

DETECTION OF FIRE FROM VIDEO AND ACOUSTIC FIRE EXTINCTION

A Summer Project Report

Submitted by

NITHISSHKRISHNA KS

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE

AND MACHINE LEARNING

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE
LEARNING, RAJALAKSHMI ENGINEERING COLLEGE CHENNAI**

602 105

CONTENTS

CONTENTS	2
ACKNOWLEDGEMENT	4
ABSTRACT	5
INTRODUCTION	6
1.1 OBJECTIVE	7
1.2 MOTIVATION	8
1.3 PROBLEM STATEMENT	9
1.4 PROPOSED WORK	9
1.5 NOVELTY IN THE PROJECT	10
CHAPTER - 2 - SYSTEM DESIGN	12
2.1 ARCHITECTURE DIAGRAMS	12
2.1.1 FIRE EXTINGUISHION PREDICTION	12
2.1.2 FIRE DETECTION	13
2.2 METHODOLOGY	14
2.2.1 DATA COLLECTION PHASE	14
2.2.2 DATA PREPROCESSING PHASE	14
2.2.3 DEVELOPING PHASE	15
2.2.4 TESTING PHASE	15
2.2.5 DEPLOYMENT PHASE	16
CHAPTER - 3 - MACHINE LEARNING MODELS FOR PROPOSED WORK	17
3.1 ABOUT MACHINE LEARNING	17
3.2 TYPES OF MACHINE LEARNING	18
3.3 CLASSIFICATION ALGORITHMS	19
3.3.1 K-NEAREST NEIGHBOURS	20
3.3.2 SUPPORT VECTOR MACHINE	20
3.3.3 DECISION TREE CLASSIFIER	21
3.3.4 RANDOM FOREST CLASSIFIER	22
3.4 EVALUATION METRICS	23
3.4.1 ACCURACY	23
3.4.2 PRECISION	24

3.4.3 RECALL	24
3.4.4 F1 SCORE	25
3.4.5 CONFUSION MATRIX	25
CHAPTER - 4 - IMPLEMENTATION	27
4.1 DATA COLLECTION	27
4.1.1 FIRE DETECTION MODEL	27
4.1.2 ACOUSTIC FIRE EXTINGUISHER MODEL	29
4.2 DATA PREPROCESSING PHASE	30
4.2.1 FIRE DETECTION MODEL	30
4.2.2 ACOUSTIC FIRE EXTINGUISHER MODEL	32
4.3 DEVELOPING PHASE	33
4.3.1 FIRE DETECTION MODEL	33
4.3.1.1 CONVOLUTIONAL LAYER	34
4.3.1.2 MAX POOLING	35
4.3.1.3 FLATTENING	36
4.3.1.4 DENSE LAYER	37
4.3.1.5 RELU ACTIVATION LAYER	37
4.3.1.6 ADAM OPTIMISER	38
4.3.1.7 EARLY STOPPING	38
4.3.1.8 REDUCED LEARNING	39
4.3.1.9 EPOCHS	40
4.3.2 ACOUSTIC FIRE EXTINGUISHING MODEL	40
4.4 TESTING PHASE	41
4.4.1 FIRE DETECTION MODEL	41
4.4.2 ACOUSTIC FIRE EXTINGUISHING MODEL	41
4.5 DEPLOYMENT PHASE	43
4.5.1 FIRE DETECTION MODEL	43
4.5.2 ACOUSTIC FIRE EXTINGUISHING MODEL	44
CHAPTER - 5 - RESULTS AND DISCUSSIONS	45
RESULTS	45
CHAPTER - 6 - CONCLUSION	52
6.1 CONCLUSION	52
6.2 FUTURE WORK	53
CHAPTER - 7 - REFERENCES	54

REFERENCES

54

CHAPTER - 1 - INTRODUCTION

INTRODUCTION

Currently, fire is detected using devices like sensors, smoke detectors.

These devices have a lot of disadvantages.

Sensors:

- Uses radioactive materials
- Extremely sensitive – False alarms are common
- Batteries have to be charged often

Smoke Detectors :

- Circuit Malfunctions
- Extremely sensitive to dust and air particles - regular maintenance is needed.
- Need high current to function

Hence, a new method to detect fire without as many disadvantages is essential. Moreover, detecting fire while fire is still small, is essential. Our Machine Learning Model takes videos as input and alerts if it has detected fire. The model has been trained with about 4320 images, so it detects all kinds of

fire like forest fires and household fires, with very less probability of false positives.

Traditionally, fire is extinguished using water, or air. Both of these extinguishing methods require fire fighters to be in direct contact with the flames, which is usually life threatening.

Recently, there are studies on whether sound waves can be used to extinguish fire. Though it hasn't been used yet, experiments have been conducted to check what kind of sound waves can be used to extinguish different kinds of flames - flames caused by gasoline, kerosene etc. We use the results of one such experiment to train our model. The model predicts if the given fire can be extinguished with a certain sound wave, when we have certain features of the sound wave and the fire.

1.1 OBJECTIVE

- The main objective of this project is to develop a new fire detection technique which can be used in real time and to determine if it can be extinguished .
- To make the detection of fire more feasible and cost effective.

- A view of helping save lives and land from fire accidents that cause by early detection and using acoustics to extinguish fire.
- To make fire detection available to everyone.
- As cameras and video surveillances are common everywhere, this technique can be used extensively.
- To predict if fire can be extinguished using sound waves.
- To reduce the risks fire fighters face everyday in putting out fires.
- To use sound, which is a resource that is available freely to put out fires, instead of water or air.

1.2 MOTIVATION

Fire accidents take many lives every year. Innocent bystanders and fire fighters often fall victim to the accidents. The existing fire detection systems need to be improved to save lives.

Using machine learning to detect fire, and predict its extinguishing status is much easier, and can help save innumerable lives. It is also faster, and cost effective, since we can use surveillance cameras, which are present everywhere. Alternatively, we can also use satellite images which provide us with the aerial views as the input to our fire detection model.

Since we are now tech dependent, and have surveillance footage almost everywhere, our models can be cost effective and life saving.

1.3 PROBLEM STATEMENT

Today, fire is detected using devices that are not always reliable. A fire detection device that detects fire as accurately as possible, with very less false negatives is ideal. We can use machine learning to build a model that can predict fire correctly using surveillance videos as input.

To put out a fire, firefighters have to be in contact with fire. Experiments are being conducted to check if acoustics can be used to extinguish fire. Machine learning can be used to check if a fire can be extinguished with a sound wave, if the attributes of the fire and sound wave are given to the model as input.

1.4 PROPOSED WORK

- Accurate fire detection is essential today, to save lives and the environment.
- Machine learning can be used to predict fire as accurately as possible.
- When the Machine Learning Model is given a video as input, it detects if there's fire in the video.

- Sound waves can be used to extinguish fire.
- Machine learning can be used to check if a sound wave can extinguish the fire.
- When certain attributes of fire, like it's source and airflow, and certain attributes of the sound wave, like it's decibel level and frequency, is given, the model can predict if the fire would be extinguished or not .
- The model predicts if the fire would be extinguished or not.

1.5 NOVELTY IN THE PROJECT

Traditional fire prediction models require extensive set up, and electricity. Our model requires very less external input. The fire prediction models currently present, have a limited dataset, and give false positives to a lot of non-fire videos. We collected data, and trained our model using the data, making sure that we get no false positives and fire is detected correctly.

Currently, there is no model that takes the features of the sound wave, and the fire, to predict if the fire would be extinguished. Our model using various classification algorithms, chooses the best one, and then predicts whether the fire would be extinguished.

The current chapter dealt with what our models aim to do. The next chapter will deal with the machine learning concepts we will be using in our model.

CHAPTER - 2 - SYSTEM DESIGN

In the previous chapter, we dealt with the machine learning basics that we will use in our models. In this chapter, we see the design of our models.

2.1 ARCHITECTURE DIAGRAMS

2.1.1 FIRE EXTINGUISHION PREDICTION

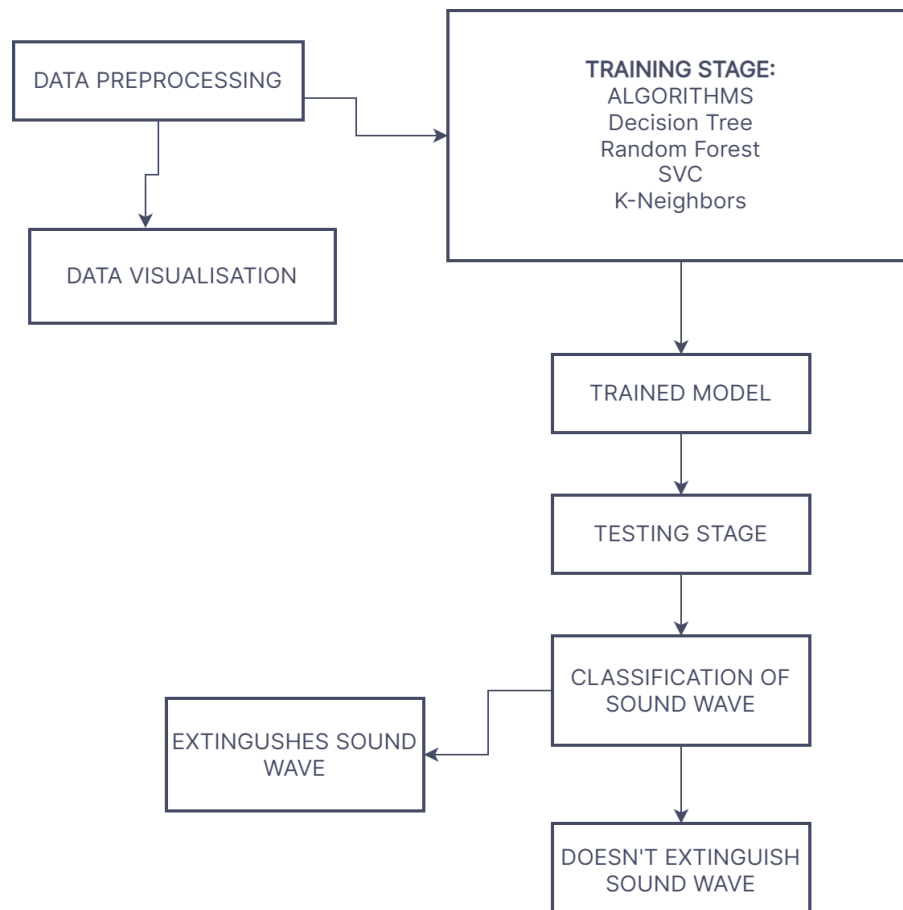


Fig 2.1- Architecture Diagram of Acoustic Fire Extinguisher

2.1.2 FIRE DETECTION

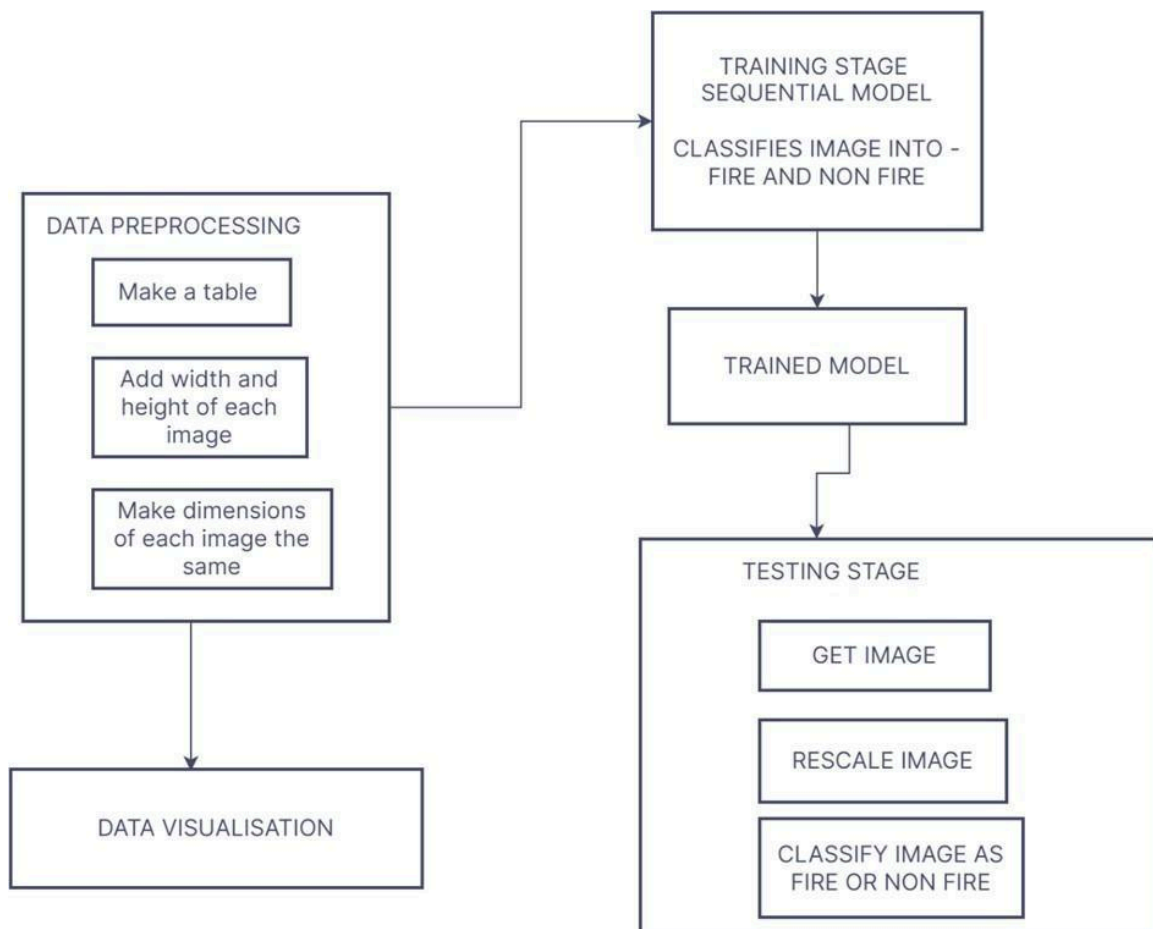


Fig 2.2 - Fire Detection using Video

2.2 METHODOLOGY

2.2.1 DATA COLLECTION PHASE

In the initial phase, we collected datasets to train our models on. We needed different datasets to cover different scenarios.

For the acoustic fire extinguisher model, we found a dataset that was the experimental results of experiments conducted.

For the fire detection, we made a dataset ourselves, and used a few datasets that we found for the model to be precise.

2.2.2 DATA PREPROCESSING PHASE

For fire detection, we went through the images to make sure the images were correctly labelled. We further made a new dataset containing the path and label. We added the height and width of the images to the dataset, resized the images and processed the image for the model to train on.

For fire extinguishion prediction, we analyse the data to make sure we have no null values.

2.2.3 DEVELOPING PHASE

In this phase we created models based on the dataset we collected. Two models were developed. One model is for detecting fire based on the input video. The other model is based on the features of fire and sound waves to predict it's extinction.

The first dataset consists of images : Fire and Non-Fire . The second consists of a table of attributes of sound like decibel, frequency, distance and attributed of fire like airflow, fuel, status and size. For training the model, the dataset is split into two sets for training and testing.

2.2.4 TESTING PHASE

In this stage, we tested our models with the testing dataset. The first model was tested using various fire and non-fire videos and false positives were reduced as much as possible. The second model was built using various classification algorithms. The algorithm which gave the highest accuracy score was used finally.

2.2.5 DEPLOYMENT PHASE

In the final phase, a web application was built using Flask to integrate our models with an user friendly UI. Both the models were connected with the website, so that the user can use both the models, without knowing about the inner details. The user can upload a video, to check if it has fire or non fire, and can also input various details of the fire and sound wave to be used to check if it can be extinguished or not.

In this chapter, we had a basic overview on how our model would be implemented. In the next chapter, we see the implementation of our model, in detail.

CHAPTER - 3 - MACHINE LEARNING MODELS FOR

PROPOSED WORK

In the previous chapter, we saw what we aim to do through this project, and why we chose this project.

In this chapter, we explain in detail about the machine learning concepts that we have used in our project.

3.1 ABOUT MACHINE LEARNING

Machine learning is a branch of artificial intelligence (AI) which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Machine learning enables systems to learn and improve from experience without being explicitly programmed. Machine learning focuses on developing computer programs that can access data and use it to learn for themselves.

ML has proven valuable because it can solve problems at a speed and scale that cannot be duplicated by the human mind alone. With massive amounts of computational ability behind a single task or multiple specific tasks, machines can be trained to identify patterns in and relationships between input data and automate routine processes.

Today, machine learning has become essential for solving problems across numerous areas, such as

- Computational finance (credit scoring, algorithmic trading)
- Computer vision (facial recognition, motion tracking, object detection)
- Computational biology (DNA sequencing, brain tumor detection)
- Automotive, aerospace, and manufacturing (predictive maintenance)
- Natural language processing (voice recognition)

3.2 TYPES OF MACHINE LEARNING

- **Supervised learning:** In this type of machine learning, algorithms are supplied with labelled training data and variables to be assessed for correlations by the algorithm is specified. Both the input and the output of the algorithm is specified.
- **Unsupervised learning:** This type of machine learning involves algorithms that train on unlabeled data. The algorithm scans through data sets looking for any meaningful connection. The data that algorithms train on as well as the predictions or recommendations they output are predetermined.
- **Semi-supervised learning:** This approach to machine learning involves a mix of the two preceding types. An algorithm may be fed

mostly labelled training data, but the model is free to explore the data on its own and develop its own understanding of the data set.

- **Reinforcement learning:** An algorithm is programmed to complete a task and positive or negative cues is given as it works out how to complete a task. But for the most part, the algorithm decides on its own what steps to take along the way.

3.3 CLASSIFICATION ALGORITHMS

Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups such as, Yes or No, 0 or 1 etc. Classes can be called as targets/labels or categories.

Classification algorithms takes labelled input data, that is, it contains input with the corresponding output.

We used the following classification algorithms :

- KNN
- SVM
- Decision Tree

- Random Forest

3.3.1 K-NEAREST NEIGHBOURS

K-Nearest Neighbour (KNN) algorithm predicts based on the specified number (k) of the nearest neighbouring data points. Here, the pre-processing of the data is significant as it impacts the distance measurements directly. Unlike others, the model does not have a mathematical formula, neither any descriptive ability.

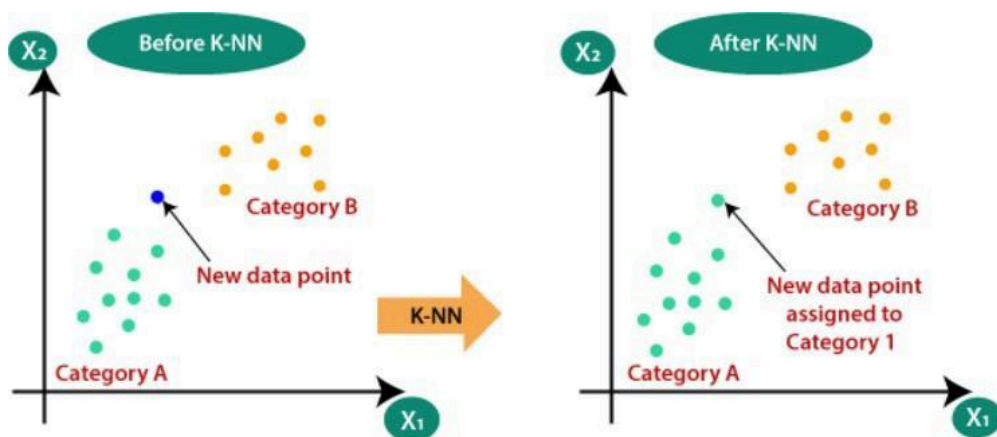


Fig 3.1 - KNN Neighbours

3.3.2 SUPPORT VECTOR MACHINE

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n -dimensional space into classes so that we can

easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane.

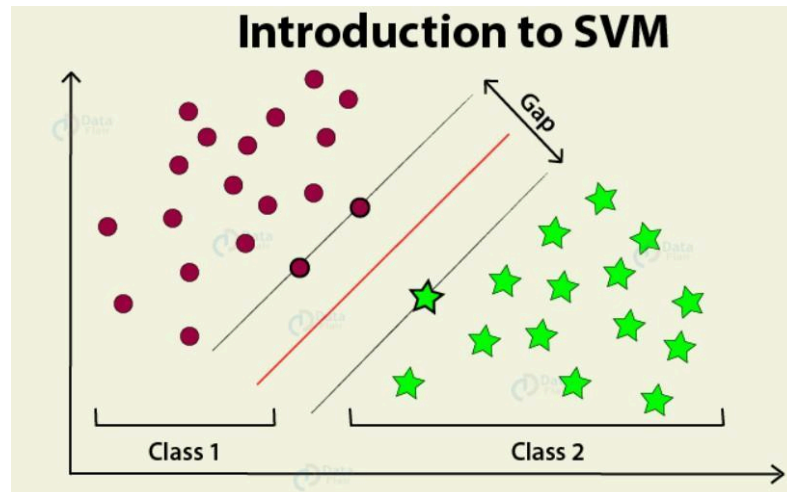


Fig 3.2 - SVM

3.3.3 DECISION TREE CLASSIFIER

Similar to a flow chart, decision tree divides data points into two similar groups at a time, starting with the "tree trunk" and moving through the "branches" and "leaves" until the categories are more closely related to one another.

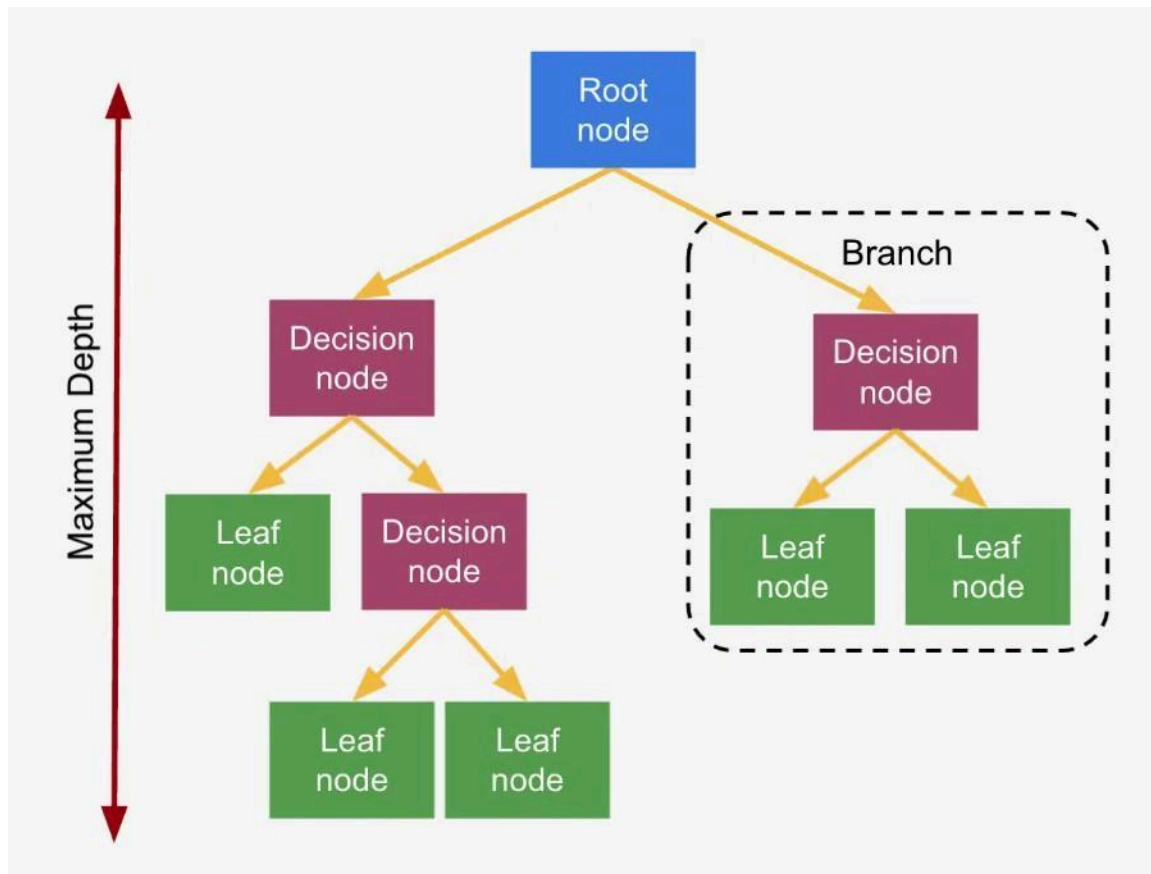


Fig 3.3 - Decision Tree

3.3.4 RANDOM FOREST CLASSIFIER

The random forest algorithm is an extension of the Decision Tree algorithm where we first create a number of decision trees using training data and then fit our new data into one of the created 'tree' as a 'random forest'. It averages the data to connect it to the nearest tree data based on the data scale.

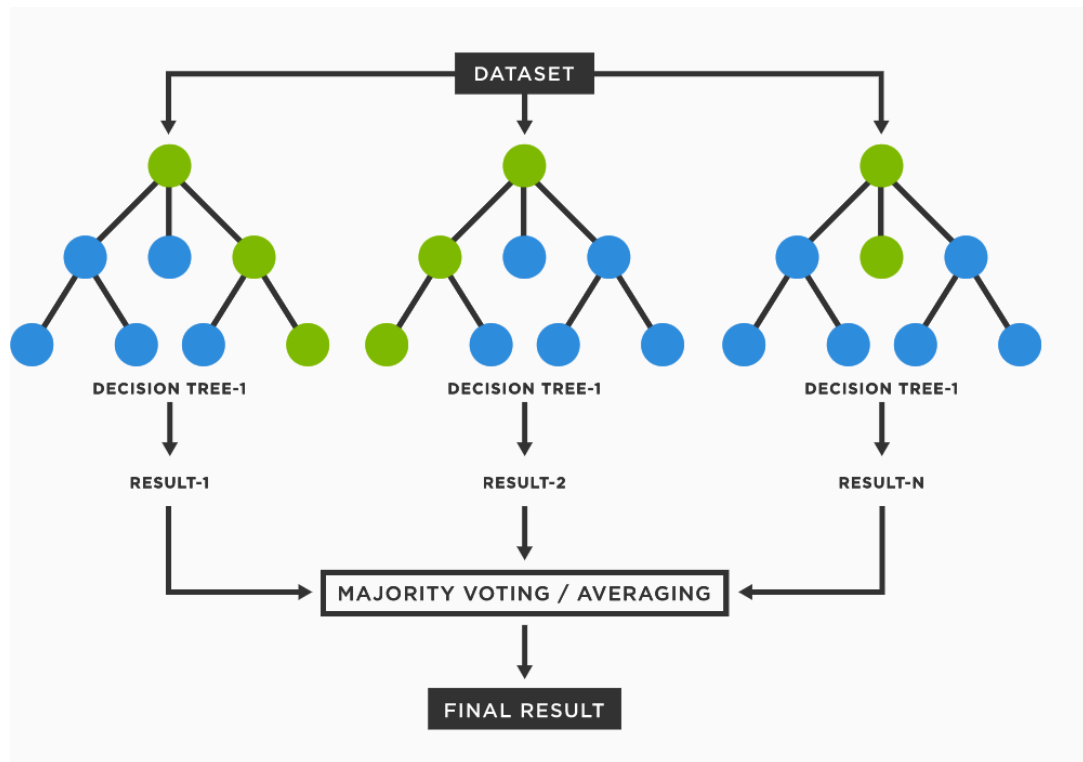


Fig.3.4 - Random Forest

3.4 EVALUATION METRICS

TP : True Positive

TN : True Negative

FP : False Positive

FN : False Negative

3.4.1 ACCURACY

Accuracy simply measures how often the classifier correctly predicts. We can define accuracy as the ratio of the number of correct predictions and the total number of predictions.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

3.4.2 PRECISION

Precision explains how many of the correctly predicted cases actually turned out to be positive. Precision is useful in the cases where False Positive is a higher concern than False Negatives.

$$\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive}+\text{FalsePositive}}$$

3.4.3 RECALL

Recall explains how many of the actual positive cases we were able to predict correctly with our model. It is a useful metric in cases where False Negative is of higher concern than False Positive. It is important in medical cases where it doesn't matter whether we raise a false alarm but the actual positive cases should not go undetected.

$$\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive}+\text{FalseNegative}}$$

3.4.4 F1 SCORE

It gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall. F1 Score is the harmonic mean of precision and recall.

$$F1 = 2 \frac{Precision \times Recall}{Precision + Recall}$$

3.4.5 CONFUSION MATRIX

Confusion Matrix is a performance measurement for the machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values. A confusion matrix is defined as the table that is often used to describe the performance of a classification model on a set of the test data for which the true values are known.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Fig 3.5 - Confusion Matrix

In this chapter, we saw about the machine learning concepts that we will be using in our model. In the subsequent chapter, we will see how our model is designed.

CHAPTER - 4 - IMPLEMENTATION

In the previous chapter, we had a basic overview on the implementation of our project. In this chapter, we see the implementation of our project in detail.

4.1 DATA COLLECTION

4.1.1 FIRE DETECTION MODEL

We used various datasets available in Kaggle and made a dataset ourselves. We used the following datasets :

- Fire Images^[4]
- Non- Fire Images^[5]
- Wildfire Dataset^[6]
- Weather Dataset^[7]
- Forest Fire Dataset^[8]

A few images from the datasets we have :

NON FIRE IMAGES :

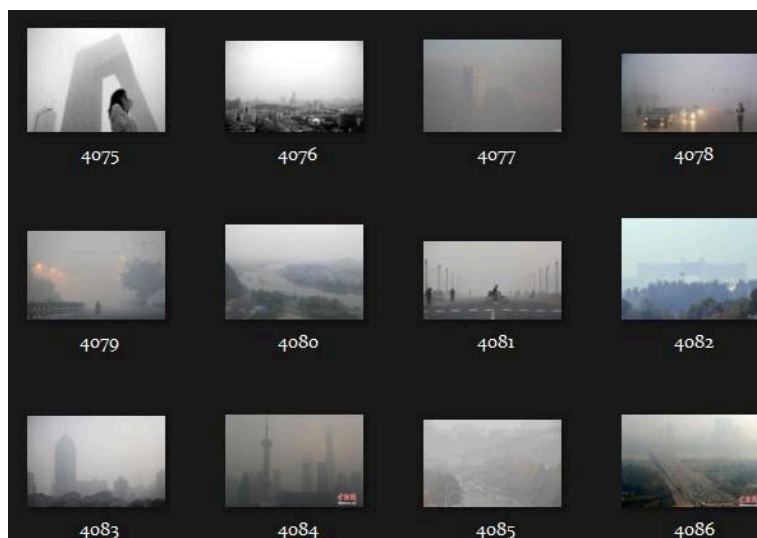
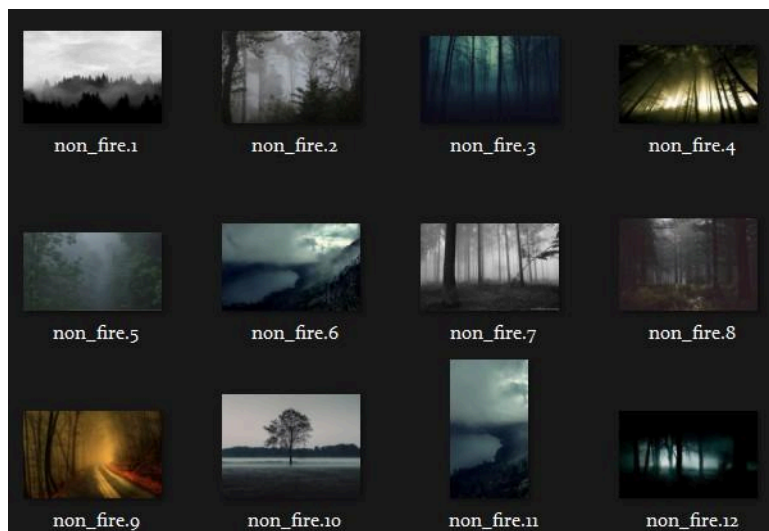
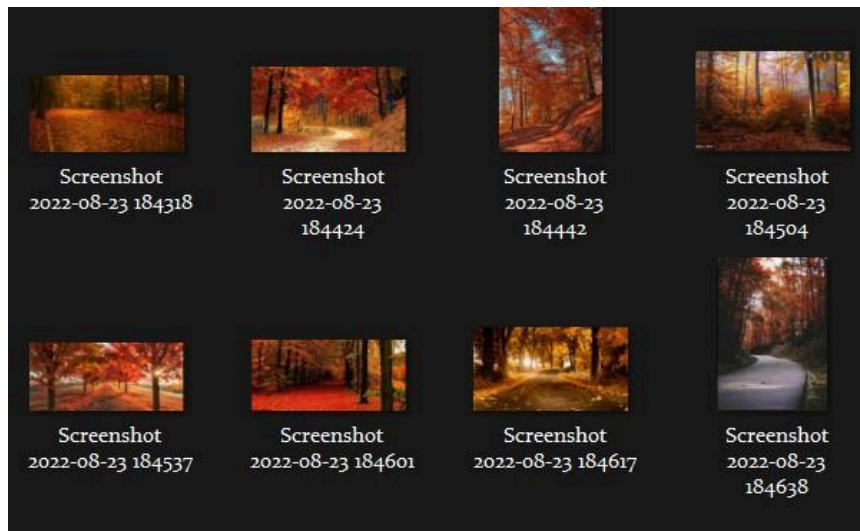


Fig 4.1 - Non Fire Images Dataset.

FIRE IMAGES :

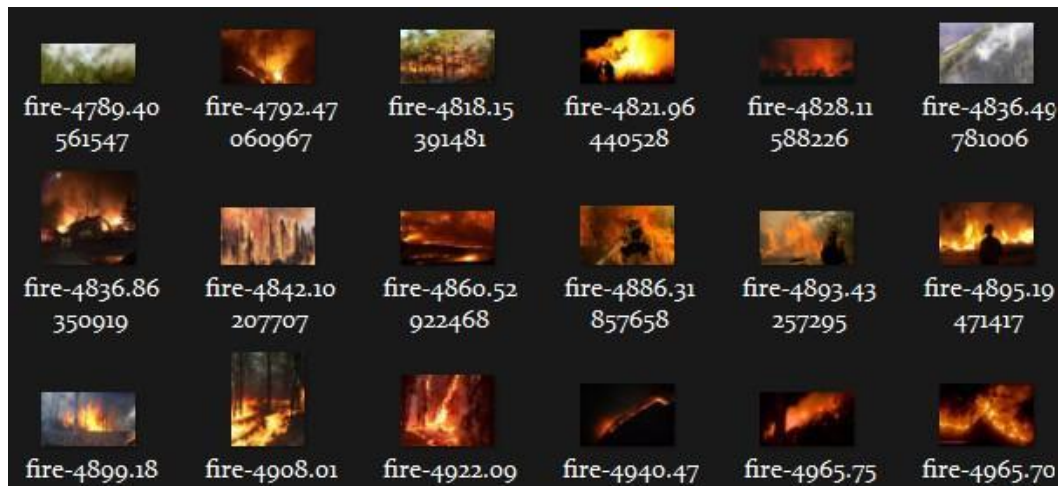
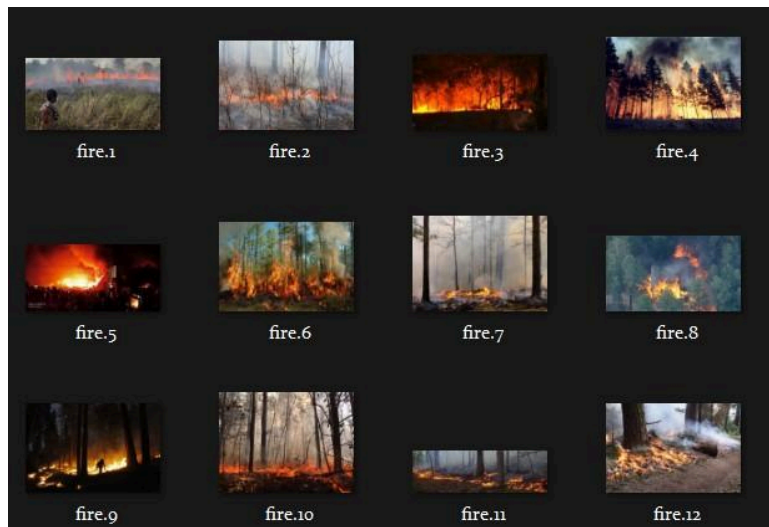


Fig 4.2 - Fire Images Dataset.

4.1.2 ACOUSTIC FIRE EXTINGUISHER MODEL

For our acoustic fire extinguisher model, we used the following dataset^[1] :

- Fire Extinguisher Dataset^[9]

Our Dataset:

SIZE	FUEL	DISTANCE	DESIBEL	AIRFLOW	FREQUENCY	STATUS
1	gasoline	10	96	0	75	0
1	gasoline	10	96	0	72	1
1	gasoline	10	96	2.6	70	1
1	gasoline	10	96	3.2	68	1
1	gasoline	10	109	4.5	67	1
1	gasoline	10	109	7.8	66	1
1	gasoline	10	103	9.7	65	1
1	gasoline	10	95	12	60	1
1	gasoline	10	102	13.3	55	1
1	gasoline	10	93	15.4	52	1
1	gasoline	10	93	15.1	51	1
1	gasoline	10	95	15.2	50	1
1	gasoline	10	110	15.4	48	1
1	gasoline	10	111	15.2	47	1
1	gasoline	10	109	15.4	46	1
1	gasoline	10	105	15.2	45	1
1	gasoline	10	111	16	44	1
1	gasoline	10	110	15.7	42	1
1	gasoline	10	106	15.4	40	1
1	gasoline	10	111	15.5	38	1
1	gasoline	10	110	15.2	36	1
1	gasoline	10	103	14.9	35	1
1	gasoline	10	109	14.9	34	1
1	gasoline	10	108	14.9	33	1

Fig 4.3 - Acoustic Fire Extinguisher Dataset.

4.2 DATA PREPROCESSING PHASE

4.2.1 FIRE DETECTION MODEL

We made a completely new table, containing the path and the label - labelling whether the image was fire or non-fire.

	Path	Label
0	C:\Users\charanya\Downloads\1\55.jpg	Fire
1	C:\Users\charanya\Downloads\archive (6)\datase...	Non_Fire
2	C:\Users\charanya\Desktop\fire_dataset\fire_im...	Fire
3	C:\Users\charanya\Desktop\fire_dataset\fire_im...	Fire
4	C:\Users\charanya\Downloads\archive (6)\datase...	Non_Fire
5	C:\Users\charanya\Desktop\fire_dataset\fire_im...	Fire
6	C:\Users\charanya\Downloads\archive (6)\datase...	Non_Fire
7	C:\Users\charanya\Desktop\fire_dataset\fire_im...	Fire
8	C:\Users\charanya\Downloads\Fire_Dataset-2\Mul...	Non_Fire
9	C:\Users\charanya\Downloads\archive (6)\datase...	Non_Fire

Fig 4.4 - Fire detection table creating using dataset

We added two columns - Height and Width of the images in the dataset^[3].

	Path	Label	Height	Width
0	C:\Users\charanya\Downloads\1\55.jpg	Fire	299	447
1	C:\Users\charanya\Downloads\archive (6)\datase...	Non_Fire	480	892
2	C:\Users\charanya\Desktop\fire_dataset\fire_im...	Fire	676	1024
3	C:\Users\charanya\Desktop\fire_dataset\fire_im...	Fire	1267	1900
4	C:\Users\charanya\Downloads\archive (6)\datase...	Non_Fire	240	424

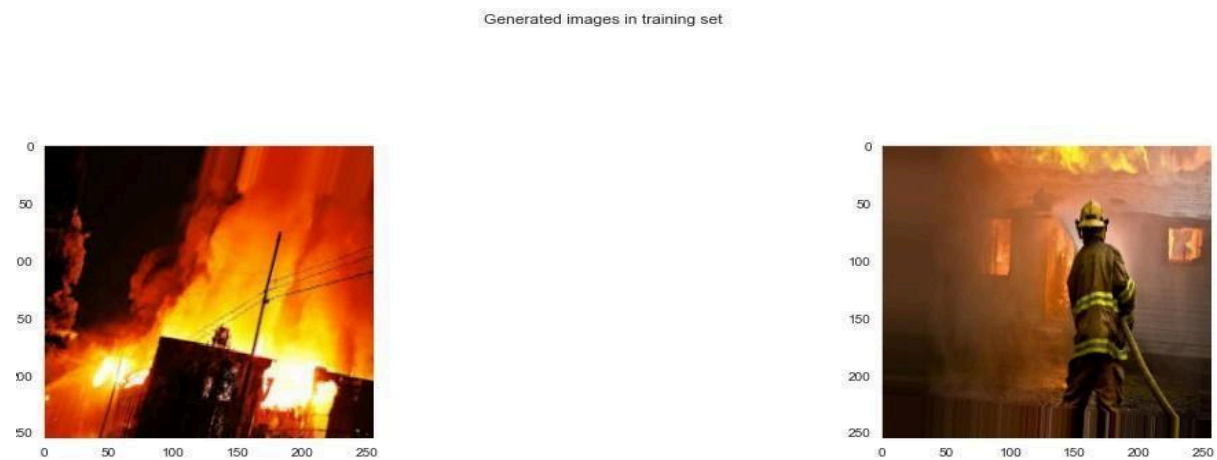
Fig 4.5 - Final Table

Since the images were of non-uniform size, all the images were made into uniform dimensions - 256 x 256

The images were then put in an ImageDataGenerator Function to be rotated, resized, zoomed randomly.

The images were also split into training set (80%) and testing set (20%) .

The input video was split into frames and stored in local folder. These frames were further resized to make them uniform.



```
Found 3507 validated image filenames belonging to 2 classes.
Found 876 validated image filenames belonging to 2 classes.
```

Fig 4.6 - Split to train and test dataset

4.2.2 ACOUSTIC FIRE EXTINGUISHER MODEL

It has no null values:

```
SIZE          0
FUEL          0
DISTANCE      0
DESIBEL       0
AIRFLOW       0
FREQUENCY     0
STATUS        0
dtype: int64
```

Fig 4.7 - No Null values

So no preprocessing was needed.

The dataset was split into training and testing dataset.

4.3 DEVELOPING PHASE

4.3.1 FIRE DETECTION MODEL

A convolutional neural network is developed to classify the images for correct labels. CNN is best to work with image data^[3].

- The most important building block of a Convolutional Neural Network (CNN) is the convolutional layer.
- Max pooling is done to help over-fitting.
- The output of the convolutional part of CNN is flattened using `flatten()` to create a single dimensional vector.
- The dense layers use a linear operation on the output of this part to classify.

In our model, we apply Convo2d and MaxPool thrice, to train the model. Initially, the input is 255x255x32 which becomes 31x31x128 after the 6 operations. We then flatten the input to get a single dimensional array of 123008 elements. We finally apply the dense operation thrice. We use Early Stopping and Reduced Learning callbacks when fitting the model. Adam Optimizer is used since it gives the best results while compiling the model.

4.3.1.1 CONVOLUTIONAL LAYER

The Convolutional Layer contains a set of filters (or kernels), parameters of which are to be learned throughout the training. The size of the filters is usually smaller than the actual image. Each filter convolves with the image and creates an activation map. For convolution the filter slid across the height and width of the image and the dot product between every element of the filter and the input is calculated at every spatial position. The first entry of the activation map is calculated by convolving the filter with the portion marked blue in the input image. The activation map is generated by repeating this process for every element of the input image. The output volume of the convolutional layer is generated by stacking the activation maps of every filter along the depth dimension.

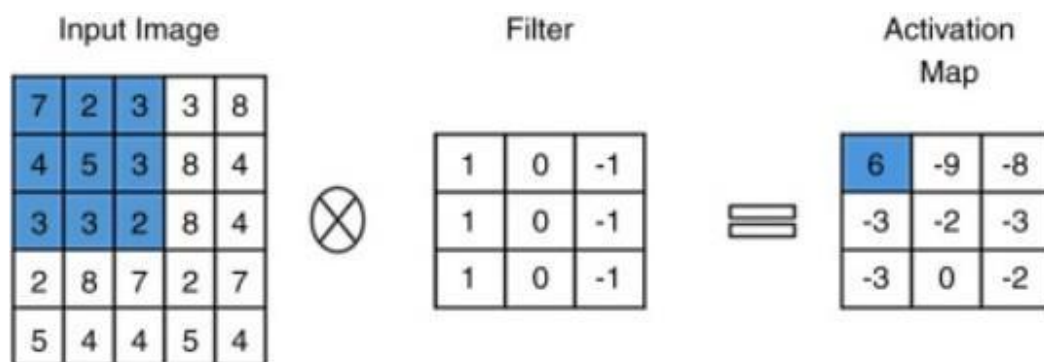


Fig 4.8 - Activation Map

4.3.1.2 MAX POOLING

The basic procedure of pooling is very similar to the convolution operation. We select a filter and slide it over the output feature map of the preceding convolutional layer. The most commonly used filter size is 2×2 and it is slid over the input using a stride of 2. Based on the type of pooling operation we've selected, the pooling filter calculates an output on the receptive field (the part of the feature map under the filter).

There are several approaches to pooling. The most commonly used approaches are max-pooling and average pooling. We use max pooling.

In max pooling, the filter simply selects the maximum pixel value in the receptive field. For example, if we have 4 pixels in the field with values 3, 9, 0, and 6, we select 9.

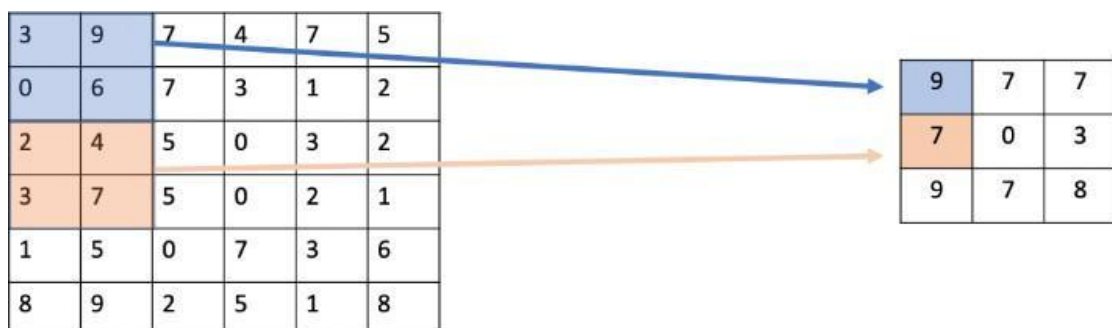


Fig 4.9 - Max Pooling

4.3.1.3 FLATTENING

Flattening involves taking the pooled feature map that is generated in the pooling step and transforming it into a one-dimensional vector. The reason why we transform the pooled feature map into a one-dimensional vector is because this vector will now be fed into an artificial neural network. Alternatively, this vector will now become the input layer of an artificial neural network that will be chained onto the convolutional neural network.

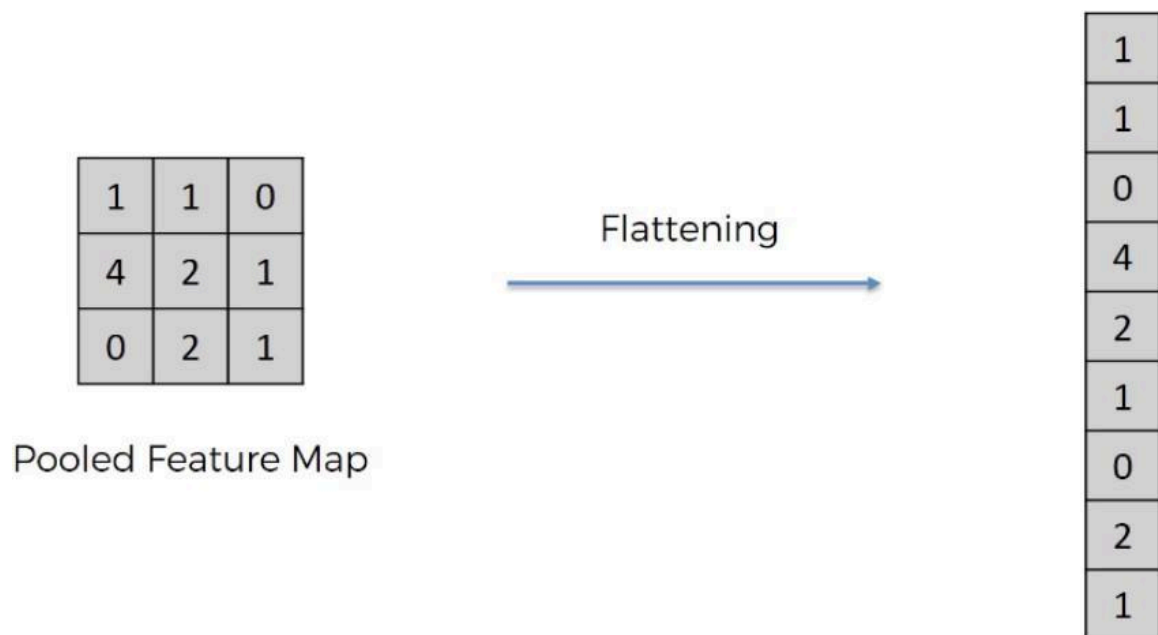


Fig 4.10 - Flattening to 1-D vector

4.3.1.4 DENSE LAYER

The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most commonly used layer in the models.

In the background, the dense layer performs a matrix-vector multiplication. The values used in the matrix are actually parameters that can be trained and updated with the help of backpropagation.

The output generated by the dense layer is an 'm' dimensional vector. Thus, dense layer is basically used for changing the dimensions of the vector. Dense layers also applies operations like rotation, scaling, translation on the vector.

4.3.1.5 RELU ACTIVATION LAYER

In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

4.3.1.6 ADAM OPTIMISER

Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

The results of the Adam optimizer are generally better than every other optimization algorithms, have faster computation time, and require fewer parameters for tuning . Adam is recommended as the default optimizer for most of the applications. It gives much higher performance.

4.3.1.7 EARLY STOPPING

Early Stopping is a regularisation technique for deep neural networks that stops training when parameter updates no longer begin to yield improvements on a validation set. Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. Early stopping is a

method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out validation dataset.

4.3.1.8 REDUCED LEARNING

A learning rate schedule is a predefined framework that adjusts the learning rate between epochs or iterations as the training progresses. It is a callback.

Callbacks are the special utilities or functions that are executed during training at given stages of the training procedure. Callbacks can help you prevent overfitting, visualize training progress, debug your code, save checkpoints, generate logs, etc.

`ReduceLROnPlateau`(Reduced learning rate) is a callback to reduce the learning rate when a metric has stopped improving. This callback monitors a quantity and if no improvement is seen for a patience number of epochs, the learning rate is reduced by factor value ($\text{new_lr} = \text{lr} * \text{factor}$).

4.3.1.9 EPOCHS

One epoch refers to one complete pass of the training dataset through the algorithm. This epochs number is an important hyperparameter for the algorithm. It specifies the number of epochs or complete passes of the entire training dataset passing through the training or learning process of the algorithm.

An epoch is made up of one or more batches, where we use a part of the dataset to train the neural network. We call passing through the training examples in a batch an iteration.

4.3.2 ACOUSTIC FIRE EXTINGUISHING MODEL

For our fire extinguishing model, we used 4 classification models.

- Decision Tree Classifier
- K Nearest Neighbours
- Random Forest Classifier
- Support Vector Machine

4.4 TESTING PHASE

4.4.1 FIRE DETECTION MODEL

The metric plots for our model on the testing dataset is :

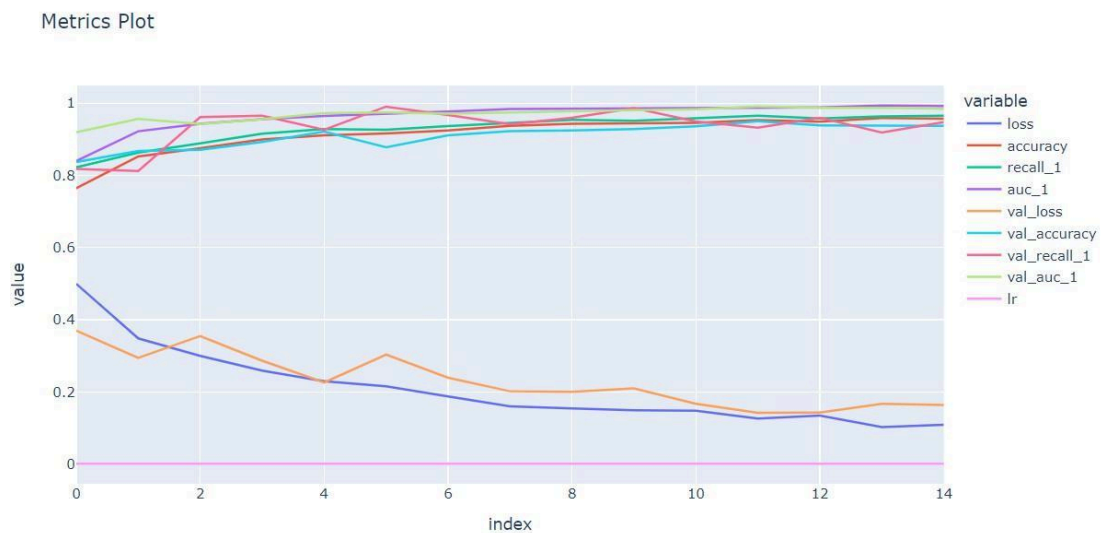


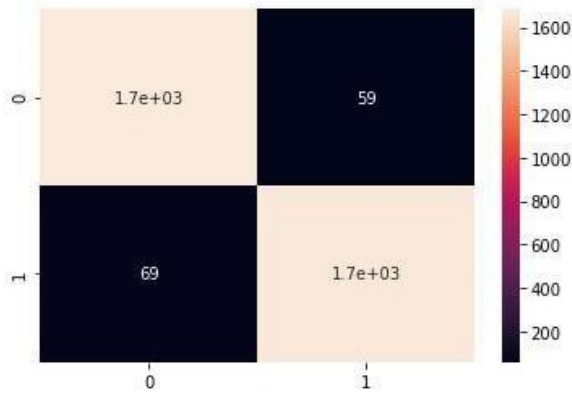
Fig 4.11- Metrics plot for fire detection

4.4.2 ACOUSTIC FIRE EXTINGUISHING MODEL

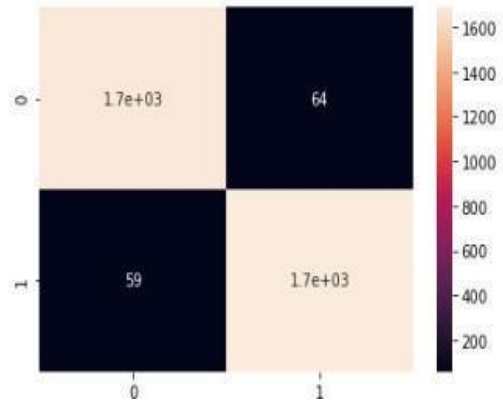
The accuracies of the different classifiers were computed, and the classifier with the best accuracy score was chosen as the final classification algorithm.

The different evaluation metrics were :

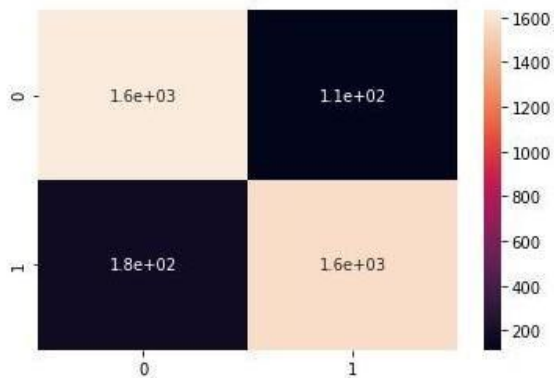
Model : RandomForestClassifier
 Train Accuracy : 100.00%
 Test Accuracy : 96.33%
 Accuracy Score: 0.96
 mean squared error: 0.04



Model : DecisionTreeClassifier
 Train Accuracy : 100.00%
 Test Accuracy : 96.47%
 Accuracy Score: 0.96
 mean squared error: 0.04



Model : KNeighborsClassifier
 Train Accuracy : 94.70%
 Test Accuracy : 91.75%
 Accuracy Score: 0.92
 mean squared error: 0.08



Model : SVC
 Train Accuracy : 89.01%
 Test Accuracy : 89.08%
 Accuracy Score: 0.89
 mean squared error: 0.11

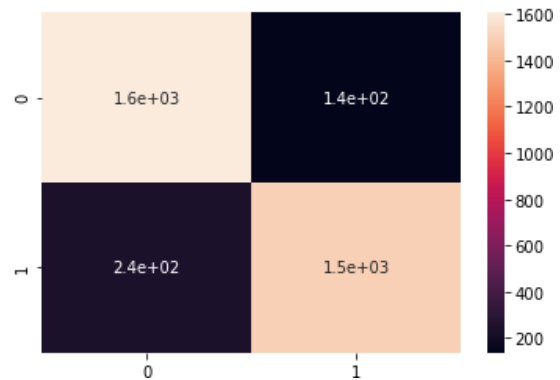


Fig 4.12 - Evaluation metrics of various models

A graph was plotted showing the different accuracy scores:

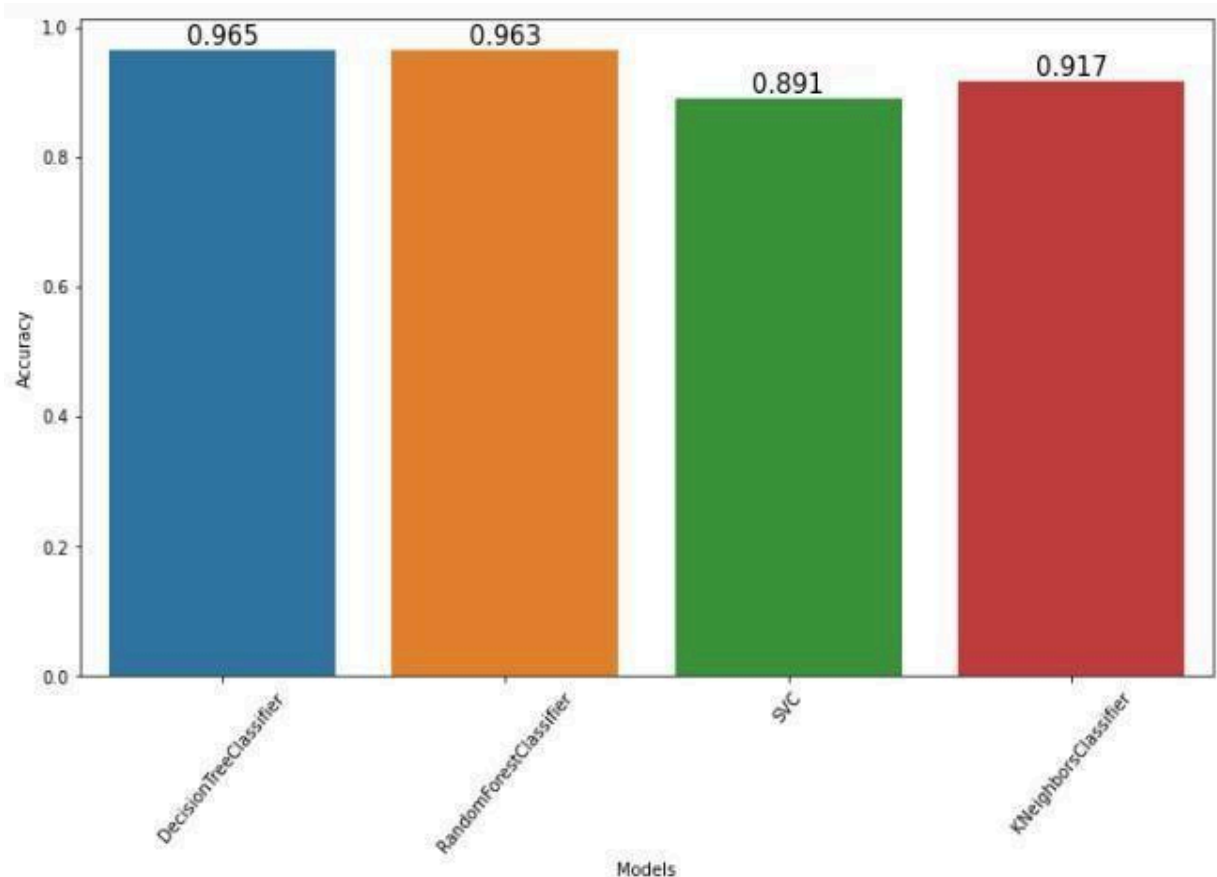


Fig 4.13 - Graph plotted using accuracy

We chose Decision Tree Classifier as it had the most accuracy score.

4.5 DEPLOYMENT PHASE

4.5.1 FIRE DETECTION MODEL

The user inputs a video and the model detects fire in the video, if fire is present in it.

The video is initially stored in local storage and is then split into frames^[2]. Each frame is then passed through the model. If the model predicts there's a fire, no other frame is checked, and FIRE is the output.

If no fire is detected and all the frames in the video is checked, NON-FIRE is printed.

In both cases, the video is shown for user's convenience.

4.5.2 ACOUSTIC FIRE EXTINGUISHING MODEL

The user inputs the various attributes needed by the model to predict if the fire would be extinguished. The result is printed along with the attributes that the user entered for their convenience.

In this chapter, we saw how our project was implemented, in detail. In the next chapter, we will see the final webpage with the working model.

CHAPTER - 5 - RESULTS AND DISCUSSIONS

RESULTS

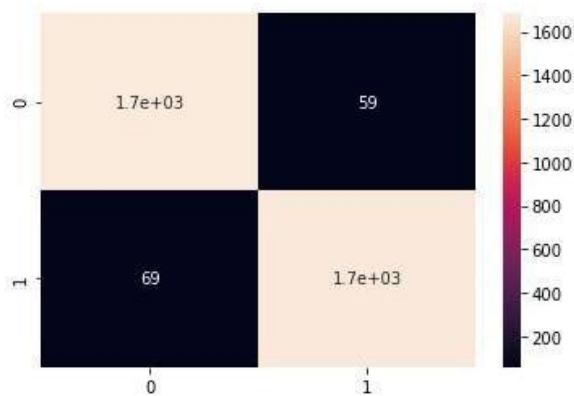
In the previous chapter, we saw how our project was implemented. In this chapter, we will see how our model works in a real time environment.

Acoustic Fire Extinguisher

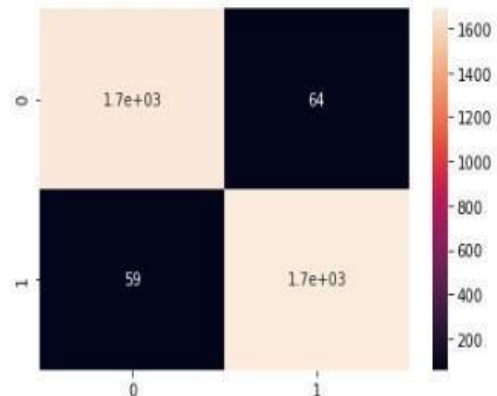
The evaluation metrics we used were Accuracy, Precision, Recall and F1 score.

The accuracies and Confusion Matrices of each classification algorithm is given below :

Model : RandomForestClassifier
 Train Accuracy : 100.00%
 Test Accuracy : 96.33%
 Accuracy Score: 0.96
 mean squared error: 0.04



Model : DecisionTreeClassifier
 Train Accuracy : 100.00%
 Test Accuracy : 96.47%
 Accuracy Score: 0.96
 mean squared error: 0.04



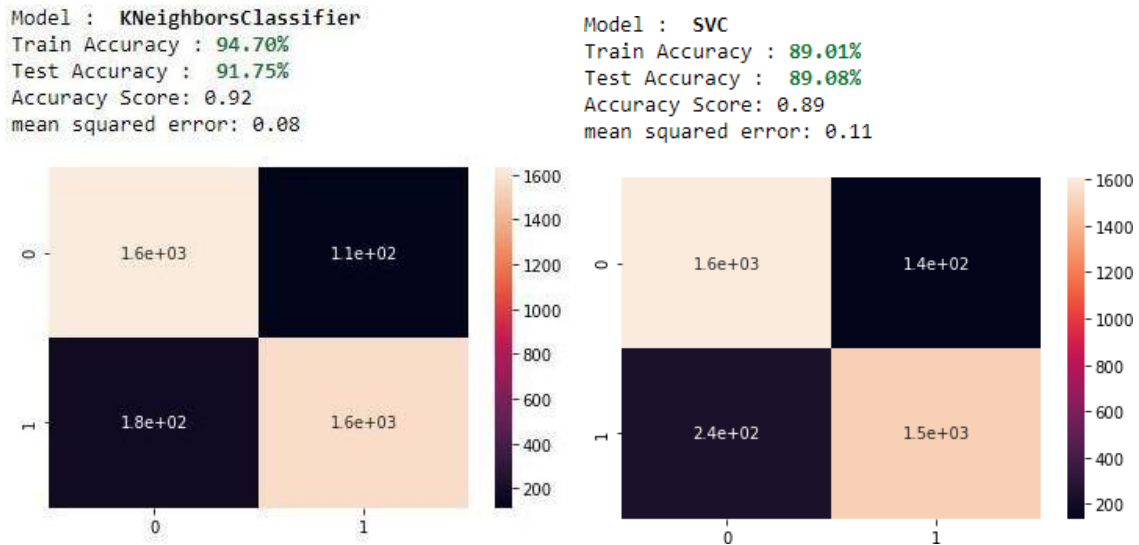


Fig 5.1- Confusion matrix

Out of Decision Tree Classifier (96.5%) , SVC (89.1%) , Random Forest Classifier (96.3%) and KNNeighbors Classifier (91.7%), Decision Tree Classifier also Provided the Highest Accuracy. So Decision Tree Classifier was chosen as the final model.

Fire Detection Using Video

Model built using the CNN layers. Recall, Early stopping, Epoch, reduce_lr_on_plateau was also used. We obtained the following results:

```
loss: 0.15
accuracy: 0.94
recall_1: 0.95
auc_1: 0.99
```

The Accuracy was 94%.

Our web application has a separate page for each model. Initially, we see the home page, which has the basic information needed by the user.

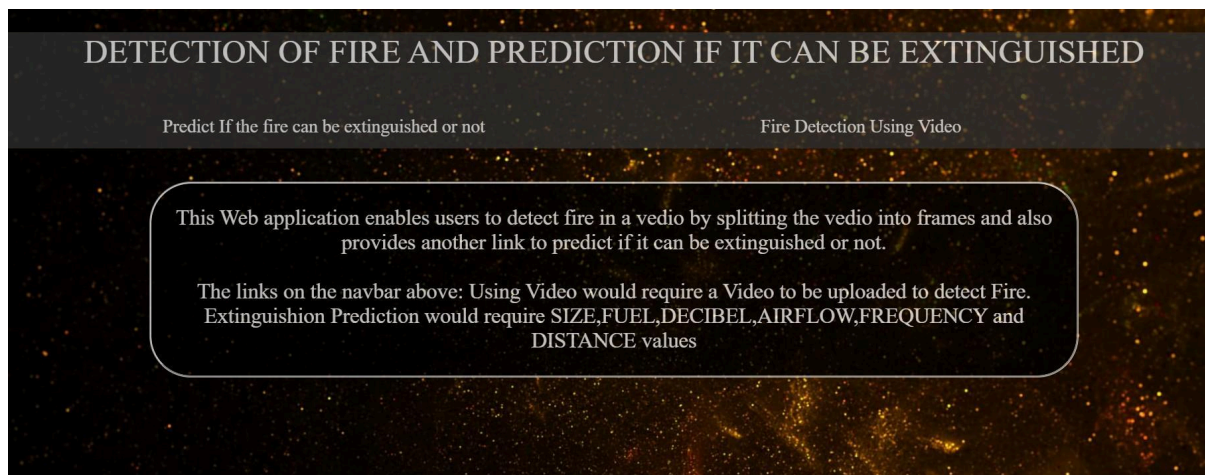
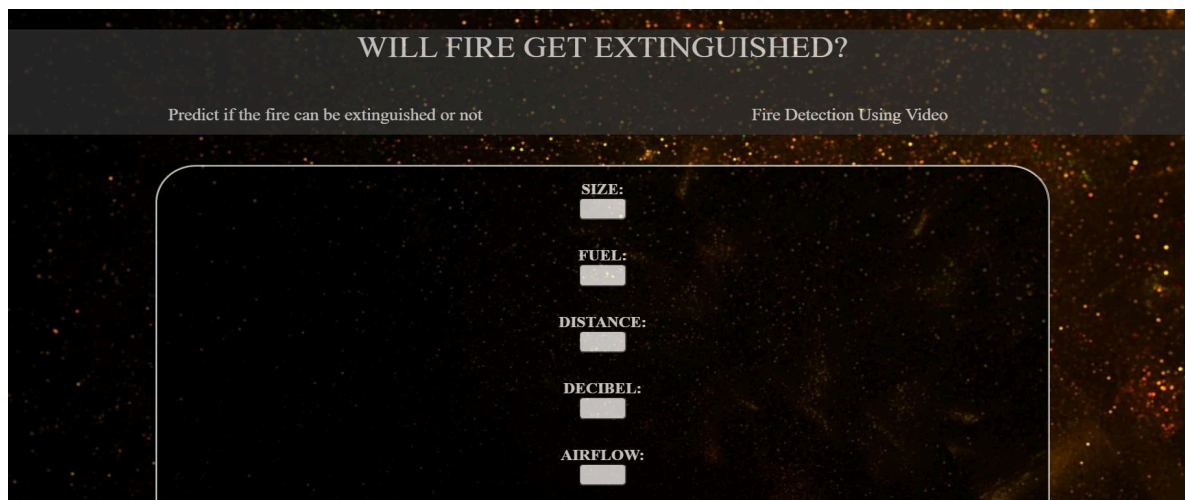
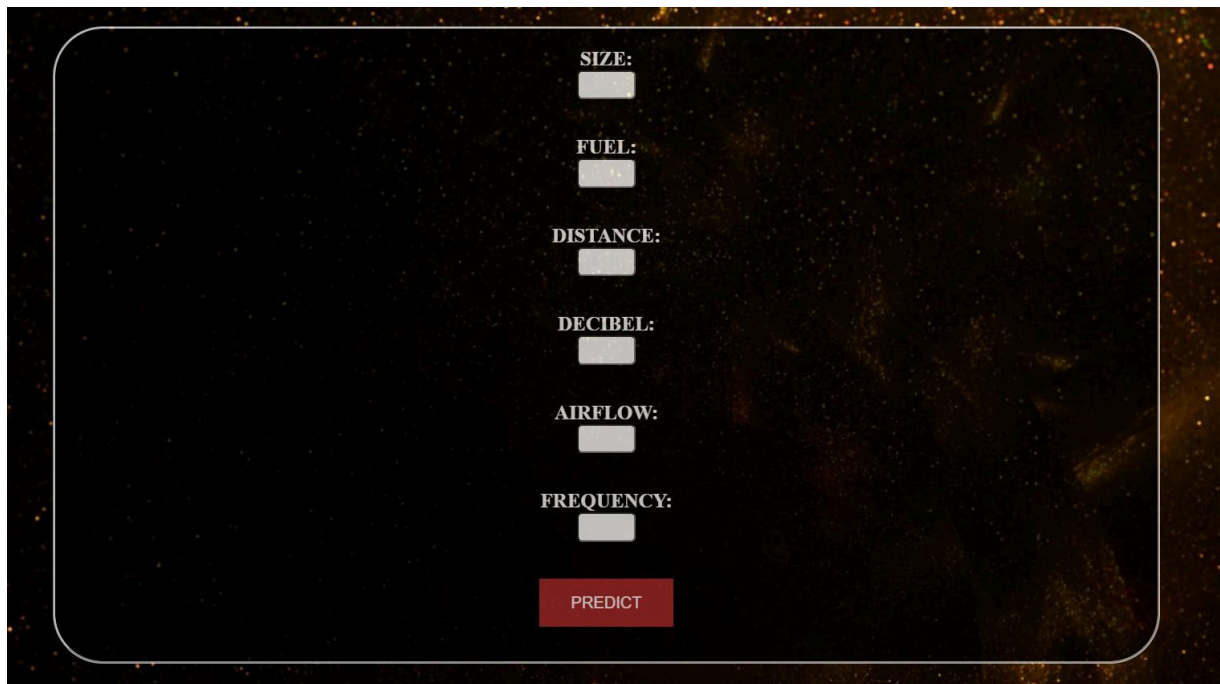


Fig 5.2 - Home page

When the user clicks on the first link to access the acoustic fire extinguishing model, it navigates them to the page below :





A screenshot of a web application interface for an acoustic fire extinguisher. The interface is set against a dark, starry background. It features a central white rounded rectangle containing six input fields, each with a label and a corresponding text input box. The labels are SIZE:, FUEL:, DISTANCE:, DECIBEL:, AIRFLOW:, and FREQUENCY:. Below these input fields is a red button with the text PREDICT in white capital letters.

Fig 5.3 - Acoustic Fire Extinguisher

Once the user enters the details and clicks predict, we get the below output:

Incase fire is extinguished,



A screenshot of the web application's output screen. The background is dark and starry. At the top, the text DETECT FIRE is displayed in white. Below it, there are two smaller lines of text: Predict If the fire can be extinguished or not on the left and Fire Detection Using Vedio on the right. In the center, a white rounded rectangle contains the output text. The text is as follows:

Extinguished!
Size: 1
Fuel: 0
Distance: 10
Decibel : 75
Airflow: 0.0
Frequency: 78

Fig 5.4.a - Acoustic Fire Extinguisher Output

Using the values entered, the predict function predicts if the fire can be extinguished or not. For the above given values, the fire can be extinguished so it gives the output as extinguished.

If fire is not extinguished,



Fig 5.4.b - Acoustic Fire Extinguisher Output

When the user clicks on the second link to access the fire prediction model, it navigates them to the page below :

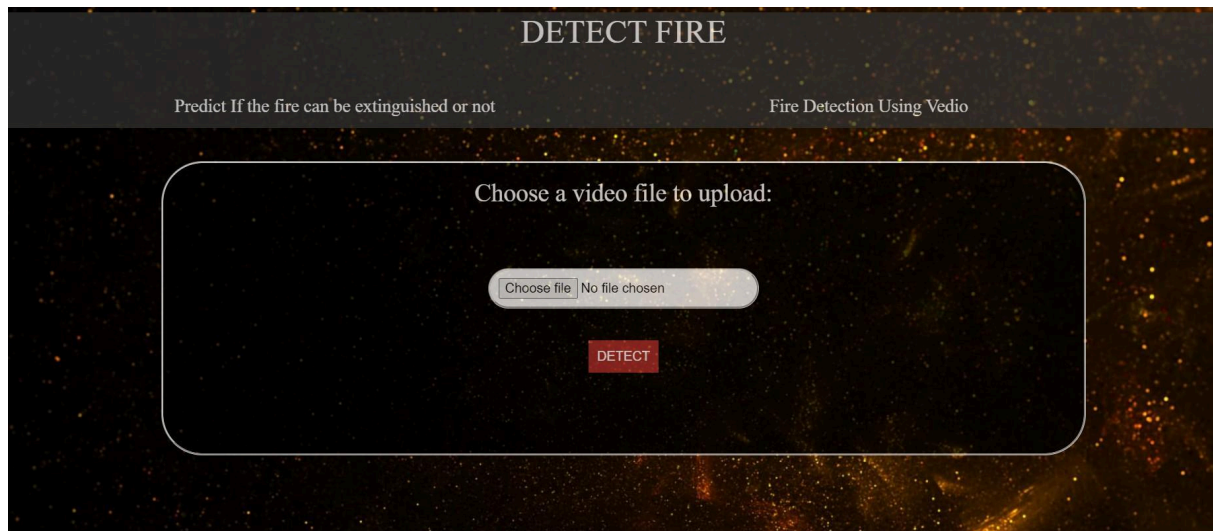


Fig 5.5 - Fire Detection Video Upload

Once the user selects a video and clicks detect, they get the below output:

If its non-fire:

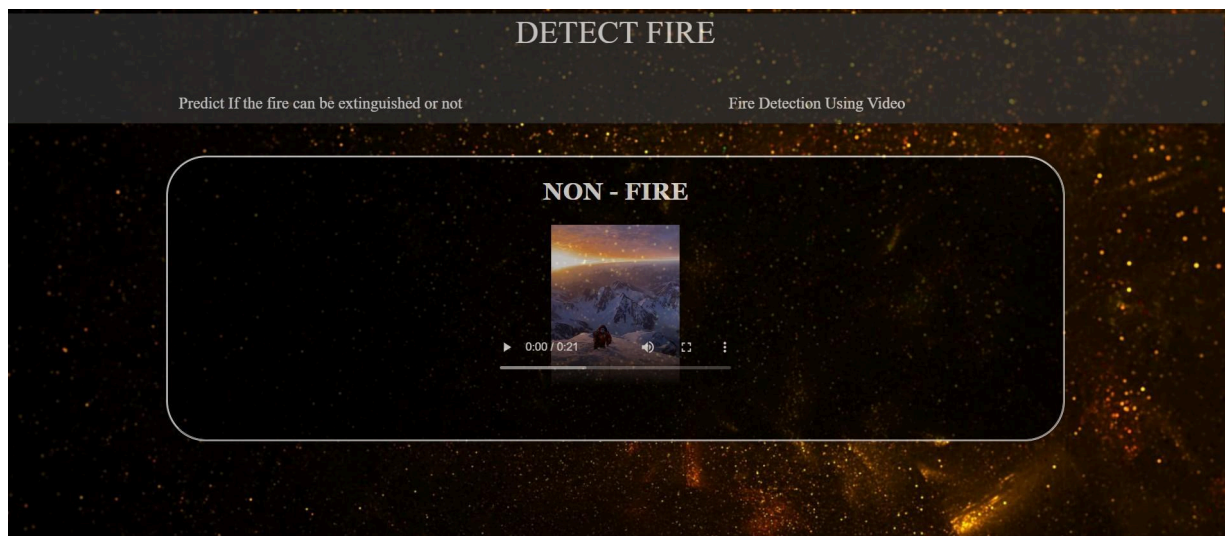


Fig 5.6.a - Output Of Fire Detection

If its fire:

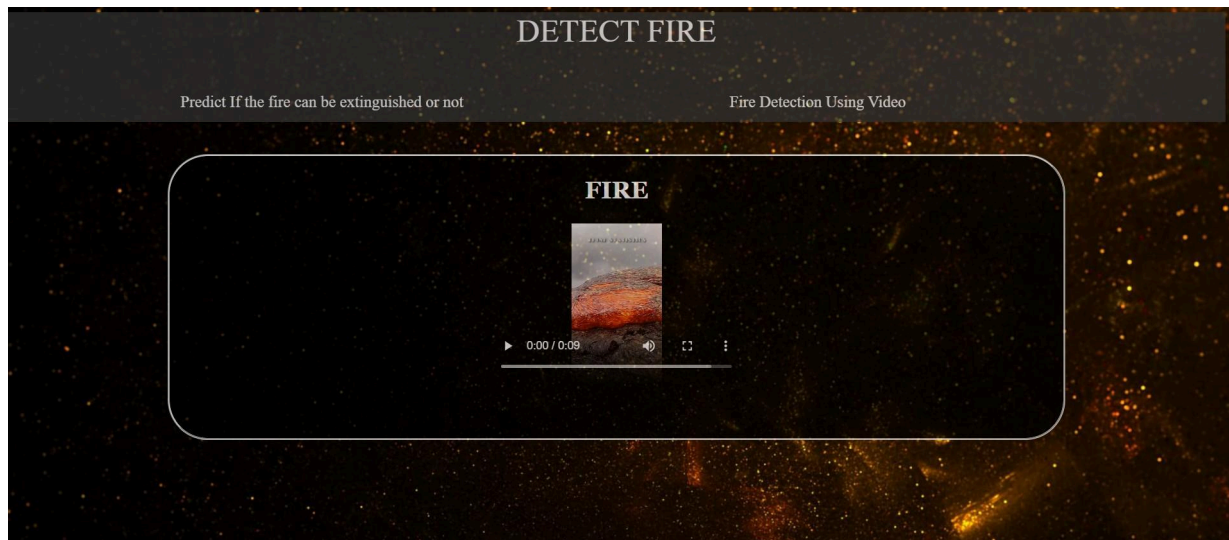


Fig 5.6.b - Output Of Fire Detection

Once the video is uploaded, the video is split into frames, and each frame has been checked for fire sequentially, when any one frame contains fire, it stops checking and the output is displayed along with the video. In the second case the first frame generated itself has fire and hence, the output is fire. In the first case, the video only has sunset, which is not been regarded as fire even though it has an orangish-red shade, then the full video is first split. These frames are then resized and fed for prediction. Since there is no frame with fire in it, we get Non Fire as output.

In this chapter, we see how our model works with the web application. In the next chapter, we see the conclusion and the scope for future work in our project

CHAPTER - 6 - CONCLUSION

In the preceding chapter, we saw how our model works in a real time environment. In this chapter, we will see the conclusion and the scope for future work in our project.

6.1 CONCLUSION

With this project, we have attempted to make a real world application that can help predict if fire can be extinguished without putting people's lives in danger, and to detect fire faster and more efficiently. We think both these models have immense importance with regard to the rising temperatures and global warming, to save lots of lives in the future.

Our models can detect fire with a high accuracy, and the probability of false positives is greatly reduced by accounting and training the models for all outliers. Our acoustic fire extinguisher model provides an easy platform to check if fire can be extinguished with a sound wave, if the attributes of the sound wave and fire are given as input.

We can improve our project, by segmenting fire in the video if fire is detected in it. Our acoustic model can be improved by taking input from the user as a sound wave instead of the attributes of the sound wave.

6.2 FUTURE WORK

The acoustic fire extinguishing model needs to have more real world data to be trained on. Instead of getting attributes from the user, the model will have to be trained on using sound waves, and extracting the attributes it needs, to make it more useful.

Our fire prediction model will need to have further work done with regard to segmenting the fire, so that its easier for the user to identify the fire.

CHAPTER - 7 - REFERENCES

REFERENCES

[1] Determining the Extinguishing Status of Fuel Flames With Sound Wave
by Machine Learning Methods

Date of publication June 11, 2021, date of current version June 22, 2021. Digital

Object Identifier 10.1109/ACCESS.2021.3088612

Authors: MURAT KOKLU AND YAVUZ SELIM TASPINAR

[IEEE Paper](#)

[2] Fire Detection using Deep Learning :

International Journal of Progressive Research in Science and Engineering

Volume-1, Issue-5, August-2020 www.ijprse.com

Authors: Suhas G , Chetan Kumar , Abhishek B S , Digvijay Gowda K A,

Prajwal R.

<https://www.journals.grdpublications.com/index.php/ijprse/article/download/14>

[1/138](#)

[3] Fire Detection on a Surveillance System using Image Processing

International Journal of Engineering Research & Technology

(IJERT) <http://www.ijert.org> ISSN: 2278-0181 IJERTV6IS050094 . Published

by : www.ijert.org Vol. 6 Issue 05, May - 2017

Authors: Prof. Amit Hatekar¹ , Saurabh Manwani² , Gaurav Patil³ , Akshat Parekh⁴ Department of Electronics and Telecommunication Thadomal Shahani Engineering College, Bandra, Mumbai

<https://www.ijert.org/research/fire-detection-on-a-surveillance-system-using-image-processing-IJERTV6IS050094.pdf>

APPENDIX

DATASETS USED

FIRE DETECTION MODEL

- [4] <https://www.kaggle.com/datasets/brsdincer/wildfire-detection-image-data>
- [5] <https://www.kaggle.com/datasets/pratik2901/multiclass-weather-dataset>
- [6] <https://www.kaggle.com/datasets/mohnishsaiprasad/forest-fire-images>
- [7] <https://www.kaggle.com/datasets/jehanbhathena/weather-dataset>
- [8] https://www.kaggle.com/datasets/kutaykutlu/forest-fire?select=train_fire

ACOUSTIC FIRE EXTINGUISHER

- [9] <https://www.kaggle.com/datasets/muratkokludataset/acoustic-extinguisher-fire-dataset>

ANNEXURE

A.1 CODE SNIPPETS

A.1.1 ACOUSTIC FIRE EXTINGUISHER

Checking for Null values and encoding string values

```
In [4]: data.isnull().sum()
```

```
In [4]: le = LabelEncoder()
data['FUEL'] = le.fit_transform(data['FUEL'])
```

```
In [5]: data.FUEL.value_counts()
```

Splitting Data , Fitting and Testing Models

```
In [10]: X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state = 64)
```

```
In [11]: models = [DecisionTreeClassifier, RandomForestClassifier, SVC, KNeighborsClassifier]
accuracy_test=[]
model = []
for m in models:
    model_name = type(m().__name__)
    print('\n\nModel : \033[01m {} \033[0m'.format(type(m).__name__))
    model_ = m()
    model_.fit(X_train, y_train)
    pred = model_.predict(X_test)
    pred_t = model_.predict(X_train)
    Acc_train = model_.score(X_train, y_train)
    Acc_test = model_.score(X_test, y_test)
    print ('Train Accuracy : \033[32m \033[01m{:.2f}% \033[30m \033[0m'.format(Acc_train*100))
    print ('Test Accuracy : \033[32m \033[01m{:.2f}% \033[30m \033[0m'.format(Acc_test*100))
    print('Accuracy Score: %.2f'% accuracy_score(y_test, pred))
    print('mean squared error: %.2f'% mean_squared_error(y_test, pred))
    cf_matrix = confusion_matrix(y_test, pred)
    sns.heatmap(cf_matrix, annot=True)
    plt.show()
    accuracy_test.append(Acc_test)
    model.append(model_name)
```

A.1.2 FIRE DETECTION

Making a new dataset from the different images in different folders


```

for dirname, _, filenames in os.walk("C:\\Users\\charanya\\Downloads\\Fire_Dataset-2\\Multi-class Weather Dataset\\Cloudy"):
    tmp2 = []
    # initialize list
    for filename in filenames:
        tmp2.append((pd.DataFrame([[os.path.join(dirname, filename), 'Non_Fire']]])))
nf3=pd.concat(tmp2)

for dirname, _, filenames in os.walk("C:\\Users\\charanya\\Downloads\\archive (6)\\dataset\\fogsmog"):
    tmp3 = []
    # initialize list
    for filename in filenames:
        tmp3.append((pd.DataFrame([[os.path.join(dirname, filename), 'Non_Fire']]])))
nf4=pd.concat(tmp3)

for dirname, _, filenames in os.walk("C:\\Users\\charanya\\Downloads\\archive (6)\\dataset\\sandstorm"):
    tmp4 = []
    # initialize list
    for filename in filenames:
        tmp4.append((pd.DataFrame([[os.path.join(dirname, filename), 'Non_Fire']]])))
nf5=pd.concat(tmp4)

for dirname, _, filenames in os.walk("C:\\Users\\charanya\\Desktop\\autumn"):
    tmp5 = []
    # initialize list
    for filename in filenames:
        tmp5.append((pd.DataFrame([[os.path.join(dirname, filename), 'Non_Fire']]])))
nf6=pd.concat(tmp5)

for dirname, _, filenames in os.walk("C:\\Users\\charanya\\Desktop\\autu"):
    tmp6 = []
    # initialize list
    for filename in filenames:
        tmp6.append((pd.DataFrame([[os.path.join(dirname, filename), 'Non_Fire']]])))
nf7=pd.concat(tmp6)

df = pd.concat([ff1,ff2,ff3,nf1,nf2,nf3,nf4,nf5,nf6,nf7])
df = df.rename(columns={0: 'Path', 1: 'Label'})

```

Adding height and width of each image to the dataset

```

In [9]: def shaper(row):
        shape = image.load_img(row['Path']).size
        row['Height'] = shape[1]
        row['Width'] = shape[0]
        return row
df = df.apply(shaper, axis=1)
df.head(5)

```

Preprocessing images in dataset

```

generator = ImageDataGenerator(
    rotation_range= 20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range = 2,
    zoom_range=0.2,
    rescale = 1/255,
    validation_split=0.2,
)

```

Splitting into two datasets for training and testing

```

In [15]: train = generator.flow_from_dataframe(df,x_col='Path',y_col='Label',images_size=(256,256),class_mode='binary',subset='training')
        val = generator.flow_from_dataframe(df,x_col='Path',y_col='Label',images_size=(256,256),class_mode='binary',subset='validation')

Found 3507 validated image filenames belonging to 2 classes.
Found 876 validated image filenames belonging to 2 classes.

```

Encoding Fire and Non Fire :

```
In [17]: class_indices = {}
         for key in train_gen.class_indices.keys():
             class_indices[train_gen.class_indices[key]] = key

         print(class_indices)

         {0: 'Fire', 1: 'Non_Fire'}
```

Building the model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense
```

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(2,2), activation='relu', input_shape=(256, 256, 3)))
model.add(MaxPool2D())
model.add(Conv2D(filters=64, kernel_size=(2,2), activation='relu'))
model.add(MaxPool2D())
model.add(Conv2D(filters=128, kernel_size=(2,2), activation='relu'))
model.add(MaxPool2D())
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Summary of model

```
In [20]: model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 255, 255, 32)	416
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 126, 126, 64)	8256
max_pooling2d_1 (MaxPooling2D)	(None, 63, 63, 64)	0
conv2d_2 (Conv2D)	(None, 62, 62, 128)	32896
max_pooling2d_2 (MaxPooling2D)	(None, 31, 31, 128)	0
flatten (Flatten)	(None, 123008)	0
dense (Dense)	(None, 64)	7872576
dense_1 (Dense)	(None, 32)	2080
dense_2 (Dense)	(None, 1)	33
=====		
Total params: 7,916,257		
Trainable params: 7,916,257		
Non-trainable params: 0		

Compiling and fitting model

```

In [21]: from tensorflow.keras.metrics import Recall,AUC
         from tensorflow.keras.utils import plot_model

In [22]: model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy',Recall(),AUC()])

In [23]: from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

In [24]: early_stopping = EarlyStopping(monitor='val_loss',patience=5,restore_best_weights=True)
         reduce_lr_on_plateau = ReduceLROnPlateau(monitor='val_loss',factor=0.1,patience=5)

In [56]: model.fit(x=train_gen,batch_size=32,epochs=15,validation_data=val_gen,callbacks=[early_stopping,reduce_lr_on_plateau])

```

Splitting Input Video into Frames And Detecting Fire in Each Frame

```

n="C:\\Users\\rsraj\\OneDrive\\Desktop\\Untitled Folder 6\\static\\uploads\\" + data
cam = cv2.VideoCapture(n)
try:
    if not os.path.exists('data'):
        os.makedirs('data')
        # if not created then raise error
except OSError:
    print ('Error: Creating directory of data')# frame
print("HI")
currentframe = 0
temp=0
cf=0
while(temp==0 and cf==0):
    ret,frame = cam.read()
    if ret:
        # if video is still left continue creating images
        name = './data/frame' + str(currentframe) + '.jpg'
        cv2.imwrite(name, frame)
        currentframe += 1
        if(currentframe>200):
            cf=1
            img = image.load_img(name)
            img = image.img_to_array(img)/255
            img = tf.image.resize(img,(256,256))
            img = tf.expand_dims(img,axis=0)
            prediction = int(tf.round(model12.predict(x=img)).numpy()[0][0])
            print(prediction)
            if(prediction==0):
                temp=1
                print(name)
                print("Fire!!")
                return name
                break
    else:
        break
if(temp==0):
    print("Non-Fire!")
return(temp)

```

A.1.3 FLASK CODE

```

UPLOAD_FOLDER = 'static/uploads/'
app = Flask(__name__, template_folder='template')
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('untitled.html')

@app.route('/index')
def index():
    return render_template('index.html')

@app.route('/index1')
def index1():
    return render_template('index1.html')

@app.route('/result', methods = ['POST'])
def upload_video():
    if request.method == 'POST':
        file = request.files['filename']
        if file:
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            result = ValuePredict(filename)
            print("PREDICTED")
            print('upload_video filename: ' + filename)
            print(result)
            a=""
            nam=""

```

```

        if result!= 0:
            a = 'FIRE'
            nam = result
        else:
            a="NON - FIRE "
        return render_template("result.html", prediction = a, filena=filename, pathna=nam)

```

```

@app.route('/predict',methods=['POST','GET'])
def predict():

    int_features = [x for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)
    xx=int_features
    print(xx)

    if prediction[0]==1:
        return render_template('res2.html',
                                prediction_text='Extinguished'.format(prediction),d1=xx[0],d2=xx[1],d3=xx[2],d4=xx[3],d5=xx[4],d6=xx[5]
                                )
    elif prediction[0]==0:
        return render_template('res2.html',
                                prediction_text='Not extinguished'.format(prediction),d1=xx[0],d2=xx[1],d3=xx[2],d4=xx[3],d5=xx[4],d6=xx[5]
                                )

app.run(debug=True, use_reloader=False)

```