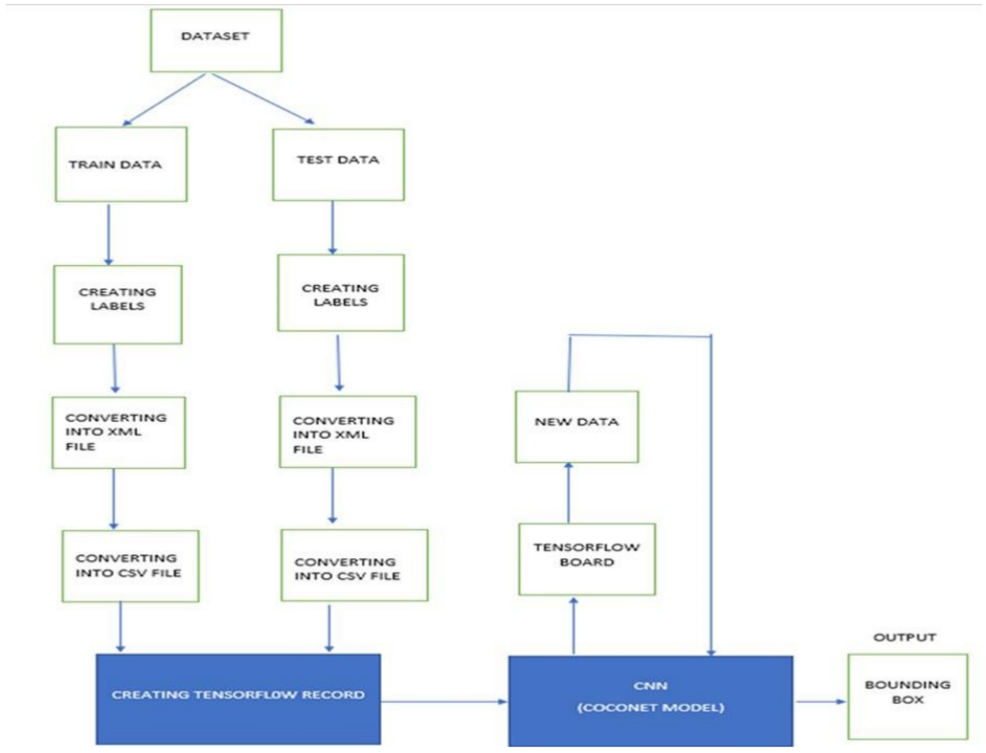
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored. Here, the DFD is used to represent the flow of the forest fire detection system.



A flow diagram is a visualization of a sequence of actions, movements within a system and/or decision points. They're a detailed explanation of each step in a process, no matter the level of complexity of that process. Here, the flow diagram is used to represent the working flow of the forest fire detection system.



5.2 Solution & Technical Architecture



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Forest Department and Environmentalist	Registration	USN-1	As a user, I can register for the application by entering correct my email, password, and confirming my password.	Only authorised government employees can be accepted	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application.	I can receive confirmation email & click confirm	High	Sprint-1

	Login	USN-3	As a user and a forest fire department staff, I will be provided with unique login ID and password.	I should enter correct email & password	High	Sprint-1
	IBM Cloud Server	USN-4	The forest fire is detected using computer vision algorithm based cameras. These cameras continuously monitor the forest and the data is sent to the server.	I can receive the data	High	Sprint-2
		USN-5	I can fetch the details/data from the cloud server.	I can access the data	High	Sprint-2
	Data Collection	USN-6	I must gather information about forest fires.	I should collect the accurate data	High	Sprint-3
		USN-7	I must draft and point out the algorithms to predict the forest fire.	I must analyse the algorithms with respect to its accuracy.	Medium	Sprint-3
	Algorithm Implementation	USN-8	I must determine the precision of each algorithm.	I must calculate the accuracy of the algorithm	High	Sprint-4
		USN-9	Extracting and assessing the Dataset	I must pre-process the dataset before training	High	Sprint-4
	Evaluaitng the Algorithm	USN-10	I must determine the precision, recall and accuracy of the algorithm.	Accuracy is essential for detecting the presence of forest fire	High	Sprint-4

6. PROJECT PLANNING & SCHEDULING

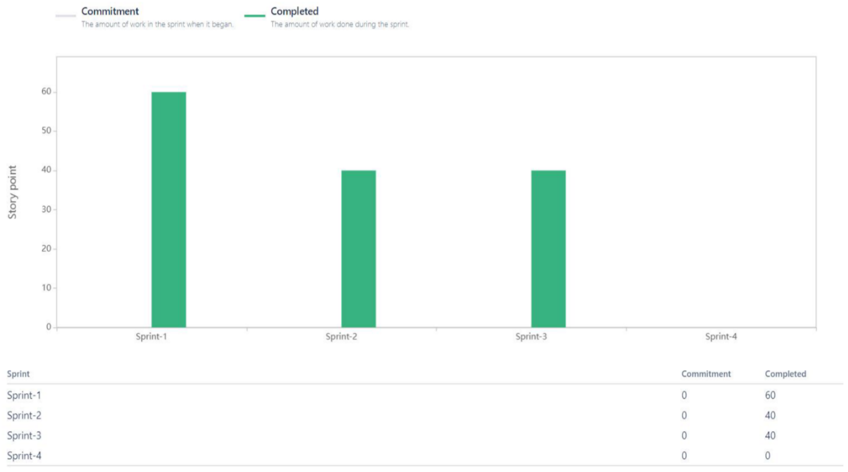
6.1 Sprint Planning & Estimation & Sprint Delivery Schedule

Sprint	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**	**Team Members**
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email,password,and confirming my password.	20	High	SHRI VARSHA R
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application usage.	20	High	MITHRRA SREE RA
Sprint-1	Login	USN-3	As a user and a forest fire department staff, I will be provided with unique login ID and password.	20	High	VANITHA LAKSHMI M

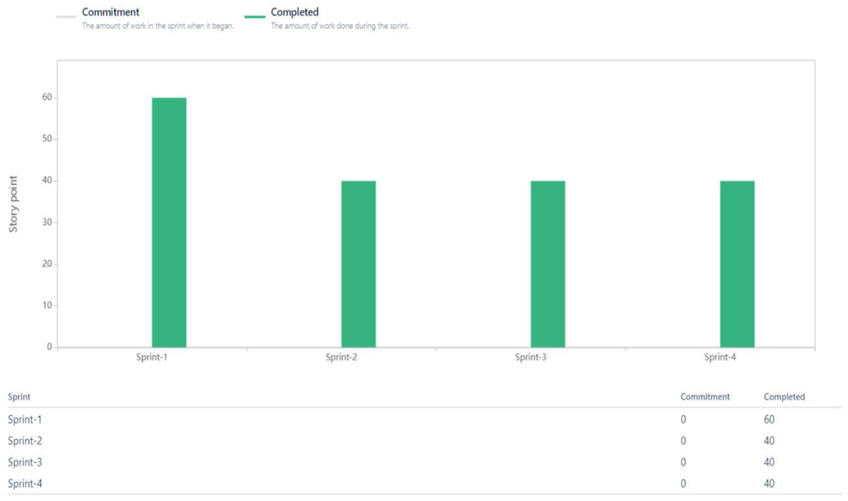
Sprint-2	IBM Cloud Server	USN-4	The forest fire is detected using computer vision algorithm based cameras. These cameras continuously monitor the forest and the data is sent to the server.	20	High	VANITHA LAKSHMI M
Sprint-2		USN-5	I can fetch the details/data from the cloud server.	20	High	VEERALAKSHMI P
Sprint-3	Data Collection	USN-6	I must gather information about forest fires.	20	High	VEERALAKSHMI P
Sprint-3		USN-7	I must draft and point out the algorithms to predict the forest fire.	20	Medium	MITHRRA SREE RA
Sprint-4	Algorithm Implementation	USN-8	I must determine the precision of each algorithm.	20	High	SHRI VARSHA R
Sprint-4		USN-9	Extracting and assessing the Dataset	20	High	VANITHA LAKSHMI M
Sprint-4	Evaluating the Algorithm	USN-10	I must determine the precision, recall and accuracy of the algorithm.	20	High	MITHRRA SREE RA

6.3 Reports from JIRA

Velocity Report:



Sprint	Commitment	Completed
Sprint-1	0	60
Sprint-2	0	40
Sprint-3	0	40
Sprint-4	0	0



Sprint	Commitment	Completed
Sprint-1	0	60
Sprint-2	0	40
Sprint-3	0	40
Sprint-4	0	40

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)

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

Average velocity of sprint-1: AV = 17/8 = 2.125
Average velocity of sprint-2: AV = 11/4 = 2.75
Average velocity of sprint-3: AV = 22/5 = 5.5
Average velocity of sprint-4: AV = 15/4 = 3.75

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

Feature

Here the arguments which are given inside the image data generator class are, rescale, shear_range, rotation range of image, and zoom range that we can consider for images, etc.

- Image shifts via the width_shift_range and height_shift_range arguments.
- Image flips via the horizontal_flip and vertical_flip arguments.
- Image rotations via the rotation_range argument
- Image brightness via the brightness_range argument.
- Image zoom via the zoom_range argument.

We can pass many other arguments inside the ImageDataGenerator Class.

The ImageDataGenerator class has three methods **flow ( ), flow_from_directory ( ), and flow_from_dataframe ( )** to read the images from a big numpy array and folders containing images.

flow_from_directory ( ) expects at least one directory under the given directory path.

Apply**flow_from_directory ( )**methodfor Train folder.

Now will apply the**flow_from_directory ( )**methodfortest folder.

- The directory must be set to the path where your training folders are present.
- The target_size is the size of your input images, every image will be resized to this size.
- batch_size: No. of images to be yielded from the generator per batch.
- “batch_size” in both train and test generators is to some number that divides your total number of images in your train set and train set respectively.
- class_mode: Set “binary” if you have only two classes to predict, if not set to “categorical”.

Keras has 2 ways to define a neural network:

- Sequential
- Function API

The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add () method.

Now, will initialize our model.

We will be adding three layers for CNN

- Convolution layer
- Pooling layer
- Flattening layer

Adding Convolutional Layer

The convolutional layer is the first and core layer of CNN. It is one of the building blocks of a CNN and is used for extracting important features from the image.

In the Convolution operation, the input image will be convolved with the feature detector/filters to get a feature map. The important role of the feature detector is to extract the features from the image. The group of feature maps is called a feature layer.

In the convolution2D function, we gave arguments that include 32,(3,3), that refers to we are applying 32 filters of 3x3 matrix filter, and input_shape is the input image shape with RGB, here 64x64 is the size and 3 represent the channel, RGB colour images.

Activation Function:These are the functions that help us to decide if we need to activate the node or not. These functions introduce non-linearity in the networks.

Adding Pooling Layer

Max Poolingselects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map. After the convolution layer, a pooling layer is added. Max pooling layer can be added using MaxPooling2D class. It takes the pool size as a parameter. Efficient size of the pooling matrix is (2,2). It returns the pooled feature maps.

Adding Flatten Layer

Now the pooled feature map from the pooling layer will be converted into one single dimension matrix or map, where each pixel in one single column, nothing but flattening. The flattening layer converts the multi-dimension matrix to one single dimension layer.

Adding Hidden layers

This step is to add a dense layer (hidden layer). We flatten the feature map and convert it into a vector or single dimensional array in the Flatten layer. This vector array is fed it as an input to the neural network and applies an activation function, such as sigmoid or other, and returns the output.

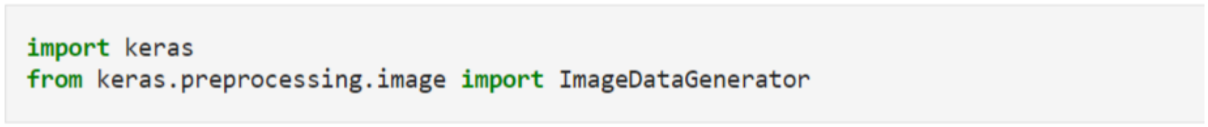
Adding output layer

This step is to add a dense layer (output layer) where you will be specifying the number of classes your dependent variable has, activation function and weight initializer as the arguments. We use add () method to add dense layers. In this layer, no need of mentioning input dimensions as we have mentions them in the above layer itself.

With both the training data defined and model defined, it's time to configure the learning process. This is accomplished with a call to the compile () method of the Sequential model class. Compilation requires 3 arguments: an optimizer, a loss function, and a list of metrics.

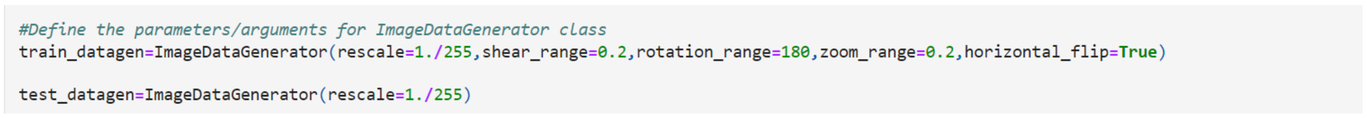
Your model is to be saved for future purposes. This saved model also is integrated with an android application or web application in order to predict something.

Importing_the_ImageDataGenerator



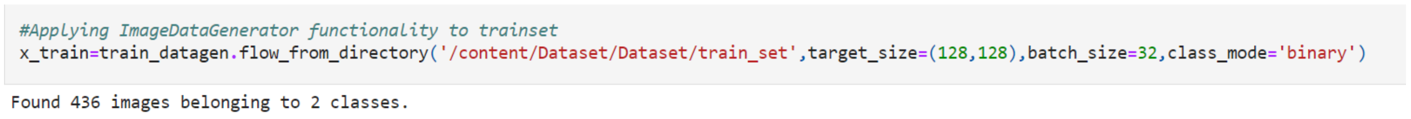
```
import keras
from keras.preprocessing.image import ImageDataGenerator
```

Parameters-ImageDataGenerator_class



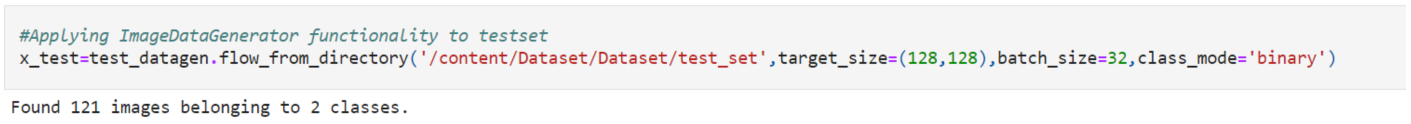
```
#Define the parameters/arguments for ImageDataGenerator class
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,rotation_range=180,zoom_range=0.2,horizontal_flip=True)

test_datagen=ImageDataGenerator(rescale=1./255)
```

ImageDataGenerator_functionality



```
#Applying ImageDataGenerator functionality to trainset
x_train=train_datagen.flow_from_directory('/content/Dataset/Dataset/train_set',target_size=(128,128),batch_size=32,class_mode='binary')
```

```
Found 436 images belonging to 2 classes.
```



```
#Applying ImageDataGenerator functionality to testset
x_test=test_datagen.flow_from_directory('/content/Dataset/Dataset/test_set',target_size=(128,128),batch_size=32,class_mode='binary')
```

```
Found 121 images belonging to 2 classes.
```

Model_Building_Libraries

```python
#import model building libraries

#To define Linear initialisation import Sequential
from keras.models import Sequential
#To add layers import Dense
from keras.layers import Dense
#To create Convolution kernel import Convolution2D
from keras.layers import Convolution2D
#import Maxpooling layer
from keras.layers import MaxPooling2D
#import flatten layer
from keras.layers import Flatten
import warnings
warnings.filterwarnings('ignore')
```

Initializing_Model

```python
#initializing the model
model=Sequential()
```

CNN_Layers

```python
#add convolutional layer
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
#add maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
#add flatten layer
model.add(Flatten())
```

Dense_Layers

```python
#add hidden layer
model.add(Dense(150,activation='relu'))
#add output layer
model.add(Dense(1,activation='sigmoid'))
```

Configure_learning_process

```python
#configure the learning process
model.compile(loss='binary_crossentropy',optimizer="adam",metrics=["accuracy"])
```

Training_model

```
#Training the model
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=x_test,validation_steps=4)
```

```
Epoch 1/10
14/14 [==============================] - 27s 2s/step - loss: 0.6515 - accuracy: 0.6445 - val_loss: 0.6824 - val_accuracy: 0.5950
Epoch 2/10
14/14 [==============================] - 27s 2s/step - loss: 0.6512 - accuracy: 0.6445 - val_loss: 0.6798 - val_accuracy: 0.5950
Epoch 3/10
14/14 [==============================] - 25s 2s/step - loss: 0.6510 - accuracy: 0.6445 - val_loss: 0.6803 - val_accuracy: 0.5950
Epoch 4/10
14/14 [==============================] - 25s 2s/step - loss: 0.6511 - accuracy: 0.6445 - val_loss: 0.6791 - val_accuracy: 0.5950
Epoch 5/10
14/14 [==============================] - 25s 2s/step - loss: 0.6509 - accuracy: 0.6445 - val_loss: 0.6803 - val_accuracy: 0.5950
Epoch 6/10
14/14 [==============================] - 25s 2s/step - loss: 0.6510 - accuracy: 0.6445 - val_loss: 0.6810 - val_accuracy: 0.5950
Epoch 7/10
14/14 [==============================] - 25s 2s/step - loss: 0.6509 - accuracy: 0.6445 - val_loss: 0.6805 - val_accuracy: 0.5950
Epoch 8/10
14/14 [==============================] - 25s 2s/step - loss: 0.6511 - accuracy: 0.6445 - val_loss: 0.6796 - val_accuracy: 0.5950
Epoch 9/10
14/14 [==============================] - 25s 2s/step - loss: 0.6510 - accuracy: 0.6445 - val_loss: 0.6804 - val_accuracy: 0.5950
Epoch 10/10
14/14 [==============================] - 25s 2s/step - loss: 0.6511 - accuracy: 0.6445 - val_loss: 0.6808 - val_accuracy: 0.5950
```

Save_Model

```
model.save("forest1.h5")
```

Prediction

```
img=image.load_img('/content/Dataset/Dataset/test_set/with fire/180802_CarrFire_010_large_700x467.jpg')
x=image.img_to_array(img)
res = cv2.resize(x, dsize=(128, 128), interpolation=cv2.INTER_CUBIC)
#expand the image shape
x=np.expand_dims(res,axis=0)
```

```
pred=model.predict(x)
```

```
1/1 [==============================] - 0s 37ms/step
```

```
pred
```

```
array([[1.]], dtype=float32)
```

Twilio_service_creation

```
pip install twilio
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting twilio
  Downloading twilio-7.15.0-py2.py3-none-any.whl (1.4 MB)
     |████████████████████████████████| 1.4 MB 5.3 MB/s
Requirement already satisfied: requests>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from twilio) (2.23.0)
Collecting PyJWT<3.0.0,>=2.0.0
  Downloading PyJWT-2.6.0-py3-none-any.whl (20 kB)
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from twilio) (2022.5)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (2022.9.24)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (1.2
4.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (2.10)
Installing collected packages: PyJWT, twilio
Successfully installed PyJWT-2.6.0 twilio-7.15.0
```

```
pip install pygobject
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pygobject in /usr/lib/python3/dist-packages (3.26.1)
```

OpenCV_video_processing

```python
#import opencv library
import cv2
#import numpy
import numpy as np
#import image function from keras
from tensorflow.keras.preprocessing import image
#import load_model from keras
from keras.models import load_model
#import Client from twilio API
from twilio.rest import Client
#import playsound package
from playsound import playsound
```

sending_alert_message

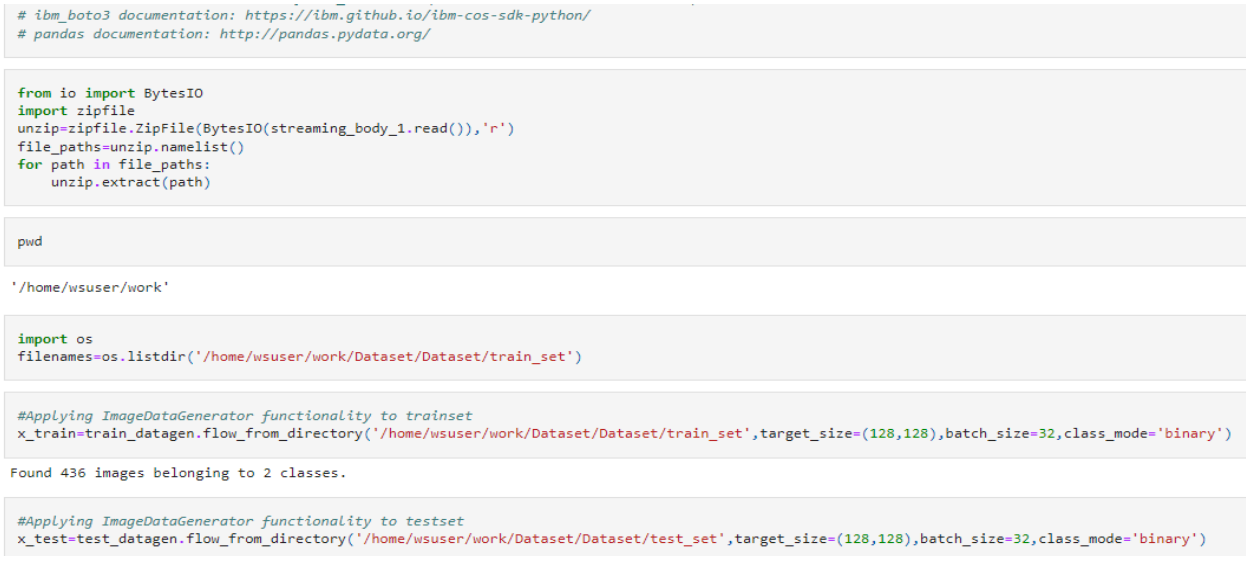
```python
#load the saved model
model=load_model('forest1.h5')
#define video
video=cv2.VideoCapture(0)
#define the features
name=['forest','with fire']
while(1):
  success,frame=video.read()
  cv2.imwrite("image.jpg",frame)
  img=image.load_img("image.jpg",target_size=(64,64))
  x=image.img_to_array(img)
  res = cv2.resize(x, dsize=(128, 128), interpolation=cv2.INTER_CUBIC)
  x=np.expand_dims(res,axis=0)
  pred=model.predict(x)
  p=pred[0]
  print(pred)
  #cv2.putText(frame,"predicted class = "+str(name[p]),(100,100),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,0))
  if pred[0]==1:
    #twilio account ssid
    account_sid='ACb252d719e2b09dbb3f44dd2f8c8be56a'
    #twilio account authentication token
    auth_token ='d401dbcf96a018136bc7ad3ded613273'
    client=Client(account_sid,auth_token)

    message=client.messages \
    .create(
        body='Forest Fire is detected,stay alert',
        #use twilio free number
        from_='+1 980 414 5862',
        #to number
        to='+91 9080590163')
    print(message.sid)
    print('Fire Detected')
    print('SMS sent!')
    playsound('/tornado-siren-in-streamwood-il-35510.mp3')
  else:
    print('No Danger')
    #break
  cv2.imshow("image",frame)
  if cv2.waitkey(1) & 0xFF == ord('a'):
     break
video.release()
cv2.destroyAllWindows()
```

TRAIN MODEL ON IBM

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

```
#Define the parameters/arguments for ImageDataGenerator class
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,rotation_range=180,zoom_range=0.2,horizontal_flip=True)

test_datagen=ImageDataGenerator(rescale=1./255)
```

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='CyfbVlhpb88HCt21fx4u1-RwsYc8kWTEnhyO199kN_R5',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'forestfiredetection-donotdelete-pr-u0rtatnl4rnifa'
object_key = 'ff.zip'

streaming_body_1 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
```



```
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
```

```
from io import BytesIO
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_1.read()),'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

```
pwd
```

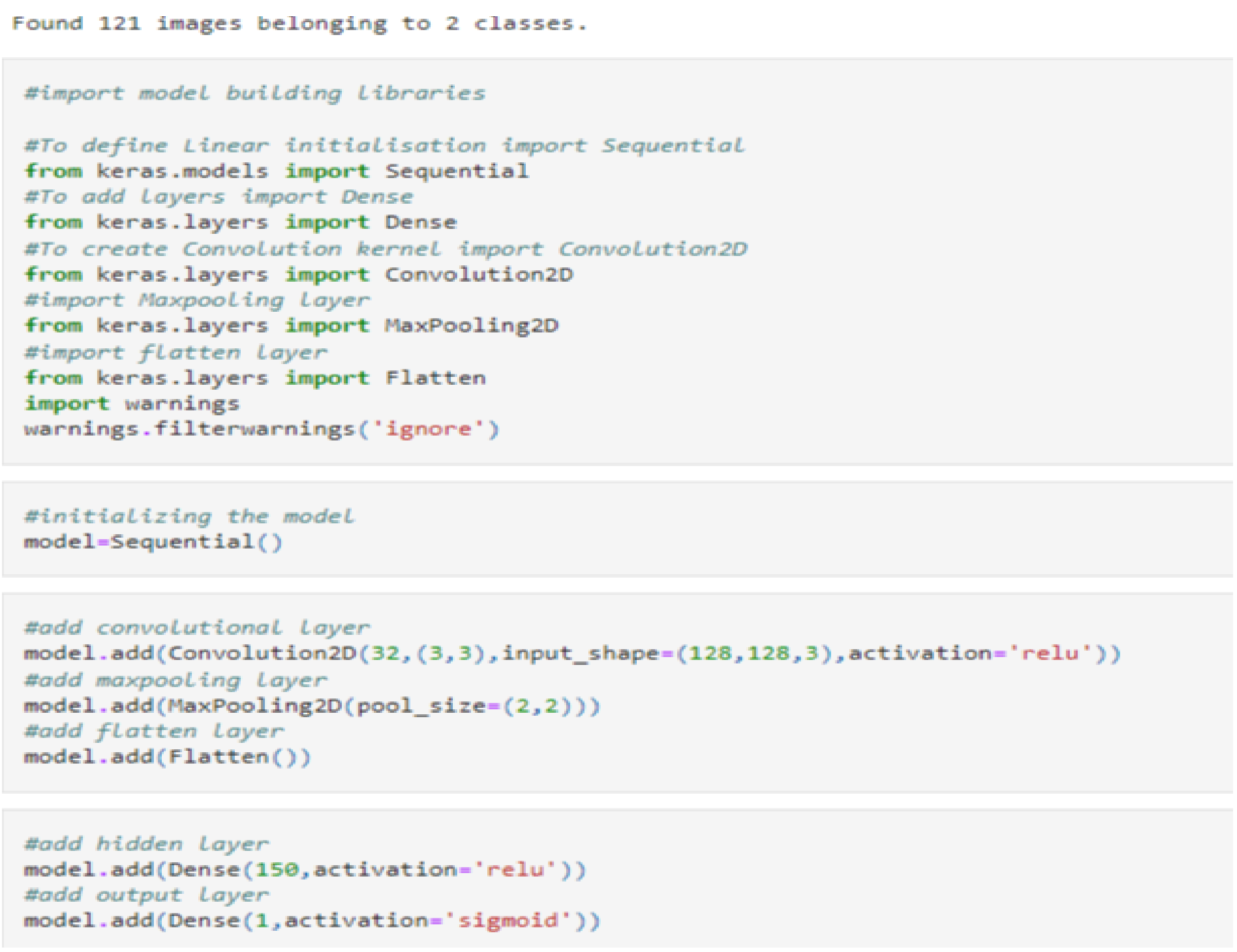
```
'/home/wsuser/work'
```

```
import os
filenames=os.listdir('/home/wsuser/work/Dataset/Dataset/train_set')
```

```
#Applying ImageDataGenerator functionality to trainset
x_train=train_datagen.flow_from_directory('/home/wsuser/work/Dataset/Dataset/train_set',target_size=(128,128),batch_size=32,class_mode='binary')
```

```
Found 436 images belonging to 2 classes.
```

```
#Applying ImageDataGenerator functionality to testset
x_test=test_datagen.flow_from_directory('/home/wsuser/work/Dataset/Dataset/test_set',target_size=(128,128),batch_size=32,class_mode='binary')
```



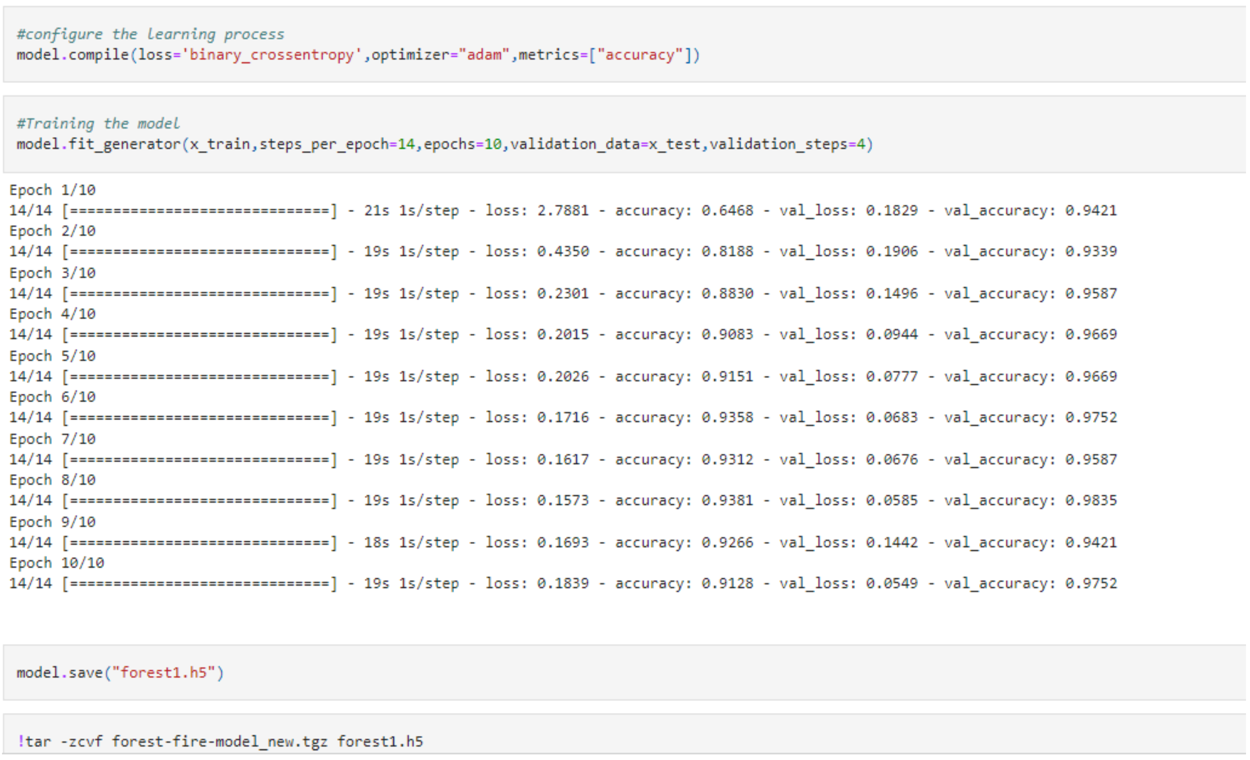
```
Found 121 images belonging to 2 classes.
```

```
#import model building libraries

#To define Linear initialisation import Sequential
from keras.models import Sequential
#To add layers import Dense
from keras.layers import Dense
#To create Convolution kernel import Convolution2D
from keras.layers import Convolution2D
#import Maxpooling layer
from keras.layers import MaxPooling2D
#import flatten layer
from keras.layers import Flatten
import warnings
warnings.filterwarnings('ignore')
```

```
#initializing the model
model=Sequential()
```

```
#add convolutional layer
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
#add maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
#add flatten layer
model.add(Flatten())
```

```
#add hidden layer
model.add(Dense(150,activation='relu'))
#add output layer
model.add(Dense(1,activation='sigmoid'))
```



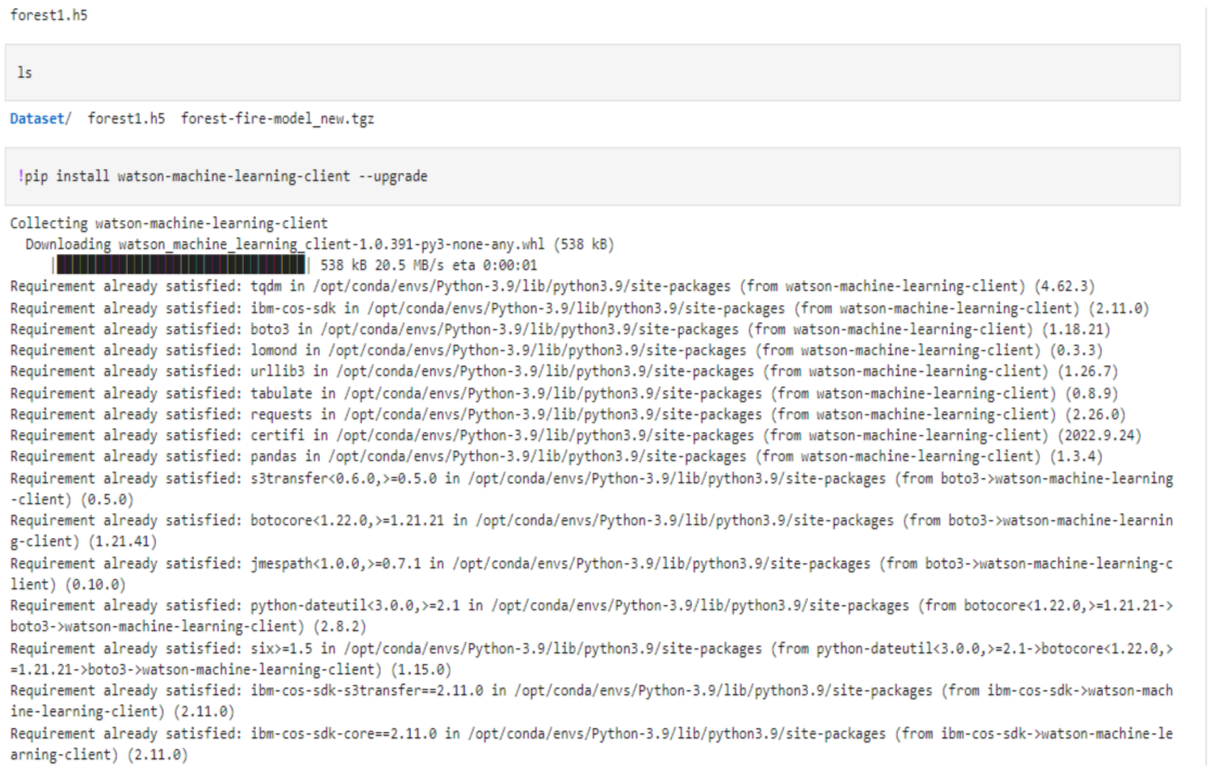
```
#configure the learning process
model.compile(loss='binary_crossentropy',optimizer="adam",metrics=["accuracy"])
```

```
#Training the model
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=x_test,validation_steps=4)
```

```
Epoch 1/10
14/14 [==============================] - 21s 1s/step - loss: 2.7881 - accuracy: 0.6468 - val_loss: 0.1829 - val_accuracy: 0.9421
Epoch 2/10
14/14 [==============================] - 19s 1s/step - loss: 0.4350 - accuracy: 0.8188 - val_loss: 0.1906 - val_accuracy: 0.9339
Epoch 3/10
14/14 [==============================] - 19s 1s/step - loss: 0.2301 - accuracy: 0.8830 - val_loss: 0.1496 - val_accuracy: 0.9587
Epoch 4/10
14/14 [==============================] - 19s 1s/step - loss: 0.2015 - accuracy: 0.9083 - val_loss: 0.0944 - val_accuracy: 0.9669
Epoch 5/10
14/14 [==============================] - 19s 1s/step - loss: 0.2026 - accuracy: 0.9151 - val_loss: 0.0777 - val_accuracy: 0.9669
Epoch 6/10
14/14 [==============================] - 19s 1s/step - loss: 0.1716 - accuracy: 0.9358 - val_loss: 0.0683 - val_accuracy: 0.9752
Epoch 7/10
14/14 [==============================] - 19s 1s/step - loss: 0.1617 - accuracy: 0.9312 - val_loss: 0.0676 - val_accuracy: 0.9587
Epoch 8/10
14/14 [==============================] - 19s 1s/step - loss: 0.1573 - accuracy: 0.9381 - val_loss: 0.0585 - val_accuracy: 0.9835
Epoch 9/10
14/14 [==============================] - 18s 1s/step - loss: 0.1693 - accuracy: 0.9266 - val_loss: 0.1442 - val_accuracy: 0.9421
Epoch 10/10
14/14 [==============================] - 19s 1s/step - loss: 0.1839 - accuracy: 0.9128 - val_loss: 0.0549 - val_accuracy: 0.9752
```

```
model.save("forest1.h5")
```

```
!tar -zcvf forest-fire-model_new.tgz forest1.h5
```



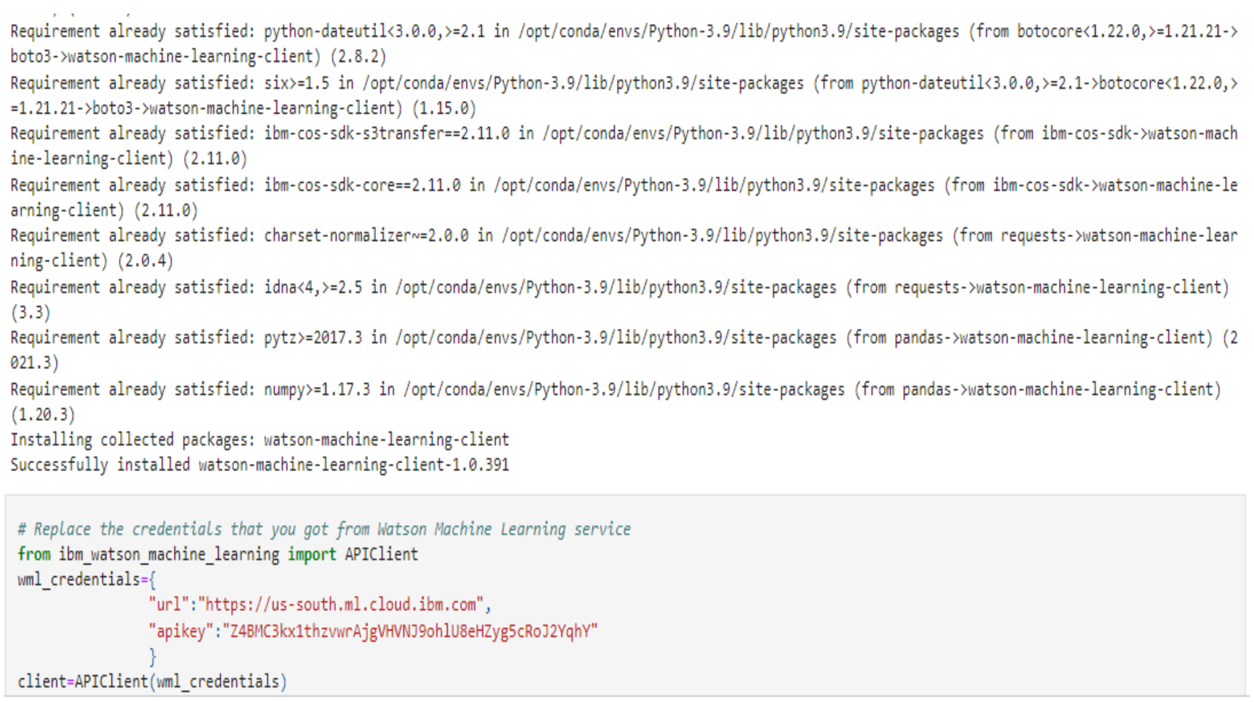
```
forest1.h5
```

```
ls
```

```
Dataset/  forest1.h5  forest-fire-model_new.tgz
```

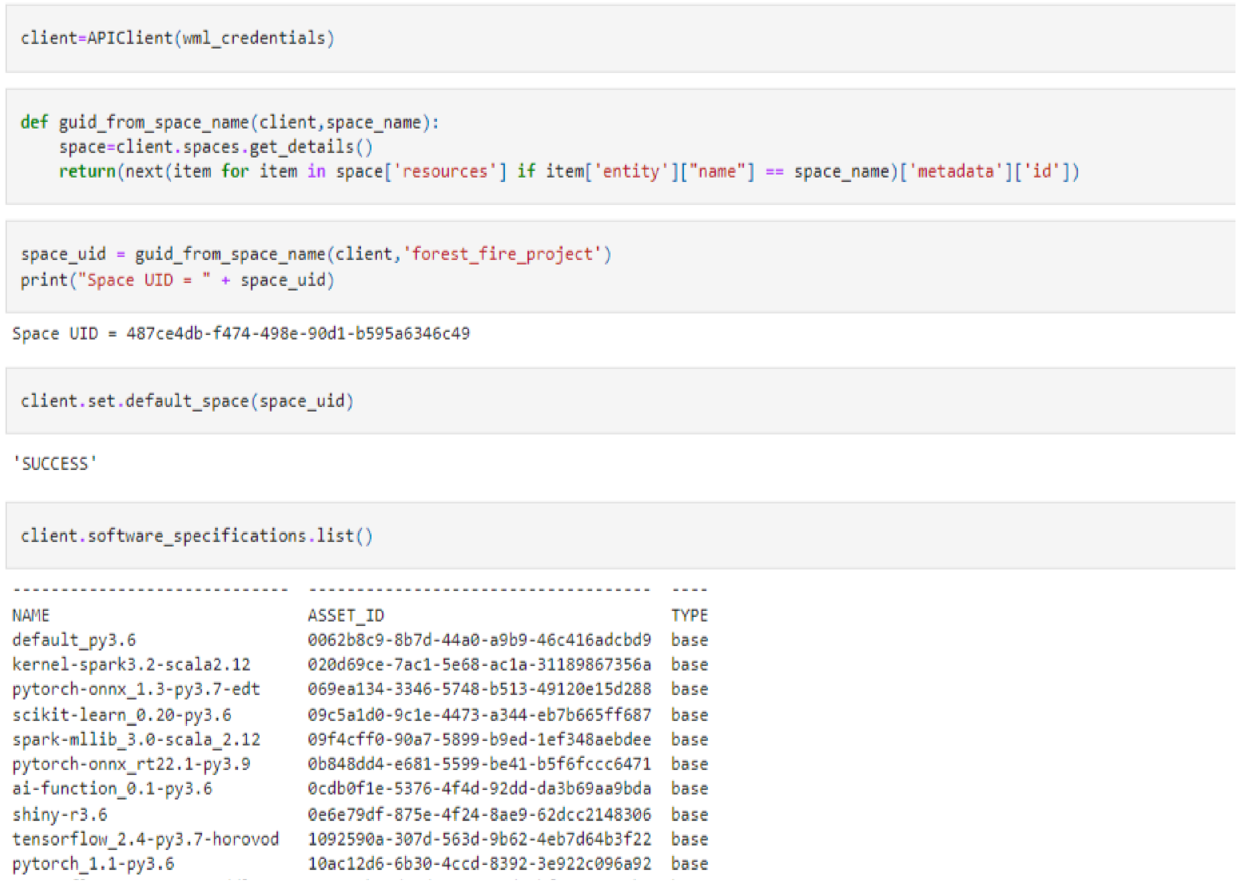
```
!pip install watson-machine-learning-client --upgrade
```

```
Collecting watson-machine-learning-client
  Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
     |████████████████████████████████| 538 kB 20.5 MB/s eta 0:00:01
Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (4.62.3)
Requirement already satisfied: ibm-cos-sdk in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.11.0)
Requirement already satisfied: boto3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.18.21)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.26.7)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.8.9)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.26.0)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2022.9.24)
Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.3.4)
Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning
-client) (0.5.0)
Requirement already satisfied: botocore<1.22.0,>=1.21.21 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learnin
g-client) (1.21.41)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-c
lient) (0.10.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from botocore<1.22.0,>=1.21.21->
boto3->watson-machine-learning-client) (2.8.2)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.22.0,>
=1.21.21->boto3->watson-machine-learning-client) (1.15.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-mach
ine-learning-client) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machine-le
arning-client) (2.11.0)
```



```
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from botocore<1.22.0,>=1.21.21->
boto3->watson-machine-learning-client) (2.8.2)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.22.0,>
=1.21.21->boto3->watson-machine-learning-client) (1.15.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-mach
ine-learning-client) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machine-le
arning-client) (2.11.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->watson-machine-lear
ning-client) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->watson-machine-learning-client)
(3.3)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client) (2
021.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client)
(1.20.3)
Installing collected packages: watson-machine-learning-client
Successfully installed watson-machine-learning-client-1.0.391
```

```python
# Replace the credentials that you got from Watson Machine Learning service
from ibm_watson_machine_learning import APIClient
wml_credentials={
                 "url":"https://us-south.ml.cloud.ibm.com",
                 "apikey":"Z4BMC3kx1thzvwrAjgVHVNJ9ohlU8eHZyg5cRoJ2YqhY"
                 }
client=APIClient(wml_credentials)
```



```python
client=APIClient(wml_credentials)
```

```python
def guid_from_space_name(client,space_name):
    space=client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']["name"] == space_name)['metadata']['id'])
```

```python
space_uid = guid_from_space_name(client,'forest_fire_project')
print("Space UID = " + space_uid)
```

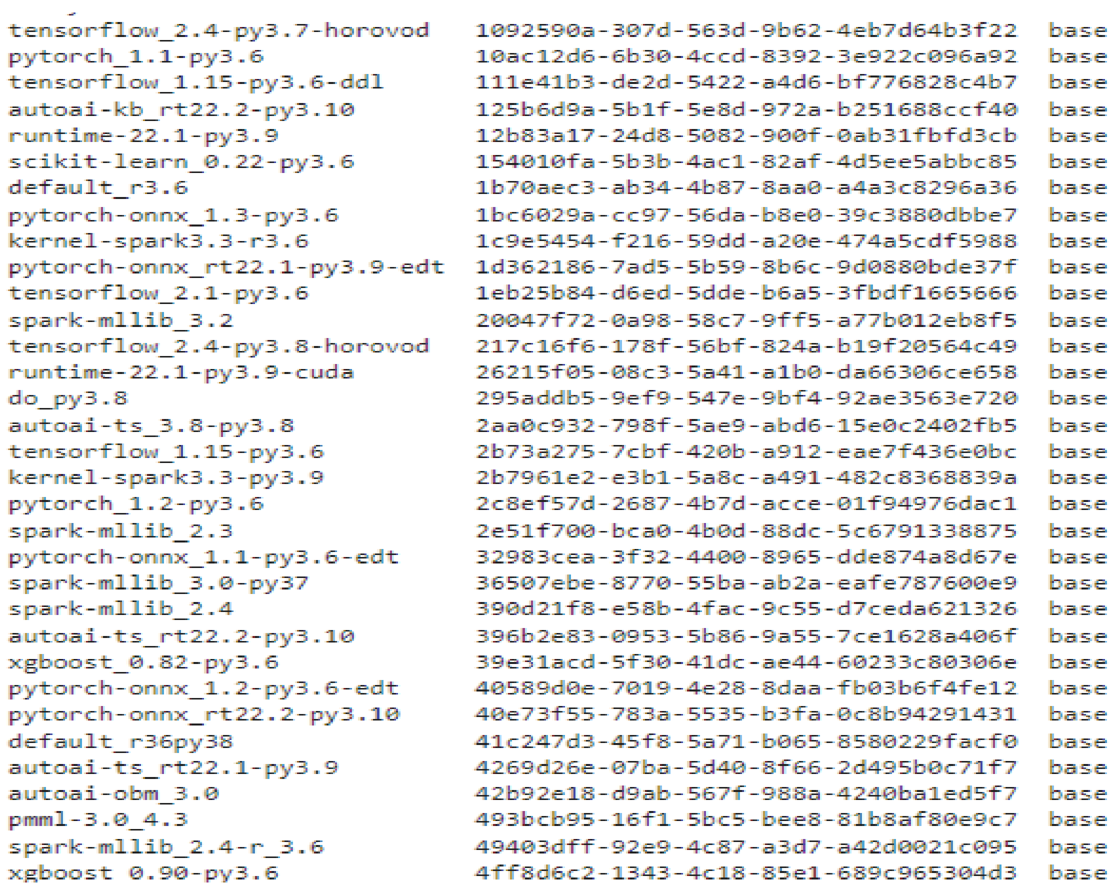
```
Space UID = 487ce4db-f474-498e-90d1-b595a6346c49
```

```python
client.set.default_space(space_uid)
```

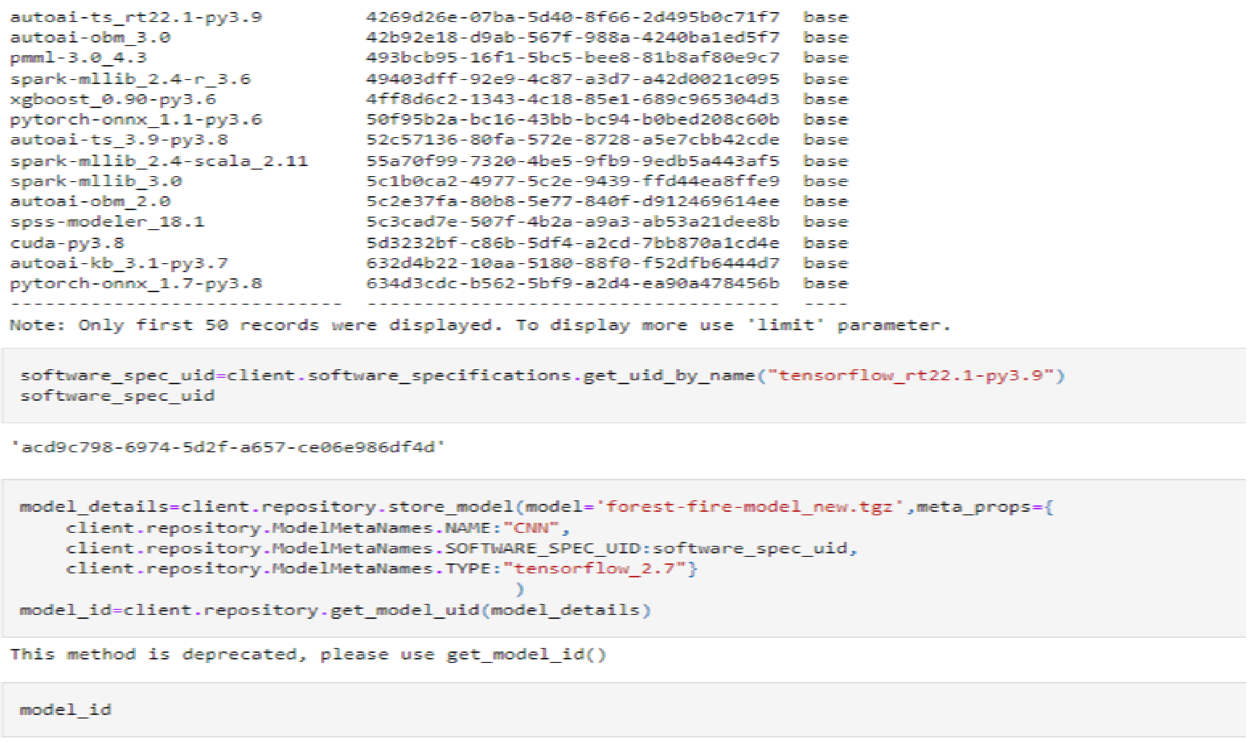
```
'SUCCESS'
```

```python
client.software_specifications.list()
```

```
----------------------------  ------------------------------------  ----
NAME                          ASSET_ID                              TYPE
default_py3.6                 0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12     020d69ce-7ac1-5e68-ac1a-31189867356a  base
pytorch-onnx_1.3-py3.7-edt    069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6       09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12    09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
pytorch-onnx_rt22.1-py3.9     0b848dd4-e681-5599-be41-b5f6fccc6471  base
ai-function_0.1-py3.6         0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
shiny-r3.6                    0e6e79df-875e-4f24-8ae9-62dcc2148306  base
tensorflow_2.4-py3.7-horovod  1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6             10ac12d6-6b30-4ccd-8392-3e922c096a92  base
```



```
tensorflow_2.4-py3.7-horovod    1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6               10ac12d6-6b30-4ccd-8392-3e922c096a92  base
tensorflow_1.15-py3.6-ddl       111e41b3-de2d-5422-a4d6-bf776828c4b7  base
autoai-kb_rt22.2-py3.10         125b6d9a-5b1f-5e8d-972a-b251688ccf40  base
runtime-22.1-py3.9              12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
scikit-learn_0.22-py3.6         154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                    1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
pytorch-onnx_1.3-py3.6          1bc6029a-cc97-56da-b8e0-39c3880dbbe7  base
kernel-spark3.3-r3.6            1c9e5454-f216-59dd-a20e-474a5cdf5988  base
pytorch-onnx_rt22.1-py3.9-edt   1d362186-7ad5-5b59-8b6c-9d0880bde37f  base
tensorflow_2.1-py3.6            1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base
spark-mllib_3.2                 20047f72-0a98-58c7-9ff5-a77b012eb8f5  base
tensorflow_2.4-py3.8-horovod    217c16f6-178f-56bf-824a-b19f20564c49  base
runtime-22.1-py3.9-cuda         26215f05-08c3-5a41-a1b0-da66306ce658  base
do_py3.8                        295addb5-9ef9-547e-9bf4-92ae3563e720  base
autoai-ts_3.8-py3.8             2aa0c932-798f-5ae9-abd6-15e0c2402fb5  base
tensorflow_1.15-py3.6           2b73a275-7cbf-420b-a912-eae7f436e0bc  base
kernel-spark3.3-py3.9           2b7961e2-e3b1-5a8c-a491-482c8368839a  base
pytorch_1.2-py3.6               2c8ef57d-2687-4b7d-acce-01f94976dac1  base
spark-mllib_2.3                 2e51f700-bca0-4b0d-88dc-5c6791338875  base
pytorch-onnx_1.1-py3.6-edt      32983cea-3f32-4400-8965-dde874a8d67e  base
spark-mllib_3.0-py37            36507ebe-8770-55ba-ab2a-eafe787600e9  base
spark-mllib_2.4                 390d21f8-e58b-4fac-9c55-d7ceda621326  base
autoai-ts_rt22.2-py3.10         396b2e83-0953-5b86-9a55-7ce1628a406f  base
xgboost_0.82-py3.6              39e31acd-5f30-41dc-ae44-60233c80306e  base
pytorch-onnx_1.2-py3.6-edt      40589d0e-7019-4e28-8daa-fb03b6f4fe12  base
pytorch-onnx_rt22.2-py3.10      40e73f55-783a-5535-b3fa-0c8b94291431  base
default_r36py38                 41c247d3-45f8-5a71-b065-8580229facf0  base
autoai-ts_rt22.1-py3.9          4269d26e-07ba-5d40-8f66-2d495b0c71f7  base
autoai-obm_3.0                  42b92e18-d9ab-567f-988a-4240ba1ed5f7  base
pmml-3.0_4.3                    493bcb95-16f1-5bc5-bee8-81b8af80e9c7  base
spark-mllib_2.4-r_3.6           49403dff-92e9-4c87-a3d7-a42d0021c095  base
xgboost_0.90-py3.6              4ff8d6c2-1343-4c18-85e1-689c965304d3  base
```



```
autoai-ts_rt22.1-py3.9          4269d26e-07ba-5d40-8f66-2d495b0c71f7  base
autoai-obm_3.0                  42b92e18-d9ab-567f-988a-4240ba1ed5f7  base
pmml-3.0_4.3                    493bcb95-16f1-5bc5-bee8-81b8af80e9c7  base
spark-mllib_2.4-r_3.6           49403dff-92e9-4c87-a3d7-a42d0021c095  base
xgboost_0.90-py3.6              4ff8d6c2-1343-4c18-85e1-689c965304d3  base
pytorch-onnx_1.1-py3.6          50f95b2a-bc16-43bb-bc94-b0bed208c60b  base
autoai-ts_3.9-py3.8             52c57136-80fa-572e-8728-a5e7cbb42cde  base
spark-mllib_2.4-scala_2.11      55a70f99-7320-4be5-9fb9-9edb5a443af5  base
spark-mllib_3.0                 5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9  base
autoai-obm_2.0                  5c2e37fa-80b8-5e77-840f-d912469614ee  base
spss-modeler_18.1               5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b  base
cuda-py3.8                      5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e  base
autoai-kb_3.1-py3.7             632d4b22-10aa-5180-88f0-f52dfb6444d7  base
pytorch-onnx_1.7-py3.8          634d3cdc-b562-5bf9-a2d4-ea90a478456b  base
------------------------------  ------------------------------------  ----
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
```

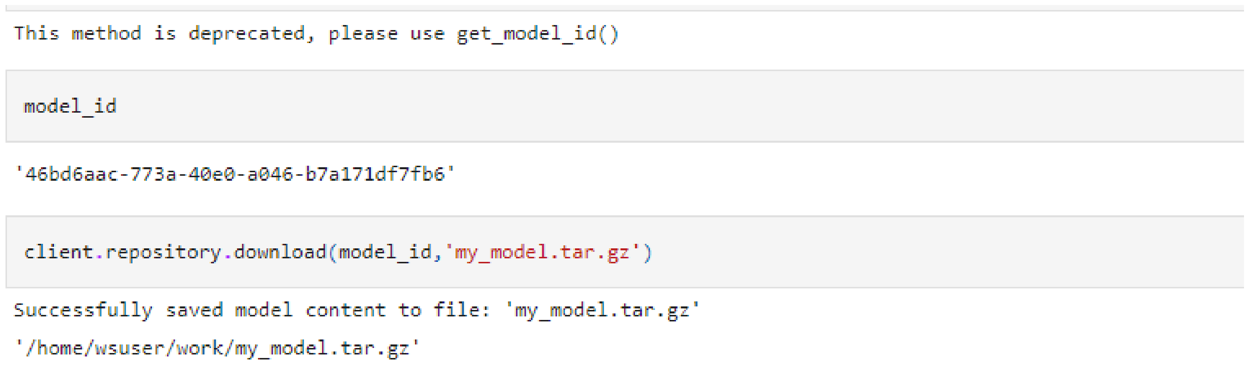
```
software_spec_uid=client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
software_spec_uid
```

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

```
model_details=client.repository.store_model(model='forest-fire-model_new.tgz',meta_props={
    client.repository.ModelMetaNames.NAME:"CNN",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid,
    client.repository.ModelMetaNames.TYPE:"tensorflow_2.7"}
                                            )
model_id=client.repository.get_model_uid(model_details)
```

```
This method is deprecated, please use get_model_id()
```

```
model_id
```



```
This method is deprecated, please use get_model_id()
```

```
model_id
```

```
'46bd6aac-773a-40e0-a046-b7a171df7fb6'
```

```
client.repository.download(model_id,'my_model.tar.gz')
```

```
Successfully saved model content to file: 'my_model.tar.gz'
'/home/wsuser/work/my_model.tar.gz'
```

8. TESTING

8.1 Test Cases

Model Performance Testing:

S.No.	Parameter	Values
1.	Model Summary	Total params: 19,052,397 Trainable params: 19,052,397 Non-trainable params: 0
2.	Accuracy	Training Accuracy – 92.43 Validation Accuracy – 97.52

8.2 User Acceptance Testing

Purpose of Document

The purpose of this document is to briefly explain the test coverageand open issuesof the early detection of forest fires project at the time of the release to User Acceptance Testing (UAT).

Defect Analysis

This report showsthe number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

Test Case Analysis

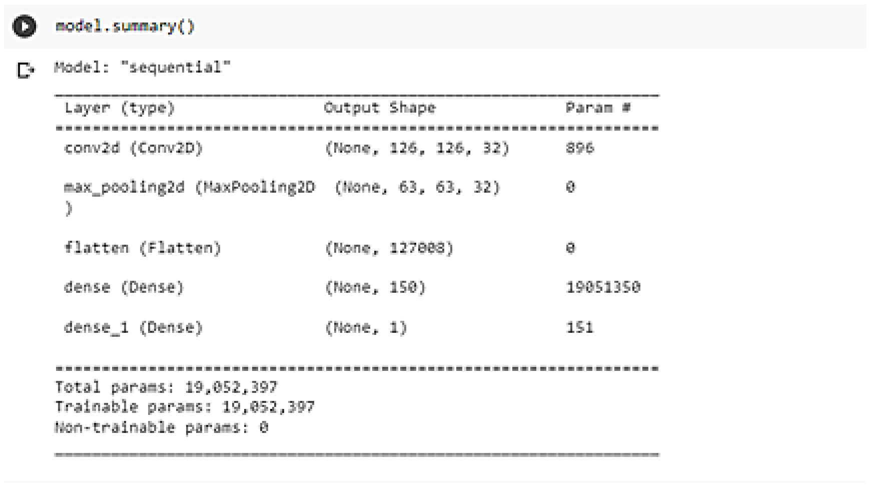
This report shows the number of test cases that have passed, failed,and untested.

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	8	0	0	8
Client Application	24	0	0	24
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	1	8
Final Report Output	5	0	0	5
Version Control	2	0	0	2

9. RESULTS

9.1 Performance Metrics

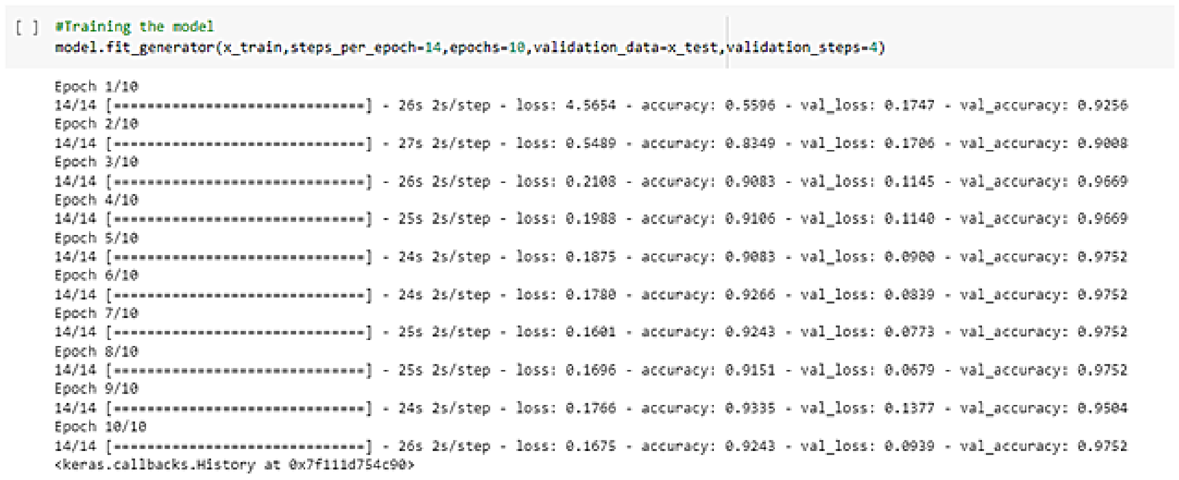
Screenshot:
Model Summary:



```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0
 )

 flatten (Flatten)           (None, 127008)            0

 dense (Dense)               (None, 150)               19051350

 dense_1 (Dense)             (None, 1)                 151

=================================================================
Total params: 19,052,397
Trainable params: 19,052,397
Non-trainable params: 0
_________________________________________________________________
```

Accuracy:



```
#Training the model
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=x_test,validation_steps=4)
```

```
Epoch 1/10
14/14 [==============================] - 26s 2s/step - loss: 4.5654 - accuracy: 0.5596 - val_loss: 0.1747 - val_accuracy: 0.9256
Epoch 2/10
14/14 [==============================] - 27s 2s/step - loss: 0.5489 - accuracy: 0.8349 - val_loss: 0.1706 - val_accuracy: 0.9008
Epoch 3/10
14/14 [==============================] - 26s 2s/step - loss: 0.2108 - accuracy: 0.9083 - val_loss: 0.1145 - val_accuracy: 0.9669
Epoch 4/10
14/14 [==============================] - 25s 2s/step - loss: 0.1988 - accuracy: 0.9106 - val_loss: 0.1140 - val_accuracy: 0.9669
Epoch 5/10
14/14 [==============================] - 24s 2s/step - loss: 0.1875 - accuracy: 0.9083 - val_loss: 0.0900 - val_accuracy: 0.9752
Epoch 6/10
14/14 [==============================] - 24s 2s/step - loss: 0.1780 - accuracy: 0.9266 - val_loss: 0.0839 - val_accuracy: 0.9752
Epoch 7/10
14/14 [==============================] - 25s 2s/step - loss: 0.1601 - accuracy: 0.9243 - val_loss: 0.0773 - val_accuracy: 0.9752
Epoch 8/10
14/14 [==============================] - 25s 2s/step - loss: 0.1696 - accuracy: 0.9151 - val_loss: 0.0679 - val_accuracy: 0.9752
Epoch 9/10
14/14 [==============================] - 24s 2s/step - loss: 0.1766 - accuracy: 0.9335 - val_loss: 0.1377 - val_accuracy: 0.9504
Epoch 10/10
14/14 [==============================] - 26s 2s/step - loss: 0.1675 - accuracy: 0.9243 - val_loss: 0.0939 - val_accuracy: 0.9752
<keras.callbacks.History at 0x7f111d754c90>
```

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES :

- This approach takes advantages of both brightness and motion features of fire in images to improve the accuracy and reliability of forest fire detection.
- Fire detection systems **increase response times**, as they are able to alert the correct people in order to extinguish the fire
- This reduces the amount of damage to the property.

DISADVANTAGES:

- It may result in over power consumption
- perfect accuracy may not be possible
- implementation cost is high

11. CONCLUSION

Initially the dataset has been prepared and we had done image processing for the images in the dataset to predict the forest fire. If the fire is predicted it alerts the respective officer by sending a message.

12. FUTURE SCOPE

The proposed system can be developed to more advanced system by integrating wireless sensors with CCTV for added protection and precision. The algorithm shows great promise in adapting to various environment.

13. APPENDIX

Source code

```
import keras
from keras.preprocessing.image import ImageDataGenerator
#Define the parameters/arguments for ImageDataGenerator class
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,rotation_range=180,zoom_
range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
#Applying ImageDataGenerator functionality to trainset
x_train=train_datagen.flow_from_directory('/content/Dataset/Dataset/train_set',target_size=(12
8,128),batch_size=32,class_mode='binary')
#Applying ImageDataGenerator functionality to testset
x_test=test_datagen.flow_from_directory('/content/Dataset/Dataset/test_set',target_size=(128,1
28),batch_size=32,class_mode='binary')
#import model building libraries
```

```
#To define Linear initialisation import Sequential
from keras.models import Sequential
#To add layers import Dense
from keras.layers import Dense
#To create Convolution kernel import Convolution2D
from keras.layers import Convolution2D
#import Maxpooling layer
from keras.layers import MaxPooling2D
#import flatten layer
from keras.layers import Flatten
import warnings
warnings.filterwarnings('ignore')
#initializing the model
model=Sequential()
#add convolutional layer
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
#add maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
#add flatten layer
model.add(Flatten())
#add hidden layer
model.add(Dense(150,activation='relu'))
#add output layer
model.add(Dense(1,activation='sigmoid'))
#configure the learning process
model.compile(loss='binary_crossentropy',optimizer="adam",metrics=["accuracy"])
#Training the model
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=x_test,validation_st
eps=4)
model.save("forest1.h5")
model.summary()
#import load_model from keras.model
from keras.models import load_model
```

```
#import image class from keras
from tensorflow.keras.preprocessing import image
#import numpy
import numpy as np
#import cv2
import cv2
#load the saved model
model = load_model("forest1.h5")
img=image.load_img('/content/Dataset/Dataset/test_set/with
fire/180802_CarrFire_010_large_700x467.jpg')
x=image.img_to_array(img)
res = cv2.resize(x, dsize=(128, 128), interpolation=cv2.INTER_CUBIC)
#expand the image shape
x=np.expand_dims(res,axis=0)
pred=model.predict(x)
pred
pip install twilio
pip install pygobject
#import opencv library
import cv2
#import numpy
import numpy as np
#import image function from keras
from tensorflow.keras.preprocessing import image
#import load_model from keras
from keras.models import load_model
#import Client from twilio API
from twilio.rest import Client
#import playsound package
from playsound import playsound
#load the saved model
model=load_model('forest1.h5')
#define video
video=cv2.VideoCapture(0)
```

```
#define the features
name=['forest','with fire']
while(1):
  success,frame=video.read()
  cv2.imwrite("image.jpg",frame)
  img=image.load_img("image.jpg",target_size=(64,64))
  x=image.img_to_array(img)
  res = cv2.resize(x, dsize=(128, 128), interpolation=cv2.INTER_CUBIC)
  x=np.expand_dims(res,axis=0)
  pred=model.predict(x)
  p=pred[0]
  print(pred)
  cv2.putText(frame,"predicted class =
"+str(p),(100,100),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,0))
  if pred[0]==1:
    #twilio account ssid
    account_sid='ACb252d719e2b09dbb3f44dd2f8c8be56a'
    #twilio account authentication token
    auth_token ='d401dbcf96a018136bc7ad3ded613273'
    client=Client(account_sid,auth_token)

    message=client.messages \
    .create(
        body='Forest Fire is detected,stay alert',
        #use twilio free number
        from_='+1 980 414 5862',
        #to number
        to='+91 9080590163')
    print(message.sid)
    print('Fire Detected')
    print('SMS sent!')
    playsound('/tornado-siren-in-streamwood-il-35510.mp3')
  else:
    print('No Danger')
    #break
  cv2.imshow("image",frame)
  if cv2.waitKey(1) & 0xFF == ord('a'):
    break
video.release()
cv2.destroyAllWindows()
```

TRAIN MODEL ON IBM

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

#Define the parameters/arguments for ImageDataGenerator class
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,rotation_range=180,zoom_
range=0.2,horizontal_flip=True)

test_datagen=ImageDataGenerator(rescale=1./255)

import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your
credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='CyfbVlhpb88HCt21fx4u1-RwsYc8kWTEnhyO199kN_R5',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'forestfiredetection-donotdelete-pr-u0rtatnl4rnifa'
object_key = 'ff.zip'

streaming_body_1 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']
```

```
# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities
to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/

from io import BytesIO
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_1.read()),'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)

pwd

import os
filenames=os.listdir('/home/wsuser/work/Dataset/Dataset/train_set')

#Applying ImageDataGenerator functionality to trainset
x_train=train_datagen.flow_from_directory('/home/wsuser/work/Dataset/Dataset/train_set',targ
et_size=(128,128),batch_size=32,class_mode='binary')


#Applying ImageDataGenerator functionality to testset
x_test=test_datagen.flow_from_directory('/home/wsuser/work/Dataset/Dataset/test_set',target
_size=(128,128),batch_size=32,class_mode='binary')

#import model building libraries

#To define Linear initialisation import Sequential
from keras.models import Sequential
#To add layers import Dense
from keras.layers import Dense
#To create Convolution kernel import Convolution2D
from keras.layers import Convolution2D
#import Maxpooling layer
from keras.layers import MaxPooling2D
#import flatten layer
from keras.layers import Flatten
import warnings
```

```
warnings.filterwarnings('ignore')

#initializing the model
model=Sequential()

#add convolutional layer
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
#add maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
#add flatten layer
model.add(Flatten())

#add hidden layer
model.add(Dense(150,activation='relu'))
#add output layer
model.add(Dense(1,activation='sigmoid'))

#configure the learning process
model.compile(loss='binary_crossentropy',optimizer="adam",metrics=["accuracy"])

#Training the model
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=x_test,validation_st
eps=4)

model.save("forest1.h5")

!tar -zcvf forest-fire-model_new.tgz forest1.h5

ls

!pip install watson-machine-learning-client --upgrade

# Replace the credentials that you got from Watson Machine Learning service
from ibm_watson_machine_learning import APIClient
wml_credentials={
            "url":"https://us-south.ml.cloud.ibm.com",
            "apikey":"Z4BMC3kx1thzvwrAjgVHVNJ9ohlU8eHZyg5cRoJ2YqhY"
            }
client=APIClient(wml_credentials)

client=APIClient(wml_credentials)
```

```
def guid_from_space_name(client,space_name):
   space=client.spaces.get_details()
   return(next(item for item in space['resources'] if item['entity']["name"] ==
space_name)['metadata']['id'])

space_uid = guid_from_space_name(client,'forest_fire_project')
print("Space UID = " + space_uid)

client.set.default_space(space_uid)

client.software_specifications.list()

software_spec_uid=client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
software_spec_uid

model_details=client.repository.store_model(model='forest-fire-model_new.tgz',meta_props={
   client.repository.ModelMetaNames.NAME:"CNN",
   client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid,
   client.repository.ModelMetaNames.TYPE:"tensorflow_2.7"}
                                        )
model_id=client.repository.get_model_uid(model_details)

model_id

client.repository.download(model_id,'my_model.tar.gz')
```

GitHub & Project Demo Link

https://drive.google.com/drive/folders/1CLcc9pZRTwOkvt_avwqTxWTO1TTHddk2?usp=share_link