Project Report Format

Team id	**PNT2022TMID49509**
Project title	**Emerging methods for early detection of forest fires**
Date	**18/11/2022**

1. Introduction

Forests are the valuable assets of the world. We should conserve forest to save our future. There are many ways that the forest can be destroyed. The major cause for the loss of forest is forest fire. Forest fire may be natural or man-made. We had created a forest fire detection model to find fire at the earliest.

1.1. Project Overview

This project is created by identifying the fire at the forest using images and video. We had trained the model using various images and tested it.

1.2. Purpose

The forest fires destroy 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 issues in the present decade.

2. Literature survey

2.1. Existing problem

ABSTRACT

Natural disasters have been causing havoc since time immemorial. Forest and rural fires are one of the main causes of environmental degradation. Wireless Sensor Networks (WSN) have been fruitful in monitoring areas remotely and detecting environmental changes. By incorporation of Data Mining and Machine Learning techniques, we can build a system for early detection of fire disasters. WSNs based on Internet of Things (IoT) helps us in remote monitoring over the internet and prediction of an event as Fire/No Fire. With multi-criteria detection, multiple attributes of a forest fire are sensed by different sensing units. The temporal data from the sensors is collected and various machine learning techniques are used to analyze the patterns of data and use them to develop classification and prediction models. Model construction is done based on available data whereas model updating and prediction is in the real-time scenario. According to the data fed from sensors onset of fire can be detected and so the warning can be raised and sent to the authorities. Early detection and prediction of fire hazards help in improving firefighting resource management and reducing the damage. Preventing wildfires will be helpful in protection of natural as well as the human habitat. It helps in addressing a wider spectrum of problems, such as situational awareness and real-time threat assessment using diverse streams of data.

INTRODUCTION

Fires play an integral role in human lives, but if uncontrolled, can be disastrous. Burnable materials catch fire easily and spread rapidly degrading the environment. The first stage of fire is called as ‘Surface Fire’ and the latter stage is known as ‘Crown Fire’. Crown fires are uncontrollable and damage the landscape. Although some safety measures have been employed, the accidents related to fire are ineluctable. There are different systems that are used for the detection of domestic and forest fires. Various alarm systems are being used today for fire detection and warning purpose [1]. In this project, we focus on employing various machine learning techniques on a system based on wireless sensor networks. There are a number of advantages of using machine learning algorithms with WSNs. If we can successfully predict the onset of the fire, a lot of damage will be reduced and environmental degradation will be decreased. Many forest areas do not have fire alarm systems installed. Fire alarms are important because they can alert you before a tragedy happens. You can, therefore, stay prepared, take necessary actions and reduce any kind of loss that might occur. Our goal is to create a technique based on sensors which will help in detecting the forest fires in the early stages. As soon as the fire is detected an alarm will be generated thereby minimizing the loss of environment, property or human life. The machine learning techniques integrated with the sensors help in detection of fire without any human help, therefore no patrolling is required. The major advantage of sensors is that they are fast and accurate. Moreover, machine learning maximizes resource utilization and improves the performance of sensor networks. We aim to evaluate the historical data and the natural events, predict the upcoming events based on acquired knowledge. Thus, the model will be capable of generating automatic warning signals whenever a dangerous situation arises, i.e., when fire or smoke is detected. Since this project is based on experimentation, it is constrained by many parameters. Primarily within the stipulated time, the correct response needs to be generated and provided to the user. One of the major issues is noise. Since we are depending on wireless sensors for our data, it is possible that this data might not be clean and may contain noise. Proper and quick preprocessing is required for optimal results. Another issue is accuracy. There is a huge possibility that a false positive response will be generated and a fake warning may be issued. By parameter optimization, we can reduce these falsepositives but not necessarily eliminate them. Also, the available computing power constraints the working of this system. The training and testing of models are compute-intensive tasks.

Overview

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices using sensors to monitor physical or environmental conditions [2]. A WSN system incorporates a gateway that provides wireless connectivity back to the wired world and distributed nodes. Energy, memory, computation, and bandwidth are the main constraints of WSN. They are widely used in fields like Air Pollution Monitoring, Forest Fire Detection, Landslide Detection, Water Quality Monitoring etc.

Machine learning can be used to teach the computers to act like human beings without being hardcoded [3]. Supervised, unsupervised, and reinforcement learning. Form the three categories of machine learning. In supervised learning, class label is present for a certain data (training data) and needs to be predicted for unknown instances [4]. In unsupervised learning, there is no class label present and implicit relationships within given data need to be discovered. When learning involves some kind of feedback mechanism for each step then it is known as Reinforcement Learning. In this mechanism, there is no precise label or error message. In our project, we focus on the supervised learning techniques for fire detection based on historical data. Some supervised learning techniques are Decision Tree, Naïve Bayes, Support Vector Machine, Logistic Regression etc.

PROPOSED SYSTEM

This project requires training to be done before deploying. Using historical data, various machine learning techniques are applied for Model learning and validation. Accordingly, the model classifies the real-time data, predicting the chances of fire.

The Modules involved in Fire Detection System are as follows:

• Pre-processing module: The data acquired from sensors is sent to the pre-processing module. It performs thresholding, cleaning, transformation and any specific enhancements required for later employed algorithms.

• Classification module: Pre-processed data is then classified using machine learning algorithms using the classification module. According to the result, alerting and alarming is done to the respective authorities.

• User Interface module: A user interface for monitoring and supervision purposes is provided. It shows real-time statistics and reports. The Working phases are as follows:

• Learning phase: Using the historical data first we train the model using various machine learning algorithms. This involves training and validation and application of accuracy improvement techniques like bagging and boosting. Once the model is trained, it is exported and can be used for deployment purposes.

• Testing phase: The system is exposed to the real-time data acquired either through cloud server or a local network. First, preprocessing is done in order to make the data suitable for algorithms to process. Cleaning is done to eliminate the noisy data followed by transformations and enhancements. Then the data is subjected to machine learning algorithms which predict the chances of onset of fire. According to the prediction results, the concerned authorities are alerted and respective mitigation measures can be taken to prevent or limit the damage.

ARCHITECTURE DIAGRAM

• Heterogeneous WSN: A heterogeneous Wireless Sensor Networks consist of a network of different sensor linked to a base station. For fire detection we measure the environmental parameters using the sensors like temperature sensor, humidity sensor, Carbon Monoxide/ Carbon Dioxide sensors. Along with the sensor data, a visual feed is also sent to the base stations at timed instances.

• Data Acquisition: Data from multiple sensors is aggregated and some minor processing is done. We check the parameters for threshold values. If any of value exceeds the threshold then data is forwarded for classification

• Pre-processing and Feature Extraction: The data either stored at base station or cloud server is fetched and pre-processing is performed. This involves cleaning of data for removal of inaccurate/corrupt data. Along with cleaning we perform transformation and enhancements to make the data suitable and ready for classification algorithms. • Classification Model: This is the heart of the system. First, using labelled data we undergo a model training phase via supervised learning strategy. Then data validation is used to check for the model accuracy and feasibility. Once the model has suitable accuracy, we can start the testing of model using unlabeled data. Multivariate data is fed as input and target class is predicted along with its probability. • Alarming and Reporting: According to the predicted class, the system generates an alert and sends to the designated authorities along with information about severity of the situation.

APPLICATIONS

Fire detection systems can be installed at various locations susceptible to hazards. Fire prone areas need to have an efficient and fast mechanism to detect and alarm the required personnel. If the response time is minimized, then mitigation measures can prevent an onset of wide range disruption, reducing the damage and cost on rehabilitation. The system is useful for following:

• Security Department
• Forest Safety Workers
• Organizations, Institutions, Hospitals, etc.
• Industrial plants

CONCLUSION

Adoption of machine learning improves the usability of Wireless Sensor Networks (WSN) in environmental analysis and monitoring. Efficiency and accuracy along with cost factor serve as key factors for the building of fire detection systems. In this project we understand the working of various machine learning techniques along with their advantages and limitations. According to evaluation factors appropriate choice can be made regarding the technique to be implemented. Also, a hybrid technique involving combination of algorithms can be suggested. This paper gives the idea about how combination of sensor data and prediction algorithms can be utilized to detect onset of fire and limit the damage caused to environment.

2.2. References

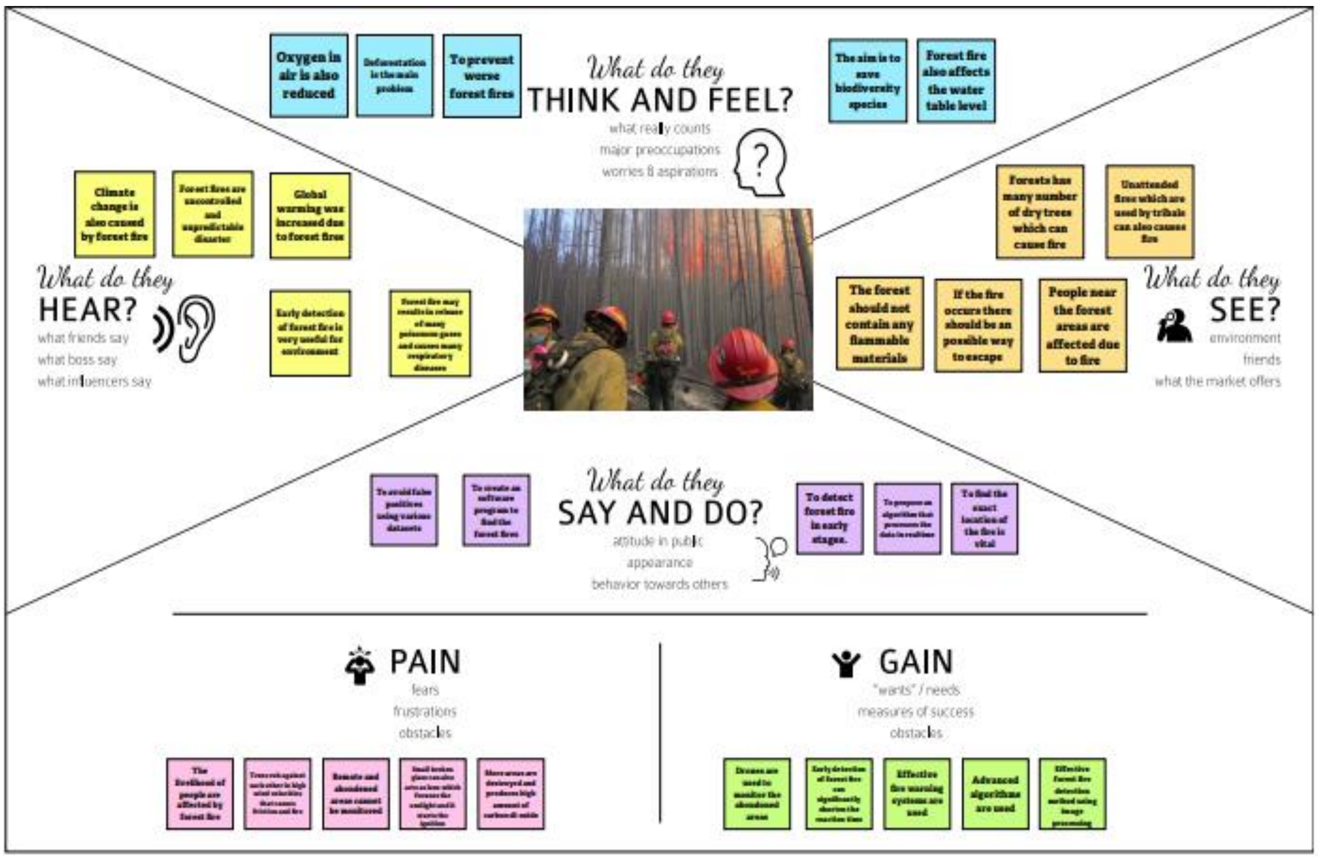
- Ryan Fireprotection Inc. Webpage, http://www.ryanfp.com/fire-alarm-history/, accessed on July 29th, 2017.
- Wikipedia Webpage, https://en.wikipedia.org/wiki/Wireless_sensor_networn, accessed on October 14th, 2017.
- Coursera Webpage, https://www.coursera.org/learn/machine-learning, accessed on October 14th, 2017
- Kdnuggets Webpage, https://www.kdnuggets.com/2016/08/10-algorithms-machine-learningengineers.html, accessed on October 13th, 2017

2.3. Problem statement definition

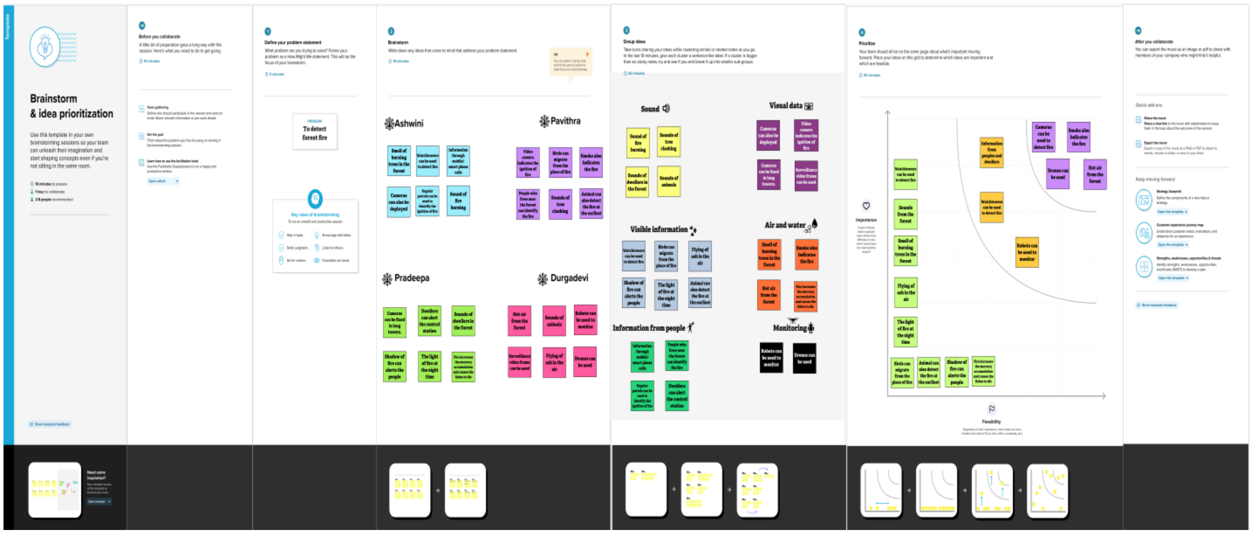
- Rakesh is a/an Ecologist Who needs To find forest fire at the earliest Because He wants to save the forest.
- Ria is a/an Researcher Who needs To protect biodiversity species Because She needs to research about the species.
- Anu is a/an Social worker Who needs To protect the tribes from forest fire Because To improve their livelihood.
- Maaran is a/an Forest officer Who needs To monitor the abandoned areas Because Those areas are more dens and it is out of reach of humans.

3. Ideation & Proposed Solution

3.1. Empathy Map Canvas



3.2. Ideation and Brainstorming

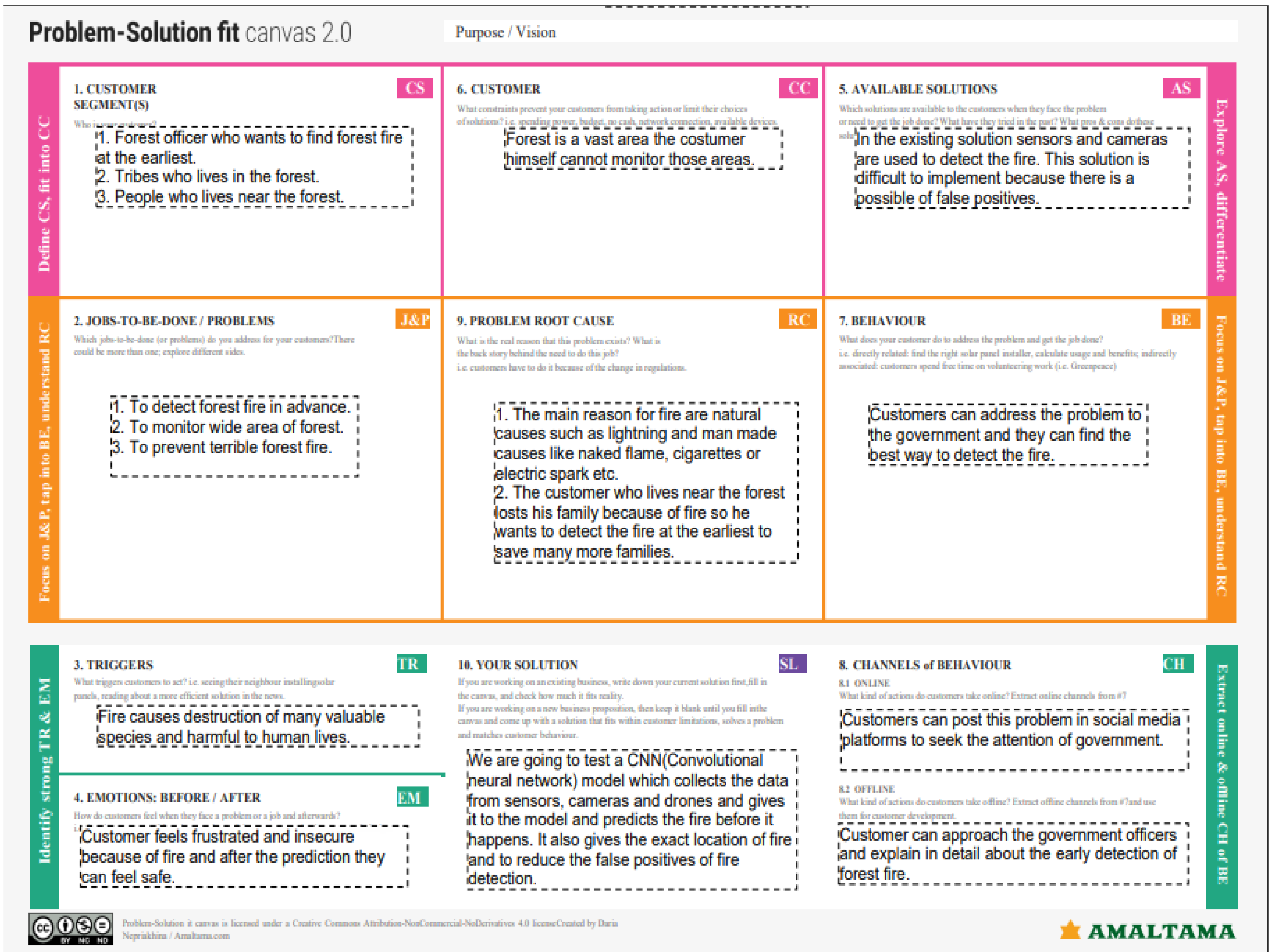


3.3. Proposed solution

S.no	Parameter	Description
1.	Problem Statement (Problem to be solved)	➢ I am a forest officer I'm trying to find forest fire but it's an difficult task to find it at the earliest because forest is a vast area, it is unable to put off the fire easily which makes me feel frustrated. ➢ I am an Tribal and my community is trying to find food and shelter because of fire, it will destruct everything which makes me feel unhappy.
2.	Idea / Solution description	➢ Abandoned areas can be monitored using drones. ➢ Cameras can be installed to capture the thermal images of fire. ➢ Heat detectors and smoke sensors can also be used to detect fire. ➢ GPS can be used to track the location of fire. ➢ Above these parameters are given as an data for convolutional neural network which can predict the forest fire. ➢ Fire alarm is used to notify the fire.
3.	Novelty / Uniqueness	The use of convolutional neural network can be able to process the image and also test the dataset also. So it is easy to predict the forest fire at the earliest and the location

		of fire is identified using GPS.
4.	Social Impact / Customer Satisfaction	➢ Can reduce the extinction of valuable animal species. ➢ Increased oxygen in air. ➢ Evacuations can be done before the fire got worse. ➢ Humans can use forest for their basic needs. ➢ Most essential trees can be saved. ➢ Tribals can live peacefully.
5.	Business Model (Revenue Model)	➢ We can make the forest as an tourist spot and can make revenue. ➢ We can collect various nuts and fruits from the forest and we can sell it. ➢ We can make many wooden products from the forest and sell it.
6.	Scalability of the Solution	The trained model is capable of adapting according to the datasets and the environmental situations.

3.4. Problem solution fit



4. Requirement analysis

4.1. Functional requirements

FR no.	Functional requirement (Epic)	Sub requirement(Story/Sub-Task)
FR-1	User collects the real time data.	• The user collects the real time data to identify the exact weather conditions.
FR-2	Cameras fixed in the forest.	• The captured data are collected from the cameras.

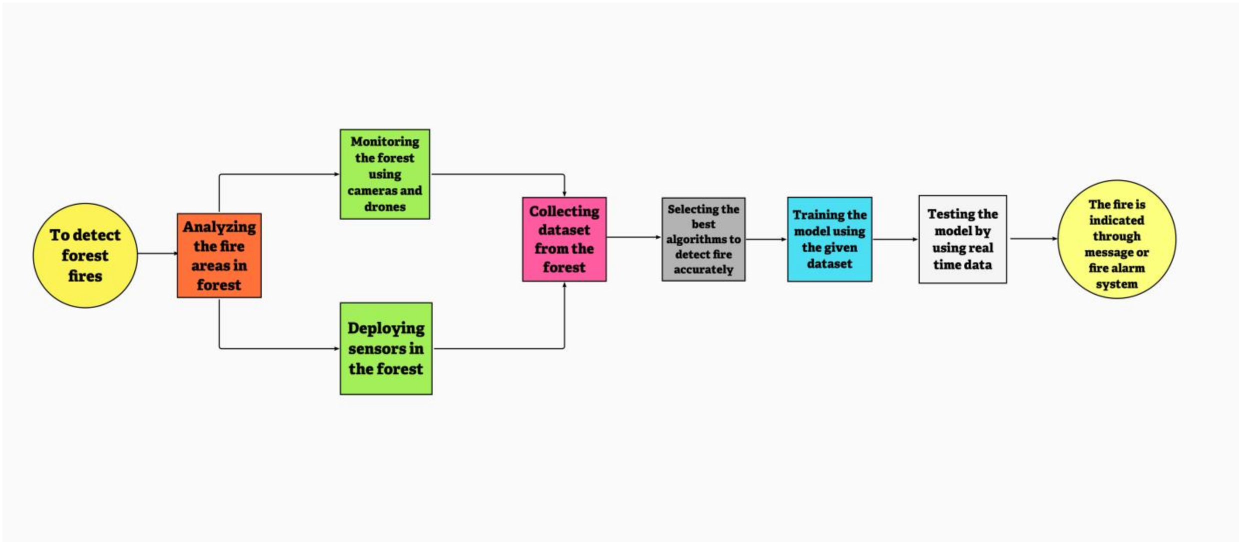
FR-3	Store the data.	• The data are stored in the cloud
FR-4	Fire Monitoring.	• The forest is continuously monitoring through the camera and drones.
FR-5	Fire detection.	• The fire is detected using CNN (Convolutional Neural Network)model.
FR-6	Notification.	• Once the fire is detected it is notified through the message and fire alarm system.

4.2. **Non-Functional requirements**

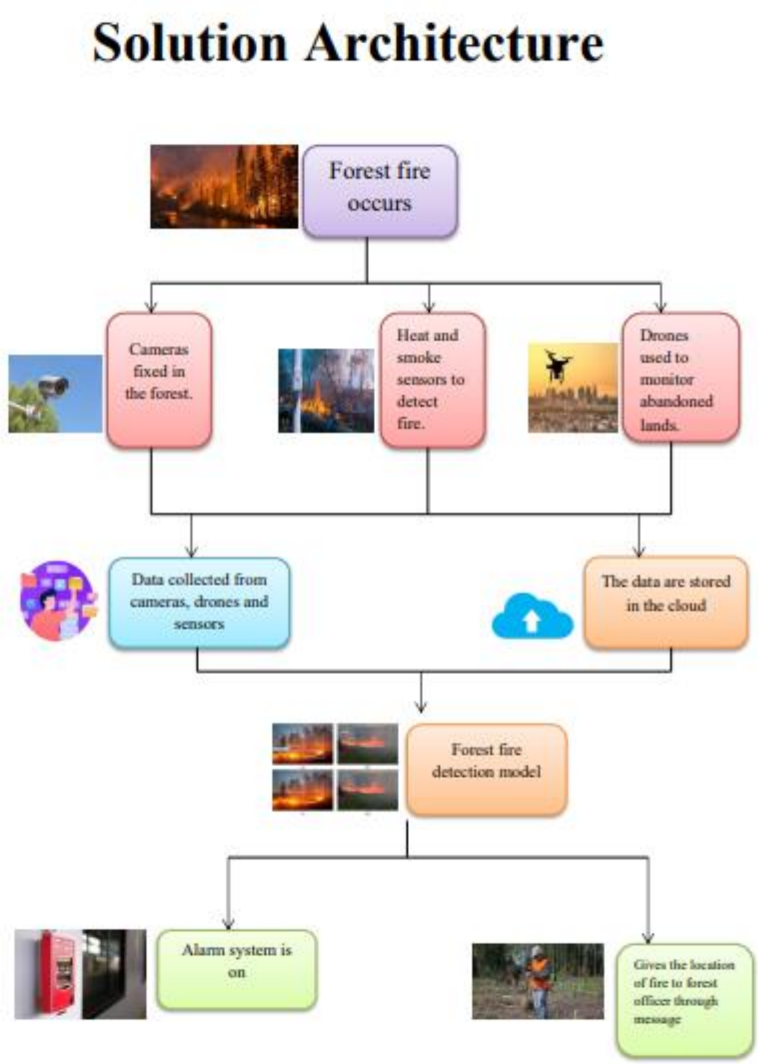
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	Usability	• Most essential trees can be saved. • Many valuable extinction species can be saved.
NFR-2	Security	• It is used to secure environment.
NFR-3	Reliability	• The model is more accurate to find the fire at the earliest.
NFR-4	Performance	• In the model, the alert message is an immediate action without any lag.
NFR-5	Availability	• The model is available at 24/7.
NFR-6	Scalability	• The trained model is capable of adapting according to the dataset and the environment situation.

5. Project design

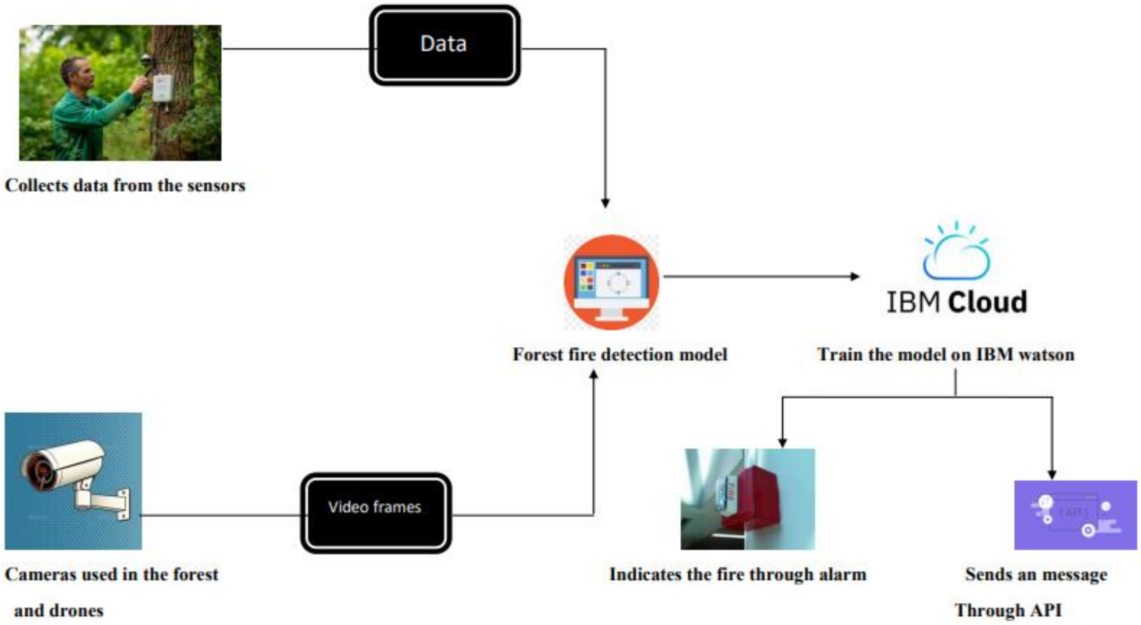
5.1. Data flow diagrams



5.2. Solution and technical architecture



Technical Architecture:



5.3. User stories

User type	Functional requirement(epic)	User story number	User story/talk	Acceptance criteria	Priority	Release
Researcher	Analyzing the forest fire	USN-1	The researcher who wants to save valuable species in the forest takes necessary actions.	He can address the problem to the government.	High	Sprint-1
Forest officer	Preventing the worst forest fires	USN-2	The forest officer is worried about the worst forest fires because it is unable to put off.	He can tell the fire department immediately to put off the fire	High	Sprint-1
Environmentalist	To detect the forest fire	USN-3	The environmentalist can collect various data from the forest to detect fire at the earliest	The data collected by him should be accurate	Medium	Sprint-2
Government	Can deploy various sensors and camera	USN-4	The government can take necessary steps by deploying heat detectors, smoke sensors and cameras in the forest	The sensors and cameras can collect real time data	High	Sprint-2
Programmer	Testing and training the forest fire detection model	USN-5	The programmer can build an forest fire detection model by training the dataset	The model can give high accuracy	High	Sprint-3
Forest officer	Notification	USN-6	The fire detected by the model can be notified by using message and fire alarm system	It notifies the forest to the forest department	High	Sprint-3

6. Project Planning and Scheduling

6.1. Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Analyzing the forest fire.	USN-1	The researcher who wants to save valuable species in the forest takes necessary actions.	2	High	Ashwini. R, Durgadevi. M
Sprint-1	Preventing the worst forest fires	USN-2	The forest officer is worried about the worst forest fires because it is unable to put off.	2	High	Pavithra. G, Pradeepa. H
Sprint-2	To detect the forest fire	USN-3	The environmentalist can collect various data from the forest to detect fire at the earliest	2	Medium	Ashwini. R, Pradeepa. H
Sprint-2	Can deploy various sensors and camera	USN-4	The government can take necessary steps by deploying heat detectors, smoke sensors and cameras in the forest	2	High	Pavithra. G, Durgadevi. M
Sprint-3	Testing and training the forest fire detection model	USN-5	The programmer can build an forest fire detection model by training the dataset	2	High	Ashwini. R, Pavithra. G
Sprint-4	Notification	USN-6	The fire detected by the model can be notified by using message and fire alarm system	2	High	Pradeepa. H, Durgadevi. M

6.1. 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	29 Oct 2022
Sprint-2	20	6 days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.2. Reports from Jira

7. Coding and solutioning

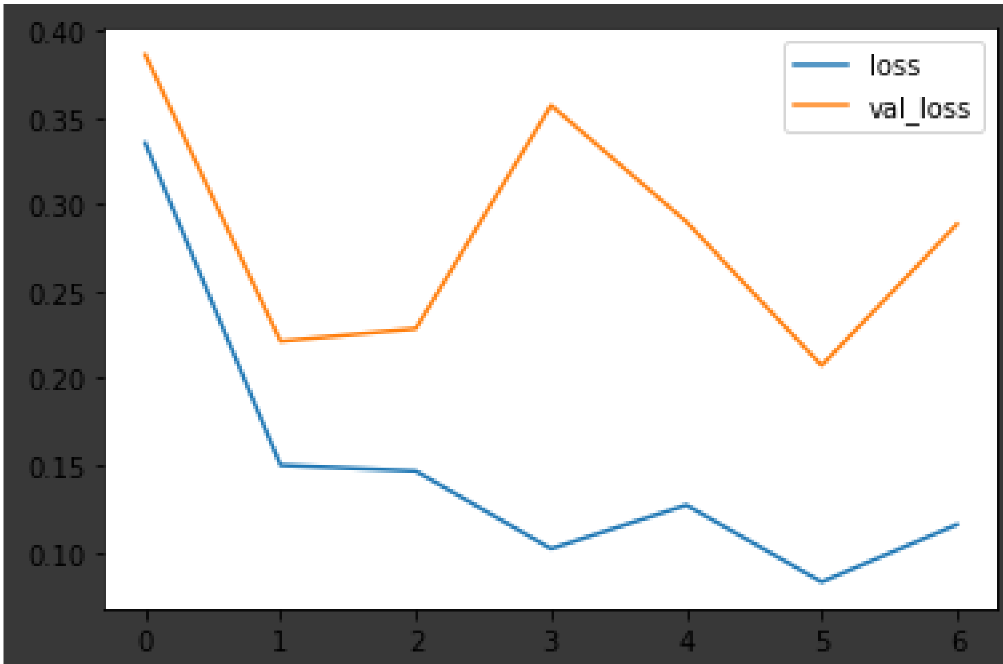
7.1. Feature 1

Image processing

```
import tensorflow as tf
import numpy as np
```

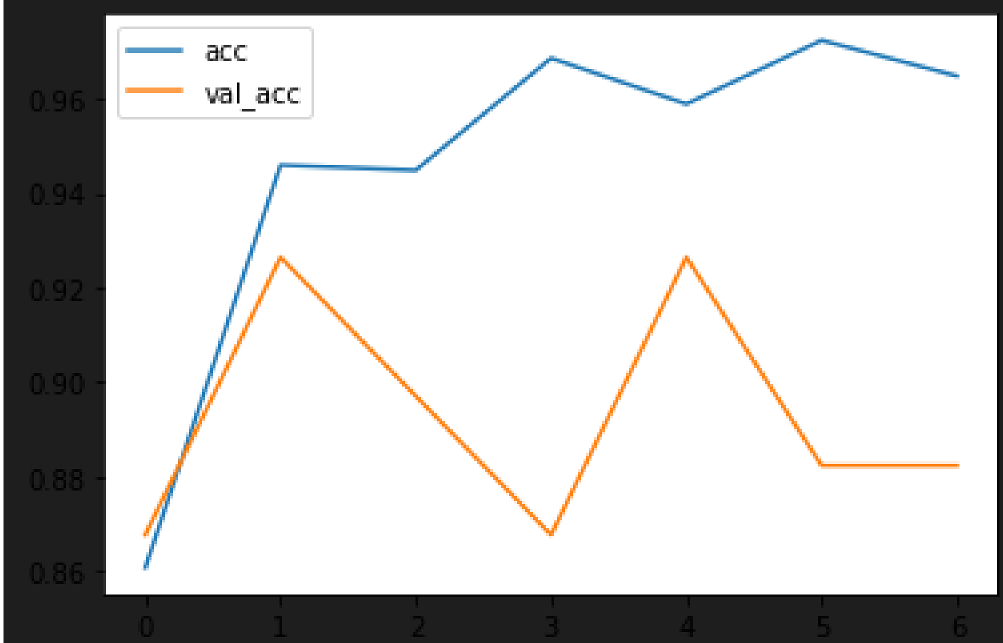
```
from tensorflow import keras
import os
import cv2
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
pwd
/content
Ls
sample_data/
from google.colab import drive
drive.mount ('/content/drive')
Mounted at /content/drive
train=ImageDataGenerator(rescale=1/255)
test=ImageDataGenerator(rescale=1/255)
train_dataset=train.flow_from_directory("C",
                                        target_size=(150,150),
                                        batch_size=32,
                                        class_mode='binary')
test_dataset=test.flow_from_directory("/content/drive/MyDrive/archive (1)/
forest_fire/Testing",
                                        target_size=(150,150),
                                        batch_size=32,
                                        class_mode='binary')
Found 1852 images belonging to 2 classes.
Found 68 images belonging to 2 classes.
test_dataset.class_indices
{'fire': 0, 'no fire': 1}
model=keras.Sequential()
model.add(keras.layers.Conv2D(32,(3,3),activation='relu',input_shape=(150,
150,3)))
model.add(keras.layers.MaxPool2D(2,2))
model.add(keras.layers.Conv2D(64,(3,3),activation='relu'))
model.add(keras.layers.MaxPool2D(2,2))
model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))
model.add(keras.layers.MaxPool2D(2,2))
model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))
model.add(keras.layers.MaxPool2D(2,2))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(512,activation='relu'))
model.add(keras.layers.Dense(1,activation='sigmoid'))
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accura
cy'])
r=model.fit(train_dataset,epochs=7,validation_data=test_dataset)
Epoch 1/7
```

```
58/58 [==============================] - 257s 4s/step - loss: 0.3354 -
accuracy: 0.8607 - val_loss: 0.3862 - val_accuracy: 0.8676
Epoch 2/7
58/58 [==============================] - 84s 1s/step - loss: 0.1499 -
accuracy: 0.9460 - val_loss: 0.2213 - val_accuracy: 0.9265
Epoch 3/7
58/58 [==============================] - 79s 1s/step - loss: 0.1465 -
accuracy: 0.9449 - val_loss: 0.2286 - val_accuracy: 0.8971
Epoch 4/7
58/58 [==============================] - 79s 1s/step - loss: 0.1017 -
accuracy: 0.9687 - val_loss: 0.3571 - val_accuracy: 0.8676
Epoch 5/7
58/58 [==============================] - 83s 1s/step - loss: 0.1269 -
accuracy: 0.9590 - val_loss: 0.2900 - val_accuracy: 0.9265
Epoch 6/7
58/58 [==============================] - 80s 1s/step - loss: 0.0826 -
accuracy: 0.9725 - val_loss: 0.2073 - val_accuracy: 0.8824
Epoch 7/7
58/58 [==============================] - 79s 1s/step - loss: 0.1157 -
accuracy: 0.9649 - val_loss: 0.2886 - val_accuracy: 0.8824
predictions=model.predict(test_dataset)
predictions=np.round(predictions)
3/3 [==============================] - 1s 210ms/step
Predictions
array([[1.], [1.], [1.], [1.], [1.], [1.], [1.], [1.], [0.], [0.], [1.],
[1.], [1.], [1.], [1.], [0.], [1.], [0.], [0.], [0.], [1.], [1.], [0.],
[1.], [1.], [0.], [1.], [0.], [1.], [1.], [1.], [0.], [1.], [1.], [1.],
[1.], [1.], [0.], [1.], [0.], [0.], [1.], [1.], [1.], [1.], [1.], [0.],
[0.], [0.], [1.], [1.], [1.], [1.], [0.], [1.], [1.], [1.], [1.], [0.],
[1.], [1.], [1.], [1.], [1.], [1.], [0.], [0.], [1.]], dtype=float32)
print(len(predictions))
68
import matplotlib.pyplot as plt
plt.plot(r.history['loss'],label='loss')
plt.plot(r.history['val_loss'],label='val_loss')
plt.legend()
<matplotlib.legend.Legend at 0x7f2b40e6c410>
```




```
plt.plot(r.history['accuracy'],label='acc')
plt.plot(r.history['val_accuracy'],label='val_acc')
plt.legend()
```

<matplotlib.legend.Legend at 0x7f2b40d5d3d0>




```
def predictImage(filename):
  img1=image.load_img(filename,target_size=(150,150))
  plt.imshow(img1)
  Y=image.img_to_array(img1)
```

```
  X=np.expand_dims(Y,axis=0)
  val=model.predict(X)
  print(val)
  if val==1:
    plt.xlabel("No fire",fontsize=30)
  elif val==0:
    plt.xlabel("fire",fontsize=30)
predictImage('/content/drive/MyDrive/archive (1)/forest_fire/Testing/fire/abc169.jpg')
```

```
1/1 [==============================] - 0s 96ms/step
```


```
[[0.]]
```

```
predictImage('/content/drive/MyDrive/archive (1)/forest_fire/Testing/no fire/abc339.jpg')
```

```
1/1 [==============================] - 0s 29ms/step
[[1.]]
```

7.2. Feature 2

Video analysis

```
model.save('forestfire.h5')
from keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import cv2
model = load_model("forestfire.h5")
img=image.load_img('/content/drive/MyDrive/archive (1)/forest_fire/Testing
/fire/abc169.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)
predictions=model.predict(test_dataset)
3/3 [==============================] - 1s 193ms/step
predictions
array([[1.3216068e-01], [9.8854232e-01], [9.1673441e-02], [1.1199379e-01],
[9.9491620e-01], [9.9985594e-01], [1.9926480e-03], [9.9912262e-01],
```

```
[2.7595463e-01], [9.9026197e-01], [4.2083380e-03], [1.5435454e-02],
[9.1657960e-01], [9.9855191e-01], [9.3416864e-01], [9.9999684e-01],
[9.9683201e-01], [9.9937695e-01], [9.9972421e-01], [9.9925059e-01],
[1.0848244e-02], [9.9993914e-01], [9.9994773e-01], [9.9817902e-01],
[1.1336886e-08], [9.9942589e-01], [9.9999529e-01], [2.2020526e-03],
[9.9775440e-01], [9.9980998e-01], [6.4162660e-01], [1.3592747e-08],
[6.0574126e-05], [4.2828145e-03], [9.9985301e-01], [9.9994171e-01],
[9.9935454e-01], [9.0613592e-01], [2.8121749e-01], [1.3694618e-05],
[9.9981529e-01], [2.3207234e-03], [9.9960369e-01], [9.9996364e-01],
[1.9189017e-02], [6.1245866e-02], [7.3057318e-01], [9.9997854e-01],
[9.0425205e-01], [9.9686402e-01], [9.9982566e-01], [9.9906653e-01],
[3.6314325e-04], [9.9985486e-01], [9.9988699e-01], [9.7590846e-01],
[9.9999541e-01], [9.9998963e-01], [7.7658974e-02], [9.9729377e-01],
[9.0998524e-01], [3.4965103e-06], [9.9934405e-01], [9.9184835e-01],
[6.5742603e-07], [9.9540293e-01], [9.5957720e-01], [9.9929482e-01]],
dtype=float32)
pip install twilio
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting twilio
  Downloading twilio-7.15.2-py2.py3-none-any.whl (1.4 MB)
     |                                | 1.4 MB 11.1 MB/s
Collecting PyJWT<3.0.0,>=2.0.0
  Downloading PyJWT-2.6.0-py3-none-any.whl (20 kB)
Requirement already satisfied: requests>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from twilio) (2.23.0)
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-
packages (from twilio) (2022.6)
Requirement already satisfied: idna<3,>=2.5 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio)
(2.10)
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)
(1.24.3)
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: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio)
(3.0.4)
Installing collected packages: PyJWT, twilio
Successfully installed PyJWT-2.6.0 twilio-7.15.2
pip install playsound
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting playsound
  Downloading playsound-1.3.0.tar.gz (7.7 kB)
Building wheels for collected packages: playsound
  Building wheel for playsound (setup.py) ... done
```

```
  Created wheel for playsound: filename=playsound-1.3.0-py3-none-any.whl
size=7035
sha256=d54aaea42aa656851b0efbbf8d86d46af49f3a6bea05bd2ea71d630e06231584
  Stored in directory:
/root/.cache/pip/wheels/ba/f8/bb/ea57c0146b664dca3a0ada4199b0ecb5f9dfcb7b7
e22b65ba2
Successfully built playsound
Installing collected packages: playsound
Successfully installed playsound-1.3.0
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)
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
from matplotlib import pyplot as plt
import librosa
from tensorflow.keras.preprocessing import image
from keras.models import load_model
cap = cv2.VideoCapture('/content/video.mp4')
if (cap.isOpened()== False):
  print("Error opening video stream or file")
while(cap.isOpened()):
ret, frame = cap.read()
  if ret == True:
    x=image.img_to_array(frame)
    res=cv2.resize(x,dsize=(128,128),interpolation=cv2.INTER_CUBIC)
 x=np.expand_dims(res,axis=0)
    model=load_model("/content/forestfire.h5")
    cv2_imshow(frame)
    predictions=model.predict(test_dataset)
    predictions = int(predictions[0][0])
    predictions
    int(predictions)
    if predictions==0:
      print('Forest fire')
      break
    else:
      print("danger")
      break
cap.release()
cv2.destroyAllWindows()
from twilio.rest import Client
from playsound import playsound
```

```
if predictions==0:
  account_sid='AC9496860c13d1e2959a984c6744e6e513'
  auth_token='c5d99441754343492a6d9046e614c4cb'
  client=Client(account_sid,auth_token)
  message=client.messages \
  .create(
      body='forest fire is detected,stay alert',
 from_='+12183046916',
 to='+918680875090')
  print(message.sid)
  print("Fire detected")
  print("SMS Sent!")
```

```
WARNING:tensorflow:5 out of the last 13 calls to <function
Model.make_predict_function.<locals>.predict_function at 0x7f3c31194200>
triggered tf.function retracing. Tracing is expensive and the excessive
number of tracings could be due to (1) creating @tf.function repeatedly in a
loop, (2) passing tensors with different shapes, (3) passing Python objects
instead of tensors. For (1), please define your @tf.function outside of the
loop. For (2), @tf.function has reduce_retracing=True option that can avoid
unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
3/3 [==============================] - 1s 210ms/step
Forest fire
SMf9f67ab498dabbd55a82b106c34ca06c
Fire detected
SMS Sent!
```

7.3. Database schema

Image Link: https://www.kaggle.com/datasets/brsdincer/wildfire-detection-image-data

Video link: https://www.storyblocks.com/video/stock/large-flames-and-smoke-of-a-forest-fire-fire-01mov-2olzxlj

8. Testing

8.1. Test cases

Date	15.11.2022
Team ID	PNT2022TMIDxxxxxx
Project Name	Emerging methods for early dete
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID
LoginPage_TC_OO1	Functional	Home Page	Collect image dataset in the forest.	By using the camera and sensors.	Gathering the image either with fire or without fire.	https://www.kaggle.com/datasets/brsdincer/wildfire-de	Dataset collected.	Dataset collected.	Pass	-	Y	
LoginPage_TC_OO2	UI	Home Page	Training the forest fire detection model.	By using the CNN image preprocessing training code.	Program were written and executed.	https://colab.research.google.com/drive/17hJlwoy0n1X	Written the program and executed it.	Working as expected.	Pass	-		BUG-1234
LoginPage_TC_OO3	Functional	Home page	Testing the forest fire detection model.	By using the CNN image preprocessing testing code.	Program were written and executed.	https://colab.research.google.com/drive/17hJlwoy0n1X	Written the program and executed it.	Working as expected.	Pass	-		
LoginPage_TC_OO4	Functional	Login page	Collect the video using CCTV.	By using the camera.	Gathering the video with fire.	https://www.storyblocks.com/video/stock/large-flames	Fire video is collected.	Working as expected.	Pass	-		
LoginPage_TC_OO5	Functional	Login page	Detecting video frame and sending SMS.	By using the CNN video frame coding and twilio.	Program were written and executed.	https://colab.research.google.com/drive/1h6ToFYjnEcgl	Written the program for video frame and executed it.	Working as expected.	Pass	-		

8.2. User acceptance testing

Defect analysis

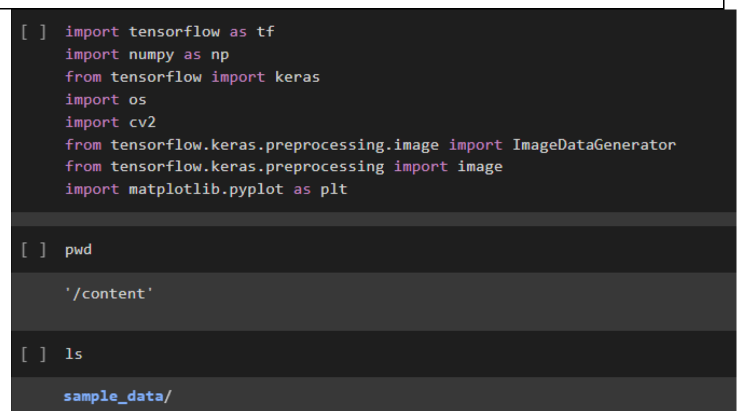
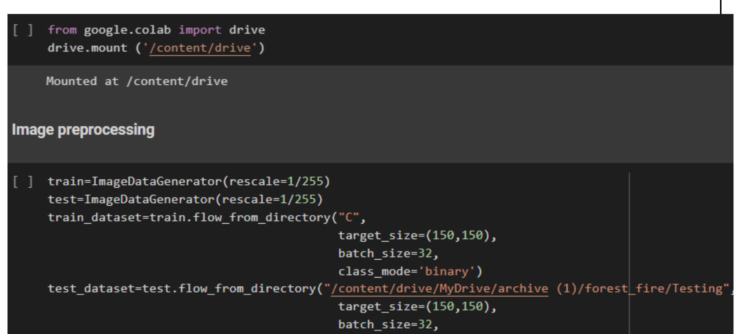
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	3	2	3	2	11
Duplicate	0	0	0	0	0
External	2	1	2	3	8
Fixed	1	2	3	2	8
Not Reproduced	0	0	6	0	6
Skipped	0	0	3	2	5
Won't Fix	0	3	2	3	8
Totals	6	8	19	12	46

Test case analysis

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9. Results

9.1. Performance metrics

S. No.	Parameter	Values	Screenshot
1.	Model Summary	-	 

```
Found 1852 images belonging to 2 classes.
Found 68 images belonging to 2 classes.
```

```
[ ] test_dataset.class_indices
```

```
{'fire': 0, 'no fire': 1}
```

Model building

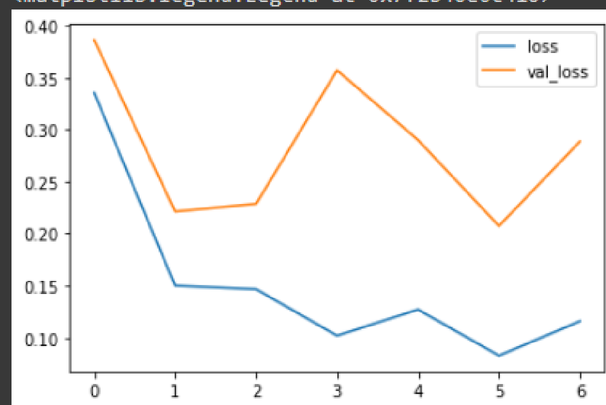
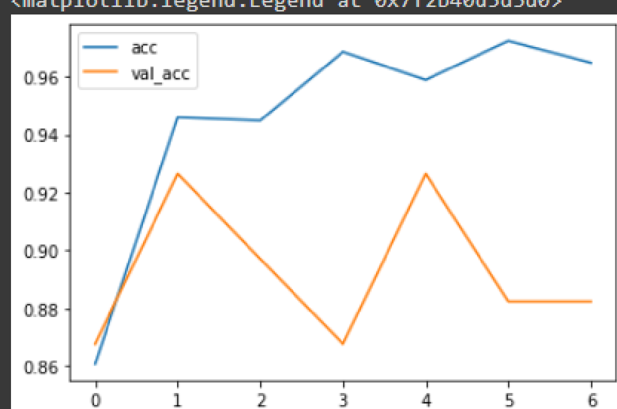
```
[ ] model=keras.Sequential()
    model.add(keras.layers.Conv2D(32,(3,3),activation='relu',input_shape=(150,150,3)))
    model.add(keras.layers.MaxPool2D(2,2))
    model.add(keras.layers.Conv2D(64,(3,3),activation='relu'))
    model.add(keras.layers.MaxPool2D(2,2))
    model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))
    model.add(keras.layers.MaxPool2D(2,2))
    model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))
    model.add(keras.layers.MaxPool2D(2,2))
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(512,activation='relu'))
```

```
[ ] model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
[ ] r=model.fit(train_dataset,epochs=7,validation_data=test_dataset)

    Epoch 1/7
    58/58 [==============================] - 257s 4s/step - loss: 0.3354 - accuracy: 0.8607 - val_loss: 0.3862 - val_accuracy: 0.8676
    Epoch 2/7
    58/58 [==============================] - 84s 1s/step - loss: 0.1499 - accuracy: 0.9460 - val_loss: 0.2213 - val_accuracy: 0.9265
    Epoch 3/7
    58/58 [==============================] - 79s 1s/step - loss: 0.1465 - accuracy: 0.9449 - val_loss: 0.2286 - val_accuracy: 0.8971
    Epoch 4/7
    58/58 [==============================] - 79s 1s/step - loss: 0.1017 - accuracy: 0.9687 - val_loss: 0.3571 - val_accuracy: 0.8676
    Epoch 5/7
    58/58 [==============================] - 83s 1s/step - loss: 0.1269 - accuracy: 0.9590 - val_loss: 0.2900 - val_accuracy: 0.9265
    Epoch 6/7
    58/58 [==============================] - 80s 1s/step - loss: 0.0826 - accuracy: 0.9725 - val_loss: 0.2073 - val_accuracy: 0.8824
    Epoch 7/7
    58/58 [==============================] - 79s 1s/step - loss: 0.1157 - accuracy: 0.9649 - val_loss: 0.2886 - val_accuracy: 0.8824
```

```
[ ] predictions=model.predict(test_dataset)
    predictions=np.round(predictions)

    3/3 [==============================] - 1s 210ms/step
```

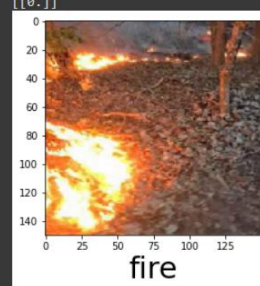
```
[ ] predictions

    array([[1.],
           [1.],
           [1.],
           [1.],
           [1.],
           [1.],
           [1.],
           [1.],
           [0.],
           [0.],
           [1.],
           [1.],
           [1.],
           [1.],
           [1.],
```

			[ ] [1.], [0.], [0.], [1.]], dtype=float32) [ ] print(len(predictions)) 68 import matplotlib.pyplot as plt plt.plot(r.history['loss'],label='loss') plt.plot(r.history['val_loss'],label='val_loss') plt.legend() <matplotlib.legend.Legend at 0x7f2b40e6c410>  plt.plot(r.history['accuracy'],label='acc') plt.plot(r.history['val_accuracy'],label='val_acc') plt.legend() <matplotlib.legend.Legend at 0x7f2b40d5d3d0> 

Testing the model

```
[ ]  def predictImage(filename):
       img1=image.load_img(filename,target_size=(150,150))
       plt.imshow(img1)
       Y=image.img_to_array(img1)
       X=np.expand_dims(Y,axis=0)
       val=model.predict(X)
       print(val)
       if val==1:
         plt.xlabel("No fire",fontsize=30)
       elif val==0:
         plt.xlabel("fire",fontsize=30)
```

```
[ ]  predictImage('/content/drive/MyDrive/archive (1)/forest_fire/Testing/fire/abc169.jpg')

     1/1 [==============================] - 0s 96ms/step
     [[0.]]
```




```
[ ]  predictImage('/content/drive/MyDrive/archive (1)/forest_fire/Testing/no fire/abc339.jpg')

     1/1 [==============================] - 0s 29ms/step
     [[1.]]
```

2.	Accuracy	Training Accuracy - 96.49 Validation Accuracy -88.24	Epoch 1/7 58/58 [==============================] - 257s 4s/step - loss: 0.3354 - accuracy: 0.8607 - val_loss: 0.3862 - val_accuracy: 0.8 Epoch 2/7 58/58 [==============================] - 84s 1s/step - loss: 0.1499 - accuracy: 0.9460 - val_loss: 0.2213 - val_accuracy: 0.92 Epoch 3/7 58/58 [==============================] - 79s 1s/step - loss: 0.1465 - accuracy: 0.9449 - val_loss: 0.2286 - val_accuracy: 0.89 Epoch 4/7 58/58 [==============================] - 79s 1s/step - loss: 0.1017 - accuracy: 0.9687 - val_loss: 0.3571 - val_accuracy: 0.86 Epoch 5/7 58/58 [==============================] - 83s 1s/step - loss: 0.1269 - accuracy: 0.9590 - val_loss: 0.2900 - val_accuracy: 0.92 Epoch 6/7 58/58 [==============================] - 80s 1s/step - loss: 0.0826 - accuracy: 0.9725 - val_loss: 0.2073 - val_accuracy: 0.88 Epoch 7/7 58/58 [==============================] - 79s 1s/step - loss: 0.1157 - accuracy: 0.9649 - val_loss: 0.2886 - val_accuracy: 0.88
3.	Confidence Score (Only Yolo Projects)	Class Detected - 2 classes Confidence Score - 80	Found 1852 images belonging to 2 classes. Found 68 images belonging to 2 classes.

10.Advantages

Early detection of forest fire gives,

- Timely information about the appearance of fire reduce the number of areas affected by this fire.
- It minimizes the cost of fire extinguishing.
- It reduces the damage caused in woods.
- It can save wildlife species.
- Many medicinal plants can be saved.

Disadvantages

- It will kill and displace wildlife.
- It alter water cycle and soil fertility.
- It is endangered to lives and livelihood of local communities.

11.Conclusion

We created the forest fire detection model with accuracy of 96.49% and validation accuracy of 88.24%. So, this model can be used to detect the forest fire and the fire detected message will be sent to the concerned mobile number.

12.Future scope

In future we can deploy an accurate detection model which can predict the fire and can reduce the false positives of the prediction.

13.Appendix

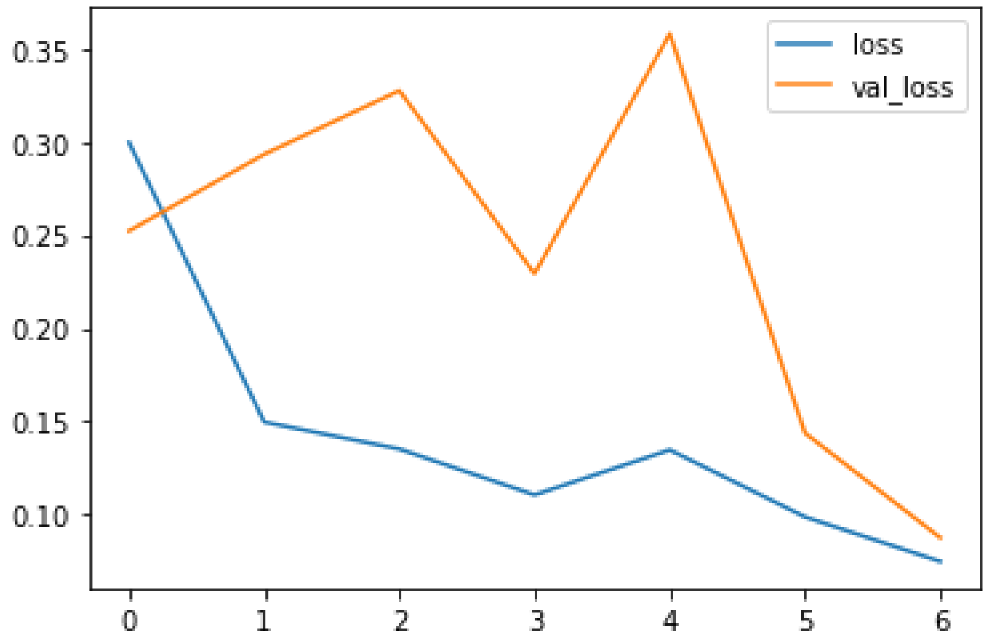
13.1. Source code

```
import tensorflow as tf
import numpy as np
from tensorflow import keras
import os
import cv2
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
pwd
'/home/wsuser/work'
 Ls
forest_fire/  forest-fire-detection-model.tgz  forestfire.h5  forestfire.tar.gb
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='OOhSbB9gBRTvqouyjMHAuqUdqJtEDaiGmKkEyJ9_mzMe',
    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-q7bq0vtwyl16o2'
object_key = 'dataset.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)
train=ImageDataGenerator(rescale=1/255)
test=ImageDataGenerator(rescale=1/255)
train_dataset=train.flow_from_directory("/home/wsuser/work/forest_fire/Training and Validation",
                                        target_size=(150,150),
                                        batch_size=32,
```

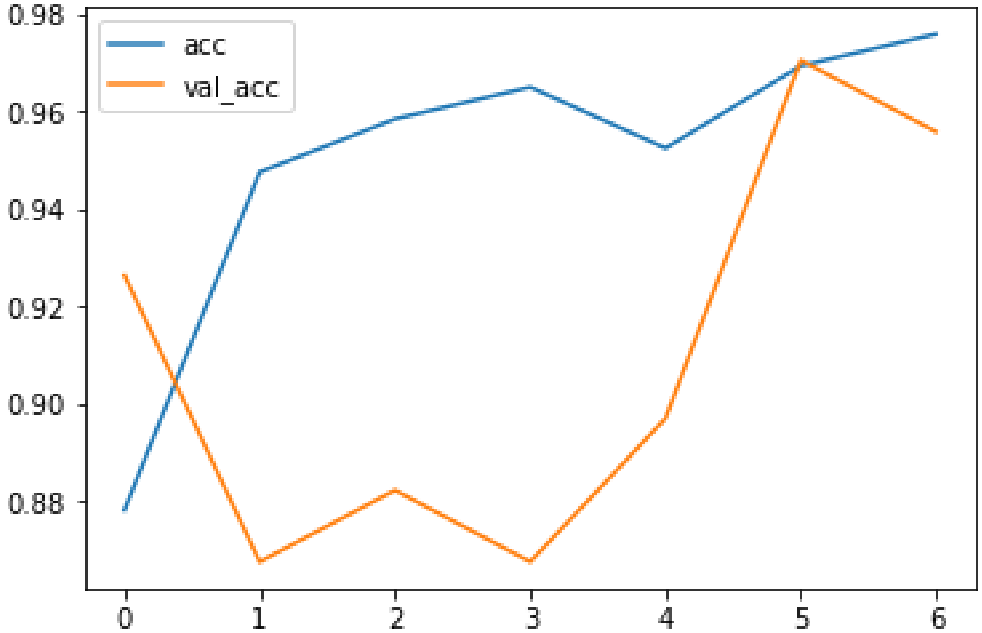
```
                                   class_mode='binary')
test_dataset=test.flow_from_directory("/home/wsuser/work/forest_fire/Testing",
                                   target_size=(150,150),
                                   batch_size=32,
                                   class_mode='binary')
Found 1832 images belonging to 2 classes.
Found 68 images belonging to 2 classes.
test_dataset.class_indices
{'fire': 0, 'nofire': 1}
model=keras.Sequential()
model.add(keras.layers.Conv2D(32,(3,3),activation='relu',input_shape=(150,150,3)))
model.add(keras.layers.MaxPool2D(2,2))
model.add(keras.layers.Conv2D(64,(3,3),activation='relu'))
model.add(keras.layers.MaxPool2D(2,2))
model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))
model.add(keras.layers.MaxPool2D(2,2))
model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))
model.add(keras.layers.MaxPool2D(2,2))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(512,activation='relu'))
model.add(keras.layers.Dense(1,activation='sigmoid'))
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
r=model.fit(train_dataset,epochs=7,validation_data=test_dataset)
Epoch 1/7
58/58 [==============================] - 66s 1s/step - loss: 0.2997 - accuracy: 0.8783 - val_loss: 0.2522 -
val_accuracy: 0.9265
Epoch 2/7
58/58 [==============================] - 65s 1s/step - loss: 0.1497 - accuracy: 0.9476 - val_loss: 0.2933 -
val_accuracy: 0.8676
Epoch 3/7
58/58 [==============================] - 64s 1s/step - loss: 0.1352 - accuracy: 0.9585 - val_loss: 0.3275 -
val_accuracy: 0.8824
Epoch 4/7
58/58 [==============================] - 64s 1s/step - loss: 0.1105 - accuracy: 0.9651 - val_loss: 0.2293 -
val_accuracy: 0.8676
Epoch 5/7
58/58 [==============================] - 64s 1s/step - loss: 0.1347 - accuracy: 0.9525 - val_loss: 0.3580 -
val_accuracy: 0.8971
Epoch 6/7
58/58 [==============================] - 65s 1s/step - loss: 0.0987 - accuracy: 0.9694 - val_loss: 0.1438 -
val_accuracy: 0.9706
Epoch 7/7
58/58 [==============================] - 64s 1s/step - loss: 0.0748 - accuracy: 0.9760 - val_loss: 0.0875 -
val_accuracy: 0.9559
predictions=model.predict(test_dataset)
predictions=np.round(predictions)
predictions
array([[0.],
       [1.],
       [0.],
       [0.],
       [1.],
       [0.],
       [1.],
       [1.],
       [1.],
```

[0.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
[0.],
[1.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[1.],
[0.],

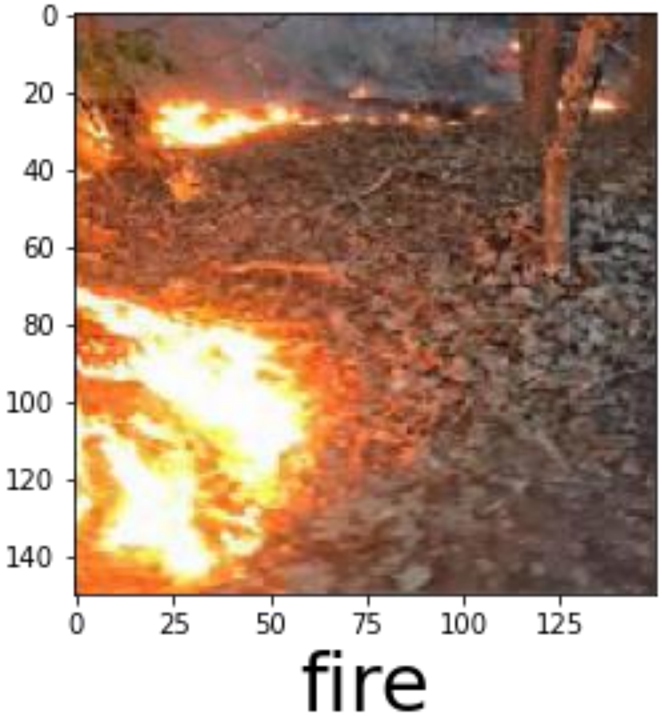
```
       [1.],
       [1.],
       [1.]], dtype=float32)
print(len(predictions))
68
import matplotlib.pyplot as plt
plt.plot(r.history['loss'],label='loss')
plt.plot(r.history['val_loss'],label='val_loss')
plt.legend()
```




```
plt.plot(r.history['accuracy'],label='acc')
plt.plot(r.history['val_accuracy'],label='val_acc')
plt.legend()
```




```
def predictImage(filename):
  img1=image.load_img(filename,target_size=(150,150))
  plt.imshow(img1)
  Y=image.img_to_array(img1)
  X=np.expand_dims(Y,axis=0)
  val=model.predict(X)
  print(val)
  if val==1:
    plt.xlabel("No fire",fontsize=30)
  elif val==0:
    plt.xlabel("fire",fontsize=30)
predictImage('/home/wsuser/work/forest_fire/Testing/fire/abc169.jpg')
```




```
model.save('forestfire.h5')
ls
forest_fire/  forestfire.h5
!tar -zcvf forest-fire-detection-model.tgz forestfire.h5
forestfire.h5
ls
forest_fire/  forest-fire-detection-model.tgz  forestfire.h5
import tensorflow as tf
tf.__version__
'2.7.2'
!pip install keras==2.2.4
Collecting keras==2.2.4
  Downloading Keras-2.2.4-py2.py3-none-any.whl (312 kB)
     |████████████████████████████████| 312 kB 15.1 MB/s eta 0:00:01
Requirement already satisfied: keras-preprocessing>=1.0.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packag
es (from keras==2.2.4) (1.1.2)
Collecting keras-applications>=1.0.6
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 17.3 MB/s eta 0:00:01
Requirement already satisfied: h5py in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from keras==2.2.4)
(3.2.1)
Requirement already satisfied: pyyaml in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from keras==2.2.
4) (5.4.1)
Requirement already satisfied: numpy>=1.9.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from kera
s==2.2.4) (1.20.3)
Requirement already satisfied: scipy>=0.14 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from keras=
=2.2.4) (1.7.3)
Requirement already satisfied: six>=1.9.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from keras==
2.2.4) (1.15.0)
Installing collected packages: keras-applications, keras
  Attempting uninstall: keras
```

```
    Found existing installation: keras 2.7.0
    Uninstalling keras-2.7.0:
      Successfully uninstalled keras-2.7.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This beha
viour is the source of the following dependency conflicts.
tensorflow 2.7.2 requires keras<2.8,>=2.7.0, but you have keras 2.2.4 which is incompatible.
Successfully installed keras-2.2.4 keras-applications-1.0.8
```

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

```
Collecting watson-machine-learning-client
  Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
     |████████████████████████████████| 538 kB 9.1 MB/s eta 0:00:01
Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machi
ne-learning-client) (4.62.3)
Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-mac
hine-learning-client) (1.3.4)
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-machi
ne-learning-client) (1.18.21)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-ma
chine-learning-client) (2.26.0)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-ma
chine-learning-client) (0.8.9)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-mac
hine-learning-client) (0.3.3)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-mach
ine-learning-client) (2022.9.24)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-mac
hine-learning-client) (1.26.7)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (f
rom boto3->watson-machine-learning-client) (0.10.0)
Requirement already satisfied: botocore<1.22.0,>=1.21.21 in /opt/conda/envs/Python-3.9/lib/python3.9/site-package
s (from boto3->watson-machine-learning-client) (1.21.41)
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: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packag
es (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-dat
eutil<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-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages
(from ibm-cos-sdk->watson-machine-learning-client) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-pac
kages (from ibm-cos-sdk->watson-machine-learning-client) (2.11.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from reque
sts->watson-machine-learning-client) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages
(from requests->watson-machine-learning-client) (2.0.4)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pand
as->watson-machine-learning-client) (2021.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pan
das->watson-machine-learning-client) (1.20.3)
Installing collected packages: watson-machine-learning-client
Successfully installed watson-machine-learning-client-1.0.391
```

In [26]:

```
from ibm_watson_machine_learning import APIClient
wml_credentials={
```

```
    "url":"https://us-south.ml.cloud.ibm.com",
    "apikey":"QRrumqchYvBpt8xjCMkXRAiDsy5b-ovLj4u35QFGp8GI"
}
client=APIClient(wml_credentials)
```

In [27]:

```
client
```

Out[27]:

```
<ibm_watson_machine_learning.client.APIClient at 0x7f491c5f3c10>
```

In [28]:

```
def guid_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'])
```

In [29]:

```
space_uid=guid_space_name(client,'forestfire')
print("Space UID"+space_uid)
Space UIDa1f920d4-7548-4cfe-888a-def2c2f4ccd7
```

In [30]:

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

Out[30]:

```
'SUCCESS'
```

In [31]:

```
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_1.15-py3.6-ddl      111e41b3-de2d-5422-a4d6-bf776828c4b7  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
```

```
xgboost_0.82-py3.6            39e31acd-5f30-41dc-ae44-60233c80306e  base
pytorch-onnx_1.2-py3.6-edt    40589d0e-7019-4e28-8daa-fb03b6f4fe12  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
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
spark-mllib_2.3-r_3.6         6586b9e3-ccd6-4f92-900f-0f8cb2bd6f0c  base
tensorflow_2.4-py3.7          65e171d7-72d1-55d9-8ebb-f813d620c9bb  base
spss-modeler_18.2             687eddc9-028a-4117-b9dd-e57b36f1efa5  base
----------------------------  ------------------------------------  ----
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
```

In [32]:

```
software_space_uid=client.software_specifications.get_uid_by_name('tensorflow_rt22.1-py3.9')
```

In [33]:

```
software_space_uid
```

Out[33]:

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

In [34]:

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

In [35]:

```
model_id=client.repository.get_model_id(model_details)
```

In [36]:

```
model_id
```

Out[36]:

```
'03cfb991-0fde-4e4d-af57-7cd62c62672f'
```

In [37]:

```
client.repository.download(model_id,'forestfire.tar.gb')
Successfully saved model content to file: 'forestfire.tar.gb'
```

Out[37]:

```
'/home/wsuser/work/forestfire.tar.gb'
```

In [38]:

```
pwd
```

Out[38]:

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

In [39]:

```
ls
forest_fire/  forest-fire-detection-model.tgz  forestfire.h5  forestfire.tar.gb
```

GitHub link: **https://github.com/IBM-EPBL/IBM-Project-20715-1659761232**

Project demo link:
https://drive.google.com/file/d/1o5ncphfZwLUKe60opTf77rkHoa64Wk7F/view?usp=share_link